Power-Efficiency Analysis of Accelerated BWA-MEM Implementations on Heterogeneous Computing Platforms

Ernst Joachim Houtgast*†, Vlad-Mihai Sima†, Giacomo Marchiori†, Koen Bertels* and Zaid Al-Ars*
*Computer Engineering Lab, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands
†Bluebee, Laan van Zuid-Hoorn 57, 2289 DC Rijswijk, The Netherlands
E-mail: {ernst.houtgast, vlad.sima, giacomo.marchiori}@bluebee.com, {k.l.m.bertels, z.al-ars}@tudelft.nl

***Abstract*—Next Generation Sequencing techniques have dramatically reduced the cost of sequencing genetic material, resulting in huge amounts of data being sequenced. The processing of this data poses huge challenges, both from a performance perspective, as well as from a power-efficiency perspective. Heterogeneous computing can help on both fronts, by enabling more performant and more power-efficient solutions.**

In this paper, power-efficiency of the BWA-MEM algorithm, a popular tool for genomic data mapping, is studied on two heterogeneous architectures. The performance and power-efficiency of an FPGA-based implementation using a single Xilinx Virtex-7 FPGA on the Alpha Data add-in card is compared to a GPU-based implementation using an NVIDIA GeForce GTX 970 and against the software-only baseline system. By offloading the Seed Extension phase on an accelerator, both implementations are able to achieve a two-fold speedup in overall application-level performance over the software-only implementation. Moreover, the highly customizable nature of the FPGA results in much higher power-efficiency, as the FPGA power consumption is less than one fourth of that of the GPU. To facilitate platform and tool-agnostic comparisons, the base pairs per Joule unit is introduced as a measure of power-efficiency. The FPGA design is able to map up to 44 thousand base pairs per Joule, a 2.1x gain in power-efficiency as compared to the software-only baseline.

***Keywords*- FPGA; GPU; Next Generation Sequencing; power-efficiency; read mapping**

I. Introduction

Next Generation Sequencing (NGS) techniques dramatically decrease the cost of sequencing genetic material. The cost to sequence one complete human genome is rapidly approaching the important $1,000 mark [1]. As a result of these falling costs, the production of genetic data is surging and is projected to rival, if not overtake, other Big Data fields such as streaming video services and astronomy [2]. A single run on a state-of-the-art X Ten sequencing machine [3] can generate up to 1.2 TB of data, which in turn requires multiple days to process, even on a large computing cluster. The extreme scale of data and tremendous computing efforts involved in processing this data necessitates the use of high performance computing solutions to face this challenge. Heterogeneous computing, and in particular reconfigurable computing, offers a large promise as a solution that enables both high performance and power-efficiency compared to traditional computing techniques. Power-efficiency is becoming at least as important as raw performance, as power consumption is an important driver to overall data center cost.

NGS data is typically processed by a complex pipeline of algorithms. In the case of DNA NGS data, the short reads as produced by the sequencer are first mapped onto a reference genome. Then, this output is sorted and duplicates are marked. Finally, mutations in the genetic material as compared to the reference are found during a variant calling stage. Only at this stage does the raw data become usable for further downstream analysis, for example by researchers or medical professionals. To illustrate the immense computational requirements, processing such a data set can easily require multiple days, even on a large cluster.

BWA-MEM is a Burrows-Wheeler Alignment based tool for mapping short reads onto a reference genome [4]. Although many other alignment tools exist (examples include [5], [6]), BWA-MEM is the de facto standard for alignment mapping and is part of the popular BWA-MEM/GATK pipeline, used in organizations around the world [7]. In the example above, BWA-MEM contributes about 36% to the overall processing time, making up a significant portion of the processing time of the entire pipeline. Therefore, it is an important target for acceleration to reduce the overall time, cost and energy of processing NGS data sets.

Similar to other mapping tools, such as [5], BWA-MEM operates using the Seed-and-Extend paradigm (see Figure 1). For each read, seeds, exactly matching subsequences between the read and the reference, are generated. Subsequently, each seed is extended in both directions using an inexact matching algorithm, similar to the popular Smith-Waterman dynamic programming algorithm [8]. The highest scoring extended seed is chosen as final alignment. The Seed Extension phase forms a major bottleneck in the BWA-MEM algorithm, requiring between 30%-50% of total execution time, depending on the computing platform [9], [10]. For example, on the highly multithreaded IBM Power8 platform, the Seed Extension phase requires almost 50% of overall execution time, allowing for an up to two-fold performance improvement, whereas on the Intel Core i7 platform, a performance improvement of up to 1.7x is possible. In this paper, accelerated implementations of the BWA-MEM Seed Extension phase on the Intel Core i7

978-1-5090-3707-0/16/$31.00

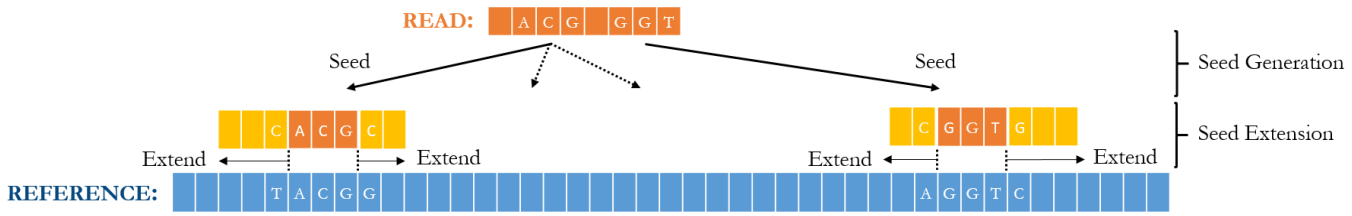

Fig. 1. BWA-MEM processes reads using the Seed-and-Extend paradigm: for each read, likely mapping locations on the reference are found by searching for exactly matching subsequences between the read and the reference (*seeds*). Then, these seeds are extended in both directions using a Smith-Waterman-like dynamic programming approach that allows for inexact matches. From all of these extended seeds, the best scoring alignment is selected.

platform are considered. However, the main idea is generally applicable to other mapping applications as well.

