FPGA-based Accelerator for High-order Epistasis Detection

Gaspar Minderico Ribeiro
Instituto Superior Técnico
Av. Rovisco Pais 1, Lisboa, Portugal
gasparmribeiro@tecnico.ulisboa.pt

***Abstract*—The classic approach to Genome-Wide Association Studies (GWAS) attempts to find a relation between one or more Single Nucleotide Polymorphisms (SNPs) contributing to the manifestation of a certain disease or trait. However, most genetic diseases do not depend on the individual effect of one or more SNPs, but rather on the interactions between several SNPs, also known as epistasis. Detecting epistasis for high-order interactions results in a huge computational complexity, as the number of SNP combinations evaluated exponentially grows with the order of the interactions. For this reason, state-of-the-art exhaustive search-based methods for epistasis detection rely on Graphics Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs) to provide high-performance solutions for second-order interactions. Nevertheless, only rare attempts are made to tackle third-order interactions, and the ones that exist are not scalable for higher-order interactions and could be further optimized.**

In this thesis, the flexibility and scalability limitations of the existing FPGA implementations are addressed, while a novel parameterizable architecture is proposed that enables the deployment of FPGA-based accelerators targeting any order of interactions in modern FPGAs. The resulting implementations can target any dataset size and provide higher performance and energy efficiency than the state-of-the-art FPGA-based architectures. In particular, performance gains of up to 80x and 3x are achieved when compared to existing second and third-order implementations, respectively, while also providing average energy efficiency improvements of 79x. Finally, the proposed architecture allowed for the implementation of a fourth-order epistasis detection accelerator in an FPGA platform.

***Index Terms*—Genome-Wide Association Studies, Epistasis Detection, FPGA Accelerator**

I. Introduction

During the past two decades, Genome-Wide Association Studies (GWAS) have been used to find a relationship between genetic variations and the manifestation of a phenotype, such as a disease or trait [1]. Typically, GWAS attempt to find an association between one or more single-nucleotide polymorphism (SNP) (which is the most common genetic variation in the human genome), with a certain disease. This association is usually performed via an observational case-control study, by comparing several SNPs from a group of patients that suffer from the disease (cases), with a group of people that do not suffer from the disease (controls). This approach has been used with variable degree of success, in identifying genetic markers associated with some diseases, such as age-related macular degeneration [2]. However, some more complex genetic diseases, such as type II diabetes [3], are not possible to explain using the traditional GWAS approach. This happens largely because gene-gene interactions and environmental factors are ignored when following a classic GWAS approach [4]. The effect of the interaction between multiple SNPs in the manifestation of a certain phenotype is known as epistasis [5].

Epistasis detection has been successfully used to study several complex diseases such as Late-Onset Alzheimer's Sisease [6] or asthma [7]. However, epistasis detection is a highly computationally intensive task, as the number of combinations to be tested increases exponentially with the order of interactions (almost 200 million pair-wise combinations exist in a dataset with 20 000 SNP, but the same dataset contains more than 1.3 trillion and 6.6 quadrillion third and fourth-order combinations, respectively). Due to that relation, most exhaustive-based methods for detecting epistasis (in which all the possible combinations are individually evaluated) are typically limited to second-order interactions, since performing higher-order epistasis detection over a larger dataset often results in an infeasible execution time.

In addition to methods based an exhaustive search, other methods attempt to reduce the computational burden by reducing the search space [8], therefore only evaluating a fraction of the SNPs available in the dataset. However, as the pre-filtering stage might filter out SNPs that are relevant to the phenotype, those methods are generally less accurate than exhaustively search all possible combinations.

Due to the mentioned computational complexity, most epistasis detection implementations often only target second-order or, at most, third-order interactions. However, detecting higher-order epistatic interactions for complex traits is absolutely crucial as they are likely to have severe implications on certain phenotypes [9]. As such, the need arises for better implementations that take advantage of the technological advances in the hardware field to overcome the computation burden associated with performing higher-order epistasis detection based on an exhaustive search over large datasets.

Methods to detect epistasis based on software running on multi-core CPUs are mostly inefficient, due to their limitations in exploiting the existent data parallelism. For example, processing a dataset with 500 000 SNPs on 2 quad-core Intel Xeon processors takes 19 hours for second-order epistasis [10]. On the other hand, thanks to their high core count, GPUs are efficient in exploring the data-level parallelism, as multiple operations can be executed in parallel [11]. As such, GPUs have been proven effective to perform second-order

[12], and even third-order in over relatively big datasets in a reasonable amount of time [12], [13]. However, not many attempts to performing fourth-order epistasis detection based on exhaustive search have been made, due to the massive amount of combinations to be tested even for a dataset with just a few thousand SNPs.

Alternatively, due to their reconfigurability, FPGAs can be used to implement a logic circuit that takes advantage of the data-parallelism existent in epistasis detection. Although FPGAs have been successfully used to implement custom accelerators in a wide range of bioinformatics applications [14], [15], such as in sequence alignment [16], [17]. Few attempts deploying accelerators for epistasis detection in FPGAs have been proposed. In particular, second [18] and third-order [19] implementations showed promising results, although they are not easily scalable for higher-order interactions, only work for a limited range of patients, and offer limited performance and energy efficiency. Improvements to the FPGA technology made in the last decades [20], suggest that, in the future, higher-order exhaustive search epistasis detection could be performed in FPGAs. However, due to the lack of scalability with the order of interaction of the existent implementations and complexity of hardware design, the need for a new method to deploy efficient custom architectures to perform higher-order epistasis detection in FPGAs arises. As such, this paper proposed a novel general architecture that can be used to deploy FPGA-based accelerators targeting any order of interactions, and datasets of any size. The generated accelerators also provide higher performance ($3\times$ faster than [19]) and energy efficiency ($79\times$ better than [18]).

II. Background on Genetics and Epistasis Detection

In order to understand epistasis, it is fundamental to realize the basis of genetics and hereditary. All the genetic information of a living being is encoded in the Deoxyribonucleic acid (DNA). The DNA is a molecule composed of two intertwined polynucleotide chains creating a double helix. Each polynucleotide chain is formed by a group of nucleotides, linked together by covalent bonds. The nucleotides themselves are the basic constituents of the DNA molecule, and are composed of a sugar, called deoxyribose, a phosphate group, and one of four nitrogen nucleobases: Adenine (A), Cytosine (C), Thymine (T), and Guanine (G). It is through the nitrogen bases in each nucleotide that the two DNA strands are linked, forming a base-pair, with the combinations A-T and C-G.

