



# Reference-Based Haplotype Phasing with FPGAs

Lars Wienbrandt<sup>(✉)</sup>, Jan Christian Kässens, and David Ellinghaus

Institute of Clinical Molecular Biology, University Medical Center Schleswig-Holstein,  
Kiel University, Am Botanischen Garten 11, 24118 Kiel, Germany  
{l.wienbrandt,j.kaessens,d.ellinghaus}@ikmb.uni-kiel.de

**Abstract.** Haplotype phasing of individual samples is commonly carried out as a precursor step before genotype imputation to reduce the runtime complexity of the imputation step and to improve imputation accuracy. The phasing process is time-consuming and generally exceeds hours even on server-grade computing systems. Loh et al. recently introduced a fast and effective reference-based haplotype phasing software named EAGLE2 which scales linearly with the number of reference samples and variants to phase. We discovered that from the several steps of the EAGLE2 phasing process, data preparation for the internally used *HapHedge* data structure already consumes about half of the total runtime in general use cases. We addressed this problem by introducing a new design for reconfigurable architectures that accelerates this part of the software by a factor of up to 29 on a Xilinx Kintex UltraScale FPGA, resulting in a total speedup of the complete phasing process of almost 2 (the theoretical limit according to Amdahl's Law) when compared to a server-grade computing system with two Intel Xeon CPUs. As a result, we reduced the EAGLE2 runtime of genome-wide phasing of 520,000 variants in 2500 samples using the *1000 Genomes Project* reference panel from 68 min to 39 min on our system while maintaining quality.

**Keywords:** Haplotype phasing · Genotype imputation · EAGLE2 · PBWT · Hardware accelerator · Reconfigurable architecture

## 1 Introduction

An individual's *diplotype* sample consists of two strands of unique nucleotide content for each chromosome, one inherited from the mother (*maternal*) and one inherited from the father (*paternal*). *Single Nucleotide Polymorphism (SNP)* microarrays, known as *genome-wide association study (GWAS)* SNP arrays, e.g. the *Global Screening Array (GSA)* from Illumina [8], allow for the genotyping of more than 650,000 biallelic SNPs (two observed alleles; counting the reference as one, and allowing for one alternative allele) across the human genome. Each pair of alleles of a SNP is experimentally measured and subsequently encoded as a genotype (thereby ignoring phase information), for example, 0 for *homozygous*

*reference* (reference allele on both strands), 1 for *heterozygous* (one strand with reference allele, the other strand with an alternative allele), and 2 for *homozygous alternative* (alternative allele on both strands). *Haplotype* information refers to the alleles found on a single strand, e.g. 0 for the reference allele and 1 for the alternative allele. Since the genotyping process does not separate the maternally and paternally derived alleles of each SNP, the haplotype phasing problem can simply be described as to guess for each heterozygous marker of a SNP genotype on which strand the alternative allele is located (e.g. on the maternal strand), implying the reference allele to be found on the other strand (e.g. the paternal strand in this event). Technically, this means to find the best suited pair of haplotype strings (encoded with 0 and 1) to a given genotype string (encoded with 0, 1 and 2).

Phase information is important for human genetics research for several reasons, among others, to screen for the occurrence of compound heterozygosity, i.e. the presence of two deleterious variants located at different locations in the same gene but on different chromosome copies (paternal and maternal) of an individual. Further, many studies have linked specific haplotypes to drug response, clinical outcomes in transplantations and to susceptibility or resistance to disease [11]. Because of the lack of cost-effective experimental approaches for obtaining phase information, the haplotype phase is commonly estimated using computational approaches [2]. Haplotype phasing is also a very important pre-processing step for *genotype imputation* which relies on pre-phased genotypes. Genotype imputation has become a standard tool in genome-wide association studies and describes an approach of predicting (or imputing) the genotypes of variants that were observed in a reference panel but were not directly genotyped on a SNP microarray [4]. Imputation servers, such as the Sanger Imputation Server [16] or the Michigan Imputation server [14], enable users to upload GWAS SNP array data to these servers to deploy well-equipped high-performance computing clusters in order to carry out phasing and imputation.

Recently, methods have been exploited that reduce the computational requirements of phasing and imputation [5]. However, Das and colleagues demand in their article that continued computational improvements in phasing and imputation tools are necessary for imputation to remain practical as millions of samples are genotyped and reference panels increase in size to tens of thousands of sequenced genomes. Following this demand, Loh et al. presented EAGLE2 [9]. According to the article, EAGLE2 is 10% more accurate and up to 20 times faster than the preliminary state-of-the-art tool SHAPEIT2 [6], due to its linear complexity in the number of samples, reference samples and variants to phase. Thus, EAGLE2 became the default phasing method in the Sanger and the Michigan Imputation Services. (As an alternative SHAPEIT2 and the similar performing HAPI-UR [17] can be chosen for the phasing step.)

