Proceedings of the 28th IEEE
EMBS Annual International Conference
New York City, USA, Aug 30-Sept 3, 2006

HARDWARE ACCELERATOR FOR PREDICTION OF EXONS

Adeel Yusuf, Student Member, IEEE and Dr. Shoab A Khan, Member IEEE

Center for Advanced Studies in Engineering, Islamabad, Pakistan
adeelyusuf@hotmail.com, shoab@carepvtltd.com

Abstract— **Gene annotation is by nature a computationally intensive problem, as it needs to process huge data size of DNA sequences. This forces the need to look for alternate ways of implementing algorithms to predict exons. The paper presents an accelerator for indexing DNA sequences. The accelerator effectively exploits the 3-periodicity property exhibited by protein coding regions and indicates their presence in the sequence. Experimental results show superior performance compared with software-based approach for evaluating exons from DNA. The accelerator based PCI pluggable card offers a great utility to scientists and engineers actively involved in indexing DNA sequences.**

I. INTRODUCTION

Automated identification of exons in DNA sequences is a fundamental step in the computational annotation of genes. The field of signal processing has had a significant influence in computational genomics [1]. Various researchers have proposed different algorithms for automated prediction of protein coding regions [2, 3]. These algorithms can be broadly categorized as intrinsic or extrinsic.

Extrinsic techniques exploit the use of external annotated DNA database or training sets to calculate variables in the splicing algorithms or search for similarities in the external database [4] or both [5]. Intrinsic techniques do not require any external dataset.

In this paper we have modeled an intrinsic technique to predict the presence or absence of exons in the DNA sequence. Protein coding regions exhibit various properties discussed in [6]. One such property namely '3 periodicity' is exploited to calculate the protein coding regions.

Each strand of DNA is a chain of chemical "building blocks", called nucleotides, of which there are four types: adenine (abbreviated A), cytosine (C), guanine (G) and thymine (T).

The 3-periodicity property [2] states that the spectral energy derived from the DFT's (Discrete Fourier Transform) of the four binary signals representing a DNA protein coding region of length N, exhibits a peak at discrete frequency k = N/3. No such peak is observed in the spectral energy of noncoding regions.

In this paper, we present a hardware accelerator for predicting the protein coding regions. The need for a hardware approach is imminent as the sequences in general span through millions of nucleotides and indexing them through software applications is not efficient.

Hardware accelerators are mostly designed in multimedia and other computationally intensive applications to exploit parallelism in the algorithm [7]. Such accelerators appreciably reduce the time of computation, which was a bottleneck when implemented in software.

The algorithm presented in [3] exploits 3-periodicity property through position count function (PCF). It was observed that PCF has inherent parallelism, which can be exploited through an efficient digital design. The contribution of this paper is to improve the DFT based gene prediction algorithm [3] by separating the computational intensive part and implementing it in hardware after making alterations in the original algorithm.

This paper presents a system level architecture design of hardware accelerator for calculating the number of different nucleotides at each of the three phases of every codon.

The paper is organized as follows. Section 2 describes the splicing algorithm. Section 3 explains the proposed architecture for realization of algorithm in hardware, while experimental results are presented in Section 4. Finally Section 5 concludes the paper.

II. SPLICING ALGORITHM

A. Binary Representation

DNA strand $D[i]$ is represented as

$$D[i] = [A\ C\ T\ T\ G\ G\ C\ A\ T\ C\ C\ T\ G\ A] \quad (1)$$

The sequence is then transformed into four binary arrays $A[i]$, $T[i]$, $C[i]$ and $G[i]$, which indicate the presence or absence of these nucleotides at location i. This can be defined as

$$A[i_o] = 1 \text{ iff } D[i_o] = A.$$
$$\text{else, } A[i_o] = 0$$

For the DNA sequence mentioned at (1)

$A[i] = [1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1]$
$T[i] = [0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 0\ 0]$
$C[i] = [0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 1\ 1\ 0\ 0\ 0]$ and
$G[i] = [0\ 0\ 0\ 0\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]$

The four sequences mentioned above are converted into two binary indicator sequences, $R[i]$ and $W[i]$.

$R[i] = A[i] + T[i]$
$W[i] = G[i] + C[i]$

Since $R[i]+W[i] = 1$, therefore, $R[i]$ provides a complete description of the DNA within a window. Sequence $W[i]$ can be derived directly from $R[i]$, and vice versa.

