Multi-Core SIMD ASIP for DNA Sequence Alignment

Nuno Filipe Simões Santos Moraes Neves

IST-UL / INESC-ID
Rua Alves Redol, 9, 1000-029 Lisboa - Portugal
Email: nuno.neves@inesc-id.pt

Abstract—A novel Application-Specific Instruction-set Processor (ASIP) architecture for biological sequences alignment is proposed in this manuscript. The presented processor achieves high processing throughputs by exploiting both fine and coarse-grained parallelism. The former is achieved by extending the Instruction Set Architecture (ISA) of a synthesizable processor to include multiple specialized SIMD instructions that implement vector-vector and vector-scalar arithmetic, logic, load/store and control operations. Coarse-grained parallelism is achieved by using multiple cores to cooperatively align multiple sequences in a shared memory plain, comprising proper hardware-specific synchronization mechanisms. To ease the programming of the sequence alignment algorithms, a compilation framework based on a suitable adaptation of the GCC back-end was also implemented. The proposed system was prototyped and evaluated on a Xilinx Virtex-7 FPGA VC707 Kit, achieving a 190MHz working frequency. A vanilla and a state-of-the-art SIMD implementations of the Smith-Waterman algorithm were programmed in both the proposed ASIP and in an Intel Core i7 processor. When comparing the achieved speedups, it was observed that the proposed ISA allows to achieve a 33x speedup, which contrasts with the 11x speedup provided by SSE2 in the Intel Core i7 processor. The scalability of the multi-core system was also evaluated and proved to scale almost linearly with the number of cores. A 800-fold speedup was achieved with a 32-core processing framework.

Index Terms—Application Specific Instruction-Set Architecture, Single-Instruction Multiple-Data, Multi-Core Architecture, Biological Sequences Alignment, Biochip Platforms

I. INTRODUCTION

The increasing demand for computational resources in some specific and prominent application domains, coupled with severe energy and power constraints, has driven the development of highly efficient processing systems, where the processor architecture plays a major role, both on the overall performance and on the resulting energetic efficiency of the whole processing system. In particular, the adopted Instruction Set Architecture (ISA) must be carefully tuned for the specific requirements of the considered set of algorithms. By following such approach, it is possible to achieve a low-power efficient solution that is, at the same time, flexible enough to support algorithmic improvements.

Among the several design options, the development of an efficient multi-core Application Specific Instruction-set Processor (ASIP), specifically adapted for biological sequence alignment algorithms, is regarded as a particularly suited approach to comply with all the performance and design requisites enumerated above. The attained processing throughput is achieved as a result of a two-fold contribution: i) inclusion of multiple specialized Single-Instruction Multiple-Data (SIMD) vector instructions in the processor’s ISA, to extensively exploit fine-grained parallelism; and ii) implementation of an extensive multi-core computational structure, composed by multiple instantiations of the designed ASIP, in order to efficiently exploit coarse-grained parallelism. The cumulative result of these two important contributions was demonstrated with different Field-Programmable Gate Array (FPGA) prototypes of the proposed multi-core SIMD ASIP, which proved capable of offering speedup values as high as 720x.

To further demonstrate that the proposed multi-core ASIP complies with the performance and efficiency requirements of the application domain, it was implemented in a 90 nm CMOS process technology. Experimental results indicate that the proposed solution achieves a performance similar to that of high-performance computing processors (e.g., an Intel Core i7) while using 20x less energy.

The manuscript is organized as follows: after the introductory motivation presented above, Section II presents a brief overview on biological sequences alignment algorithms, as well as on their current state-of-the-art SIMD implementations. In Section III, the new and adapted ISA for this specific application domain is presented, while its architecture implementation is presented in Section IV. In Section V, the developed multi-core processing structure, composed of multiple instantiations of the designed ASIP is presented. Section VI presents the obtained experimental results and Section VII concludes the presentation, with the enumeration of the most important contributions.

II. BIOLOGICAL SEQUENCES ALIGNMENT

The Smith-Waterman (SW) algorithm [1], characterized by an $O(mn)$ time complexity, is a widely established Dynamic Programming (DP) algorithm to obtain the local alignment between a query sequence ($q$) and a database sequence ($d$), of sizes $m$ and $n$ respectively. It operates in two distinct phases: it starts by filling a score matrix $H$, followed by a traceback phase over this matrix. The matrix is filled by using an affine gap penalty model [2] (see eq. 1), where $\alpha$ and $\beta$ represent the...
cost of gap opening and extension, and \( Sbc(q[i], d[j]) \) denotes the substitution score value obtained by aligning character \( q[i] \) against character \( d[j] \). The initial conditions are given by \( H(i, 0) = H(0, j) = E(i, 0) = F(0, j) = 0 \).

