International Conference on Computational Science, ICCS 2010

Using the reconfigurable massively parallel architecture COPACOBANA 5000 for applications in bioinformatics

Lars Wienbrandt*,1, Stefan Baumgartb, Jost Bisselb, Carol May Yen Yeo, Manfred Schimmlera

aDepartment of Computer Science, Christian-Albrechts-University of Kiel, Hermann-Rodewald-Str. 3, 24118 Kiel, Germany
bSciEngines GmbH, Fraunhoferstr. 13, 24118 Kiel, Germany

Abstract

Currently several computational problems require high processing power to handle huge amounts of data, although underlying core algorithms appear to be rather simple. Especially in the area of bioinformatics, algorithms implemented in PCs do not utilize all hardware functionalities provided by standard CPUs. As the demand for efficient utilization and speed up increases, this leads to a boost in the trend of implementing dedicated hardware. Hardware implementations can be done very fast and are cost effective on reconfigurable devices such as FPGAs. With 128 low-cost FPGAs residing on the COPACOBANA 5000 and in combination with a high-throughput systolic bus system, this machine therefore provides a dynamic solution for massively parallel computations with reconfigurable capabilities.

This paper describes the advantages of this architecture based on the implementation of efficient solutions designed for two well-known algorithmic problems in bioinformatics: Smith-Waterman Alignment and DNA Motif Finding.

Keywords:
Parallel Processing, FPGA, Reconfigurable Hardware, Energy Efficiency, High Throughput, DNA Motif Search, Smith-Waterman Alignment

1. Introduction

Algorithms in bioinformatics most of the time appear to be rather simple in their structure, such as the famous Smith-Waterman alignment [1] which forms the basis for heuristic algorithms. An example of such heuristic algorithms is BLAST [2], whereby the matrix entries of integer values are calculated without much complexity. Other algorithms, e.g. in terms of DNA motif finding, such as Gibb’s Sampling [3], PROJECTION [4] or the widely accepted MEME [5], use probability values, which are often represented as floating points. These values are not as easy to handle as integer values, but in general, all mentioned algorithms acting on pure DNA data have another advantage: Since DNA strings only consist of characters of the alphabet \( \mathcal{L} = \{ A, C, G, T \} \), the input data can be encoded in portions of two bits.

*Corresponding author

Email addresses: lw1@informatik.uni-kiel.de (Lars Wienbrandt), stefanbaumgart@sciengines.com (Stefan Baumgart), jostbissel@sciengines.com (Jost Bissel), cmy@informatik.uni-kiel.de (Carol May Yen Yeo), masch@informatik.uni-kiel.de (Manfred Schimmler)

1877-0509 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.procs.2010.04.114
In general, most of such algorithms are highly computationally intensive by nature because they possess at least quadratic complexity while handling huge amounts of data from several mega- up to giga- or even terabytes. Since Moore’s Law cannot cope with increasing requirements of data handling in the area of bioinformatics, the only solution is to harness massive parallelization. The easiest way of realizing a massively parallel application is the implementation on a PC cluster. However, this solution has definite disadvantages. Firstly, hardware resources of the processing units (such as CPUs) cannot be fully utilized due to fixed instruction sets. Even successful accelerating implementations of the previously mentioned MEME-algorithm using e.g. graphics hardware (GPUs) [6] have this disadvantage. Secondly, the costs to acquire a large number of PCs can be immense. Moreover, additional costs for room to allocate the cluster, connections, maintenance and possible installation of a temperature control unit have to be evaluated. Even energy costs have to be considered nowadays.

With the emergence of the new architecture COPACOBANA 5000, these problems were reduced [7]. This paper aims to describe the capabilities of this machine that can cater to the needs of parallel applications. The latter, which can be described as a hardware problem, e.g. with the help of a hardware description language like VHDL, are generally suitable to implement on COPACOBANA 5000. This is because hardware implementations are more flexible as compared to CPUs as they possess fixed instruction sets. In other words, the hardware is designed to specifically cater for the algorithm to fully utilize the existing resources. COPACOBANA 5000 provides 128 reconfigurable Xilinx Spartan-3 5000 FPGAs, connected in a high throughput systolic bus system. Each FPGA is supplemented with additional 32MB of SDRAM and the entire machine is controlled by an internal PC with a quad-core CPU, 4GB DDR-II RAM and up to 8TB of hard disk space. This provides ample resources to handle most problems in bioinformatics. Due to the ability of reconfiguration, the design process is made easier.