Genes, defined in biology as sequences of nucleotides, are responsible for the transfer of information between an individual and their offspring, causing the inheritance of physical traits (such as the color of the eyes or hair). The whole human genome is constituted by 23 pairs of chromosomes (sequences of DNA), in which, each gene occupies a specific fixed position inside a chromosome, maned a *locus*. A variant of a gene in the same *locus* is called an allele (different alleles can cause different phenotypical traits). Each *locus* is defined by two alleles, one in each chromosome of the pair, each one of those alleles is inherited by one of the parents.

A Genome-wide association study (GWAS) is an observational study of genetic variants in the entire genome of an organism, encompassing multiple individuals, displaying different phenotypes for a certain trait or disease. The main objective of such methods is to find if any genetic variations can be associated with the genetic trait being tested. The most common genetic variation in the human genome is the Single-nucleotide polymorphism (SNP), which represents a difference in a single nucleotide, at a specific position in the DNA. GWAS typically focus on the association between SNPs and the phenotype, considering that the SNPs have independent effects on the said phenotype or disease. This approach has been successfully used to identify a susceptibility to some diseases, such as as myocardial infarction [21] or age-related macular degeneration [22]. However, as this approach neglects the effect of the interaction of two or more SNPs, it provides limited results on more complex genetic diseases.

Accordingly, Epistasis can be defined as the interaction between genes to the definition of a phenotype. It takes into account that a phenotype can be caused by not only the independent effect of a *locus*, but also by the effect that different *loci*, have on each other on creating that phenotype [23].

A. Epistasis Detection

Exhaustive search methods to detect epistasis are typically divided in the creation of contingency tables for all possible combinations and the use of a statistical test (often referred to as objective function) to extract information from the contingency tables. These reflect the frequency distribution of all possible SNP combinations in the dataset throughout the individuals. In particular, one contingency table is created for each combination of SNPs, which is then used to compare the frequency of each interaction.

The number of contingency tables to be created (one for each possible combination) depends on the number of SNPs in the dataset and the order of the interactions, and is given by:

$$\frac{\prod_{i=0}^{k-1}(N-i)}{k!}, \tag{1}$$

where N is the number of SNPs in the dataset and K represents the order of interactions. As each SNP can assume 3 different states and each table holds information relative to the cases and controls group, the number of entries of a contingency table is given by $(2 * 3^K)$. Table I shows an example of half a contingency table (either for cases or controls), for second-order interactions between SNPs, where each entry of the table (n_{xy}) denotes the number of times which that interaction occurs throughout all the individuals.

In an attempt to decrease the computational burden of epistasis detection, a new approach [11] proposed that the SNPs can be represented in a notation that allows for the use of simple bit-wise operations to calculate the entries of the

TABLE I
HALF OF A CONTINGENCY TABLE FOR PAIR-WISE INTERACTIONS

		SNP B		
		0	1	2
SNP A	0	n_{00}	n_{01}	n_{02}
	1	n_{10}	n_{11}	n_{12}
	2	n_{20}	n_{21}	n_{22}

TABLE II
BINARY REPRESENTATION OF ONE SNP

SNP A	0	1	2	(...)	2	0
A0	1	0	0	(...)	0	1
A1	0	1	0	(...)	0	0
A2	0	0	1	(...)	1	0

contingency tables, by coding one SNP as a combination of 3 bits in a one-hot notation (depicted in Table II). The three vectors needed to code a SNP are referred to as genotype zero vector (G0), genotype one vector (G1) and genotype two vector (G2), and are as wide as the number of patients in the dataset. Note that G2s can be inferred by a bit-wise XOR between G0 AND G1. Therefore, only 2 bits are needed to store each SNP in the dataset, as G2 of a SNP is generated in hardware when needed.

Using this representation, the entries of the contingency tables are generated by simply performing a bit-wise AND between all the three vectors of the SNPs that compose the K^{th} order combination and a population count (PopCount) operation (to count the number of ones) over the resulting 3^K vectors.

After the creation of the contingency tables (which represent the SNP-SNP interactions), it is necessary to extract meaningful information from them. To do so, a statistical test needs to be performed using the generated data. The functions that are used to obtain relevant information from the generated contingency tables are often referred to as objective functions. These are used to evaluate the significance that each interaction has on the phenotype. Several objective functions are used in epistasis detection, including functions based on the chi-square test [24], ROC-curves [25] and regression modules [26]. There are also objective functions based on information theory [27], such as information gain [28] and mutual information [19], [27]. Some methods even propose the use on two or more objective functions for detection epistasis [29]. The method proposed in this paper adopts Mutual Information (MI) as the objective function, due to its simple implementation in hardware.

MI is an information theory concept used to quantify the mutual dependence between two random variables and it can be defined using their entropy, such that:

$$I(X;Y) = H(X) + H(Y) - H(X,Y) \tag{2}$$

MI can be used as an objective function, as the genotype and the disease (phenotype) can be defined as the random variables X and Y as in Equations 3 and 4 respectively, where n_x represents entries of a contingency table, N_x the total number of patients, and the indexes 0 and 1 represent cases and controls, respectively. In Equation 3, the accumulation is done for all entries of the contingency table. Note that for a balanced dataset, containing the same amount of cases and controls results in $H(Y) = 1$, which simplifies the calculation.

$$H(X) = -\sum[P(n_0+n_1) * \log_2(P(n_0+n_1))] \tag{3}$$

$$H(Y) = -[P(N_0) * \log_2(P(N_0)) + P(N_1) * \log_2(P(N_1))] \tag{4}$$

Using Equations 2, 3 and 4 and attending to the fact that $P(n) = \frac{n}{N}$, where N is the total number of patients in the data set, the MI calculation for each contingency table is given by

$$I(X;Y) = \frac{N[H(X) - H(X,Y)]}{N} - H(Y), \tag{5}$$

where

$$N[H(X) - H(X,Y)] = \sum[n_0 * \log_2(n_0) + n_1 * \log_2(n_1) - (n_0+n_1) * \log_2(n_0+n_1)], \tag{6}$$

The higher the mutual information value of an interaction, the more likely that interaction is to influence the phenotype (disease).

B. FPGAs in Epistasis Detection

FPGA-based accelerators for exhaustive search epistasis detection of second and third-order interactions have been proposed [18], [19]. However, they still lack in flexibility, scalability, performance and energy efficiency limiting their applicability.

