Procedia Computer Science

Volume 80, 2016, Pages 639–649

ICCS 2016. The International Conference on Computational Science

Genome-Wide Association Interaction Studies with MB-MDR and maxT multiple testing correction on FPGAs

Sven Gundlach[1], Jan Christian Kässens[2], and Lars Wienbrandt[2]

[1] Institute of Medical Informatics and Statistics, University Medical Center Schleswig-Holstein, Campus Kiel, Germany
gundlach@medinfo.uni-kiel.de

[2] Department of Computer Science, Christian-Albrechts-University of Kiel, Germany
{jka,lwi}@informatik.uni-kiel.de

Abstract

In the past few years massive amounts of data have been generated for genetic analysis. Existing solutions to analyze this data concerning genome-wide gene interactions are either not powerful enough or can barely be managed with standard computers due to the tremendous amount of statistical tests to be performed. Also, common approaches using cluster or cloud technologies for parallel analysis are operating at the edge of what is currently possible.

This work demonstrates how FPGAs are able to address this problem. We present a highly parallel, hardware oriented solution for genome-wide association interaction studies (GWAIS) with MB-MDR and the maxT multiple testing correction on an FPGA-based architecture. We achieve a more than 300-fold speedup over an AMD Opteron cluster with 160 cores on an FPGA-system equipped with 128 Xilinx Spartan6 LX150 low-cost FPGAs when analyzing a WTCCC-like dataset with 500,000 markers and 5,000 samples. Furthermore, we are able to keep pace with a 256-core Intel Xeon cluster running MB-MDR 4.2.2 with an approximative version of maxT, while we achieve a 190-fold speedup over the sequential execution of this version on one Xeon core.

Keywords: GWAIS, epistasis, SNPs, Chi-squared test, FPGA, reconfigurable hardware, high-performance computing, bioinformatics

1 Introduction

The detection of gene-gene interactions, such as epistasis, has become a major topic in molecular and quantitative genetics. Genome-wide association studies (GWAS) are conducted in order to reveal the influence of genetic markers such as single-nucleotide polymorphisms (SNPs) on complex genetic disease traits [13]. With the current genotyping technologies allowing the collection of millions of markers per sample, large datasets can be created. The association of the samples to a specific phenotype, which can be as simple as a binary trait, i.e. either case or

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.354

control, allows statistical testing for the correlation of genetic characteristics to the trait. Since testing for a correlation of single markers to a trait is not powerful enough in most cases [14, 15], biologists consider at least a pairwise or even higher-order post-GWAS analysis called genome-wide association interaction studies (GWAIS). As GWAS can only identify single markers that have significant main effects, but markers that require interactions with other markers to show their association with the phenotype are not detected by GWAS if their main effects are small [9]. This introduces a high computational burden since the number of tests to perform increases exponentially in the order of the analysis, i.e. a quadratic number for pairwise interactions and a cubic number for 3rd-order interactions. In particular, $\binom{n}{2} = \frac{n(n-1)}{2}$ tests have to be performed for an exhaustive pairwise interaction analysis, if n is the number of markers. This already implies 500 billion tests for a dataset with 1 million markers.

Many tools have emerged in order to address this severe computational problem, e.g. BOOST [22], iLOCi [16] and TEAM [26] which run on standard computers or clusters. Other tools addressing GPU technology in order to significantly speedup the computation process are available, e.g. GBOOST [25], GWIS [6], epiGPU [10] and SHEsisEPI [11]. Some of those tools were already implemented on FPGAs, such as BOOST [5] and iLOCi [24]. Also, third-order interactions can already be analyzed in reasonable time with GPU or FPGA technology [4, 12].

However, most of the tools provide a fast test for interactions, but do not consider the problem that for a large number of tests the number of expected false positive errors (i.e. type I errors) is very high [7]. BiForce [8] adresses this problem by using Bonferroni correction [3] which is however not powerful enough for family-wise error rate (FWER) correction [20]. When the MDR tool emerged, the creators considered controlling the error rate by permutations of the trait [17]. This behaviour was inherited by its extension MB-MDR [2]. MB-MDR categorizes the characteristics of SNP combinations into three groups of potential association: *high risk*, *low risk* and *no evidence*. The statistical test, that can be a simple Chi-squared test, is performed multiple times in a number of iterations with a randomly permuted trait. The p-value of the tests is then adjusted using the maxT multiple testing correction method [23].