In the following, section 2 describes the structure of the COPACOBANA 5000. Section 3 illustrates examples of the implementation of bioinformatics applications on this architecture, whereby a solution for the Smith-Waterman algorithm and the initial implemented BMA algorithm [8, 9], specifically developed for a hardware implementation, adapted to this architecture is described.

2. COPACOBANA 5000

2.1. History of COPACOBANA

Initially introduced in 2006, the Cost Optimized PARallel COdeBreaker and ANALyzer, COPACOBANA, has evolved to a family of parallel FPGA-based high performance computers. It was originally developed for applications in cryptanalysis. One very successful example is the breaking of DES ciphers in less than one week [10], but also A5/1 [11] and other cryptographic techniques such as AES or RSA were attacked [12]. The first COPACOBANA machine contains 120 Xilinx Spartan-3 1000 FPGAs, which are connected in a single-master-multiple-slave bus system. This system is designed for moderate data throughput since cryptanalytical applications generally do not require high performance bus systems. The adaption of applications in the area of bioinformatics proved to be promising in the beginning – simple algorithms, such as BMA for DNA motif finding, were easily implementable in hardware. The speedup gained was magnificent. However, after further analyzation of the problem, it turns out that the total runtime is still not very applicable. Since most algorithms in bioinformatics have a high dependency on huge amounts of genome data, the bus speed of the original COPACOBANA is no longer sufficient. Additionally, the need for memory increases, e.g. in the area of DNA fragment assembly raw data in amounts of several hundred gigabytes. Therefore, a
redesign of the original COPACOBANA was required in order to cater for broader applications besides bioinformatics and cryptanalysis. The redesign, introduced as COPACOBANA 5000 [7], is commercially available as RIVYERA (ReconfIgure Versatilely Your Efficient Raw Architecture, fig. 1).

2.2. COPACOBANA 5000 Architecture

The COPACOBANA 5000 architecture consists of a backplane with 18 slots, FPGA cards, enclosure with power supply and an internal PC. Every slot is identical and a typical setup is equipped with 16 FPGA cards. The FPGA machine can be connected via multiple full bandwidth PCI-Express controllers to an integrated off-the-shelf standard PC. Using a standard PC main board, various features can be easily attached to the FPGA machine, such as ethernet ports, firewire or fibre channel. Each of the FPGA cards carries eight high performance FPGAs interconnected in a one dimensional array, also known as a systolic chain. Additional components are provided to support the mentioned functional units such as a 1.5kW main power supply unit, three high-performance fans and a 19-inch rack of three height units (3HE) for the housing. Figure 2 shows the general machine architecture.

2.2.1. General Improvements

COPACOBANA 5000 increases the amount of provided logical elements for the user by a factor of 4.5 times as compared to the original COPACOBANA 1000. Xilinx Spartan-3 5000 FPGAs offer more block RAM as well as more logical resources. The FPGA interconnection and the design of the boards have been generalized. Figure 3 shows the basic architecture of a single FPGA of the system. A mixture of FPGA cards with different vendors is possible. This allows to combine the benefits of certain FPGAs equipped with DSPs with those providing primarily logical resources. The setup and sequence of FPGA cards can be selected by the user to allow the best match with the application needs.

2.2.2. Memory

In order to avoid bottlenecks during data access, each FPGA offers directly attached 32MB of DRAM. A set of memory controllers with different size and features is available through the API. An additional memory expansion slot allows access up to 32GB of FLASH memory on a single SDHC memory card. For future expansion, the interface provided is compatible to the upcoming SDXC standard. A small I/O Expansion port enables additional data transfer providing bi-directional independent differential lanes.

Figure 2: COPACOBANA 5000 machine schema
2.2.3. The Standard PC

The internal standard PC allows storage extensive algorithms to rely on up to 8TB of SATA connected hard disk drives. The storage can be accessed with controller cards interfacing the PC via PCI-Express. Nevertheless, the integrated PC can be utilized to provide processing power to parts of the algorithm which are not likely to be speeded up using FPGAs. Furthermore, it allows users to incorporate the benefits of FPGAs with the opportunities of GPU based processing by combining FPGA processing power with GPUs and CPUs. Using PCI-Express as standard interface Direct Memory Access (DMA) allows data flow between GPU and FPGAs without additional coordination of the CPU and vice versa.