The architecture for pair-wise epistasis detection proposed in [18] manages to perform second-order epistasis detection on a dataset with 500 000 SNPs and 5000 patients in 4 minutes. Which results in a speedup of about $285\times$ when compared to an implementation running on two Intel Xeon quad-core CPUs, with both implementations running the iLOCi objective function [10]. This architecture is organized in a systolic array topology, where each Processing Element (PE) stores one SNP and processes it against the SNPs that are being streamed throughout all the PEs. This architecture is implemented in a RIVYERA S6-LX150 accelerator, containing 128 Xilinx Spartan-6 LX150 FPGAs, which results in a very high power consumption (780W).

An other architecture to detect third-order epistasis in an FPGA proposed in [19] is implemented in a Virtex7-VX690T and in a Kintex7-K325T FPGA. This implementation provides a speedup of $363\times$ and $181\times$ when implemented in a Virtex7-VX690T and in a Kintex7-K325T FPGA, respectively, when compared against a software application running in an Intel

Core-i7 Sandy Bridge @3.20 GHz 6-core CPU, for a dataset with 10 000 SNPs and 5 000 samples.

Similarly to the second-order architecture [18], this architecture is based on a systolic array for the creation of contingency tables. However, each PE stores data referring to two SNPs (to process against the SNP being streamed) to create the contingency tables for third-order interactions, which means that each PE stores twice the data of the PEs in the second-order architecture.

Despite providing significant performance improvements over CPU applications, the mentioned architectures are designed for a specific number of patients, which means they can not be used with any dataset. Also, they are not scalable with the order of interactions, as the amount of data to be stored in each PE increases with the order of interactions, which would exhaust the BRAM availability limiting the number of PEs that can be implemented.

Consequently, the accelerator architecture for FPGA platforms that is now proposed is generated by a specially devised method that allows the parameterization of the architecture for any order of interactions and any number of patients. Such a solution presents a two-fold benefit when compared to existing FPGA-based solutions, by not only enabling the deployment of specialized accelerators for higher than third-order epistasis detection in FPGAs, and by allowing a design-time parameterization to support an arbitrary number of patients, according to the targeted dataset. It is also capable of providing higher performance and energy efficiency than the state-of-the-art implementations.

III. Specialized Accelerator Architecturefor High-order Epistasis Detection

The proposed system is composed of an FPGA-based custom accelerator, to create contingency tables and calculate their MI value. The accelerator is connected to a host GPP who runs the software application that is responsible to control the processing order and to manage the data transfers from the memory that holds the dataset to the FPGA.

The proposed system was implemented in Xilinx Zynq-7000 and Zynq-Ultrascale+ System on Chips (SoCs). The Zynq family of SoC provides an ARM GPP tightly-coupled an FPGA device that can be used as an accelerator. The communication between the GPP and the FPGA-based accelerator and the data transfers from the memory to the FPGA are made though an AXI4 interface that is managed by a software application running in the GPP. However, the implementation of the proposed system is not limited to the use of a Zynq SoC or to a tightly coupled system. A board containing an FPGA can, for example, be connected to a host computer via PCI-Express, as the proposed accelerator architecture can be implemented in any FPGA.

A simplified overview of the proposed generic architecture is depicted in Figure 1. The proposed architecture is composed by a chain of Contingency Table Units (CTUs), and a set of reconstruction units, shared among several CTUs, depending on the number of patients in the dataset. The MI scores for each contingency table are then calculated and the best values are stored. The *Count Last* unit in the beginning of the array, is used to count the number of received words and serves as a control unit for the systolic array. The proposed architecture can be divided into two main sections: (i) the creation of the partial contingency tables in the systolic array, and their subsequent reconstruction; (ii) and the calculation of the MI value for each table.

A. Data Representation

The proposed accelerator accepts datasets stored in the binary representation detailed in Section II-A streamed one SNP at a time. The number of words that are used to send an entire SNP must be even, as the first word corresponds to the G0 of the first $R/2$ cases and $R/2$ controls, and the second word is used to send the G1 of the same patients (R represents the width of the interface). As such, the number of words used to stream one SNP is given by Equation 7 where $\#Patients$ is the number of patients in the dataset.

$$\#Words\_SNP = \left\lceil \frac{\#Patients}{R/2} \right\rceil + \left(\left(\frac{\#Patients}{R/2} \right) \mod 2 \right) \quad (7)$$

Each *save value* unit stores the X best MI values (represented in single-precision floating-point format), and as many 32-bit words as the order of interactions (to identify the combination). Consequently, after the processing of all the possible combinations, each *save value* unit streams X MI values and $(X * K)$ SNP identifiers (where K corresponds to the order of interactions) to the Host, each of them stored in 32-bits. Therefore, the number of words that are streamed to the host is given by Equation 8, where N_save_units is the number of save units implemented in the design.

$$\#Words\_res = \left\lceil \frac{32 * (1 + K)}{R} \right\rceil * X * N\_save\_units \quad (8)$$

The host is then responsible to sort the combinations and present the X best combinations to the user.

Whenever the number of patients is not divisible by $R/2$, dummy patients are added to that dataset that are ignored by the architecture, until a number of patients divisible by $R/2$ produced, this is contemplated in Equation 7 by the ceiling of the first term. Also, if the first term of the equation does not produce an even number of words, another word is added, meaning that $R/2$ dummy patients are added to the last pair of words.