\[
H(i, j) = \max \begin{cases} 
H(i - 1, j - 1) + Sbc(q[i], d[j]), \\
E(i, j), \\
F(i, j)
\end{cases}
\]

\( F(i, j) = \max \begin{cases} 
H(i - 1, j) - \alpha, \\
F(i - 1, j) - \beta
\end{cases} \quad (1) 
\]

\( E(i, j) = \max \begin{cases} 
H(i, j - 1) - \alpha, \\
E(i, j - 1) - \beta
\end{cases} \)

To speedup the alignment, several hardware accelerators were developed, like those proposed in [3], [4], [5] and [6], but such solutions lack the adaptability and flexibility provided by General Purpose Processors (GPPs) and by ASIPs. On the other hand, various SIMD parallelization approaches on GPPs have also been presented [7]. One of the fastest was proposed by M. Farrar [7], who adopted a pre-computed query profile for the entire database sequence, and optimized the processing scheme by using a striped access pattern, where the computations are carried out in several separate stripes that cover different parts of the query sequence (see Fig. 1a).

The query is divided into \( p \) equal length segments of size \( t = \lfloor (m + p - 1)/p \rfloor \), where \( p \) denotes the number of vector elements that can be accommodated in a SIMD register (each SIMD vector element is assigned to one distinct segment). The length of each segment is given by \( t = \lfloor (m + p - 1)/p \rfloor \), where padding zeros are inserted whenever the query size \( m \) is not long enough to completely fill all the segments. Each matrix column, corresponding to a database symbol \( d[j] \), is processed in \( t \) iterations, where each iteration simultaneously processes \( p \) query symbols, separated by \( t - 1 \) lines in the score matrix. Fig. 1b illustrates the data dependencies between the last segment and the first segment of the next column of the score matrix \( H \). With this processing pattern, it is possible to move the conditional statements related to the commitment of the vertical dependencies to an independent lazy loop, executed outside the inner loop, where they have to be considered only once, before starting the processing of the next database symbol, thus reducing the impact of the vertical dependencies.

The above cumulative set of contributions and improvements led to significant speedup values of the alignment, which makes Farrar’s technique [7] one of the fastest SIMD implementations of the SW algorithm. For this reason this algorithm is integrated in many current high performance alignment frameworks, such as the latest versions of SSEARCH [8].

To further accelerate this class of algorithms, a new ASIP, specifically adapted for biological sequence alignment algorithms, is proposed in this manuscript. The attained processing throughput is achieved as a result of a two-fold improvement in the original architecture: i) extension of the processor ISA to support multiple specialized SIMD vector instructions, to extensively exploit fine-grained parallelism; and ii) implementation of an extensive multi-core computational structure, composed by multiple instantiations of the designed ASIP, in order to efficiently exploit coarse-grained parallelism.

III. DEDICATED SIMD INSTRUCTION SET FOR BIOLOGICAL SEQUENCES ALIGNMENT

The proposed ISA was defined targeting the acceleration of the classic local and global sequence alignment procedures (such as the several SIMD implementations of the Needleman-Wunsch and Smith-Waterman algorithms [9], [10], [7]). Due to its higher performance and prevalence in most widely established bioinformatics applications, Farrar’s SIMD implementation [7] will be herein adopted as the elected case-study.

By analyzing the algorithm’s pseudo-code (see Fig.2(a)), it is clear that the adoption of vector arithmetic instructions will potentially accelerate this algorithm. These instructions should not only speedup the operations between vectors, but they might also facilitate the several operations between vectors and scalars, which are particularly useful when subtracting the gap penalties. The shifting of the \( F \) and \( H \) vectors can also be efficiently implemented with a vector element shift instruction. Since all these new instructions will be dealing with SIMD vectors, it is also advantageous to include new memory access instructions, to handle vector-sized variables.

A special attention should be also devoted to the definition of optimized control instructions. This effort is justified by the significant predominance of loop procedures in these DP algorithms (generally implemented with conditional branch instructions), as well as the severe penalties that these control instructions generally impose on deep pipeline architectures. In particular, a new specialized branch instruction to simultaneously assert a branch condition in all vector elements, without any additional processing, will significantly increase the achieved performance (e.g.: execution of the lazy loop).

Although not limited at this respect, the proposed instruction set and the corresponding data-path (see section IV) provides support for the same register and vector-element sizes as Intel SSE2 (used by Farrar [7]), i.e. 128-bit registers, with 8 or 16 elements. Furthermore, the vector elements of each register...
Fig. 2. Farrar’s SIMD implementation of the SW algorithm [7]: (a) Pseudo-code definition; (b) Intel SSE2 assembly; (c) Proposed ISA assembly code. Instructions outlined in bold face belong to the specialized ISA. Instructions marked with shading are those that were left out, including the multiplication and division operations. By comparing Farrar’s [7] algorithm implementation based on Intel SSE2 ISA (see Fig. 2(b)) with an implementation based on the proposed ISA (see Fig. 2(c)), it can be observed that an immediate gain, regarding the number of instructions, is promptly achieved with the proposed ISA, with more visible advantages in the lazy-loop. The major contributor to this reduction is the new set of vectorized control instructions, that significantly reduce the control overhead.