In MB-MDR 3.0.3 van Lishout et al. presented an efficient implementation of the maxT method [21]. However, due to a large number of necessary iterations, e.g. 1,000, and the large number of tests in each iteration, the runtime of MB-MDR 3.0.3 is not reasonable unless executed on a high-performance computing system such as a CPU-cluster. Thus, we address this problem with FPGA-technology. In the following, we demonstrate how our implementation of MB-MDR 3.0.3 on an FPGA-system with 128 Xilinx Spartan6 FPGAs outperforms an AMD Opteron cluster with 160 cores with a speedup of more than 300.

In the meantime, the same problem has been addressed by van Lishout et al. themselves. They re-designed the maxT method, now referred to as *gammaMaxT*, by approximating the necessary maximum operation in each iteration. In MB-MDR 4.2.2 [20] they have recently reached an up to 10^4-fold speedup compared to their old method with an acceptable trade-off in result quality. We show that we are still able to keep pace with MB-MDR 4.2.2 and the gammaMaxT method executed on an Intel Xeon cluster with 256 cores with the analysis of a dataset with about 500,000 markers on 5,000 samples such as those provided by the Wellcome Trust Case Control Consortium (WTCCC) [19]. Still, the speedup of our method over the sequential execution of this method is 190-fold.

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

control		SNP A w	SNP A h	SNP A v
SNP B	w	U_{00}	U_{01}	U_{02}
	h	U_{10}	U_{11}	U_{12}
	v	U_{20}	U_{21}	U_{22}

HLO matrix		SNP A w	SNP A h	SNP A v
SNP B	w	R_{00}	R_{01}	R_{02}
	h	R_{10}	R_{11}	R_{12}
	v	R_{20}	R_{21}	R_{22}

Figure 1: Contingency tables and corresponding HLO-matrix for a pair of markers *SNP A* and *SNP B*. A_{ij} and U_{ij} reflect the number of affected and unaffected samples with the respective combination of genotypes i, j. R_{ij} indicates if samples with markers i for *SNP B* and j for *SNP A* have a high statistical (H) or a low statistical (L) risk or if there is no statistical evidence (O).

2 MB-MDR with *maxT* Multiple Testing Correction

The *Model-Based Multifactor Dimensionality Reduction (MB-MDR)* analysis has first been described by Calle et al. [1, 2] to detect gene-gene interactions associated with diseases. It is a model-based version of the MDR [17] method with the main idea to merge multi-locus genotypes into categories in order to increase the detection power. Our FPGA-based implementation presented in this paper is based on the performance oriented optimization MB-MDR 3.0.3 by van Lishout et al. [21]. We concentrate on the detection of pairwise gene-gene interactions on biallelic markers in case-control datasets (i.e. binary traits), which is a common use case [19]. The method consists of three key steps described in the following sections, which lead to a fast separation of highly significant and low significant interactions.

2.1 Contingency Tables and HLO-Matrices

In the first step, two two-dimensional contingency tables are created for each pair of genetic markers in the input dataset, one for all case samples and one for the controls. They are also referred to as *affected-subjects matrix* A and *unaffected-subjects matrix* U. Since we assume biallelic markers, three possible genotypes are available, the homozygous wild type (w), the heterozygous type (h) and the homozygous variant type (v). Thus, the contingency tables have the dimensions 3×3. Each cell in a table reflects the number of samples of either cases (affected subjects) or controls (unaffected subjects) which carry the respective combination of genetic information at the corresponding pair of markers. For example, the entry A_{01} of the affected-subjects matrix generated from a pair (x, y) of genetic markers indicates the number of cases in the input dataset, which carry the homozygous wild genotype at marker x and the heterozygous type at marker y. The structure of contingency tables for cases and controls is depicted in Fig. 1.

With the information in the affected-subjects and unaffected-subjects matrices an *HLO-matrix* R is generated for each pair of markers. Again, this matrix has dimensions 3×3. In each cell it contains the category in which the corresponding genotype combination can be classified, i.e. H for a high risk that a person that carries these genotypes may be affected, L for a low risk, and O to indicate the absence of evidence for the affection status. The categorization is done following the strategy in [21].Let A_{pq} be a cell of the affected-subjects matrix and U_{pq} a cell of the unaffected-subjects matrix, and let N_A and N_U be the total number of affected and unaffected samples respectively.

