1,000x Faster than PLINK: Genome-Wide Epistasis Detection with Logistic Regression Using Combined FPGA and GPU Accelerators

Lars Wienbrandt*, Jan Christian Kässens, Matthias Hübenthal, and David Ellinghaus

Institute of Clinical Molecular Biology, University Medical Center Schleswig-Holstein, Campus Kiel, Kiel University, Germany
{l.wienbrandt,j.kaessens,m.huebenthal,d.ellinghaus}@ikmb.uni-kiel.de

Abstract. Logistic regression as implemented in PLINK is a powerful and commonly used framework for assessing gene-gene (GxG) interactions. However, fitting regression models for each pair of markers in a genome-wide dataset is a computationally intensive task. Performing billions of tests with PLINK takes days if not weeks, for which reason prefiltering techniques and fast epistasis screenings are applied to reduce the computational burden.
Here, we demonstrate that employing a combination of a Xilinx UltraScale KU115 FPGA with an Nvidia Tesla P100 GPU leads to runtimes of only minutes for logistic regression GxG tests on a genome-wide scale. In particular, a dataset of 53,000 samples genotyped at 130,000 SNPs was analyzed in 8 minutes, resulting in a speedup of more than 1,000 when compared to PLINK v1.9 using 32 threads on a server-grade computing platform. Furthermore, on-the-fly calculation of test statistics, p-values and LD-scores in double-precision make commonly used prefiltering strategies obsolete.

Keywords: genome-wide association study (GWAS) · genome-wide interaction study (GWIS) · gene-gene (GxG) interaction · linkage disequilibrium (LD) · BOOST · hardware accelerator · hybrid computing · heterogeneous architecture

1 Introduction

Gene-gene (GxG) interactions (epistasis) are believed to be a significant source of unexplained genetic variation causing complex chronic diseases. Several studies provided evidence for statistical GxG interaction between the top disease-associated single nucleotide polymorphisms (SNPs) of complex chronic diseases, including ankylosing spondylitis [21], Behçet's disease [15], type 2 diabetes [14], and psoriasis [6]. Particularly, in psoriasis a significant interaction ($p = 6.95 \times 10^{-6}$) as measured by logistic regression has been detected between the genes

* corresponding author

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

ERAP1 (*rs27524*) and *HLA-C* (*rs10484554*). The biological consequence of this interaction is that the *ERAP1* SNP only has an effect in individuals carrying at least one copy of the risk allele at the *HLA-C* SNP.

In general, detection of GxG interactions poses a great challenge for genome-wide association studies (GWAS) due to the computational burden of testing billions of pairs of SNPs (as a result of the number of tests being quadratic in the number of SNPs). Traditional logistic regression analysis is still the gold-standard to detect statistical GxG interactions in case/control studies, but too slow in practice to screen for GxG interactions on a genome-wide scale. Thus, many approximate methods for epistasis screening have been proposed applying a variety of heuristic and filtering techniques to conduct genome-wide interaction studies (GWIS) in a reasonable amount of time. Well-established methods include the Kirkwood Superposition Approximation (KSA) of the Kullback-Leibler divergence implemented in BOOST [23] as well as the joint effects test introduced by Ueki et al. [22]. Another exhaustive interaction method, called GWIS [7], employs a permutation-based approach to calibrate test statistics. Similarly, MB-MDR [3] uses permutations to adjust the p-value of the significance test. However, it is able to reduce the dimensionality of any problem into one dimension categorizing into high-risk, low-risk and no evidence groups before calculating a chi-squared test statistic. Other tools defining different test statistics include BiForce [8], iLOCi [18] and EDCF [27]. The latter uses a clustering approach in order to reduce the computational burden. Recently, entropy-based measures for GxG interaction detection gained increasing attention. A well-written overview can be found in [5].

However, no convincing GxG loci have been identified exclusively from GWIS using these approaches. Many of the methods derive an upper bound on the test statistic in order to prune the search space and conduct follow-up model-fitting analysis using logistic regression on a pre-filtered subset of pairs [24]. Furthermore, the computational load for preliminary epistasis screenings is not negligible. Accordingly, several tools emerged to speedup this process employing hardware accelerators, such as GPUs in GBOOST [28] or SHEsisEpi [9]. Another way to reduce the computational burden is to reduce the number of SNPs in advance by pre-filtering for linkage disequilibrium (LD), although it can be shown that SNPs supposed to be in LD may also reveal an interaction effect [2,10].

An attempt to reduce the computational load for logistic regression tests is made in [17] by using GLIDE [11]. To our knowledge, GLIDE is the fastest currently available GPU implementation of the logistic regression GxG interaction test. More recently, CARAT-GxG [16] emerged. It also offers linear regression including covariate analysis on GPUs, but provides a poor performance when compared to GLIDE (12 days for a dataset containing 500,000 SNPs and not more than 1,000 samples using 32 Nvidia Tesla M2070 GPUs).

