Procedia Computer Science

Volume 29, 2014, Pages 220–230

ICCS 2014. 14th International Conference on Computational Science

FPGA-based acceleration of detecting statistical epistasis in GWAS

Lars Wienbrandt[1]*, Jan Christian Kässens[1], Jorge González-Domínguez[2], Bertil Schmidt[2], David Ellinghaus[3], and Manfred Schimmler[1]

[1] Department of Computer Science, Christian-Albrechts-University of Kiel, Germany
{lwi,jka,masch}@informatik.uni-kiel.de
[2] Institute of Computer Science, Johannes Gutenberg University Mainz, Germany
{j.gonzalez,bertil.schmidt}@uni-mainz.de
[3] Institute of Clinical Molecular Biology, Christian-Albrechts-University of Kiel, Germany
d.ellinghaus@ikmb.uni-kiel.de

Abstract

Genotype-by-genotype interactions (epistasis) are believed to be a significant source of unexplained genetic variation causing complex chronic diseases but have been ignored in genome-wide association studies (GWAS) due to the computational burden of analysis. In this work we show how to benefit from FPGA technology for highly parallel creation of contingency tables in a systolic chain with a subsequent statistical test. We present the implementation for the FPGA-based hardware platform RIVYERA S6-LX150 containing 128 Xilinx Spartan6-LX150 FPGAs. For performance evaluation we compare against the method iLOCi[9]. iLOCi claims to outperform other available tools in terms of accuracy. However, analysis of a dataset from the Wellcome Trust Case Control Consortium (WTCCC) with about 500,000 SNPs and 5,000 samples still takes about 19 hours on a MacPro workstation with two Intel Xeon quad-core CPUs, while our FPGA-based implementation requires only 4 minutes.

Keywords: GWAS, epistasis, pairwise gene-gene interaction, contingency tables, FPGA technology

1 Introduction

High-throughput genotyping technologies allow the collection of hundreds of thousands to a few million genetic markers, such as single nucleotide polymorphisms (SNPs), from individual DNA samples. In genome-wide association studies (GWAS) these genotypes are typically measured for several thousand individuals and then linked to a given phenotype of each individual, such as the presence (case) or absence (control) of an associated disease. In classical GWAS each genetic marker is analyzed separately in order to identify markers showing differences in

*corresponding author

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2014

doi:10.1016/j.procs.2014.05.020

genotype frequencies between cases and controls. Unfortunately, this approach is generally not powerful enough to model complex traits for which the detection of joint genetic effects (epistasis) needs to be considered [5, 7]. In (2-way) statistical epistasis each pair of measured markers is therefore tested in order to discover significant interactions that explain the given phenotype. Consequently, a number of algorithms have been developed to address the problem of detecting epistasis in recent years [1, 13, 17]. The main goal of these approaches is to find pairs of SNPs whose joint values show a statistically significant difference between cases and controls and thus they provide a list with these pairs that could explain a substantial proportion of genetic variation leading to disease.

Computing epistasis is highly time-consuming due to the large number of pairwise tests to be calculated; e.g. already for a moderately-sized dataset consisting of 500,000 SNPs there are about 125 billion pairwise interaction tests to be performed. Thus, many existing tools for calculating epistasis [6, 11, 14] require several days for processing moderately-sized datasets and several weeks to months for processing large-scale datasets on a standard CPU. Since both the availability and size of GWAS datasets are increasing rapidly, finding faster solutions is of high importance to research in this area. In this paper we address this problem by taking advantage of both fine-grained (by using of reconfigurable hardware (FPGAs)) and coarse-grained parallelism (using a number of FPGAs in parallel). Our parallel architecture is based on a systolic chain of processing elements for pairwise contingency table creation and a subsequent statistical test. Since contingency table creation is a common operation, our solution is easily adaptable to accelerate a large variety of epistasis tools by interchanging the statistical test implementation. In this paper we have chosen the test method of iLOCi [9] as a proof-of-concept. We show that this approach leads to an acceleration of between two and three orders of magnitude compared to the CPU-based approach.