The contributions in this paper include:

- A comparison of both performance and power-efficiency of an FPGA-accelerated implementation to software-only and GPU-based implementations.
- The introduction of the base pairs per Joule (bp/J) unit as measure of power-efficiency, allowing for platform-agnostic comparison of system-level power-efficiency on genomic data sets.

The remainder of this paper is organized as follows. Related work is discussed in Section II. Architectural details of the FPGA and GPU implementations are provided in Section III. The results on performance, scalability and power-efficiency are given in Section IV. Then follows a discussion of the results in Section V. The conclusions are given in Section VI.

II. Related Work

A major part of BWA-MEM execution time is spent in the Seed Extension phase. The inexact mapping algorithm used is similar to the Smith-Waterman algorithm. This algorithm has received much acceleration attention (e.g., [11], [12]). However, for multiple reasons most implementation ideas are not directly applicable to BWA-MEM, the main ones being:

- The highest speedup is generally obtained when performing mapping of very long sequences that contain thousands of bases, whereas NGS reads and BWA-MEM are more focused towards the mapping of short reads of at most a few hundred base pairs.
- For load balancing purposes, these implementations batch alignments of similar length together. In the case of BWA-MEM, this is impractical as the inexact mapping calls are dynamically generated for alignments of varying length, which makes the batching strategy inefficient due to the large communication and temporary data overheads this would require.

Although many accelerated Seed-and-Extend based mapping tools have been proposed (for example [13]), results from these implementations are not directly comparable. In bioinformatics, exactness of results is critical, as larger population studies can take several years to complete and intermediate results need to be comparable. Even a change in version is often unacceptable. Note that BWA-ALN, for which accelerated implementations do exist, is a different algorithm.

A kernel-level acceleration effort specific to the BWA-MEM algorithm is reported in [14], where one of the BWA-MEM kernels, the inexact matching phase, has been accelerated on an FPGA for an up to 26x kernel-level speedup. However, due to the above mentioned reasons, only the kernel-level speedup is reported, and no overall application-level speedup is mentioned. In our experience, accommodating the kernel implementation into a full application is far from a trivial task. The authors acknowledge that the significantly varied input data would pose a challenge for the Smith-Waterman algorithm, but disregard the additional challenges a full-application implementation needs to face.

To our knowledge the only application-level accelerated integrated implementations of BWA-MEM that exist are: an FPGA-accelerated implementation of the Seed Extension phase [15] achieving a 1.5x speedup, further improved in [16] for an overall 2.6x speedup; and a GPU implementation [9], further improved to achieve an up to 2x speedup [17]. The FPGA implementation used here builds on [15], and a comparison of the implementation here is made to the improved GPU implementation. This paper focuses on power-efficiency, besides overall application performance, as for many scenarios, such as processing in a large scale data center, this is at least as important as absolute performance.

III. Architecture Design and Implementation

In this section, first the BWA-MEM algorithm is briefly described, along with the features that both the FPGA and GPU implementations share. Then, the details of the FPGA-accelerated implementation on the Alpha Data card are given. Finally, details of the GPU implementation are briefly discussed (further details can be found in [17]).

The original BWA-MEM algorithm operates in a serial fashion (refer to Figure 2). The input is processed in batches of reads, that are processed one-by-one by two major kernels: Seed Generation, where seeds (exactly matching subsequences) are generated for each read, and Seed Extension, where the generated seeds are extended allowing for inexact matches. This process repeats itself until the input is exhausted. These phases take full advantage of multithreading, as reads are completely independent from each other and can be processed in parallel. To improve the utilization of system resources when using an accelerator, the BWA-MEM algorithm has been reorganized into a fully pipelined organization.

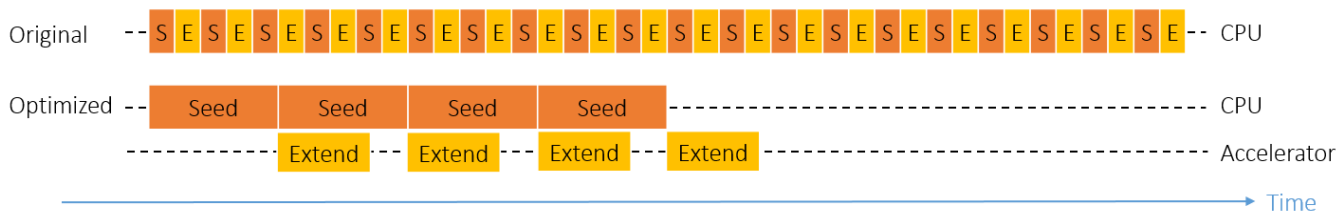

Fig. 2. BWA-MEM processes the input reads one-by-one. For each read, the seeds are generated and then, these seeds are extended. To reduce communication overhead, the accelerated implementations described here use an optimized program architecture, whereby work of multiple reads is batched together. First, seeding is performed for a large group of reads. Then, such a batch is offloaded onto an accelerator that performs the Seed Extension. Its execution is overlapped with work on the CPU. As long as the Seed Extension phase requires less time on the accelerator than the Seeding on the CPU, its execution time is effectively "hidden".

