Hardware Acceleration for Thermodynamic Constrained DNA Code Generation

Qinru Qiu*, Prakash Mukre*, Morgan Bishop**, Daniel Burns**, Qing Wu*
*Department of Electrical and Computer Engineering, Binghamton University, Binghamton, NY 13902
**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 produce 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 crosshybridized duplexes based on the nearest-neighbor model. The transformed algorithm preserves the physical data locality and hence is suitable to be implemented using systolic array. A novel hybrid architecture that 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 barcode to identify individual modules in complex chemical libraries [3]. These applications rely on the specific hybridization between DNA code word and its Watson-Crick complement. The key to success in DNA computing is the availability of a large collection of DNA code word pairs that do not crosshybridize.

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 strand. 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 [8][12] was proven to be effective and accurate estimation for the free energy. In [14], 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 was generated based on the edit distance constraint with length 16 and edit distance 10, consist of 132 pairs, composing such codes can take several days on a cluster of 10 G5 processors.

In [9], 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 extend 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 using a modified genetic algorithm that uses a locally exhaustive, mutation-only heuristic tuned for speed. The proposed

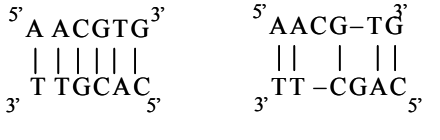
architecture successfully reduced the DNA codeword search time from 6 days (on 10 Pentium processors) to 1.5 hours and achieved an effective 1000X speed-up.

The edit distance metric only provides the first order approximation of the free energy of the DNA duplexes. To improve the quality of the code words, more accurate metric based on the thermodynamics of DNA duplexes must be considered. This paper focuses on implementing the nearest-neighbor based free energy calculation on the 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 crosshybridized (CH) duplexes based on the nearest-neighbor model. The transformed algorithm preserves the physical data locality and hence is suitable to be implemented using systolic array. A new hardware accelerator for accelerating the discovery of DNA under thermodynamic constraints is further presented. 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 provides the necessary biological background and the nearest-neighbor model for free energy calculation. Section 3 introduces the weighted t-stem block insertion-deletion codes. Section 4 gives the transformed algorithm and its hardware implementation using 2D systolic array. Section 5 presents our problem formulation and the solution technique in hardware GA. Section 6 provides a performance comparison between the software version and the hardware version of the codeword search. Section 7 presents final conclusions.

2. BACKGROUND

The DNA molecule is a nucleic acid. It consists of two oriented oligonucleotide sequences. One end of it is denoted as 3' and the other as 5'. There are four types of bases, denoted briefly as A, T, C, and G. Each base can pair up with only one particular base through hydrogen bonds: A+T, T+A, C+G and G+C. Sometimes we say that A and T are complementary to each other while C and G are complementary to each other. A *Watson-Crick complement* of a DNA sequence is another DNA sequence which replaces all the A with T or vise versa and replaces all the T with A or vise versa, and also switches the 5' and 3' ends. A DNA sequence binds most stably with its Watson-Crick complement and the structure they form is called *Watson-Crick* (*WC*) *duplex*. Figure 1 (a) shows an example of a WC duplex. We refer to the non-WC duplex as *crosshybridized* (*CH*) *duplex*. Figure 1 (b) shows an example of a CH duplex. Only WC duplexes are needed during DNA computing. Therefore, it is important to design the DNA codes such that a fixed temperature can be found that is well above the melting point of all CH duplexes and well below the melting point of all WC duplexes that can form from strands in the code.



(a) WC duplex (b) CH duplex

Figure 1 WC duplex and CH duplex

The thermodynamics of binding of nucleic acids has been widely studied and reported in the literature. The nearest-neighbor (NN) model [12] was proven to be effective and accurate for the thermodynamic energy estimation. 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:

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

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, $\Delta G^{o}{}_{T,stack}(i)$ gives the thermodynamic energy of Watson-Crick NN duplex i. Their values at 37°C are given in Table 1.

