

PREDICTION OF PROTEIN CODING REGIONS ON A PROGRAMMABLE VLIW ARCHITECTURE

Adeel Yusuf, Dawood Khan and Dr. Shoab A Khan

Center for Advanced Studies in Engineering, Islamabad, Pakistan
adeel@case.edu.pk, dawood.khan@gmail.com, shoab@carepvlttd.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 a hardware-based approach in which a Digital Signal Processor is programmed to compute the computationally expensive part of the algorithm. The processor effectively exploits the 3-periodicity property exhibited by protein coding regions and indicates their presence in the sequence. Experimental results show superior performance when compared with implementation on a general purpose Intel processor for evaluating exons from DNA. This 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 Trimedia digital signal processor PNX1300EH for predicting the protein coding regions. The VLIW (Very Long Instruction Word) architecture is capable of executing many operations in a single-clock cycle. It contains multiple functional units that execute primitive instructions in parallel. The long instruction that is fetched from the program is composed of a number of primitive instructions, each of which is fed to one of the functional units, such that they are all executed explicitly in parallel, and controlled by the programmer (or compiler). The need for an alternate approach is imminent as the sequences in general span through millions of nucleotides and indexing them through software applications is not efficient.

DSPs are mostly programmed in multimedia and other computationally intensive applications to exploit parallelism in the algorithm [7, 8]. Such processors appreciably reduce the time of computation, which was a bottleneck when implemented in software on a general purpose processor like Intel.

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 efficient programming of VLIW processor. The contribution of this paper is to improve the DFT based gene prediction algorithm [3] by separating the computational intensive part and implementing it on a VLIW core after making alterations in the original algorithm.

This paper presents a system level programming model of Trimedia PNX1300EH processor 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 PNX1300 architecture. Section 4 explains the proposed programming model for implementation of algorithm on the Trimedia DSP, while experimental results are presented in Section 5. Finally Section 6 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)

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

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 288-point Bartlett window $W[n]$ is used with maximum value of 144 instead of 1 for fixed point implementation of floating point values.

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

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

$$W[n] = [0, 1, 2, \dots, 143, 144, 143, \dots, 0]$$

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

$$C_{\omega}^A(s) = \sum_{i=0}^{\lfloor \frac{N-1}{\omega} \rfloor} A[\omega i + s] \text{ for } (0 \leq s < \omega) \quad (3)$$

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

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

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

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

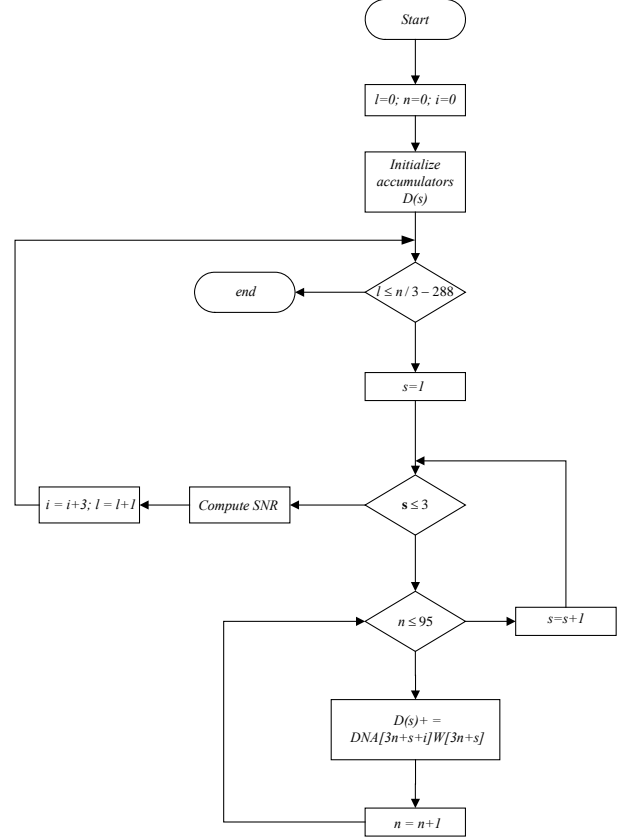


Figure 2.1: Data Flow Diagram of splicing Algorithm

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

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

III. PNX 1300 ARCHITECTURE

We chose PNX1300EH processor to map the subject algorithm upon. PNX1300 processor achieves over seven billion operations per second. PNX1300s are comparable in ease of programmability to general-purpose processors. The SDE enables application development entirely in the C and C++ languages. PNX1300 Series processor leverages a powerful C/C++-programmable very-long instruction word (VLIW) TriMedia CPU to coordinate on-chip activities. Independent, on-chip, bus-mastering DMA peripheral units manage and format datastream I/O and accelerate

processing of algorithms. Memory hierarchy manages internal I/O and streamlines access to external memory [9].

