Hardware Acceleration for Thermodynamic Constrained DNA Code Generation

Qinru Qiu[1], Prakash Mukre[1], Morgan Bishop[2], Daniel Burns[2], and Qing Wu[1]

[1] Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902

[2] Air Force Research Laboratory, Rome Site, 26 Electronic Parkway, Rome, NY 13441

qqiu@binghamton.edu, pmukre1@binghamton.edu, Morgan.Bishop@rl.af.mil, Daniel.Burns@rl.af.mil, qwu@binghamton.edu

Abstract. Reliable DNA computing requires a large pool of oligonucleotides that do not cross-hybridize. In this paper, we present a transformed algorithm to calculate the maximum weight of the 2-stem common subsequence of two DNA oligonucleotides. The result is the key part of the Gibbs free energy of the DNA cross-hybridized duplexes based on the nearest-neighbor model. The transformed algorithm preserves the physical data locality and hence is suitable for implementation using a systolic array. A novel hybrid architecture that consists of a general purpose microprocessor and a hardware accelerator for accelerating the discovery of DNA under thermodynamic constraints is designed, implemented and tested. Experimental results show that the hardware system provides more than 250X speed-up compared to a software only implementation.

1 Introduction

A single DNA strand (i.e. oligonucleotides) is a sequence of four possible nucleotides denoted as A, C, G and T. Short DNA sequences can be synthesized easily and be used for different applications, including high density information storage [2], molecular computation of hard combinatorial problems [1], and molecular barcodes to identify individual modules in complex chemical libraries [3]. These applications rely on the specific hybridization between DNA code words and their Watson-Crick complements. The key to success in DNA computing is the availability of a large collection of DNA code word pairs that do not cross-hybridize.

The capability of hybridization between two oligonucleotides is determined by the base sequences of the hybridizing oligonucleotides, the location of potential mismatches, the concentrations of the molar strand, the temperature of the reaction and the length of the sequences [4]. The *melting temperature* (T_m) is a parameter that characterizes these factors [4]. It is defined as the temperature at which 50% of the DNA molecules have been separated to single strands. Another closely related measure of the relative stability of a DNA duplex is its Gibbs free energy, denoted as ΔG^O. The nearest-neighbor (NN) model [7][10] was proven to be an effective and accurate

estimation of the free energy. In [12], the concept of *t-stem block insertion-deletion codes* was introduced that captures the key aspects of the nearest neighbor model. In the same reference, a dynamic programming algorithm is presented to calculate the maximum weight of the t-stem common subsequence.

Search methods for DNA codes are extremely time-consuming [5], and this has limited research on DNA codeword design, especially for codes of length greater than about 12-14 bases. For example, the largest known DNA codeword library, which has been generated based on the edit distance constraint with length 16 and edit distance 10, consists of 132 pairs, and composing such codes takes several days on a cluster of 10 G5 processors, with no guarantee of optimality.

In [8], we presented a novel accelerator for the composition of reverse complement, edit distance, DNA codes of length 16. It incorporates a hardware GA, hardware edit distance calculation, and hardware exhaustive search which extends an initial codeword library by doing a final scan across the entire universe of possible code words. The proposed architecture consists of a host PC, a hardware accelerator implemented in reconfigurable logic on a *field programmable gate array* (FPGA) and a software program running in a host PC that controls and communicates with the hardware accelerator. The proposed architecture uses a modified genetic algorithm that uses a locally exhaustive, mutation-only heuristic tuned for speed. The architecture reduces the search time from 6+ days (on 10 Pentium processors) to 1.5 hours, achieving an effective 1000X speed-up, and it produces locally optimum codes.

The edit distance metric only provides a first order approximation of the free energy of binding of DNA duplexes. To improve the quality of DNA codes, more accurate metrics based on the thermodynamics of binding of DNA duplexes must be considered. This paper focuses on implementing the nearest-neighbor based free energy calculation on a reconfigurable hardware accelerator. We present a transformed algorithm to calculate the maximum weight of the 2-stem common subsequence of two DNA oligonucleotides. The result is the key part of the Gibbs free energy of the DNA cross-hybridized (CH) duplexes based on the nearest-neighbor model. The transformed algorithm preserves the physical data locality and hence is suitable for implementation using a systolic array. A new hardware accelerator for accelerating the discovery of locally optimum DNA codes with thermodynamic constraints is described. At this writing the proposed architecture provides more than 250X speed-up compared to a software only implementation.

