Asian Journal of Research in Computer Science

1(2): 1-10, 2018; Article no.AJRCOS.42834

Acceleration of Biological Sequence Alignment Using Residue Number System

Hassan Kehinde Bello[1*] and Kazeem Alagbe Gbolagade[2]

[1]Department of Computer Science, Federal Polytechnic, Offa, Nigeria.
[2]Department of Computer Science, Kwara State University, Malete, Nigeria.

Authors' contributions

This work was carried out in collaboration between authors HKB and KAG. Both authors read and approved the final manuscript.

Article Information

DOI: 10.9734/AJRCOS/2018/v1i224735
Editor(s):
(1) Dr. Stephen Mugisha Akandwanaho, Department of Information Systems and Technology, University of KwaZulu-Natal, South Africa.
Reviewers:
(1) R. Mahalakshmi, India.
(2) Geraldo Francisco Donegá Zafalon, Sao Paulo State University, Brazil.
(3) Manish Mahajan, CGC College of Engineering, India.
Complete Peer review History: http://prh.sdiarticle3.com/review-history/25615

Short Research Article

Received 2nd May 2018
Accepted 11th July 2018
Published 19th July 2018

ABSTRACT

Smith-Waterman Algorithms (SWA) is becoming popular among researchers especially in the field of bioinformatics. The algorithm performance is better among other known alignment algorithms because of the high level of accuracy it exhibits. However, the algorithm performance is at low speed due to its computational complexity. Researchers are concerned with this problem and are looking for various ways to address the issue. Different approaches are adopted to improve the speed, such as the use of a systolic array to accelerate the algorithm, use of recursive variable expansion (RVE) method approach; some implemented the algorithm on software and hardware, etc. This paper used Residue Number System (RNS) approach to the algorithm of Smith-Waterman and carried out hardware implementation on Quartus II, 64-Bit version 12.1 (Cyclone II family) VHDL application software.

Keywords: Smith-Waterman algorithm; bioinformatics; algorithm; recursive variable expansion; computational complexity; residue number system.

**Corresponding author: E-mail: hassan.bello@fedpoffaonline.edu.ng;*

1. INTRODUCTION

The importance of biological sequence alignment cannot be overemphasized in computational biology, where functionalities are compared to find the optimal alignment between any two sequences of varying or same lengths. Alignment score is computed based on the total number of gap penalty, matches and mismatches in the alignment. Biological sequence alignment helps in the discovery of functional, structural and evolutionary information in the sequences of DNA, RNA and Proteins [1].

Biological sequence alignment is classified into local and global alignment. In local sequence alignment, the optimal value is calculated from the most similar sub-region common to both sequences while in global alignment, the two sequences must be of similar length and the optimal value is computed from beginning to the end of the sequences [2].

There are many techniques proposed to solve the sequence alignment method, like Dynamic Programming (DP), Heuristic, Linear programming, Hidden Markov model and T-Coffee [3]. Smith-Waterman algorithm (SWA) and Needleman Wunsch algorithm (NWA) are based on dynamic programming (DP) technique with space and time complexity of O(*mn*) [4] where *m* and *n* are the lengths of the two sequences being considered for alignment. Heuristic algorithms such as Fast-All (FASTA) and Basic Local Alignment Search Tool (BLAST) are based on approximations and fast at the expense of accuracy while the NWA and SWA algorithms are commonly used to find global and local alignment respectively [5,6,7,8,9].

SWA is the most common biological sequence alignment algorithms; it is very accurate with a high computational cost at the expense of speed. These constraints of SWA have been tolerated in the past, but due to exponential growth in the size of the biological database, there is a need to enhance the acceleration of the algorithm. Some researchers approached the problem using a systolic array to accelerate the algorithm, and some use recursive variable expansion, some implemented the algorithm on software and hardware, etc. Residue Number System (RNS) is a number system that can improve the acceleration of any application that uses the operations of addition, subtraction and multiplication. It has also been successively applied in Fourier application, Digital Signal Processing, digital filtering, image processing, cryptography, among others.