Example: Using Table 1, the NN free energy of DNA duplex $\begin{matrix} 5'-CGTTGA-3' \\ 3'-GCAACT-5' \end{matrix}$ can be calculated as:

$\sum_{i\in \text{WC NNs}} \Delta G^{o}{}_{T,stack}(i) = \Delta G_{37,stack}(CG) + \Delta G_{37,stack}(GT) + \Delta G_{37,stack}(TT) + \Delta G_{37,stack}(TG) + \Delta G_{37,stack}(GA)$

$= -2.17 - 1.44 - 1.00 - 1.45 - 1.3 = -5.35 \text{ kcal mol}^{-1}$.

Table 1 Nearest-neighbor thermodynamic parameters for DNA Watson-Crick pairs at 37^{O}C [12]

	A	C	G	T
A	-1	-1.44	-1.28	-0.88
C	-1.45	-1.84	-2.17	-1.28
G	-1.3	-2.24	-1.84	-1.44
T	-0.58	-1.3	-1.45	-1.00

While the parameters $\Delta G^{o}{}_{T,initiation}$, $\Delta G^{o}{}_{T,symmetry}$, and $\Delta G^{o}{}_{T,AT\text{ Terminal}}$ can be obtained in a straightforward manner, the NN free energy (i.e. $\sum_{i\in WC\text{ NN}} \Delta G^{o}{}_{T,stack}(i)$) 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.

3. T-STEM BLOCK INSERTION-DELETION CODES

In the rest of the paper, we adopt some notations that are used in reference [14]. We use [*n*] to denote the set {0, …, *n*-1} and (*n*) to denote the sequence 1, 2, …, *n*. We call $\alpha \prec (n)$ a string if and only if it is a subsequence of consecutive integers, e.g., $\alpha = i, i+1,\ldots,i+k$. Let $[q]^n$ denote the set of sequences of length *n* with entries in [*q*]. For $x = x_1, \ldots, x_n$ with $x \in [q]^n$ and $\sigma = i_1, i_2, ..., i_k$ where $\sigma \prec (n)$, we use $x_\sigma \prec x$ to denote the subsequence $x_{i_1}, x_{i_2}, ..., x_{i_k}$ and *x*[*i*] to denote the *i*th entry in sequence *x*. Given a non-negative real-valued function, Ω, on [*q*], we define the weight of subsequence x_σ as $\|x_\sigma\|_\Omega = \sum_{i\in\sigma} \Omega(x_i)$.

For $\sigma \prec (n)$, a substring $\beta \prec \sigma$ is called a block of σ if β is not subsequence of any substring α of σ with $\beta \neq \alpha$. Denote σ as a sequence of blocks $\beta_1, \beta_2, ..., \beta_i, ..., \beta_l$, if the difference between the endpoint of β_i and the starting point of β_{i+1} is greater than or equal to *t*, then σ is a *t-gap sequence* of (*n*). It is denoted as $\sigma \in G_t(n)$. Given $\sigma \prec (n)$, $\tau \prec (m)$ with $|\sigma| = |\tau|$ and $\sigma \in G_t(n), \tau \in G_t(m)$, let $f : \sigma \to \tau$ be a unique mapping, σ and τ are said to be *t-gap block isomorphic* (denoted as $\sigma \underset{t}{\cong} \tau$) if $\alpha \prec \sigma$ is a string $\Leftrightarrow f(\alpha) \prec \tau$ is a string. For $x \in [q]^n$ and $y \in [q]^m$, if $x_\sigma = y_\tau$ and $\sigma \underset{t}{\cong} \tau$, then we say x_σ and y_τ are t-gap block isomorphic and denote it as $x_\sigma \underset{t}{\cong} y_\tau$.

Definition 1 For $2 \le t \le n-1$ and $x, y \in [q]^n$, we define the *weight of the longest common t-gap block subsequence* of *x* and *y* as $\phi^t_{\Omega,q}(x,y) \equiv \max\{\|x_\sigma\|_\Omega : x_\sigma \underset{t}{\cong} y_\tau\}$. The *weighted distance* of two t-gap block insertion-deletion codes is defined as $\Phi^t_{\Omega,q}(x,y) \equiv \min(\|x\|_\Omega, \|y\|_\Omega) - \phi^t_{\Omega,q}(x,y)$. When *t* = 1 and $\|x_\sigma\|_\Omega = |x_\sigma|$, $\Phi^1_{\Omega,q}(x,y)$ is the Levenshtein insertion-deletion distance.