In this paper, we show that we are able to perform an exhaustive genome-wide logistic regression analysis for SNP-SNP interactions on datasets consisting of hundreds of thousands of SNPs and tens of thousands of samples in minutes, thus eliminating the needs for epistasis screening or LD-filtering as a preprocessing

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

step. If required, LD-filtering can directly be applied as a postprocessing step, thanks to on-the-fly calculation of r^2. Furthermore, we perform our calculations in double-precision floating point format in order to overcome precision problems that may occur during floating point accumulations.

We run our benchmark against PLINK v1.9 using 32 threads on a computing system with two Intel Xeon E5-2667v4 eight-core CPUs. We already gain a 10-11 times speedup by sacrificing the support for sample covariates (re-enabling it in our method is still under development) and adapting the logistic regression test to be used with contingency tables. This reduces the computational complexity from $\mathcal{O}(NT)$ to $\mathcal{O}(N+T)$ (with N indicating the number of samples and T the number of iterations required for a single test). By harnessing a combination of only two hardware accelerators, namely a Xilinx Kintex UltraScale KU115 FPGA and an Nvidia Tesla P100 GPU, we gain another 100 times speedup resulting in a total of >1,000 times speedup compared to multi-threaded PLINK on a server-grade platform. Exemplary, for analyzing a dataset consisting of 130k SNPs and 53k samples our method requires only 8 minutes while PLINK running with 32 threads almost requires 6 days. Finally, it turns out that our method is even more than 300 times faster than GLIDE, which harnesses 12 Nvidia GTX 580 GPUs.

2 Pairwise Epistasis Testing

2.1 Logistic Regression Test

In this article we address the efficient implementation of a genotype-based statistical test for binary traits. Let Y be a random variable correlated with the trait. Correspondingly, for the trait being a disease, we define the two possible outcomes of Y as $Y = 1$ if the sample is a *case* affected by the disease, and $Y = 0$ if the samples is a *control* unaffected by the disease. Furthermore, for a pairwise test, we define X_A and X_B as random variables correlated with the observation of genotypes at SNPs A and B, respectively. The possible outcomes of $X_{A/B}$ are $g_{A/B} \in \{0, 1, 2\}$ representing the observed genotype (0 = homozygous reference, 1 = heterozygous, 2 = homozygous variant). PLINK [4,20] uses the following multiplicative logistic regression affection model with β_3 indicating the interaction effect of SNPs A and B.

$$\ln \frac{P(Y=1|X_A=g_A, X_B=g_B)}{P(Y=0|X_A=g_A, X_B=g_B)} = \beta_0 + \beta_1 g_A + \beta_2 g_B + \beta_3 g_A g_B \tag{1}$$

PLINK employs Newton's method to iteratively obtain ML estimates of the model parameters. It firstly generates a covariate matrix $\mathbf{C}$ with entries C_{ij} whereby i indicates a sample of the input dataset and $j \in 0, 1, 2, 3$ indicates a column for each β_j. The matrix is defined as follows:

$$C_{i0} = 1, \quad C_{i1} = g_{iA}, \quad C_{i2} = g_{iB} \quad \text{and} \quad C_{i3} = g_{iA} g_{iB} \tag{2}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

In detail, for a variable number of iterations $t = 0, \ldots, T-1$, fitting the vector $\boldsymbol{\beta}$ is performed in a stepwise manner. $\boldsymbol{\beta}^{(0)}$ is initialized with $\beta_j^{(0)} = 0 \; \forall j$ for the first iteration $t = 0$.

1. For each sample i, compute intermediate variables

$$p_i^{(t)} = \bar{p}_i^{(t)} - y_i \quad \text{and} \quad v_i^{(t)} = \bar{p}_i^{(t)} \left(1 - \bar{p}_i^{(t)}\right) \tag{3}$$

where

$$\bar{p}_i^{(t)} = \left(1 + \exp\left(-\sum_j \beta_j^{(t)} C_{ij}\right)\right)^{-1}. \tag{4}$$

2. Compute gradient

$$\boldsymbol{\nabla}^{(t)} = \left\{\nabla_j^{(t)}\right\}_{j=0}^{3} = \sum_i C_{ij} p_i^{(t)}. \tag{5}$$

3. Compute Hessian matrix

$$\boldsymbol{H}^{(t)} = \left\{h_{jk}^{(t)}\right\}_{j,k=0}^{3} = \begin{cases} 0 & \text{if } k > j \\ \sum_i C_{ij} C_{ik} v_i^{(t)} & \text{if } k \leq j \end{cases}. \tag{6}$$

4. Compute $\boldsymbol{\Delta\beta}^{(t)} = \left\{\Delta\beta_j^{(t)}\right\}_{j=0}^{3}$ by efficiently solving the linear system

$$\boldsymbol{H}^{(t)} \boldsymbol{\Delta\beta}^{(t)} = \boldsymbol{\nabla}^{(t)} \tag{7}$$

using the Cholesky decomposition of $\boldsymbol{H}^{(t)}$.

5. Update model parameters

$$\boldsymbol{\beta}^{(t+1)} \leftarrow \boldsymbol{\beta}^{(t)} - \boldsymbol{\Delta}\beta^{(t)}. \tag{8}$$

If $\sum_j \Delta\beta_j^{(t)}$ approaches zero, i.e. there is no more significant change, the process stops with $\boldsymbol{\beta}^{(t+1)}$ as the current result. Otherwise, the next iteration is started with step 1. However, if the change does not converge to zero, the process stops after a fixed number of iterations. PLINK uses at maximum 16 iterations and a close-to-zero threshold of 0.0001. Additional tests for convergence failure are implemented but omitted here for the sake of brevity.