It is also important to note that another significant advantage of the proposed ASIP arises from the fact that it adopts a strict Reduced Instruction Set Computer (RISC) paradigm based on a shallow pipeline structure, contrasting with Intel’s Complex Instruction Set Computer (CISC) model. As a consequence, the observed difference in the number of instantiated instructions, together with the RISC single-cycle per instruction ratio (instead of CISC multiple-cycle per instruction), will significantly augment the processing gain, as it will be demonstrated in section VI.

IV. SIMD PROCESSOR ARCHITECTURE

The MB-LITE [11] soft-core was used as the base architecture for the implementation of the proposed ISA, not only due to its simple and portable processing structure, but also because it is a compliant implementation of the well known MicroBlaze ISA. Furthermore, since the GNU Compiler Collection (GCC) already supports the MicroBlaze processor, adding the new instructions’ mnemonics and op-codes was easily accomplished by conveniently adapting the corresponding back-end.

The MB-LITE design is highly configurable and is relatively easy to adapt to the proposed ISA. Accordingly, some groups of instructions were left out, including the multiplication and barrel shifter operations, as well as all floating point and special register operations. In fact, the reduced hardware resources that are required by this core were also taken into account, prospecting the bases for a scalable multi-core processing platform to exploit coarse-grained parallelism.

Despite being fully parameterizable, the configuration of the designed SIMD module that was adopted for this specific case-study uses 128-bit registers with 16 8-bit SIMD elements. To support the proposed extension of the ISA, the execution unit had to be modified, by extending its original ALU to include a new SIMD module. As an example, the addition and subtraction operations require one adder per SIMD vector element, together with some extra multiplexing logic. Since different types of SIMD operations are supported (vector-vector, vector-scalar and inner-vector), the required vector elements have to be selected from the corresponding registers and only then does the execution unit perform all the parallel arithmetic operations. The results are then chosen based on appropriate control signals. Also, and according to Farrar’s [7] algorithm implementation properties, a result saturation option is provided for the addition and subtraction operations.

The new maximum instruction, particularly useful for this class of algorithms, deserved a special attention. It was based on the already existing compare instruction, comprising a subtraction followed by a signal evaluation. Therefore, the
same logic can be used to implement these two instructions, requiring only a multiplexer to choose the maximum between the two operands. To avoid increasing the critical-path, the decision logic was moved to the next pipeline stage and to the pipeline forwarding lines. This new maximum instruction not only substitutes one compare and one branch instruction, but it also prevents the pipeline flush (gaining 3 or 4 clock cycles), depending on whether the branch has delay slots or not.

Whenever possible, the same opcode was assigned to the new SIMD instructions as their non-SIMD counterparts, by using unused fields to distinguish them in the processor control unit. With this option, it was possible to re-use most of the original decoding structures, except for a few control signals that had to be generated from such bit-fields.

The final ASIP pipelined architecture, with the above described modifications, is presented in Fig. 4.

A. Compiler Implementation

The conducted implementation of the compiler that was specifically developed for the proposed ASIP is based on the well-known GCC family [12]. The base GCC structure was extended to support the proposed architecture, in order to immediately allow programs to be written in the target processor’s Assembly Language (ASM). Since GCC already supports the MicroBlaze processor, adding the new mnemonics and opcodes was straightforward. Currently, support is provided only at the back-end level. In the future, a particular attention will also be given to the middle-end, since it allows other optimizations using the Abstract Syntax Tree (AST). This will allow for further leverage of the specific aspects of the underlying ISA, including automatic vectorization. Nonetheless, support for the C/C++ programming languages can be provided by relying on intrinsic functions that directly use the SIMD instructions.

V. MULTI-CORE PROCESSING PLATFORM

Many High Throughput Short Read (HTSR) sequencing applications require the alignment of multiple query sequences to one or more database sequences. This requirement adds a thread-level parallelism to the computation, where multiple cores concurrently align multiple query sequences with one or more database sequences. To allow this parallel computation, a shared memory is used to store the database and query sequences [13] (see Fig. 5). The computation is controlled by a master core, which manages the sequence alignment queue and the multiple processing elements. To initiate the sequence alignment, the master core needs to communicate a minimal set of data to the target processing core, which consists of the address (in main memory) and the length of the query and database sequences.