Multiple reads are processed in groups, and the two phases are executed in parallel and are overlapped. Thus, Seed Generation executes simultaneously on the CPU with Seed Extension on the accelerator, resulting in a large performance improvement.

A. FPGA Design and Implementation

The FPGA-accelerated BWA-MEM offloads the Seed Extension phase onto hardware (refer to Figures 1 and 2). After seeds have been generated, they are transferred to the FPGA on-board memory. Typically, a single read will result in multiple seed locations on the reference genome being found. Seeds that are mapped close together on the reference genome are grouped into chains. Seeds are processed one-by-one. Seed Extension for a seed can be skipped depending on the result of previous extensions. On average only one seed per chain requires extension. Moreover, the length of the extension to be performed is dependent on the part of the read that forms the seed. This interdependence between executions makes this phase unsuitable for streaming and, therefore, the FPGA performs both this control logic, as well as the inexact mapping algorithm itself. The resulting mappings are stored in on-board memory and transferred back to the host system.

The Seed Extension module design is based on the design in [15]. The inexact mapping is similar to the Smith-Waterman algorithm. To find the optimal mapping, a 2D similarity matrix is filled. This is implemented as a systolic array, with anti-diagonals of the matrix being calculated in parallel. Each cycle, one processing element (or PE) of the systolic array computes a value of the 2D similarity matrix. Hence, the execution time is reduced from O(M $\times$ N) to O(M+N), where M and N are the length of the reference and the read. This is the reason why certain Smith-Waterman implementations, at least for longer alignments, are able to achieve speedups of several magnitude. In this paper, only short reads of up to 150 base pairs are considered, which is the read length of contemporary sequencers, such as the Illumina HiSeq X. The minimum seed length for BWA-MEM is 19 symbols. Therefore, each inexact mapping engine contains 131 PEs. To improve utilization for shorter reads, *early exit points* as described in [15] are implemented.

The implementation described here uses an Alpha Data add-in card with a single Xilinx Virtex-7 FPGA (details can be found in Section IV). A floorplan of the design is shown in Figure 3. The design contains six modules that are able to process the Seed Extension phase. Within each module, the larger block contains the systolic array logic, the smaller block is filled with control logic. Between the six modules resides the logic that distributes reads over the modules and is responsible for I/O. The rest of the area is taken up by interconnection-related logic, such as the PCI-Express interface and the memory controller.

A significant difference to the design in [15] and [16] is the fact that the Alpha Data card used here contains only a single Virtex-7 FPGA, whereas [15] and [16] use the Convey HC-2^{EX} as implementation platform, which contains four user-configurable Virtex-6 FPGAs. As the design here is limited by the amount of LUTs available, and the Virtex-7 FPGA on the Alpha Data card contains 432,368 LUTs versus 474,240 LUTs per Virtex-6 FPGA on the Convey, this means only about 23% of the resources are available as compared to the Convey platform. This, in turn, requires a careful selection to place only those modules on the FPGA that most benefit from acceleration. Hence, the decision was made to only implement the Seed Extension phase in hardware. In total, about 71% of all LUTs is utilized and the Seed Extension modules run at a clock rate of 160 MHz.

B. GPU Implementation

Similar to the FPGA implementation, the GPU implementation, further described in [17], also accelerates the Seed Extension phase. However, in contrast to the FPGA implementation, which contains six Seed Extension modules, the available execution resources on the GPU depend on the actual GPU model being used. A batch of reads is sent to the GPU as a grid of thread blocks, where each read is mapped onto a single block of 32 threads. This ensures scheduling is automatically taken care of by the GPU, with per-read granularity, based on the available execution resources. In the case of the test platform, an NVIDIA GeForce GTX 970, there are thirteen multiprocessors available. Up to 32 thread blocks can be active per multiprocessor at any single time, due to limits on the amount of registers and shared memory available.

Similar to the FPGA implementation, the Seed Extension phase is logically split between the control logic, which loops over all the seeds of a single read, and the actual inexact mapping. This control logic code is executed by a single