2 Statistical Epistasis

2.1 Background

To perform large-scale epistasis studies, a lot of methods for pairwise interaction tests exist [13, 17], balancing between reducing runtimes and keeping the error rate low. CPU-based approaches, such as BOOST [14, 16], MDR [11], MB-MDR [4], iLOCi [9] etc., often result in long runtimes for exhaustive searches. Therefore prefiltering techniques may be applied to reduce the amount of SNP pairs, as in SIXPAC [10], SNPRuler [15], SNPHarvester [22], TEAM [24] and Screen and Clean [21]. Other methods take advantage from special architectures, such as GPUs, to perform an exhaustive analysis, e.g. GBOOST [23], SHEsisEpi [3], EpiGPU [2] and others.

However, most of these methods have one thing in common, they compute contingency tables (see Sect. 2.2) for each SNP pair before calculating a significance value. Our approach presented in this paper focuses on exhaustive analysis of SNP pairs and shows how FPGA technology can be applied to create and prepare contingency tables for significance tests concurrently on a large scale. We target the RIVYERA architecture with 128 FPGAs (see Sect. 3 for details) to significantly speedup pairwise interaction tests. RIVYERA is already successful in accelerating other bioinformatics applications, such as Smith-Waterman alignments or BLAST database searches [18, 19, 20] and hence, shows promise for this target as well.

Furthermore, the calculation of the significance value in our method is exchangeable. Thus, our solution may be applied to many existing methods performing an exhaustive analysis. Other approaches may not directly be adapted to our method, but prefiltering techniques, such

cases		SNP A		
		w	h	v
SNP B	w	n_{00}^{case}	n_{01}^{case}	n_{02}^{case}
	h	n_{10}^{case}	n_{11}^{case}	n_{12}^{case}
	v	n_{20}^{case}	n_{21}^{case}	n_{22}^{case}

controls		SNP A		
		w	h	v
SNP B	w	n_{00}^{ctrl}	n_{01}^{ctrl}	n_{02}^{ctrl}
	h	n_{10}^{ctrl}	n_{11}^{ctrl}	n_{12}^{ctrl}
	v	n_{20}^{ctrl}	n_{21}^{ctrl}	n_{22}^{ctrl}

Figure 1: Contingency tables for cases and controls. n_{ij}^{case} and n_{ij}^{ctrl} reflect the number of occurrences for the corresponding genotype combination in the current SNP pair.

as filtering of single SNPs, can trivially be implemented in the preprocessing phase. In our presentation we chose the iLOCi method [9] (see Sect. 2.3) as an example for an exhaustive search.

2.2 Contingency Tables

A typical GWAS dataset consists of two groups of samples (cases and controls) which are genotyped at a set of marker positions (SNPs). In this paper we consider biallelic markers for diploid organisms which is the common use case, i.e. genotypes may appear as homozygous wild (w), heterozygous (h) or homozygous variant (v) type. For pairwise interaction analysis a contingency table is created for each pair of SNPs separately for case and control group. Therefore, with n denoting the total number of SNPs, $n(n-1)/2$ tables have to be created (instead of n^2 due to the symmetry of SNP pairs). For moderate-size datasets, such as the Wellcome Trust Case Control Consortium (WTCCC) datasets [12] with about 500,000 SNPs, this implies about 125 billion tables. This huge amount of calculations is challenging for any computing system.

Contingency tables in pairwise interaction tests with biallelic markers have dimension 3×3, one entry for each possible combination of genotypes. The entries reflect the number of occurrences each combination of genotypes appears in the dataset for the corresponding SNP pair in either case and control group (see Fig. 1).

2.3 iLOCi

We have chosen iLOCi as an example application to test and compare against our implementation. iLOCi claims to outperform other tools, such as MDR [11] or BOOST [14], in terms of accuracy. However, according to the authors, analysis of a WTCCC dataset [12] with about 500,000 SNPs and 5,000 samples still takes about 19 hours on a MacPro workstation.

iLOCi computes a significance value p^{diff} for every possible pair of SNPs. This value is based on the previously created contingency tables as in Fig. 1 and is calculated as follows.

$$p^{\text{diff}} \quad = \quad \left| p^{\text{ctrl}} - p^{\text{case}} \right| \tag{1}$$

