Elsevier Editorial System(tm) for Pattern Recognition

Manuscript Draft

Manuscript Number:

Title: A Coprocessor Architecture for Fast Protein Structure Prediction

Article Type: Special Issue on Bioinformatics

Section/Category:

Keywords: Protein structure prediction; PSIPRED; PSI BLAST; Neural network; VLSI architecture.

Corresponding Author: Dr. Chaitali Chakrabarti,

Corresponding Author's Institution: Arizona State University

First Author: Rahim Khoja

Order of Authors: Rahim Khoja; Mehul Marolia; Tinku Acharya; Chaitali Chakrabarti

Manuscript Region of Origin:

A Coprocessor Architecture for Fast Protein Structure Prediction

Rahim Khoja[a], Mehul Marolia[a], Tinku Acharya[a, b] and Chaitali Chakrabarti[a, i]

[a]Department of Electrical Engineering, Arizona State University, Tempe, USA.
[b]Avisere Inc., Tucson, Arizona, USA.

[i]Corresponding author; Tel.: 1 (480) 965-9516; Fax: 1 (480) 965-8325; Email: chaitali.chakrabarti@asu.edu; Address: ASU Main Campus, Ira A Fulton School of Engineering, Department of Electrical Engineering, PO Box 875706, Tempe, Arizona 85287-5706.

Abstract

Predicting protein structure from amino acid sequence is computationally very intensive. In order to speed up protein sequence matching and processing, we present a novel coprocessor architecture for fast protein structure prediction. The architecture consists of systolic arrays to speed up the data intensive sequence alignment and structure prediction steps, and finite state machines for the control dominated steps. The architecture has been synthesized using Synopsys DC Compiler in 0.18 micron CMOS technology and details of the area and timing performance are provided. A procedure to develop architectures with area-time trade-offs has also been presented.

Keywords: Protein structure prediction; PSIPRED; PSI BLAST; Neural network; VLSI architecture.

1. Introduction

In the last century, biologists have made vast progress in studying the characteristics and complexity of living organisms. In this century significant advances have been made in understanding the cellular processes, in terms of molecular force interaction, that defines the diversity in living organisms. The progress and advancement in computational molecular biology (also called *bioinformatics*) will probably lead us to answer some of the fundamental questions such as ?Why is one person different from another?, or ?Why is one person more susceptible to a particular disease than another?, or ?What is the cure of AIDS or cancer?. Today, bioinformatics is an interdisciplinary area of study involving diverse fields such as biology, biochemistry, medicine, genetics, computer science, information technology, mathematics, statistics, physics, etc. [1, 2].

A better understanding of the cell structure at the molecular level and knowledge of the underlying chemical structures and functions has led to a special interest in the study of protein molecules. Proteins are organic polymers built by polypeptide chains of amino acids inside a cell. Many of the hormones and enzymes in our body are proteins, and the functionality and characteristics of the proteins are responsible for many of the cellular functions. Proteins are the most varied group of molecules involved in biochemical processes and understanding them is a core area of research in bioinformatics.

In order to understand the structure of a protein, we start with the DNA (deoxyribonucleic acid) of an organism that is a chain of four types of bases *Adenine* (A), *Cytosine* (C), *Guanine* (G), and *Thymine* (T). The DNA is *transcribed* to produce m-RNA (messenger ribonucleic acid), which is then *translated* to produce protein in our body. Each nonoverlapping triplet (codon) of DNA bases is treated as an amino acid and the translation process maps the triplets of four bases (A, C, G, T) to the set of 20 amino acids which form the basic building block of proteins [2]. The length of a protein represented in the form of a linear chain of amino acids, can vary from 10s to 1000s. The molecular interactions cause the linear chain of amino acids to fold and twist, resulting in different functionalities. The protein structure can be predicted by matching an amino acid query sequence with existing sequences in biological databases. Since a query sequence ranges from 50 to 400 amino acids and the databases have ~500 thousand amino acid sequences, the prediction problem is computationally very intensive, and processing this enormous data quickly is an area of active research.

According to Moore?s law of genomics, ?The PDB (Protein Data Bank) of known structures will continue to double every 3 years and the number of sequences in want of a predicted structure will continue to double every few months? [3]. Thus, in order to process the large volume of protein data, it is becoming increasingly important to develop faster and simpler techniques for protein structure prediction. Experimental ways of finding the protein structure is very time consuming, expensive and does not meet the current demands. Also, they do not provide insight into how the proteins fold due to different molecular interactions and why they fold into a particular structure only. Methods such as ab initio [4] that require exhaustive molecular force interaction study to develop the protein structure, fails to provide the necessary speed. Hence, the trend has shifted towards developing structure prediction methods and modeling techniques which can predict the corresponding three dimensional protein structure from the amino acid sequences.

There are numerous structure prediction methods [5-7] that differ in the type of approach as well as the degree of accuracy. Most of these methods are based on homologous template matching and use databases of existing predicted structures to develop the three dimensional structure for the amino acid sequence under study. We have selected PSIPRED [8] as the preferred method of prediction because it is simple and demonstrates a high level of accuracy.

The PSIPRED algorithm predicts the secondary structure of a protein by using neural networks to process the PSSM (Position Specific Scoring Matrix) generated by PSI-BLAST (Position Specific Iterated ? Basic Local Alignment Search Tool) [9]. The steps involve scanning the database to get a list of highly probable hit locations which are then extended with dynamic programming methods to align with the query sequence. The process of generating the alignment profile is iterated, and the PSSM matrix is used by the neural network to predict the secondary structure.

The computationally intensive nature of the PSIPRED algorithm makes it necessary to develop a special purpose VLSI architecture that would serve as a coprocessor. In this paper, we present a specialized architecture that uses a combination of systolic arrays for sequence alignment and structure prediction and finite state machines for the remaining steps. The architecture has been implemented using VHDL and synthesized using Synopsys DC Compiler in 0.18 micron CMOS technology. The synthesized architecture requires 663 940 units of gate area and 346 KB of memory, and can be clocked at 100 MHz. It can predict the

secondary structure of a query sequence of length ~150 amino acids using a database of 135 million amino acids in less than 10 seconds. To the best of our knowledge, this is the first VLSI architecture for protein structure prediction proposed in the literature to date.

To better understand the performance of the proposed architecture, we have analyzed it with respect to datapath size, memory and timing requirements. We have suggested a procedure for allotment of area and memory resources to achieve the desired timing. We have also considered the resource requirements to facilitate processing larger query sequences and the ever increasing protein databases.

The rest of the paper is organized as follows. We first explain the protein structure prediction algorithm in section 2. Then we present the details of the proposed architecture in section 3. Section 4 highlights the synthesis results for the architecture, the tradeoffs between area (including memory requirement) and time, and the effects of query sequence lengths and database sizes on the architectural performance. Lastly, the concluding remarks are made in section 5.

2. Protein Structure Prediction Technique

Protein is a biopolymer of amino acids arranged in a specific sequence that fold into a three-dimensional shape. The shape of a three dimensional protein molecule is determined by the sequential order of the amino acids. The folded structure can be constructed with three types of substructure (secondary structure): helix structure (α), sheet structure (β) or other coil structures (γ).

Of the various secondary structure prediction methods that exist today [4-7], PSIPRED is a method that demonstrated average accuracy of 77.3% at CASP3 (Third Critical Assessment of Structure Prediction) meeting [8]. We selected this method because of its simplicity and the capability of giving the highest level of accuracy published to date.