In this paper, we present a method to accelerate EAGLE2 [9] by using reconfigurable hardware. We discovered that for general use cases target preparation as the first step in the phasing process already consumes around half of the total

runtime, due to the large amount of data that has to be processed here. This step includes the analysis of the reference panel by creating a *condensed reference* individually for each target. The condensed reference mainly consists of 1 bit haplotype information for every reference sequence at each heterozygous site in the target plus 1 bit of consistency information marking a segment between heterozygous sites inconsistent or not. Furthermore, an *identity-by-descent (IBD)* check is performed and the condensed reference information is required in transposed format for efficient usage. We show that these steps are perfectly suited for a hardware design by implementing this part of EAGLE2 on a *Field Programmable Gate Array (FPGA)*. We run a performance and quality benchmark of our implementation using the *1000 Genomes Project (1000G)* [12] haplotype data as reference panel.

To our knowledge, phasing and imputation processes in general have been rarely addressed with alternative architectures such as FPGAs or GPUs. An at that time successful approach was made by Chen et al. with Mendel-GPU [3] in 2012 using a GPU-accelerator for imputation, but we have not found adequate literature describing the utilization of FPGAs to target this problem.

## 2 Haplotype Phasing with EAGLE2

### 2.1 EAGLE2 Core Overview

Po-Ru Loh et al. introduced EAGLE2 as a haplotype phasing software that uses a phased reference panel to determine the phase of the genotypes of a diploid target sample [9]. The phasing process in EAGLE2 consists of several parts. We give an overview here with details following in the subsequent sections. The input data (reference panel and target data) is required in packed and indexed *Variant Call Format (VCF)* (.vcf.gz or .bcf), whereby .bcf is the packed binary format, which is fastest.

The first part after reading the input data is to prepare an adapted reference for each target. We call this the *condensed reference* of a target. In the following, we will focus on this part for FPGA acceleration. It is divided into several steps:

1. Find the  $k$  best fitting haplotype sequences from the reference.
2. Reduce the remaining sequences to *split sites*, i.e. all sites containing a heterozygous genotype in the target plus some sites required to split large segments without any heterozygous sites. This step also preserves the information if segments between split sites are consistent with the target, or not.
3. Check for *Identity-by-descent (IBD)*, i.e. parts of a reference sample that are too similar with the target sample are masked. This keeps the process from creating a bias on certain samples from the reference.

The second part is to generate a data structure based on a *Position-based Burrows Wheeler Transformation (PBWT)* [7] from the resulting condensed reference. Loh et al. call this a *HapHedge*, which is based on a graph that allows quick access to all information from the PBWT plus meta information. For more information on the HapHedge, we refer to the supplementary material of [9].

The third and final part of the phasing is to perform a *beam search* through possible haplotype sequences fitting the target, following the probability model introduced in Sect. 2.2. The beam search is also performed in a series of substeps:

1. Fast pre-phasing. The parameters of the beam search are adjusted to create a fast rough estimation of the phased haplotypes, whereby the most confident positions are used as constraints for the next step.
2. Fine phasing. The beam search is performed again, but with parameters forcing a better estimation for the cost of runtime. To keep the runtime growth to a minimum, this step uses the constrained sites from the first step, such that only uncertain positions are phased again.
3. Reverse fine phasing. The previous step is repeated in reverse order, i.e. started from the end and forwarded until the beginning. The phase confidences of each heterozygous site are compared and the phase with the higher probability is chosen.

Previously described steps are repeated for each target. The phased output is written in VCF format again, whereby the user can choose if the output should be packed or binary packed.

## 2.2 Probability Model

To understand why preliminary steps are necessary, we describe the underlying probability model for the phase estimation first. We focus on the main equations here, further details can be found in the original article [9].

The core probability of two haplotype paths  $h^{\text{mat}}$  and  $h^{\text{pat}}$  forming the current target genotypes  $g$  up to a site  $m$  is defined as:

$$P(h_{1\dots m}^{\text{mat}}, h_{1\dots m}^{\text{pat}} | g_{1\dots m}) \approx \underbrace{P(g_{1\dots m} | h_{1\dots m}^{\text{mat}}, h_{1\dots m}^{\text{pat}})}_{\varepsilon^{n_{\text{err}}}} P(h_{1\dots m}^{\text{mat}}) P(h_{1\dots m}^{\text{pat}}) \quad (1)$$

$n_{\text{err}}$  is the number of consistency errors between the pair of haplotype paths to the genotypes, e.g. if both paths show a common allele at a site (homozygous) while the genotype should be heterozygous at that site, this counts as an error.  $\varepsilon$  is a fixed error probability set to 0.003 per default.

The haplotype path probability is assumed to be:

$$P(h_{1\dots m}) \approx \sum_{x=m-H}^{m-1} P(h_{1\dots x}) f(h_{x+1\dots m}) P(\text{rec } m|x) \quad (2)$$

The parameter  $H$  indicates the size of a “history”, i.e. how many sites should be looked back at in order to calculate the actual probability of the current site  $m$ . This is set to 30 for the fast pre-phasing step and set to 100 in the fine phasing steps.  $P(\text{rec } m|x)$  is the recombination probability between the two sites  $m$  and  $x$ . It is computed as a function dependent on the genetic distance between the sites, the effective population size and the number of references (see [9] for details).