To compute the alignment score for multiple query sequences, the architecture illustrated in Fig. 6 is now proposed. It is composed of: a memory element, to store both the biological sequences and the alignment scores; a master core, which is responsible for managing the sequence alignment queue; multiple processing cores based on the specialized SIMD ASIP; and a mutex circuit, to handle core synchronizations. All elements are interconnected by an AMBA 3 AHB-Lite compatible shared bus. To reduce the amount of data that is transferred between the master and the processing cores, it was considered the shared memory model studied in [14], specifically developed for this application domain.

Each of the processing cores is composed of the specialized SIMD ASIP presented in section IV, an instruction memory, a local (scratchpad) memory, a Direct Memory Access (DMA) controller and an on-chip network interface (see Fig. 6b).
B. Synchronization Mechanism

can be performed by using the usual load/store instructions. Therefore, access to the DMA registers (to configure the DMA
creation of an interrupt routine that handles all of the data
The DMA controller can also flag an interruption to the ASIP
This approach reduces the number of bus requests and allows
the query or reference sequences to the scratchpad memory).

A. Data Communication

The proposed architecture supports various types of inter-
connections (e.g., shared bus, ring/mesh network-on-chip, etc.)
with minimum changes to the base structure. The consid-
ered prototyping implementation, which is described in this
manuscript, uses a shared bus coupled with an arbiter to
manage the access to the bus. The adopted bus protocol is
AMBA 3 AHB-Lite compatible (with multi-layer support),
requiring a minimum of two clock cycles to transmit the data:
the first to request access to the bus and the second to transmit
the data. Naturally, whenever the bus is unavailable (busy),
additional clock cycles are required. To minimize the data
transfer time, the bus arbiter also supports a burst mode, where
a single bus request is used to transmit multiple data packets.
In this case, a minimum of \( n + 1 \) clock cycles are required for
transmitting \( n \) data packets.

To further reduce the contention in the system bus, each
processing element makes use of its local scratchpad memory,
to store temporary data, and of its internal DMA controller,
to handle most accesses to the main memory (e.g., to prefetch
the query or reference sequences to the scratchpad memory).
This approach reduces the number of bus requests and allows
to hide the communication time with the computation time.
The DMA controller can also flag an interruption to the ASIP
whenever a given copy request is finished. This allows the
creation of an interrupt routine that handles all of the data
defetching operations.

To ease the programming task (and compiler development),
the ASIP adopts a typical memory mapped I/O organization.
Therefore, access to the DMA registers (to configure the DMA
transfers and check their status) or to the scratchpad memory
can be performed by using the usual load/store instructions.

B. Synchronization Mechanism

To allow an efficient cooperation between the different
cores, the multi-core architecture includes a multi-register
mutex circuit, with each register supporting two states: \textit{locked}
by core \( k \) and \textit{unlocked}. This circuit works as follows: when
one core attempts to read from an unlocked mutex (register),
a value of ‘1’ is returned and the mutex locks. After that, all
other cores that read from such mutex receive the value ‘0’
until the initial core unlocks it by writing the value ‘1’. All of
the write and read operations are atomic, assured by specific
bus arbitration logic, thus guaranteeing that only one core has
access to a given mutex at any given time.

VI. EXPERIMENTAL RESULTS

To evaluate the proposed ASIP, as well as its integration in
the multi-core processing framework, a thorough performance
analysis of the complete system is presented in this section.
The first analysis evaluates the impact of the proposed SIMD
instruction-set on the required hardware resources and on
the processor’s maximum clock frequency. Afterwards, the
attained processing speedup is evaluated by comparing the
executions of the vanilla (sequential) version of the SW
algorithm and of Farrar’s [7] SIMD version. To assess the
ASIP’s performance, several implementations of the processor
(in different FPGAs) are be compared with two off-the-shelf
low-power processors: \( i \) an Intel Atom E665C, running at
1.3 GHz; and \( ii \) an ARM Cortex-A9, running at 533 MHz.
Furthermore, the ASIP’s performance is also compared to
that of a high performance processing solution, as those
present on autonomous and mobile platforms. In particular,
the proposed processing structure was synthesized targeting
a 90nm CMOS process technology and compared to a state-
of-the-art Intel Core i7 3820 (running at 3.6 GHz), which,
despite its high throughput, is unable to be used in low-power
environments.

The scalability of the proposed multi-core structure with
the number of instantiated processing cores is also analyzed
by considering a raw performance metric (number of Cell
Updates per Second (CUPS)), a raw energy metric (number of
Cell Updates per Joule (CUPJ)) and an energy-delay product
related metric (Cell Updates per Joule-Second (CUPJS)).

to evaluate the performance of the proposed multi-core
processing structure, the complete alignment system was also
prototyped in three different platforms: \( i \) a Xilinx Zynq 7020
SoC; \( ii \) an Altera Arria II GX FPGA; and \( iii \) a Xilinx Virtex
7 FPGA (XC7VX485T). The synthesis and place-&-route for
the FPGAs were performed using Xilinx ISE 14.4 and Altera
Quartus II v12.0. Accurate clock cycle measurements were
obtained by using: \( i \) Modelsim SE 10.0b, for the FPGAs
implementations; \( ii \) the PAPI library, for the Intel Core i7; and
\( iii \) the system timing functions for the Intel Atom and ARM
Cortex-A9. The obtained values were subsequently divided by
the number of repetitions and the processor clock frequency.