The PSIPRED algorithm takes a sequence of amino acids as a query sequence input and predicts the corresponding secondary structure. It is implemented in two steps: (1) PSI-BLAST, for multiple sequence alignment followed by updating a sequence profile iteratively to generate position specific scoring matrix (PSSM) and (2) secondary structure prediction and filtering using neural networks. The key components of the algorithm and their interactions are shown in Fig. 1.

2.1 PSI-BLAST

Basic Local Alignment Search Tool (BLAST) [10] is a heuristic tool based on dynamic programming [11] to find sequence similarities in protein and DNA sequences. The central idea of BLAST is that any statistically significant alignment between query and database sequences must have at least one word pair of length *w* scoring above a threshold *T*. Once the location of this word pair is known, it is extended on either side to obtain High Scoring Pair (HSP) of aligned sequences with a score S. This score is obtained from the BLOSUM-62 [12] substitution matrix by comparing the aligned amino acids in the HSP. PSI-BLAST [9] is an iterated version of BLAST which generates a Position Specific Scoring Matrix (PSSM) from gapped alignments between multiple sequences. Database searches using this matrix are more sensitive and are better able to detect weaker homologies than pairwise comparison methods [13]. The key steps of PSI-BLAST are summarized below:

1) A hash table is generated which contains a list of words scoring above threshold *T* when aligned with a word in the query sequence.

2) The protein sequence is scanned using the hash table and a hit list is generated containing the locations having two high scoring words in close neighborhood.

3) A list of High Scoring Pairs (HSP) is generated corresponding to the hit list locations having a high score S when extended. This score *S* is normalized as $S? = (\lambda S - \ln k)/\ln 2$, where, λ and *k* are statistically determined parameters.

4) The HSPs are extended using dynamic programming to generate local alignment with error $< E = mn/2^{S?}$ where *m* is the length of the query sequence, *n* is the length of the database sequence and *S?* is the normalized score.

5) Steps 2 to 4 are repeated for all the database sequences. A multiple sequence alignment is obtained which is used to calculate new scores for the PSSM matrix (of size 20 x length of query sequence) [14].

6) Steps 1 to 5 are repeated three times after which the PSSM matrix is read by the neural network for secondary structure prediction.

2.2 Secondary Structure Prediction

The secondary structure prediction step of PSIPRED computes, for each amino acid, the confidence level that it is likely to be an α, β or γ structure. This stage consists of two neural networks connected in a feed forward configuration. The first neural network (trained using non-redundant set of protein sequences) scans a window of (2n+1) amino acids and predicts the secondary structure of the central element of window depending on the amino acid?s ?n? preceding and ?n? succeeding amino acids. The second neural network filters the erroneously predicted and structurally improbable profiles.

In the hardware implementation of the neural network, the synaptic weights are calculated using matrix multiplication. The input vector is multiplied by weights stored in the network and products are summed up to give outputs based on the sigmoid activation function. Back propagation algorithm is used for offline training.

3. PSIPRED Architecture

In this section we describe the details of the coprocessor architecture to implement PSIPRED. The coprocessor performs sequence matching and multiple sequence alignment using PSI-BLAST to generate PSSM. It then processes the PSSM using neural networks to predict the secondary structure in terms of α, β and γ structures. The architectural modules and interconnections between the modules of the coprocessor are shown in Fig. 2.

The main processor provides the coprocessor with the protein database, the query sequence, the hash table, and the statistical constants. These are first loaded into the shared memory and later on, parts of it are loaded to the individual unit?s memory as required. The hash table is a list of words with scores greater than threshold *T* when aligned with words of size 3 from the query sequence (using BLOSUM-62 scoring matrix). It helps the coprocessor to scan the database at very high speeds.

The coprocessor consists of specialized units corresponding to every step in the PSIPRED algorithm. These include hit list generation, ungapped extension, gapped extension, PSSM generation and secondary structure prediction from PSSM. The query is processed in a pipelined fashion through the different units. As soon as the hit list unit provides an input to the ungapped unit, it processes the data and passes an output to the gapped unit. After the gapped unit finishes processing its inputs, the PSSM unit works on the ~50 database

sequences obtained from the gapped unit and generates a PSSM matrix. This sequence of events is repeated two more times and after a total of three iterations, the neural network unit processes the PSSM matrix to predict the secondary structure of the query sequence.

The coprocessor?s memory organization consists of (1) a shared memory structure which stores part of the database sequence, query sequence etc. from the main processor and is accessed by multiple units, and (2) internal memory in each of the specialized units to store the intermediate results. The main controller provides the interface between the main processor and the coprocessor, and also synchronizes and controls the specialized units. In the rest of this section, we describe the functionality and the architectural details of each of the specialized units.

3.1 Hit List FSM

Hit list is a list of the most probable alignments between the database sequence and the query sequence that could generate a High Scoring Pair (HSP). Each of the Hit List FSMs shown in Fig. 2 reads the hash table and database sequence from the shared memory and loads them to its own memory (Memory 1.1 and Memory 1.2 respectively). It then uses this data to generate a list of locations which satisfy the two-hit criteria. The FSM steps can be summarized as follows:

1) Reset the signals and initialize the offset values and range of input dataset

2) Read a word from database sequence (for proteins, a word consists of three consecutive amino acids)

3) Read hash table entry of that database word to get the hit locations

4) Read the corresponding previous hits on same alignments

5) Check two-hit criteria; if the new hit locations are within distance *A* (generic constant) of the previous hits on same alignment then it is most probably a HSP

6) Report the database locations and query locations that match two-hit criteria by setting respective flag in list of alignments

The corresponding FSM is implemented in 6 states to minimize the number of clock cycles per input amino acid. It has also been optimized to reduce the memory read write operations to a minimum. The hit list generation is the most computationally intensive step of PSIBLAST algorithm since it has to compare the entire protein database against the query sequence using the preprocessed hash table. In our architecture we

have replicated the hardware to include two hit list generation units to match its throughput with that of the next unit in the pipeline.

3.2 Ungapped Extension FSM

The ungapped extension unit processes the hit list to generate a list of statistically significant HSPs (Fig. 1). It reads high scoring words from both the hit list FSMs alternately. These hits are then extended in both directions till the score falls below the best score found for shorter extensions (in any direction) by a certain value *X* (generic constant).

The ungapped unit consists of an FSM controller and three local memory modules. The FSM starts by reading the locations of the HSP from the hit list unit and the query and database sequences from Memory 2.1 and 2.2 respectively. In the next state it calculates the pair-wise score by comparing the database and the query amino acid using BLOSUM-62 matrix [12] that is stored in Memory 2.3. The obtained score is compared against the best case score; if the difference is greater than *X*, the next word in the hit list is read and the process is repeated. But if the difference is less than *X*, then the alignment is extended further by reading the next set of amino acids from the query and database sequences and a new score is calculated. The HSP thus generated is written back in Memory 2.1, which is then read by the gapped extension unit.

3.3 Gapped Extension Unit

Gaps are introduced in the local alignment using the dynamic programming algorithm of [15]. The edit distance H_{ij} between the amino acid query sequence *i* and the database sequence *j* is calculated by a systolic array. Fig. 3 (a) shows the data flow graph of the systolic array. Here *m* is the length of the query sequence and *n* is the length of the database sequence. The edit distance defined as the number of insertion/deletion and substitution operations required to convert one string into the other, is described in Fig. 3 (b). Here S_{ij} is the replacement cost obtained from BLOSUM-62 which can be high for amino acids that share a common ancestor and low (negative) for those which are a result of several mutations. Also g = -4 is the cost of inserting/deleting a character from the string.

