High Performance and Resource Efficient Biological Sequence Alignment

Laiq Hasan Zaid Al-Ars Mottaqiallah Taouil
Computer Engineering Laboratory, Delft University of Technology,
Mekelweg 4, 2628 CD, Delft, The Netherlands
E-mail: L.HASAN@TUDELFT.NL

***Abstract*—In this paper, we present a novel method based on hardware partitioning to reduce the execution time and improve the resource utilization of biological sequence alignment, resulting in a higher performance as compared to conventional approaches. The paper shows that the method reduces the execution time and improves the resource utilization up to 33.3%. Further, equations are derived, showing the general trend of execution time reduction, resource utilization improvement and hence performance enhancement.**

***Index Terms*—Bioinformatics, Sequence Alignment, Smith-Waterman Algorithm, High Performance, Resource Utilization**

I. Introduction

Sequence alignment is an essential activity in Bioinformatics, used to identify regions of similarity between DNA or protein sequences [1], [2]. The similarity may be a consequence of functional , structural or evolutionary relationship between the sequences [3]. Various methods are available for local and global sequence alignments [4]. Heuristics based approaches like BLAST, FASTA and HMMER [5], [6], [7] are fast, but they do not guarantee an optimal alignment. Although slow in aligning long sequences, the *Smith-Waterman (S-W)* algorithm [8], based on *dynamic programming (DP)* [9], is a method that finds an optimal local alignment between two DNA or protein sequences (the *query sequence* N_q and the *subject sequence* N_s).

Work has been done on accelerating the S-W algorithm in hardware [10], [11], [12], [13], [14], where the main focus is on full scale implementation, i.e. utilizing the maximum available *processing elements (PEs)* on a given platform. Such implementations improve the performance in terms of cell updates per second, but increase the hardware overhead cost as a consequence. The reason is that the number of unused PEs increases with the increasing length of the query sequence (N_q).

In this paper, we present a novel method based on hardware partitioning to reduce the execution time and improve the resource utilization, resulting in a higher performance as compared to other traditional approaches.

The remainder of the paper is organized as follows:

Section II presents the method to reduce the execution time and improve the resource utilization by hardware partitioning. Section III presents generalized equations to reduce the execution time, improve the resource utilization and hence enhance the performance. Section IV concludes the paper.

II. Hardware Partitioning

For the S-W local alignment, a matrix $H_{i,j}$ is used to keep track of the degree of similarity between the two sequences to be aligned i.e. N_q and N_s. Each element of the matrix $H_{i,j}$ is calculated according to the following equation:

$$H_{i,j} = \max \begin{cases} 0 \\ H_{i-1,j-1} + S_{i,j} \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases} \quad (1)$$

where $S_{i,j}$ is the similarity score of comparing sequence N_q to sequence N_s and d is the penalty for a mismatch.

Figure 1 shows the data flow graph and resource utilization of the S-W hardware implementation, where the length of the query sequence (N_q) is the same as the number of PEs (N). Figure 1(a) shows that the elements in each operation (anti diagonal) are computed in parallel, as they are independent of each other. In Figure 1(b), each row shows the number of PEs available during each operation, whereas the solid black cells represent the number of PEs utilized. Clearly, not all PEs are utilized in the startup and final phase of the calculation, resulting in an increasing overhead cost.

In the following subsections, the method based on hardware partitioning to minimize this overhead cost is presented, leading to reduced execution time, improved resource utilization and hence an enhanced performance. Subsection II-A presents the theoretical concept behind the method and Subsection II-B shows an example to elaborate the method.

A. Theoretical concept

A parallelized S-W algorithm requires $N_s + N_q - 1$ operations for computing the entire $H_{i,j}$ matrix [15]. Since every operation performed by one S-W PE takes time T_{PE}, the total execution time is given by,

$$T_{exec} = (N_s + N_q - 1) \times T_{PE}$$

where $T_{PE} = C_{PE} \times T_{cycle}$, such that C_{PE} is the number of cycles consumed by 1 PE and T_{cycle} is the cycle time.