The result of the logistic regression test in PLINK is composed of three components, namely the test statistic, its approximate p-value and the odds-ratio. The test statistic χ^2 is calculated as

$$\chi^2 = \frac{\beta_3}{\varepsilon^2}. \tag{9}$$

ε is the standard error for the $g_A g_B$-term in (1). It can directly be determined by solving the linear system $\boldsymbol{H}^{(t)}\boldsymbol{e} = (0, 0, 0, 1)$ and defining $\varepsilon^2 = e_3$.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

cases ($Y=1$)		SNP A 0	1	2
SNP B	0	n_{00}^{case}	n_{01}^{case}	n_{02}^{case}
	1	n_{10}^{case}	n_{11}^{case}	n_{12}^{case}
	2	n_{20}^{case}	n_{21}^{case}	n_{22}^{case}

controls ($Y=0$)		SNP A 0	1	2
SNP B	0	n_{00}^{ctrl}	n_{01}^{ctrl}	n_{02}^{ctrl}
	1	n_{10}^{ctrl}	n_{11}^{ctrl}	n_{12}^{ctrl}
	2	n_{20}^{ctrl}	n_{21}^{ctrl}	n_{22}^{ctrl}

Fig. 1. Contingency tables for cases and controls. n_{ij} reflect the number of occurrences for the corresponding genotype combination in a given pair of SNPs.

According to PLINK, the test statistic is assumed to follow a chi-squared distribution χ_1^2 with one degree of freedom, which implies that the approximate p-value can directly be determined from its cumulative distribution function. Finally, the odds-ratio is defined as e^{β_3}.

Obviously, steps 1 to 3 in each iteration have linear complexity in N, i.e. $\mathcal{O}(N)$ whereby N is the number of samples. Let T be the number of iterations, then $\mathcal{O}(NT)$ is the total complexity for a single test. In the next Sects. 2.2 and 2.3, we show how to do a linear precomputing step to generate a contingency table and how to apply the contingency table in the logistic regression test, which results in a constant computation complexity for each iteration.

2.2 Contingency Tables

For any SNP pair (A, B) a contingency table represents the number of samples in a dataset that carry a specific genotype information. In particular, an entry n_{ij} represents the number of samples that carry the information $g_A = i$ at SNP A and $g_B = j$ at SNP B. Thus, a contingency table for pairwise genotypic tests contains 3×3 entries. Since we are focusing on binary traits, we require a contingency table for each state, w.l.o.g. one for the *case* and *control* group, respectively, and denote their entries by n_{ij}^{case} and n_{ij}^{ctrl} (see Fig. 1).

For a given SNP pair generating the contingency tables is clearly linear in the number of samples. In the next section (Sect. 2.3) we show how to incorporate contingency tables into logistic regression.

2.3 Logistic Regression with Contingency Tables

The information in the contingency tables for case and control group can be used to simplify steps 1 to 3 in Sect. 2.1. Steps 4 and 5 as well as the calculation of the test statistic, the odds-ratio and the p-value remain the same.

1. From a given contingency table we compute the following intermediate variables.

$$p_{ij}^{(t)} = \left(1 + \exp\left(-\left(\beta_0^{(t)} + i\beta_1^{(t)} + j\beta_2^{(t)} + ij\beta_3^{(t)}\right)\right)\right)^{-1} \tag{10}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

$$p_{ij}^{(t),\mathrm{ctrl}} = p_{ij}^{(t)}, \quad p_{ij}^{(t),\mathrm{case}} = p_{ij}^{(t)} - 1, \quad v_{ij}^{(t)} = p_{ij}^{(t)} \left(1 - p_{ij}^{(t)}\right) \left(n_{ij}^{\mathrm{case}} + n_{ij}^{\mathrm{ctrl}}\right) \tag{11}$$

2. The gradient $\boldsymbol{\nabla}^{(t)}$ from (5) can now be computed as

$$\boldsymbol{\nabla}^{(t)} = \left(\sum_{ij} N_{ij}^{(t)}, \sum_{ij} iN_{ij}^{(t)}, \sum_{ij} jN_{ij}^{(t)}, \sum_{ij} ijN_{ij}^{(t)} \right) \tag{12}$$

where

$$N_{ij}^{(t)} = \left(n_{ij}^{\mathrm{case}} p_{ij}^{(t),\mathrm{case}} + n_{ij}^{\mathrm{ctrl}} p_{ij}^{(t),\mathrm{ctrl}} \right) \tag{13}$$

3. The Hessian matrix $\boldsymbol{H}^{(t)}$ from (6) evaluates to

$$\boldsymbol{H}^{(t)} = \left\{ h_{pq}^{(t)} \right\}_{p,q=0}^{3} = \begin{pmatrix} \sum v_{ij}^{(t)} & 0 & 0 & 0 \\ \sum iv_{ij}^{(t)} & \sum i^2 v_{ij}^{(t)} & 0 & 0 \\ \sum jv_{ij}^{(t)} & \sum ijv_{ij}^{(t)} & \sum j^2 v_{ij}^{(t)} & 0 \\ \sum ijv_{ij}^{(t)} & \sum i^2 jv_{ij}^{(t)} & \sum ij^2 v_{ij}^{(t)} & \sum i^2 j^2 v_{ij}^{(t)} \end{pmatrix} \tag{14}$$

where each sum is evaluated over all indexes i and j.