The gapped extension unit consists of local memory, a systolic array and an FSM to regulate the signals flowing in and out of the array. The local memory consists of 4 memory modules: Memory 3.1 stores the

database sequence, Memory 3.2 stores the substitution cost matrix, Memory 3.3 stores the query sequence and the resulting matching database sequences and Memory 3.4 stores the intermediate results from the systolic array. The multiple state FSM starts by reading a part of the query sequence from Memory 3.3 and mapping it onto each processing element (PE) in the systolic array. In the next state it loads the corresponding row from BLOSUM-62 substitution matrix from Memory 3.2 into each PE. The database sequence from Memory 3.1 is then scanned through the array to compute the alignment scores. If *k* is the number of available PEs and k < m, then the data flow graph is partitioned using the Locally Parallel and Globally Sequential (LPGS) scheme into *k* blocks. The switching of blocks is also controlled by the FSM as it reads and writes the intermediate results in and out of Memory 3.4.

Once the query and database sequences have been scanned through this array, backtracking is initiated. This is done by finding the largest score calculated in the matrix and tracing backwards. At the end of backtracking, the pairwise alignment between the two sequences is obtained. This database sequence which has a score sufficient enough to report as a hit is stored in Memory 3.3 along with the query sequence. This procedure is repeated for several database sequences and a multiple alignment is thus obtained.

3.4 PSSM Calculation Unit

The multiple sequence alignment obtained from BLAST has *m* columns and *x* rows, where *m* is the length of the query sequence and *x* is the number of database sequences present in the multiple alignment. The PSSM matrix calculated from the multiple alignment also has *m* columns but 20 rows, one for each amino acid. For *m* different columns, the substitution score in each column should be different. This is because the scores depend not only upon amino acids appearing in that column but also appearing in other columns. The procedure is implemented by a state machine in the following way

1) In the first few states, the multiple alignment is pruned so that all rows that are >98% identical to the query are purged, and only one copy is retained.

2) A simpler reduced multiple alignment Mc is constructed for each column. This is a set of sequences that have an amino acid present in that particular column.

3) The weighted frequency of an amino acid f_i is calculated as the observed frequency of the amino acid in a particular column averaged with the weight of the sequence [16].

4) The pseudo-count frequency g_i of an amino acid is calculated as

$g_i = \sum_j (f_j / P_j) q_{ij}$ (1)

where P_j is the background probability of amino acid *j* occurring in a sequence and q_{ij} is the target frequency which depends upon substitution scores between amino acids *i* and *j* in BLOSUM-62 matrix. Amino acids favored by the substitution matrix get higher pseudo-count frequencies.

5) The position specific score is then calculated as log (Q_i/P_i) where P_i is the background probability and Q_i is the estimated probability of ith amino acid in the desired position

$Q_i = (\alpha f_i + \beta g_i) / (\alpha + \beta)$ (2)

where α = Nc ? 1 (where Nc is the mean number of different amino acid types in various columns of the reduced multiple alignment Mc), and β is an arbitrary weight, empirically calculated as 10.

The local memory consists of 4 memory modules: Memory 4.1 provides the background probabilities of amino acids, Memory 4.2 supplies the target frequencies as per BLOSUM 62 matrix, Memory 4.3 keeps a list of log values which are accessed to calculate the value of log Q_i and Memory 4.4 stores the new score values. The scores obtained from one iteration are used in the next iteration of PSI-BLAST with minor modifications. This process is repeated for three iterations, after which the scores are sufficiently sensitive to be used by the neural network for secondary structure prediction.

3.5 Secondary Structure Prediction and Filtering Unit

A neural network architecture is used for secondary structure prediction from PSSM and for filtering the prediction. The first neural network (containing one hidden layer) is a feed forward neural network which looks at a window of 15 amino acids to predict the secondary structure for the central element. For 20 amino acids plus an entry to mark end of sequence, there are 315 input neurons (15x21) and three output neurons corresponding to the three types of secondary structures. According to the heuristics in [8], 75 hidden neurons give sufficiently accurate results and hence we program the architecture to implement a 315 x 75 x 3 layer neural network.

The second feed forward neural network contains one hidden layer and looks at a window of 15 secondary structures (predicted by the first neural network) to give a filtered output for the central element. For three types of secondary structures and an end marker, we require 60 input neurons (15x4) and three output

neurons. For this neural network, 60 hidden neurons are found to give sufficient accuracy and hence the architecture is configured to implement a 60 x 60 x 3 layer neural network [8].

The neural network used for secondary structure prediction can be implemented as shown in Fig. 4. The processing elements of the neural network are configured as a systolic array and are used for matrix multiplication operation. To provide high speed data access, two memories are provided as shown in Fig. 4 which stores the pair of matrix vectors which are input to the processing elements and the intermediate result matrix at various stages. The controller FSM synchronizes the operations between the processing elements and the memory interface [17].

The neural network has two modes of operation; prediction and training. Since the training is an offline routine (using back propagation algorithm), the online prediction of protein secondary structures can be made faster by increasing the number of processors in the systolic array. The processing element described in Fig. 5 can be configured to carry out MAC operation during the prediction run and can carry out addition and multiplication operations during training run. It has sigmoid processing available on one of the inputs (A): F(A)=A or sigmoid(A). This simple processing element can be used to implement multiple operations on inputs A, B, C and register D as shown below.

Reset : A = 0 B = 0 C = 0 => F(A) * B + C = 0

MAC : A = in1 B = in2 => F(A) * B + D => D

Add : A = 1 B = in1 C = in2 => B + C

Multiply : A = in1 B = in2 C = 0 => A * B

The sigmoid activation function scales input in the range -8 to +8 non-linearly into a value between 0 and 1. For simplifying the hardware, the sigmoid function can be calculated using 15 segment piecewise linear approximation (PWL). To approximate sigmoid function for numbers between -8 to 0, the number is first converted to a positive number and then approximated using the PWL method and then converted back to the negative number [18].

3.6 Main Controller

The main controller coordinates and synchronizes the computational flow through the different units so that all units get an uninterrupted supply of data to support the pipelined processing. It synchronizes the operation of specialized units using handshaking signals, and providing parameters such as memory offsets, control parameters, values of various constants, and the required dataset sequences. It also takes care of loading the shared memory with required set of database sequences, query sequence and pre-processed hash table from the query sequence. Recall that the protein database is very large (~450 K sequences) and thus it is not feasible to load the entire database. The main controller loads a section of the database sequence at a time. In fact, it pre-fetches blocks of 200 protein sequences at a time (~300 amino acids per sequence) into the shared memory and provides one sequence at a time to the different units as required.

4. Results

The VHDL model of the architecture described in section 3 was compiled and simulated using Cadence NcLaunch and SimVision 05.10-s006. We refer to this architecture as Configuration 1. The architecture was synthesized using Synopsys DC Compiler v2002.05 in 0.18µm CMOS technology. It has a total gate area of 663 940 units and memory of size 346KB. Using the Configuration 1 architecture, the time taken to process a standard query of length ~150 amino acids against the entire database of 135 million is under 10 seconds, which is a small fraction of what it takes on a high end general purpose processor.

In the rest of this section, we first present a case study in section 4.1, followed by area and time calculations in section 4.2, and the area time trade offs for this architecture in section 4.3. We also present the effect of query and database length on the architectural performance in section 4.4.

4.1. Case Study

In order to understand the computational load of each of the components, we present a detailed analysis for the case when the query sequence is of length ~150 amino acids, the protein database has ~450 K sequences, and the average length of a database sequence is ~300 amino acids. Fig. 6 lists the volume of data that is processed by the different units for this case. These calculations are based on the example given in [19].