B. Contingency Table Creation

Contrarily to the state-of-the-art architecture [19], only the first unit of the systolic array stores more than one SNP, independently of the order of interactions. As such, to process interactions of order K, $(K-1)$ SNPs are stored in the first, from which, $(K-2)$ are propagated through the systolic

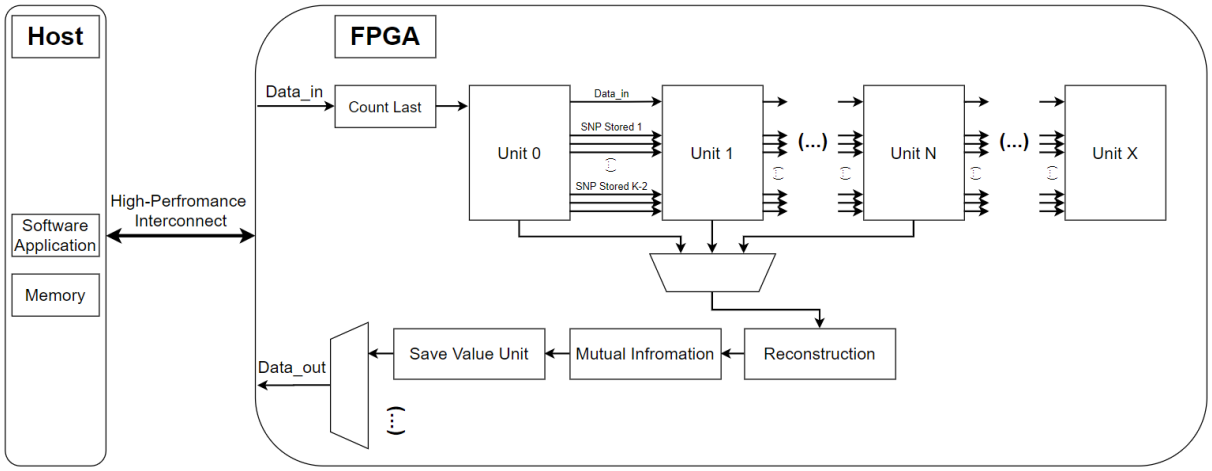

Fig. 1. Generic architecture overview

array ate the same time as the SNP that is streamed from the memory. Therefore, each unit in the systolic array will have access to the SNP that is being streamed from the memory, the SNP that it stores, and the $(K-2)$ SNPs that are stored in the first unit, totalizing the K SNPs needed to process a K^{th}-order interaction. This solution allows for the use of significantly less BRAMs than the state-of-the-art implementation.

In CTUs that compose the systolic (depicted in Figure 2), only one SNP is stored (except in the first), since the $(K-2)$ SNPs that are stored in the first unit are being streamed throughout the systolic array at the same rate as the SNP that comes through the data_in port. As such, every unit has access to different combinations of SNPs of the K^{th} order, having to store only one SNP, requiring each unit to have $[1+(K-2)*3]$ inputs, and output ports, and additional output ports to sent the contingency tables to the reconstruction unit.

The functioning of the CTUs is divided into three phases: (i) the creation of a complete contingency table for $(K-1)^{th}$-order (necessary to the reconstruction of the missing contingency table entries); (ii) the generation of the third of the contingency table that depends on G0 of the SNP that is streamed from the memory; (iii) the generation of the third of the contingency table that depends on G1 of the same SNP. The second and third stages are executed alternately.

1) First Phase: The first phase of the execution of a CTU is the creation of a complete contingency table for $(K-1)^{th}$-order, whose entries are referred to as "$n_{(K-1)}$". This process is initiated while the $(K-1)^{th}$ SNP is being streamed from memory and stored in the first stage of the unit's pipeline. As the complete contingency table is to be generated, a NOR port is used to generate G2 of the stored SNPs.

In the second and third pipeline stages, all the $3^{(K-1)}$ possible combinations between the (K-1) stored SNPs are processed (by performing an AND between G0, G1, and G2 of the stored SNPs) and the subsequent PopCount. The results of this stage correspond to partial entries of the $(K-1)^{th}$-order contingency table. As such, in the fourth and fifth stages, the values are accumulated until all the patients that compose the SNP have been streamed from the memory, while the accumulated values are stored in the "POP_count (K-1)" registers. Therefore, at the end of the first phase, these $3^{(K-1)}$ registers will store an entire $(K-1)^{th}$-order contingency table.

2) Second Phase: The second phase of execution of a CTU is initiated when the last SNP of the combination starts to be received through the "Data_IN" port. In the second pipeline stage, the first set of the G0 vector of the SNP is processed against the first set of the G0, G1 and G2 vectors of the stored SNPs. This generates $3^{(K-1)}$ vectors that are PopCounted in the third stage. As the results of the PopCount correspond to partial entries, the values are stored in the "Tab REG (K-1), 0" set of registers (in the fifth stage), and accumulated with the next set of values corresponding to the G0 of the streamed SNP. However, as the second word that the unit receives refers to the first set of the G1 vector of the streamed SNP, the accumulation is only done every two clock cycles.

At the end of the execution phase, the "Tab REG (K-1), 0" set of registers hold the third of a contingency table that is dependent on the G0 vector of the last SNP of the combination "$n_{(K-1),0}$".

3) Third Phase: As partial G0 and G1 vectors of a SNP are streamed one after the other, the second and third phases are executed alternately. The third execution stage of a CTU is similar to the second. However, the values produced in this stage are the contingency table entries dependent on the G1 vector of the last SNP of the combination "$n_{(K-1),1}$". The partial values produced in this stage are stored in the "Tab REG (K-1), 1" set of registers. To alternate between both execution phases one of the registers in the fifth pipeline stage is deactivated and the following multiplexer is controlled accordingly.

Since G2 from the SNPs that are streamed from the communication interface is never generated, the entries of the contingency table that depend on that vector (represented by $n_{[(K-1),2]}$), are not generated in the CTUs. To calculate