Let n_{ij}^{case} denote the genotype counts for the combination i and j of the current SNP pair in all case samples ($i, j \in \{0, 1, 2\}$ corresponding to the three possible genotypes). p_{ij}^{case} then denotes the relative probability $\frac{n_{ij}^{\text{case}}}{n^{\text{case}}}$ whereby n^{case} is the number of samples from the case group. Thus,

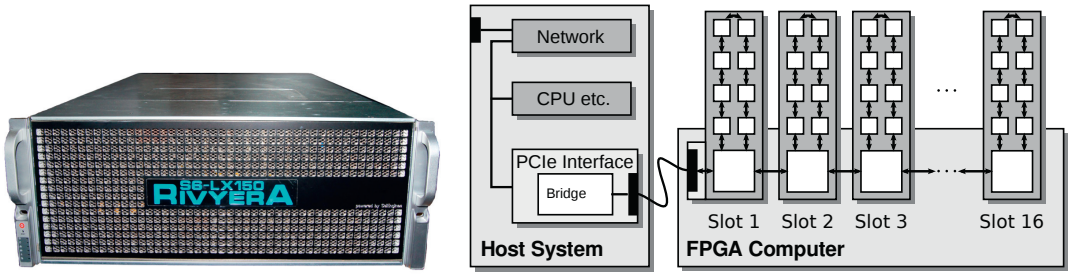

Figure 2: The RIVYERA S6-LX150 system.

p^{case} is calculated as follows.

$$p^{\text{case}} = \frac{p_{00}^{\text{case}} - p_{02}^{\text{case}} - p_{20}^{\text{case}} + p_{22}^{\text{case}}}{\sqrt{(p_{0\bullet}^{\text{case}} + p_{2\bullet}^{\text{case}})\,(p_{\bullet 0}^{\text{case}} + p_{\bullet 2}^{\text{case}})}} \tag{2}$$

$$= \frac{\frac{n_{00}^{\text{case}}}{n^{\text{case}}} - \frac{n_{02}^{\text{case}}}{n^{\text{case}}} - \frac{n_{20}^{\text{case}}}{n^{\text{case}}} + \frac{n_{22}^{\text{case}}}{n^{\text{case}}}}{\sqrt{\left(\sum_j \frac{n_{0j}^{\text{case}}}{n^{\text{case}}} + \sum_j \frac{n_{2j}^{\text{case}}}{n^{\text{case}}}\right)\left(\sum_i \frac{n_{i0}^{\text{case}}}{n^{\text{case}}} + \sum_i \frac{n_{i2}^{\text{case}}}{n^{\text{case}}}\right)}} \tag{3}$$

$$= \frac{n_{00}^{\text{case}} - n_{02}^{\text{case}} - n_{20}^{\text{case}} + n_{22}^{\text{case}}}{\sqrt{\left(\sum_j n_{0j}^{\text{case}} + \sum_j n_{2j}^{\text{case}}\right)\left(\sum_i n_{i0}^{\text{case}} + \sum_i n_{i2}^{\text{case}}\right)}} \tag{4}$$

The same applies analogously to the control samples for calculation of p^{ctrl}.

The higher the p^{diff} value the greater is the probability for an interaction [9]. Thus, unlike other algorithms, such as BOOST [14], which filter the results by a threshold, iLOCi saves the n best results (e.g. $n = 1000$) and presents the corresponding SNP pairs in a sorted list.

3 RIVYERA S6-LX150 Architecture

In 2008 the computing platform RIVYERA [8], originally developed for cryptanalysis, was introduced for problems related to bioinformatics. Here, the specific model RIVYERA S6-LX150 is presented.

The basic structure consists of two elements, a multiple-FPGA system and a server grade mainboard with standard PC components. The FPGA system consists of up to 16 FPGA modules with 8 Xilinx Spartan6-LX150 FPGAs each (upgrades allow up to 16 FPGAs on one module). Furthermore, each FPGA is connected to 256 MB DDR3-SDRAM. The mainboard is equipped with two Intel Xeon E5-2620 CPUs (6 cores @ 2GHz each) with 128 GB of RAM running a Linux OS. This system is later referred to as *host*.