The hit list generation unit takes the entire database as an input and generates a list of high scoring words as described in section 2.1. This list has approximately 1.5 million words, and thus the hit list unit reduces the search space by a factor of 90 (135 million/1.5 million). This list of high scoring words is then expanded by the ungapped extension unit which finds only 8000 of them to be of any significance to be passed over to the gapped unit. Thus the ungapped extension unit reduces the search space by a factor of approximately 200. The gapped extension unit now has to gap extend these HSPs and eventually only ~50 database sequences are found to have a close match with the query sequence. Next, we present the area and time parameters for the coprocessor architecture for this particular case.

4.2. Area and Timing Performance

Table 1 lists the area and memory required by the individual units of the Configuration 1 architecture and Table 2 lists the timing performance. The area of each of the units is obtained from Synopsys DC compiler. The memory sizes are calculated in the following way. Each hit list unit requires 66 KB of memory to store a part of the database sequence, the hash table and the hit list. With two hit list units the memory requirement doubles to 132 KB. The ungapped extension unit stores the query sequence, a part of the database sequence, and the substitution matrix which amounts to a total of 4 KB. The gapped extension unit stores the substitution matrix, the query sequence and a part of database sequence. It also requires memory to store intermediate results from the PEs. The total memory required by this unit is 105 KB. The PSSM calculation unit requires approximately 5 KB of memory. It reads different values from this memory as per equations 1 and 2 to calculate the new substitution scores. It also needs memory to store the new PSSM matrix. The neural network unit requires 100 KB of memory to store its intermediate results and internal weights. Thus a total of 346 KB of memory is required by the Configuration 1 architecture.

The time it takes for each unit to process an input is shown in Fig. 6 and the operation times of the different units are presented in Fig. 7. The hit list unit processes one amino acid every 4 clock cycles. Hence to process the entire database it takes 135 million X 4 = 540 million clock cycles. After it processes ~90 amino acids which takes 360 (90 X 4) clock cycles, it passes an output to the ungapped extension unit. But the ungapped extension unit is capable of processing an input every 200 clock cycles! Thus the hit list unit is 1.8 (~2) times slower than the ungapped unit and it is clear that we need two hit list units to work in parallel to

give an output every 180 (360/2) clock cycles. With this parallelization, the time it takes for the hit list units to search for high scoring words in the entire database is now 270 million clock cycles.

Now the next unit performs ungapped extensions on 1.5 million words, each requiring 200 clock cycles. Hence in one iteration it requires 1.5 million X 200 = 300 million clock cycles. The ungapped extension unit passes one in about 200 extensions to the gapped unit which is equivalent to one output every 40 K (200 X 200) clock cycles. For throughput matching, we require that the gapped extension unit be able to process an input in at least 40 K clock cycles. Since one PE in the gapped unit takes about 48 150 clock cycles, we add another PE and bring down the processing time to around 25 650 clock cycles. The time taken to extend 8000 HSPs is 25 650 X 8 000 = 205.2 million clock cycles provided that it does not have to wait for inputs from the ungapped unit. Apart from this, the 50 database sequences have to be written to a memory for the PSSM calculation unit which takes another 50 X 300 (average database length) = 15 K clock cycles. Hence the gapped unit spends 205 215 K (205 200 K + 15 K) active clock cycles per iteration. But, since the gapped unit works faster than the ungapped unit, it has to wait for the ungapped unit to provide it with inputs. As a result, the gapped unit spends a minimum of 300 million clock cycles plus the time taken to process the last input (~26 K clock cycles), for a total of ~300 million clock cycles.

The PSSM calculation unit generates a new PSSM matrix in another 100 K clock cycles. The number of cycles in each iteration is approximately 300 million (due to gapped unit) plus 0.1 million (due to PSSM unit) resulting in a total of 300.1 million clock cycles. Since the whole process is repeated three times (see Fig. 3), the total number of clock cycles is equal to 900.3 million clock cycles.

After the three iterations of PSI BLAST algorithm have been run, the secondary protein structure prediction algorithm is initiated. As per section 2.2, this step takes about 10 million clock cycles. Thus the total time taken to process the query sequence is under a 1000 million clock cycles which translates to under 10 seconds on a 100 MHz coprocessor.

4.3. Area-Time trade offs

While the Configuration 1 architecture predicts the secondary structure in less than 10 seconds (which is much faster than most general purpose processors) in many situations, it might be necessary to reduce the prediction time further. One way of achieving this would be to increase the level of parallelism in the

proposed architecture. The rest of the section describes our procedure to selectively parallelize some of the specialized units.

First, we consider doubling the number of hit list units. The number of ungapped extension units have to be doubled as well to match the timing. As a result, the time taken by these two units is quite close (about 405 million cycles and 450 million cycles respectively). The time taken by the gapped extension unit also needs to match with that of the hit list unit and the ungapped extension unit.

In order to determine the level of parallelism in the gapped extension unit, we make use of Fig. 8 which plots the time taken per extension and the gate area as a function of the number of PEs. This plot has been derived using synthesized results according to which each PE requires about 55 K gate units and the FSM requires about 100 K gate units of area. From this plot, we see that when the number of PEs is larger than 15, the gain in time is not worth the price paid in increase in area.

Finally we do not consider parallelizing the PSSM unit and the neural network unit since it causes a small reduction in the total computation time at the expense of a reasonable increase in the gate area. Table 2 displays the area and time results of several configurations corresponding to different levels of parallelism.

Fig. 9 plots the computation time in seconds versus the gate area and memory in KB of the coprocessor architecture. From this figure we see that in order to reduce the overall computation time, the price paid in terms of area and memory is substantial. Thus while the coprocessor can be parallelized to significantly reduce the prediction time, there is a limit to the benefits that can be obtained by the parallelization.

4.4. Effect of Query length and Database size

4.4.1 Query Length

The area and timing analysis given in section 4.2 is based on a typical query sequence length of ~150 amino acids. But in reality the query sequences can range from ~50 amino acids to ~400 amino acids. As the length of the query sequence increases, the time taken to process it also increases. In terms of the architecture, the memory size increases though there is no change in the datapath. We elaborate on these features next.

A longer query sequence increases the time taken by each of the units. The hit list unit now produces more hits requiring 4 clock cycles per extension, which implies a linear increase in time. In the ungapped unit, the

length of the HSP increases with larger query sequences, and thus takes more time to be generated. The systolic array in the gapped extension takes more time since it has to match the database with a longer input sequence. The PSSM calculation unit now has a larger PSSM matrix to calculate and requires more clock cycles. The neural network unit also takes more time since it has to process a larger PSSM matrix.

In addition to the increase in the computation time, each unit now requires larger memory as shown in Fig. 10. The increase in memory for hit list units is mainly due to larger hash table requirements while the increase in memory requirement for gapped extension unit is due to the larger PSSM matrix. In the neural network, the increase in memory is due to the large number of intermediate results generated. Memory requirements of the PSSM calculation unit and the ungapped extension unit do not increase considerably. Overall, the memory requirement increases almost linearly with the increase in the length of the query sequence.

4.4.2 Database size

As the size of the database increases, there is a substantial increase in the search space, which translates to an increase in the time taken by our architecture. In fact, there is an appreciable increase in the processing time in the earlier stages of the pipelined architecture. For instance, the hit list unit requires more effort as it is likely to generate a longer hit list and the ungapped extension unit requires more effort as it has to extend more probable HSPs. The increase in effort in the gapped extension unit is not substantial as the search space increases only mildly. The PSSM calculation unit does a few more calculations while the neural network unit is unaffected by the increase in database size. Thus an increase in the database size causes an increase in the computational load of the hit list unit, the ungapped extension unit and to some degree the gapped extension unit. The corresponding increase in the computation time can be compensated by parallelizing the hit list unit and the ungapped extension unit as described in section 4.3.