A. Biological Benchmarking Setup

In the considered evaluation and performance analyses,
the used Deoxyribonucleic acid (DNA) dataset is composed
of several reference sequences, ranging from 128 to 16384
elements, and a query sequence of length 64. The reference
sequences correspond to a random selection of sub-
sequences of the \textit{Homo sapiens} chromosome Y genomic contig.
Fig. 7. Required hardware resources and corresponding operating frequency of the multi-core system when implemented in different prototyping platforms.

B. Required Hardware Resources and Timing Analysis

To evaluate the attained performance and the resources overhead introduced by the proposed SIMD ISA, the original MB-LITE processor and the proposed ASIP were implemented on the previously referred prototyping devices. For a fair comparison, the multiplier and the barrel-shifter were deactivated in the original MB-LITE core.

The left chart of Fig. 7 presents the hardware resources and the maximum operating frequencies for both the MB-LITE and the ASIP core on the Zynq SoC. As it can be observed, the number of LUTs increased approximately 5x, mostly due to the extra logic that is required for the parallel arithmetic operations and the additional multiplexing logic. The number of registers and RAM/FIFO blocks duplicated, because the increase in the register size, from 32 to 128 bits, requires the use of 2 block-RAMs per read-port, instead of just one. On the other hand, the processor maximum operating frequency decreased by about 42 MHz (21%), mostly due to the added multiplexing logic that is required to implement the SIMD instructions. Nonetheless, this decrease is by far compensated by the larger than 30x increase of the ISA’s performance, as will be described in the following subsections.

The remaining charts of Fig. 7 present the hardware resources occupied by the proposed multi-core ASIP implementations. As expected, these results follow an almost linear relation with the number of instantiated cores.

C. SIMD Instruction Set Evaluation

In order to assess the benefits of the SIMD ISA extension introduced in the proposed ASIP, the number of clock cycles required to execute a DNA sequence alignment procedure was accurately measured. Furthermore, the proposed architecture was also validated by comparing it against three distinct superscalar GPPs: i) the Intel Atom E665C processor; ii) the ARM Cortex-A9 processor; and iii) the Intel Core i7 3820 processor. For this test, both the vanilla (sequential) and Farrar’s SIMD versions of the SW algorithm were considered. The sequence alignment code was compiled with GCC 4.6.2 (using the corresponding back-ends).

In Fig. 8 it is presented the average execution time (in number of clock cycles), to execute the DNA sequence alignment on the proposed ASIP and on the three GPPs. In Fig. 8a, it can be observed that the proposed ISA allows the proposed ASIP to achieve a speedup of up to 33x, which is a much higher value than the speedups obtained with any of the other processors: 7.97x in the ARM Cortex-A9, 17.67x in the Intel Atom E665C and 19.73x in the Intel Core i7 3820.

The second measurement of the ISA efficiency was conducted by comparing the average number of clock cycles required to perform a single sequence alignment cell update

<table>
<thead>
<tr>
<th>Sequence Size</th>
<th>Intel Atom E665C</th>
<th>ARM Cortex-A9</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clock Cycles [×10^6] Speedup</td>
<td>Clock Cycles [×10^6] Speedup</td>
</tr>
<tr>
<td>128</td>
<td>0.494 0.044 11.18</td>
<td>0.245 0.047 5.20</td>
</tr>
<tr>
<td>256</td>
<td>1.003 0.070 14.37</td>
<td>0.502 0.096 5.19</td>
</tr>
<tr>
<td>512</td>
<td>1.948 0.121 16.09</td>
<td>0.990 0.187 5.29</td>
</tr>
<tr>
<td>1024</td>
<td>3.757 0.219 17.13</td>
<td>1.991 0.366 5.44</td>
</tr>
<tr>
<td>2048</td>
<td>7.579 0.449 16.89</td>
<td>3.882 0.799 4.92</td>
</tr>
<tr>
<td>4096</td>
<td>14.758 0.845 17.46</td>
<td>6.969 1.542 5.64</td>
</tr>
<tr>
<td>8192</td>
<td>28.645 1.654 17.32</td>
<td>22.325 3.055 7.31</td>
</tr>
<tr>
<td>16384</td>
<td>56.372 3.190 17.67</td>
<td>48.494 6.081 7.97</td>
</tr>
<tr>
<td>Average</td>
<td>16.01</td>
<td>5.87</td>
</tr>
</tbody>
</table>

(a) Execution time (in clock cycles) for different DNA sequences alignments in the considered execution platforms.