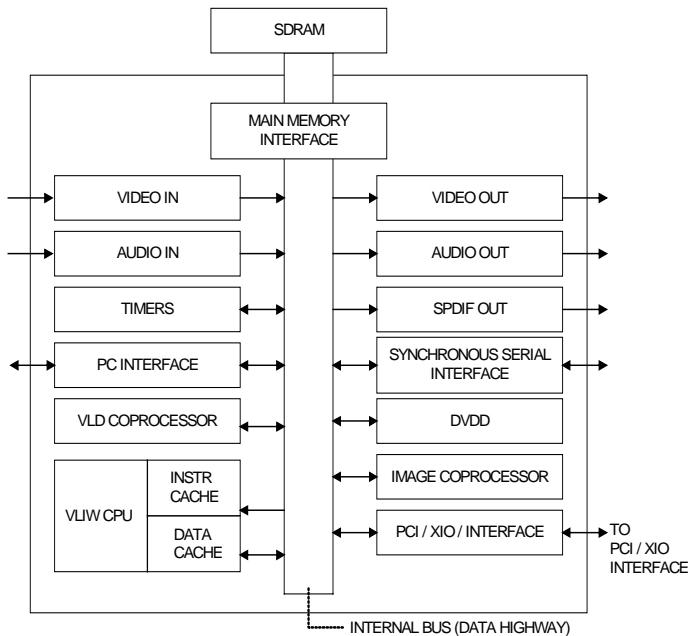


Figure 3.1: Block Diagram of PNX1300 Architecture

PNX1300 series is basically used for multimedia applications and there were many inherent features of the processor, which were not exploited due to nature of the algorithm used. However, the main reason for selecting PNX1300 was the number of registers available for processing. Since our algorithm needs 288 bits of incoming data of DNA and 288 values of Bartlett window, the most suitable option was choosing a Trimedia multimedia processor due to number of available registers. Some of the key features are appended below:

Clock Speed PNX130x	143, 180, or 200 MHz
Issue Slots	5
Functional Units	27, pipelined
Registers	128 (32 bits wide)
Special/SIMD Ops	32

Table 3.1: Performance table of basic functional units

Name	Qty	Latency	Recovery
constant	5	1	1
integer ALU	5	1	1
memory load/store	2	3	1
shift	2	1	1
DSPALU	2	2	1
DSP multiply	2	3	1
branch	3	3	1
float ALU	2	3	1
integer/float mul	2	3	1
float compare	1	1	1
float sqrt./divide	1	17	16

IV. TRIMEDIA PNX1300 PROGRAMMING MODEL

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 partitioned 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 programmable digital signal processor (DSP) card. The PCI application is designed to load DNA sequences in an on-chip register 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.

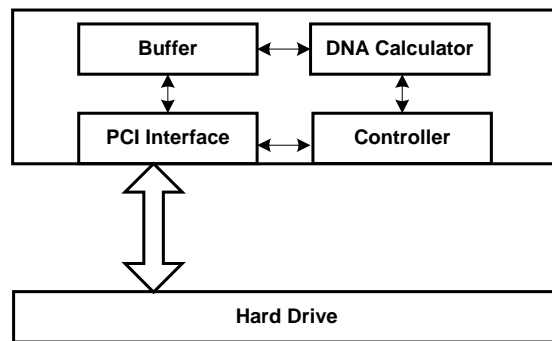


Figure 4.1: Behavioral Model

B. State Diagram

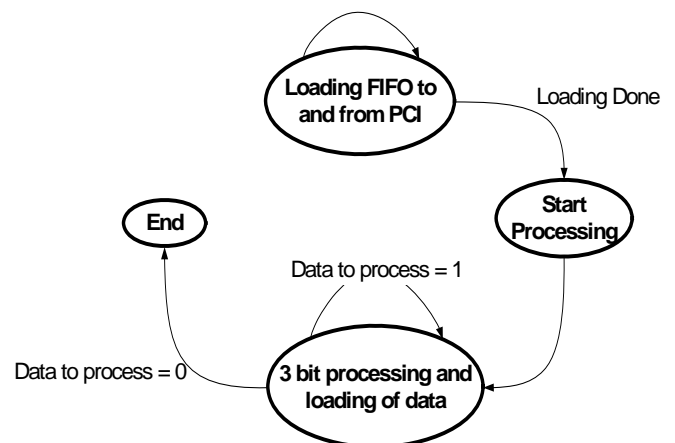


Figure 4.2: State Diagram

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 nine 32-bit registers via PCI interface. After loading the first 288 bits processing starts and values of the magnitude and average are calculated. For every 288 bits processed, 3 bits are loaded in the registers before the accelerator recommences its computation. The value of