5. Conclusion

In this paper, we presented a novel coprocessor architecture for implementing one of the most popular protein structure prediction algorithms, PSIPRED. To the best of our knowledge, this is the first special purpose VLSI architecture for protein structure prediction proposed in the literature. We achieved several orders of speedup in algorithm execution time by pipelining the operations, exploiting the pipelining and

parallelism capabilities of systolic arrays, and optimizing the FSM implementations. We implemented the PSIPRED algorithm in VHDL and synthesized it using Synopsys DC in 0.18 micron CMOS technology. The synthesized architecture requires 663 940 units of gate area and 346 KB memory. It can be clocked at 100 MHz and can predict protein structures of average sized query sequence in a fraction of a minute.

We provided architectural details as well as an analysis of the area (including memory) and timing of each of the units. We also presented a procedure to reduce the timing requirements by replicating a selected set of units and matching the throughput of consecutive stages. Further more, we analyzed the impact of large query sequence length and database size on timing and memory requirements of this architecture.

We are currently working on optimizing the synthesized architecture so that it can be clocked at higher frequencies. Our final goal is to develop a programmable architecture that is capable of supporting multiple protein structure prediction algorithms on the same architectural platform. We believe that this work would be very useful in demonstrating the capability of specialized VLSI architectures to overcome the computational road block posed by problems in bioinformatics.

References

[1] R. Nigam, Bioinformatics: Issues and Challenges, in Information Technology: Principles and Applications, A.K. Ray and T. Acharya (Eds.), Prentice-Hall of India, New Delhi, 2004.

[2] S. Mitra and T. Acharya, Data Mining: Multimedia, Soft Computing, and Bioinformatics, John Wiley & Sons, Hoboken, New Jersey, 2004.

[3] R. H. Kretsinger, R. E. Ison, S. Hovmoller, Prediction of Protein Structure, Methods in Enzymology, Academic Press 2004, vol. 383, pp. 27.

[4] D. T. Jones, Successful ab initio prediction of the tertiary structure of NK-lysin using multiple sequences and ecognized supersecondary structural motifs, Proteins: Struct. Funct. Genet., vol. S1, pp.185-191, 1997.

[5] P. Y. Chou, & G. D. Fasman, Conformational parameters for amino acids in helical, sheet, and random coil regions calculated from proteins, Biochemistry, vol. 13, pp. 211-222, 1974.

[6] B. Rost, & C. Sander, Prediction of protein secondary structure at better than 70% accuracy, J. Mol. Biol., vol. 232, pp. 584-599, 1993.

[7] C. Geourjon, & G. Deleage, SOPMA: significant improvements in protein secondary structure prediction by consensus prediction from multiple alignments, Comp. Appl. Biosci., vol. 11, pp. 681-684, 1995.

[8] D. T. Jones, Protein Secondary Structure Prediction Based on Position-specific Scoring Matrices, J. Mol. Bio., vol. 292, pp. 195-202, 1999.

[9] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. H. Zhang, Z. Zhang, W. Miller, & D. J. Lipman, Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucl. Acids Res., vol. 25, pp. 3389-3402, 1997.

[10] S. F. Altschul, W Gish, W Miller, E. W. Myers, & D. J. Lipman, Basic Local Alignment Search Tool, J. Mol. Bio., vol. 215, pp. 403-410, 1990.

[11] R. A. Wagner, & M. J. Fischer, The string to string correction problem, J. of the ACM, vol. 21, pp. 168-173, 1974.

[12] S. Henikoff, & J. G. Henikoff, Natl. Acad. Sci. USA, vol. 89, pp. 10915-10919, 1992.

[13] R. L. Tatusov, S. F. Altschul, & E. V. Koonin, Detection of conserved segments in protiens: Iterative scanning of sequence databases with alignment blocks, Proc. Natl. Acad. Sci. USA, vol. 91, pp. 12091-12095, December 1994.

[14] I. Eidhammer, I. Jonassen, W. R. Taylor, Protein bioinformatics : an algorithmic approach to sequence and structure analysis, J. Wiley & Sons, New York, 2004.

[15] T. F. Smith, & M. S. Waterman, Identification of common molecular subsequences, J. Mol. Biol., vol. 147, pp. 195-197, 1981.

[16]S. Henikoff, & J. G. Henikoff, Position based sequence weights, J. Mol. Biol., vol. 243, pp. 574-578, 1994.

[17] S. Pal, D. Zhang, Neural Networks and Systolic Array Design, Series in Machine Perception and Artificial Intelligence ? Vol. 49, World Scientific Publishing Co., Singapore, 2002.

[18] P. Murtagh, A. C. Tsoi, Implementation issues of sigmoid function and its derivative for VLSI digital neural networks, IEEE Proceedings - E, vol. 139, no. 3, pp. 207-214, May 1992.

[19] http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/psi1.html

About the author - RAHIM KHOJA received his Bachelors degree in Electronics Engineering from Mumbai University, India in 2003 and is currently working towards his Masters degree in Electrical Engineering at Arizona State University. His research interests include VLSI and low power architectures. He is currently researching VLSI architectures for Bioinformatic Applications at Arizona State University.

About the author - MEHUL MAROLIA received his Bachelors degree in Electronics Engineering from Mumbai University, India in 2003 and is currently working towards his Masters degree in Electrical Engineering at Arizona State University. His area of specialization includes VLSI design and architecture. His research interest involves architectures for bioinformatics applications.

About the author - TINKU ACHARYA received his PhD from University of Central Florida in 1994. He is the Chief Technology Officer of Avisere Inc., Tucson, Arizona and an Adjunct Professor at Arizona State University. He is inventor of 98 awarded US and European patents, and author of 4 books; Data Mining, JPEG2000, Image Processing (Wiley, USA), and Information Technology (Prentice Hall, India).

About the author - CHAITALI CHAKRABARTI received her PhD from the University of Maryland, College Park, in 1990. She has been on the faculty of Arizona State University since 1990. Her research interests are low power system design, VLSI algorithms and architectures for communications, signal processing and bioinformatics. She is the Associate Editor of the IEEE Transactions on Signal Processing and the Journal of VLSI Signal Processing.

List of Figures

List of Tables

Table 1.

Hardware Unit Description	Area (Unit Gates)	Memory (KB)
Hit List Generation Unit (2)	93 158	132
Ungapped Extension Unit	55 028	4
Gapped Extension Unit (2 PEs)	210 000	105
PSSM Calculation Unit	143 930	5
Neural Network Unit	161 842	100

Table 2.

Configuration		Hit list generation	Ungapped extension	Gapped extension	PSSM calculation	Neural Network	Total
1.	No of units	2	1	2	1	1	7
	Time (clock cycles)	810 000 000	900 000 000	615 645 000	300 000	10 000 000	~10 sec
	Area (K unit gates)	93 158	55 028	210 000	143 930	161 842	663 940
	Memory (KB)	132	4	105	5	100	346
2.	No of units	4	2	3	1	1	11
	Time (clock cycles)	405 000 000	450 000 000	435 645 000	300 000	10 000 000	~5 sec
	Area (K unit gates)	186 316	110 056	265 000	143 930	161 842	867 144
	Memory (KB)	264	8	105	5	100	482
3.	No of units	8	4	7	1	1	21
	Time (clock cycles)	202 500 000	225 000 000	229 941 000	300 000	10 000 000	~2.5 sec
	Area (K unit gates)	372 632	220 112	485 000	143 930	161 842	1 383 516
	Memory (KB)	528	16	105	5	100	754
4.	No of units	16	8	15	1	1	41
	Time (clock cycles)	101 250 000	112 500 000	147 645 000	300 000	10 000 000	~1.7 sec
	Area (K unit gates)	745 264	440 224	1 475 000	143 930	161 842	2 966 260
	Memory (KB)	1 056	32	105	5	100	1 298

Fig. 1.