2.2.4. The Systolic Bus System

COPACOBANA 5000 offers a high-performance bus system. The interconnection between the individual FPGAs and between the FPGA-cards is organized as a one dimensional array or systolic chain. The general idea of a systolic chain is to provide fast point-to-point connections between every two neighbours. The speed of a bus system depends on several directly dependent factors which have to be balanced carefully in a cost optimized system e.g. the length and number of wires and the achievable clock speed and data throughput. On the one hand, a systolic like architecture typically results in shorter wires achieving higher frequencies and therefore higher data throughput. On the other hand, one of the typical problems is the latency of large chains and the usability. These problems have been eliminated by using a bus architecture and routing scheme implemented in an API which has already been introduced in an earlier paper [7] and are omitted here. The point-to-point interconnections consist of eight pairs of wires in each direction. Each pair is driven by low voltage differential signalling (LVDS) with a speed of 250MHz, thus achieving a data-rate of 2Gbit/s.

3. COPACOBANA 5000 in Bioinformatics

3.1. Smith-Waterman Alignment

Dealing with genomic sequences is one of the key topics in the area of bioinformatics. Due to the vast amount of raw sequence data faced by scientists, it is indispensable for them to have computer-based assistance in selecting the most interesting sequence candidates.

The alignment of nucleotide sequences deals with the problem of finding the best fitting alignment of two nucleotide sequences against each other. Algorithms used to handle this problem may be classified as either heuristic or non-heuristic. The heuristic alignment algorithms such as BLAST [2] have become common tools to search for alignments since they are much faster than non-heuristic types. Although they produce a large amount of false results and may not succeed in finding all the correct solutions, they outperform non-heuristic algorithms by far, in terms of computing time and have therefore gained broad acceptance within the group of molecular biologists.
In recent years FPGA systems are used to accelerate all kinds of alignment algorithms and now it is able to cater for non-heuristic types, such as Smith-Waterman [1]. In this section, it is demonstrated how COPACOBANA 5000 can be used for this purpose.

3.1.1. Smith-Waterman Algorithm

The Smith-Waterman [1] algorithm is capable to find the best alignment of a sequence to another in a non-heuristic manner. For convenience, the sequence that is searched for is called *query sequence* while the one that is searched in is called *database sequence*. In order to find the best of every possible alignment, a score is generated. These scores are calculated by the simple scoring function:

$$H(i, j) = \max \begin{cases} 
0 & \text{H(i-1, j) + GapPenalty} \\
H(i, j-1) + \text{GapPenalty} & \text{H(i-1, j-1) + Match/Mismatch}
\end{cases}$$

$H$ is a matrix and the values for *GapPenalty*, *Match* and *Mismatch* can be user defined. $H$ is the so-called *scoring matrix*. In a software system, the algorithm evaluates the entire scoring matrix and outputs the maximum of all $H(i, j)$. This mechanism is memory intensive since memory consumption raises quadratically – the matrix contains $n \cdot m$ cells while $n$ and $m$ are the lengths of the query and database sequence respectively. Additionally, standard processors encounter the problem that the four DNA bases $A, C, G$ and $T$ can be encoded using only two bits, which is very inefficient for 32- or even 64-bit architectures.

3.1.2. Implementation

For handling the memory limitations mentioned above, it is essential to know that in a biologist’s workflow, it is very likely to align thousands of query sequences at a time based on the examination of the maximal scores of the alignments. These scores are analyzed and the actual alignment of a very small selection of query sequences with the highest scores may easily be computed again on a standard PC. Hence, it is not important to store the whole matrix, but only the maximum and the values needed for computation. The output is simply the maximum cell value.

In terms of parallelization, the algorithm can easily be handled. Considering the scoring matrix, the top-down axis can be viewed as the processing elements, the left-right axis as time intervals. Thus, every processing element (PE) calculates one line, i.e it is responsible for exactly one character of the query sequence. Therefore, referencing the value of the left neighbour is simply accessing its own value of the previous clock cycle. In this way, referencing the value from the top or top-left neighbour means referencing the value of the preceding PE one or two clock cycles before respectively. Additionally, the maximum value that has occurred so far is updated. With this parallelization scheme, the algorithm processes the matrix diagonally with the database sequence streaming through the chain of processing elements. Figure 4 shows two consecutive steps evaluating a scoring matrix.