SNR is stored in a separate 32 bit wide FIFO. After every computation the previous value of SNR is 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. Program Implementation

The Trimedia processor is equipped with a data cache and an instruction cache. User programs the instruction cache whereas the data cache in our case is continuously receiving data from the PC hard disk. As a rule, the processor is stalled whenever there is a cache-miss and remains in that state until the cache is updated [10]. If this issue is not addressed while programming, the processor might spend a significant proportion of the total time in stall cycles, as signal-processing algorithms tend to have a bad locality. The number of stall cycles in our case was minimized by continuously updating the data cache through efficient programming techniques.

The code written for the algorithm employs use of intrinsics, which directly address the processor for efficient implementation of functions. We used an interactive process to refine our code, by checking the assembly generated for the inefficiencies and reformulating the C code. In addition, the compiler does not generate the SIMD operations. The programmer must write these operations in the C code.

The code for the kernel designed for the algorithm is shown below:

```
int W[288]={0};
double func(int l);
void bartlett(void);

//Kernel
for(n=0; n<288/U; n+=1)
{
    //U1
    if(DNA[U*n + 0+i]&0x01)Dsum0 +=W[U*n + 0];
    if(DNA[U*n+1+i]&0x01)Dsum1+=W[U*n+1];
    if(DNA[U*n+2+i]&0x01)Dsum2 +=W[U*n +2];

    //U2
    if(DNA[U*n+3+i]&0x01)Dsum3+=W[U*n+3];
    if(DNA[U*n+4+i]&0x01)Dsum4+=W[U*n+4];
    if(DNA[U*n + 5+i]&0x01)Dsum5 +=W[U*n+ 5];

    .....

    //U32
    if(DNA[U*n+93+i]&0x01)Dsum3+=W[U*n+93];
    if(DNA[U*n+94+i]&0x01)Dsum4+=W[U*n+94];
    if(DNA[U*n+95+i]&0x01)Dsum5+=W[U*n+95];
    Dsum[0] = Dsum0+Dsum3;
    Dsum[1] = Dsum1+Dsum4;
    Dsum[2] = Dsum2+Dsum5;
}
}
```

The code makes use of the fact that Trimedia processors are equipped with 128 (32 bit wide) registers, which should always be populated in order to continuously, update the data cache. This will ultimately reduce stall cycles and hence improve efficiency of the code designed.

V. RESULTS

Various versions of C code were developed for the algorithm discussed in the earlier sections. Each version build upon the preceding versions. Version 1 implemented the algorithm in C as is and the other versions started to enhance the earlier versions. The succeeding versions brought in the loop unrolling of the kernel and multiple accumulators to decrease the number of cycles required for loop kernel.

A. Basic Implementation:

For this implementation we chose Bartlett window of length 288, which is convolved with the data sequence of length 'n' as a test. We also experimented with other type of windows (hanning, hamming, Bartlett, rectangular etc) and there results have been shown in the given figures. It can be inferred from the figures and the results generated from experimenting with different types of windows that, the aim of increasing the exon prediction rate while minimizing false alarms is best achieved with Bartlett window. Hamming and hanning are also good options but they increase the false alarms i.e. predicting non-coded regions as coded. The basic implementation was developed in Matlab to confirm the accuracy of the algorithm.