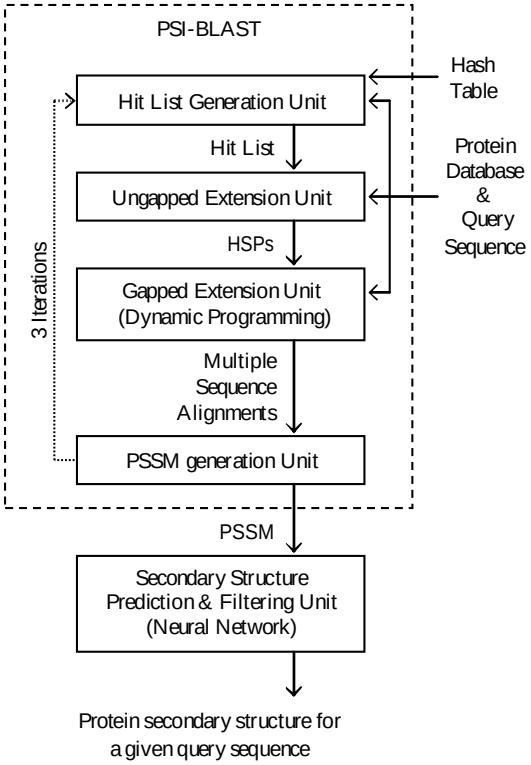

PSI-BLAST
Hash Table
Hit List Generation Unit
Hit List
Protein Database & Query Sequence
Ungapped Extension Unit
HSPs
Gapped Extension Unit (Dynamic Programming)
3 Iterations
Multiple Sequence Alignments
PSSM generation Unit
PSSM
Secondary Structure Prediction & Filtering Unit (Neural Network)
Protein secondary structure for a given query sequence

Fig. 2.

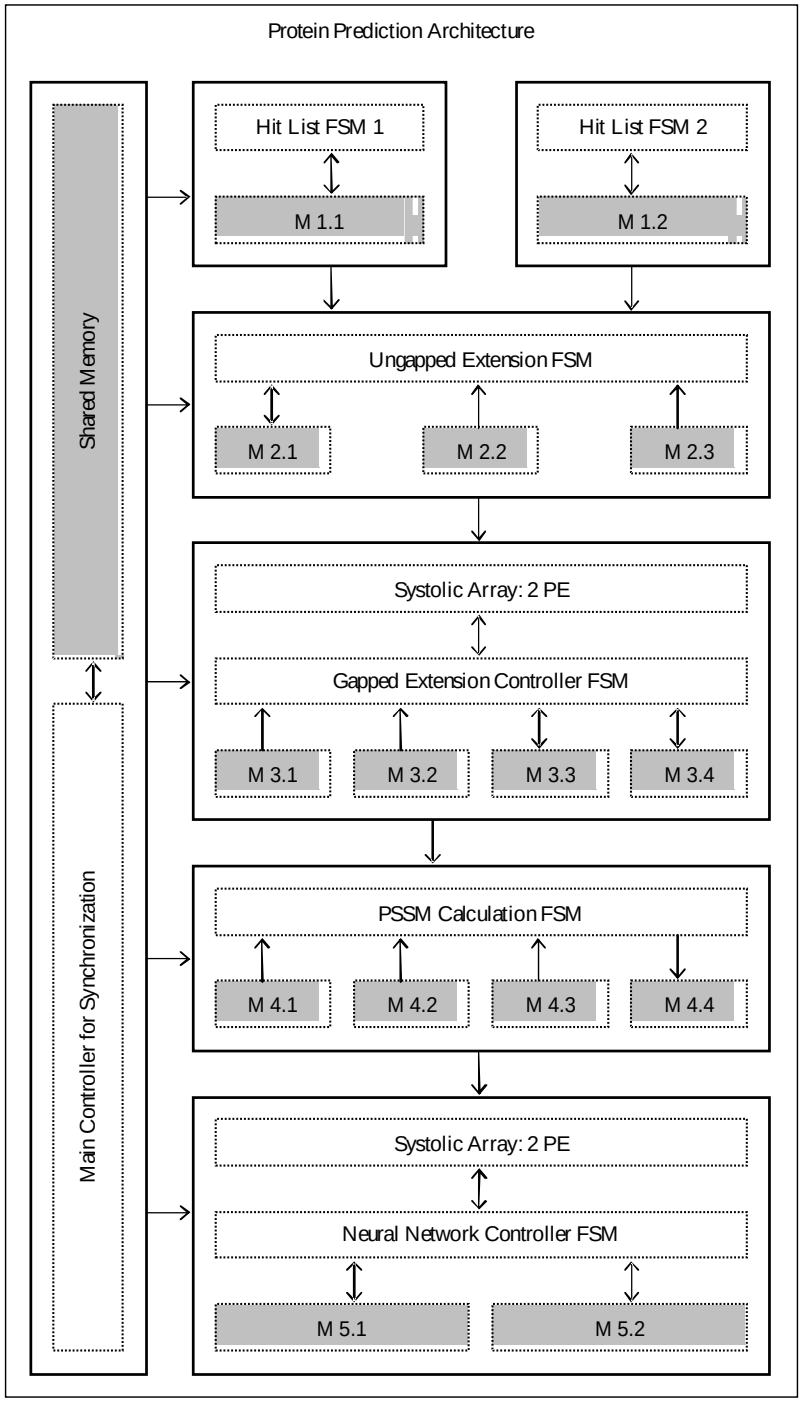
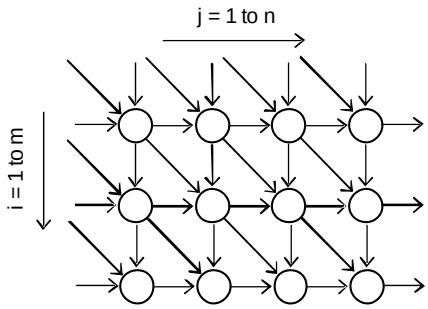
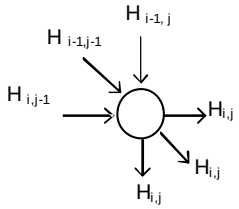

Protein Prediction Architecture
Shared Memory
Main Controller for Synchronization
Hit List FSM 1
M 1.1
Hit List FSM 2
M 1.2
Ungapped Extension FSM
M 2.1
M 2.2
M 2.3
Systolic Array: 2 PE
Gapped Extension Controller FSM
M 3.1
M 3.2
M 3.3
M 3.4
PSSM Calculation FSM
M 4.1
M 4.2
M 4.3
M 4.4
Systolic Array: 2 PE
Neural Network Controller FSM
M 5.1
M 5.2

Fig. 3.



(a)



$$H_{(i,j)} = \max\{H_{(i-1,j-1)} + S_{ij}, H_{(i-1,j)} + g, H_{(i,j-1)} + g\}$$

(b)

Fig. 4.

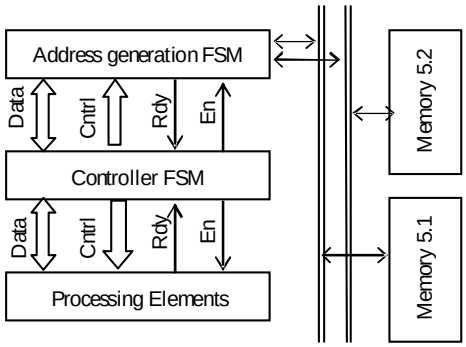

Address generation FSM
Data
Cntrl
Rdy
En
Controller FSM
Data
Cntrl
Rdy
En
Processing Elements
Memory 5.2
Memory 5.1

Fig. 5.

