European Journal of Computer Science and Information Technology
Vol.9, No.2, pp.35-60, 2021
Print ISSN: 2054-0957 (Print),
Online ISSN: 2054-0965 (Online)

ACCELERATING SMITH WATERMAN ALGORITHM FOR OPTIMIZING GENE SEQUENCES ALIGNMENT USING PARALLEL RESIDUE NUMBER SYSTEM ARITHMETIC BASED ARCHITECTURE.

Olatunbosun Lukumon Olawale[1]
ICT Department of Computer Science,
The Federal University of Agriculture, Abeokuta, Nigeria.

Lawal Tunde Dauda[2]
ICT Department of Computer Science,
The Federal Polytechnic, Offa, Nigeria

Babatunde Ronke Seyi[3]
ICT Department of Computer Science,
Kwara state University, Malete, Nigeria.

Gbolagade Kazeem Alagbe[4]
ICT Department of Computer Science,
Kwara State University, Malete, Nigeria.

ABSTRACT: *The understanding of evolutionary relationships among biological entities enabled computer scientist's researchers to developing various arithmetic number bases systems with remarkable genetic sequence alignment technique, where weaker analyses in homologous sequence are acceleratory detected and implemented. Using SWA for continuous improvement in the sensitivity of genes and proteins regulatory sequence alignment, three moduli sets are presented for DNA sequence alignment with residue of each modulus independently of each other, concurrently by parallel ALUs without carry propagation among them. Finally, we developed a reconfigurable RNS-SWA based arithmetic architecture with design support tool characterized in addition (ADD), subtraction (SUB), multiplication (MUX) using RNS conversion technique properties on a multi ALU system which provide structural VHDL simulation descriptions summary for the RNS-SWA Based DNA sequences; improving low Power, high speed linear processing acceleration required in genetic sequences computing. Our proposed design work when implemented in PLD-RNS-SWA based high profile acceleration comparator, exhibits a greater significant percentage speed when compare with the non RNS state of art profiler in real time processing design supporting a very high speed integrated Circuits*.

KEY WORDS: Bioinformatics. Residue Number System, Smith Waterman Algorithm, DNA Sequence Alignment, Programmable Logic Design, Moduli set, Hardware Accelerator Architecture,

INTRODUCTION

Residue Number Systems (RNS) allow the distribution of large dynamic range computations over small modular rings, which allow the speed up of computations. This characteristic is predominantly known, and already used in many mathematical computing application including biological computing sequences. Every living organism is made up of living cells which consist of genetic information that make it different and unique from another organism. These genetic information are carried by a chemical known as DNA in the nucleus of the cell.[14];[17]The DNA of an organism consists of an interwoven strands that forms a double helix. Each strand is built from residues of molecules called nucleotide. A nucleotide consists of two parts viz: a phosphate group and a sugar group called deoxyribose, these two parts from the ribbon like backbone of the DNA strand and are identical in all nucleotides.[13],[17]

One of the paramount parameter to be considered in the field of Bioinformatics computing is the emergence of precise accelerated high speed accurate performance computing architecture device. This become necessary as the volume of data to be processed grows continuously and demands for high speed accuracy in computing resources are becoming competitive, indispensable and should be taken into consideration. Many genetic algorithms such as FAST, BLAST, Needleman Wunsch Algorithm and SWA etc has been developed and used in recent past with defect in either accuracy or delay in computational speed process. These two factors demanding ability to rapidly sequence DNA information accurately are very essential tools in bioinformatics sequence computing, for example, the study of evolutionary trends correlating DNA information with disease where two genes were noted to be involved in the origins of breast cancer (Miki el at 1994) [13] such research is only possible through the help of high speed and accurate sequence comparison.

The choice of SWA which is the most accurate genetic algorithm for sequence alignment performs at the expense of delay computational time has been adopted in this work because of it capability to maneuver between the two ways classification models; the local and the global biological sequence alignment. In the first, the optimal value is calculated from the most similar sub-region common to both sequences while in the latter, the two sequences must be of similar length and the optimal value is computed from beginning to the end of the sequences [2]. In this phenomenon, the maximum efficiency can often be actively achieved simultaneously; where problem oriented approaches are used with demanding for the help of high speed sequence comparator. Residue Number System (RNS) is a non-weighted number system, with the absence of carries between the digits the arithmetic operations has been used to address the computation challenges of smith waterman dynamic algorithm based on Bioinformatics sequences.

This research paper work enhance acceleration of SWA performance features for the implementation of high-speed performance computing architecture algorithms for DNA sequences using a conjugal Moduli sets with special method of data representation in Residue Number System and conversion technique to achieve the goal. [2],[5].

RELATED RESEARCH WORKS

Several research attempts have been reported for the systematic design of SWA performance implementation modules and compare both hardware and software requirement performance with the complete and fractional parts of the algorithm for the gene sequences. This is based among other factors on CRT representation, decomposition of residue number into the summation of its components and the final multi-operand modulo adders.

In [Gbolagade.et al,. 2009] carry out an implementation on 0(n) Residue Number System to Mixed Radix Conversion Technique. The Technique when compare with the state of art reduces the number of arithmetic operation by 5.6% and 38.64% for moduli set of length four and ten respectively [4].

In [Hassan Kehinde Bello and Kazeem Alagbe Gbolagade. 2018] proposed studies on the improvement on acceleration of Biological Sequence Alignment using Residue Number System. The percentage speed gained was compared with the state of art in which the work is 140.63% faster than [4] and also the run-time of the hardware implementation of the research work is 40.66% indicating Timing simulation of RNS-SWA architecture critical delay of 10.38 ns better than [8].

In [Schroeder A. et al., 2014], work on implementation of the streaming architecture of the Graphics processing Units (GPUs) which can be used for biological sequence database scanning. GPUs are single-chip processors, used primarily for computing 3D functions, but is also a good candidate for bioinformatics applications such as sequence alignments [19]. To achieve an efficient mapping on this type of architecture, the authors have formulated the SWA in terms of computer graphics primitives and claimed that the evaluation of their implementation on a high-end graphics card shows a speedup of almost sixteen compared to Pentium-IV, 3.0 GHz processor. [19]

In [Laiq Hasan and Zaid Al-Ars 2007], the work divided the SWA into a number of functions, and then the time complexity of each function is measured, thus term known as code profiling. A software only implementation of the SWA is profiled on Pentium-IV, 3.2 GHz processor, using the GNU profiler. The profiling results identify the most time consuming function. This function is then designed in VHDL.[8] The processing run time of software -only implementation on Pentium-IV, 3.2 GHz processor and hardware implementation on Virtex II Pro FPGA are compared to evaluate the percentage runtime improvement. The results show that the hardware implementation is 35.82 times faster than its equivalent software-only implementation [7].

In [Steve Margem, 2006], use the power of reconfigurable computing to accelerate substantially the performance of the SWA. The percentage time spent on calculating the elements of the matrix, Hi,j, was cut down by nearly a third and the absolute time spent on the algorithm dropped from 6,461 seconds to a little over 100 seconds, approximately 64 times faster than the equivalent software-only implementation on AMD Opteron processors[17].

In [Oliver T. et al., 2005], showed a new approach to bio-sequence database scanning using reconfigurable FPGA-based hardware platforms to gain high performance at low cost.[14] Their FPGA implementation achieves a speedup of approximately 170, as compared to a Pentium-IV, 1.6GHz processor.

In [Chiang J. et al., 2006], also studied the improvement of the computational processing time of the SWA using Custom Instructions (CIs) on an FPGA board.[11] This was done by first writing the SWA in pure software and replacing the portion which was the most computationally intensive with an FPGA custom instruction. Particularly, the designed CIs were on an Altera Nios II integrated development environment. The Nios II soft microprocessor was instantiate on an FPGA to allow rapid prototyping of new designs. Finally, they compared the processing runtime between the pure software and the hardware acceleration versions to calculate the percentage of runtime improvement. The results showed that the hardware accelerated algorithm improvement the processing runtime by an average of 287%. Thus using FPGA CIs is a promising direction for further research in improving genomic sequence searching [11].