Firstly, it is tested if $A_{pq} + U_{pq} < 10$ or $N_A + N_U - A_{pq} - U_{pq} < 10$. In this case, there is not enough data to conclude a risk factor and hence, R_{pq} is set to O. Otherwise, a test statistic is calculated as follows with $a = A_{pq}$, $b = U_{pq}$, $c = N_A - A_{pq}$ and $d = N_U - U_{pq}$.

$$T = \frac{(ad - bc)^2(a + b + c + d)}{(a + b)(c + d)(b + d)(a + c)} \tag{1}$$

This test statistic follows a χ^2-distribution, and thus, if the significance falls below a threshold of 0.1 there is again no evidence for the risk status and R_{pq} will be set to O as well. In the opposite event, R_{pq} is set to H indicating a high risk if $(ad - bc) > 0$, and it is set to L if $(ad - bc) \leq 0$.

2.2 Test Statistic

In the second step, the affected-subjects and unaffected-subject matrices together with the HLO-Matrix are used to calculate a test statistic $T_{0,j}$ for each pairwise combination of markers. Only the best n values are sorted and kept in a *Real Data* vector $\langle T_{0,1} \cdots T_{0,n} \rangle$ such that $T_{0,1} \geq T_{0,2} \geq \cdots \geq T_{0,n}$. n is set to 1,000 per default.

The implemented test statistic $T_{0,j}$ is the maximum of two χ^2-tests with one degree of freedom. The first one H/LO calculates the association between the trait and being a member of the H category rather than the L or O category, while the second one L/HO computes the association between the trait and the belonging to the L category rather than the H or O category. Both tests are exactly implemented as in Eq. 1, with the following redefinition of the values a, b, c, and d. a states the number of affected samples in the H category while b states the number of unaffected samples in that category, c and d are respectively the number of affected and unaffected samples belonging to one of the other categories. This applies respectively for the second test with a and b denoting the number of affected and unaffected samples in the L category.

2.3 maxT Multiple Testing

In order to obtain the p-values for the interaction effect according to the calculated test statistic, multiple correlated tests are performed by applying the *maxT* method [23]. The number of iterations B (i.e. the number of tests) can be user-defined while a higher number of iterations leads to more precise p-values. B is set to 1,000 iterations per default. maxT is implemented based on the optimized method presented by van Lishout et al. in [21].

After calculating the n-best test statistics in the *Real Data* vector in the previous step, B random permutations of the trait are generated. Afterwards, the test statistics $T_{i,j}$ are calculated according to Sect. 2.2 for each permutation i. Thereby, the test statistics are not sorted, but forced into the order of the *Real Data* vector. The vectors $\langle T_{i,1} \cdots T_{i,n} \rangle$ for $1 \leq i \leq n$ are referred to as $Permutation_i$ vectors. In order to reduce type II errors, the monotonicity of each permutation vector is enforced, i.e. $T_{i,n}$ is replaced by the maximum of all test statistics $T_{i,n+1}, \ldots, T_{i,m}$ with m being the number of all possible marker pairs. Furthermore, for each $n > j \geq 1$: $T_{i,j}$ is replaced by $T_{i,j+1}$ if $T_{i,j} < T_{i,j+1}$.

The p-values p_j can now simply be obtained by counting the number of permutations where $T_{i,j} \geq T_{0,j}$ and normalized by the number of iterations.

$$p_j = \frac{|S_j|}{B + 1} \quad \text{with } S_j = \{i \,|\, T_{i,j} \geq T_{0,j}, 0 \leq i \leq B\} \tag{2}$$

The monotonicity of the p-value vector is enforced as well in order to keep type I errors low. Thus, p_{j+1} is replaced by p_j if $p_{j+1} < p_j$.

3 Implementation Details

The basic structure of our design is a systolic chain architecture of processing elements (PEs) for the creation and transfer of contingency tables adapted from [24] and [5]. We modified the PEs and divided the chain structure as described in Sect 3.1 to fit our requirements. The calculations and the program flow is based on MB-MDR 3.0.3 presented in [21].