The weight of the longest common t-gap block subsequence of *x* and *y* (i.e. $\phi^t_{\Omega,q}(x,y)$) can be calculated using dynamic programming. For $x, y \in [q]^n$ and $t \le i, j \le n$, let $M^t_{\Omega,i,j}$ denote $\phi^t_\Omega(x_{[1,i]}, y_{[1,j]})$ and $suf(x,y)$ denote the length of the longest common suffix between *x* and *y*. It is proved ([14]) that:

$$M^t_{\Omega,i,j} = \max(M^t_{\Omega,i-1,j}, M^t_{\Omega,i,j-1}, D^t_{\Omega,i,j}\}, \quad (1)$$

where $D^t_{\Omega,i,j}$ is defined as

$$D^t_{\Omega,i,j} = \begin{cases} \max\left\{\left\|x_{[i-r+1,i]}\right\|_\Omega + M^t_{\Omega,i-r-t+1,j-r-t+1} : 1 \le r \le suf(x_{[1,i]}, y_{[1,j]})\right\} & if\ suf(x_{[1,i]}, y_{[1,j]}) \ge 1 \\ 0 & \text{otherwise} \end{cases}. \quad (2)$$

Given two sequences $x, y \in [q]^n$, $x_\sigma = y_\tau$ with a unique mapping $f: \sigma \to \tau$, a *t-stem* exists if and only if subsequences $x_{[i,i+t-1]} = y_{[f(i),f(i)+t-1]}$. Let σ^t_τ be the sequence of the first index of all the t-stems. For $x \in [q]^n$, let $d_q(x_{[a,a+t-1]})$ be a unique number in $[q^t]$ to represent $x_a x_{a+1} ... x_{a+t-1}$, we define $x^{(t)} \in [q^t]^{n-t}$ as a sequence whose *i*th element is equal to $(d_q(x_{i,i+t-1}])$. For $2 \le t \le n-1$, it can be proved that if $\left|\sigma^t_\tau\right| \ne 0$, then $x^{(t)}_{\sigma^t_\tau} \underset{t}{\cong} y^{(t)}_{f(\sigma^t_\tau)}$.

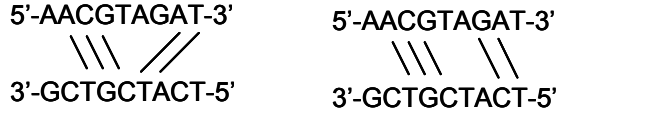
Definition 2 Let Ω be a weight function on $[q^t]$, the *maximum weighted t-stem common subsequence* is defined as $\psi^t_\Omega(x,y) = \max\left\{\left\|x^{(t)}_{\sigma^t_\tau}\right\|\right\}$. The *t-stem code distance* is defined as $\Psi^t(x,y) = \min\left(\left\|x^{(t)}\right\|_\Omega, \left\|y^{(t)}\right\|_\Omega\right) - \psi^t_\Omega(x,y)$.

It is proved ([14]) that the maximum weighted t-stem common subsequence of *x* and *y* is equal to the weight of the longest common t-gap block subsequence of $x^{(t)}$ and $y^{(t)}$, i.e. $\psi^t_\Omega(x,y) = \phi^t_{\Omega,q^t}\left(x^{(t)}, y^{(t)}\right)$.

Let the CH duplex between *x* and $\bar{y}$ be denoted as $x : \overleftarrow{\bar{y}}$, where $\bar{y}$ is the WC complement of *y* and $\overleftarrow{\bar{y}}$ is a representation of $\bar{y}$ in reversed order. If the duplex $x : \overleftarrow{\bar{y}}$ have a secondary structure, then its free energy of nearest neighbor stacked pairs can be calculated as $\psi^2_\Omega(x,y)$, where the weight function Ω is equal to $-\Delta G^o_{T,stack}$.

Example: Consider the CH duplex $\begin{matrix} 5'-AACGTAGAT-3' \\ 3'-GCTGCTACT-5' \end{matrix}$. It corresponds to strings *x* = *AACGTAGAT* and *y* = *CGACGATGA*. Because $x_{[2,4][8,9]} = y_{[3,5][6,7]}$, we have $\sigma = [2,4][8,9]$, $\tau = [3,5][6,7]$, $\sigma^2_\tau = 2,3,8$, and $\left\|x^{(2)}_\sigma\right\| = \left\|x^{(2)}_{2,3,8}\right\| = -\Delta G_{37,stack}(AC) - \Delta G_{37,stack}(CG) - \Delta G_{37,stack}(AT) = 4.49\ kcal/mol$.