In [Yamaguchi Y. et al., 2002], proposed a high speed sequence alignment using run-time reconfigurable computing. With this approach, it is demonstrated that high performance can be achieved using off-the-shelf FPGA boards. The performance is almost comparable with dedicated hardware systems. The time for comparing a query sequence of 2048 elements with a database sequence of 64 millions elements by the SWA is about 34 seconds, which is about 330 times faster than a desktop computer with a Pentium-III, 1.0 GHz processor.[20]

METHODOLOGY

The reconfigurable RNS-SWA Based Arithmetic architecture, using appropriate RNS computer arithmetic property techniques is implemented to solving the computational area challenges associated with SWA. First, with three moduli set $2^{n-2} + 1$, 2^n, $2^{n-2} - 1$ selection and design comparator having a dynamic range of 2n bits placed on a PLD system with hardware description language VHDL design and on a Very High Speed Integrated Circuit (VHSIC).The known inherent RNS arithmetic features is acquired to speeding up the SWA computation rate in the RNS architecture net work.

Parallel conversions implementation in the RNS digital system which comprises of the functional units of RNS Processor 1, 2, 3 components; adders, subtractors, shifters, Buses for transfer of $H_{i,j}$ data between the different Processor components and multipliers**,** Registers **,** Arithmetic and Logic Units (ALUs) and comparators where MAX $H_{i,j}$ output Residues from the computation is returned and compared with non RNS the state of art profiler using CRT to accomplished the goal.

Proposed Conjugal Moduli Set Selection Criteria.

In this work we find it necessary to select balanced three conjugal moduli set with respect to the magnitude and complexity of DNA sequence computations such that any failure, resulting time delay for some threads is avoided.

We consider that the magnitude of the largest modulus which dictates the speed of the arithmetic operations enable us to use moduli set with smaller dynamic range; where matrix partitioning has to be used in this work based on the fact that the comparison of two long strings is done using a divide and conquer approach.

Efficiency of the RNS moduli set is principally considered and high efficiency is more desirable, example the RNS (15|13|11) require 12 bits it can represent 212 = 4096, whereas only 2145 numbers are presented in which the efficiency is 52%

Selection of the conjugal moduli is in this work is based on relatively (pair wise) prime with gcd (mi, mj) = 1 for all mi ≠ mj compliance which simplify that binary to RNS and RNS to binary conversions as implemented are simple RNS arithmetic.

The conjugal Moduli numbers set is restricted to power of 2 with optimum large dynamic range to avoid overflow. Noting that the smaller the moduli, the faster the arithmetic operations and also the higher the dynamic range of the conjugal moduli set, the faster it's forward conversion and the slower its reverse conversion.

Each moduli *mi* is kept as small as possible such that operations modulo mi which require minimum computational time has a well balanced decomposition of the dynamic range. This means that the difference in word length between these moduli is as small as possible.

The RNS Modular Arithmetic Computations.

RNS is based on modular non-positional representation of numbers. Each digit in RNS is a residue of the division by a number called a base. All bases for each digit are form a moduli set. The uniqueness of the representation of numbers in RNS is only guaranteed under the condition that all moduli are pair wise co prime. Let $m_{1,}m_{2,}m_{3\ldots}m_n$ be a given moduli set.

It determines a unique RNS. A number X in this RNS can be represented as follows: X = $(x_{1,}x_{2.}x_{3\ldots..}x_n)$, where $x_i = x \bmod m_i$ for all i = 1,2,3,…n and in according to Chinese Remainder Theorem X should belong to the interval [O,M], where $M = m_{1,}m_{2,}m_{3\ldots}m_n$ is dynamic range of the RNS.

With such a representation, addition and multiplication can be done in parallel. Let X = $(x_{1,}x_{2,}x_{3,}$ $x_n)$ and Y = $(y_{1,}y_{2,}y_{3,}\ldots.y_n)$ be numbers in RNS then their sum S = X+Y = $(s_{1,}s_{2,}s_{3,}\ldots s_n)$ and product Q = X-Y = $(q_{1,}q_{2,}q_{3,}\ldots.q_n)$ can be computed using following formulas $s_i = (x_i + y_i) \bmod m_i$ and q_i = $(x_i - y_i) \bmod m_i$, for all I = 1,2,3,..n. These operations do not require carries between digits as in positional number systems.

This property allows performing these operations independently. However, computations in RNS require a number of specific operations, without which it is impossible to represent numbers in RNS [3],

Parallel RNS Arithmetic Operations.

The main advantage of RNS in the context of high performance computing is the opportunity to do some operations in parallel; Addition and multiplication have asymptotical complexity O(N) and $O(N^2)$ respectively. In the best case multiplication has complexity $O(N\log_2 N \log_2 \log_2 N)$ **[6].** Using this property and combining addition and multiplication, we can achieve a dramatic increase in speed of computations, which are done in parallel in reverse conversion chamber of RNS comparator architecture.

For any n - bit nonnegative integer X in the range $0 \le X \le 2^{n-2} - 1$ can be represented in the weighted binary system as

$$X = \sum_{i=0}^{N-1} b_i 2^i \quad \text{Where } b \in [0, 1] \qquad [1]$$

The binary value of X can be converted into a set of n residues as x, where xi = X mod mi. The values of xi can be found by the following steps:

$$X = \sum_{i=0}^{N-1} b_i 2^i \qquad [2]$$

Let X mod mi = $\left| X \right|_{mi}$ Then

$$\left| X \right|_{mi} = \left| \sum_{i=0}^{N-1} bi \; | \; 2^i | \right|_{mi} \quad \text{the term} \left| 2^i \right|_{mi}$$

Also for any n - bit signed integer X in the range $0 \le X \le 2^n - 1$, the residues of X can be represented in the 2^1s complement form as;

$$x_{2^i} = (bn\ bn_{-1} \ldots\ldots b1\ b0) = - bn\ 2^n + \sum_{i=0}^{n-1} b_i 2^i \qquad [3]$$

Let xi = X mod mi

$$\text{Then } xi = \left| b_n (mi - \left| 2^n \right|_{mi}) + \sum_{i=0}^{n-1} \left| b_i \right|_m 2^i \right|_{mi} \qquad [4]$$

The value of $\left| 2^i \right|_{mi}$ can be pre computed from x_i

Based on the length of the moduli set. The value of X parameter is further enhanced form equation [1] above. This is achieved by substitute in eq [5] with

$$X = x_1 + m_i \quad k1m1 + k2m1m2 + k3\ m1m2m3 + kn\ m1m2\ldots\ldots mn\text{-}1 \quad m2m3\ mn\text{-}1 \qquad [5]$$

$$X = x1 + m1 \left| k1(x2\text{-}x1) + k2m2\ (x3\text{-}x2) + \ldots..\ kn\ mn\ (\ xn + 1 - xn\ \text{-}1) \right| m2m3\ mn\text{-}1 \qquad [6]$$

Such that $\left|k1m1\right| m2m3 = 1$ where $k_1 = \left|m_1\text{-}1\right| m2m3$ and $\left| kn\ m1\ m2\ \ldots..mn - 1 \right| m_n = 1$

$$-\frac{M\text{-}1}{2} \leq X \leq \frac{M\text{-}1}{2} \quad \text{if M is odd} \qquad [7]$$

$$-\frac{M}{2} \leq X \leq \frac{M}{2} \_ 1 \quad \text{if M is even} \qquad [8]$$

The conversion of X from Eq 4 representation to RNS can be done in K iterations. But if we consider a full parallelization, with n modular multiplier-adders MMA, this conversion can tend to be logarithmic where O (log K) such that (MMA) forms the basic operator of RNS computing. If $m_1, m_2 \ldots m_n$ are set of moduli, then the dynamic range (M) is the product of all the moduli set.

It is noted that when $M = (m_1, m_2, \ldots, m_n)$ and $M^i = (m^i_1, m^i_2, \ldots, m^i\ n)$ form the two RNS bases such that we use from Eq [4] the proof of the Chinese Remainder Theorem and Lemma can be shown, such that a solution of the following System: $X = x_1 \pmod{m_1}$, $x_2 \pmod{m_2}$, $x_3 \pmod{m_3}$ … $x_n \pmod{m_n}$ can be drawn from Eq [4], Eq [5], Eq [6] [7] [8] then every representable number (X) satisfy either of the Eq [4,5,6]

The Computations Technique and Complexity in SWA

The Smith Waterman algorithm SWA which finds the optimal local alignment between two sequences has the total time complexity of the SWA is O(M + N) + O(MN) + O(MN) = O(MN).

The total footprint of the SWA is also O(MN), as it fills a single matrix size MN. In order to reduce the O(MN) complexity of the matrix fill stage, multiple entries of the H(i, j) are calculated in parallel [8; 4];[13] which makes it compares segments of all possible lengths and optimizes the similarity measure.