The remainder of this paper is organized as follows: Section 2 describes the transformed algorithm and its hardware implementation using a 2D systolic array. Section 3 presents our formulation of the problem, and the solution technique in hardware GA. Section 4 provides a performance comparison between the software version and the hardware version of the codeword search. Section 5 presents final conclusions.

2 Calculation of NN Free Energy Using 2D Systolic Array

The thermodynamics of binding of nucleic acids has been widely studied and reported in the literature. The nearest-neighbor (NN) model [10] was proven to be an effective and accurate estimate of the thermodynamic binding energy. The NN model

assumes that stability of a DNA duplex depends on the identity and orientation of neighboring base pairs. There are 10 possible NN pairs: AA/TT, AT/TA, TA/AT, CA/GT, GT/CA, CT/GA, GA/CT, CG/GC, GC/CG, and GG/CC. Based on the NN model, the total free energy change of a DNA duplex at temperature T can be calculated by the following equation:

$$\Delta G^{o}{}_{T}(total) = \Delta G^{o}{}_{T,initiation} + \Delta G^{o}{}_{T,symmetry} + \sum_{i \in Watson-Crick\ \mathrm{NNs}} G^{o}{}_{T,stack}(i) + \Delta G^{o}{}_{T,AT\ \mathrm{Terminal}} \quad \textbf{(1)}$$

where $\Delta G^{o}{}_{T,initiation}$ is the initiation energy, $\Delta G^{o}{}_{T,symmetry}$ is a parameter that reflects whether the duplex is self-complementary, $\Delta G^{o}{}_{T,AT\ \mathrm{Terminal}}$ is a parameter that accounts for the differences between duplexes with terminal AT versus terminal GC, and $\Delta G^{o}{}_{T,stack}(i)$ gives the thermodynamic energy of Watson-Crick NN duplex i, which is determined by the structure of the primary sequence of the DNA duplex. This work focuses on accelerating the calculation of NN free energy using reconfigurable hardware and applies it to hardware based DNA code word search.

We developed a dynamic programming algorithm to calculate the NN free energy based on the technique presented in [12]. Given a CH duplex $x:y'$, where y' is the Watson-Crick complement of y, we define 3 matrices. They include a *suffix matrix* (***s***) which stores the longest common suffix between x and y, a *weighted suffix matrix* (***ws***) which stores the accumulated weight of each common stem-2 and an energy matrix (***e***) which stores the accumulated free energy of the possible NNs. The value of the ijth entry of these matrices can be calculated using the following equations.

$$s_{ij} = \begin{cases} s_{i-1,j-1} + 1 & \text{if } x[i] = y[j] \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(2)}$$

$$ws_{i,j} = \begin{cases} ws_{i-1,j-1} + w(x[i-1], x[i]) & \text{if } x[i] = y[j] \,\&\, x[i-1] = y[i-1] \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(3)}$$

$$e_{ij} = \begin{cases} \max(ws_{i,j} - ws_{i-1,j-1} + e_{i-2,j-2}, ws_{i,j} - ws_{i-2,j-2} + e_{i-3,j-3}, \\ \quad \ldots\ldots, ws_{i,j} - ws_{i-s_{ij},j-s_{ij}} + e_{i-3,j-3}, \ e_{i,j-1}, e_{i-1,j}) & \text{if } x[i] = y[j] \\ \max(e_{i-1,j-1}, e_{i,j-1}, e_{i-1,j}) & \text{otherwise} \end{cases} \quad \textbf{(4)}$$

The parameter $w(a[i\text{-}1],a[i])$ is the stack-pair free energy between nearest-neighbor base pairs $a[i\text{-}1]$ and $a[i]$. The bottom right entry of the ***e*** matrix gives the NN free energy of $x:y'$.