Let *A*, *C*, *G*, *T* be encoded as 0, 1, 2, and 3, then *x* = 0, 0, 1, 2, 3, 0, 2, 0, 3 and *y* = 1, 2, 0, 1, 2, 0, 3, 2, 0. $x^{(2)}$ = 0, 1, 6, 11, 12, 2, 8, 3 and $y^{(2)}$ = 6, 8, 1, 6, 8, 3, 14, 8. It is easy to see that $x^{(2)}_{2,3,8} = y^{(2)}_{3,4,6}$. Let $\sigma = 2,3,8$ and $\tau = 3,4,6$, because any string in σ corresponds to a string in τ, and the gaps between blocks in σ and τ are equal to 2, we say $\sigma \underset{2}{\cong} \tau$ and $x^{(2)}_\sigma \underset{2}{\cong} y^{(2)}_\tau$. Note that although $x^{(2)}_{2,3,7,8} = y^{(2)}_{3,4,5,6}$, because a string



(a) Secondary structure corresponding to $\sigma^2_\tau = 2,3,8$

(b) Secondary structure corresponding to $\sigma^2_\tau = 2,3,7$

Figure 2 Represent DNA secondary structure using 2-stem insertion-deletion codes

in $\tau = 3,4,5,6$ does not necessarily correspond to a string in $\sigma = 2,3,7,8$, therefore, $x^{(2)}_{2,3,7,8}$ and $y^{(2)}_{3,4,5,6}$ are not t-gap block isomorphic. Using equation (1) and (2) we can find that $\psi^2_\Omega(x,y) = \phi^t_\Omega(x^{(2)}, y^{(2)}) = \left\| x^{(2)}_{2,3,7} \right\| = -\Delta G_{37,stack}(AC) - \Delta G_{37,stack}(CG) - \Delta G_{37,stack}(GA) \quad = 4.91\, kcal/mol$. Figure 2 shows the secondary structures in the CH duplex that for $\sigma^2_\tau = 2,3,8$ and $\sigma^2_\tau = 2,3,7$.

In this work, we estimate the NN free energy of a CH duplex by calculating their maximum weighted 2-stem common subsequence. In the next section, we will present a dynamic programming algorithm that is suitable for 2D systolic array implementation.

4. CALCULAT NN FREE ENERGY USING 2D SYSTOLIC ARRAY

Based on the equations (1) and (2), a dynamic programming algorithm was developed. Given a CH duplex $x : \overleftarrow{y}$, 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 *ij*th entry of these matrices can be calculated using the following equations.

$$s_{ij} = \begin{cases} s_{i-1,j-1} + 1 & \text{if } x[i] = y[j] \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

$$ws_{i,j} = \begin{cases} ws_{i-1,j-1} + w(x[i-1], x[i]) & \text{if } x[i] = y[j] \;\&\; x[i-1] = y[i-1] \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

$$e_{ij} = \begin{cases} \max(ws_{i,j} - ws_{i-1,j-1} + e_{i-2,j-2}, ws_{i,j} - ws_{i-2,j-2} + e_{i-3,j-3}, \\ \quad \ldots\ldots, ws_{i,j} - ws_{i-s_{ij},j-s_{ij}} + e_{i-3,j-3}, \; e_{i,j-1}, e_{i-1,j}) & \text{if } x[i] = y[j] \\ \max(e_{i-1,j-1}, e_{i,j-1}, e_{i-1,j}) & \text{otherwise} \end{cases}. \tag{5}$$

The parameter $w(a[i\text{-}1],a[i])$ is the stack-pair free energy specified in Table 1. The bottom right entry of the ***E*** matrix gives the NN free energy of $x : \overleftarrow{y}$.