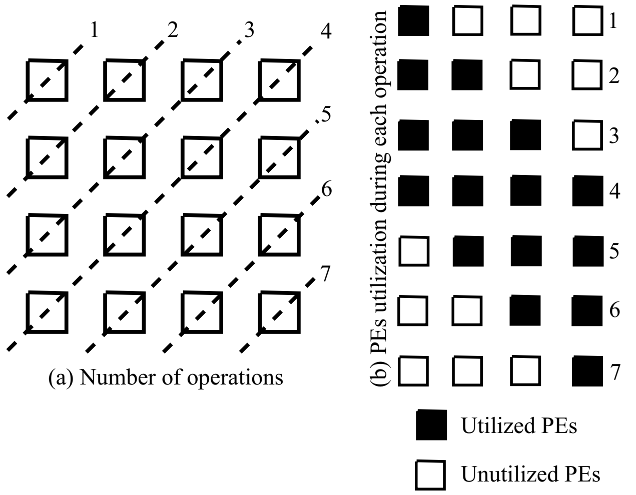

Fig. 1. Number of operations and PEs utilization during each operation

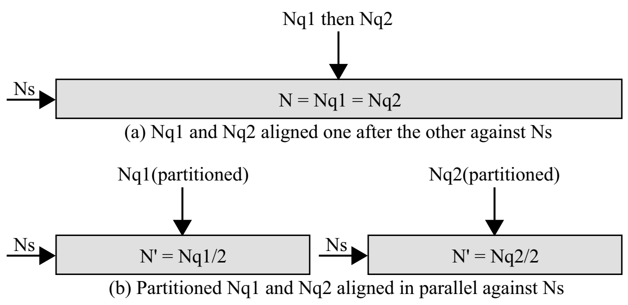
Nq1 then Nq2
Ns
N = Nq1 = Nq2
(a) Nq1 and Nq2 aligned one after the other against Ns
Nq1(partitioned)
Nq2(partitioned)
Ns
N' = Nq1/2
Ns
N' = Nq2/2
(b) Partitioned Nq1 and Nq2 aligned in parallel against Ns

Fig. 2. 2-sequence alignment (a) Sequential (b) Partitioned and in parallel

If two query sequences (N_{q1} and N_{q2}) need to be aligned one after the other against the same subject sequence (N_s), as shown in Figure 2(a),

then the execution time becomes,

$$T_{exec}1 = (2N_s + N - 1) \times T_{PE} \tag{2}$$

where $N_{q1} = N_{q2} =$ The nubmer of PEs (N)

The resource utilization ratio for this case is,

$$\text{Utilization ratio} = \frac{\text{PEs utilized}}{\text{PEs available}} = \frac{2N_s \times N}{(2N_s + N - 1) \times N} = \frac{1}{1 + \frac{N-1}{2N_s}} \tag{3}$$

Figure 2(b) shows that each query sequence is partitioned into two parts and is processed in two passes, in parallel with the other. The number of PEs utilized by each query sequence is half its size, and is given as,

$N^{'} = \frac{N_{q1}}{2} = \frac{N_{q2}}{2} = \frac{N}{2}$

The execution time for this case is given by,

$$T_{exec}2 = (2N_s + N^{'} - 1) \times T_{PE} \tag{4}$$

The resource utilization ratio for this case is,

$$\text{Utilization ratio} = \frac{2N_s \times N^{'}}{(2N_s + N^{'} - 1) \times N^{'}} = \frac{1}{1 + \frac{N^{'}-1}{2N_s}} \tag{5}$$

B. Example of the process

Figure 3 shows an example, where,
$N_{q1} = N_{q2} = N_s = N = 4, \quad N^{'} = 2$

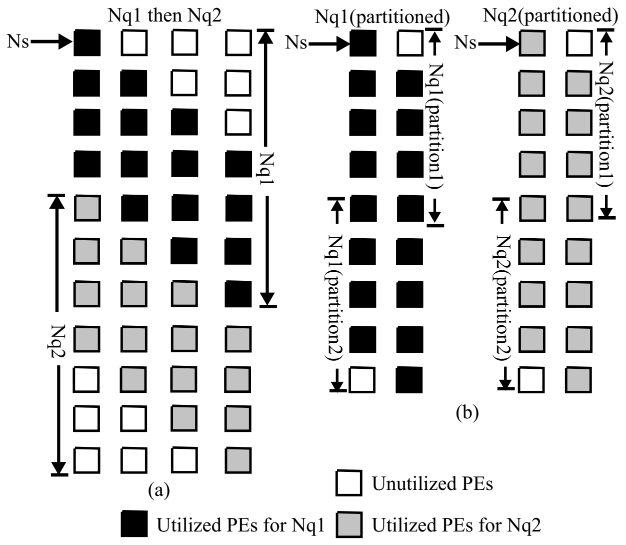

Fig. 3. 2-sequence alignment example (a) Sequential (b) Partitioned and in parallel