Systolic array processing has been widely used in parallel computing to enhance computational performance. The general systolic architecture has $N \times N$ connected processors, as shown in Figure 1 (b). Each processor performs an elementary calculation. The processor $P(i,j)$ reads data from its up stream neighbors $P(i\text{-}1,j)$, $P(i,j\text{-}1)$ and $P(i\text{-}1, j\text{-}1)$, and propagates the results to its down stream neighbors $P(i+1,j)$, $P(i,j+1)$

and $P(i+1, j+1)$. After an initialization, or latency period that fills the pipeline, the array generates one result per 2 clock periods.

Equations (2)~(4) cannot be directly mapped to a 2D systolic array architecture because to calculate e_{ij} we need the value of $ws_{i-d,j-d}$ ($e_{i-d,j-d}$), $1 \le d \le s_{ij}$. The variable e_{ij} is calculated by processor $P(i,j)$. The variables $ws_{i-d,j-d}$ and $e_{i-d,j-d}$ are calculated by processor $P(i-d, j-d)$. If the calculation of e_{ij} is performed at clock period t, then the calculations of $ws_{i-d,j-d}$ and $e_{i-d,j-d}$ for the same DNA duplex are performed at clock period $t-2d$. Because cells in the systolic array will register the new input and update their results every 2 clock periods, it is not possible for us to access the values of $ws_{i-d,j-d}$ and $e_{i-d,j-d}$ at clock period t if d is greater than 1. One way to handle this problem is to store the values of $ws_{i-d,j-d}$ and $e_{i-d,j-d}$ in memory or in registers. Because the maximum value of s_{ij} can be as high as the length of the DNA strand, which in our case is 16, this solution would require duplication of each cell in the systolic array 16 times. This is not practical as it significantly increases the hardware cost.

In this work, we use function transformation to simplify the hardware design. We define a *minimum weighted suffix matrix* (***min_ws***) which stores the minimum value of the difference between $ws_{i-d,j-d}$ and $e_{i-d-1,j-d-1}$, where $1 \le d \le s_{ij}$. The ijth entry of ***min_ws*** can be calculated as

$$min\_ws_{ij} = \begin{cases} \min(min\_ws_{i-1,j-1}, ws_{ij} - e_{i-1,j-1}) & \text{if } x[i] = y[j] \\ 1{,}000{,}000 & \text{otherwise} \end{cases} \quad (5)$$

when $x[i] \neq y[j]$, min_ws_{ij} will be set to an extremely large number, otherwise, it is the minimum between $min\_ws_{i-1,j-1}$ and ws_{ij}-$e_{i-1,j-1}$. The calculation of e_{ij} and ws_{ij} is transformed into the following equations.

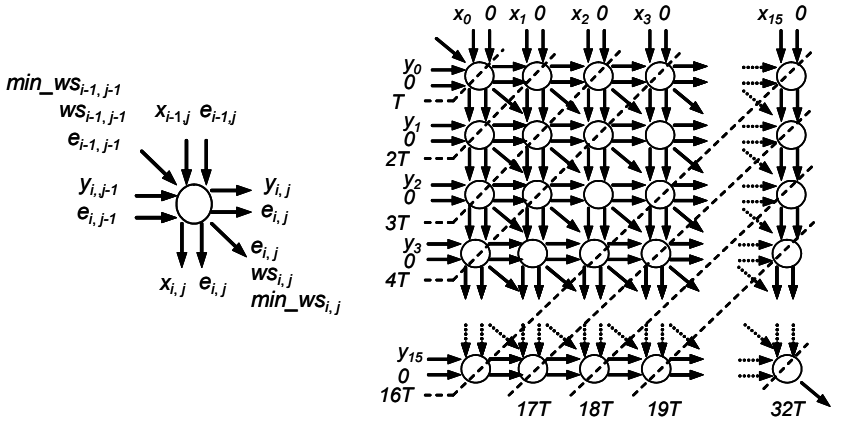
$$ws_{i,j} = \begin{cases} ws_{i-1,j-1} + w(x[i-1], x[i]) & \text{if } x[i] = y[j] \,\&\, min\_ws_{i-1,j-1} \neq 1{,}000{,}0 \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

$$e_{ij} = \begin{cases} \max(ws_{i,j} - min\_ws_{i-1,j-1}, e_{i,j-1}, e_{i-1,j}) & \text{if } x[i] = y[j] \\ \max(e_{i-1,j-1}, e_{i,j-1}, e_{i-1,j}) & \text{otherwise} \end{cases} \quad (7)$$

Equations (5)~(7) are equivalent to equations (2)~(4), however, only information from adjacent cells is needed in the calculation, hence, they can be implemented using the systolic array architecture.