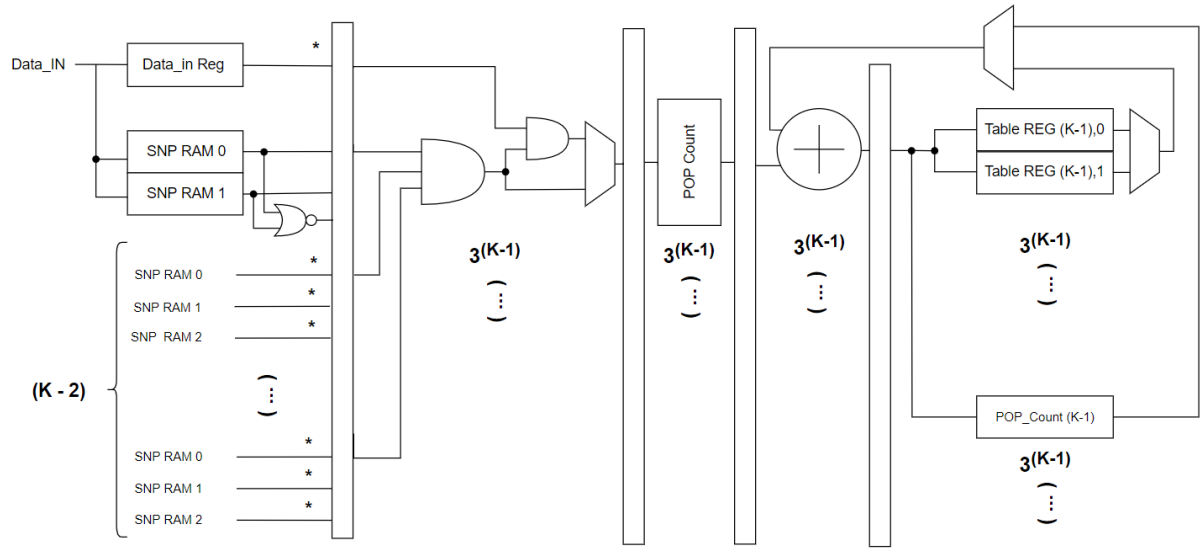

Fig. 2. Generic CTU datapath

the missing values, the contingency table of the $(K-1)^{th}$ combination (generated in the CTUs first phase of execution), is used in accordance with the Equations 10 and 9

$$n_{(K-1)} = n_{[(K-1),0]} + n_{[(K-1),1]} + n_{[(K-1),2]}, \quad (9)$$

$$n_{[(K-1),2]} = n_{(K-1)} - (n_{[(K-1),0]} + n_{[(K-1),1]}), \quad (10)$$

where $n_{(K-1)}$ represents the entries of the contingency table for the $(K-1)^{th}$-order combination, and $n_{[(K-1),0]}$, $n_{[(K-1),1]}$ and $n_{[(K-1),2]}$ are entries of the contingency table for the K^{th} order combination, resulting in an AND between $n_{(K-1)}$ and vectors G0, G1 and G2 of the next SNP, respectively. As such the equality expressed in Equation 9 is always valid.

Equations 9 and 10 can be more easily understood by taking in account an example for a third-order contingency table. An entry of a second-order contingency table, such as "10", when combined with another SNP results in three entries of a third-order contingency table, "100", "101" and "102", depending on the vectors G0, G1 and G2 of that SNP, respectively. As the number of patients is the same for all SNPs, the number in the entry "10" of the second-order contingency table, is equal to the sum of the entries "100", "101", "102" of the third-order table, this relation is defined by Equation 9. As the vector G2 of the last SNP is neither streamed nor generated in the CTU, the reconstruction units implement Equation 10.

Each reconstruction unit accepts six vales per clock cycle (three for cases and three for controls), one from each port of the CTUs. As such, the amount of reconstruction units needed for each group of CTUs is equal to the number of values sent by each port per clock cycle, as they have to processed in parallel. As the number of patients increase, each reconstruction unit will be shared among more CTUs, which increases the efficiency of the architecture.

The partial tables that are created in the accelerator's CTUs, are sent to the reconstruction units, where their missing values are calculated. This section details the process of sending the partial contingency tables to the reconstruction units, which is done differently depending on the number of patients in the targeted dataset. This happens because the values that are generated in each CTU must be sent to the reconstruction units before the next values are generated. The number of clock cycles between the generation of two complete sets of values in the CTUs depends on the number of patients. As such, for more patients, there are more available clock cycles for the transfer and reconstruction of the partial tables, which creates an opportunity for sharing the reconstruction units.

C. Mutual Information Calculation

As N and $H(Y)$ (see Equation 5) are constant, to sort the combinations by their MI values, only $N[H(X) - H(X,Y)]$ (given by Equation 6) needs to be calculated. Each MI unit (depicted in Figure 3), calculates a partial MI value for a contingency table. Instead of calculating the value of $(n*log_n)$ (as it would occupy a lot of the available resources), the results are obtained using a look-up table implemented with dual-port BRAMs. To save resources, and as only 3 $(n*log_n)$ operations are done in each MI unit, one of the dual-port BRAMs is shared between two MI units.

The MI units accept a pair of values per clock cycle (a case and a control). Consequently, 3 MI units are implemented for every reconstruction unit. As each unit produces a partial MI value, the values created are added and accumulated until all partial MI values have been summed. The X best MI values, along with the identifiers of the combination, are stored in each save value unit (each using X comparators) and are then sent to the host computer, guaranteeing that within the X times the number of save value units values sent to the host, the best X values are present. The host is responsible for sorting

TABLE III
$K = 2$ COMPARISON FOR 500 000 SNPS AND 5000 PATIENTS

Device	Time	Comb/sec x10^{12}	Power (W)	Energy (kJ)	EDP (J*s) x10^{6}
RIVYERA [18]	4m	2.61	780	180.00	43.20
Zynq-7000 [Proposed]	4m 16s	2.44	9.14	2.34	0.60
Zynq-U+ [Proposed]	3m 10s	3.29	11.25	2.12	0.40

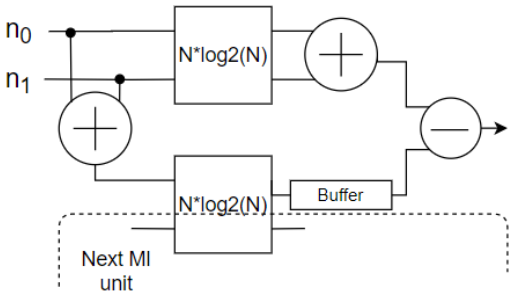

Fig. 3. Mutual information unit overview

TABLE IV
PROPOSED $K = 2$ ARCHITECTURE RESULTS FOR 5000 PATIENTS

#SNPs	50k	100k	200k	500k
Time	3s	10s	41s	4m 16s
Energy (J)	25.8	86.0	352.5	2021.1

the received values, store the best X values and complete the calculation of the MI value as in Equation 5, by dividing them by the number of patients and subtracting $H(Y)$.

D. Second-Order Optimizations

Although the desired general architecture enables the creation of FPGA-based accelerators for any order of combinations, by taking advantage of the smaller contingency tables that are created for second-order interactions, some optimizations that are specific for this case can be made. These optimizations do not apply to the general method due to the added complexity in the reconstruction of the contingency tables, as a new reconstruction unit would have to be created for each order of interactions, invalidating the scalability of the design with the order of interactions.