<table>
<thead>
<tr>
<th>Sequence Size</th>
<th>Intel Core i7 3820</th>
<th>Proposed ASIP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Clock Cycles [×10^6] Speedup</td>
<td>Clock Cycles [×10^6] Speedup</td>
</tr>
<tr>
<td>128</td>
<td>0.150 0.031 11.68</td>
<td>0.651 0.094 32.50</td>
</tr>
<tr>
<td>256</td>
<td>0.309 0.052 12.34</td>
<td>1.302 0.094 31.26</td>
</tr>
<tr>
<td>512</td>
<td>0.610 0.084 13.31</td>
<td>2.604 0.108 31.41</td>
</tr>
<tr>
<td>1024</td>
<td>1.200 0.188 13.61</td>
<td>5.209 0.316 31.49</td>
</tr>
<tr>
<td>2048</td>
<td>2.552 0.386 15.98</td>
<td>10.416 0.549 29.86</td>
</tr>
<tr>
<td>4096</td>
<td>5.108 0.691 17.57</td>
<td>20.834 0.627 33.20</td>
</tr>
<tr>
<td>8192</td>
<td>10.872 0.951 19.73</td>
<td>41.665 1.386 30.05</td>
</tr>
<tr>
<td>16384</td>
<td>19.857 1.100 18.22</td>
<td>83.325 2.757 30.22</td>
</tr>
<tr>
<td>Average</td>
<td>15.82</td>
<td>31.23</td>
</tr>
</tbody>
</table>

(b) Average number of clock cycles to perform one single cell update.
in the considered processors. These values were obtained by dividing the total number of clock cycles \((c)\) by the length of the reference and query sequences \((m \times n, \text{respectively})\) – \(c/(m \times n)\) – and are presented in Fig. 8b. From the presented chart, it can be concluded that, even though the off-the-shelf Intel Core i7 and ARM Cortex-A9 processors can issue two instructions per clock cycle, the proposed ASIP offers a 2.3x and 1.5x speedup when compared to them, respectively. On the other hand, the Intel Core i7, using macro-op fusion and multiple instruction issue (up to 6 micro-ops per clock cycle [15]), achieves an average of 1.24 cycles per cell update, whereas the proposed ASIP achieves 2.55 cycles per cell update for a 128-bit SIMD vector. This result is well explained by the fact that, for the considered benchmarks, the Intel Core i7 issues an average of 2 Instructions Per Cycle (IPC), whereas the proposed architecture has an IPC of 1 (by design).

**D. Scalability of the Multi-Core Processing Structure**

To evaluate the performance gains of a multi-core alignment structure, the coarse-grained parallel architecture described in Section V was implemented. All processing cores were based on the proposed ASIP, exploiting the ISA benefits that were presented in the previous subsection. The integrated scratchpad and shared memories were set to an address-width of 15-bit. In order to evaluate the shared bus contention, which constraints the multi-core scalability, the reference sequence used by the alignment algorithm was stored in the shared memory, and each processing core requests access to the bus (once per iteration) to obtain the corresponding symbol. All other variables were stored in the scratchpad memory of each processing core.

Fig. 7 also presents the occupied hardware resources and the maximum operating frequency of the multi-core processing structure for different aggregates of processing cores on different prototyping platforms. A maximum of 8 processing cores can be prototyped in the Zynq SoC and in the Arria II GX FPGA due to the limitations imposed by the available Block-RAM and logic resources, respectively. Regarding the Virtex-7 FPGA, it has sufficient resources to implement the 32 cores of the multi-core prototype system. In all cases, the operating frequency remains practically constant as the number of cores increases to 8. On the Virtex-7 FPGA the frequency has a slight reduction with the number of cores. It can also be observed that the Zynq SoC achieves almost the same operating frequency than the larger Virtex-7 FPGA and an almost 3x higher frequency than the Arria II GX FPGA, which mainly results from the different technology of the latter FPGA.

In Fig. 9 it is presented the clock cycle and the absolute time speedups, compared to a single 128-bit SIMD ASIP core, when processing the considered benchmark dataset. Fig. 9a depicts the architecture scalability in terms of the attained clock cycle speedup. This analysis allows to verify the multi-core architecture scalability independently of the implementation platform. It is possible to observe that the speedup increases almost linearly for configurations of up to 16 cores. With additional cores, the contention in the shared bus becomes the limiting factor, thus reducing the effectiveness of the extra cores and resulting in a sub-linear speedup increase. Nonetheless, when considering the non-SIMD sequential implementation as the speedup reference, a maximum of 830x processing time speedup can be obtained with a 32-core configuration.