The hardware design of the 2D systolic array can be derived directly from equations (5)~(7). The systolic array is an $n \times n$ array of identical cells. Each cell in the array has 7 inputs, among which the inputs $e_{i-1,j}$ and x[i-1, j] are from the cell that is located above, the inputs $e_{i,j-1}$ and $y[i, j-1]$ are from the cell that is located to the left, and the inputs $e_{i-1,j-1}$, $ws_{i-1,j-1}$ and $min\_ws_{i-1,j-1}$ are from the cell that is located to the upper left. Each cell performs the computations that are described in equations (5)~(7). For cell (i,j), the outputs $x_{i,j}$ and $y_{i,j}$ are equal to the inputs $x_{i-1,j}$ and $y_{i,j-1}$. Figure 1 (a) gives the structure of each cell, including its input/output connections and the computation implemented. The variables $x_{i,j}$ and $y_{i,j}$ are represented as 2 bit binary numbers with A=00, C=01, G=10, and T=11. The variables $e_{i,j}$, $ws_{i,j}$ and $min\_ws_{i,j}$ are represented as 14 bit signed integer numbers.

The overall architecture of the 2D systolic array as well as the data dependency and timing information are shown in Figure 1 (b). In order to prevent ripple through operation, the cells in the even columns and even rows or odd columns and odd rows are synchronous to each other and perform computations in the same clock period. The rest of the cells are also synchronous to each other but perform the computation in the next clock period. Streams of operands enter a set of shift registers along the edges of the array that synchronize the presentation of bases in the operands with the results of calculations that propagate through the array diagonally.



(a) Cell architecture (b) 2D systolic array

Fig. 1. 2D systolic array for maximum weighted 2-stem common subsequence

3 Problem Formulation and Solution Technique

We consider each DNA codeword as a sequence of length n in which each symbol is an element of an alphabet of 4 elements. Let $G(x:y)$ denote the nearest neighbor free energy of duplex $x:y'$. In this work, we focus on searching for a set of DNA codeword pairs S, where S consists of a set of DNA strands of length n and their re-

verse complement strands e.g. $\{(s_1, s_1'), (s_2, s_2'), \ldots\}$, where (s_1, s_1') denotes a strand and its Watson-Crick complement. The problem can be formulated as the following constrained optimization problem:

$$\max |S| \tag{8}$$

such that

$$g - range \le \max\left(G(s_1 : s_1'), G(s_1' : s_1)\right) \le g, \tag{9}$$

$$g - range \le \max_{s_2 \in S, s_2 \ne s_1} \left(G(s_1 : s_2), G(s_1 : s_2'), G(s_1' : s_2), G(s_1' : s_2')\right) \le g \tag{10}$$

where g and *range* are user defined threshold called *CH upper bound* and *CH range*. Equation (8) indicates that our objective is to maximize the size of the DNA codeword library. Constraints (9)~(10) specify that the NN free energy of any CH duplexes must be lower than or equal to g but greater than or equal to g-*range*. The range was initially introduced because we thought that adding the code words that are too far away from the rest of the library would restrict future growth of the library. Therefore, we only add code words that are "just good enough". Later in the experiments we found that the range has little impact on library size, however, it has a significant impact on the convergence speed of the GA.

The optimization problem is solved using a genetic algorithm. A genetic algorithm (GA) is a stochastic search technique based on the mechanism of natural selection and recombination. Potential solutions, which are also called *individuals*, are evolved from generation to generation, with *selection*, *mating*, and *mutation* operators that provide an effective combination for exploring the global search space.