2. BACKGROUND

The Residue Number System (RNS) is an integer that is capable of supporting high speed concurrent automatic [10]. It may be defined as a set of relatively prime integer called moduli which can be denoted as m_1, m_2, m_3 …m_n, where m_i is the i^{th} modulus [11] and gcd(m_i, m_j) = 1 for i≠j . Each X can be represented as a set of smaller integers called $x_1,x_2,x_3…x_n$, where X mod m_i = x_i; $x_i = |X|_{mi}$

RNS has the capabilities to support parallel carry-free addition, borrow-free subtraction and single step multiplication without partial fraction [12,13,14]. Many moduli sets are used in RNS with different Dynamic Range (DR). The most popular 3n-bit DR moduli set known as traditional moduli set is $\{2^n-1, 2^n, 2^n+1\}$ [15]. Others are $\{2^{n-1}, 2^n-1, 2^n+1\}$, $\{2^{n-1}-1, 2^n-1, 2^n\}$ [16]. The DR of 3n-bits produced by this moduli set can be used by any application with higher DR. 4n-bit DR 4 moduli set such as $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$, $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$, $\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1\}$ [17] were suggested. In order to raise parallelism in RNS arithmetic, 5n-bit moduli set were suggested $\{2^n, 2^{2n}-1, 2^{2n}+1\}$, $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ etc were also proposed.

2.1 Selection of Moduli in RNS

In RNS, the Dynamic Range (DR) is the product of all the moduli such that the interval can be uniquely represented in RNS [18,19], this is an important aspect to be considered in the choice of moduli in RNS. Proper choice of moduli when designing RNS system is very essential [20]; this is because the moduli choice affects the complexity of forward conversion, reverse conversion and RNS arithmetic circuits. Also, the speed of the resulting conversion depends on selected moduli [21]; therefore, moduli selection and data conversion are very critical in RNS to binary conversion [20,22]. However, it is a common fact that as the number of moduli increases, the speed of residue arithmetic units increases and the conversion from residue-binary becomes slower and complex [23]. Abdullahi and Skavantzos stated that the moduli set m_i and m_j satisfy the following criteria:

- They must be pairwisely prime (i.e. gcd(m_i, m_j) = 1
  Each moduli m_i should be as small as possible so that operation modulo mi require minimum computation time
- The moduli m_i, m_j should imply simple binary to Residue Number system and Residue Number System to binary conversion.
- The moduli product should be large enough to implement the decision dynamic range.

However, in a case where a large dynamic range (DR) is required, large moduli might be in a better performance. In addition, another fundamental area where attention should be taken is the hardware selection, which is a determining factor for RNS performance. The simplified conversion equation of the system is computed by using adders, like Carry Propagation Adder (CPA), Carry Save Adder (CSA) [24,25], etc.

2.1.1Lemma

If $m_1, m_2, \ldots m_n$ are set of moduli, then the dynamic range (DR) is the product of all the moduli set, say M, then every representable number (X) satisfy either of the following

$$-\frac{M-1}{2} \leq X \leq \frac{M-1}{2} \text{ if M is odd} \quad (1)$$

$$-\frac{M}{2} \leq X \leq \frac{M}{2} - 1 \text{ if M is even} \quad (2)$$

2.2 The Smith-Waterman Algorithm

The optimal local alignment of two sequences X and Y is given by the algorithm of SW [26] in equation (3) for the computation of local alignment of matrix $M_{i,j}$

$$M(i,j)=\text{Max}\begin{cases} 0 \\ M(i-1,j-1) + S(x_i,y_j) \text{ match/mismatch} \\ M(i-1,j) + g \\ M(i,j-1) + g \end{cases} \quad (3)$$

where M(0,0) = 0, M(0,j) = g × j and M(i,0) = g × i, for 1 ≤ i ≤ n,1 ≤ j ≤ m. The g is the penalty for inserting a gap in any of the sequence and M(i,j) is the score for match/mismatch, depending upon whether X[i] = Y[j] or X[i] ≠ Y[j] [27]. The dynamic programing step of a matrix M to compute maximum value of M(i,j) is shown in Fig. 1.