However, SWA is fairly demanding of time and memory resources; in order to align two sequences of lengths **m** and **n**, O (kmn) time and space are required. As a result, it has largely been replaced in practical use by the RNS algorithm; which guaranteed to find optimum hardware acceleration for SWA alignments. When obtaining the local alignment, a matrix Hi;j is used to keep track of the degree of similarity between the two sequences to be aligned (A_i and B_j) [4][13]**.**

Table1 illustrates the $H_{i,j}$ algorithm process with each element of the matrix $H_{i,j}$ is calculated in accordance with equation **7.**

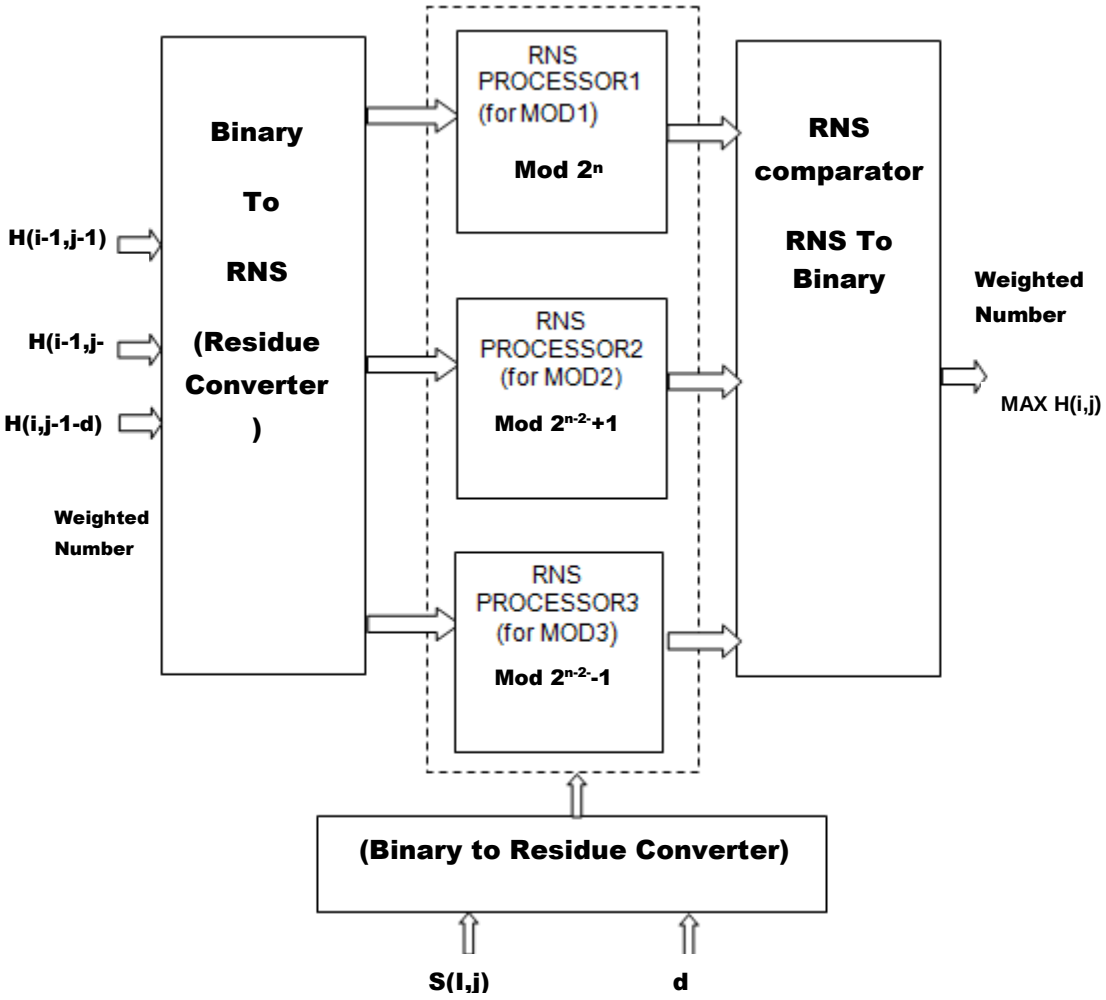


Fig.1. – Proposed Reconfigurable Schematic RNS-SWA Based Architecture Accelerator for DNA Sequences computation.

THE SMITH-WATERMAN ALGORITHM IMPLEMENTATION.

The SW algorithm finds the optimal local alignment between two sequences, such that a matrix H**i;j** is used to keep track of the degree of similarity between the two sequences to be aligned (A_i and B_j) [8][4].[13] Each element of the matrix $H_{i,j}$ is calculated according to the following equation:

$$H_{i,j} = \max \begin{cases} 0 \\ Hi-1, j-1 + Si,j \quad \text{Diagonal entry} \\ Hi-1, j-d \quad \text{Upper entry} \quad \ldots\ldots\ldots \\ Hi, j-1-d \quad \text{Left entry} \end{cases} \qquad [9]$$

Where**:**

H is the matrix value of the essential cell with H (i; j) is the maximum similarity score between the two sequences. S is the score of the cell $S_{i;j}$ which is the similarity score of comparing sequence A_i to sequence B_j and **d** is the gap alignment and penalty for a mismatch. i, j describe row and column Diagonal, Upper and Left entries are the matrices entry position relative to the current $H_{(i;j)}$ calculation.

.

The instruction selection set of SW algorithm	**SWA Pseudo Code implementation**	**Software Matrix implementation for DNA Sequences.**	**Remark**
1. Initialization of matrix considering the two sequences A and B. **2**. Matrix filling with the suitable scores. The two sequences are set in a matrix form by means of **A+ 1** column and **B+1** row with the values in the first row and first column set to zero. **3.** The Trace back Matrix begin from the position having the highest value, pointing back, consequently find out the possible predecessor, then go to next predecessor and continue until it reach the score **0** the optimal alignment of smith program.	**1** Declare an n x m similarity matrix; **2** Initialize the top row (i = 0) and left column (j = 0) with 0; ⎡→ **3** for i = 1; i < length (Sequence); i++ do ⎢ ⎢ ⎡→**4** for j = 1; j < length (Sequence); j++ do ⎢ ⎢→ **5** H(i,j) = max{0;H(i-1; j-1)+S(i;j);H(i-1; j)-d; H(i; j-1)- d}; ⎢ ⎢ ⎢ ⎣→**6** end ⎣→ **7** end **8** → Save index of term that contributed to the calculated value in H(i,j); **9** → Find maximum value in **n x m** matrix; **10** → Using saved indices in **8**, trace back to find **0** encountered;	In calculating the local alignment, matrix H(i; j) is used to keep track of the degree of similarity between the two sequences Ai and Bj to be aligned. Each element of the matrix H(i; j) is calculated in equation:5. SWA finds the optimal local alignment between two sequences. local alignment longest common subsequence (LCS) result for the two DNA sequences, A and B is obtained with the resulting alignment as: **G, C, C, A, G**	The resulting LCS alignment is obtained as: **G, C, C, A, G**

Table1 illustrates the $H_{i,j}$ algorithm process with each element of the matrix $H_{i,j}$ calculated in accordance with equation 9.

M*N Table Matrix implementation of the Smith Waterman Algorithm Based Dynamic Program.

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

Tab1M*N Matrix Initialization: $S_{i,j.}$ For i = j = 0

		G	C	C	C	T	A	G	C	G
	0	0	0	0	0	0	0	0	0	0
G	0	1	1	1	1	1	1	1	1	1
C	0	1	2	2	2	2	2	2	2	2
G	0	1	2	2	2	2	2	3	3	3
C	0	1	2	3	3	3	3	3	4	4
A	0	1	2	3	3	3	4	4	4	*
A	0	1	2	3	3	3	4	4	4	4
T	0	1	2	3	3	4	4	4	4	4
G	0	1	2	3	*	4	4	5	5	5

		Ⓖ	Ⓒ	C	Ⓒ	T	Ⓐ	G	C	Ⓖ
	0	0	0	0	0	0	0	0	0	0
Ⓖ	0	1	1	1	1	1	1	1	1	1
Ⓒ	0	1	2	2	2	2	2	2	2	2
G	0	1	2	2	2	2	2	3	3	3
Ⓒ	0	1	2	3	3	3	3	3	4	4
Ⓐ	0	1	2	3	3	3	4	4	4	4
A	0	1	2	3	3	3	4	4	4	4
T	0	1	2	3	3	4	4	4	4	4
Ⓖ	0	1	2	3	3	4	4	5	5	5

M*N Table 3: The SWA Base DP Matrix and the Trace Back Path

M*N Table 2: Matrix Fill in:
$S_{i,j.}$ = + i for i → (-n ≤ i ≤ n)

Review of Matrix sequence alignment statistics.