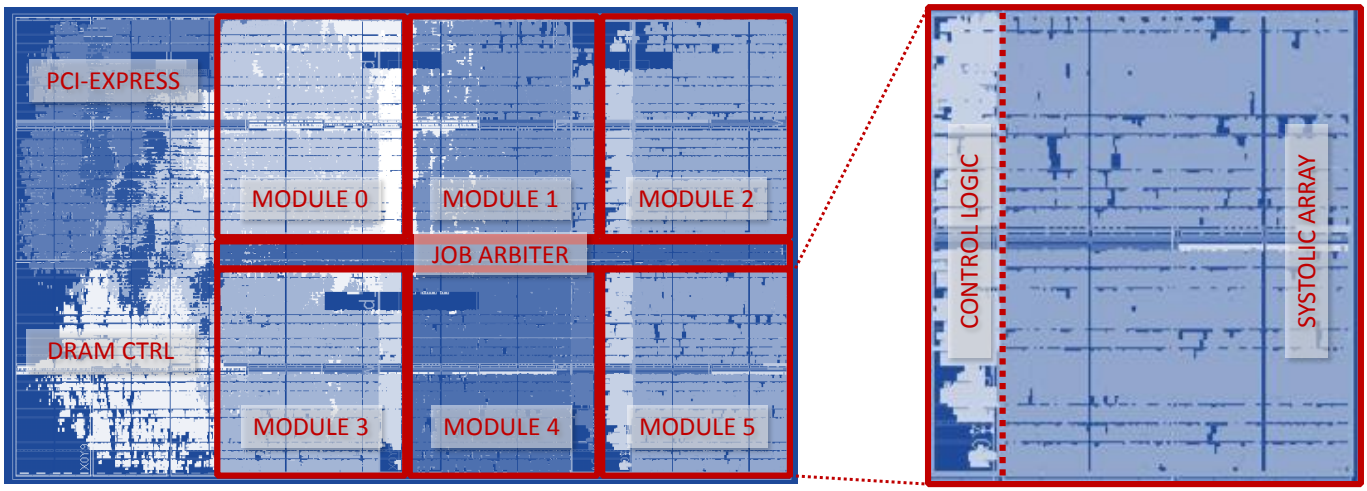

Fig. 3. Floorplan of the FPGA design with six Seed Extension modules. Each module contains control logic to loop over all seeds to be extended, and the Seed Extension systolic array with 131 Processing Elements (see inset). In between the Seed Extension modules resides the arbitrating logic. The remaining area is filled with interconnection-related logic, such as the PCI-Express controller and the DRAM controller.

thread, whereas the inexact mapping code is performed in a systolic array-like manner using 32-PEs at a time. In this case, each CUDA thread acts as a separate PE. Hence, longer extensions require that multiple passes are made over the similarity matrix. The choice to use only 32-threads at a time allows for the use of the intra-warp shuffle functionality, which allows CUDA threads to access each other's registers, to implement data transfer between the PEs, eliminating the need of temporary storage of row data while calculating the similarity matrix elements.

IV. Experimental Results

Tests have been performed using a system with an Intel Core i7-4790 at 3.6 GHz with eight logical cores (four physical cores), SpeedStep and Hyper-Threading enabled. The system contains 16 GB of RAM. Tests were run using CentOS 7.1. To minimize power consumption, no unnecessary services were running. In addition, for the FPGA-accelerated tests, an Alpha Data ADM-PCIE-7V3 card with a Xilinx Virtex-7 XC7VX690T-2 and 16 GB of on-board RAM was added to the system [18]. The Seed Extension modules run at a clock rate of 160 MHz. For the GPU-accelerated tests, an NVIDIA GeForce GTX 970 with 1664 CUDA cores with a maximum clock frequency of up to 1.25 GHz and 4 GB of on-board RAM was added to the system. The GPU is allowed to go into lower power states when idle to conserve power. The software-only results were gathered without either of these cards installed. An emonPi energy measurement unit from OpenEnergyMonitor [19] was used to measure idle and load power utilization of the entire system under test. The current probe was connected directly to the mains power cord to measure system level power consumption, taking hundreds of samples per second.

BWA-MEM version 0.7.8 was used. Tests were performed using publicly available data from the Genome Comparison & Analytic Testing (GCAT) framework [20]. The single-ended alignment (150bp-se-small-indel) and pair-ended alignment (150bp-pe-large-indel) data sets were used. Each data set contains about eight million reads of 150 base pairs, or about 1.2 billion base pairs in total. The reads were aligned against the reference human genome (UCSC HG19). The GCAT online sequence alignment quality comparison service was used to verify that results of the FPGA-accelerated and the GPU-accelerated versions are indistinguishable from those obtained with the software BWA-MEM algorithm.

In the remainder of this section, performance results on all three platform are shown for the single-ended and pair-ended GCAT tests. The scalability of the heterogeneous platforms is investigated to find the optimal balance between host CPU count and accelerator performance. Finally, the power-efficiency of the various platforms is determined.

A. Performance Analysis

The performance, scalability and power-efficiency results have been gathered using the GCAT data sets. Each contains about eight million reads with about 1.2 billion base pairs, for a file size of about 3 GB for the single-ended data set and 2x 1.5 GB for the pair-ended data set. A typical complete Illumina X Ten sequencing run generates a data set about 400x larger, or approximately 1.2 TB. The implementation has been verified to scale up to work without issues on such larger data sets, but, for reasons of practicality, tests have been performed on the smaller data set.

Performance results on the single-ended and pair-ended GCAT data sets are summarized in Table I. The execution time is shown both as time required for the Seed Extension phase on the accelerator hardware, and for the overall application wall clock time. To facilitate cross-platform comparisons, the results are converted into throughput in millions of base pairs per second. Both the FPGA and GPU implementations are able to achieve a two-fold improvement to performance, the FPGA implementation being slightly faster.