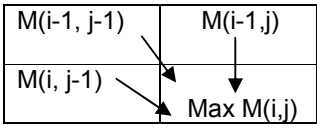

Fig. 1. Dynamic programming step

The time complexity of the initialization step is O(*m* + *n*), where *m* is the number of rows and *n* is the number of columns in the matrix M.

2.2.1 Steps in the algorithm of Smith-Waterman

Step 1: Initialization:

At the initialization step, the matrix M(i,j) will be initialized with $M_{0,j}$ = 0 and $M_{i,0}$ = 0, for all i and j as shown in Fig 2. At this step, the time complexity is given by is O(*m* + *n*), where *m* is the total number of rows and *n* is the total number of columns in the sequences X and Y respectively.

Example:

Dynamic programming of a matrix M is illustrated in Fig. 2, Fig. 3 and Fig. 4.

Given the sequences X: GAGATC and Y: GCTAGCT with match = 2, mismatch = 1 and gap = 1

Step 2: Matrix filling

The equation 1 will be used to fill up all the entries in the matrix Mi,j based on given scoring parameter (fig. 4). At this step, the time complexity is equal to the number of cells in the matrix i.e O(*mn*).

Example:

Given the sequences X: AGGTCA and Y: CGTGTAA with match = 2, mismatch = 1 and gap = 1

Step 3: Trace back

At the end of step 2 above, the cell with the highest score will be located at the bottom right of the matrix and traced back to get the optimal local alignment, as shown in Fig 5. The time complexity of the trace back is given by O(*m*+ *n*).

		G	A	G	A	T	C
	0	0	0	0	0	0	0
G	0						
C	0						
T	0						
A	0						
G	0						
C	0						
T	0						

Fig. 2. Initialization step

		G	A	G	A	T	C
	0	0	0	0	0	0	0
G	0	2	1	2	1	1	1
C	0	1	3	2	3	2	3
T	0	1	2	4	3	5	4
A	0	1	3	3	6	5	6
G	0	2	2	5	5	7	6
C	0	1	3	4	6	6	9
T	0	1	2	4	5	8	8