The M*N Table Matrix above illustrates the Pairwise sequence alignment algorithms which assign a score to the alignment of each pair of sequences. A larger score implies a closer biological relationship. Iterative sequence alignment tool, SWA builds on these pairwise sequence alignment algorithms. In each iteration the pairwise sequence alignment algorithm is used to search a large sequence in database leading to a list of hits ordered by their scores. From the high scoring alignments, a multiple alignment is created. This determines the scoring system of the next iteration. The crucial step between iterations is deciding which of the hits to keep or rejected as

irrelevant putative members of the family. A reliable quantitative criterion for this decision is reflected in Eq. 10 with the optimal local alignment of two sequences:

A = G C C C T A G C G : B = G C G C A A T G Such that:

$$S_{i,j}=\begin{cases}+2 \text{ if } (A_i = B_j) \\ -1 \text{ else}\end{cases} \quad \text{Where } d = 2 \qquad \textbf{[10]}$$

The matrix $H_{i,j}$ and the trace back path shown in red bold digits. The best score found in the matrix is **5** and the corresponding optimal local alignment is **A: G C C C T A G C G** and **B: G C G C A A T — G**

The SWA Comparator Process with DNA Sequence

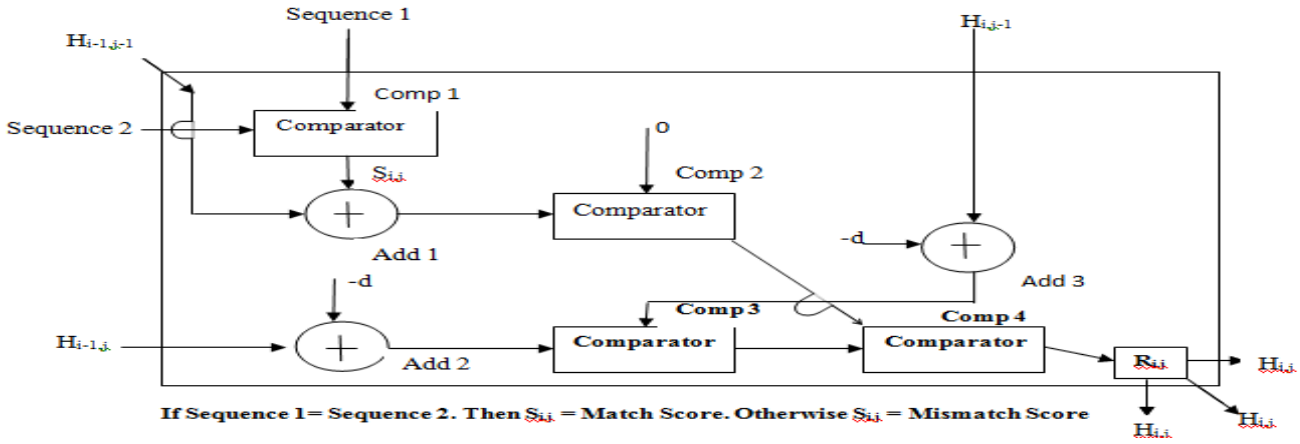


Fig.2. Block diagram description of a basic cell for computing Hi;j values of Eq.1.

Computation of SWA Using Traditional Acceleration Technique.

The figure2 above illustrates the block diagram of a basic cell for computing elements of the Hi;j matrix in accordance with traditional acceleration approach. The initial comparator (Comp1) compares the two input sequences yielding the corresponding outputs value of Si;j, base on the values of the match and mismatch scores, such that Si;j equates match score, if the corresponding characters of Sequence1 and Sequence2 are equal, otherwise Si;j equates mismatch score. Add1 is an adder that adds the diagonal element Hi¡1; j¡,1 and the value of Si;j **.**

The second comparator (Comp2) is a comparator that compares the output of the Add1 with a constant value 0 and outputs the greater of the two numbers. Add2 is an adder that adds the left element Hi¡1;j and -d, where d is the gap penalty. Add3 is an adder that adds the upper element Hi;j¡1 and -d. Comp3 compares the outputs of Add2 and Add3 and outputs the greater of the two

numbers. Comp4 compares the outputs of Comp2 and Comp3 and results the greater of the two numbers.

The output of Comp4 is the corresponding Hi;j value, which is stored in register Ri;j. The matrix is initialized with the value zero. The gap penalty is assumed to have a value zero and a simple scoring scheme is assumed, such that Si;j = 2, if there is a match otherwise Si;j = 0.

The next paragraph is accompanied by the architectural organization of the acceleration logic of the RNA SWA based implementation made up of three major building blocks:

ACCELERATION IMPLEMENTATION OF RNS-SWA BASED HARDWARE ARCHITECTURE.

Methods for Hardware Acceleration Implementation of the Matrix Fill Step are executed in three parallel forms.

Forward converter

The forward converter implies the Binary to RNS Conversion decompose the binary value into an array of power of two values, and sum them up with modular adders. The operands input which are indispensable to the RNS processor are either in the decimal or binary format, and must be converted into their respective residues before they are used for the computation. In this conversion unit, the memory less conversion process of the SWA inputs: H (i-1; j -1), S (i; j), H (i-1; j), H (i; j-1) and d, as in Eq [7] is executed by the Binary to RNS Converter into their residue equivalents.

Using restricted conjugal moduli set m = 2^{n-2}-1, 2^n, 2^{n-2}+1 where n = 4, **M =** 240 with 2n Bits dynamic range having sub matrices element restricted value within the interval of -120, +119. The unique decimal numbers in the RNS system is principal dictator and determinant and is based on the sizes, the numbers and the length of sub matrix string such that the residue generation values is greatly applied and simplified in the combinational logic.

The proposed conjugal moduli set is fed into the VHDL software application of Quartus II version 4.0 using schematic embedded software capture tool. The screen short schematic of the RNS forward converter is represented using the design tool shown in Figure 3.

Implementation is carried out by partitioning the binary number into blocks and then concurrently carrying out modular exponentiation on all the partitions. A 16 bit residue is implemented with the speed of the residue computation further accelerated using 4 times as many multiplexer enabling modular exponentiation to be performed in a clock cycle.

The dynamic range M of the decimal number for the conjugal moduli set is an 8 bit binary number partitioned into two 4 bits as x and y with second nibble added to the in two's complement X→ X_0, X_1, X_2, X_3 called the high order bits of the binary representation and Y→ Y_0, Y_1, Y_2, Y_3, as low order 2n bits. Both X and Y bits are added by a parallel adder (PARDD).

The second stage of addition emanates from X_8 and form operand specifying the sum from $PARDD^1$, identity called P1, P2, P3, P4,P5 P6, P7,P8. Execution between $PARD^1$ and $PARD^{11}$ is the combinatorial process done by the carry-out (C_{out}) and the sum from $PARD^1$ logic. $|X|2^n \rightarrow M \bmod 16$ represents the sum without the (C_{out}) ; $|X|2^{n-2}+1 \rightarrow M \bmod 5$; $|X|2^{n-2}-1 \rightarrow M \bmod 3$ is the 4 bit Y. The residues produced are then used to execute carry free addition, borrow free subtraction by the three RNS processors as shown in Figure 1. Each of the residue processors does concurrent data processing, independent of each other, and thereby speeding up the arithmetic operation involves in the calculation of the SWA

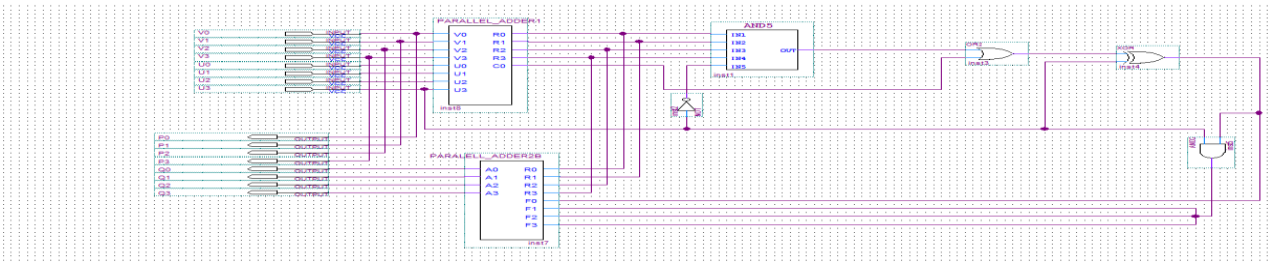


Figure 3: Schematic diagram of the RNS Forward Converter

The RNS-SWA Based Processor

The RNS-SWA based arithmetic operations is the microprocessor stage. The design logic consists of two multiplexers (MUXs), each consisting of eight inputs and four outputs, two modulus 16 parallel adders, one modulus 5 and 3 both parallel adder and a control unit that controls the data selections in the two MUXs .