The function  $f(h_{x\dots m})$  returns the frequency of the given sequence  $h_{x\dots m}$  in all reference sequences from sites  $x$  to  $m$ . Since this function is called  $H$  times for each site and for each active haplotype path in the beam search at that site, it is crucial to get this information as fast as possible. The HapHedge data structure in combination with the PBWT allow to access this information in approximately constant time (see [9] and [7] for details) while a naive implementation would be linear in the number of references.

### 2.3 Beam Search

In order to estimate the phase on a heterozygous target site, EAGLE2 computes the probability to keep the phase ( $P_m^{\text{keep}}$ ) at such a site  $m$ . For this purpose, it performs a beam search. It starts at the first heterozygous site with a probability of 0.5 and creates two consistent haplotype paths up to that site. This pair of haplotype paths is called a *diplotype* by Loh et al. The process continues over all heterozygous sites and extends each haplotype path in a diplotype by 0 and 1, i.e. for each diplotype, it creates four new diplotypes with the corresponding extension. A fixed parameter  $P$  for the beam width controls the maximum number of diplotypes in the beam search, i.e. only the  $P$  most probable diplotypes (according to Eq. 1) are kept in the beam.  $P$  is set to 30 in the fast pre-phasing step and set to 50 during fine phasing. Loh et al. use a forward extension before doing the final phase call for a site. The beam front is extended over a number of  $\Delta$  heterozygous sites in advance before doing the phase call on the current site.  $\Delta$  is set to 10 in the fast pre-phasing and set to 20 in the fine phasing steps. The keep phase probability is now easily computed as the sum of the probabilities of those diplotypes that keep their phase at the current heterozygous site in comparison to the last heterozygous site divided by the sum of all probabilities from diplotypes that are heterozygous at the current site.

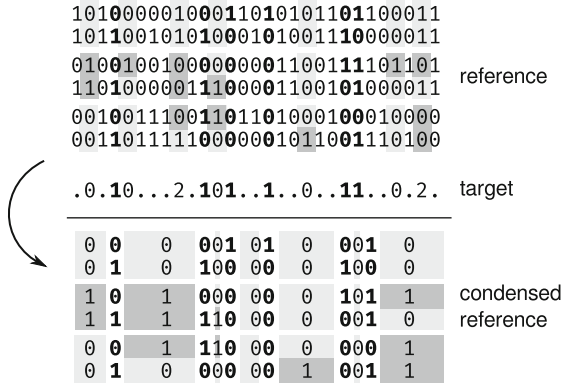
$$P_m^{\text{keep}} = \frac{\sum \{P(h_{1\dots m}^{\text{mat}}, h_{1\dots m}^{\text{pat}} | g_{1\dots m}) \mid \text{diplotype keeps phase}\}}{\sum \{P(h_{1\dots m}^{\text{mat}}, h_{1\dots m}^{\text{pat}} | g_{1\dots m}) \mid h_m^{\text{mat}} \neq h_m^{\text{pat}}\}} \quad (3)$$

In the final phase call, the phase will be switched if that probability is less than 0.5. Note, that the extension of the diplotype may also be homozygous and thus, its probability has to be ignored for the current phase call.

EAGLE2 performs further optimizations on the beam search, such as merging similar diplotypes. We omit explanations here and refer to [9] again.

### 2.4 Condensed Reference

Phase calling is only necessary for heterozygous target sites since the maternal and the paternal haplotype paths carry the same information at homozygous sites. For this reason a *condensed reference* is created from the original reference panel for each target. The condensed reference carries only information of the reference samples at the target's heterozygous sites (and some other split sites in areas of large homozygosity) and combines the information on segments between



**Fig. 1.** Process of creating a condensed reference from a series of reference sequences. Heterozygous target sites are highlighted in bold. For homozygous sites inconsistent haplotypes in the reference are highlighted in dark grey, consistent ones in light grey. Consequently, consistent and inconsistent segments in the condensed reference are highlighted likewise.

two split sites to one bit indicating consistency with the target for each reference haplotype.

In order to reduce the computational burden from a large reference panel, a preliminary step of creating a condensed reference is to choose the  $k$  best fitting reference haplotypes from the panel, i.e. the  $k$  haplotype sequences with the lowest number of consistency errors when compared to the actual target.  $k$  is a user-defined parameter that defaults to 10,000. Thus, in the default configuration, this step is only necessary when choosing a large reference panel with more than 5,000 samples (i.e. 10,000 haplotypes), such as the HRC1.1 panel [13].