Example: Consider $x = 5'AATGA3'$ and $\overleftarrow{y} = 3'CATGG5'$ (i.e. $y = 5'GTACC3'$,) the matrix **S**, **WS**, and **E** can be calculated as the following and the NN free energy of the CH duplex is 2.33.

$$\mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}, \; \mathbf{WS} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.88 & 0 & 0 \\ 0 & 0 & 0 & 2.33 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \; \mathbf{E} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.88 & 0.88 & 0.88 \\ 0 & 0 & 0.88 & 2.33 & 2.33 \\ 0 & 0 & 0.88 & 2.33 & 2.33 \end{bmatrix}.$$

Systolic array processing has been widely used in parallel computing to enhance the performance. Its general architecture is given in Figure 3 (b). It has $N \times N$ connected processors. Each processor performs an elementary calculation. The processor $P(i,j)$ reads data from its upper 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 the initialization period that is needed to fill the pipeline, a systolic array generates one result per 2 clock periods.

Equation (3)~(5) 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 $P(i\text{-}d, j\text{-}d)$. If the calculation of

e_{ij} is performed at clock period t, then the calculations of $ws_{i-d,j-d}$ and $e_{i-d,j-d}$ of 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 data 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 memorize the values of $ws_{i-d,j-d}$ and $e_{i-d,j-d}$ by adding extra storage elements. 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 requires us to duplicate 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 *ij*th entry of **MIN_WS** can be calculated as

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

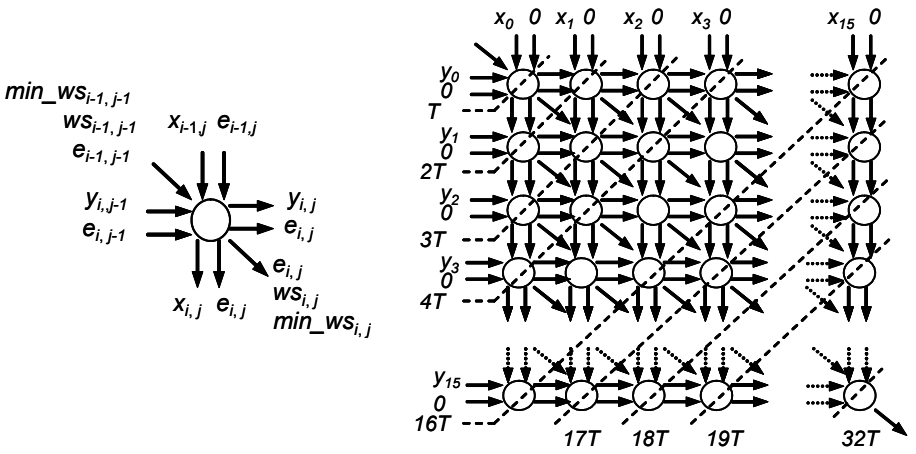
when $x[i] \neq y[j]$, min_ws_{ij} will be set to an extremely large number, otherwise, it is the minimum between $min\_ws_{i-1,j-1}$ and ws_{ij}-$e_{i-1,j-1}$. The calculation of e_{ij} and ws_{ij} is transformed into the following equations.

$$ws_{i,j} = \begin{cases} ws_{i-1,j-1} + w(x[i-1], x[i]) & \text{if } x[i] = y[j] \,\&\, min\_ws_{ij} \neq 1{,}000{,}000 \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$e_{ij} = \begin{cases} \max(ws_{i,j} - min\_ws_{i-1,j-1}, e_{i,j-1}, e_{i-1,j}) & \text{if } x[i] = y[j] \\ \max(e_{i-1,j-1}, e_{i,j-1}, e_{i-1,j}) & \text{otherwise} \end{cases}. \quad (8)$$

Equations (6)~(8) are equivalent to equations (3)~(5), 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 (6)~(8). 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 coming from the cell that is located above, the inputs $e_{i,j-1}$ and $y[i, j-1]$ are coming 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 coming from the cell that is located to the upper left. Each cell performs the computations that are described in equations (6)~(8). 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 3 (a) gives the structure of each cell, including its input/output and the computation implemented. The variable $x_{i,j}$ and $y_{i,j}$ are represented as 2 bit binary numbers with A=00, C=01, G=10, and T=11. The variable $e_{i,j}$, $ws_{i,j}$ and $min\_ws_{i,j}$ are represented as 14 bit signed integer numbers.