Fig. 3. Matrix filling

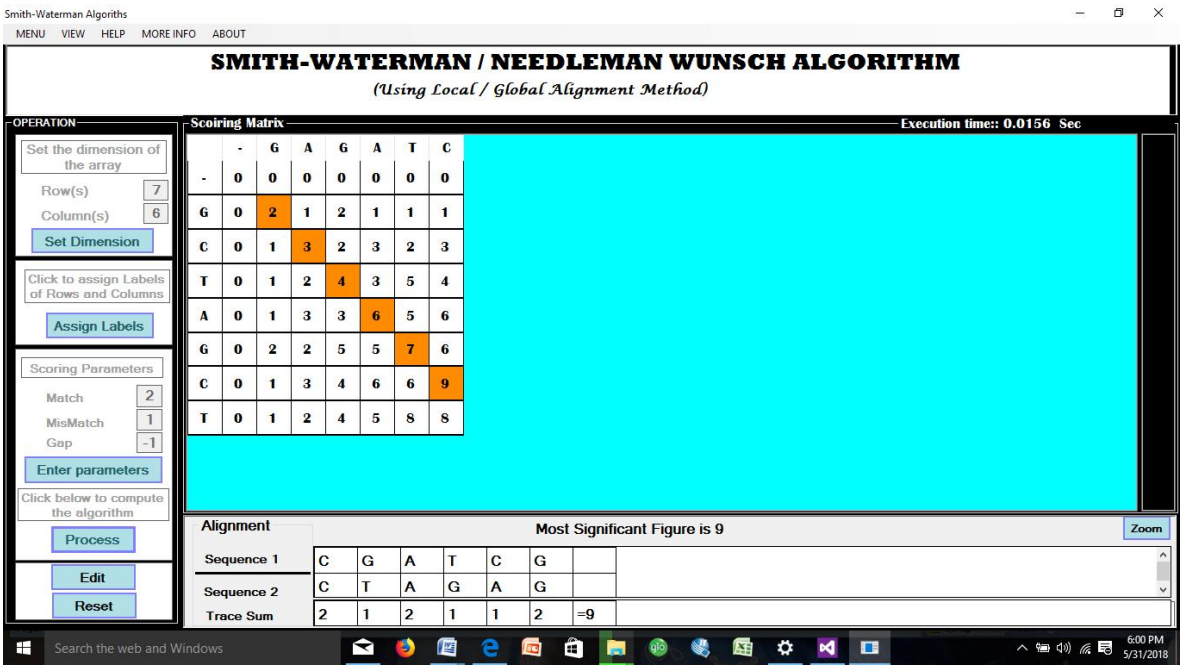

Fig. 4. Trace back

At the end of step 3, the total time complexity of Smith-Waterman algorithm is given by O(*m*+*n*) + O(*mn*) + O(*m*+*n*) = O(*mn*). The space complexity of the algorithm is given by O(*mn*) because the size of the matrix is *m* x *n* [26]. The corresponding optimal alignment of the two sequences with the best score of 9 is computed in fig. 4.

3. METHODOLOGY

This research work aims to apply RNS to improve the speed of SWA, The functions in the algorithm were studied and identified the one that consumes most time as depicted in table 4. Therefore, the speed of this function is improved through the three RNS moduli set m = 2^{2n+1}-1, 2^{n}-1, 2^{2n}-1. In the implementation, our n = 2 gives m = {31,2,15} with a dynamic range (DR) of 930. Since the DR is an even number, equation (2) is applied, meaning the range of numbers that can be represented by this scheme is given to be -465 $\leq$ X $\leq$ 464, and therefore, the values representable by this dynamic range for signed number are given in Table 1. On applying the scoring parameters (gap, match/mismatch) and RNS to equation (3), the range of our values will fall within Table 1.

3.1 The RNS- SWA Conversion and Design Entry

The first step is to convert the number in binary number to RNS format. The memoryless RNS converter is input into Quartus II version 12.1 VHDL application software using Altera Cyclone II EP2C20F484C7. This allows the use of tools embedded in the software to create a logic design. The software directs the RNS comparator to pick the best alignment value after several comparisons, as shown in Fig. 5. The best alignment is produced at a reduced computation time.

3.1.1 The RNS processor

After binary conversion to RNS numbers, it follows RNS processing where the inherent properties of RNS like carry free addition, borrow free subtraction, etc. will be applied with respect to equation (3) as displayed in Fig. 5. The sequence involve (i) addition of upper diagonal elements, M(i-1,j-1) + $S_{i,j}$; addition of left elements, M (i,j-1) + g and (ii) addition of upper elements, M (i-1,j) + g where g and $S_{i,j}$ are the gap and match/mismatch values respectively. This is sequence been controlled by the control unit.

3.1.2 The compilation and simulation step

The compilation is done in order to convert the source code to object code which is logically down loaded to a target device and the software simulation is carried out by the software to ensure that the logic circuits function as expected.

Table 1. Table of residues for MOD 31, MOD 15 and MOD 2 for signed numbers in hexadecimal numbers

Decimal number	Hexadecimal number	MOD(31,15,2)
-465	E2F	(4,1,1)
-464	E30	(5,1,0)
-463	E31	(6,2,1)
.	.	.
.	.	.
.	.	.
-4	FC	(4,C,0)
-3	FD	(5,D ,1)
-2	FE	(6,E,0)
-1	FF	(7,F,1)
0	0	(0,0,0)
1	01	(1,1,1)
2	02	(2,2,0)
3	03	(3,3,1)
4	04	(4,4,0)
.	.	.
.	.	.
.	.	.
462	1CE	(28,12,0)
463	1CF	(29,13,1)
464	1D0	(30,14,0)

3.1.3 The RNS forward converter

The Moduli set of this research work M = (31, 15, 2) will give a DR of 930. The inputs variables (M(i,j-1), M(i-1,j), M(i-1,j-1), S(i,j) and d) in decimal/binary are converted to residue number system by binary to RNS converter. This is referred to as the forward converter [28]. The three RNS processors then act on the input values by applying the inherent properties of RNS i.e. the carry-free addition, borrow-free subtraction, etc. Each of the three processors is independent of one another, as shown in Fig 6.