When applying the specific optimizations for second-order interactions, the G2 of any SNP in never generated. As such, the 10 (out of 18) entries of the contingency tables that depend on those vectors are not generated, reducing the resource utilization of each CTU, as each only produces 8 values, instead of the 12 that would be produced by using the general method. However, this implies the utilization of a more complex reconstruction unit, following Equations 11 and 12.

$$\begin{cases} n_{(0,2)} = n_{(0,X)} - (n_{(0,0)} + n_{(0,1)}) \\ n_{(1,2)} = n_{(1,X)} - (n_{(1,0)} + n_{(1,1)}) \\ n_{(2,0)} = n_{(X,0)} - (n_{(0,0)} + n_{(1,0)}) \\ n_{(2,1)} = n_{(X,1)} - (n_{(0,1)} + n_{(1,1)}) \\ n_{(2,2)} = n_{(2,X)} - (n_{(2,0)} + n_{(2,1)}) \end{cases} \tag{11}$$

$$n_{(2,X)} = N - (n_{(0,X)} + n_{(1,X)}) \tag{12}$$

Where $n_{(0,X)}$, $n_{(1,X)}$ and $n_{(2,X)}$ are the number of ones in the G0, G1, and G2 of the first SNP of the combination, respectively ($n_{(X,0)}$ and $n_{(X,1)}$ follow the same notation for the second SNP of the combination). Those values are generated for all SNPs in the *count last* unit at the beginning of the systolic array and propagated trough the array along with the SNP data.

As the contingency tables for second-order interactions are smaller, only two ports are used to send the values to the reconstruction units (one for cases and one for controls), and only one value per clock cycle is sent through them. As such, eight clock cycles are needed to send the 16 values (4 contingency table entries plus $n_{(0,X)}$, $n_{(1,X)}$, $n_{(X,0)}$ and $n_{(X,1)}$, for both cases and controls). However, as the reconstruction unit takes nine clock cycles to output the entire contingency table (2 values per clock cycle), one stall is added, which means that each CTU takes nine clock cycles to send their values to the reconstruction unit. Consequently, the number of words per SNP can not be less than nine, which (as the used bandwidth is 64 bits) translates to a minimum of 257 patients. Therefore if the number of patients is less than 257, dummy patients that do not influence the result are added until the condition is met. Despite impairing the performance of the architecture for a dataset with less than 257 patients, this solution increases the number of CTUs that can be implemented for a number of patients larger than that threshold.

Contrarily to the general architecture, when implementing the specific optimizations for second-order interactions, at most, only one reconstruction unit is implemented for each contingency table. That reconstruction unit can be shared among several CTUs depending on the number of patients in the dataset. In the best case scenario, only one reconstruction unit could be implemented for the entire systolic array.

The MI units used in this case are the same as in the general architecture. However, as the reconstruction units only output two values per clock cycle, only one MI unit is needed for each reconstruction unit.

TABLE V
$K = 3$ TIMING AND PERFORMANCE RESULTS FOR 20 000 SNPS AND 5000 PATIENTS

Device	Time	Comb/sec x10^{12}	Power (W)	Energy (kJ)	EDP (J*s) x10^{6}
Virtex-7 [19]	4h 33min	0.41	-	-	-
Zynq-7000 [Proposed]	2h 20m	0.80	14.28	119.95	1007.58
Zynq-U+ [Proposed]	1h 53m	0.98	15.71	106.52	722.21
Virtex-7 [Proposed]	1h 29m	1.24	-	-	-

TABLE VI
PROPOSED $K = 3$ ARCHITECTURE RESULTS FOR 5000 PATIENTS

#SNPs	5k	10k	15k	20k
Time	2m	18m	59m	2h20m
Energy (kJ)	1.93	15.13	50.75	119.86

IV. EXPERIMENTAL RESULTS

This section presents a thorough evaluation of the proposed accelerator for second, third and fourth-order epistasis detection implementations. The presented results comprise the execution time, the number of combinations tested per second (comb/sec), as well as the power and energy consumption and the Energy-Delay Product (EDP). The obtained results are compared to the state-of-the-art FPGA-based implementations [18], [19] mentioned in Section II-B.

The utilized boards for the implementation of the proposed accelerator are a ZYNQ7 Mini-ITX (Zynq-7000) featuring a Zynq-700 SoC (running the accelerator at 250MHz) and a ZYNQ-Ultrascale+ ZCU102 (Zynq-U+) featuring a more modern Zynq-Ultrascale+ SoC (running the accelerator at 322MHz) that provides higher energy efficiency due to the smaller transistor technology in the FPGA fabric (16nm vs 28mn). All the implementations were made using Xilinx Vivado 2017.2 design suite, and the power consumption values presented were estimated by the Vivado power estimator tool. The state-of-the-art designs are implemented in a RIVYERA S6-LX150 [18] (RIVYERA) and in a Virtex-7 690T [19], for second and third-order interactions, respectively.

In Tables III and V, presented in the following sections, the comparable state-of-the-art implementation results, are displayed in the first line of the tables in italic.

A. Second-order Epistasis Detection

The results regarding execution time and energy consumption of the proposed second-order accelerator implemented in a Zynq-7000 targeting 5000 patients are presented in Table IV (the design draws 8.6W of power).

Table III displays results regarding the performance and energy consumption of performing second-order epistasis detection in a dataset with 500 000 SNPs and 5000 patients, for both the sate-of-the-art implementation (first line of the table), and the proposed accelerator. The execution times and the Comb/sec of both implementations are similar (a speedup of 1.26× is obtained when comparing the state-of-the-art implementation [18] with the proposed accelerator implemented in the Zynq-U+).

However, despite producing similar execution times, since the proposed accelerator is implemented in a single FPGA, its power consumption is much lower. Therefore, the total energy consumption of the proposed accelerator is 77× and 81× less than the state-of-the-art FPGA-based accelerator [18], when implemented in the Zynq-7000 and in the Zynq-U+, respectively. This demonstrated the much higher energy efficiency of the proposed architecture, which can be measured by the EDP value also presented on the table.

B. Third-order Epistasis Detection

The results regarding execution time and energy consumption of the proposed third-order accelerator implemented in a Zynq-7000 targeting 5000 patients (using 14.2W of power) are presented in Table VI.

The timing and Comb/sec results of the execution of third-order epistasis detection on a dataset containing 5000 patients and 20 000 SNPs for the proposed and the state-of-the-art architecture are represented in Table V. As the state-of-the art architecture is implemented in a Virtex-7 FPGA, for better performance comparison, the proposed accelerator was also implemented in the same FPGA. This resulted in an accelerator that is more than 3× faster than the state-of-the art. No conclusion can be drawn in regards to the power consumption or energy efficiency with the sate-of-the art implementation, as that information is not available in [19]. Power and energy consumption results in the Virtex-7 690T FPGA are also not depicted in Table V, as the FPGA-based accelerator was implemented, not the entire system.