Figure 3(a) depicts the case, where two query sequences (N_{q1} and N_{q2}) are aligned one after the other, against the same subject sequence (N_s). The solid black cells in Figure 3(a) represent the data flow and PEs utilization for N_{q1}, whereas the light gray cells for N_{q2}. In Figure 3(b), the hardware is split in two equal partitions such that the two query sequences are aligned in parts and in parallel with each other, against the same subject sequence (N_s). The solid black cells in Figure 3(b) represent the data flow and PEs utilization for N_{q1}, whereas the light gray cells for N_{q2}.

$$\% \text{ Time reduction} = \frac{T_{exec}1 - T_{exec}2}{T_{exec}1} = \frac{(2N_s + N - 1) \times T_{PE} - (2N_s + N^{'} - 1) \times T_{PE}}{(2N_s + N - 1) \times T_{PE}} = \frac{N - N^{'}}{2N_s + N - 1} \tag{6}$$

Substituting values in Equation 6 results in 18.18% reduction in the execution time.

Substituting values for the given example in Equation 3,
Utilization ratio $= \frac{1}{1+\frac{4-1}{2\times 4}} = 0.73 = 73\%$

Similarly, substituting the same values in Equation 5,
Utilization ratio $= \frac{1}{1+\frac{2-1}{2\times 4}} = 0.89 = 89\%$

Thus 16% better resource utilization ratio is achieved by applying the hardware partitioning method.

In practice, the lengths of the query and subject sequences are 500 characters long in most cases [16]. To evaluate a practical case, consider $N_{q1} = N_{q2} = N_s = N = 500, \quad N^{'} = 1$. Substituting these values in Equation 6, a 33.3% reduction in the execution time is achieved. To evaluate the resource utilization improvement, the values are substituted in Equations 3 and 5, showing thereby an improvement of 33.3% in resource utilization.

III. Generalizing the Hardware Partitioning Method

To generalize the method, consider P number of query sequences that needs to be aligned against the same subject sequence (N_s), such that the length of each query sequence is equal to the number of available PEs (N). Figure 4(a) depicts the case, where P query sequences are aligned one after the other against N_s. Here $1 \leq P \leq i$, such that $i > 1$ is an integer. Figure 4(b) shows that the hardware is partitioned into Q parts such that P query sequences are aligned in parts and in parallel with the others, against the same N_s. The number of PEs utilized by each query sequence in this case is, $N^{'} = \frac{N_{q1}}{Q} = \frac{N_{q2}}{Q} = ... = \frac{N_{qi}}{Q} = \frac{N}{Q}$

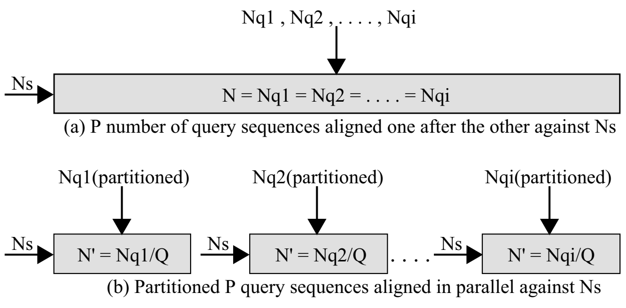
Nq1 , Nq2 , ... , Nqi
Ns
N = Nq1 = Nq2 = ... = Nqi
(a) P number of query sequences aligned one after the other against Ns
Nq1(partitioned)
Nq2(partitioned)
Nqi(partitioned)
Ns
N' = Nq1/Q
Ns
N' = Nq2/Q
. . . .
Ns
N' = Nqi/Q
(b) Partitioned P query sequences aligned in parallel against Ns

Fig. 4. P-sequence alignment (a) Sequential (b) Partitioned and in parallel

The execution time becomes,

$$T_{exec} = (P \times N_s + N^{'} - 1) \times T_{PE} \quad (7)$$

Table I shows the execution time in microseconds for various number of query sequences (Ps) and valid number of hardware partitions (Qs), such that P is divisible by Q. The execution time is computed as per Equation 7, where $N_q = N_s = 500$ and $T_{PE} = 10\ ns$. The table demonstrates that the execution time decreases with the increasing number of hardware partitions (Qs), for all Ps.

Figure 5 shows execution time reduction by applying the hardware partitioning for various number of query sequences (Ps). The T_{exec} versus Q curves, shown in the figure for various number of Ps, demonstrate that the execution time decreases with the increasing number of hardware partitions, where T_{exec} is computed as per Equation 7.