Given a codeword library S, the fitness of each individual d reflects how well the corresponding codeword fits into the current codeword library. Two values define the fitness, the *reject_num* and *max_match*. The *reject_num* is the number of codeword*s* in the library which do not satisfy the condition (9)~(10) and

$$\max\_match = \max_{s_2 \in S, s_2 \ne s_1} \left(G(s_1 : s_2), G(s_1 : s_2'), G(s_1' : s_2), G(s_1' : s_2')\right).$$

A traditional GA mutation function might randomly pick an individual in the population, randomly pick a pair of bits in the individual representing one of its 16 bases, and randomly change the base to one of the 3 other bases in the set of 4 possible bases. In the proposed algorithm, however, we randomly select an individual, but then exhaustively check all of the 48 possible base changes. This is an attempt to speed beneficial evolution of the population by minimizing the overhead that would be associated with randomly picking this individual again and again in order to test those mutations. We also specify that if none of the 48 mutations were beneficial, a random individual will be generated to replace the individual. For more details about the genetic algorithm and its hardware implementation, refer to [8]. In this work, we extend the architecture of the hardware GA presented in [8] to incorporate the consideration of nearest-neighbor free energy. The 2D systolic array that is presented in section 4 is used as the fitness evaluation module and the main state machine controller of the GA is modified so that it checks constraints (9)~(10).

4 Experimental Results and Discussions

A hardware accelerator that uses a stochastic GA to build DNA codeword libraries of codeword length 16 has been designed, implemented, and tested. The design was implemented on the reconfigurable computing platform that is composed of a desktop computer and an Annapolis WildStar–Pro FPGA board [9]. The FPGA board is plugged into the PCI-X slot of the host system. The WildStar-Pro uses one XC2VP70 FPGA that has 74,448 programmable logic cells. The hardware accelerator uses about 80% of the logic resources. It runs at a 45 MHz clock frequency. A hardware based code extender that uses exhaustive search to complete the codeword library generated from GA was also designed and implemented. All the code word libraries that have been found are verified using the online tool SynDCode[11]. Since the GA is a stochastic algorithm, all results reported are the average of 5 runs.

The first set of experiments compares the performance of the hardware-based and the software-only DNA codeword search. Two versions of each search algorithm were implemented. They are denoted as "deterministic search" (DS) and "randomized search" (RS). A population size of 16 was used for both versions. The population for DS was initialized using 16 sequential integer values from 0x000003F0 to 0x000003FF, which correspond to DNA codewords 3'AATTTAAAAAAAAAAA'5 through 3'TTTTTAAAAAAAAAAA'5, while the population for RS was initialized randomly. When a new codeword is found, or when none of the mutated codewords has lower fitness than the original individual, a new individual is generated to replace the original one. In DS, a counter is used to generate the new individual. The counter is initialized to 0x000006D6. In RS, the new individual is generated randomly. We found that random search is more effective than the deterministic search. However, in order to compare the speed of hardware-based implementation and software-based implementation, we must ensure that the two systems perform exactly the same computation tasks. This is achievable only with a deterministic algorithm. All experiments were run with g = 8.5 and *range* = 1.0, and were terminated after 300 code word pairs were found.

Figure 2 shows the time required to build large thermodynamically constrained DNA code word libraries, for software running on a single processor workstation, and for the hardware accelerator. The lower curves indicate faster speed. As we can see, the software-based deterministic search has the lowest performance, while the hardware-based random search has the highest performance. The hardware-based deterministic search provides approximately 240X speed-up compared to the software-only version while the hardware-based random search provides approximately 260X speed-up compared to the software-only version. Compared to deterministic search, random search provides approximately 3.7X and 4X speed-ups using software-only and hardware-based implementations respectively. The plot also shows that the curves for software-only implementation and the hardware-based implementation are almost parallel to each other, which indicates that they both have the same complexity. Therefore, the performance gain that has been achieved by using hardware acceleration is a constant ratio.