C. Fourth-order Epistasis Detection

Using the proposed general architecture for fourth-order interactions results on the first FPGA-based accelerator for exhaustive search fourth-order epistasis detection. As such, there is no other implementation to compare the obtained results against. Table VIII shows the estimated execution time and energy consumption of an architecture targeting 4000 patients (which has a power consumption of 13.7W) for different number of SNPs in the Zynq-7000 board.

The same architecture was also implemented in the Zynq-U+ board and in a Virtex-7 FPGA. Table VII present the results

TABLE VII
$K = 4$ RESULTS FOR 2000 SNPS AND 4000 PATIENTS

Device	Time	Comb/sec x10^{12}	Power (W)	Energy (kJ)	EDP (J*s) x10^{6}
Zynq 7000 [Proposed]	2h 38m	0.28	13.72	130.07	1233.06
Zynq-U+ [Proposed]	2h 8m	0.35	14.83	114.12	876.44

TABLE VIII
PROPOSED $K = 4$ ARCHITECTURE RESULTS FOR 4000 PATIENTS

#SNPs	800	1200	1600	2000
Time	4m	21m	1h6m	2h38m
Energy (kJ)	3.54	17.45	54.08	130.56

of those implementations fora dataset with 2000 SNPs. Using a Zynq-Ultrascale+ SoC to implement the accelerator it is possible to perform exhaustive fourth-order epistasis detection in about two hours.

D. Discussion

According to the results obtained, by following the proposed method, specialized architecture to detect epistasis targeting any order of interactions and adaptable to any number of patients can be easily created. The complete proposed system can be implemented in any board featuring a Zynq SoC. However, the generated logic designs can be implemented in any FPGA. The generated architectures when implemented in larger FPGAs, can be implemented with more CTUs, which in turn, as corroborated by the implementations on the Virtex-7 690T, provides higher performance, the architectures are, therefore, scalable with the amount of FPGA resources available. The architectures can also run at different clock speeds, as verified by the implementations on the Zynq-7000 and the Zynq-Ultrascale+ SoCs, where the design run at 250MHz and 322MHz, respectively.

V. CONCLUSIONS

The increase in the computation complexity to perform high-order epistasis detection based on exhaustive search methods motivates the use of DSAs that can take advantage of the existent data parallelism, such as GPUs and FPGAs. However, the existent FPGA-based accelerators for epistasis detection, despite providing good results when comparing to CPU implementations, lack the flexibility to be used with any number of patients and the scalability to be adapted to higher-order epistasis, being limited to second, and third-order interactions. The proposed architecture addresses the limitations of the existent implementations by providing a general architecture that can be used to deploy FPGA-based accelerators for any order of interactions and targeting any dataset. The increased performance and energy efficiency obtained when using the proposed second and third-order accelerators allow for the implementation of the first FPGA-based accelerator for fourth-order epistasis detection.

The proposed architecture was implemented for second, third, and fourth order interactions, targeting numbers of patients ranging from 64 to 8000 in the ZYNQ7-Mini ITX and Zynq-Ultrascale+ ZCU102 boards and in a Virtex-7 690T FPGA. When comparing the considered implementations of the proposed architecture with state-of-the art-FPGA implementations for second and third-order epistasis detection, it is clear that the proposed architectures provide $3\times$ faster execution times, for third-order epistasis detection when using similar hardware, and significantly lower energy consumption, when comparing against the state-of-the-art implementation targeting second-order interactions. Despite producing better performing architectures than the state-of-the-art, the main advantage of the method proposed in this thesis is the added flexibility that it provides by allowing for an easy generation of highly efficient architectures targeting any order of interactions and any dataset.

REFERENCES

[1] T. A. Manolio, "Genomewide Association Studies and Assessment of the Risk of Disease," *New England Journal of Medicine*, vol. 363, no. 2, pp. 166–176, 2010.

[2] J. L. Haines, M. A. Hauser, S. Schmidt, W. K. Scott, L. M. Olson, P. Gallins, K. L. Spencer, S. Y. Kwan, M. Noureddine, J. R. Gilbert, N. Schnetz-Boutaud, A. Agarwal, E. A. Postel, and M. A. Pericak-Vance, "Complement Factor H Variant Increases the Risk of Age-Related Macular Degeneration," *Science*, vol. 308, no. 5720, pp. 419 LP – 421, apr 2005. [Online]. Available: http://science.sciencemag.org/content/308/5720/419.abstract

[3] J. L. Vassy, M.-F. Hivert, B. Porneala, M. Dauriz, J. C. Florez, J. Dupuis, D. S. Siscovick, M. Fornage, L. J. Rasmussen-Torvik, C. Bouchard, and J. B. Meigs, "Polygenic Type 2 Diabetes Prediction at the Limit of Common Variant Detection," *Diabetes*, vol. 63, pp. 2172–2182, 2014. [Online]. Available: http://diabetes.diabetesjournals.org/lookup/suppl/doi:10.2337/db13-1663/-/DC1.

[4] J. H. Moore, F. W. Asselbergs, and S. M. Williams, "Bioinformatics challenges for genome-wide association studies," *BIOINFORMATICS REVIEW*, vol. 26, no. 4, pp. 445–455, 2010. [Online]. Available: https://academic.oup.com/bioinformatics/article/26/4/445/244836

[5] W. H. Wei, G. Hemani, and C. S. Haley, "Detecting epistasis in human complex traits," *Nature Reviews Genetics*, vol. 15, no. 11, pp. 722–733, 2014. [Online]. Available: http://dx.doi.org/10.1038/nrg3747

[6] J. C. Turton, J. Bullock, C. Medway, H. Shi, K. Brown, O. Belbin, N. Kalsheker, M. M. Carrasquillo, D. W. Dickson, N. R. Graff-Radford, R. C. Petersen, S. G. Younkin, K. Morgan, and F. Panza, "Investigating Statistical Epistasis in Complex Disorders," *Journal of Alzheimer's Disease*, vol. 25, pp. 635–644, 2011. [Online]. Available: http://www.broadinstitute.org/mpg/snap/ldsearch.php

[7] T. D. Howard, G. H. Koppelman, A. Xu, S. L. Zheng, D. S. Postma, D. A. Meyers, and E. R. Bleecker, "Gene-gene interaction in asthma: Il4ra and il13 in a dutch population with asthma," *American Journal of Human Genetics*, vol. 70, no. 1, pp. 230–236, 2002.