TABLE I
EXECUTION TIME AND SPEEDUP FOR THE GCAT ALIGNMENT QUALITY BENCHMARK

		Seed Extension Phase		Overall Application		
Test	**Platform**	**Execution Time**	**Speedup**	**Execution Time**	**Speedup**	**Throughput**
Single-Ended	Software-Only	237 s	-	552 s	-	2.2 Mbp/s
Data	FPGA-Accelerated	129 s	1.8x	272 s	2.0x	4.5 Mbp/s
	GPU-Accelerated	144 s	1.6x	278 s	2.0x	4.3 Mbp/s
Pair-Ended	Software-Only	246 s	-	572 s	-	2.1 Mbp/s
Data	FPGA-Accelerated	130 s	1.9x	289 s	2.0x	4.1 Mbp/s
	GPU-Accelerated	141 s	1.7x	293 s	2.0x	4.1 Mbp/s

On a kernel-level, the Seed Extension phase itself is up to 1.9x faster on the FPGA and up to 1.7x faster on the GPU, as compared to software-only execution. This is equivalent to a speed of about 15 and 14 logical Intel Core i7 cores, respectively. Since the execution time of the accelerated Seed Extension implementations only requires about half the overall application execution time, it is clear that in both cases, the accelerator is not fully utilized. Therefore, both the FPGA and GPU accelerators can be used to accelerate a system that is more powerful than the system as tested here, which only contains eight logical processor cores. This is explored in the next section.

B. Scalability Analysis

From the differences in execution time between the Seed Extension phase and the overall application, as described in Table I, it is clear that the FPGA and GPU implementations are not fully utilized. The time spent by the accelerators in the Seed Extension phase is only about half the overall application execution time. This implies that the accelerator hardware is not busy 100% of the time. Therefore, a faster host system would still be able to be accelerated for the maximum speedup of 2x. In contrast, if the accelerated implementations could not keep up with the host, overall speedup would fall below 2x. In this case, more Alpha Data cards, or more and/or faster GPUs could be used. Hence, the objective is to design a system for which all system resources (the CPU cores and either an FPGA or GPU-accelerator) are fully utilized.

In Table II, the estimated *scalability* of the accelerated platforms is shown, which is expressed in number of logical CPU cores for which the accelerator is able to provide the maximum two-fold speedup. In order to estimate the optimal ratio of logical CPU cores for each accelerated platforms, the following assumptions have been made. It is assumed that overall application times decreases linearly with additional CPU cores. Furthermore, it is assumed that the maximum speedup is achieved as long as the time required for the Seed Extension phase does not exceed overall application time.

TABLE II
SCALABILITY OF THE ACCELERATED IMPLEMENTATIONS

	Execution Time			
Platform	**Seed Ext.**	**Overall**	**Utilization**	**Scalability**
FPGA (4 modules)	171 s	272 s	63%	12.7 cores
FPGA (5 modules)	146 s	275 s	53%	15.1 cores
FPGA (6 modules)	129 s	272 s	47%	16.8 cores
GPU	144 s	278 s	52%	15.4 cores

The scalability results are presented for three different FPGA designs. These designs vary in the number of Seed Extension modules that were placed onto the FPGA logic. Although the time required for the Seed Extension phase decreases significantly when more modules are available, this proves to only have a slight impact on the overall execution time. The reason for this is the fact that Seed Extension performance is already fast enough. In contrast, scalability results are much improved. The performance and power-efficiency results in the other sections all consider the six module FPGA design. This six module design is able to support a host system with up to sixteen logical Intel Core i7 cores at 3.6 GHz, for example the Intel Core i7-5960X. The GPU implementation is able to support a host system with up to fifteen logical cores.

C. Power-Efficiency Analysis

To measure the power-efficiency of the different platforms, an emonPi energy monitor was used to track the system-level power consumption as measured at the power plug. For the power-efficiency tests, only the single-ended data set was used, although the pair-ended data set should yield similar results, given that the execution profile is similar. In order to minimize the idle power draw, the tests were performed without active GUI, and with a bare system with only an HDD, SSD and optical drive present. In case of the accelerated platforms, the respective accelerator card was added to the system.

To present a quick qualitative overview of the results, an example trace of the power consumption for a single test is shown in Figure 4. As expected, the accelerated platforms use more instantaneous power than the software-only system, both under load and when idle. However, they are also able to map reads at a much higher rate, and both finish in about half the time of the software-only implementation. The results are summarized in Table III. This table gives the power consumption, performance and energy efficiency for each platform. Both the measured data is presented, as well

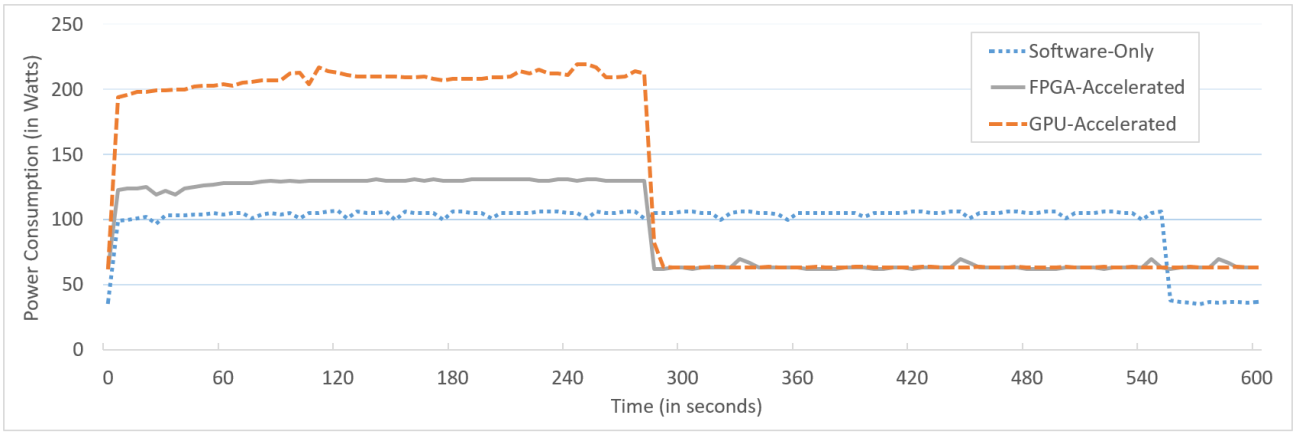

Fig. 4. Example trace of the power consumption over time for the software-only, FPGA-accelerated and GPU-accelerated platforms. The software-only system consumes the least power, both when idle and under load. However, the additional power utilized by the accelerated platforms results in much faster completion of the test, as both implementations finish in much less than three hundred seconds. The FPGA-accelerated platform has the highest power-efficiency.