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

Figure 3 Calculating maximum weighted 2-stem common subsequence using 2D systolic array

The overall architecture of the 2D systolic array as well as the data dependency and timing information are shown in Figure 3 (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 the computation 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. In this way, the results propagate through the array diagonally.

5. 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 $G(x:\overleftarrow{y})$ denote the nearest neighbor free energy of duplex $x:\overleftarrow{y}$. 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 reverse complement strands e.g. $\{(s_1, \overline{s_1}), (s_2, \overline{s_2}), \ldots\}$, where $(s_1, \overline{s_1})$ denotes a strand and its Watson-Crick complement. The problem can be formulated as the following constrained optimization problem:

$$\max |S| \quad \text{such that} \tag{8}$$

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

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

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*. For any DNA duplex, the weakest stacked pair is the AT pair with $\Delta G^o_{37,stack}(AT) = 0.88$ and $\Delta G^o_{37,stack}(TA) = 0.58$. Therefore, for $y, x \prec [A,C,G,T]^{16}$, $\min(\left\|x^{(2)}\right\|_\Omega, \left\|y^{(2)}\right\|_\Omega) = 7*(0.58+0.88)+0.58 = 10.8$. Based on definition 2, the weighted t-stem distance between x and y is greater than $10.8-g$ and less than $10.8-g+range$ if $g - range \le \psi^2_\Omega(x,y) = G(x:\overleftarrow{y}) \le g$. Therefore, constraints (9)~(10) ensure that the t-stem distance between any non-WC pairs in the library is within the range [10.8-*g*+*range*, 10.8-*g*]. The range was initially introduced because we thought that adding the code words that are too far away from the rest of the library will restrict the 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 the size of the library, 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. Solutions, which are also called *individuals*, are evolved from generation to generation, with *selection*, *mating*, and *mutation* operators that provide an effective combination of exploration of 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, *reject_num* and *max_match*. The *reject_num* is the number of codewords in the library which does not satisfy the condition (9)~(10) and $\max\_match = \max_{s_2 \in S, s_2 \ne s_1}\left(G(s_1:\overleftarrow{s_2}), G(s_1:\overleftarrow{s_2}), G(\overline{s_1}:\overleftarrow{s_2}), G(\overline{s_1}:\overleftarrow{s_2})\right)$.

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 to 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 original one. For more details about the genetic algorithm and its hardware implementation, please refer to [9][11]. In this work, we extend the architecture of the hardware GA presented in [9] to incorporate the consideration of nearest-neighbor free energy. The 2D systolic array that is presented in section 4 is used as fitness evaluation module and the main state machine controller of the GA is modified so that it checks all the constraints (9)~(10).

6. EXPERIMENTAL RESULTS

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 [10]. The FPGA board is plugged into the PCI-X slot of the host system. The WildStar-Pro uses XC2VP70 FPGA that has 74,448 programmable logic cells. The hardware accelerator uses about 80% of the logic resources. It is running at 45 MHz clock frequency. A hardware based code extender that uses exhaustive search to complete the codeword library generated from GA is also designed and implemented. All the code word libraries that have been found are verified using the online tool SynDCode[13]. Since 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 search algorithms are implemented. They are denoted as "deterministic search" (DS) and "randomized search" (RS). The population size is 16. The population of the DS was initialized using 16 sequential data from 0x000003F0 to 0x000003FF, which corresponds to DNA codeword 3'AATTTAAAAAAAAAAA'5 and 3'TTTTTAAAAAAAAAAA'5, while the population of the RS was initialized randomly. When a new codeword is found, or when none of the mutated codeword has lower fitness than the original individual, a new individual will be generated to replace the original one. In the DS, a counter is used to generate the new individual. The counter is initialized to 0x000006D6. In the RS, the new individual is generated randomly. The 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 are running with g = 8.5 and *range* = 1.0. They are terminated after 300 code word pairs have been found.