B. Bartlett Window

Bartlett window with zero valued end-points is used, instead of the conventional rectangular window to prevent leakages of power of Fourier Transform over into adjacent frequencies. A 351-point Bartlett window W[n] is used with maximum value of 175 instead of 1 for fixed point implementation of floating point values.

$$W[n] = \begin{cases} n, & 0 \le n < \frac{1}{2}(N-1) \\ 350-n, & \frac{1}{2}(N-1) \le n < (N-1) \end{cases} \qquad (2)$$

Eq(2) produces a constant array $W[n]$.

$W[n] = [0,1,2,\ldots\ldots,174,175,174,173,\ldots\ldots\ldots,0]$

To realize the Bartlett window a bank of 351(8 bit wide) registers is created.

C. Position Count Function:

The binary DNA signal $R[i]$ is convolved with $W[i]$ to give the input for PCF.

$$A[i] = \sum_{i=0}^{N-1} R[i]W[i]$$

The signal $A[i]$, $(0 \le i < N)$ is parsed into non--overlapping words of length $\omega = 3$. The position count function (PCF) [3] for $A[i]$ is defined as

$$C_{\omega}^{A}(s) = \sum_{i=0}^{\left\lfloor \frac{N-1}{\omega} \right\rfloor} A[\omega i + s] \; for \; (0 \le s < \omega) \qquad (3)$$

PCF counts the number of 1's at phase s in the ω-bit parsed words.

PCF in digital design is realized by the induction of three Wallace Trees. In this reduction scheme partial products are divided into group of three each. These partial product groups are reduced simultaneously through carry save reduction technique. Each carry save reduction compresses 3 layers to 2 layers. These layers are then, with other layers from other partial product groups, form a reduced matrix. The reduced matrix is further divided into group of three layers each. This process continues until only two rows are left. At this stage any CPA (Carry Propagate Adder), which is used to perform full addition can be used to compute the final product. In our design we have used Ripple Carry Adder (RCA) to perform this task.

In digital design this scheme is used for partial product reduction but here the same is employed for parallel addition of all the Bartlett window values multiplexed with $R[i]$ at position 0,1 and 2 of every codon.

D. Calculation Of Magnitude, Average and SNR Values

The values of average (R_{av}), magnitude $(R[N/3])$ and SNR as calculated in [3] are

$$\left|\tilde{R}_{av}\right|^2 = \frac{1}{(N-1)}\left(N - \sum_{s=0}^{\omega-1} D_{\omega}^{R}(s)\right)\sum_{s=0}^{\omega-1} D_{\omega}^{R}(s) \qquad (4)$$

where, $D_{\omega}^{R}(s) = \sum_{n=0}^{\left[\frac{N}{\omega}\right]-1} A[\omega n + s]$