The bus system implemented on the RIVYERA FPGA computer is organized as a systolic chain, i.e. every FPGA on an FPGA module is connected by fast point-to-point connections to each neighbor forming a ring. An additional member of this ring forms the communication controller. It provides the interconnection of each module to its neighboring modules and, on the first module, the uplink to the host via PCIe. An API hides control of the complete bus system and ensures transparency of the communication to the developer. Besides normal point-to-point transmissions, the API provides broadcast facilities and methods for configuring the FPGAs. A picture and the design structure of the RIVYERA S6-LX150 system is shown in Fig. 2.

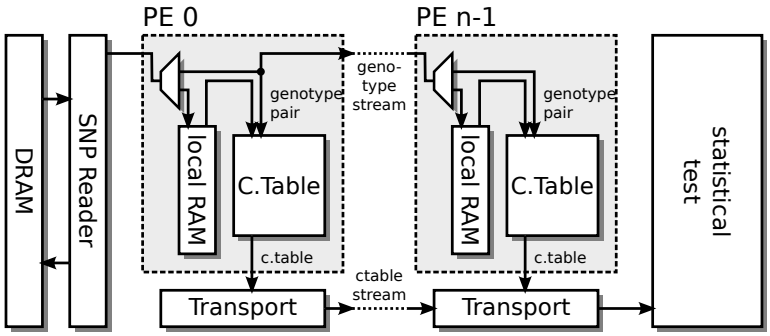

Figure 3: Overview of the systolic chain of processing elements for contingency table creation.

4 Parallel Creation of Contingency Tables

4.1 Systolic Chain of Processing Elements

We have designed a systolic chain of processing elements (PEs) on each FPGA to concurrently generate as many contingency tables as possible. The genotype data is distributed among all FPGAs such that each FPGA processes two intervals of SNPs with all corresponding genotypes of all samples. The data has to be organized in genotypes grouped by cases and controls for each SNP. The contingency tables are created while the data from both intervals is streamed SNP-wise through the chain.

Each PE contains a local memory to store the complete genotype data for one SNP and a number of counters for the entries of the contingency table. The genotype data stream from the previous PE in the chain is directly provided to the next PE after one clock cycle. However, if the local SNP memory has not been initialized yet, the data of the first SNP is streamed into this memory and not provided to the next PE. After initialization of the local SNP, the next SNP in the stream is directly compared to the stored SNP genotype by genotype. For each pair of genotypes the corresponding counter is incremented. Since the genotypes are ordered by case and control group, the contingency table for each group is ready after processing the last genotype of a SNP for the corresponding group in the stream. The tables are provided to a transport bus afterwards and carried to a postprocessing unit which could be an entity calculating a statistic. After each SNP in the stream, the counters of the PE are reset for the next contingency table.

If the streaming process has finished all SNPs from both initial intervals, it starts all over but leaving out the first k SNPs, if k is the number of PEs in the chain. In the next iteration the first $2k$ SNPs are left out, and so on. The process stops until all SNPs from the first interval are to be left out. This way all possible SNP pairings in the first interval, i.e. each pair contains only SNPs from the first interval, and between both intervals, i.e. each pair contains a SNP from each interval, are processed. This makes an efficient distribution of the whole set of SNPs among all available FPGAs possible (see Sect. 4.3). Figures 3 and 4 show the design overview of the systolic chain of PEs and the processing sequence of an example dataset of six SNPs and three PEs.

After calculating the statistic, the results are filtered before being provided to the host. The filter could be threshold-based or, as for iLOCi, storage of the n best results.