Six main tasks are executed in multiple iterations on multiple FPGA workers controlled by one host. Firstly, genotypes from the dataset are collected for each marker in *genotype vectors*, which are then distributed to each worker FPGA. The distribution exactly follows the one described in [5]. After each FPGA has stored its part of the genotype vectors in its local memory, the data is streamed vector-wise from the memory unit through the systolic chain of PEs that continuously create the respective contingency tables for cases and controls. As soon as tables are ready, they are forwarded to a pipelined analysis unit that for each pair of case and control tables computes the test statistic as the maximum of two χ^2-tests according to Eq. 1. This process is described in Sect. 3.2. The n best test results (n =1,000 per default) are stored in local memory and eventually returned to the host in the first iteration. In all other iterations the previously stored results are updated and forced into monotonicity as described in Sect. 3.3. After the last iteration the p-values for the n best results are calculated on the host system.

3.1 Creating and Transferring Contingency Tables

Our systolic chain architecture of processing elements, as it is illustrated in Fig. 2, requires the genotype vectors to be sorted by cases and controls, which is already done intuitively while parsing the input data. Each PE contains nine counters, one for each entry of a contingency table, and a local memory that is able to store a complete genotype vector for one marker. Genotypes are streamed vector by vector with two genotypes in one clock cycle. For each PE in the chain its first genotype vector from the stream is stored in its local memory and not forwarded to the next PE, this is only done for all following vectors. During the streaming process, the PEs compare the genotypes from the stream to their locally stored ones and increment the corresponding counters of the contingency table according to the observed genotype combination. After the cases of one genotype vector are complete, all PEs have finished contingency tables for the affected subjects, which are immediately forwarded through the chain by a *transport bus* to a *table buffer*. The same applies respectively for the unaffected subjects (controls) afterwards. As soon as a pair of tables is ready in the buffer, it is forwarded to the analysis unit. For more details we refer to [24] and [5].

To prevent the transport bus from overloading, requirements to the input dataset have to be made. The bus is able to transfer one of the nine counter values of a table in one clock cycle. Meanwhile, two genotypes of input data can be streamed. Thus, in order to ensure the table has been transferred before the next one is ready, the following condition has to be met.

$$\min\{\text{no. cases}, \text{no. ctrls}\} \geq \text{genotypes/cycle} \times \text{no. counters} \times \text{no. PEs} \tag{3}$$

For several datasets this condition might quickly become too restrictive. Thus, we introduced the possibility to divide the chain of processing elements into multiple parts each featuring its own analysis unit, as it is shown in Fig. 2. For example, in Sect. 4 we evaluate a design with two chains and 66 PEs per chain. According to Eq. 3 the design is able to analyze datasets with at least 1,188 cases and controls.

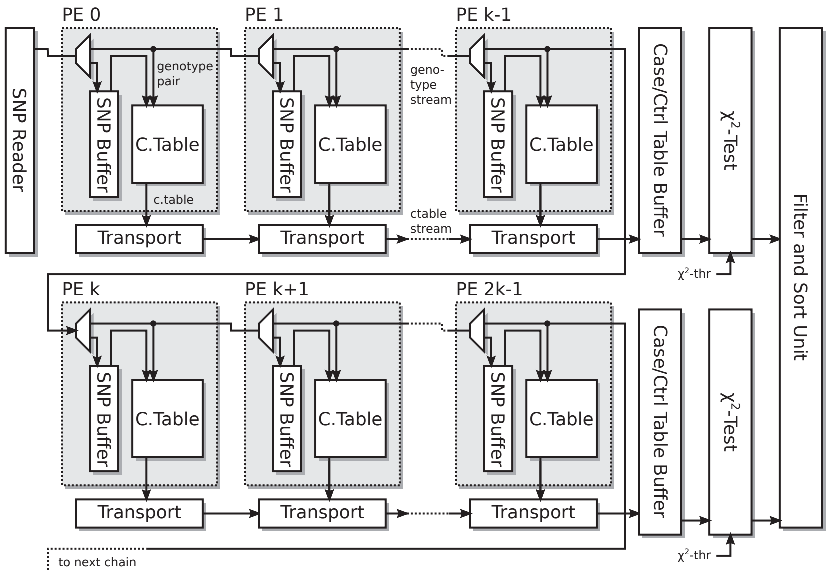

Figure 2: Systolic chain structure of processing elements with multiple chains.

3.2 Chi-Squared Pipeline

The generation of the HLO-matrix for each pair of markers requires the application of a χ^2-test with one degree of freedom to each combination of genotypes, which is computed by one χ^2-test unit sequentially behind the table buffer. The parameters for each test are taken from the contingency tables for the affected and unaffected subjects according to Eq. 1 in Sect. 2.1. The implementation of the test is fully pipelined such that a computation can start within every clock cycle. Afterwards, the HLO-matrix is generated by deciding whether each genotype combination belongs to the high-risk (H), low-risk (L) or no-evidence (O) category.