3.1.3. Performance Results

Common FPGA implementations of the Smith-Waterman algorithm only use a single or a few FPGAs [13, 14]. Hence, they are either limited in the length of the query sequence or needed to reinitialize their chain of processing elements. Implementing the Smith-Waterman algorithm on COPACOBANA 5000 offers new ways for scaling opportunities. Now, there are 128 instead of one FPGA available. Additionally, in contrast to the restricted inter-process communication ability of the original COPACOBANA, all FPGAs can communicate in a systolic manner. Thus, it is
possible to simply align 128 small query sequences at the same time, or to align a query sequence that is 128 times the former possible size before COPACOBANA 5000 was available. It is even able to mix different query lengths while processing them altogether in parallel.

Table 1 demonstrates the performance by the alignment of 3,685 20-mers against the human genome, comparing COPACOBANA 5000 to a standard PC (AMD Opteron at 2.2 GHz) and a Cray XD1 using one FPGA [14].

3.2. DNA Motif Finding

DNA sequence motifs are often related to transcription factor binding sites (TFBS) which are needed by the biological protein synthesis machinery to start the transcription of genes. Genes can be reduced to a sequence of base characters from the alphabet \( L = \{A,C,G,T\} \), hidden randomly in the entire sequence of the DNA strand. Proteins binding on sites several bases before the genes have influence on their transcription. They can either emphasize or suppress a transcription, and hence they are referred to as transcription factors.

TFBS are sites of short lengths whereby their sequences are unknown and unpredictable in advance. Since many genes are transcribed in similar conditions, the only assumption made on such motifs is, that there are several similar instances distributed over the entire genome sequence. Thus, the motif finding problem can be defined as searching for sequence patterns of short lengths occurring in several instances which appear similar, but not necessarily equal.

3.2.1. BMA Algorithm

The BMA algorithm [8] was originally developed to find motif instances of fixed short length (e.g. 12) while motif instances are described as motif kernels. The design of BMA was specifically focussed on a hardware implementation. Hence, these motif kernels are internally represented as Boolean matrices and its main operation is pattern matching. BMA sets stricter restrictions to its motif instances as compared to other algorithms. Motif instances have either preserved positions, i.e. exactly one character is allowed here, or semi-preserved positions, i.e. two different characters are allowed here. Figure 5 shows an example of a Boolean matrix with only one semi-preserved position – which implies a motif kernel size of \( k = 2 \). The intention of BMA is to find the potential motif instances by increasing the motif kernel size \( k \), i.e. creating semi-preserved from initially preserved positions, by analyzing the structure of the input sequence.

To give an overview a short summary of the algorithm structure is stated as follows:

1. Let \( l \) be the desired motif size, \( k \) the desired motif kernel size (number of semi-preserved positions in motif instances).
2. For every possible sequence of length \( l \) initialize a Boolean matrix.
3. Match every subsequences of length \( l \) of the input sequences against every Boolean matrix.
4. Count the exact matches which will be the score of the matrix. Also for every entry in the matrix, count those subsequences which have exactly one mismatch in this particular position.
5. After analysing the whole input sequence, switch the matrix entry with the highest count from 0 to 1 (weaken the matrix – let a preserved position become semi-preserved).
6. Repeat steps 2 to 5 for \( k \) times.

This algorithm has already been implemented for the COPACOBANA 1000 [9], hence the next sections give an overview of the implementation and describe the steps which were needed for the adaption to the new architecture COPACOBANA 5000.
A 011000
C 010010
G 100100
T 000001
⇒ Motif kernel size \( k = 2 \), motif size \( l = 6 \):
Strings GAAGCT and GCAGCT are derived.

Figure 5: A Boolean matrix with one semi-preserved position.

3.2.2. Implementation and Optimization

The original solution for solving this problem on the original COPACOBANA is apparent. There are hardware entities called search entities, designed for storing a Boolean matrix of fixed size and the capability to perform pattern matching of a two-bit encoded genetic string of the same length against this matrix. A perfect match triggers a counter for the matrix score while the counters for the mismatching positions – the scoring matrix – are stored in the block RAM of the FPGA. An additional control unit implements the interface to the COPACOBANA 1000 bus system. Its tasks are to process commands for initialization, matrix update or result retrieval. Basically, this control unit receives sections of the input sequence and routes them in substrings of the correct length to the search entities. The results are pre-compared between the entities such that upon result retrieval, the best results are fetched first.