The logic in the control unit controls the sequencing of these additions by exploiting the inherent potential properties of RNS to do carry-free Arithmetic without partial product. These binary/decimal values are converted into residues numbers by the Binary to RNS Converter (BRC), called **(**RNS forward Conversion) [14],[11],[17].

The residues produced using two or three sets of four bit-sliced 3 or 2-to-1 multiplexers, two modulus 16 parallel adders, one either modulus 5 or 3 or both. The parallel adder and a control unit are then used to execute carry free addition, borrow free subtraction by the three RNS processors [Fig1] in accordance with Equation 5.The Sequential process of these arithmetic logic operations is tabulated as follow:

S/N	Sequential Process	Logical Component of Arithmetic Operation Process	Components
Step1	Diagonal Addition	H(i-1,j1+S(i,j)	H(i-1,j-1; S(i, j),
Step2	Upper Addition	H(i-1,j)+ (- d)	H (i-1,j);(-d).
Step3	Left Addition	H(i,j-1) + (- d)	H(i, j-1);(-d)

Table2: Sequential process of arithmetic logic operations.

The Component H(i-1,j1 is added to S(i,j) as diagonal addition to produce H(i-1,j-1;S(i, j),also is component H(i-1,j) is added to (- d) as upper addition producing H(i-1,j);(-d).lastly component H(i,j-1) is added to (- d) as left addition yielding H(i, j-1);(-d) Each of the residue processors does concurrent data processing, independent of each other, and thereby speeding up the arithmetic operation involves in the SWA calculation.Fig.2.

RNS-SWA Based Reverse Comparator Implementation.

The RNS magnitude comparison stage the RNS-SWA reverse comparator performs reverse conversion of the residue results of the arithmetic operation by the RNS processor to twos complement (M) binary representation and compares them with zero and with each other. The decimal values corresponding to the four values H(i-1, j-1) + S(i, j), H(i -1, j) –d, H(i, j-1) –d and 0 being compared [Fig.2] are read into two different registers in various clock cycles and then compared by a binary comparator.

The maximum value for the matrix score assignment is output to H (i,j) as shown in Fig.1 which ends the comparisons process. Two sequential processes (Diagonal versus Left) Addition are compared, yielding to the maximum summation of the three values. These three stages are implemented on a PLD system employing the inherent RNS arithmetic properties.

Parameter Combinational Resource Entities	M RNS Forward Converter	RNS-SWA Processors	Final- RNS-SWA Reverse Converter
Flow Status		Successful – Mon Jul	27. 21:15:52. 2020
Revision Name	MODULUS 16-5-3	RNS-Processor	Final-SWA Processor
Top-Level Entity Name	MODULUS 16-5-3	RNS-Processor	Final-SWA Processor
Family	MAX 7000 B	Stratix II	Stratix II
Total Combination Function		27	143
Device	EPM 7032 BT 144- 5	EP2S16F484C3	EP2S16F484C3
Total I/O PINs	20/36 [55%]	26/343 [7%]	32/343 [9%]
Total Registers / Used	0/32 [0%]	10	46
Total Processors / Used	1 Processor [100%]	2 Processor [20%]	4 Processor [0%]
No Detected on Machine		4[100%]	
Maximum Used / Allowed		3[75%]	
Total memory bits		0/419,328 [0%]	0 / 920,448 [0%]
Total LABs			27 / 1,057 [2 %]
Total PLLs		0/6 [0%]	0/6 [0%]
Total DLLs		0/2 [0%]	0/2 [0%]
Total ALUTs		34/12,480 [1%]	193/12,480 [1%]
DSP block 9-bit element		0/96 [0%]	0/96 [0%]
Parallel expanders	9/30 [30%]		
Sharable expanders	12/32 [37%]		
Number of p-terms used	89		
Maximum Fan-out node	V0		
Maximum Fan-out	18		
Logic cells	20/32 [62%]		187 / 10,570 [1 %]
Dedicated Input/clock Pins	0/2 [0%]		
Clock pins			5 /16 [31%]
Global clocks			3 /16 [18%]
Regional clocks			0 / 16 (0 %)
Global signal	0		3
DIFFIOCLKs			0 / 16 (0 %)
SERDES transmitters			0 / 44 (0 %)
Maximum fan-out			44
Total fan-out			741
Average fan-out used	3.44	1.33	3.50

Table3: Summary of Simulation and Parallel compilation report of the Circuit Resources Utilization for RNS-SWA Based Architecture.

The performance of the proposed accelerator was evaluated in terms of speed and hardware cost. Table 3 shows the summary report of the parallel and final compilation and the performance

evaluation of RNS accelerator implementation where 2 out of 12,480 total logic elements within the device are used with a negligible number of the logic cell 193/12,480 (1%) within the device are also used when implemented on EP2S16F484C3 device (Cyclone 11).

From fig.3 below it is seen that the simulation tool displays B0 and B1 in respect of 0 and 1 and also indicate at initial start of simulation 20.0 ns, with V = v_3 v_2 v_1 v_0 = 0010 and U = u_3 u_2 u_1 u_0 = 0010, which signifies in twos complement the representation of the number is 0010 0010 .3410. at the said time P = V= 34 MOD 16 = 2 = p_3 p_2 p_1 p_0 = 0010 and Q = 34 MOD 5 and 4 = q_3 q_2 q_1 q_0 = 0100.

In the implementation, our n = 4 gives m = {16, 5, 3} with a dynamic range (M) of 240. Since the M is an even number, Eq [5] is applied, meaning the range of numbers that can be represented by this scheme is given to be - 120 ≤ X ≤ 119, which makes, the values representable by this dynamic range for signed number as shown in Table 4. On applying the scoring parameters {gap, match or mismatch} and RNS to equation [7], the range of our values will fall within Table 4.

Decimal Number	**-120**	**-119**	**-118**	**..**	**-3**	**-2**	**-1**	**..**	**0**	**1**	**2**	**3**	**..**	**117**	**118**	**119**
Hexadecimal No.	**88**	**89**	**8A**	**..**	**FD**	**FE**	**FF**	**..**	**00**	**01**	**02**	**03**	**..**	**75**	**76**	**77**
Mod [16]	**8**	**9**	**A**	**..**	**D**	**E**	**F**	**..**	**0**	**0**	**2**	**3**	**..**	**5**	**6**	**7**
Mod [5]	**3**	**4**	**1**	**..**	**3**	**4**	**0**	**..**	**0**	**1**	**2**	**3**	**..**	**0**	**1**	**2**
Mod [3]	**1**	**2**	**0**	**..**	**1**	**2**	**2**	**..**	**0**	**1**	**2**	**0**	**..**	**2**	**0**	**1**
Conjugal Mod [16, 5 and 3]	**8,3.1**	**9,4,2**	**A,1,0**	**..**	**D,3,1**	**E,4,2**	**F,0,2**	**..**	**0,0,0**	**0,1,0**	**2,3,2**	**3,3,0**	**..**	**5,0,2**	**6,1,0**	**7,2,1**

Table 4: The Residues Table for Conjugal Mod 16, 5 and 3 for signed numbers in hexadecimals.