The test statistic is then computed via the maximum of two further χ^2-tests H/LO and L/HO according to Sect. 2.2. The required parameters are generated during the creation of the HLO-matrix by the following summations.

$$a_{H/LO} = \sum \{A_{pq} \,|\, (ad-bc) > 0\} \qquad a_{L/HO} = \sum \{A_{pq} \,|\, (ad-bc) \leq 0\} \tag{4}$$

$$b_{H/LO} = \sum \{U_{pq} \,|\, (ad-bc) > 0\} \qquad b_{L/HO} = \sum \{U_{pq} \,|\, (ad-bc) \leq 0\} \tag{5}$$

$$c_{H/LO} = N_A - a_{H/LO} \qquad c_{L/HO} = N_A - a_{L/HO} \tag{6}$$

$$d_{H/LO} = N_U - b_{H/LO} \qquad d_{L/HO} = N_U - b_{L/HO} \tag{7}$$

The computation of the tests are started in two consecutive clock cycles in a second fully pipelined χ^2-test unit. Thus, the two results are available after the pipeline latency in two consecutive clock cycles as well, and the maximum of both tests can directly be determined after a simple comparison. The χ^2-test chain with both test units and the HLO-matrix generation is illustrated in Fig. 3.

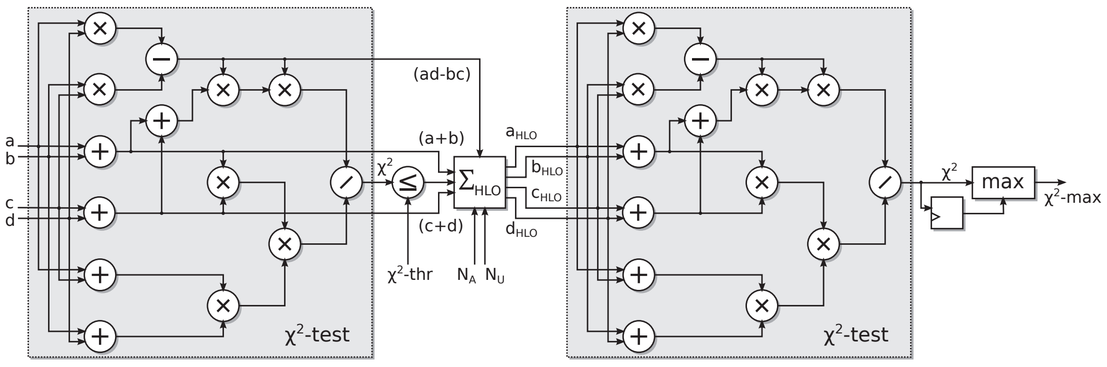

Figure 3: Scheme for the χ^2-test chain with two pipelined χ^2-test units. Note, that most registers have been omitted for readability.

3.3 Sorting Results and Shuffling

In order to store the n best results of the first iteration, we decided to use a simple insertion sort with linear runtime in local FPGA memory. Since we have to prepare for results arriving in consecutive clock cycles, we immediately compare the results to the current smallest element in the n-best list and keep only those results with a higher test value in a preliminary buffer before finally inserting it into the list.

We keep the n best test values together with the corresponding marker pair identifiers. However, to prepare for the upcoming permutation iterations the same identifiers are stored again in a separate list sorted in the ascending order of their generation. This way, the test statistics could easily be updated at the end of each iteration according to Sect. 2.2 as the marker pairs are evaluated in the same order in each iteration. Furthermore, the maximum value of all pairs that are not in the n-best list must be stored to enforce the monotonicity of the vector correctly.

Since all worker FPGAs compute their own local n-best list, the results have to be merged on the host, where the p-values are calculated as well. Furthermore, the host is responsible to provide the permuted data to the FPGAs at the beginning of each iteration. The permutation for the next iteration is therefore prepared on the host concurrently to the running analysis of the current iteration on the FPGAs. Each permutation randomly swaps the genotype data of a number of case and control samples such that some samples that appeared as a case now act as control and vice versa. Note, that swapping the samples does not change the amount of cases and controls.