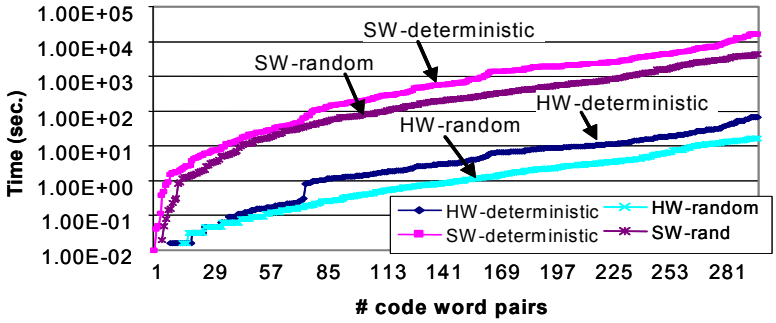
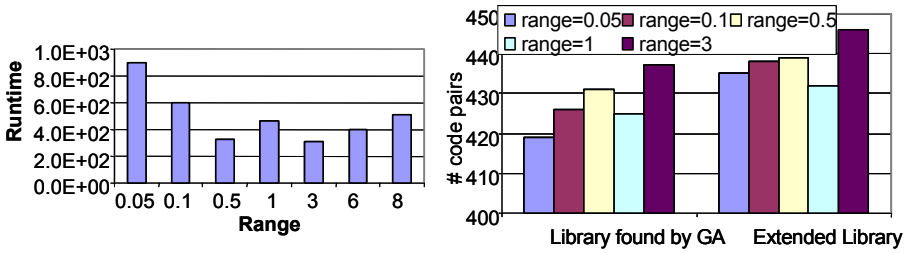


Fig. 2. Comparison between hardware-based and software-based implementation

The second set of experiments evaluates the impact of CH range on the speed and quality of the code word search. Figure 3 (a) gives the time to find 400 code word pairs for different CH ranges. In the next experiment, we ran the GA until it converged (i.e. could not find any new code words for 10 minutes), and then used exhaustive search to complete the codeword library. Figure 3 (b) shows the size of the final library. As we can see, the GA converges faster when the range is set to an appropriate value. For example, compared to range = 0.5, the runtime of GA is 26% and 24% longer at range= 0.05 and 3.0 respectively. Contrary to our original belief, the distance range does not have significant impact on library size. The size of final locally optimum libraries found with the addition of ES differ by only 3%. Exhaustive search usually finishes within 2 hours, depending on the number of words not found by GA.

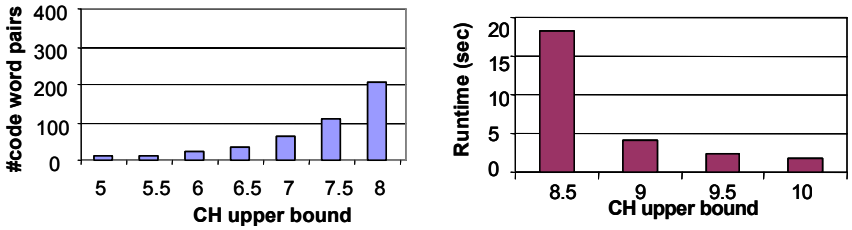


(a) Time to find 400 code word pairs (b) Library size under different range

Fig. 3. Impact of different ranges on the search speed and library size

The third set of experiments compared the search speed for different CH upper bounds (g). We varied the CH upper bound from 6.5 to 10.0 and ran GA-based code word search. Figure 4 (a) shows the number of code word pairs found in 5 minutes for CH upper bounds from 5 to 8.0 while Figure 4 (b) shows the runtime required to find 300 code word pairs for CH upper bound from 8.5 to 10. The results indicate

that the time to find 300 code words increases exponentially as CH upper bound increases.



(a) # code word pairs found in 5 minutes. (b) Time to find 300 code word pairs.

Fig. 4. Code word search under different CH upper bound

The significance of the hardware accelerator is that for the first time it enables us to evaluate different code word search algorithms and explore the lower bound of optimal code word library size in a reasonable amount of time. For example, without the hardware accelerator, each experiment in our second set would have taken take more than 20 days.