The second step is to copy the information from the reference only at split sites. Since the haplotypes are encoded in sequences of 0 and 1 the size is one bit per reference and split site. Another bit is used to encode the condensed information of the following segment. In particular, 0 encodes a consistent segment and 1 encodes an inconsistent segment, i.e. at least one consistency error is found between this haplotype sequence and the target genotypes in this segment. So, the condensed reference is represented as a bit array with a bit sequence for every reference where each odd bit represents a heterozygous site of the target and its reference information, and each even bit (the *inconsistency bit*) represents the segment between two heterozygous sites and the information if the segment is consistent with the target or not. Note, that the frequency lookup  $f(h_{x...m})$  in Eq. 2 requires this information for counting sequences in the condensed reference. Only sequences over consistent segments will be counted. Figure 1 depicts the process of condensing the reference.

A third step added to the creation of the condensed reference is to check reference samples for *Identity-by-descent (IBD)*. Regions of a sample are considered to be in IBD with the target if the region is large enough and the reference sample

and the target are exactly equal in that region. If an IBD situation is encountered the program would generate a bias towards this sample. So, the region in this sample is excluded in the analysis by setting the corresponding inconsistency bits to 1 in the condensed reference. EAGLE2 distinguishes between two types of IBD. If a region of 10 to 20 split sites is found to be equal to the target only the region in the reference haplotype with the latest error will be masked. If the equality spreads over more than 20 split sites, both haplotype sequences belonging to the reference sample will be masked in that region.

### 3 Implementation

Due to the large amount of data that has to be processed, our analyses revealed that in normal use cases the process to create a condensed reference for each target already consumes around 50% of the runtime. We targeted this part of the haplotype phasing for acceleration on reconfigurable architectures, in particular FPGAs. Our computing architecture consists of a server-grade mainboard hosting two Intel Xeon E5-2667v4 8-core CPUs @ 3.2 GHz and 256 GB of RAM, and an Alpha Data ADM-PCIE-8K5 FPGA accelerator card equipped with a recent Xilinx Kintex UltraScale KU115 FPGA with two attached 8 GB SODIMM memory modules, connected via PCI Express Gen3 x8 allowing high-speed communication with the host. The system runs a Ubuntu 19.04 Linux OS (Kernel version 5.0).

#### 3.1 Target Preparation on the Host System

Firstly, the host system loads the input data, i.e. the target data and the reference data into memory. VCF files are stored in a variant major format, i.e. one line of a file contains all haplotype/genotype information of the included samples for one variant. Both files are read concurrently variant after variant, and only bi-allelic sites common in target and reference are kept for the analysis. The data is implicitly transposed to sample-major as required for the target pre-processing.

Since all targets can be handled independent of each other, EAGLE2 spawns multiple threads with each thread phasing a single target at a time. The number of threads running in parallel is defined by a user parameter. In the case of the FPGA acceleration, we prepared the FPGA design to handle several targets concurrently as well. Our design is capable of creating the condensed reference for 48 targets in parallel. For each bunch of 48 targets, the genotype data is encoded in two bits per genotype and aligned in 256 bit words (the PCIe transmission bus width of the FPGA). The host allocates page-locked transmission buffers to ensure fast data transfers between host and FPGA. The target data is copied into the buffers and transferred to the FPGA. For initialization of the 48 processing pipelines, the host determines target specific constants. These are the number of split sites (mainly heterozygous sites) in each target as well as the pre-computed size of the condensed reference. Since the size of the reference is too large to be kept in local FPGA RAM in general, this data is prepared as a compact bit

array in page-locked memory on the host. The haplotype data is encoded in one bit per haplotype and each haplotype sequence is again aligned to the FPGA's PCIe transmission word width of 256 bit. The process of creating the condensed reference of the current targets starts with streaming the reference data to the FPGA.

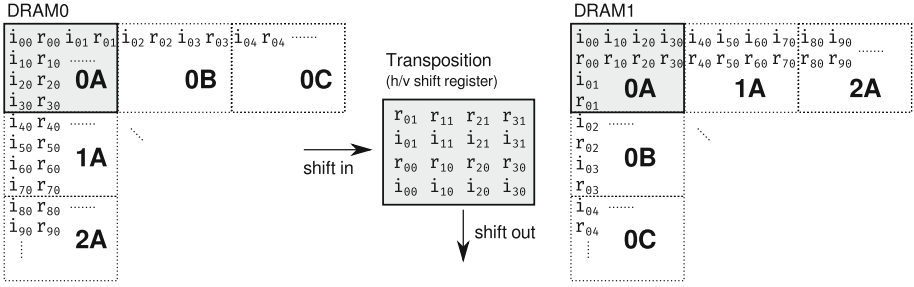
### 3.2 Creating the Condensed Reference on the FPGA