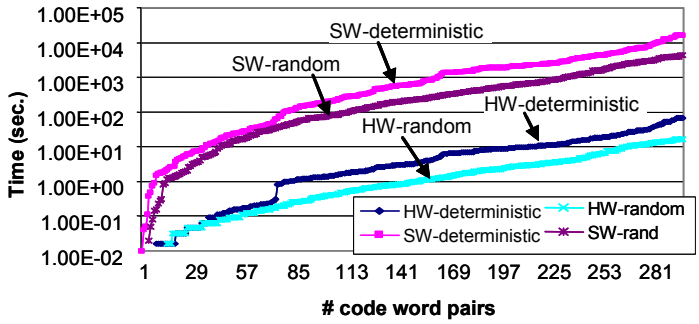
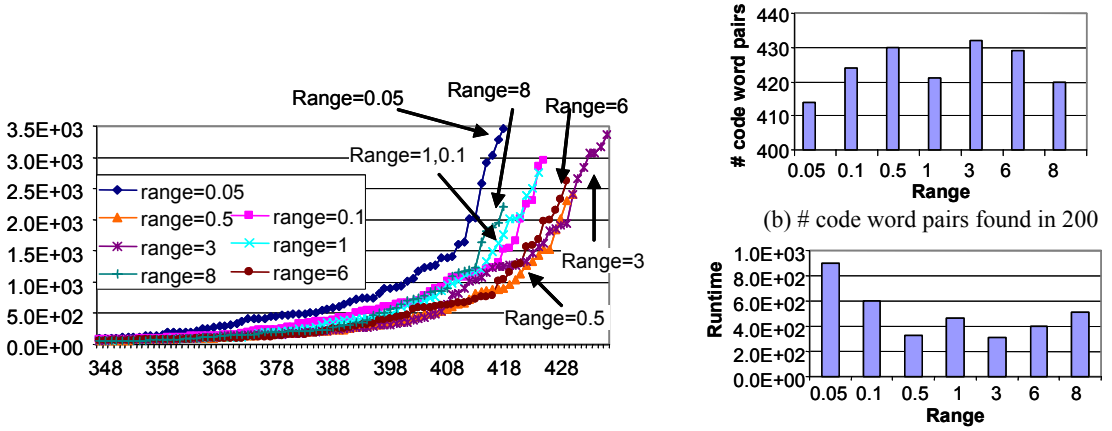


Figure 4 Comparison between hardware-based and software-based implementation

Figure 4 shows the time it takes to build a large thermodynamic constrained DNA code word library using software on a single processor workstation and the hardware accelerator. The lower curve indicates 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 the deterministic search, the random search provides approximately 3.7X and 4 X 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.

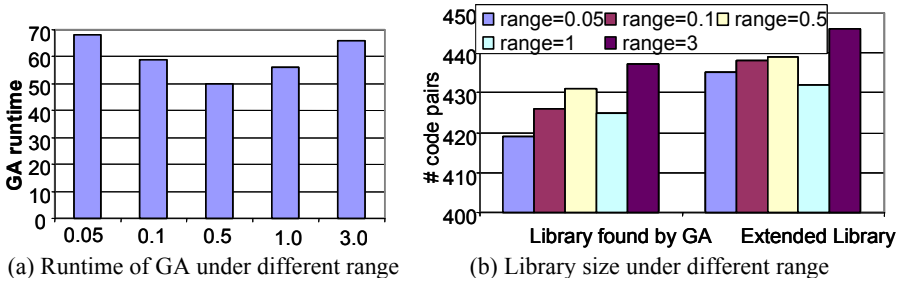


(a) Impact of *range* to the speed of the code word search

(b) # code word pairs found in 200

(c) Time to find 400 code word pairs

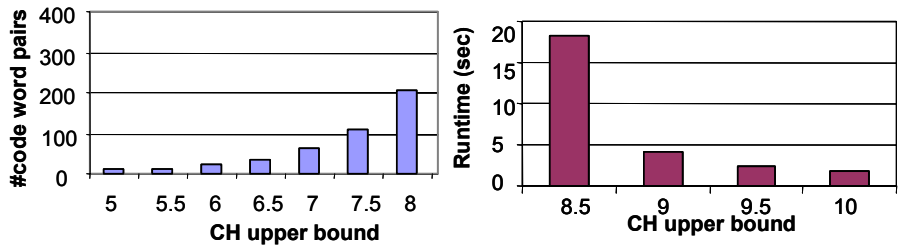
Figure 5 Impact of different ranges on the search speed