Obviously, the complexity of each iteration step is now constant, i.e. $\mathcal{O}(1)$. As in Sect. 2.1, let N be the total number of samples and T the number of iterations. We recall the complexity of the method used by PLINK with $\mathcal{O}(NT)$. Our proposed method improves this complexity to $\mathcal{O}(N+T)$ which can directly be observed in a significant increase in computation speed (see Sect. 4).

2.4 Linkage Disequilibrium

Our ultimate aim is the exhaustive testing of all SNP pairs on a genome-wide scale without pre-filtering with regard to linkage disequilibrium (LD). However, to be able to apply posthoc LD filtering we compute the r^2-score on-the-fly. r^2 is a measure of similarity between two SNPs. It is defined as

$$r^2 = \frac{D^2}{p_A(1-p_A)p_B(1-p_B)} \quad \text{with} \quad D = p_{AB} - p_A p_B. \tag{15}$$

D is the distance between the observed allele frequency p_{AB} at loci A and B and the expected allele frequency $p_A p_B$ assuming statistical independence. Thus, r^2 is a normalized measure for D which can be used for comparison of different SNP pairs. The allele frequencies p_A and p_B can directly be determined as

$$p_A = \frac{2n_{00} + 2n_{10} + 2n_{20} + n_{01} + n_{11} + n_{21}}{2N} \tag{16}$$

$$p_B = \frac{2n_{00} + 2n_{01} + 2n_{02} + n_{10} + n_{11} + n_{12}}{2N} \tag{17}$$

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

whereby $n_{ij} = n_{ij}^{\text{case}} + n_{ij}^{\text{ctrl}}$ for all i, j. Unfortunately, the determination of the allele frequency p_{AB} from genotypic data is not straightforward. This is due to the unknown phase when two heterozygous genotypes face each other in a SNP pair. Basically, it can be defined as

$$p_{AB} = \frac{2n_{00} + n_{01} + n_{10} + x}{2N} \tag{18}$$

with x meeting $x \leq n_{11}$. x has to satisfy the following equation whose solution is omitted here for simplicity:

$$(f_{00} + x)(f_{11} + x)(n_{11} - x) = (f_{01} + n_{11} - x)(f_{10} + n_{11} - x)x \tag{19}$$

where f_{ij} is the number of allele combinations ij we know for sure, e.g. $f_{00} = 2n_{00} + n_{01} + n_{10}$ and $f_{11} = 2n_{22} + n_{21} + n_{12}$.

PLINK does not compute the r^2-score jointly with the logistic regression test. However, one can create a table of r^2 scores explicitly for all pairs of a given range (`--r2` switch), or compute the r^2-score for a single pair (`--ld` switch). This process is in linear complexity for each pair of SNPs to determine the respective allele frequencies. In comparison, we are using the information of the precomputed contingency table which allows us to calculate the r^2-score in constant time.

3 Implementation

3.1 Heterogeneous FPGA-GPU Computing Architecture

Our implementation targets a heterogeneous FPGA-GPU computing architecture. We improved our architecture proposed in [26] by adding high-end off-the-shelf components, namely a server-grade mainboard hosting two Intel Xeon E5-2667v4 8-core CPUs @ 3.2 GHz and 256 GB of RAM, an NVIDIA Tesla P100 GPU, and an Alpha Data ADM-PCIE-8K5 FPGA accelerator card.

The GPU accelerator is equipped with 16 GB of graphics memory and is connected via PCI Express Gen3 x16. The FPGA accelerator hosts a recent Xilinx Kintex UltraScale KU115 FPGA with two attached 8 GB SODIMM memory modules. It is connected via PCI Express Gen3 x8 allowing high-speed communication with the host and the GPU. The system runs a Ubuntu 17.10 Linux OS (Kernel version 4.13).

Due to driver restrictions, it is currently not possible to perform direct peer transfers, i.e. moving data from an FPGA accelerator to a GPU or vice-versa. Therefore, both devices are placed in slots that are served by the same CPU to reduce transmission overhead as described in [13].

According to the PCIe specifications, the net transmission rate between FPGA and GPU is about 7.3 GB/s. This absolutely fits our application demands, such that the transmission interface does not become a bottleneck.

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

3.2 Task Distribution

Similar to our method for testing third-order SNP interactions based on information gain [26], we split the application into three subtasks. Firstly, the creation of pairwise contingency tables (see Sect. 2.2) is done by the FPGA module. Secondly, all computations required for the logistic regression test based on the contingency tables are performed by the GPU. And thirdly, the host collects and filters the results created by the GPU.

As before, data transmission between the modules is performed by DMA transfers via PCI Express, and since there is no direct connection between the FPGA and the GPU module, the transmission of the contingency tables is redirected via the host memory.