The central part of the FPGA design contains 48 parallel processing pipelines, each producing a stream of condensed reference data for each target. One unit controls the pipeline input. It distributes the raw target data from the host to the corresponding pipeline which stores it in a local buffer implemented in the FPGA's *block RAM (BRAM)*. When the host starts sending the reference haplotype data, the first haplotype sequence is stored in a local BRAM buffer. The second sequence is directly streamed together with the first sequence from the buffer through all pipelines at the same time such that in each clock cycle a pair of haplotypes (one bit maternal and one bit paternal) is processed in the pipeline. This pair of haplotypes forms a phased genotype from the reference. It is directly compared with the corresponding unphased genotype in the target in the *checkIncon* unit. The unit produces two streams (one for the maternal reference and one for the paternal reference) of condensed data, i.e. as long as the target genotypes are homozygous it only records an inconsistency to the corresponding reference by a boolean flag. Whenever the target shows a heterozygous site, *checkIncon* generates a 2-bit output for each stream consisting of the current inconsistency flag and the current reference haplotype at that site.

The resulting pair of 2-bit condensed reference data continues to the *checkIBD* unit. The unit keeps a buffer for the data of the last 64 split sites in a simple shift register implemented in logic. This way, we can set up to 64 preliminary sites to inconsistent in one clock cycle if an IBD situation occurs (which would add a certain bias towards the current sample, see Sect. 2.4). The actually applied numbers depend on the type of IBD and can be set by the host via a runtime constant. For detection of an IBD situation the unit simply counts the number of past pairwise (maternal and paternal) consistent sites, and it keeps a flag that marks the sequence where the last inconsistency occurred. After at least 10 consistent sites (first IBD type) the unit marks the sequence with the last error as inconsistent along these sites. If the number of consistent sites extends to 20 (second IBD type) the unit marks both sequences along these sites as inconsistent. This process continues until the first inconsistent site is reported from the *checkIncon* unit.

All pipeline outputs are connected via a simple shared bus system with a 512-bit word width transferring the condensed data directly to the first DRAM module. In order to convert the data stored in sample-major format to variant-major, as required by the phasing process on the host system (see Sect. 3.3), we designed a data transmission unit that transposes blocks of  $512 \times 512$  bit data. It is implemented as a shift register that is able to shift data horizontally and vertically. The data is organized in a mesh of  $512 \times 512$  cells that are





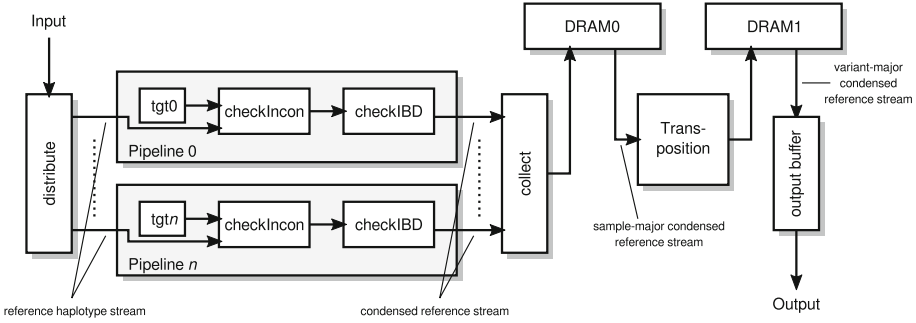
**Fig. 2.** Example of data transposition. The transposition unit represents a horizontal and vertical shift register. The condensed reference in the first DRAM module is transposed block-wise and stored in the second DRAM module.  $r_{nm}$  refers to the reference allele of haplotype  $n$  at call site  $m$ ,  $i_{nm}$  refers to the inconsistency bit from haplotype  $n$  for the segment between call sites  $m - 1$  and  $m$ .

each connected with its horizontal and vertical neighbors. So, 512 sample-major RAM words from 512 subsequent samples shifted horizontally into the mesh can be read out vertically in variant-major format afterwards, such that each word contains the information of 512 samples for one site. In particular, since reference information and inconsistency information is stored in alternating bits for each sample, the transposed output contains alternating words for reference and inconsistency information for subsequent segments and sites. The output is stored in the second DRAM module from where it can directly be fetched from the host (via an output buffer).

The transposition unit also controls the way the condensed reference is distributed in blocks, i.e. the source addresses in the first DRAM module of the 512 words to be shifted into the unit. And it calculates the destination addresses in the second DRAM module of the output words, such that subsequent words belong to the same variant and the host can directly read variant-major sequences containing all information on the samples for one variant in one sequence. See Fig. 2 for an example. The complete FPGA design is illustrated in Fig. 3.

### 3.3 Final Target Processing on the Host System

For target phasing we adapted the procedure of EAGLE2 described in Sect. 2. However, it is worth to mention that in the original EAGLE2 software the condensed reference information is kept in sample-major format. Thus, for creating the *HapHedge* data structure the information is read “vertically”, i.e. the current site information is picked from each corresponding sample words by bit-masking. We omitted this time-consuming implicit transposition by directly providing transposed data from the FPGA. The rest of the process is implemented as explained in the supplementary material of the original article [9].