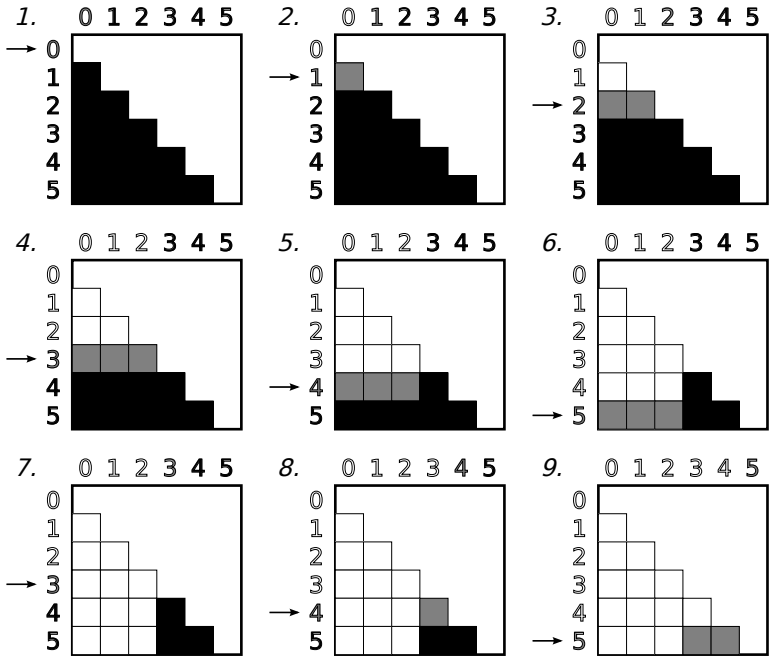

Figure 4: Sequence of processing an example dataset of six SNPs with a chain of three PEs in nine steps. Black squares indicate SNP pair combinations to be processed while white squares indicate already processed pairs. Grey squares are currently being processed while an arrow on the vertical axis indicates the currently streamed SNP.

4.2 Specifics Regarding iLOCi

For the implementation of the iLOCi significance test, we have optimized the overall structure regarding the requirements in Eqs. (1) and (4). Obviously, the same calculations are made for p^{case} and p^{ctrl}. Hence, we have implemented Eq. (4) only once, sharing the resources for both calculations. Furthermore, the entries of the contingency table are required as sums. Thus, we have changed the representation of a contingency table to contain only three values a, b and c according to the following definitions whereby n_{ij} denotes n_{ij}^{case} or n_{ij}^{ctrl} respective to the current group on turn.

$$a = n_{00} - n_{02} - n_{20} + n_{22} \tag{5}$$

$$b = \sum_j n_{0j} + \sum_j n_{2j} \tag{6}$$

$$c = \sum_i n_{i0} + \sum_i n_{i2} \tag{7}$$

These values are calculated on-the-fly by each PE while streaming the genotype data. The calculation of p for p^{case} and p^{ctrl} respectively directly follows from Eq. (4).

$$p = \frac{a}{\sqrt{bc}} \tag{8}$$

The implementation of Eq.(8) requires FPGA resources for one multiplier, one square root extractor and one divider. The multiplication is processed in integer arithmetics while square

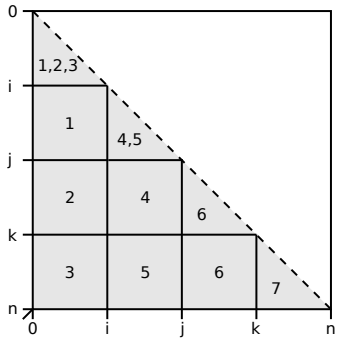

Figure 5: Example SNP distribution among seven FPGAs. Only SNP pairs below the dashed line have to be considered.

root extraction and division require double precision floating point arithmetics. Each component is implemented as a pipeline such that in every clock cycle new input data can be processed. This is necessary since all PEs provide their contingency tables at once with only one clock cycle delay. Furthermore, the FPGA resources are optimally utilized.

p^{case} is calculated before p^{ctrl} for each concurrently processed SNP pairs. Thus, these values have to be stored in a FIFO buffer. After calculating p^{ctrl} the corresponding p^{case} is extracted from the buffer and the difference according to Eq. (1) is computed. Finally, the n-best results of this FPGA are stored in a buffer and provided to the host which collects the final n-best results from all FPGAs afterwards.

4.3 Distribution of Data

To achieve good load balancing for a large number of FPGAs the distribution of genotype data has to follow a certain scheme. Due to the symmetry of the contingency tables only $n(n-1)/2$ tables have to be created for cases and controls (with n the number of SNPs). Therefore, we employ a scheme that enables us to create almost redundant-free SNP pairings while keeping the workload balanced.

Figure 5 shows an example how SNPs would be distributed if seven FPGAs were available. Each FPGA receives two SNP intervals, one interval on the horizontal and one on the vertical axis. According to Sect. 4.1, a rectangular tile of the table space is calculated with this data. Exploiting the symmetry again, the triangle below the symmetry axis (dashed line) is always processed as well for no extra cost. Unfortunately, it is inevitable that a few of these triangles are calculated multiple times. Hence, duplicate results may be produced by different FPGAs which are filtered by the host software. One FPGA (no. 7 in the figure) is reserved for the remaining triangle which is not covered by the calculation of a rectangle.