The input dataset is assumed to be in binary PLINK format, i.e. three files in `.bed`, `.bim`, `.fam` format. The output is in plain text format containing for each result the information on the respective SNP pair (name and ID), χ^2 test statistic, odds-ratio, approximate p-value and r^2-score.

Contingency table creation on the FPGA. The FPGA pipeline for contingency table generation is based on our previous work for pairwise [12,25] and third-order interactions [26]. Thus, we omit details here and only remark the differences.

Shortly summarized, the pipeline consists of a chain of 480 process elements (PEs) divided into two subchains of 240 PEs each. After a short initialization phase, the chain produces 480 contingency tables in parallel while the genotype data of one SNP is streamed through the pipeline at a speed of 266 MHz and 8 genotypes/cycle. This sums up to a peak performance of about 40.8 million contingency tables per second for a dataset containing about 50,000 samples, as used in our performance evaluation in Sect. 4.

In previous publications, we used a sparse contingency table representation lacking support for unknown genotypes. The disadvantage of such a design is, that datasets containing unknown genotypes could not be supported because the assumption that the sum of all entries stays the same over all tables is disproved in the presence of unknowns. In order to remove this limitation, we now transfer complete tables from the FPGA to the GPU. Unfortunately, this increases the transmission rate significantly. Therefore, we encode each table entry into two bytes, i.e. 18 B per table. For each pair of corresponding case and control tables 4 B for the pair ID is added, which accumulates to 40 B per table pair. Hence, the peak transmission rate for the example above is about 816 MB/s, compared to 245 MB/s required for the sparse representation. However, the peak transmission rate of the architecture is 7.3 GB/s according to PCIe Gen3 specifications which theoretically allows us to process datasets down to 5,200 samples without the transmission link becoming the bottleneck.

Processing contingency tables on the GPU. The GPU stores the buffers from the FPGA containing contingency tables in graphics memory. We used

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

a transmission buffer size of 256 MB which may hold up to 6.7 million table pairs. The computation process follows a simple parallelization scheme over GPU threads. By setting the block size to the maximum supported block size and the grid size to evenly distribute the contingency tables over the blocks, each thread processes exactly one contingency table pair, and only one kernel call per buffer is required.

Logistic regression and LD computation have been implemented as described in Sects. 2.3 and 2.4. However, in contrast to PLINK, we use the double precision floating point format in all our computations. The output is written into a result buffer. We provide one result buffer for each table transmission buffer, which is transferred to the host as soon as processing a table buffer has finished.

By evenly distributing the contingency tables over the blocks, we most likely introduce an unequal load resulting from a varying number of Newton iterations per thread. However, the average number of iterations per block remains virtually constant.

Transmission buffer management and result collection on the host. We used a similar transmission buffer management as presented in [26], but introduced some improvements. In order to reduce transmission overhead, we used different adapted buffer sizes for contingency table transmission between FPGA and GPU, and result transmission from GPU to host. We used a transmission buffer size of 256 MB for contingency tables leading to 230.4 MB for results (reserving space for one result per contingency table pair). As before, the buffers are page-locked to ensure a fast transmission without delay, and the number of buffers allocated for each connection is equal (eight per default).

Multiple threads on the host system perform the collection of results by filtering by a given significance threshold and finally providing them sorted with regard to the test statistic. For this purpose, the min-max fine heap data structure [1,13] is employed. Each thread keeps its own instance of a min-max heap to avoid lock conditions and inserts a result only if the test statistic exceeds the threshold. Then, the output file is composed by iteratively extracting the single best result over all heaps until the heaps are drained or the number of requested results is reached, whichever occurs first.

The complete workflow on our heterogeneous FPGA-GPU-based architecture is illustrated in Fig. 2.

4 Performance Evaluation

For performance evaluation we prepared six datasets based on in-house cohorts. Dataset “A” and “B” contain 14,513 and 19,085 cases of autoimmune diseases, respectively, and share a common collection of 34,213 healthy controls. Modified instances of sets “A” and “B” were generated by applying an LD filter with r^2-threshold of 0.2, resulting in sets “A LD” and “B LD”, respectively. Furthermore, we reduced the latter two datasets to only comprise SNPs located on

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

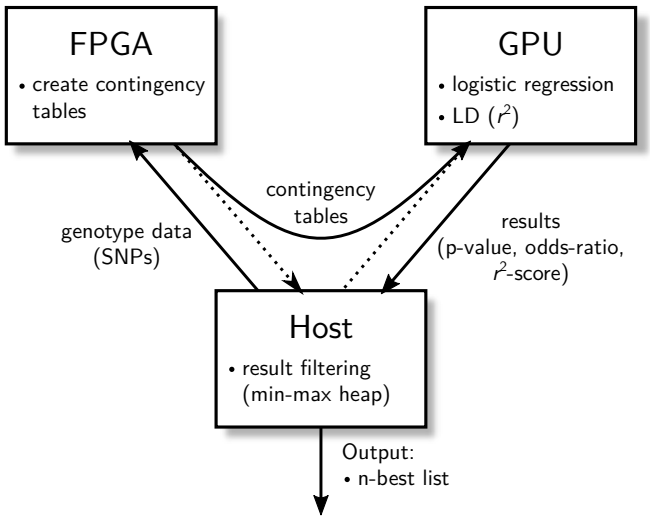

Fig. 2. Workflow on our heterogeneous system. 1. Genotypic data is sent to the FPGA. 2. For each pair of SNPs the FPGA creates contingency tables. 3. The contingency tables are sent to the GPU employing a memory buffer on the host. 4. The GPU calculates the logistic regression and LD. 5. Results (p-value, odds-ratio and LD-score) are transferred back to host. 6. Result are filtered using a min-max heap on host.