3.2 The RNS Comparator

The comparator in Fig 5 computes the final optimal value M(i,j) from M(i-1,j), M(I,j-1), M(i-1,j-1) using equation 3. It comprises of thee registers and two multiplexers. The RNS-SWA comparator is used in a Quartus II, 64-Bit version 12.1 (Cyclone II family) VHDL application software and the graphic entry or schematic capture tool embedded in the software are fully used. This is later compiled in order to translate the source code to object code. A: typical RNS-

SWA based signal processor of our design is shown in Fig. 5.

Table 2 shows the parallel compilation and table 3 shows the summary report of the final compilation. 2 out of 18,752 total logic elements within the device are used and a negligible number of the logic cell 190 / 12,400 (1%) within the device are also used when implemented on EP2C20F484C7 device (Cyclone II).

Table 2. Parallel compilation report

Usage by Processor	% Time Used
3-4 processors	0.0%
2 processors	20.0%
1 processor	100.0%
Number detected on machine	4
Maximum used	2
Maximum allowed	2
Average used	1.33

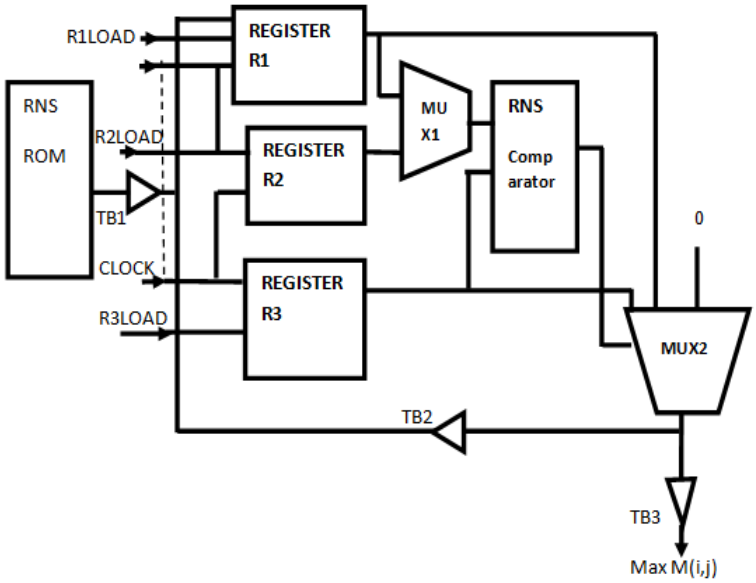

Fig. 5. RNS comparator schematic diagram

Table 3. Flow summary of SWA_RNS processor

Status	Successful - Tue May 29 18:17:46 2018
Quartus II 64-Bit Version	12.1 Build 243 01/31/2013 SP 1 SJ Web Edition
Revision Name	RNS_COMPARATOR
Top-level Entity Name	RNS_COMPARATOR
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Cyclone II
Total logic elements	2 / 18,752 (< 1 %)
Total combinational functions	2 / 18,752 (< 1 %)
Dedicated logic registers	0 / 18,752 (0 %)
Total registers	44
Total pins	7 / 315 (2 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Total ALUTs	190 / 12,400 (1 %)
Embedded Multiplier 9-bit elements	1 / 52 (< 1 %)
Total PLLs	0 / 4 (0 %)

4. PERFORMANCE EVALUATION

To inspect the performance evaluation of our work, we compare it with another Hasan & Al-Ars (2007) in the state of the art.

From Table 4, the labeled Fill_Matrix_2 is identified to have consumed 72.33% of the total time. It is this Fill_Matrix_2 that needs to be implemented on VHDL. The time 5.23 ms is the total time when the code is repeated 100 times.