Fig. 9b presents the achieved processing time speedup by taking into account the device maximum operating frequency of each individual core configuration.

**E. ASIC Synthesis**

The proposed ASIP was also synthesized using the Faraday Standard Cell Library targeting the UMC 90nm CMOS process (library FSD0A) and an exhaustive performance and energy evaluation was performed. Fig. 10 presents the obtained results in terms of the total circuit area, operating frequency and dynamic power consumption for different period constraints, from 5 ns (200 MHz) to the minimum achievable period constraint for the 90nm technology. As expected, the main drawback of operating the circuit with higher frequencies is the increase in dynamic power, which goes from 24.2 mW (200 MHz) to a maximum of 98 mW (769 MHz) for a single
core system. Despite its increase, this power consumption response respects the low-power requirements of autonomous embedded platforms. This is especially relevant considering that, when embedded in a diagnosis system, the proposed ASIP will only work during reduced amounts of time, after the sample biological sequence is acquired and while the sequence alignment is being performed.

Finally, the complete multi-core structure was also synthesized targeting the same 90nm CMOS process technology. From the synthesis results, presented in Table II (see Section VI-F), it is possible to observe that a maximum operating frequency of 685 MHz is attained up to a 16-core configuration, while the 32-core configuration presents a decrease of 16%.

F. Performance and Energy Efficiency Evaluation

Besides the presented evaluation in terms of the resulting speedup values, several efficiency metrics are also used to study the computational and energy efficiency of the proposed multi-core platform. In particular, three different metrics are used to characterize the multi-core ASIP: i) the attained raw throughput, ii) the energy efficiency, and iii) the performance-energy efficiency. These metrics were also determined for the three considered GPPs. Among them, the Intel Atom and the ARM Cortex-A9 may potentially cope with the energy constraints of the target mobile platform, whereas the Intel Core i7 was only considered in order to compare the performance achieved by the proposed multi-core platform with that of a GPPs.

To compare the attained raw throughput, the CUPS metric was adopted, which is typically used in this application domain. This metric given by $(q \times m \times n)/t$, where $q$ is the number of processed query sequences, $m$ and $n$ are the length of the query and reference sequences, respectively, and $t$ is the corresponding runtime in seconds. The obtained throughputs are shown in Fig. 11b and account for the maximum operating frequency of each implementation platform for the corresponding number of cores.

As it can be concluded from Fig. 11b, the single-core ASIP implementations in the considered FPGAs achieve throughputs very close to that of the ARM Cortex-A9 processor. On the other hand, the two considered Xilinx devices with an 8-core configuration and running at less than 200 MHz attain throughputs similar to that of the Atom processor, running at 1.3 GHz. Finally, a 32-core CMOS implementation is able to achieve a performance similar to that of a 4-core Intel Core i7 running at an around 5x lower clock frequency and consuming around 230x less power (see Fig. 11a).

From the energy efficiency study, performed by using the power estimation tools of the Xilinx ISE and Quartus II frameworks, the GPP Thermal Dissipation Power (TDP) values, from the corresponding data-sheets (see Table I), and the Intel energy performance counters. Table II presents the obtained power consumption for the FPGAs and the ASIC, considering the worst-case power estimation for the used hardware resources at the maximum operating frequencies. For the Intel Atom and the ARM Cortex-A9, the TDP was divided by the number of available cores in each processor, whereas for the Intel Core i7, the performance counters were used to accurately measure the power consumption. In Fig. 11a it is presented the power consumption of the considered platforms, where the division between High Performance Computing (HPC) and embedded platform was purposely set to include the Intel Atom processor.

After obtaining the energy consumption, given by the product of the execution time with the total supplied power, a performance efficiency metric based on the number of cell updates can be obtained in order to study the efficiency of the different configurations. The adopted CUPJ metric is given by the total number of processed cells divided by the total consumed energy. Fig. 11c represents the average CUPJ evolution for different core configurations. From these results, it is possible to observe that the FPGA and the ASIC implementations of the proposed multi-core ASIP clearly surpass the energy efficiency of all of the considered GPPs. Also, with configurations of up to 8 cores, the energy efficiency of the FPGA implementations

![Fig. 10. Frequency, area and power scaling of the synthesized ASIC. The bold line represents the area, while the dashed line represents the dynamic power consumption, both in relation to the operating frequency.](image-url)
increases up to a steady state value, being the implementations on the Zynq FPGA the most efficient among the three. This is explained by the lower dynamic power values, coupled with the exponential growth of the number of processed cells and with the reduced shared bus contention. For the Virtex-7 FPGA, it can be observed that the 16-core configuration presents the highest energy efficiency, hence corresponding to the best trade-off between energy consumption, maximum operating frequency, amount of hardware resources, number of processed cells and shared bus contention. As expected, the ASIC implementation outperforms all other implementations by a factor of over 4x.