TABLE III
POWER CONSUMPTION, PERFORMANCE AND POWER-EFFICIENCY FOR SOFTWARE-ONLY AND ACCELERATED PLATFORMS.

			Power Consumption		Performance		Energy Efficiency		
Test	**Platform**	**Logical Cores**	**Idle Power**	**Load Power**	**Execution Time**	**Application Speedup**	**Total Energy**	**Base Pairs Per Energy**	**Efficiency Improvement**
Measured Data	Software-Only	8	37 W	105 W	552 s	-	58 kJ	20.7 kbp/J	-
	FPGA-Accelerated	8	62 W	129 W	272 s	**2.0x**	35 kJ	34.1 kbp/J	**1.6x**
	GPU-Accelerated	8	63 W	210 W	278 s	**2.0x**	58 kJ	20.5 kbp/J	**1.0x**
Estimated Data	Software-Only	15	37 W	164 W	294 s	-	48 kJ	24.8 kbp/J	**1.2x**
	FPGA-Accelerated	15	62 W	188 W	147 s	**2.0x**	28 kJ	43.2 kbp/J	**2.1x**
	GPU-Accelerated	15	63 W	269 W	148 s	**2.0x**	40 kJ	30.0 kbp/J	**1.4x**
	Software-Only	16	37 W	172 W	276 s	-	47 kJ	25.2 kbp/J	**1.2x**
	FPGA-Accelerated	16	62 W	196 W	138 s	**2.0x**	27 kJ	44.1 kbp/J	**2.1x**
	GPU-Accelerated	16	63 W	277 W	144 s	**1.9x**	40 kJ	29.9 kbp/J	**1.4x**

as results for estimated, more well-balanced, systems with a larger number of logical CPU cores. First, the results for the measured data are discussed, afterwards the estimated data.

Similar to the trace shown in Figure 4, as expected the power draw of the software-only system is the lowest, both when idle at 37 W and under load at 105 W. The accelerated platforms show significantly increased idle power consumption, at 62 W for the FPGA platform and at 63 W for the GPU platform, due to the addition of more hardware into the system. Also, both platforms show a higher power consumption under load, at 129 W for the FPGA platform and at 210 W for the GPU platform. However, this is compensated by much improved performance, as each implementation is able to achieve a two-fold speedup as compared to software-only execution. In addition, the FPGA-accelerated implementation is much more power-efficient as compared to the other two platforms: whereas the software-only and GPU-accelerated platforms both require 58 kJ to complete the entire test, the FPGA-accelerated platform requires only 35 kJ, an energy efficiency improvement of 60%. On this system, use of a GPU-accelerator merely provides the user a trade-off between performance and power consumption, as overall power-efficiency remains the same as compared to the software-only platform.

To put the relative power efficiency of the FPGA and the GPU into perspective, consider their respective power consumption when executing the same workload. The difference in load and idle power on the software-only platform implies that the host CPU uses about 68 W of power under load. If it is assumed that the host CPU requires a similar power draw when running the accelerated platforms, then the power consumption under load used exclusively for powering the accelerators can be computed, being 25 W for the FPGA and 105 W for the GPU. Hence, the FPGA power consumption is less than one fourth of the GPU, showing the clear advantage in energy efficiency of the reconfigurable platform. Moreover, the FPGA does not seem to consume any additional power under load, as its logic is always active.

To facilitate comparisons of power-efficiency for mapping a certain data set of reads, the data is also reported as *number of base pairs mapped per Joule of energy*. Using this measure, the power-efficiency for a given data set can be evaluated across platforms, architectures and mapping tools. However, it is important to note that performance of the various tools is not the only measure of interest, as mapping tools can differ greatly in their mapping quality, which is the ability to accurately map reads. As the data set used in these

tests contains about eight million reads, each with 150 base pairs, the resulting power-efficiency for the FPGA-accelerated platform is about 34 kbp/J, a 60% improvement in energy efficiency as compared to the software-only platform.

Based on the scalability results obtained in the previous section, Table III also contains the estimated performance and power-efficiency for two more well-balanced systems that contain more than eight logical CPU cores. The estimated optimum configurations are shown for both the FPGA and the GPU platforms. Note that the optimum configuration for power-efficiency does not necessary need to be able to obtain exactly the maximum two-fold speedup. The following assumptions have been used for the estimation:

- For each platform, system idle power draw remains identical to the measured platform, as no additional idle power draw is assumed for the additional CPU cores (due to perfect power gating of CPU cores);
- Additional power under load required by the additional CPU cores is scaled linearly based on the difference between software-only idle and load power consumption;
- The accelerators do not require additional power for the scaled system, as their workload stays identical;
- Overall application performance scales linearly in CPU core count.