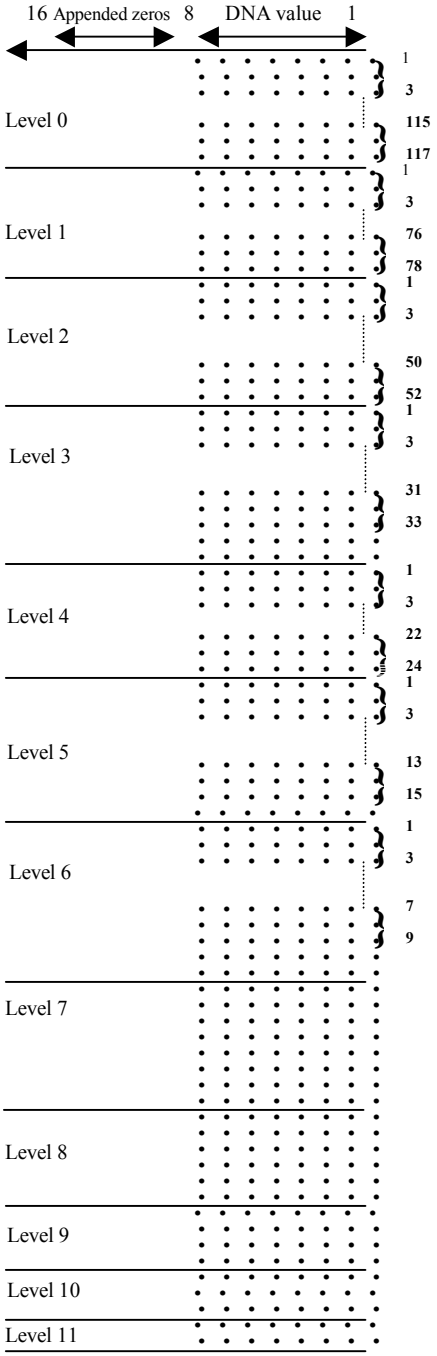

Figure 2.1: 16-bit Wallace tree reduction scheme for accelerator

$$\left|\tilde{R}[N/3]\right|^2 = \frac{1}{2}\left[\begin{array}{l}\left(C_\omega^R(0)-C_\omega^R(1)\right)^2+\left(C_\omega^R(1)-C_\omega^R(2)\right)^2 \\ +\left(C_\omega^R(2)-C_\omega^R(0)\right)^2\end{array}\right] \quad (5)$$

$$SNR[l_1] = \frac{\left|\tilde{R}[N/3]\right|^2}{2\left|\tilde{R}_{av}\right|^2} \quad (6)$$

III. ARCHITECTURE

A. The Behavioral Model

A major issue while implementing high-density architecture is to manage processes at a very high rate thus minimizing the wait time. Keeping in mind this constraint for controller, it has been portioned in hardware and software. Hardware software co implementation enables us to use flexibility of software while also making use of the high speed of hardware.

The system is designed on a PCI (Peripheral Component Interconnect, a type of PC interface) based FPGA (Field Programmable Gate Array, an one-chip programmable breadboard) card. The PCI application is designed to load DNA sequences in an on-chip FIFO(First In First Out) and to fetch results for final processing and display. The controller based upon its previous knowledge decides the process and time slot for the execution of each call.

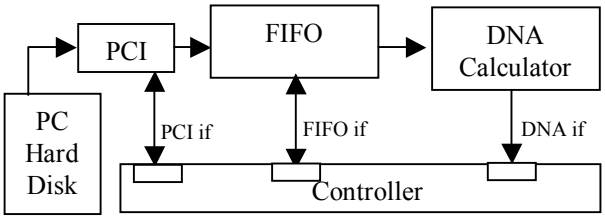

Figure 3.2: Behavioral Model

B. State Diagram

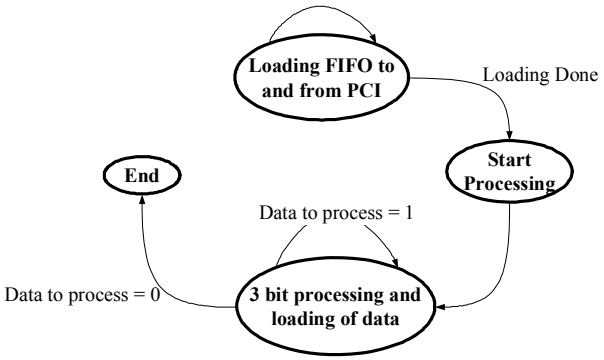

Figure 3.2: State Diagram

The state diagram shown above indicates the procedural steps involved in the computation of the SNR value mentioned at Eq. (6). Values of the DNA sequence are loaded into a 351-bit FIFO via PCI interface. After loading the first 351 bits processing starts and values of the magnitude and average are calculated. For every 351 bits processed, 3 bits are loaded in the FIFO before the accelerator recommences its computation. The values of Magnitude and Average are stored in two separate 32 bit wide FIFOs. After every computation the previous values of Magnitude and Average are moved to the PC hard disk via PCI interface. When all the values of DNA sequence are accounted for and there is no further bit left to be processed, the application exits.

C. Structural Model

The structural model at figure 3.3 shows the interconnections of sub-components in the accelerator. First 351 values of the DNA sequence are loaded from the FIFO to the register. These are multiplexed with the corresponding values of Bartlett window using 351 muxes. The output of muxes is represented as

$$Mux[i] = B[i] \text{ iff } R[i] = 1$$

These multiplexers are grouped into three separate groups. The muxes with first value of codon give output to Wallace 1.Similarly muxes with 2nd and 3rd values of every codon are connected to Wallace 2 and 3 respectively. The output of RCA 1,2 and 3 is the count sum of all 1st, 2nd and 3rd values of every codon in the sequence respectively.

Finally integrators and multipliers are connected as shown in Figure 3.3 to realize Eq. (4) and (5).