This scheme obviously works for a number of available FPGAs in the sequence of triangular numbers (1, 3, 6, 10, 15, 21, 28, ...) plus one for the last triangle. As our RIVYERA system contains 128 FPGAs, the nearest number is 120. Hence, to avoid idle FPGAs at all, we split all available FPGAs in two groups with slightly different interval sizes. With this configuration applied on a WTCCC dataset, each FPGA gets two intervals of about $30,000$ SNPs with all corresponding $5,000$ genotypes for each SNP. Since genotypes are coded in a two bit representation, only 75 MB of memory are required, easily fitting in the local DRAM.

Architecture		Energy	Time	Speed (M tests/s)
RIVYERA S6-LX150	780 W	0.05 kWh	4 m	520.83
2x Intel Xeon quad-core @ 2.4 GHz	260 W	4.94 kWh	19 h	1.83
nVidia GeForce GTX Titan	250 W	25.50 kWh	∼102 h	∼0.34

Table 1: iLOCi performance for analysis of a dataset with 500,000 SNPs and 5,000 samples. GPU results are extrapolated.

5 Performance Evaluation and Results

Our implementation of the iLOCi test on RIVYERA S6-LX150 contains 100 PEs on each FPGA for creation of contingency tables and allows to store up to 1000 best results. The FPGA resources are utilized by about 80% regarding the FPGA's slices, leaving enough buffer for more complex tests. The chip frequency is 100 MHz for IO and 150 MHz for genotype streaming, counting and calculation of the significance. The FPGA specification has been developed using the Xilinx ISE Design Suite and the VHDL programming language.

iLOCi [9] takes about 19 hours on a MacPro workstation with two Intel Xeon quad-core CPUs @ 2.4 GHz for a WTCCC dataset [12] with 500,000 SNPs and 5,000 samples. In contrast, our RIVYERA S6-LX150 implementation requires less than 4 minutes, leading to a speedup of more than 285. As iLOCi uses the OpenCL interface, we are also able to measure its performance on a GPU system, specifically an nVidia GeForce GTX Titan. Due to a surprisingly poor performance on this system, we have used the iLOCi example dataset with only 8,000 SNPs and 4,901 samples to extrapolate the runtimes for the WTCCC dataset. We have set the number of stored best results to 1,000 but applied no extra parameters. The runtime is 94 s to calculate the $\sim 8,000^2/2 = 32 \times 10^6$ tests. Extrapolated to the $\sim 500,000^2/2$ tests of a WTCCC dataset, the runtime would be more than four days.

Table 1 lists runtimes of the implementations including their respective power consumption during the operation and their total energy consumption for this task. For the CPU and GPU versions, we have chosen the corresponding thermal design power specification without any peripheral, while we have actually measured the energy consumption of the RIVYERA S6-LX150 system with the on-board IPMI interface.

Furthermore, we have compared the runtime of our iLOCi implementation against other methods that perform an exhaustive creation of contingency tables on the same dataset size (see Table 2). To accomplish this, we have determined the speed in tests per second and interpolated from the published results of the corresponding authors. We made an exception for GBOOST [23]. Since it is freely available and performs best in our compared GPU solutions, we measured the performance on two of our own systems as well. All results have to be treated carefully since the comparison is done over different architectures. However, it shows that our implementation on the RIVYERA architecture is able to outperform other architectures by far.

6 Conclusion

Recent advances in high-throughput genotyping technologies establish the need for fast implementations of statistical epistasis in GWAS. Recent work has shown how GPUs can be used to accelerate such methods. In this paper we have demonstrated that reconfigurable hardware based on FPGAs is another promising alternative for this task. We have presented an efficient

Method	Architecture	Time	Speed (M tests/s)
RIVYERA iLOCi	FPGA (128x Spartan6-LX150)	4 m	520.83
GBOOST *	GPU (GeForce GTX Titan)	1 h 01 m	34.23
GBOOST *	GPU (GeForce GTX650Ti)	2 h 41 m	12.97
GBOOST [23]	GPU (GeForce GTX285)	2 h 43 m	12.81
EpiGPU [2]	GPU (GeForce GTX580)	2 h 55 m	11.90
SHEsisEpi [3]	GPU (2x GeForce GTX285)	27 h	1.29
iLOCi [9]	CPU (2x Xeon quad-core @ 2.4 GHz)	19 h	1.83
BOOST [14]	CPU (@ 3 GHz)	121 h	0.29