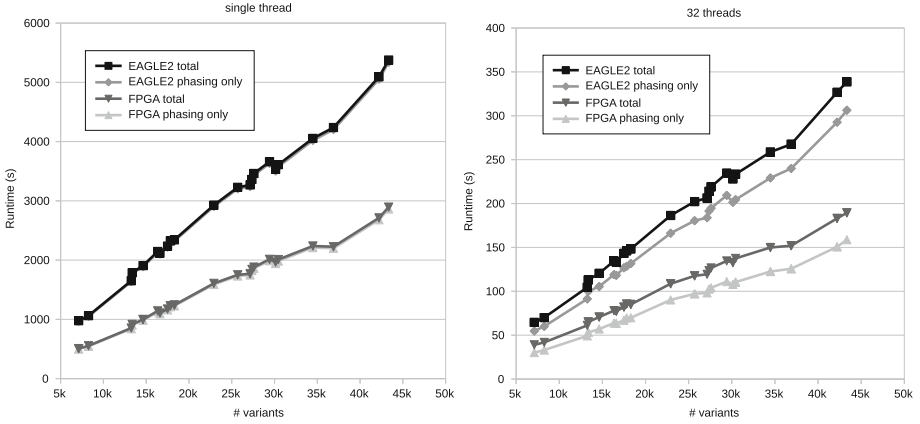
**Fig. 3.** FPGA design for creating the condensed reference. Several targets are handled concurrently in separate pipelines. The streamed reference data is continuously compared to the targets and the condensed information including IBD check is extracted. The raw sample-major condensed data is temporarily stored in the first local DRAM module before it is transposed to variant-major. The transposition output is stored in the second local DRAM module before it is transferred to the host, where it can directly be used for the phasing process.

## 4 Performance Evaluation

We performed a quality check and a runtime evaluation on our architecture from Sect. 3 as described in the following. In all tests we used the binary packed *variant call format* (.bcf) for input and output data, and ran the benchmark without an additional system load.

### 4.1 Phasing Quality

We measure phasing quality by calculating the mean *switch error rate*, i.e. we phase a number of targets with already known phase and compare the output to the original afterwards by counting *switch errors*, i.e. the number of phase switches different to the original. We chose the publicly available 1000 Genomes reference panel [12], containing more than 31 million phased markers from 2504 unrelated diploid samples of different ancestry. To create a real-world oriented test data set, we created chromosome-wise subsets with all samples from the panel but reduced the markers to those that can also be found on the *Global Screening Array (GSA)* [8], resulting in about 520,000 quality-controlled markers in total. We used the software *bcftools* [15] for creating the test data.



**Fig. 4.** Runtimes (in s) of single-threaded (left) and multi-threaded (32 threads, right) EAGLE2 and FPGA-accelerated runs. The graphs separate total and phasing-only runtimes for each run.

Firstly, we phased the test data against the original 1000 Genomes Project reference panel using the EAGLE2 software in its currently latest build, version 2.4.1 [10]. Note, that although the reference and the target data contain the same samples, we did not create a bias towards copying the phase information from the original sample because the IBD check (see Sect. 2.1) ensures that for each targeted sample the same sample in the reference is masked out. So, each target sample was phased using only the phase information of all other samples. We ran the phasing fully parallelized with 32 threads (`--numThreads 32`) and with a single thread (`--numThreads 1`). The EAGLE2 software set the number of phasing iterations automatically to 1, other parameters were left as default. The EAGLE2 software reports its total runtime in its output log divided into reading, phasing and writing time. For evaluation, we used the total runtime and the phasing runtime.

Secondly, we phased the test data against the 1000 Genomes Project reference panel again using our FPGA-accelerated version of EAGLE2. As before, we performed the phasing twice, once fully parallelized with 32 threads and once with only a single phasing thread. We reported the runtimes distributed by phasing and IO in the tool using the C standard library. As for the EAGLE2 run, we ran a single phasing iteration.

Finally, we counted the switch errors in the output data sets by comparing them to the original reference. As a result, we counted an average of 4491.5 genome-wide switch errors per sample for the EAGLE2 output. Our FPGA acceleration created 4528.8 switch errors on average, which is slightly larger. Taking the 520,789 variants into account, we computed mean switch error rates of 0.00862 and 0.0087 respectively. Concluded, our FPGA acceleration of EAGLE2 phasing maintains the mean switch error rate with a negligible difference.

## 4.2 Phasing Runtime

In the previous section (Sect. 4.1, we recorded the phasing and total runtimes for each chromosome-wise subset for single- and multi-threaded (32 threads) original EAGLE2 and FPGA-accelerated runs. The total runtimes of the complete genome-wide input dataset with 520,789 variants from 2504 samples was 17 h and 53 min for the original EAGLE2 single-threaded run (17 h 44 min phasing only). The single-threaded FPGA-accelerated run was measured with 9 h and 37 min (9 h 31 min phasing only). This leads to a total speedup of 1.86.