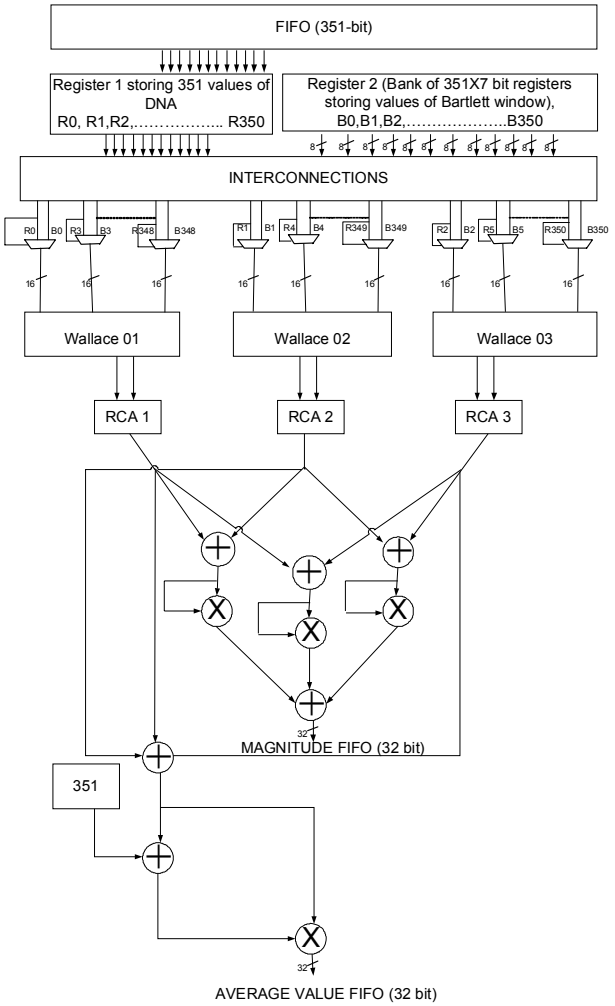

Figure 3.3: Structural Model

IV. RESULTS

The algorithm has been tested on Chromosome III of C.elegans (Accession No. NC003281) downloaded from NCBI database [8]. The experiments conducted verify that the hardware approach requires far less physical computational units i.e. adders, multipliers etc then its sequential counterpart.

Table 4.1: Sub-Components for Sequential and Hardwired

	Multipliers	Adders
Sequential	3870	3694
Hardware Accelerator	4	351

The algorithm proposed by [3] was analyzed and the number of sub-components required by functional units were calculated. The same was also done on the proposed accelerator. The results shown in the table 4.1 indicate less space on chip required.

Experiments also indicate that number of adders used in the accelerator can be reduced by 1/3rd if we reuse Wallace 1 to compute values of Wallace 2 and 3. However controller design will become far more complicated.

On the other hand inherent parallelism in the algorithm can be further exploited by addition of more groups of Wallace trees. This will result in further speeding up the annotation process; thus proving scalability of our design.

The hardwired design proposed also led to a significant reduction in the number of computational operations performed indicating reduced processing power required.

Table 4.2: Computations Performed per Functional Unit

		Bartlett	PCF	Av.	Mag.
Sequential	Adder	526	1053	4	5
	Multiplier	702	1053	5	4
Hardware Acc.	MUX	351	0	0	0
	Adder	0	115	2	4
	Multiplier	0	0	1	3

Table 4.2 indicates a significant reduction in the number of operations needed to perform the Bartlett Windowing and PCF operation. This is because Bartlett was realized by the introduction of multiplexers and PCF was achieved by employing Wallace trees. However, Average and Magnitude did not show significant difference in the sequential and hardwired approaches.

Table 4.3: Time Units Consumed

Time units Comparison	
Sequential	3343
Hardware Accelerator	119

The throughput of the accelerator was measured by the time units consumed by both sequential and hardwired designs. It was assumed that both multiplication and addition require one time unit.

Another significant achievement is the memory allocation on board which in our case was restricted to only three FIFOs. This was achieved considering the fact that at any stage of the algorithm a maximum of 351 values of DNA are used which gives a 32-bit output of Average and Magnitude.

Finally the complexity of hardware was significantly reduced by normalizing the values of Bartlett Eq. (2), which resulted in transformation of all values in fixed point format.

Hence it can be safely stated that the accelerator presented in this paper requires less space on chip and gives maximum throughput without introducing any complexity in hardware design and memory requirements.

V. CONCLUSION

An alternate approach to solve the protein-coding problem is presented. In this paper, we provided