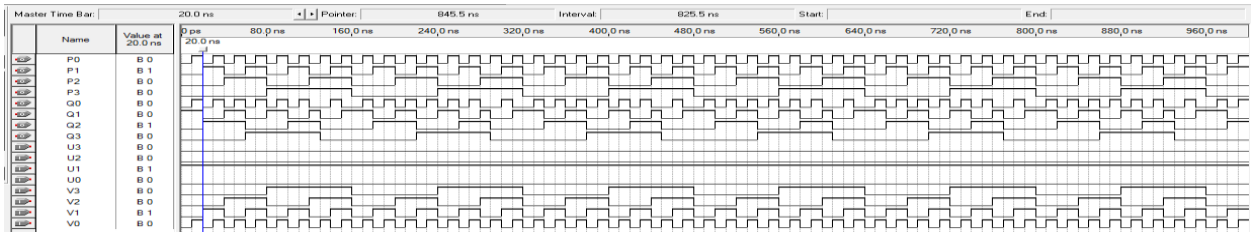


Figure: 3 Simulation snapshots view of the RNS-SWA Based Forward Converter

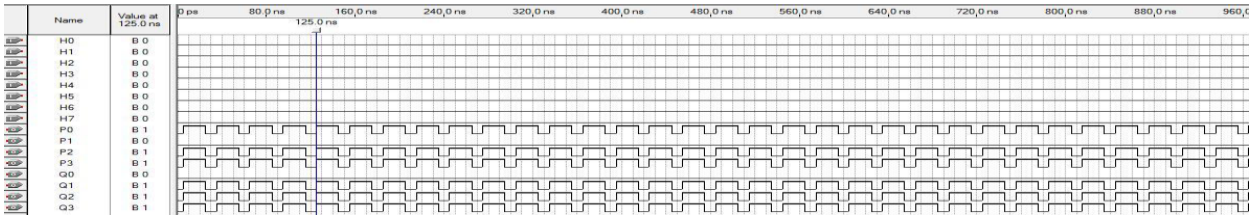


Figure 4. Simulation results of the RNS-SWA Based Processor

PERFORMANCE EVALUATION OF THE ACCELERATOR IMPLEMENTATION.

The % runtime improvement ratio of the RNS-SWA based implementation as compared to the state of art [9] is computed as

The code was run using (6.4 GHz) processor, with the time period of the clock is

$\frac{1}{6.4\text{GHz}}$ = 0.15625ns. [Proposed]

No of Clock cycle = Clock Ticks * 64
The actual times consumed by fill matrix functions.

$= \frac{5.23}{100}$ ms

$= 0.05232\ ms \rightarrow 52.32\ \mu s$

Total Simulation delay = 0.0146 µs

The % runtime improvement is calculated using the equation below:

$$= \left(\frac{\frac{1}{\text{Hardware time}} \quad \frac{1}{\text{Fill matrx-2 time}}}{\frac{1}{\text{Fill matrx-2 time}}} \right) * 100\ \% \qquad [11]$$

$$= \left(\frac{\frac{1}{\text{Hardware time A}} - \frac{1}{\text{Hardware time B or Soft ware-Runtime}}}{\frac{1}{\text{Hardware time B or Soft ware-Runtime}}} \right) * 100\ \% \qquad [12]$$

$$= \left(\frac{\frac{1}{(12.012)*10^{-9}} - \frac{1}{(14.6)*10^{-9}}}{\frac{1}{(14.6)*10^{-9}}} \right) *100\ \% \quad = 21.56536\%$$

Clock speed = 185.53 MHz the runtime improvement design is achieved by substituting the software runtime value and the propagation delay of the proposed accelerator from [Eq.11] into [Eq.12]. Thus:

Percentage Runtime Ratio: = $(14.6 \times 10^{-9}) / (12.012 \times 10^{-9})* 100$
$=121.5451$

Hardware Runtime Improvement
= 82.272%

The performance of the proposed accelerator improvement was evaluated in terms of speed and hardware cost. The timing simulation of the proposed accelerator shows:
The total delay = 12.012ns

Function	Related Scheme [9][14]	Proposed Scheme	(%) Ratio Deference
The time period of the clock [MHz]	0.3120	0.15625	49.9190
No of Clock cycle	16769280	19210426	12.7000
Total times consumed by Fill matrix functions µs]	52.3200	65.5300	20.1587
Total Simulation delay **[ns]**	14.6010	12.0120	21.5534
Hardware Runtime Improvement[%]	**72.3300**	**82.2720**	**12.0840**

Table 5: Performance Evaluation Accelerator Implementation.

The percentage runtime improvement of the hardware accelerator implementation of the fill matrix is achieved relatively as shown in Eq. 12 above.

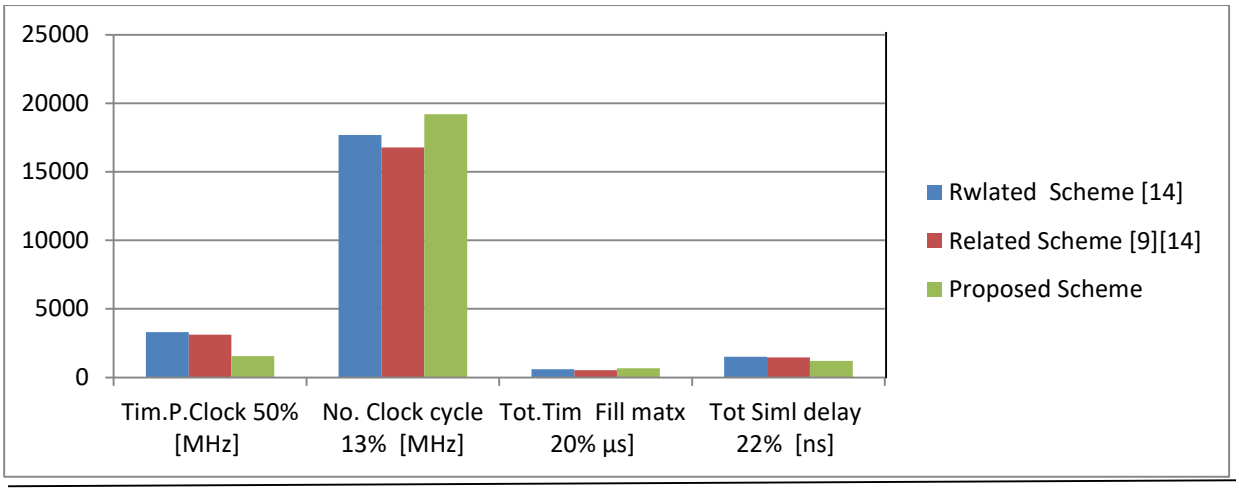


Fig.5. Graphical implementation report for performance evaluation accelerator.

Based on the implementation of the conjugal moduli set $2^{n-2} + 1$, 2^n, $2^{n-2} - 1$ and the runtime improvement of the hardware accelerator implementation. It is seen in Fig.5 that total simulation delay in our proposed scheme exhibits an approximate delay difference of 22% **ns** significance compare with [9][14]. Also the time period of the clock and it running time cycle, couple with total time of matrix fill exhibits corresponding 50% MHz, and 13% and 20% µs respectively, this is an indication of efficient design procedure expected of RNS arithmetic algorithm for improving acceleration of SWA based gene sequences.

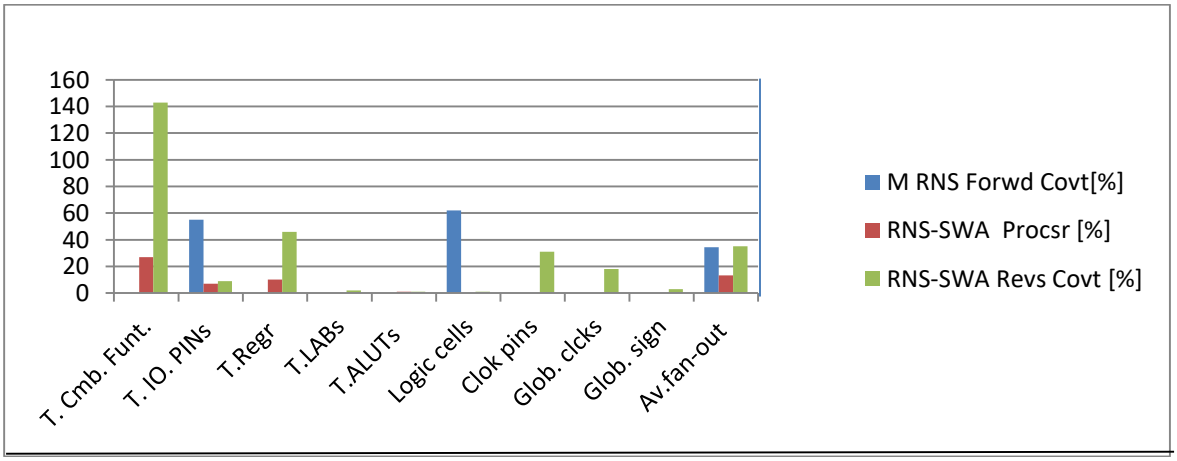


Fig. 6. Graphical implementation Report of Circuit Resources Utilization for the Proposed Architecture.