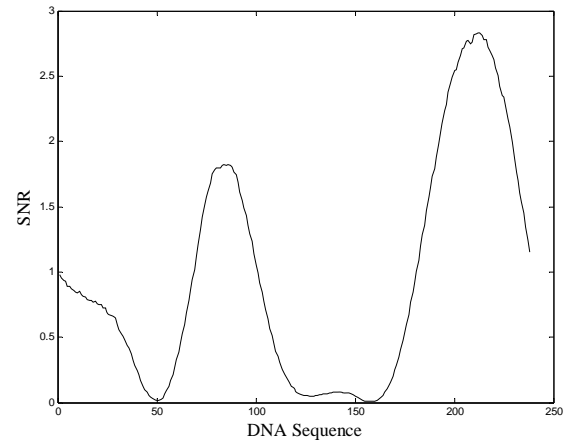


Figure 5.1 SNR plot of a reference DNA sequence when convolved with hamming window

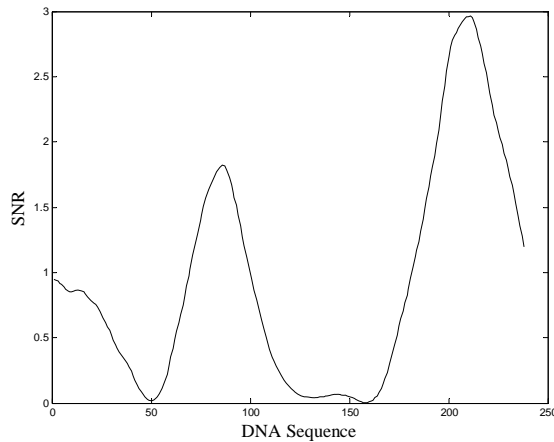


Figure 5.2 SNR plot of a reference DNA sequence when convolved with Bartlett window

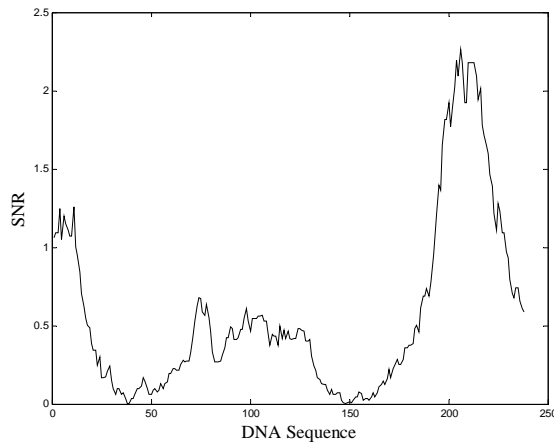


Figure 5.3 SNR plot of a reference DNA sequence when convolved with Rectangular window

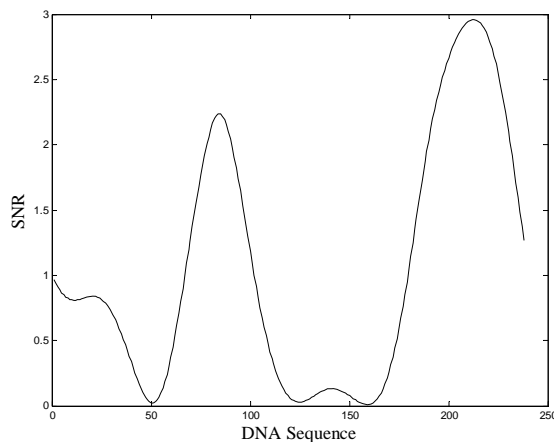


Figure 5.4 SNR plot of a reference DNA sequence when convolved with hanning window

B. C - implementation (Version 1)

Version 1 is the C-implementation of the Matlab code. In this code the window operating on the DNA sequence is

moved by 3 nucleotides after every iteration till the end of the DNA sequence is reached. The samples corresponding to the first, second and third position of every codon in the DNA sequence are added into three different accumulators, and at the end of the iteration, window is shifted by one codon through the DNA sequence. This implementation takes 2740 cycles to compute the result.

C. Effect of Loop Unrolling (Version 2)

The version two is better performance wise and generates as this version uses loop unrolling technique to reduce the loop overhead. This version only requires 784 cycles. The only catch is that we have to unroll 96 times; this is because we have to synchronize the window with the three shifts in every cycle ($288/3 = 96$) and 96 values of DNA sequence are held in form of ones and zeros in three integers which are loaded simultaneously with fears of synchronizations problems between the window and the sequence being processed. The cycles for this are less as compared to version 1 but still there is significant room for improvement.

D. Maximizing use of registers (Version 3)

The version 2 used only one accumulator since we are loading three integers and they all are being processed independently we chose to use three different accumulators to store the values of the convolutions sum. We reached this conclusion after taking feedback from the assembly file generated by the Trimedia tool chain. The number of cycles that we got after this modification is 339.