The resource utilization ratio is given by,

$$\begin{aligned}\text{Utilization ratio} &= \frac{P \times N_s \times N^{'}}{(P \times N_s + N^{'} - 1) \times N^{'}} \\ &= \frac{1}{1 + \frac{N^{'}-1}{P \times N_s}}\end{aligned} \quad (8)$$

where the utilization ratio is dependent on the $\frac{N^{'}-1}{P \times N_s}$ term. The smaller the $\frac{N^{'}-1}{P \times N_s}$ term, the better the resource utilization. The $\frac{N^{'}-1}{P \times N_s}$ term in itself decreases with the increasing

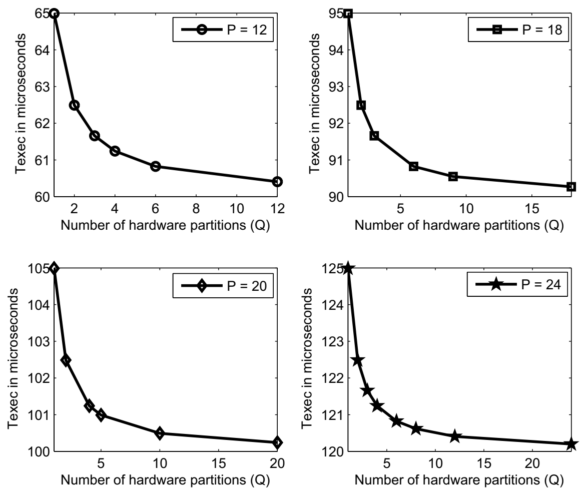

Fig. 5. Execution time reduction by hardware partitioning

number of hardware partitions (i.e. decreasing $N^{'}$), so an increasing number of hardware partitions leads to a better resource utilization, as shown in Figure 6. The figure demonstrates that the resource utilization ratio improves with the increasing number of hardware partitions for various number query sequences (Ps), where the resource utilization ratio is computed as per Equation 8.

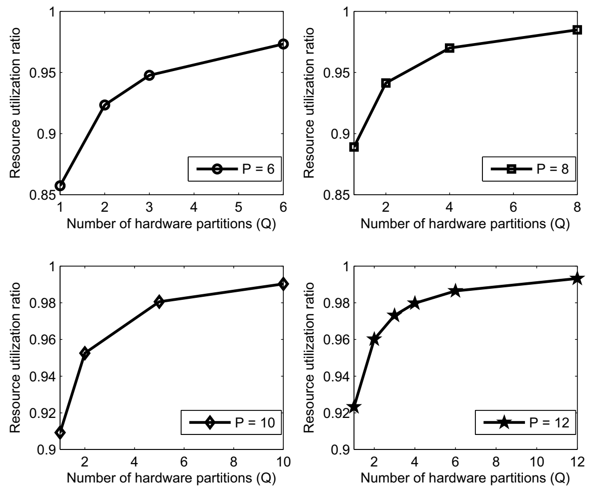

Fig. 6. Resource utilization improvement by hardware partitioning

Table II shows the resource utilization ratio for various number of query sequences (Ps) and valid number of hardware partitions (Qs), such that P is divisible by Q. The resource utilization ratio is computed as per Equation 8, where $N_q = N_s = 500$. The table demonstrates that the resource utilization ratio improves with the increasing number of hardware partitions (Qs), for all Ps.

TABLE I

Execution time (T_{exec}) in μsec for various number of query sequences (Ps) and hardware partitions (Qs)

P \ **Q**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**10**	**12**	**18**	**20**	**24**
12	65	62.5	61.6	61.2	—	60.8	—	—	—	60.4	—	—	—
18	95	92.5	91.6	—	—	90.8	—	90.5	—	—	90.3	—	—
20	105	102.5	—	101.2	101	—	—	—	100.5	—	—	100.2	—
24	125	122.5	121.6	121.2	—	120.8	120.6	—	—	120.4	—	—	120.2

TABLE II

Resource utilization ratio for various number of query sequences (Ps) and hardware partitions (Qs)

P \ **Q**	**1**	**2**	**3**	**4**	**5**	**6**	**8**	**9**	**10**	**12**	**18**
6	0.8574	0.9234	0.9477	—	—	0.9733	—	—	—	—	—
8	0.8891	0.9414	—	0.9699	—	—	0.9849	—	—	—	—
10	0.9093	0.9526	—	—	0.9806	—	—	—	0.9903	—	—
12	0.9232	0.9602	0.9731	0.9798	—	0.9865	—	—	—	0.9933	—
18	0.9475	0.9731	0.9819	—	—	0.9909	—	0.9940	—	—	0.9970

The same theory applies for reducing the execution time and improving the resource utilization ratio for the case, when there is only one query sequence and the database is split into equal parts, such that the same query sequence is scanned against all parts of the database in parallel.