The speedup for the multi-threaded run is similar. With 32 phasing threads on our 2x Intel Xeon computing system the total runtime of the original EAGLE2 run was measured with 1 h and 8 min (1 h 1 min phasing only) and 39 min (32 min phasing only) for the FPGA-accelerated run. This time, the difference between total and phasing only runtimes has an impact on the calculated speedup, which dropped to 1.73 due to IO. Without IO it results to 1.88.

As we described in Sect. 3, the FPGA acceleration targets only 50% of the total phasing process. Thus, according to Amdahl’s Law [1], we have not expected a speedup exceeding 2 for the phasing. In order to quantify the speedup for the part which was accelerated by the FPGA alone, we introduced a time measure in our single-threaded run of that part which was not accelerated ( $t_{\text{notacc}}$ ). We assumed this part to be nearly the same as in the original EAGLE2 run. So we calculated our speedup of the FPGA part by putting the differences to the complete phasing only times from the original run ( $t_{\text{PhEAGLE}}$ ) and the FPGA accelerated run ( $t_{\text{PhFPGA}}$ ) into relation as described in Eq. 4. The resulting speedup factor of the FPGA-only part for the complete data set is 19.99 whereby we observed a speedup of 29.02 for chromosome 21.

$$\text{Speedup}_{\text{FPGA}} = \frac{t_{\text{PhEAGLE}} - t_{\text{notacc}}}{t_{\text{PhFPGA}} - t_{\text{notacc}}} \quad (4)$$

Additionally, the introduced time measure of the not accelerated part allowed us to exactly identify the ratio of this part to the complete phasing, and in reverse conclusion, the ratio of the part accelerated by the FPGA. For the complete data set we calculated this ratio to be 48.8%, and thus, according to Amdahl’s Law, the theoretical maximum speedup of the phasing part to be  $1/(1-0.488) = 1.95$ , which we have almost reached.

Figure 4 shows the runtimes of the single- and multi-threaded EAGLE2 runs as well as for the single- and multi-threaded FPGA-accelerated runs plotted over the number of phased variants from each input subset. The graphs show a nearly linear behaviour as expected. Tables 1 and 2 show the runtimes and speedups for selected chromosome-wise subsets of our input data including the worst and best runs.

**Table 1.** Wall-clock runtimes (in s) and speedup of FPGA-accelerated EAGLE2 phasing using a single thread only.  $t_{\text{notacc}}$  is the time for the phasing part which was not accelerated by the FPGA. Speedup is shown for the total runtime and for the part accelerated by the FPGA according to Eq. 4. Total runtimes include phasing and data IO. The table shows selected chromosome-wise subsets,  $\Sigma$  is the sum of all 22 subsets.

Dataset		EAGLE2		FPGA accel.			% accelerated	Speedup	
chr	# variants	Total	Phasing	Total	Phasing	$t_{\text{notacc}}$	by FPGA	Total	FPGA
chr1	43344	5376.24	5337.55	2893.89	2863.74	2729.07	48.9	1.86	19.37
chr2	42210	5093.09	5052.83	2712.61	2680.65	2551.46	49.5	1.88	19.36
chr6	36892	4237.13	4203.38	2225.90	2200.34	2080.43	50.5	1.90	17.70
chr10	27360	3362.27	3335.68	1844.03	1823.93	1739.58	47.8	1.82	18.92
chr13	17819	2323.75	2303.55	1227.39	1212.45	1159.92	49.6	1.89	21.77
chr17	16612	2119.15	2101.84	1107.58	1094.92	1049.73	50.1	1.91	23.28
chr21	7107	977.28	966.60	506.71	499.26	482.58	50.1	1.93	29.02
chr22	8262	1062.41	1051.59	555.09	546.32	527.11	49.9	1.91	27.30
$\Sigma$	520789	64366.04	63832.30	34636.55	34231.12	32672.01	48.8	1.86	19.99

**Table 2.** Wall-clock runtimes (in s) and speedup of FPGA-accelerated EAGLE2 phasing using 32 threads on a system with two Intel Xeon E5-2667v4 and Xilinx Kintex UltraScale KU115 FPGA. Speedup is shown for the total runtime and for the phasing part only. Total runtimes include phasing and data IO. The table shows selected chromosome-wise subsets,  $\Sigma$  is the sum of all 22 subsets.