Table 2: Performance comparison of different GWAS methods based on exhaustive creation of contingency tables for analysis of a dataset with 500,000 SNPs and 5,000 samples. Results are interpolated from the publications of the corresponding authors, except those marked with (*) have actually been measured.

and flexible parallel architecture for contingency table creation with a subsequent statistical test. Since contingency table creation is common to most epistasis tools our architecture can thus be used to accelerate a large variety of tools by simply interchanging the statistical test core. As a proof-of-concept we have implemented the iLOCi algorithm. This leads to a speedup of between two and three orders of magnitude on the RIVYERA S6-LX150 system compared to the CPU-based iLOCi implementation. Furthermore, we have shown speedups of one to two orders of magnitude compared to the GPU-based implementations GBOOST and EpiGPU.

The utilized Spartan6-LX150 FPGA is still based on 45 nm technology. Newer technology on the market, e.g. a cutting-edge Xilinx Kintex7 based on 28 nm technology provides around three times more logic resources, 10 times more DSPs for faster floating-point computations and 7 times more BRAM for local storage. Furthermore, the smaller transistors in the 28 nm structure allow faster reaction times and thus, higher frequencies. Extrapolating our results to a Kintex7, we expect a further performance gain of around one order of magnitude. A RIVYERA system equipped with 128 FPGAs of this type would make it possible to compute statistical epistasis of large-scale datasets consisting of around 5 million SNPs and ten thousand individuals in less than one hour.

Our future work includes evaluating our existing architecture to other epistasis algorithms and extending it to calculate 3-way interactions.

Acknowledgments

This study makes use of data generated by the Wellcome Trust Case-Control Consortium. A full list of the investigators who contributed to the generation of the data is available from http://www. wtccc.org.uk. Funding for the project was provided by the Wellcome Trust under award 076113 and 085475.

The project was supported through the DFG Clusters of Excellence “Inflammation at Interfaces”.

References

[1] Heather J Cordell. Detecting Gene-Gene Interactions that Underlie Human Diseases. *Nature Review Genetics*, 10(6):392–404, 2009.

[2] Gibran Hemani, Athanasios Theocharidis, Wenhua Wei, and Chris Haley. EpiGPU: exhaustive pairwise epistasis scans parallelized on consumer level graphics cards. *Bioinformatics*, 27(11):1462–1465, April 2011.

[3] Xiaohan Hu, Qiang Liu, Zhao Zhang, Zhiqiang Li, Shilin Wang, Lin He, and Yongyong Shi. SHEsisEpi, a GPU-enhanced genome-wide SNP-SNP interaction scanning algorithm, efficiently reveals the risk genetic epistasis in bipolar disorder. *Cell Research*, 20:854–857, May 2010.

[4] François Van Lishout, Jestinah M Mahachie John, Elena S Gusareva1, Victor Urrea, Isabelle Cleynen, Emilie Théâtre, Benoît Charloteaux, Malu Luz Calle, Louis Wehenkel, and Kristel Van Steen. An efficient algorithm to perform multiple testing in epistasis screening. *BMC Bioinformatics*, 14(138), April 2013.

[5] Brendan Maher. Personal Genomes: the Case of the Missing Heritability. *Nature*, 456(7218):18–21, 2008.

[6] Jonathan Marchini, Peter Donnelly, and Lon R Cardon. Genome-wide strategies for detecting multiple loci that influence complex diseases. *Nature Genetics*, 37:413–417, 2005.

[7] Jason H. Moore, Folkert W. Asselbergs, and Scott M. Williams. Bioinformatics Challenges for Genome-Wide Association Studies. *Bioinformatics*, 26(4):445–455, 2010.

[8] Gerd Pfeiffer, Stefan Baumgart, Jan Schröder, and Manfred Schimmler. A Massively Parallel Architecture for Bioinformatics. In *ICCS2009, Lecture Notes in Computer Science*, volume 5544, pages 994–1003. Springer, 2009.