4 Performance Evaluation

The evaluation of our FPGA-based MB-MDR implementation targets a SciEngines RIVYERA S6-LX150 machine [18] equipped with 128 Xilinx Spartan6 LX150 low-cost FPGAs. The built-in host system features an Intel Core i7-950 quad-core processor at 3GHz and 12GB DRAM, and runs a Linux OS. The FPGA resources allowed us to implement 132 processing elements distributed over two chains and clocked at 125MHz on each FPGA. We compared the performance of our design to the results of MB-MDR 3.0.3 in [21] and MB-MDR 4.2.2 in [20]. The test systems were respectively a 160-core computer cluster consisting of 40 AMD Opteron 2352

# SNPs	MB-MDR 3.0.3 sequential	MB-MDR 4.2.2 sequential	MB-MDR 3.0.3 160 cores	MB-MDR 4.2.2 256 cores	128 Spartan6 FPGA/54 PEs
10,000	5d13h	52m15s	1h03m	1m05s	10m43s
100,000	$\sim$ 1.5y*	2d16h35m	4d09h	22m15s	1h01m49s
1,000,000	-	$\sim$ 270d*	1y73d*	1d01h12m	3d09h23m

Table 1: Runtimes of FPGA-based MB-MDR with 54 PEs per FPGA for datasets with 1,000 samples and a varying number of markers compared to MB-MDR 3.0.3 on a 160-core Opteron cluster from [21] and to MB-MDR 4.2.2 on a 256-core Xeon cluster from [20]. Values marked with an asterisk (*) are linearly extrapolated.

# SNPs	MB-MDR 3.0.3 sequential	MB-MDR 4.2.2 sequential	MB-MDR 3.0.3 160 cores	MB-MDR 4.2.2 256 cores	128 Spartan6 FPGA/132 PEs
10,000	$\sim$ 19d	2h11m	2h38m	2m43s	9m30s
100,000	$\sim$ 3.75y	6d17h	10d23h	55m38s	1h08m28s
1,000,000	-	1y310d	$\sim$ 3y	2d15h	3d12h20m

Table 2: Runtimes of FPGA-based MB-MDR with 132 PEs per FPGA for datasets with 2,500 samples and a varying number of markers compared to linearly extrapolated results of MB-MDR 3.0.3 on a 160-core Opteron cluster from [21] and MB-MDR 4.2.2 on a 256-core Xeon cluster from [20].

quad-core CPUs at 2.1 GHz, and a 256-core computer cluster consisting of 64 Intel Xeon L5420 quad-core CPUs running at 2.5 GHz.

For a direct comparison, we generated several SNP datasets with 1,000 samples distributed over 500 cases and 500 controls, and a varying number of markers between 10,000 and 1,000,000 SNPs. Since our design is intentionally designed to target larger datasets, it was not able to use its full resources on such a low number of samples, but had to run with only 54 processing elements per FPGA for these tests. Therefore, in order to measure its full performance, we generated similar datasets with an increased number of 2,500 samples (1,250 cases and 1,250 controls). The runtimes are listed in Tables 1 and 2 including results for a sequential run on only one core.

In Table 3 we also investigated a dataset with about 500,000 genetic markers at 5,000 samples distributed over 2,000 cases and 3,000 controls such as those provided by the Wellcome Trust Case Control Consortium (WTCCC) [19]. This example was used to compare the energy

Method	Runtime	System power	Energy consumption
MB-MDR 3.0.3 (160 cores)	$\sim$ 1.5y	40×75W = 3,000W	39,420.0 kWh
MB-MDR 4.2.2 (256 cores)	1d05h52m	64×50W = 3,200W	102.4 kWh
128 Spartan6 (FPGA/132 PEs)	1d19h08m	468W	20.2 kWh

Table 3: Runtimes and energy consumption of FPGA-based MB-MDR for a WTCCC-like dataset with about 500,000 markers and 5,000 samples compared to MB-MDR 3.0.3 on a 160-core Opteron cluster and to MB-MDR 4.2.2 on a 256-core Xeon cluster. Runtimes for software MB-MDR are interpolated via a regression model from the results in [21] and [20] respectively.

consumption of the test systems as well. The power consumption of the FPGA-system has been measured directly with the internal power supply via IPMI interface, and the power of the other systems has been calculated according to the CPU specifications alone.