The time consumed by matrix fill stage function is 0.0523 ms with hardware delay 0.0146*μs* and the time consumed by software equivalent is 52.32 *μs*

$$\% \text{ runtime} = \left\{ \frac{\frac{1}{0.0146\ x\ 10^{-6}} - \frac{1}{52.3\ x\ 10^{-6}}}{\frac{1}{52.32\ x\ 10^{-6}}} \right\} \text{x } 100$$

= 3582% (35.82 times)

Timing simulation of RNS-SWA architecture critical delay is 10.38 ns. Comparison between the total delay [27] of hardware implementation denoted as LZ_hw and hardware implementation of this work is HK_hw is given as the percentage runtime improvement over their work.

The % runtime improvement ratio of the RNS-SWA implementation to [27] is computed as

$$\left(\frac{\frac{1}{HK\_hw}}{\frac{1}{LZ\_hw}} \right) \text{x } 100$$

$$\% \text{ runtime ratio} = \frac{14.6 \text{ x } 10^{-9}}{10.38 \text{ x } 10^{-9}} \text{ x } 100$$

= 140.63%

Runtime improvement over the hardware = $\left\{ \frac{\frac{1}{10.38 \text{ x } 10^{-9}} - \frac{1}{14.6 \text{ x } 10^{-9}}}{\frac{1}{14.6 \text{ x } 10^{-9}}} \right\} \text{x } 100$

= 40.66%

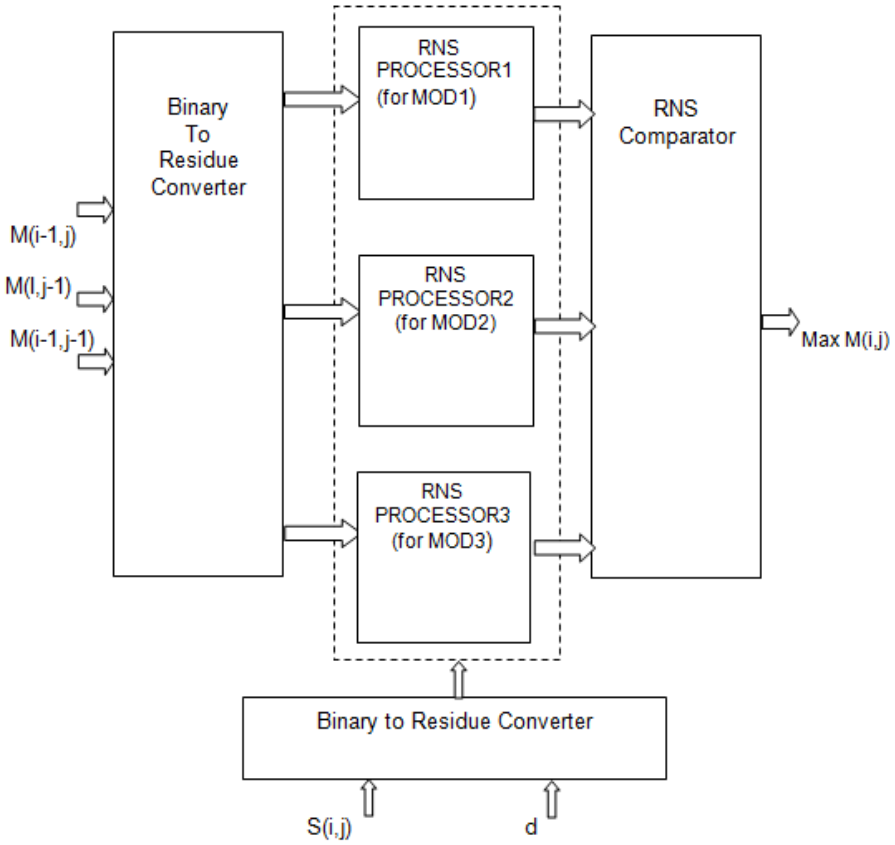

Fig. 6. A typical RNS-SWA based signal processor

Table 4. [27] Result for software implementation of SWA

Function	No. of calls	No. of clock ticks	No. of clock CYCLES	Total Time (ms)	% Time
Initial_Matrix	100	71944	2302208	0.718	9.93
Fill_Matrix_1	100	32392	1036544	0.323	4.47
Fill_Matrix_2	4800	524040	16769280	5.23	72.33
Trace_back_1	100	31232	999424	0.312	4.31
Trace_back_2	500	64944	2078208	0.648	8.96