[9] Jittima Piriyapongsa, Chumpol Ngamphiw, Apichart Intarapanich, Supasak Kulawonganunchai, Anunchai Assawamakin, Chaiwat Bootchai, Philip J. Shaw, and Sissades Tongsima. iLOCi: a SNP interaction prioritization technique for detecting epistasis in genome-wide association studies. *BMC Genomics*, 13(Suppl 7):S2+, 2012.

[10] Snehit Prabhu and Itsik Pe'er. Ultrafast genome-wide scan for SNP–SNP interactions in common complex disease. *Genome Research*, 22(11):2230–2240, November 2012.

[11] Marylyn D. Ritchie, Lance W. Hahn, Nady Roodi, L. Renee Bailey, William D. Dupont, Fritz F. Parl, and Jason H. Moore. Multifactor-Dimensionality Reduction Reveals High-Order Interactions among Estrogen-Metabolism Genes in Sporadic Breast Cancer. *Am. J. Hum. Genet.*, 69(1):138–147, July 2001.

[12] The Wellcome Trust Case Control Consortium. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, 447(7145):661–78, June 2007.

[13] Kristel van Steen. Travelling the world of gene–gene interactions. *Briefings in Bioinformatics*, 13(1):1–19, 2011.

[14] Xiang Wan, Can Yang, Qiang Yang, Hong Xue, Xiaodan Fan, Nelson L.S. Tang, and Weichuan Yu. BOOST: A Fast Approach to Detecting Gene-Gene Interactions in Genome-wide Case-Control Studies. *Am. J. Hum. Genet.*, 87(3):325–340, September 2010.

[15] Xiang Wan, Can Yang, Qiang Yang, Hong Xue, Nelson L.S. Tang, and Weichuan Yu. Predictive rule inference for epistatic interaction detection in genome-wide association studies. *Bioinformatics*, 26(1):30–37, 2010.

[16] Xiang Wan, Can Yang, Qiang Yang, Hongyu Zhao, and Weichuan Yu. The complete compositional epistasis detection in genome-wide association studies. *BMC Genetics*, 14(7), February 2013.

[17] Yue Wang, Guimei Liu, Mengling Feng, and Limsoon Wong. An empirical comparison of several recent epistatic interaction detection methods. *Bioinformatics*, 27(21):2936–2943, 2011.

[18] Lars Wienbrandt. Bioinformatics Applications on the FPGA-Based High-Performance Computer RIVYERA. In Wim Vanderbauwhede and Khaled Benkrid, editors, *High-Performance Computing Using FPGAs*, pages 81–103. Springer, 2013.

[19] Lars Wienbrandt, Stefan Baumgart, Jost Bissel, Florian Schatz, and Manfred Schimmler. Massively parallel FPGA-based implementation of BLASTp with the two-hit method. In *ICCS2011, Procedia Computer Science*, volume 1, pages 1967–1976, 2011.

[20] Lars Wienbrandt, Daniel Siebert, and Manfred Schimmler. Improvement of BLASTp on the FPGA-based High-Performance Computer RIVYERA. In *ISBRA2012, Lecture Notes in Bioinformatics*, volume 7292, pages 275–286, 2012.

[21] Jing Wu, Bernie Devlin, Steven Ringquist, Massimo Trucco, and Kathryn Roeder. Screen and Clean: a tool for identifying interactions in genome-wide association studies. *Gentic Epidemiology*, 34(3):275–285, April 2010.

[22] Can Yang, Zengyou He, Xiang Wan, Qiang Yang, Hong Xue, and Weichuan Yu. SNPHarvester: a filtering-based approach for detecting epistatic interactions in genome-wide association studies. *Bioinformatics*, 25(4):504–511, 2009.

[23] Ling Sing Yung, Can Yang, Xiang Wan, and Weichuan Yu. GBOOST: a GPU-based tool for detecting gene-gene interactions in genome-wide case control studies. *Bioinformatics*, 27(9):1309–1310, 2011.

[24] Xiang Zhang, Shunping Huang, Fei Zou, and Wei Wang. TEAM: efficient two-locus epistasis tests in human genome-wide association study. *Bioinformatics*, 26(12):i217–i227, July 2010.