The results show, that the FPGA-system clearly outperforms the former version of MB-MDR with the exact calculation of the maxT method although it was run on an Opteron cluster system with 160 cores. The speedup is more than 300 on the dataset with 1 million markers and 2,500 samples. However, for the datasets with only 1,000 samples, we still outperform the Opteron cluster with a speedup of about 130, but the Xeon cluster system with 256-cores running the newer version of MB-MDR is about three to ten times faster than our system. One reason is that the maxT maximum approximation in this version provides a significant speedup over the former version, and another reason is that the FPGAs are forced to use only 54 processing elements due to the low number of samples. Using our full performance with 132 PEs on the datasets with 2,500 samples and the WTCCC-like dataset, we are almost able to catch up with the new version on the Xeon cluster while gaining a speedup of more than 140 to 190 when compared to the sequential execution on the datasets with 100,000 and 1 million markers respectively. We find this a significant result since it shows that a number of low-cost FPGAs is able to keep pace with a CPU cluster employing twice as many computing cores running an approximative computation while our calculation is exact according to the former version of MB-MDR. Furthermore, the measured energy consumption of the FPGA system is five times lower than the energy required of the Xeon CPUs alone without considering peripherals, infrastructure, cooling etc.

5 Conclusion

The statistical detection of gene-gene interactions requires a suitable statistical test and a proper calculation of the significance of the test results as well. This is generally acquired by calculating the p-value of a test result using multiple testing correction methods. In addition to previous publications [5, 12, 24] that have proven FPGAs to be powerful on detecting gene interaction candidates with tools such as BOOST [22] or iLOCi [16], this paper shows that multiple testing correction methods for gene interaction detection can be extremely powerful on FPGAs as well. We demonstrated this by implementing MB-MDR with maxT multiple testing correction on 128 Xilinx Spartan6 LX150 low-cost FPGAs. Our design was based on the fast maxT algorithm presented in [21] and implemented in MB-MDR 3.0.3. The performance evaluation revealed a magnificent speedup of more than 300 when compared to this version of MB-MDR running on a 160-core Opteron CPU cluster system. Furthermore, the presented implementation is able to keep up with a 256-core Xeon CPU cluster running the newer version MB-MDR 4.2.2 [20] that significantly reduces its computation time by approximating the maximum calculation in the maxT iterations. However, the speedup compared to the sequential execution of MB-MDR 4.2.2 is still up to 190, and the energy consumption compared to the Xeon cluster is reduced to at least one fifth.

For future work investigations have already been started to analyze the abilities of our FPGA system to reduce the runtime by maximum approximation with the gammaMaxT method as in MB-MDR 4.2.2 [20]. As stated in that publication the slight loss in precision is acceptable if the runtime can be significantly reduced. This is achieved by reducing the amount of association tests to compute the maximum T_{max} of all test statistics that are less than the n best values. Despite that MB-MDR 3.0.3 only computes adjusted p-values for the n best test statistics, it still computes the total amount of m association tests to determine T_{max}. Therefore, the further development of MB-MDR 4.2.2 includes the estimation of T_{max} on the basis of only

10^6 test statistics, which is far less than e.g. $\approx 5 * 10^9$ test statistics needed for $100,000$ SNPs in MB-MDR 3.0.3. In order to map this easily onto FPGAs the estimation of T_{max} could be done by the host, while the computation of the association tests would be done by the FPGAs as before. The main bottleneck would then be the efficient communication between the host and the FPGA workers, as sending the genotypes from the dataset to the FPGA workers and returning the test statistics to the host would cost more time than the association tests itself. To avoid this bottleneck the permutation of the dataset as well as the estimation of T_{max} would have to be carried out by the FPGA workers as well. This would allow the FPGA workers to independently work through the iterations without waiting for the host. Hence, the p-values would still be computed by the host, but would not stall the computation of the association tests.

Thus, by investigating these approaches we look forward to be able to outperform a modern CPU cluster system with a future version of our FPGA-based MB-MDR, and we expect another fundamental speedup by porting the design to more recent FPGA technology such as Xilinx 7-series and UltraScale FPGAs.

References

[1] M. L. Calle, V. Urrea, N. Malats, and K. van Steen. MB-MDR: Model-Based Multifactor Dimension Reduction for detecting interactions in high-dimensional genomic data. *Ann. Hum. Genet.*, 75, Jan. 2007.

[2] T. Cattaert, M. L. Calle, S. M. Dudek, 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.

[3] O. J. Dunn. Multiple Comparisons Among Means. *J. Am. Stat. Assoc.*, 56(293):52–64, Mar. 1961.