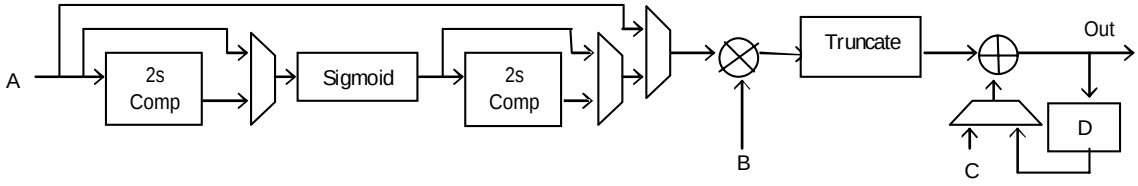


Fig. 6.

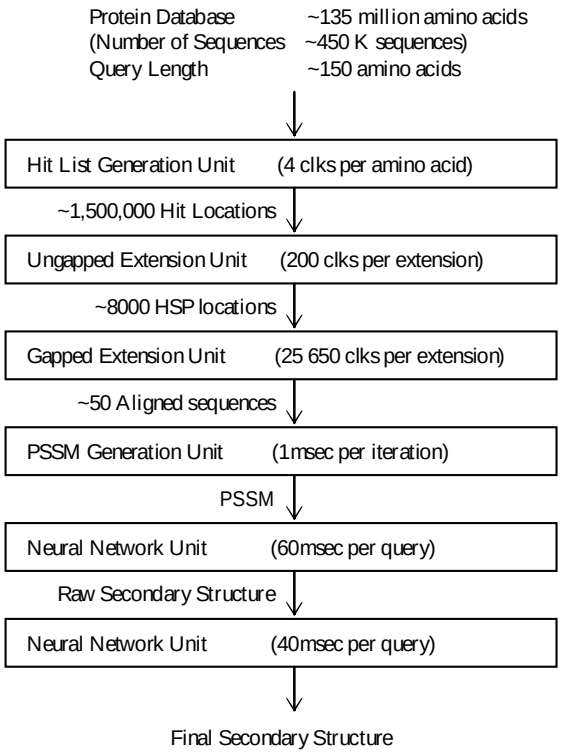

Protein Database ~135 million amino acids
(Number of Sequences ~450 K sequences)
Query Length ~150 amino acids
Hit List Generation Unit (4 clks per amino acid)
~1,500,000 Hit Locations
Ungapped Extension Unit (200 clks per extension)
~8000 HSP locations
Gapped Extension Unit (25 650 clks per extension)
~50 Aligned sequences
PSSM Generation Unit (1msec per iteration)
PSSM
Neural Network Unit (60msec per query)
Raw Secondary Structure
Neural Network Unit (40msec per query)
Final Secondary Structure

Fig. 7.

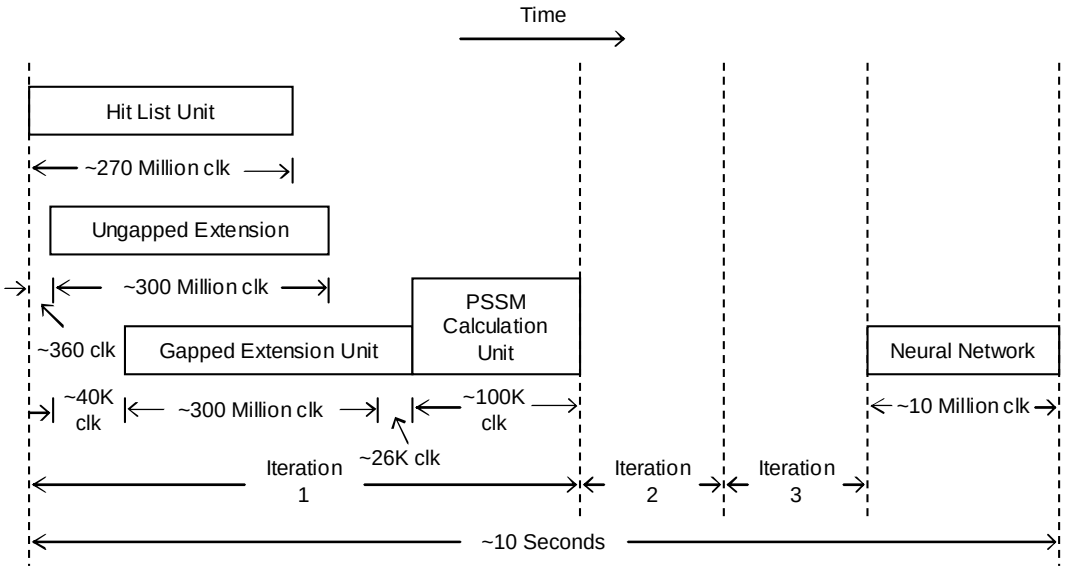

Time
Hit List Unit
~270 Million clk
Ungapped Extension
~300 Million clk
~360 clk
Gapped Extension Unit
PSSM Calculation Unit
Neural Network
~40K clk
~300 Million clk
~26K clk
~100K clk
~10 Million clk
Iteration 1
Iteration 2
Iteration 3
~10 Seconds

Fig. 8.

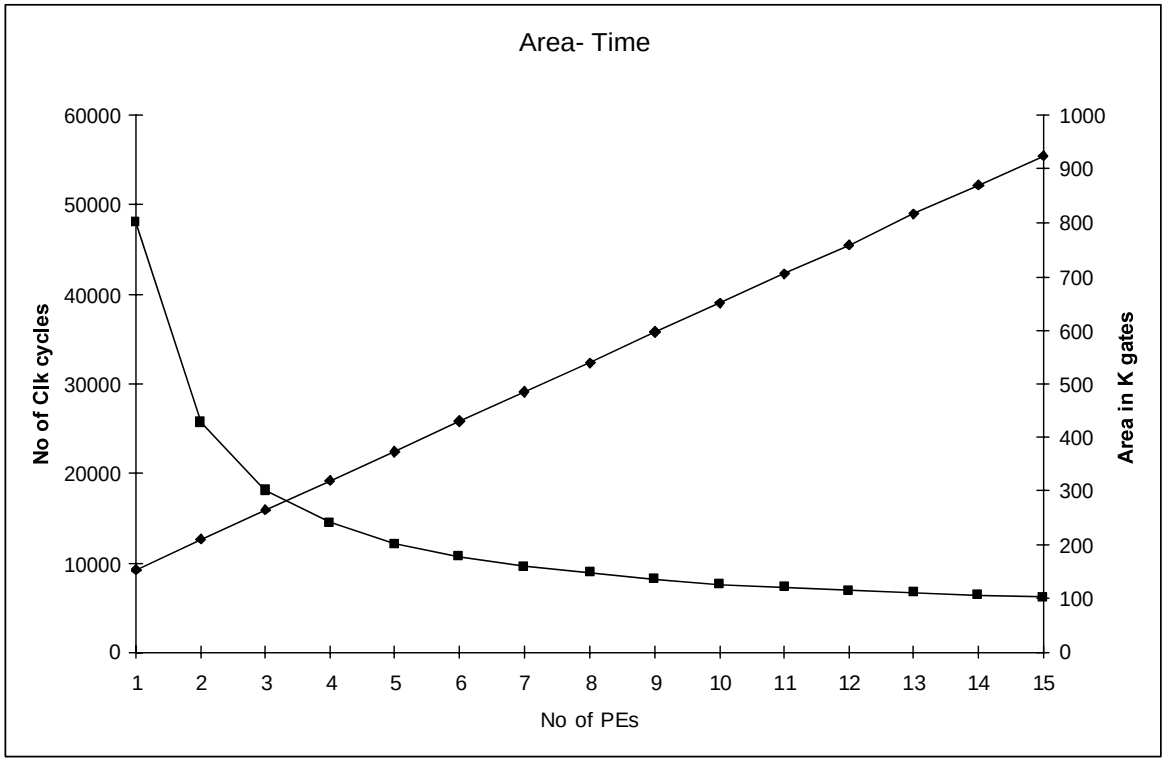

Area- Time
No of Clk cycles
Area in K gates
No of PEs
60000
50000
40000
30000
20000
10000
0
1000
900
800
700
600
500
400
300
200
100
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15

Fig. 9.