Based on the current algorithm implemented on COPACOBANA 1000, there exists the capacity available for further optimization of the original implementation with the incorporation of COPACOBANA 5000 hardware. Firstly, the number of the search entities that can be accommodated in each FPGA is increased to a factor of four times more. This is obvious due to the migration of the Spartan-3 5000 FPGA, which provides about four times more system logic as compared to the previous FPGA. Secondly, the major factor that contributes to a huge reduction in computation time lies in the revamp of the initialization procedure. In the previous design, each search entity is initialized to search for one of the combination of all possible motif instances (i.e. \( 4^l \)). However, according to the original description of the algorithm in [8], it is sufficient to take only the substrings of the input sequence for initialization. Given a small genome file (e.g. 230,000bp for Cowpox virus), this procedure saves over 98% of runtime on the original architecture (ignoring the additional time needed for the new initialization routine).

With this optimized modification and the incorporation of the faster data rate of the COPACOBANA 5000, huge speedup factors are attainable. The following section portrays the performance results achievable in terms of price, speedup and energy consumption in more details.

3.2.3. Performance Results

After the extrapolation of the results obtained in reference to [9] and recalculation of the optimized version of the algorithm, Table 2 tabulates the computation times needed for two different genome files tested on various hardware platforms. The software implementation has been written in C++ and optimized for PC usage. As seen from the table, for a small file, such as the Cowpox virus, the computation time is expedited by more than a factor of 10,000 fold, compared to a PC. For larger files, such as Bacillus subtilis, the speedup is even greater than 34,000.

<table>
<thead>
<tr>
<th>Target Architecture</th>
<th>Cowpox virus (230kbp)</th>
<th>Bacillus subtilis (5, 9Mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Runtime (vs. PC)</td>
<td>Speedup</td>
</tr>
<tr>
<td>COPACOBANA 5000</td>
<td>1.66s</td>
<td>10,554</td>
</tr>
<tr>
<td>COPACOBANA 1000</td>
<td>3m</td>
<td>97</td>
</tr>
<tr>
<td>Xeon 5150 2.6GHz dual core</td>
<td>2h</td>
<td>2.43</td>
</tr>
<tr>
<td>Pentium IV 2.8GHz</td>
<td>4h52m</td>
<td>1</td>
</tr>
</tbody>
</table>
Since the cost of energy has continually been on the rise, one important factor to consider is energy consumption. Table 2 also illustrates the energy consumption using different hardware platforms, calculated by measuring the power consumption while running the algorithm, multiplied with the runtime. In other words, based on the table projection, the COPACOBANA 5000 is able to search for more motifs present in a large genome file within a significantly reduced time and energy consumption.

4. Summary and Outlook

This paper has demonstrated that the concept of COPACOBANA 5000 has been successfully realized. Applications in the main topics of bioinformatics were implemented, each outperforming PC clusters by order of magnitudes – in terms of runtime, and in terms of energy reduction. Moreover, these are just minute examples to indicate the potential of the machine. Plenty of algorithms, including other areas besides bioinformatics, are capable to be implemented on COPACOBANA 5000, with each one obtaining a promising high speedup and reduced energy costs. Further research will be done in optimizing the capabilities of the machine, e.g. fine-tuning the protocol for the bus system to gain even higher data throughput. In terms of applications for COPACOBANA 5000 the subsequent step is to investigate another main problem in bioinformatics – DNA assembly. Assembly algorithms generally require large processing power and the capability to handle huge amounts of raw data. Hence, COPACOBANA 5000 seems to be adequate to face this task.

At the same time, the DNA motif search application is being optimized by developing a new algorithm that improves runtime and the quality of results. In cryptanalysis, due to four times more logical gates available in the Spartan-3 5000 FPGAs, the expected runtime for DES breaking can be reduced to about two days while breaking A5/1 in realtime poses no further problems (referring to the stated runtimes in [10] and [11]).

Altogether, the reconfigurable COPACOBANA 5000 (s. fig. 1) is capable to speedup processes in many areas, such as up to 750-fold in terms of Smith-Waterman alignment or 34,000-fold in terms of BMA motif search, while at the same time reducing energy consumption. Hence, COPACOBANA 5000 has demonstrated its ability to cater to the needs of massively parallelized algorithms in bioinformatics.

References