The result presented shows the circuit consumption levels of the various bar chart classifiers used for the conjugal moduli set simulation experiments. From the result presented in the figure 6 above

and considering all the very top ten level bar chart simulation experiment, indicate that one or two or both out of the three simulation classifications dataset used in this work from the decision bar chart classifier algorithms shows a negligible lowest mean utilization of less 2% consumptions of the hardware resources.

Testing Result and Time Consumed in SWA.

Number of cells	1*SCM (ms)	Run/cell (ms)	Number of cells	1*SCM (ms)	Run/cell (ms)	Number of cells	1*SCM (ms)	Run/cell (ms)
2*2	0.119	0.03	24*24	13.992	0.024	46*46	51.319	0.024
4*4	0.395	0.025	26*26	16.389	0.024	48*48	55.842	0.024
6*6	0.909	0.025	28*28	17.974	0.023	50*50	60.599	0.024
8*8	1.607	0.025	30*30	21.845	0.024	52*52	65.488	0.024
10*10	2.487	0.025	32*32	23.457	0.023	56*54	70.607	0.024
12*12	3.550	0.025	34*34	28.031	0.024	56*56	75.974	0.024
14*14	4.795	0.024	36*36	31.401	0.024	58*58	81.537	0.024
16*16	6.224	0.024	38*38	35.004	0.024	60*60	87.268	0.024
18*18	7.888	0.024	40*40	36.625	0.023	62*62	93.123	0.024
20*20	9.738	0.024	42*42	42.780	0.024	64*64	99.210	0.024
22*22	11.772	0.024	44*44	46.923	0.024			

Table 6: The SWA performance in custom instruction through Cyclone IV board.

The table above shows the identical length that is being tested in this work and is at ranges 1 to 64 base pair shows the performance of custom instruction through Cyclone 4 board. The time taken for each cell is average with 0.024 ms/cell. Increasing for the full run time is reduced by 3.319 to 1.065 with average runtime from 0.02 to 0.03

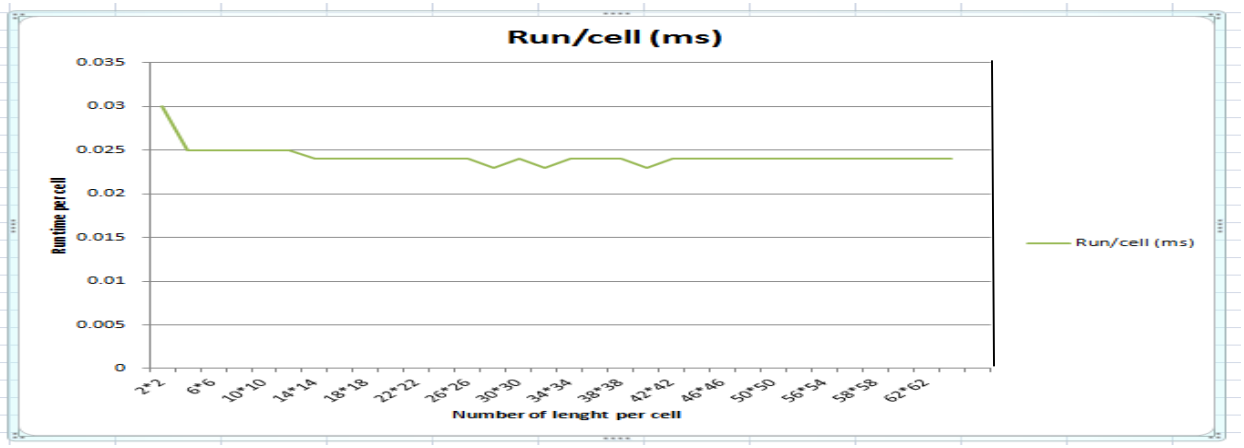


Figure 7: SWA Operating Characteristics Curve Performance in Cyclone IV board.

The justification and demand for the enhanced implementation of the RNS-SWA accelerator arises. Fig.7 above illustrates the weakness of SWA, when the length per cell increases, the run time per cell fluctuate which later maintain constant.

Comparison Scheme for Improving Bit Efficiency for the Three Moduli Set

The bit efficiency of our proposed scheme in compares is better than the existing related scheme of O (n) linear complexity.

Proof:

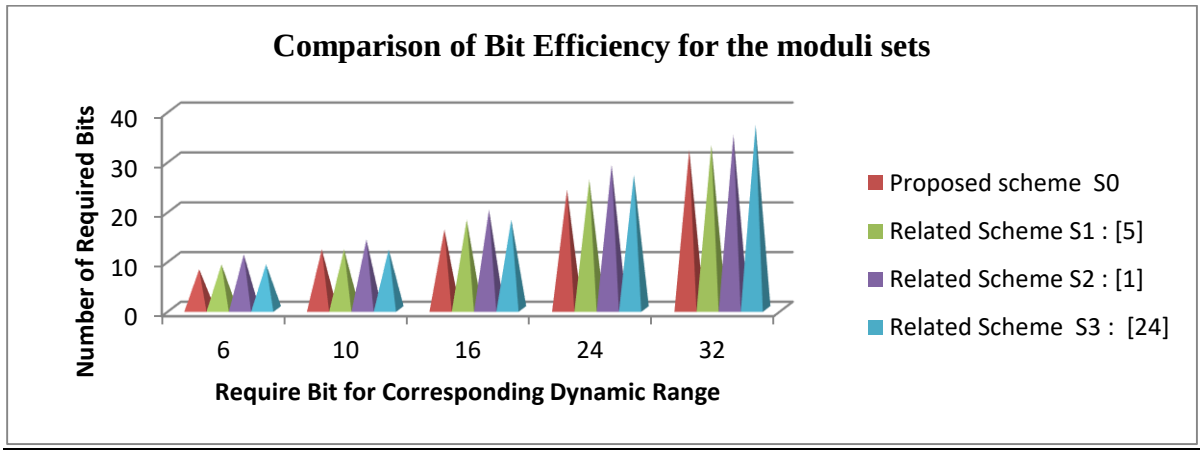
Given the result for three moduli set, a three moduli set $\{m_1 , m_2 , m_3\}$ and three moduli set $\{m_1^1, m_2^1 , m_3^1\}$ such that the Bit count of $m_1 , m_2 , m_3 \rightarrow [\log (m_1)] + [\log (m_2)] + [\log (m_3)]$ is better than that of $\{m_1^1, m_2^1 , m_3^1\} \rightarrow [\log(m_1^1)] + [\log(m_2^1)] + [\log(m_3^1)]$

If $m_1 + m_2 + m_3 \geq (\{m_1^1 + m_2^1 + m_3^1\}$ and considering the tree types of three-moduli set, in table 7 i.e. our proposed scheme $(2^{n-2}- 1,2^n,2^{n-2} + 1)^1$ one (2n, 2n+1,2n-1) related scheme and another one related scheme 2^n, $2^n+1,2^n-1$ and lastly related scheme $(2^n, 2^n-1,2^{n-1}-1$.with $s_0 + s_1 + s_2 + s_3$ respectively: $S_0^1 \leq s_1, s_0^1 \leq s_2, s_0^1 \leq s_3$ the equality can only exist when 2n can be represented as 2^n , less than otherwise

	Proposed Scheme		Related Scheme **[5]**		Related Scheme **[1]**		Related Scheme **[24]**	
N	3-Moduli Set S_0 **$(2^{n-2}- 1,2^n,2^n+1)$**	**No of Bits**	3-Moduli Set S_1 **: (2n, 2n+1,2n-1)**	**No of Bits**	3-Moduli Set S_2 **: $(2^n, 2^n+1,2^n-1)$**	**No of Bits**	3-Moduli Set S_3 **: $(2^n, 2^n-1,2^{n-1}-1)$**	**No of Bits**
6	2,7,3	8	6,7,5	9	8,9,7	11	8,7,3	9
10	3,16,5	12	12.13.11	12	17,17,15	14	16,15,7	12
16	7,32,9	16	42,43,41	18	64,65,63	20	64,63,31	18
24	15,64,17	24	256,257,255	26	512,513,511	29	512,511,255	27
32	51,128,33	32	1626,1627,1625	33	2048,2049,2047	35	4096,4097,2047	37

Table 7: Comparison of Bit efficiency for three moduli set of proposed scheme with existing related scheme.