(a) Runtime of GA under different range

(b) Library size under different range

Figure 6 Impact of different ranges on the library size



(a) # code word pairs found in 5 minutes

(b) Time to find 300 code word pairs

Figure 7 Code word search under different CH upper bound

The second set of experiments evaluates the impact of CH range on the speed and quality of the code word search. We varied the *CH range* from 0.05 to 3 and run the GA based code word search with g=8.5. The search stops after it found 300 code word pairs. Figure 5 (a) shows the search time under different CH ranges. Figure 5 (b) and (c) gives the number of code word pairs found in 200 seconds and the time to find 400 code word pairs under different CH ranges. The results show that the search speed is slower when the CH range is either too large or too small. In the next experiment, we run the GA until it converges (i.e. cannot find any new code word for 10 minutes,) then we use exhaustive search to complete the codeword library. Figure 6 (a) shows the runtime of GA under different ranges and Figure 6 (b) shows the size of the library found by GA and the size of the final library. As we can see, the GA converges faster when setting the range to appropriate value. Compared to range = 0.5, the runtime of GA is 26% and 24% longer at range= 0.05 and 3.0 respectively. Opposite to our original believe, the distance range does not have significant impact on the library size. The sizes of libraries found by GA at different ranges only have 4% difference and the sizes of final libraries only have 3% difference. The exhaustive search usually finishes within 2 hours.

The third set of experiments compares the search speed for different CH upper bounds (g). We varied the CH upper bound from 6.5 to 10.0 and run the GA-based code word search. We stop the search when it

found 300 code word pairs or the run time exceeds 15 minutes. Figure 7 (a) shows the number of code word pairs found in 5 minutes for CH upper bounds from 5 to 8.0 while Figure 7 (b) shows the runtime to find 300 code word pairs for CH upper bound from 8.5 to 10. The results indicate that as the CH upper bound increases, the chances to find a code word increases exponentially.

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

7. CONCLUSIONS AND FUTURE WORK

In this work, we propose systolic array architecture to calculate the nearest-neighbor free energy of DNA duplexes. Hardware accelerator has been developed that searches for DNA codeword based on thermodynamic energy constraints. In the future, we plan to extend the current architecture to search and extend codes word of at least length 32.

8. REFERENCES

[1] L. M. Adleman, "Molecular Computation of Solutions to Combinatorial Problems," *Science*, vol. 266, pp. 1021-1024, November 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," *Proceedings of Optical Data Storage*, 2003.

[3] S. Brenner and R. A. Lerner, "Encoded Combinatorial Chemistry," *Proc. Natl. Acad. Sci. USA*, vol 89, pp5381-5383, June 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*, vol. 287, pp.39-58, 2002.

[6] S.-Y. Shin, I.-H. Lee, D. Kim, and B.-T. Zhang, "Multiobjective Evolutionary Optimization of DNA Sequences for Reliable DNA Computing", *IEEE Transactions on Evolutionary Computation*, vol. 9(20), pp.143-158, 2005.

[7] 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), pp.903-911, 2005.

[8] J. Santalucia, "A Unified View of polymer, dumbbell, and oligonucleotide DNA nearest neighbor thermodynamics", *Proc. Natl. Acad. Sci., Biochemistry*, pp. 1460-1465, February 1998.

[9] Qinru Qiu, D. Burns, Q. Wu and Prakash Mukre, "Hybrid Architecture for Accelerating DNA Codeword Library Searching*," to appear in Proc. IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, April 2007.

[10] http://www.annapmicro.com/

[11] D. Burns, K. May, T. Renz, and V. Ross, "Spiraling in on Speed-Ups of Genetic Algorithm Solvers for Coupled Non-Linear ODE System Parameterization and DNA Code Word Library Synthesis," *MAPLD International Conference*, 2005.

[12] J. SantaLucia, Jr. and D. Hicks, "The thermodynamics of DNA Structural Motifs," *Annu. Rev. Biophys. Biomol. Struct.* 33:415-40, 2004.

[13] M. A. Bishop, A. J. Macula1, T. E. Renz, "SynDCode: Cooperative DNA Code Generating Tool," *Proc. of 3rd Annual Conference of Foundations of Nanoscience*, April, 2006.

[14] 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," *Lecture Notes in Computer Science*, Vol. 3384/2005, pp. 90-103, Springer Berlin/Heidelberg