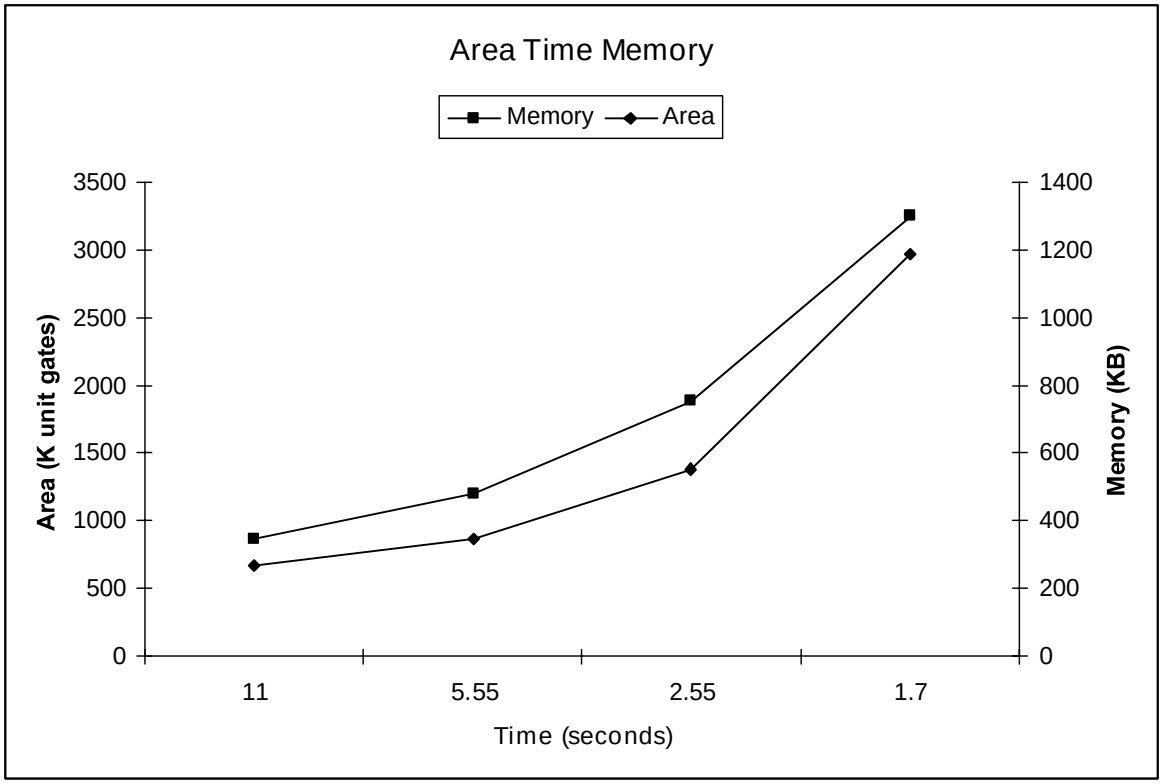
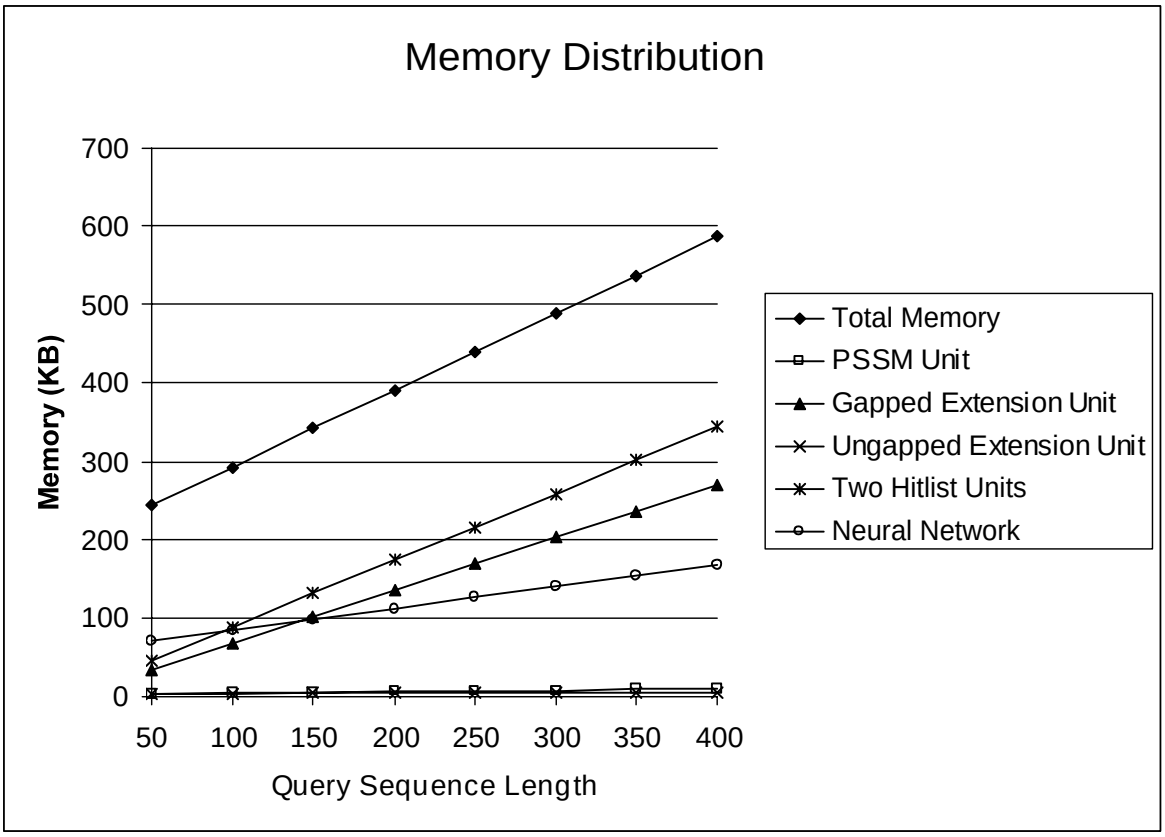

Area Time Memory
Memory
Area
3500
3000
2500
2000
1500
1000
500
0
Area (K unit gates)
1400
1200
1000
800
600
400
200
0
Memory (KB)
11
5.55
2.55
1.7
Time (seconds)

Fig. 10.


Memory Distribution
Memory (KB)
700
600
500
400
300
200
100
0
50
100
150
200
250
300
350
400
Query Sequence Length
Total Memory
PSSM Unit
Gapped Extension Unit
Ungapped Extension Unit
Two Hitlist Units
Neural Network