5. RESULTS AND DISCUSSION

The algorithm of SWA uses dynamic programming approach; Hasan & Al-Ars (2007) [27] identifies the part of the function that consumes more computational time and attempts to improve the speed of that function. This slight improvement in speed enhances the performance of the entire algorithm. The aim of this research work is to improve the record of Hasan & Al-Ars (2007) [27] whose work is known to be faster on the state of the art. This research work is centered on the function that consumes more computational time in the algorithm of Smith-Waterman. The residue number system was introduced to solve the problem and compared with the results of Hasan & Al-Ars (2007) [27]. The percentage speed gained in our work is 140.63% faster than [27] and also the run-time of the hardware implementation of our research work is 40.66% better than [27].

6. CONCLUSION

This research work investigates the likely possibility of using Residue Number System in the implementation of the algorithm of Smith-Waterman. Three moduli sets were used with a dynamic range of 5n-bits and designed RNS comparator. The simulated result of our research work is compared with the work of Hasan and Al-Ars (2007) [27] in terms of speed and hardware implementation. Our findings show a better improvement than the work of Hasan and Al-Ars [27]. This means there is hope for the bioinformatics community in addressing the speed bottleneck in SWA. With our result, RNS is a good candidate to implement the algorithm of Smith-Waterman.

COMPETING INTERESTS

Authors have declared that no competing interests exist.

REFERENCES

1. Zubair N, Zaid A, Koen Bertels, Mudassir Shabbir. Acceleration of Smith-Waterman using recursive variable expansion. Conference Paper; 2008. DOI:10.1109/DSD.2008.32 Source:DBLP
2. Hassan KB, Kazeem AG. Application of Smith-Wateman and NeedlemanWunsch algorithm in pairwise sequence alignment of deoxyribonucleic acid. Proc. of the 1st International conference of IEEE Nigeria Computer Chapter In collaboration with Dept. of Computer Science, University of Ilorin, Ilorin, Nigeria; 2016.
3. FN Muhamad, RB Ahmad, SM Asi MN Murad Performance analysis of Needleman-Wunsch algorithm (Global) and Smith-Waterman Algorithm (Local) In reducing search space and time for DNA sequence alignment. 1st International Conference on green sustainable computing. IOP Conf. series: Journal of Physics: Con. Series 1019 (2018) 012085. DOI 10.1088/1742-6596/1019/1/012085
4. Zubair N, Zaid A, Koen B, Mudassir S. Acceleration of Smith-Waterman using recursive variable expansion. Conference Paper. DOI: 10.1109/DSD.2008.32
5. Needleman S, Wunsch C, A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology. 1970;48:443–453.
6. Altschul SF, Gish W, Miller W, Myers EW. Lipman DJ. Basic Alignment search tool. J. Mol. Biol. 1990;403-410.
7. Smith T, Waterman M. Identification of common molecular subsequences. J. Mol. Biol. 1987;147:195–197.
8. Choi Y, Sims GE, Murphy S, Miller JR, Chan AP. Predicting the functional effect of amino acid substitutions and indels. PloS One. 2012;7(10):e46688.

9. Lee BJ, Shin MS, Oh YJ, Oh HS, Ryu KH. Identification of protein functions using a machine-learning approach based on sequence-derived properties. Proteome Science. 2009;7(1):27.
10. Andreas Pearson-Lars Bengttson. Forward and reverse converters and moduli set selection in signed digit residue number system; 2008.
11. Omar Abdelfattah. Data Conversion in Residue numbers system. A thesis submitted to Department of Electrical & Computer Engineering McGill University Montreal, Canada; 2011.
12. Bello HK, Gbolagade KA. A MRC Based RNS to binary converter using the moduli set $\{2^{2n+1}-1, 2^{n-1}, 2^{2n}-1\}$. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2017;6(7).