The adopted performance-energy efficiency metric, given in Cell Updates per Joule-Second (CUPJS), can be regarded as an inversion and normalization of the commonly used Energy-Delay Product (EDP) metric. In fact, the adopted CUPJS is obtained by inverting the EDP and by multiplying it with the total number of processed cells. Fig. 11d depicts the calculated performance-energy efficiency in CUPJS for the considered platforms. By comparing only the used FPGAs, it is possible to observe that the implementations on both Xilinx devices are almost 4x more efficient than those on the Altera Aria II GX FPGA. On the other hand, when comparing the performance-energy efficiency of the ARM Cortex-A9, it is possible to observe that its efficiency is very close to the efficiency offered by a single-core ASIP implementation on the Virtex-7 FPGA. When looking at the Intel Atom, it is possible to observe that its performance-energy efficiency is higher than that off the ARM processor, by the ASIC implementations on the Aria II GX FPGA and by the single and 2-core ASIP implementations on both Xilinx devices. Nevertheless, the ASIC implementations for 4, 8, 16 and 32 cores on both Xilinx devices are capable of achieving a much higher performance-energy efficiency. It is possible to observe that only the 16 and 32-core implementations of the ASIC on the Virtex-7 FPGA outperform the single-core Intel i7. Finally, the ASIC implementation presents the highest performance-energy efficiency, among all of the considered platforms (the single-core ASIC, providing $58 \times 10^{12}$ CUPJS, is almost 3x as efficient as the Intel Core i7 processor, with $21 \times 10^{12}$ CUPJS).

VII. Conclusions

The currently established biologic sequences alignment algorithms that allow the computation of the optimal alignment solutions by using DP techniques require large run-times when executed on current GPPs. Moreover, even with the highest attainable performances, implementations of those algorithms with ASIC solutions are characterized by their lack of flexibility and the high production costs. The envisaged integration of the developed architecture in embedded platforms, imposes power and performance constraints to the processing components of the system. To tackle this problem, a new specifically adapted ASIP architecture is proposed and a multi-core processing structure is developed.

To exploit a fine-grained parallelism model, the proposed ASIP features an extended SIMD ISA that allows achieving speedup levels of over 30x regarding to a sequential implementation. Furthermore, the in-order single-instruction issue implementation of the proposed SIMD ISA was able to achieve speedups of about 2.3x and 1.5x, when compared with equivalent SIMD implementations on dual-instruction issue NEON and SSE implementations of the ARM Cortex A9 and of the Intel Atom E665C, respectively. In fact, the obtained experimental results show that the architecture of

---

**Fig. 11. Multi-core system performance in comparison with the embedded ARM Cortex-A9 and Intel Atom E665C (depicted by the horizontal lines), and the high-end Intel Core i7 3820.**

(a) Total power consumption for the embedded and HPC platforms (lower is better).

(b) Raw throughput, given in Cell Updates per Second (CUPS) (higher is better).

(c) Energy efficiency, using the Cell Updates per Joule (CUPJ) metric (higher is better).

(d) Performance-Energy efficiency, using an inverted Energy-Delay Product (EDP) metric, the Cell Updates per Joule-Second (CUPJS) (higher is better).
the proposed ASIP, based on a single-instruction issue and a 5-stage pipeline implementation, can achieve a performance comparable to that of an out-of-order Intel Sandy Bridge micro-architecture, issuing 6 micro-operations per clock cycle, demonstrating that the proposed ASIP with the new ISA is especially fit for developing many-core platforms for biological sequence alignment.

To exploit the coarse-grained parallelism model, a many-core platform was developed and prototyped on different FPGA devices. It was demonstrated that a linear speedup can be achieved with up to 16 processing cores, since no relevant contention on the interconnection bus exists. When the number of instantiated processors was further increased, a gradual (but expected) sub-linear behavior was observed in the attained speedup. Nevertheless, when considering the cumulative speedup resulting from using both the SIMD ISA and the multi-core architecture, the proposed system is capable of achieving speedup values as high as 800x, with 32 cores.

Finally, and targeting the integration with biochips and other portable diagnosis systems, a 90nm CMOS implementation of the proposed multi-core processing structure was considered. Experimental results show that a performance level similar to that of an Intel Core i7 processor can be achieved using 20x less energy. This demonstrates the viability of the proposed system on such platforms.

ACKNOWLEDGMENT

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT), under project “HELIX: Heterogeneous Multi-Core Architecture for Biological Sequence Analysis” (reference number PTDC/EEA-ELC/113999/2009), project “Threads: Multitask System Framework with Transparent Hardware Reconfiguration” (reference number PTDC/EEA-ELC/117329/2010) and project PEst-OE/EEI/LA0021/2013.

REFERENCES