To see the effect of execution time reduction and utilization ratio improvement on performance, we observe the performance equations, given as follows,

$$\text{Performance} = \frac{N_q \times f_{op}}{C_{PE}} \times \text{Utilization ratio} \qquad (9)$$

where f_{op} is the operating frequency.

Performance may also be given by,

$$\text{Performance} = \frac{\text{Total operations}}{T_{exec}} = \frac{N_q^2}{T_{exec}} \qquad (10)$$

Equations 9 and 10 imply that higher resource utilization ratio and lower execution time improve the performance.

In comparison with the traditional approaches, the initialization process for the proposed hardware partitioning method is modified, such that the initialization input is equal to a predefined value at the start of the computation. For every succeeding array computation, the initialization input is a feed back from the last PE in the partitioned array.

IV. Conclusion

In this paper, we presented a novel biological sequence alignment method to reduce the execution time and improve the resource utilization, resulting in a higher performance as compared to conventional approaches. Generalized equations for reducing the execution time, improving the resource utilization and hence enhancing the performance are presented. The paper demonstrated that with the increasing number of hardware partitions (i.e. decreasing $N^{'}$), an execution time reduction and resource utilization improvement of up to 33.3% is achieved.

References

[1] Martin Vingron and Michael S. Waterman, "Sequence Alignment and Penalty Choice: Review of Concepts, Case Studies and Implications", *Journal of Molecular Biology*, vol. 235, pages 1–12, 1994.

[2] Asim YarKhan and Jack J. Dongarra, "Biological Sequence Alignment on the Computational Grid Using the GrADS Framework", *Future Generation Computer Systems*, vol. 21(6), pages 980–986, June 2005.

[3] D.M. Mount, "Bioinformatics: Sequence and Genome Analysis", *Cold Spring Harbor Laboratory Press*, Cold Spring Harbor, NY, 2nd ed., 2004.

[4] L. Hasan, Z. Al-Ars and S. Vassiliadis, "Hardware Acceleration of Sequence Alignment Algorithms - An Overview", *Proceedings of International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS'07)*, pages 96–101, Rabat, Morocco, September 2–5, 2007.

[5] S. F. Altschul, Gish, W. Miller, W. Myers and D. J. Lipman, "A Basic Local Alignment Search Tool", *Journal of Molecular Biology*, vol. 215, pages 403–410, 1990.

[6] W. R. Pearson and D. J. Lipman, "Rapid and Sensitive Protein Similarity Searches", *Science*, vol. 227, pages 1435–1441, 1985.

[7] Sean R. Eddy, "Profile Hidden Morkov Models", *Bioinformatics Review*, vol. 14(9), pages 755–763, July 1998.

[8] T. F. Smith and M. S. Waterman, "Identification of Common Molecular Subsequences", *Journal of Molecular Biology*, vol. 147, pages 195–197, 1981.

[9] R. Giegerich, "A Systematic Approach to Dynamic Programming in Bioinformatics", *Bioinformatics*, vol. 16, pages 665–677, 2000.

[10] A. B. Buyukkur and W. Najjar, "Compiler Generated Systolic Arrays for Wavefront Algorithm Acceleration on FPGAs", *International Conference on Field Programmable Logic and Applications (FPL08)*, Heidelberg, Germany, September 2008.

[11] A. Di Blas et al., "The UCSC Kestrel Parallel Processor", *IEEE Transactions on Parallel and Distributed Systems*, vol. 16(1), pages 80–92, 2005.

[12] A. Schroder et al., "Bio-Sequence Database Scanning on a GPU" *HICOMB*, 2006.

[13] L. Hasan and Z. Al-Ars, "An Efficient and High Performance Linear Recursive Variable Expansion Implementation of the Smith-Waterman Algorithm", *31st Annual International Conference of the IEEE EMBS*, pages 3845–3848, Minneapolis, Minnesota, USA, September 2009.

[14] L. Hasan, Z. Al-Ars, Z. Nawaz and K.L.M. Bertels, "Hardware Implementation of the Smith-Waterman Algorithm Using Recursive Variable Expansion", *Proceedings of 3rd Inernational Design and Test Workshop IDT08*, Monastir, Tunisia, December 2008.

[15] L. Hasan, Z. Al-Ars, M. Taouil and K.L.M. Bertels, "Performance Optimization for Biological Sequence Alignment", *Submitted to the 28th IEEE International Conference on Computer Design (ICCD 2010)*, Amsterdam, The Netherlands, October 3–6, 2010.

[16] T. Oliver, B. Schmidt and D. Maskell, "Hyper Customized Processors for Bio-Sequence Database Scanning on FPGAs", *FPGA'05*, Monterey, California, USA, February 20–22, 2005.