While it is true that the hardware accelerator does not explicitly consider constraints preventing bulges or internal loops, the free energy metric checking in a 2D systolic does impose those constraints implicitly by covering all sliding of the mers against each other. We believe that it should be possible to extend this work to include other secondary constraints commonly used in DNA code design, such as CG content, disallowing specific sequences, and checking all concatenations of two library words against each other (i.e. 32 mers vs 32 mers) in future hardware versions. Interestingly, scaling up to 32 mer x 32 mer checking may or may not result in longer checking times. The challenge of using hardware to calculate the free energy of DNA codewords of length 32 is that it may require more programmable hardware resources than any present single chip FPGA can provide. Possible solutions are to implement a large systolic array using multiple connected FPGAs and perform all computations in parallel, or implement a small systolic array on one FPGA and time-multiplex the computation, or await larger future generation FGPAs. While the first two solutions are feasible today, compared to the first solution, the second solution has lower cost but also lower performance. Careful tradeoff decisions must be made based on the available resources, and the given cost and performance requirements. It is also noted that DNA code design problem is only slightly different than the tag-antitag and probe set design problems faced in composing diagnostic micro arrays, where mers of length 25-60 must be checked in many alignments against longer mers drawn from large and potentially multiple genomes. Hardware accelerators similar to our own should be adaptable to that problem. Finally, DNA codes designed in-silico for both problems must be checked by fabrication and wet chemistry experiments run under use conditions to verify their true utility

5 Conclusions

In this work, we propose a novel systolic array architecture to calculate the nearest-neighbor free energy of DNA duplexes that is based on a transformed version of a dynamic programming approach. A single chip FPGA hardware accelerator has been developed that builds large, locally optimum libraries of DNA codewords with GA and exhaustive search, both based on thermodynamic energy constraints. The present version, run at 45 MHz clock frequency, provided more than a 250X speedup over a software only approach running on a 2.5 GHz Pentium processor.

6 References

[1] L. M. Adleman.: Molecular Computation of Solutions to Combinatorial Problems. Science. 266, 1021--1024 (1994).

[2] M. Mansuripur, P.K. Khulbe, S.M. Kuebler, J.W. Perry, M.S. Giridhar, and N. Peyghambarian.: Information Storage and Retrieval using Macromolecules as Storage Media. In: Optical Data Storage (2003).

[3] S. Brenner and R. A. Lerner.: Encoded Combinatorial Chemistry. In: Natl. Acad. Sci. USA, 89, pp. 5381--5383, (1992).

[4] R. Deaton and M. Garzon.: Thermodynamic Constraints on DNA-based Computing. Computing with Bio-Molecules: Theory and Experiments. Springer-Verlag.

[5] A. Brenneman and A. Condon.: Strand Design for Biomolecular Computation. Theoretical Computer Science, 287, 39--58 (2002).

[6] F. Tanaka, A. Kameda, M. Yamamoto, and A. Ohuchi.: Design of Nucleic Acid Sequences for DNA Computing based on a Thermodynamic Approach. Nucleic Acids Research. 33(3), 903--911 (2005).

[7] J. Santalucia.: A Unified View of polymer, dumbbell, and oligonucleotide DNA nearest neighbor thermodynamics. In: Natl. Acad. Sci., Biochemistry, pp. 1460--1465 (1998).

[8] Qinru Qiu, D. Burns, Q. Wu and Prakash Mukre.: Hybrid Architecture for Accelerating DNA Codeword Library Searching. In: IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (2007).

[9] Annapolis Micro System, http://www.annapmicro.com/

[10] J. SantaLucia, Jr. and D. Hicks.: The thermodynamics of DNA Structural Motifs. Annu. Rev. Biophys. Biomol. Struct. 33:415--440 (2004).

[11] M. A. Bishop, A. J. Macula1, T. E. Renz.: SynDCode: Cooperative DNA Code Generating Tool. In: 3rd Annual Conference of Foundations of Nanoscience (2006).

[12] A.G. D'yachkov, A.J. Macula, W.K. Pogozelski, T.E. Renz, V.V. Rykov, and D.C. Torney.: A Weighted Insertion-Deletion Stacked Pair Thermodynamic Metric for DNA Codes. In: Lecture Notes in Computer Science, vol 3384, pp. 90--103, Springer Berlin/Heidelber (2005).