All platforms show a higher energy efficiency as compared to the base eight logical CPU core system. The faster execution results in a lower overall execution time, in turn requiring less power draw of the base system components. Conversely, a system with fewer cores would show lowered power-efficiency. The FPGA-accelerated platform is able to achieve an up to 2.1x improvement in power-efficiency, as compared to the eight core software-only platform, and is able to map up to 44 kbp/J. The GPU-accelerated platform obtains a 46% power-efficiency improvement on the more well-balanced system.

Note that the above results use a conservative estimate for the power-efficiency of the accelerated platforms. These platforms are the most efficient while fully utilized: only in such a situation no unnecessary idle power loss is accumulated while the accelerator is being idle. However, this is not the case for the system on which the results have been gathered, as in that system, the accelerators were idle about half of the overall execution time. Hence, the measured power consumption includes this idle power loss, which means that the results as presented here are a conservative estimate of the power-efficiency for both platforms. In practice, the power-efficiency of a more well-balanced system should be even higher.

To illustrate the dependency of the power-efficiency on the number of logical CPU cores in use, Figure 5 shows the estimated normalized power-efficiency for a system with varying number of cores, compared to the base eight core software-only platform. Under all circumstances, the FPGA platform is the most power-efficient. Both accelerated systems show peak efficiency for a system with about sixteen cores. Hence, a system with an Intel Core i7-5960X processor would be a good matchup. A system with less cores is hampered by under-utilization of the accelerator, whereas for a system with more cores, the accelerator will not be able to keep up with the CPU cores, which, as a result, will be partially idle. In such a situation, load balancing between host and accelerator, as in [9], could improve system resource utilization, resulting in a more graceful drop-off in performance and power-efficiency.

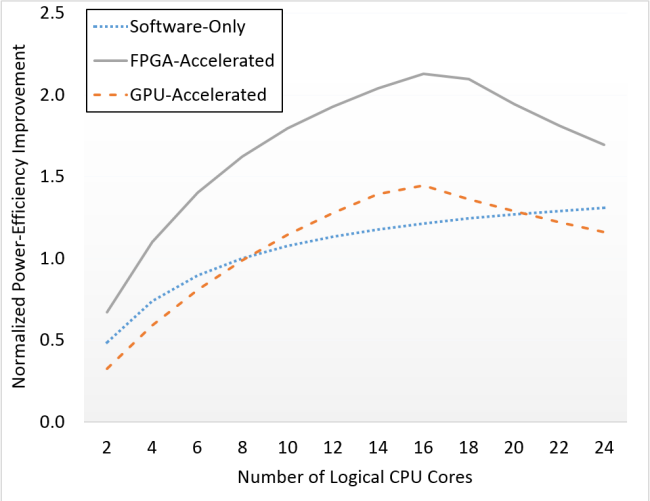

Fig. 5. Estimated power-efficiency for a system with varying CPU core count. The FPGA-accelerated platform is the most power-efficient.

V. Discussion

Accelerate an algorithm such as BWA-MEM, which is characterized by the fact that it contains multiple performance-critical kernels, is always challenging. Hence, as per Amdahl's law, acceleration of a single kernel can only yield limited overall application speedup. In the case of BWA-MEM on the Intel Core i7 platforms, a maximum speedup of 1.7x is implied. Even so, the FPGA and GPU implementations manage to exceed this maximum speedup by at the same time implementing a pipelined program organization, thus achieving a two-fold improvement to execution time. Platforms where the Seed Extension phase consumes a larger part of total application execution time should be able to see an even larger performance increase.

Both the FPGA and GPU-accelerated platforms manage to obtain a two-fold performance improvement. However, it is interesting to compare the nature of both architectures further. The fixed function units on the GPU are able to process large numbers of reads in parallel at a high clock frequency of up to 1.25 GHz. However, a large number of instructions is required to perform a single systolic array cell update, and a large amount of logic is required to provide the massive threading and parallelism. In contrast, the much lower clock frequency and highly customizable nature of the FPGA allows for much more power-efficient processing, resulting in the fact that the FPGA requires less than one fourth of the power consumption of the GPU, while performing the same workload.

The highest power-efficiency of any system including accelerator hardware will only be obtained through a careful balance of system components. Utilizing an accelerator that

is overpowered compared to the rest of the system reduces power-efficiency, whereas a well-balanced system can provide much improved power-efficiency. As shown here, the results for an optimally-balanced system are much better compared to the baseline system. A further improvement to power-efficiency of 29% for the FPGA platform was obtained, and even a 46% improvement to power-efficiency for the GPU platform. This balance of components is especially important when considering cloud-based solutions for data processing, as the available options that can be selected, in particular when accelerators are involved, are restricted to those offered by the cloud provider. The optimal combination may not be always available. It is an ongoing effort to monitor the best available options, making the optimal trade-off between power and power-efficiency.

VI. Conclusions

In this paper, an accelerated implementation of the BWA-MEM algorithm is introduced that targets the Alpha Data add-in card with a single Xilinx Virtex-7 FPGA. This design is able to provide a two-fold improvement to overall application performance by offloading the Seed Extension phase onto the FPGA and through better pipelining of the application. Scalability analysis shows that the current design is able to provide this maximum two-fold gain in performance for a system with up to sixteen logical CPU cores.