    ISSN: 2278 – 1323.
13. Siewobr H, Gbolagade KA. Modulo operation free reverse conversion in the $\{2^{2n}+1-1, 2^{n}, 2^{2n}-1\}$ moduli set. International Journal of Computer application (0975 b-8887). 2014;85(18).
14. Kazeem AG An efficient MRC based RNS to Binary Converter for the $\{2^{2n}-1, 2^{n}, 2^{2n+1}-1\}$ moduli set. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2013;2(10).
15. MohammedReza T, Elham K, Mohammad E, Keivan N. Efficient reverse converter design for five moduli set $\{2^{n}, 2^{2n+1}-1, 2^{n/2}-1, 2^{n/2}+1, 2^{n}+1\}$. Journal of Computations and Modeling. 2012;2(1):93-108.
16. Wang W, Swang MNS, Ahmad MO, Wang Y. A high speed residue to binary converter and scheme of VLSI implementation. IEEE Transaction Circuit System II Analog Digital Signal Processing. 2000;47(12):1576-1581 .
17. Cao B, Strikantan T, Chang CH. Efficient reverse converters for the our- moduli sets $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{n+1}-1\}$ and $\{2^{n}-1, 2^{n}, 2^{n}+1, 2^{n-1}-1\}$ proc. IEEE Comput. Digit Tech. 2005;152:687–695 .
18. Rami A, Mehdi H, Mehdi G. A novel High Dynamic range 5-moduli set $\{2^{2n+1}, 2^{2n}+1, 2^{n}+1, 2^{n/2}+1, 2^{n/2}-1\}$ with efficient reverse converter and review improving modular multiplication dynamic range. Journal of Global Research in Computer Science. 2012;3(1).
19. Amir Sabbagh Molahosseini, Azadeh Alsadat Emrani Zarandi, Paulo Martins, Leonel Sousa. A multifunctional Unit for designing efficient RNS-based datapaths. INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal. Digital Object Identifier. 2017;1600-276.

    DOI: 10.1109/ACCESS.2017.2766841
20. Gbolagade KA, Chaves R, Sousa L, Cotofana SD. An Improved RNS reverse converter for the $\{2^{2n+1}-1,2^{n},2^{n}-1\}$ moduli set IEEE International Conference on Circuits and Systems (ISCAS 2010). Paris, France. 2010;2103-2106.
21. Abdullahi M, Skavantzos A. A system approach for selecting practical moduli set for RNS. In Proceeding of the 27th IEEE Symposium in System Theory. 1995;445–449.
22. Premkuma B. Corrections to An RNS to binary converter in a three moduli set with common factors, IEEE Trans. On Circuits and Systems-II: Analog and Digital Processing. 2004;51(1):43.
23. Ing. Dina Younes. Residue Number System based building blocks for applications in digital signal processing. Available:https://coredownload/pdf
24. Augusta Angel M, Vijay MM. High speed RNS to binary converter design using parallel prefix adders.International Journal of innovative Research in computer and communication Engineering. An ISO 3297:2007 Certified Organization. 2015;3.
25. Kuttimani M, Rajalingam A. Muthumanicckam, Mrs. R. Sornalatha Design and implementation of RNS reverse converter using parallel prefix adders. International Journal of Computer Applications (0975 - 8887). 2015;117(6).
26. Hassan Kehinde Bello, Kazeem Alagbe Gbolagade. Acceleration of algorithm of Smith-Waterman using recursive variable expansion. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET). 2018;7;(5).

    ISSN: 2278 – 1323.
27. Hasan Z. Al-Ars. Performance improvement of the Smith-Waterman algorithm. Annual Workshop on Circuits, Systems and Signal Processing (ProRISC 2007), Veldhoven, The Netherlands; 2007.

28. Kwame O, Boatteng, Edward Baagyere. A Smith-Waterman algorithm accelerator based on residue number system. International Journal of Electronics and Communication Engineering. International Research Publication. 2012;5:1.

Peer-review history:
The peer review history for this paper can be accessed here:
http://prh.sdiarticle3.com/review-history/25615