Dataset		EAGLE2		FPGA accel.		Speedup	
chr	# variants	Total	Phasing	Total	Phasing	Total	Phasing
chr1	43344	338.71	306.30	189.41	158.70	1.79	1.93
chr2	42210	326.69	292.64	183.05	150.70	1.78	1.94
chr6	36892	267.58	239.95	151.93	125.65	1.76	1.91
chr10	27360	213.93	191.44	123.52	102.48	1.73	1.87
chr13	17819	146.11	128.47	85.46	69.65	1.71	1.84
chr17	16612	133.18	118.17	77.27	63.46	1.72	1.86
chr21	7107	64.54	54.71	39.03	30.16	1.65	1.81
chr22	8262	70.05	60.02	41.78	33.07	1.68	1.81
$\Sigma$	520789	4093.11	3635.05	2360.24	1935.15	1.73	1.88

## 5 Conclusions and Future Work

With around 50% of the total runtime, data preparation for reference-based phasing with EAGLE2 [9] is the most time consuming part of the software. In this paper, we showed how to accelerate this part by a factor of up to 29 with the help of reconfigurable hardware, leading to a total speedup of almost 2 for the complete process, which is the theoretical maximum according to Amdahl’s Law [1]. While preserving the switch error rate of the original software, we reduced genome-wide phasing of 520,000 standard variant markers in 2500 samples from more than 68 min to 39 min on a server-grade computing system.

The single-threaded runtime was reduced in the same manner from more than 18 h to 9.5 h. We used the commonly used and publicly available 1000 Genomes Project [12] phased haplotypes as reference panel.

We are already examining the possibilities of harnessing GPU hardware for supporting the final phasing step by creating the PBWT data structure, which is the second most time consuming step in the phasing process. Furthermore, since phasing is generally only used as preliminary process for genotype imputation, introducing hardware support for corresponding software, such as PBWT [7] or minimac4 [5], as they are used in the Sanger or Michigan Imputation Services [14, 16], is our next goal.

## References

1. Amdahl, G.M.: Validity of the single processor approach to achieving large scale computing capabilities. In: Proceedings of the April 18–20, 1967, Spring Joint Computer Conference, AFIPS 1967 (Spring), pp. 483–485. ACM, New York (1967). <https://doi.org/10.1145/1465482.1465560>
2. Browning, S.R., Browning, B.L.: Haplotype phasing: existing methods and new developments. *Nat. Rev. Genet.* **12**, 703–714 (2011). <https://doi.org/10.1038/nrg3054>
3. Chen, G.K., Wang, K., Stram, A.H., Sobel, E.M., Lange, K.: Mendel-GPU: haplotyping and genotype imputation on graphics processing units. *Bioinformatics* **28**(22), 2979–2980 (2012). <https://doi.org/10.1093/bioinformatics/bts536>
4. Das, S., Abecasis, G.R., Browning, B.L.: Genotype imputation from large reference panels. *Annu. Rev. Genomics Hum. Genet.* **19**, 73–96 (2018). <https://doi.org/10.1146/annurev-genom-083117-021602>
5. Das, S., Forer, L., Schönherr, S., et al.: Next-generation genotype imputation service and methods. *Nat. Genet.* **48**, 1284–1287 (2016). <https://doi.org/10.1038/ng.3656>
6. Delaneau, O., Zagury, J.F., Marchini, J.: Improved whole chromosome phasing for disease and population genetic studies. *Nat. Methods* **10**, 5–6 (2013). <https://doi.org/10.1038/nmeth.2307>
7. Durbin, R.: Efficient haplotype matching and storage using the positional Burrows-Wheeler transform (PBWT). *Bioinformatics* **30**(9), 1266–1272 (2014). <https://doi.org/10.1093/bioinformatics/btu014>
8. Illumina: Infinium Global Screening Array-24 Kit. <https://www.illumina.com/products/by-type/microarray-kits/infinium-global-screening.html>
9. Loh, P.R., Danecsek, P., Palamara, P.F., et al.: Reference-based phasing using the haplotype reference consortium panel. *Nat. Genet.* **48**, 1443–1448 (2016). <https://doi.org/10.1038/ng.3679>
10. Loh, P.R., Price, A.L.: EAGLE v2.4.1, 18 November 2018. <https://data.broadinstitute.org/alkesgroup/Eagle/downloads/>
11. Tewhey, R., Bansal, V., Torkamani, A., Topol, E.J., Schork, N.J.: The importance of phase information for human genomics. *Nat. Rev. Genet.* **12**, 215–223 (2011). <https://doi.org/10.1038/nrg2950>
12. The 1000 Genomes Project Consortium: A global reference for human genetic variation. *Nature* **526**, 68–74 (2015). <https://doi.org/10.1038/nature15393>

13. The Haplotype Reference Consortium: A reference panel of 64,976 haplotypes for genotype imputation. *Nat. Genet.* **48**, 1279–1283 (2016). <https://doi.org/10.1038/ng.3643>
14. US National Institutes of Health: Michigan Imputation Server. <https://imputationserver.sph.umich.edu/>
15. Wellcome Sanger Institute: SAMtools/BCFtools/HTSlib. <https://www.sanger.ac.uk/science/tools/samtools-bcftools-htslib>
16. Wellcome Sanger Institute: Sanger Imputation Service. <https://imputation.sanger.ac.uk/>
17. Williams, A.L., Patterson, N., Glessner, J., Hakonarson, H., Reich, D.: Phasing of many thousands of genotyped samples. *Am. J. Hum. Genet.* **91**, 238–251 (2012). <https://doi.org/10.1016/j.ajhg.2012.06.013>