Since we are loading 96 proteins, 96 values of window and doing 96 additions and 6 extra additions to add the three values of accumulators. This runs for three cycles and $(96 \times 3) = 288$ (window size). So the minimum number of cycles we can achieve ideally is $(96 \times 3) + 18 = 306$ cycles.

In the final version of our programmed code we have achieved the goal for this optimized number of cycles. The extra 16 cycles are for overheads, function calls and returns.

The algorithm has been tested on Chromosome III of *C.elegans* (Accession No. NC003281) downloaded from NCBI database [11]. The experiments conducted verify that when the algorithm is effectively mapped on a programmable VLIW core, it requires far less computational units than its sequential counterpart programmed on a superscalar.

To experimentally verify the efficiency of our program we profiled a DNA sequence of 1000 nucleotides and used it as input data for the algorithm. The results are appended in table 5.1, which indicates the performance of our programmed code and compares it with that of the superscalar.

The superscalar was profiled by using the c profiler and the readings were consistent for over 1000000 times, whereas Trimedia simulator was used to calculate the cycles of all the versions of our VLIW program.

Table 5.1: Cycles consumed when algorithm is implemented on superscalar and on Trimedia VLIW core

	Cycles Consumed					Total
	Bartlett	PCF	Av	Mag	SNR	
Super-scalar	696	3093	96	98	40	4023
Version 1	488	2020	96	98	38	2740
Version 2	96	488	88	92	20	784
Version 3	96	194	14	16	19	339
Min possible cycles	96	163	07	09	20	306

The throughput of the accelerator was measured by the cycles consumed by both sequential and hardwired designs.

Another significant achievement is the memory allocation on board which in our case was available upto 128 registers. Hence, we could perform unrolling of the loop and hence, the throughput of the system increased. 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 programmed processor presented in this paper gives maximum throughput at the cost of intelligent programming as per the capabilities of the processor provided.

VI. CONCLUSION

An alternate approach to solve the protein-coding problem is presented. In this paper, we provided justification that programming an algorithm for annotation of genes on a programmable VLIW core is by far the most effective way for indexing DNA.

The approach of hardware implementation can be further extended by employing hybrid methods to improve upon the accuracy of detection of the protein coding regions.

VII. REFERENCES

[1] P. P. Vaidyanathan and B.-J. Yoon, "The role of signal-processing concepts in genomics and proteomics," *Journal of the Franklin Institute, Special Issue on Genomics*, 2004.

[2] V. R. Chechetkin and A. Y. Turygin, "Size-dependence of Three-periodicity and Long-range Correlations in DNA Sequences," *Physics Letters A*, vol. 199, 1995, pp. 75-80.

[3] S. Datta, and A. Asif, A Fast DFT based Gene Prediction Algorithm for Identification of Protein Coding Regions, in *Proceedings of IEEE International Conference in Acoustics, Speech, and Signal Processing, ICASSP '05*, Philadelphia, PA, vol. 3, pp. 113-116, Mar. 2005.

[4] W. Gish and D.J. States. "Identification of protein coding regions by database similarity search," *Nature Genet.*, pages 266--272, 1993.

[5] C. Burge, "Identification of genes in human genomic DNA," Ph.D. dissertation, Stanford University, May 1997.

[6] Fickett, J. and Tung, C. (1992). "Assessment of protein coding measures," *Nucleic Acids Research*, 20(24):6441--6450.

[7] Sima, M.; Cotofana, S.D.; Vassiliadis, S.; van Eijndhoven, J.T.J.;Vissers,K.A., "Pel reconstruction on FPGA-augmented TriMedia," *Very Large Scale Integration (VLSI) Systems*, IEEE Transactions on Volume 12, Issue 6, June 2004 Page(s):622 – 635

[8] De Strycker, L.; Termont, P.; Vandewege, J.; Haitsma, J.; Kalker, A.; Maes, M.; Depovere, G.; "An implementation of a real-time digital watermarking process for broadcast monitoring on a Trimedia VLIW processor," *Image Processing and Its Applications*, 1999. Seventh International Conference on (Conf. Publ. No. 465) Volume 2, 13-15 July 1999 Page(s):775 - 779 vol.2

[9] TM130x PCI Media Processor (TriMedia) Hardware data book May 2000 . ©1999, Philips Electronics North America Corporation.

[10] TriMedia Programmers Reference Manual (SDE 2.0) ©1999, Philips Semiconductors

[11] National Centre for Biotechnology Information (NCBI). [Online]. Available:<http://www.ncbi.nlm.nih.gov/>.