[8] C. Niel, C. Sinoquet, C. Dina, and G. Rocheleau, "A survey about methods dedicated to epistasis detection," *Frontiers in Genetics*, vol. 6, no. SEP, 2015.

[9] T. F. Mackay and J. H. Moore, "Why epistasis is important for tackling complex human disease genetics," p. 125, jun 2014. [Online]. Available: http://genomemedicine.biomedcentral.com/articles/10.1186/gm561

[10] J. Piriyapongsa, C. Ngamphiw, A. Intarapanich, S. Kulawonganunchai, A. Assawamakin, C. Bootchai, P. J. Shaw, and S. Tongsima, "iLOCi: a SNP interaction prioritization technique for detecting epistasis in genome-wide association studies." *BMC genomics*, vol. 13 Suppl 7, no. Suppl 7, 2012.

[11] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU Computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, may 2008.

[12] R. Nobre, A. Ilic, S. Santander-Jimenez, and L. Sousa, "Exploring the Binary Precision Capabilities of Tensor Cores for Epistasis Detection," *Proceedings - 2020 IEEE 34th International Parallel and Distributed Processing Symposium, IPDPS 2020*, pp. 338–347, 2020.

[13] J. González-Domínguez and B. Schmidt, "GPU-accelerated exhaustive search for third-order epistatic interactions in case-control studies," *Journal of Computational Science*, vol. 8, pp. 93–100, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.jocs.2015.04.001

[14] L. Wienbrandt, "Bioinformatics applications on the FPGA-based high-performance computer RIVYERA," in *High-Performance Computing Using FPGAs*. New York, NY: Springer New York, 2013, vol. 9781461417, pp. 81–103.

[15] A. Surendar, "FPGA based parallel computation techniques for bioinformatics applications," *International Journal of Research in Pharmaceutical Sciences*, vol. 8, no. 2, pp. 124–128, 2017.

[16] S. A., A. M., and S. P. P., "A parallel reconfigurable platform for efficient sequence alignment," *African Journal of Biotechnology*, vol. 13, no. 33, pp. 3344–3351, 2014.

[17] N. Neves, N. Sebastião, A. Patricio, D. Matos, P. Tomás, P. Flores, and N. Roma, "BioBlaze: Multi-core SIMD ASIP for DNA sequence alignment," in *2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors*, 2013, pp. 241–244.

[18] L. Wienbrandt, J. C. Kässens, J. González-Domínguez, B. Schmidt, D. Ellinghaus, and M. Schimmler, "FPGA-based acceleration of detecting statistical epistasis in GWAS," *Procedia Computer Science*, vol. 29, pp. 220–230, 2014.

[19] J. C. Kässens, L. Wienbrandt, J. González-Domínguez, B. Schmidt, and M. Schimmler, "High-speed exhaustive 3-locus interaction epistasis analysis on FPGAs," *Journal of Computational Science*, vol. 9, pp. 131–136, jul 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S187775031500068X

[20] S. M. Trimberger, "Three Ages of FPGAs: A Retrospective on the First Thirty Years of FPGA Technology: This Paper Reflects on How Moore's Law Has Driven the Design of FPGAs Through Three Epochs: The Age of Invention, the Age of Expansion, and the Age of Accumulation," *IEEE Solid-State Circuits Magazine*, vol. 10, no. 2, pp. 16–29, mar 2018.

[21] K. Ozaki, Y. Ohnishi, A. Iida, A. Sekine, R. Yamada, T. Tsunoda, H. Sato, H. Sato, M. Hori, Y. Nakamura, and T. Tanaka, "Functional SNPs in the lymphotoxin-α gene that are associated with susceptibility to myocardial infarction," *Nature Genetics*, vol. 32, no. 4, pp. 650–654, 2002. [Online]. Available: https://doi.org/10.1038/ng1047

[22] R. J. Klein, C. Zeiss, E. Y. Chew, J. Y. Tsai, R. S. Sackler, C. Haynes, A. K. Henning, J. P. SanGiovanni, S. M. Mane, S. T. Mayne, M. B. Bracken, F. L. Ferris, J. Ott, C. Barnstable, and J. Hoh, "Complement factor H polymorphism in age-related macular degeneration," *Science*, vol. 308, no. 5720, pp. 385–389, apr 2005. [Online]. Available: /pmc/articles/PMC1512523/?report=abstract https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1512523/

[23] H. J. Cordell, "Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans," *Human Molecular Genetics*, vol. 11, no. 20, pp. 2463–2468, 2002.

[24] R. J. Tallarida and R. B. Murray, "Chi-Square Test BT - Manual of Pharmacologic Calculations: With Computer Programs," R. J. Tallarida and R. B. Murray, Eds. New York, NY: Springer New York, 1987, pp. 140–142. [Online]. Available: https://doi.org/10.1007/978-1-4612-4974-0_43

[25] B. Goudey, D. Rawlinson, Q. Wang, F. Shi, H. Ferra, R. M. Campbell, L. Stern, M. T. Inouye, C. S. Ong, and A. Kowalczyk, "GWIS–model-free, fast and exhaustive search for epistatic interactions in case-control GWAS." *BMC genomics*, vol. 14 Suppl 3, no. Suppl 3, pp. 1–18, 2013.

[26] X. Wan, C. Yang, Q. Yang, H. Xue, X. Fan, N. L. Tang, and W. Yu, "BOOST: A fast approach to detecting gene-gene interactions in genome-wide case-control studies," pp. 325–340, 2010.

[27] P. G. Ferrario and I. R. Kö, "Transferring entropy to the realm of GxG interactions." [Online]. Available: https://academic.oup.com/bib/article-abstract/19/1/136/2566836

[28] T. Hu, Y. Chen, J. W. Kiralis, R. L. Collins, C. Wejse, G. Sirugo, S. M. Williams, and J. H. Moore, "An information-gain approach to detecting three-way epistatic interactions in genetic association studies," *Journal of the American Medical Informatics Association*, vol. 20, no. 4, pp. 630–636, 2013.

[29] X. Li, "A fast and exhaustive method for heterogeneity and epistasis analysis based on multi-objective optimization," *Bioinformatics*, vol. 33, no. 18, pp. 2829–2836, 2017.