To facilitate platform-agnostic comparison of power-efficiency on mapping genomic data sets, the base pairs per Joule measure is proposed as a unit to express platform power-efficiency. The performance and power-efficiency is compared to a GPU-based implementation, which is also able to obtain a two-fold performance improvement. However, the FPGA implementation is much more power-efficient and can map 34,000 base pairs per Joule of energy, an improvement of 60% compared to the software only and GPU-platforms. Design space exploration shows that a more well-balanced platform, designed to fully utilize the accelerator's potential, can provide an up to 2.1x improvement in power-efficiency, providing a mapping power-efficiency of up to 44,000 base pairs per Joule. Due to the highly customizable nature of the FPGA, its power-efficiency is much higher compared to the GPU and software-only platforms. The FPGA consumes less than one fourth of the power the GPU requires when executing the same workload, showing the large benefit of customizable hardware, as compared to more fixed function hardware.

The authors expect to see increased use of FPGA hardware in the bioinformatics context, as the improvement in speed and power-efficiency will help to reduce the bottleneck of an important part of the widely used BWA-MEM/GATK pipeline. More generally, it offers a large promise of power-efficiency gains for the often extremely large computational challenges in this domain.

References

[1] National Human Genome Research Institure, "DNA Sequencing Costs," http://www.genome.gov/sequencingcosts/, accessed: 2015-12-22.

[2] Z. Stephens, S. Lee, F. Faghri, R. Campbell, C. Zhai, M. Efron, R. Iyer, M. Schatz, S. Sinha, and G. Robinson, "Big Data: Astronomical or Genomical?" *PLoS Biology*, vol. 13, no. 7, 2015.

[3] Illumina, "HiSeq X Specification Sheet," http://www.illumina.com/content/dam/illumina-marketing/documents/products/datasheets/datasheet-hiseq-x-ten.pdf, accessed: 2015-07-15.

[4] H. Li, "Aligning Sequence Reads, Clone Sequences and Assembly Contigs with BWA-MEM," *arXiv preprint arXiv:1303.3997*, 2013.

[5] B. Langmead and S. L. Salzberg, "Fast Gapped-Read Alignment with Bowtie 2," *Nature methods*, vol. 9, no. 4, pp. 357–359, 2012.

[6] C.-M. Liu, T. Wong, E. Wu, R. Luo, S.-M. Yiu, Y. Li, B. Wang, C. Yu, X. Chu, K. Zhao, and R. Li, "SOAP3: Ultra-Fast GPU-Based Parallel Alignment Tool for Short Reads," *Bioinformatics*, vol. 28, no. 6, pp. 878–879, 2012.

[7] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, and M. DePristo, "The Genome Analysis Toolkit: a MapReduce Framework for Analyzing Next-Generation DNA Sequencing Data," *Genome research*, vol. 20, no. 9, pp. 1297–1303, 2010.

[8] T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[9] E. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "GPU-Accelerated BWA-MEM Genomic Mapping Algorithm Using Adaptive Load Balancing," in *Architecture of Computing Systems–ARCS*. Springer, 2016, pp. 130–142.

[10] M. J. Jaspers, "Acceleration of Read Alignment with Coherent Attached FPGA Coprocessors," Master's thesis, TU Delft, Delft University of Technology, 2015.

[11] P. Zhang, G. Tan, and G. R. Gao, "Implementation of the Smith-Waterman Algorithm on a Reconfigurable Supercomputing Platform," in *Proceedings of the 1st international workshop on High-performance reconfi- gurable computing technology and applications: held in conjunction with SC07*. ACM, 2007, pp. 39–48.

[12] T. Oliver, B. Schmidt, and D. Maskell, "Hyper Customized Processors for Bio-sequence Database Scanning on FPGAs," in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. ACM, 2005, pp. 229–237.

[13] J. Arram, K. H. Tsoi, W. Luk, and P. Jiang, "Reconfigurable Acceleration of Short Read Mapping," in *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*. IEEE, 2013, pp. 210–217.

[14] Y.-T. Chen, J. Cong, J. Lei, and P. Wei, "A Novel High-Throughput Acceleration Engine for Read Alignment," in *Field-Programmable Custom Computing Machines, 2015. FCCM 2015. 23rd Annual IEEE Symposium on*. IEEE, 2015.

[15] E. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "An FPGA- Based Systolic Array to Accelerate the BWA-MEM Genomic Mapping Algorithm," in *Intl. Conf. on Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2015.

[16] N. Ahmed, V. Sima, E. Houtgast, K. Bertels, and Z. Al-Ars, "Heterogeneous Hardware/Software Acceleration of the BWA-MEM DNA Alignment Algorithm," in *Proc. of the IEEE/ACM Intl. Conf. on Computer-Aided Design*, ser. ICCAD, 2015.

[17] E. Houtgast, V. Sima, K. Bertels, and Z. Al-Ars, "An Efficient GPU-Accelerated Implementation of Genomic Short Read Mapping with BWA-MEM," in *Proc. International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies*, Hong Kong, China, July 2016.

[18] Alpha Data, "Alpha Data ADM-PCIE-7V3 Product Information," http://www.alpha-data.com/dcp/products.php?product=adm-pcie-7v3, accessed: 2015-12-14.

[19] The OpenEnergyMonitor Project, "OpenEnergyMonitor Website," http://openenergymonitor.org/, accessed: 2015-12-14.

[20] G. Highnam, J. J. Wang, D. Kusler, J. Zook, V. Vijayan, N. Leibovich, and D. Mittelman, "An Analytical Framework for Optimizing Variant Discovery from Personal Genomes," *Nature comm.*, vol. 6, 2015.