chromosomes 5 and 6. The resulting sets have been denoted by "A LD chr5,6" and "B LD chr5,6". An overview of these datasets can be found in Tab. 1.

Our target system was the architecture described in Sect. 3.1. We compiled our implementation with GCC 5.4.1 and CUDA 9.0. The FPGA code was written in VHDL and compiled with Xilinx Vivado 2017.3. For comparison, we used the to date most recent 64-bit PLINK v1.9 built published on Jan 9th, 2018. We ran PLINK on all six datasets in two ways. Firstly, we computed the standard logistic regression tests with flags `--epistasis --epi1 5e-8` which filters the results by a genome-wide significance threshold of $5 \cdot 10^{-8}$ according to the approximate p-value. Secondly, we applied a faster epistasis screening test with the BOOST [23] method (included in PLINK) with the same threshold flag, but replacing `--epistasis` with `--fast-epistasis boost`. Both runs used all available 32 threads (`--threads`) on our 2x Intel Xeon E5-2667v4 system.

We ran our implementation from a hybrid built, i.e. using both accelerators (Xilinx Kintex UltraScale KU115 FPGA and Nvidia Tesla P100 GPU) and a CPU-only built using all 32 threads. In contrast to PLINK, our implementations perform all calculations in double-precision floating point format, while PLINK only uses single-precision. Furthermore, we calculated the r^2-score in order to test for linkage disequilibrium on all SNP pairs, which PLINK does not.

We verified the correctness of our implementation by comparing our results with the PLINK results. At first, we encountered a lot of differences in the score and also in the order. Thus, we modified the source code of PLINK to let it do the

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

calculations in double-precision as well. This modification increased the runtime of PLINK by a factor of about 5.7, but the results were almost exactly equal now, showing that the inconsistencies were caused by the different precisions. We believe the remaining small inconsistencies were due to numerical problems in PLINK when accumulating small floating-point values over all samples in steps 2 and 3 of computing the logistic regression test (see (5) and (6) in Sect. 2.1).

The wall-clock runtimes were measured with the GNU time command and without additional system load. The results are listed in Tab. 2. The measures demonstrate that by applying our method that reduces runtime complexity by using contingency tables, we can gain a 10-11 times speedup. With an additional application of the combination of two hardware accelerators, namely FPGA and GPU, we gain an additional speedup of about 100, resulting in a total computation speed that it is more than 1,000 times faster than that of PLINK on a server-grade high-end platform. The performance is underlined by the additional burden on our implementation, which is a higher calculation precision and the additional on-the-fly r^2-score computation, which is not performed by the PLINK software. Furthermore, our full logistic regression test is still almost 7 times faster than the quick but imprecise pre-scanning method BOOST [23].

However, Tab. 2 also shows, that small datasets, that have a very short runtime on our hybrid system, do not gain a high speedup. The reason is a large overhead for file reading, buffer preparation, device initialization and file output. In particular, these processes take about 10 seconds for the two smallest datasets in our ensemble, which implies a total pipeline run of less than two seconds for the real task. Since this processes cannot be simplified for a single task, we implemented a scheduling system that allows exclusive access to the accelerator pipeline for parallel tasks, but pre- and post-processing can be run concurrently.

We exemplary compared our computational speed to GLIDE [11]. For this purpose, we extrapolated GLIDE's presented interaction speed to the number of samples used in our evaluation datasets. Combined with the number of interaction tests required for our data, we calculated the runtime of GLIDE for dataset "A" as 44.7 hours and for dataset "B" as 48.9 hours. This leads to a speedup of 361 and 364 respectively.

Table 1. Overview of datasets.

Dataset	# samples	# SNPs	Comment
A LD chr5,6	48,726	5,725	Disease A, LD0.2-filtered, only chromosomes 5,6
B LD chr5,6	53,298	5,725	Disease B, LD0.2-filtered, only chromosomes 5,6
A LD	48,726	37,358	Disease A, LD0.2-filtered
B LD	53,298	37,358	Disease B, LD0.2-filtered
A	48,726	130,052	Disease A, complete
B	53,298	130,052	Disease B, complete

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

Table 2. Wall-clock runtimes and speedup of the hybrid FPGA-GPU logistic regression test compared to PLINK [19] logistic regression (`--epistasis`) and PLINK BOOST (`--fast-epistasis boost`) and our CPU-only implementation, all using 32 threads on two Intel Xeon E5-2667v4 processors. Our CPU-only and hybrid implementations additionally calculate the r^2-score (LD) and do all computations in double-precision format (vs. single-precision without r^2 in PLINK).