[4] J. González-Domínguez and B. Schmidt. GPU-accelerated exhaustive search for third-order epistatic interactions in casae-control studies. *Journal of Computational Science*, 8:93–100, 2015.

[5] J. González-Domínguez, L. Wienbrandt, J. C. Kässens, et al. Parallelizing Epistasis Detection in GWAS on FPGA and GPU-accelerated Computing Systems. *IEEE/ACM Trans. Comput. Biol. Bioinf.*, 12:982–994, Jan. 2015.

[6] B. Goudey, D. Rawlinson, Q. Wang, et al. GWIS: Model-free, Fast and Exhaustive Search for Epistatic Interactions in Case-Control GWAS. Lorne Genome 2013, February 2013.

[7] E. S. Gusareva and K. van Steen. Practical aspects of genome-wide asociation interaction analysis. *Hum. Genet.*, 133:1343–1358, Nov. 2014.

[8] A. Gyenesei, J. Moody, C. A. Semple, et al. High-throughput analysis of epistasis in genome-wide association studies with BiForce. *Bioinformatics*, 28(15):1957–1964, May 2012.

[9] T. H. Hamza, H. Chen, E. M. Hill-Burns, et al. Genome-wide gene-environment study identifies glutamate receptor gene grin2a as a parkinson's disease modifier gene via interaction with coffee. *PLoS Genet*, 7(8):e1002237, 2011.

[10] G. Hemani, A. Theocharidis, W. Wei, et al. EpiGPU: exhaustive pairwise epistasis scans parallelized on consumer level graphics cards. *Bioinformatics*, 27(11):1462–1465, Apr. 2011.

[11] X. Hu, Q. Liu, Z. Zhang, 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.

[12] J. C. Kässens, L. Wienbrandt, J. González-Domínguez, et al. High-Speed Exhaustive 3-locus Interaction Epistasis Analysis on FPGAs. *Journal of Computational Science*, 9:131–136, 2015.

[13] R. J. Klein, C. Zeiss, E. Y. Chew, et al. Complement factor H polymorphism in age-related macular degeneration. *Science (New York, N.Y.)*, 308(5720):385–9, Apr. 2005.

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

[15] J. H. Moore, F. W. Asselbergs, and S. M. Williams. Bioinformatics Challenges for Genome-Wide Association Studies. *Bioinformatics*, 26(4):445–455, 2010.

[16] J. Piriyapongsa, C. Ngamphiw, A. Intarapanich, et al. iLOCi: a SNP interaction prioritization technique for detecting epistasis in genome-wide association studies. *BMC Genomics*, 13(Suppl 7):S2+, 2012.

[17] M. D. Ritchie, L. W. Hahn, N. Roodi, et al. Multifactor-Dimensionality Reduction Reveals High-Order Interactions among Estrogen-Metabolism Genes in Sporadic Breast Cancer. *Am. J. Hum. Genet.*, 69(1):138–147, May 2001.

[18] SciEngines GmbH. `http://www.sciengines.com`.

[19] 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.

[20] F. van Lishout, F. Gadaleta, J. H. Moore, et al. gammaMAXT: a fast multiple-testing correction algorithm. *BioData Mining*, 8(36), Nov. 2015.

[21] F. van Lishout, J. J. M. Mahachie, E. S. Gusareva, et al. An efficient algorithm to perform multiple testing in epistasis screening. *BMC Bioinf.*, 14(1):138, 2013.

[22] X. Wan, C. Yang, Q. Yang, 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.

[23] P. H. Westfall and S. S. Young. *Resampling-Based Multiple Testing: Examples and Methods for p-Value Adjustment*. John Wiley & Sons, 1 edition, Jan. 1993.

[24] L. Wienbrandt, J. C. Kässens, J. González-Domínguez, et al. FPGA-based Acceleration of Detecting Statistical Epistasis in GWAS. *Procedia Computer Science*, 29:220–230, 2014.

[25] L. S. Yung, C. Yang, X. Wan, et al. GBOOST: a GPU-based tool for detecting gene-gene interactions in genome-wide case control studies. *Bioinformatics*, 27(9):1309–1310, 2011.

[26] X. Zhang, S. Huang, F. Zou, et al. TEAM: efficient two-locus epistasis tests in human genome-wide association study. *Bioinformatics*, 26(12):i217–i227, July 2010.