In Table 7 comparison were performed for three-moduli set with different schemes indicated as: S_1 : [5], S_2 : [1], S_3 : [24] such that all these moduli exhibits co-prime numbers.



The results has illustrated graphically in table 7 Figure 8 respectively as compared it is observed that bits required in our proposed scheme is minimum than that of other schemes of order O(n) hence our proposed algorithm generates the most efficient moduli set than all other related schemes [1],[5],[24] given in the Literature. This is an indication that our selected working moduli set parameters is able to optimized the time complexity and the corresponding bit efficiency to the dynamic range for the Reconfigurable accelerated RNS SWA based processor

Steps	Process	Implementations	Remark
1	Moduli Set Selection	The first stage implementation was done by carefully considering the magnitude and the bits efficiency of the conjugal modulus with respect to the dynamic range which dictates the speed and architecture resources optimization of the RNS arithmetic operations.	Magnitude of moduli sets is highly considered
2	The Design Entry	RNS forward converter is entered into a Quartus II version 4.0 VHDL application software using the schematic embedded capture tool where selected pre-stored library logic functions are interconnected to enabling the logic design.	Both logic design and functions are integrated
3	Implementation	The implementation process is device dependent fitting resulting in a bit output.	Logics are mapped into

			specific device
4	Functional Simulation	This done by the software to confirm the expected functionalities of the logic circuit which also to verify that the correct outputs is produced for a specified set of inputs.	Verifies the correctness of I/O logic circuit
5	Timing Simulation	This is employed to ensure that the circuit works at the design frequency without either timing problems or propagation Delays that may affect the overall operational implementations of the circuit and hardware device.	Post design work put into the specific Device

Table 8: Summary table for the Implementations Process

CONCLUSION

The act of performing a Smith Waterman algorithm search task for biological gene pairs is both time consuming and computer resources intensive. In this work, we design a RNS SWA processor which can be reconfigured dynamically to compute some pre determined DNA functions where the unit operations need was analyzed and sequenced in terms of the inputs and arithmetic operations. The local alignment reduces the running time and increases accuracy of the sequence matching within two sequences. The improvement of parallel sequencing based RNS reduces greatly the system times, area and memory utilization. A reconfigurable RNS processor for specific conjugal moduli set $2^{n-2} + 1$, 2^n, $2^{n-2} – 1$ and design comparator having a dynamic range of 2n bits were proposed with CRT and implemented in PLD system. The design generates a high speed conversion process with an efficient significant hardware requirement for the DNA sequences. The timing simulation accelerator of the proposed RNS-SWA architecture indicates that the total delay summation is 12.0120ns at a clock speed of 185.53 MHz The percentage speed gained in our work shows a significant percentage ratio difference of 12.0840 which is 82.273 % faster than the non RNS state of art profiler [9].This is an indication that there is hope in future prospect for the computer scientist researchers in the field of RNS arithmetic and bioinformatics in addressing the speed threats challenge in SWA.

FUTURE WORK

In the upcoming future, we will try to a built redundant Residue Number System Architecture Model Based DNA Processor capable to detect and correct single bit LCS error in DNA sequences and also include the possibility to build an optimal RNS Polynomial Arithmetic algorithm with

capability to optimize Scaling division complexity for addressing biological gene sequences computing complication.

REFERENCES

Abdallah, M. Skavantzos, A. (1995) A systematic approach for selecting practical moduli sets for residue number systems, 27th Southeastern Symposium on System Theory, pp. 445.

Altschul, S. F., et al., (1990), A Basic Local Alignment Search Tools. *In Journal of Molecular Biology, vol 215, pp403 – 410.*

Parhami, B.(1990) Generalized signed-digit number system. *A unifying framework for redundant number representation, IEEE Transactions on Computers,Vol. 39, No. 1, pp. 89-98,*

Chang, C.H. Molahosseini, A.S., Zarandi, A.A.E., Tay, T.F.A (2015). New Paradigm to Data path Optimization for Low-Power and High Performance Digital Signal Processing Applications. *IEEE Circuits and Systems Magazine .15 (4),26-44.*

Chaitali B. D,, Partha.G and Amitabha.S.(2012).Design of a reconfigurable DSP processor with bit efficient Residue Number System. International Journal of VLSI design and Communication Systems (VLSICS) Vol.3, No.5,

Gbolagade. K.A. and Coton et al. (2009). An implementation on 0(n) Residue Number System to Mixed Radix Conversion Technique. *Proceedings –IEEE International Symposium on Circuits and Systems.*

Gbolagade. K.A. (2012). 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).*

Gbolagade K.A et al.. Chaves R, Sousa. L, Cotofana .S. D. (2010). An Improved RNS reverse converter for the {22n+1- 1,2n, 2n-1} moduli set. *An IEEE International Conference on Circuits and Systems (ISCAS 2010). Paris, France.2010;2103-2106.*

Hasan..L. and Al-Ars, .Z.(2007). Performance improvement of the Smith Waterman Algorithm. *An Annual workshop on circuits, systems and signal processing (ProRISC), (Veldhoven, The Netherlands), November 29 - 30 2007.*

Hassan. K. B. and Gbolagade .K. A. (2018). An Acceleration of Biological Sequence Alignment Using Residue Number System. *Asian Journal of Research in Computer Science 1(2): 1-10, 2018; Article no.AJRCOS.42834*

Hassan .K .B. Kazeem .A. G. (2016). Application of Smith-Wateman and Needleman Wunsch 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.*

Hassan. K B. Gbolagade. K. A. (2018) Acceleration of algorithm of Smith-Waterman using Recursive variable expansion. *International Journal of Advanced Research in Computer Engineering and Technology (IJARCET). 2018;7;(5). ISSN: 2278 – 1323*
.
Chiang, J. Shaw, M. Stenberg, and K. Truong, (2006) Hardware accelerator for genomic sequence alignment, August 30 - September 3, 2006.

Kwame O. Boateng and Edward Y. Baagyere (2012)**, A** Smith-Waterman Algorithm Accelerator Based on Liao
.
Y.K, et al (2004) M. L. Yin, and Y. Cheng.. A parallel implementation of the Smith-Waterman Algorithm for Massive Sequences Searching. September 1-5, 2004.

Miki,Y.,J. Swensen, D.Shattuck-Eidens, P.A. Futreal and K.Harshman et al,(1994).A Strong candidate for the breast and ovarian cancer susceptibility gene BRCA1 Science,266;66-71.

Olatunbosun. L.O. et al 2019] Lawal .T. D. and Gbolagade .K. A.(2019) An Efficient RNS Arithmetic in Bioinformatics- sequences. IJCSI International Journal of Computer Science Issues, Volume 16, Issue 6, November 2019. ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784.

Oliver.T.B. et al Schmidt, and D. Maskell, (2005) hyper customized processor for bio-sequence database scanning on FPGAs,

Omar Abdel Fattah (2011). Data Conversion in Residue numbers system. A thesis submitted to Department of Electrical and Computer Engineering McGill University Montreal, Canada; 2011

Park. et al (1996) Karplus K., Barrett, C. Hughey, D. Haussler, T. Hubbard, and C. Chothia Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. J. Mol. Biol., 284(4) : 1201– 1210.

Residue Number System (2012) **.** International Journal of Electronics and Communication Engineering. ISSN 0974 2166 Volume 5, Number 1 (2012), pp. 99-112.

Margerm, (2006) Reconfigurable computing in real world applications, Cray Inc., FPGA and Structured ASIC, February, 7, 2006.

Smith, T.F. and Waterman, M. S., (1981), Identification of common molecular sub sequences, journal of molecular Biology, vol. 147, pp 195-197. Stanford University, USA,

Schroeder, M. R.(1984) Number Theory in Science and Communication. Germany: Springer-Verlag, 1984.

Wei, Wang. Swamy, , M.N.S and. Ahmad, M.O. (2003) Moduli Selection in RNS for Efficient VLSI Implementation, International Symposium on Circuits and Systems, vol.4, pp. IV-512- IV-515, 25-28 .

Yamaguchi.Y. et al (2002) T. Maruyana, Y. Miyajima, and A. Konagaya,(2002) "High speed homology search using run-time reconfiguration,

Yang, B. H. W.(2002). A parallel Implementation of Smith -Waterman Sequence Comparison Algorithm.