Dataset	PLINK		Hybrid		Speedup	
	log.reg.	BOOST	CPU-only	FPGA-GPU	CPU-only	Hybrid
A LD chr5,6	15 m 48 s	7 s	1 m 32 s	11 s	10.30	86.18
B LD chr5,6	16 m 48 s	8 s	1 m 39 s	12 s	10.18	84.00
A LD	11 h 09 m 38 s	4 m 05 s	57 m 51 s	50 s	11.58	803.56
B LD	11 h 49 m 32 s	4 m 34 s	1 h 02 m 45 s	53 s	11.31	803.25
A	5 d 14 h 06 m	49 m 34 s	11 h 40 m 12 s	7 m 25 s	11.49	**1,084.85**
B	5 d 18 h 18 m	54 m 34 s	12 h 39 m 05 s	8 m 03 s	10.93	**1,030.81**

5 Conclusions and Future Work

In this paper, we presented two ways of improving performance of PLINK's logistic regression epistasis test [4,20]. Firstly, we reduced the computational complexity from $\mathcal{O}(NT)$ to $\mathcal{O}(N+T)$ for a single test by introducing contingency tables (see Sect. 2). This already led to a speedup of a factor of more than 10 for all our example datasets, although we were even calculating in double-precision format.

The second improvement was made by applying a two-step hardware acceleration pipeline (see Sect. 3). By generating contingency tables on a Kintex UltraScale KU115 FPGA and computing the logistic regression based on the tables on an Nvidia Tesla P100 GPU, we gained a total speedup of more than 1,000 when compared to the original PLINK v1.9 software run with 32 threads on a server-grade two processor (Intel Xeon E5-2667v4) system.

Furthermore, we demonstrated that by employing contingency tables, the LD-score r^2 can be computed on-the-fly. In combination, this provides a powerful tool for epistasis analysis on large datasets, making LD-filtering deprecated as a pre-processing step.

Consequently, we are able to calculate a full logistic regression test in double-precision format on all pairs of hundreds of thousands of SNPs with tens of thousands of samples in a few minutes and allow to filter the results by score and/or by LD in the post-processing stage.

Currently, our method does not support the use of a covariate matrix as additional user input. However, we are currently working on a solution based on weighted contingency tables in order to be able to incorporate covariate information.

In order to make the system available for the scientific community, we are currently working on a much more powerful successor by enhancing it with three additional Xilinx UltraScale FPGAs and Nvidia Tesla P100 GPUs. Furthermore,

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

we aim to develop a web interface to allow scientists to perform genome-wide epistasis tests on our system.

References

1. Atkinson, M.D., Sack, J.R., Santori, N., et al.: Min-max heaps and generalized priority queues. Communications of the ACM **29**(10), 996–1000 (October 1986)
2. Bulik-Sullivan, B.K., Loh, P.R., Finucane, H.K., et al.: LD Score regression distinguishes confounding from polygenicity in genome-wide association studies. Nature Genetics **47**, 291–295 (Feb 2015). https://doi.org/10.1038/ng.3211
3. Cattaert, T., Calle, M.L., Dudek, S.M., et al.: Model-Based Multifactor Dimensionality Reduction for detecting epistasis in case-control data in the presence of noise. Ann. Hum. Genet. **75**(1), 78–89 (2011)
4. Chang, C.C., Chow, C.C., Tellier, L.C., Vattikuti, S., Purcell, S.M., Lee, J.J.: Second-generation PLINK: rising to the challenge of larger and richer datasets. Gigascience **4**, 1–16 (Dec 2015). https://doi.org/10.1186/s13742-015-0047-8
5. Ferrario, P.G., König, I.R.: Transferring entropy to the realm of GxG interactions. Briefings in Bioinformatics pp. 1–12 (Oct 2016). https://doi.org/10.1093/bib/bbw086
6. Genetic Analysis of Psoriasis Consortium, et al.: A genome-wide association study identifies new psoriasis susceptibility loci and an interaction between HLA-C and ERAP1. Nature Genetics **42**, 985–990 (Oct 2010). https://doi.org/10.1038/ng.694
7. Goudey, B., Rawlinson, D., Wang, Q., et al.: GWIS: Model-free, Fast and Exhaustive Search for Epistatic Interactions in Case-Control GWAS. Lorne Genome 2013 (February 2013)
8. Gyenesei, A., Moody, J., Semple, C.A., et al.: High-throughput analysis of epistasis in genome-wide association studies with BiForce. Bioinformatics **28**(15), 1957–1964 (May 2012). https://doi.org/10.1093/bioinformatics/bts304
9. Hu, X., Liu, Q., Zhang, Z., et al.: SHEsisEpi, a GPU-enhanced genome-wide SNP-SNP interaction scanning algorithm, efficiently reveals the risk genetic epistasis in bipolar disorder. Cell Res. **20**, 854–857 (May 2010)
10. Ibrahim, Z.M., Newhouse, S., Dobson, R.: Detecting epistasis in the presence of linkage disequilibrium: A focused comparison. 2013 IEEE Symp. CIBCB pp. 96–103 (Apr 2013). https://doi.org/10.1109/CIBCB.2013.6595394
11. Kam-Thong, T., Azencott, C.A., Cayton, L., et al.: GLIDE: GPU-Based Linear Regression for Detection of Epistasis. Human Heredity **73**, 220–236 (9 2012). https://doi.org/10.1159/000341885
12. Kässens, J.C., Wienbrandt, L., et al.: Combining GPU and FPGA technology for efficient exhaustive interaction analysis in GWAS. In: 2016 IEEE 27th Int. Conf. on ASAP. pp. 170–175 (Jul 2016). https://doi.org/10.1109/ASAP.2016.7760788
13. Kässens, J.C.: A Hybrid-parallel Architecture for Applications in Bioinformatics. No. 2017/4 in Kiel Computer Science Series, Department of Computer Science, CAU Kiel (2017). https://doi.org/10.21941/kcss/2017/4, dissertation, Faculty of Engineering, Kiel University.
14. Keaton, J.M., Hellwege, J.N., Ng, M.C.Y., et al.: Genome-wide Interaction with Selected Type 2 Diabetes Loci Reveals Novel Loci for Type 2 Diabetes in African Americans. Pac. Symp. Biocomput. **22**, 242–253 (Dec 2016). https://doi.org/10.1142/9789813207813_0024

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28

15. Kirino, Y., Bertsias, G., Ishigatsubo, Y., et al.: Genome-wide association analysis identifies new susceptibility loci for Behçet's disease and epistasis between HLA-B*51 and ERAP1. Nature Genetics **45**, 202–207 (Jan 2013). https://doi.org/10.1038/ng.2520
16. Lee, S., Kwon, M.S., Park, T.: CARAT-GxG: CUDA-Accelerated Regression Analysis Toolkit for Large-Scale Gene–Gene Interaction with GPU Computing System. Cancer Informatics **13s7**, CIN.S16349 (2014). https://doi.org/10.4137/CIN.S16349
17. van Leeuwen, E.M., Smouter, F.A.S., Kam-Thong, T., et al.: The Challenges of Genome-Wide Interaction Studies: Lessons to Learn from the Analysis of HDL Blood Levels. PLoS One **9**, e109290 (Oct 2014). https://doi.org/10.1371/journal.pone.0109290
18. Piriyapongsa, J., Ngamphiw, C., Intarapanich, A., et al.: iLOCi: a SNP interaction prioritization technique for detecting epistasis in genome-wide association studies. BMC Genomics **13**(Suppl 7), S2 (2012). https://doi.org/10.1186/1471-2164-13-s7-s2
19. Purcell, S., Chang, C.: PLINK v1.90p 64-bit (9 Jan 2018), www.cog-genomics.org/plink/1.9/
20. Purcell, S., Neale, B., Todd-Brown, K., et al.: PLINK: A Tool Set for Whole-Genome Association and Population-Based Linkage Analyses. American Journal of Human Genetics **81**, 559–575 (Sep 2007). https://doi.org/10.1086/519795
21. The Australo-Anglo-American Spondyloarthritis Consortium (TASC), et al.: Interaction between ERAP1 and HLA-B27 in ankylosing spondylitis implicates peptide handling in the mechanism for HLA-B27 in disease susceptibility. Nature Genetics **43**, 761–767 (Jul 2011). https://doi.org/10.1038/ng.873
22. Ueki, M., Cordell, H.J.: Improved Statistics for Genome-Wide Interaction Analysis. PLoS genetics **8**(4), e1002625 (2012). https://doi.org/10.1371/journal.pgen.1002625
23. Wan, X., Yang, C., Yang, Q., et al.: BOOST: A Fast Approach to Detecting Gene-Gene Interactions in Genome-wide Case-Control Studies. Am. J. Hum. Genet. **87**(3), 325–340 (2010)
24. Wang, Y., Liu, G., Feng, M., Wong, L.: An empirical comparison of several recent epistatic interaction detection methods. Bioinformatics **27**(21), 2936–2943 (2011)
25. Wienbrandt, L., Kässens, J.C., González-Domínguez, J., et al.: FPGA-based Acceleration of Detecting Statistical Epistasis in GWAS. Proc. Computer Science **29**, 220–230 (2014). https://doi.org/10.1016/j.procs.2014.05.020
26. Wienbrandt, L., Kässens, J.C., et al.: Fast Genome-Wide Third-order SNP Interaction Tests with Information Gain on a Low-cost Heterogeneous Parallel FPGA-GPU Computing Architecture. Proc. Computer Science **108**, 596–605 (2017). https://doi.org/10.1016/j.procs.2017.05.210
27. Xie, M., Li, J., Jiang, T.: Detecting genome-wide epistases based on the clustering of relatively frequent items. Bioinformatics **28**(1), 5–12 (Jan 2012)
28. Yung, L.S., Yang, C., Wan, X., et al.: GBOOST: a GPU-based tool for detecting gene-gene interactions in genome-wide case control studies. Bioinformatics **27**(9), 1309–1310 (2011)

ICCS Camera Ready Version 2018
To cite this paper please use the final published version:
DOI: 10.1007/978-3-319-93701-4_28