計測自動制御学会東北支部 第 282 回研究集会 (2013.07.17)
資料番号 282-10

ショートリード配列マッピングのための FPGA アクセラレータの開発

Development of an FPGA-Based Accelerator for DNA Short-Read Mapping Using Burrows-Wheeler Alignment

○張山昌論*, ハシタ ムトゥマラ ウィシディスーリヤ*, 亀山充隆*

○ Masanori Hariyama*, Hasitha Muthumala Waidyasooriya*, Michitaka Kameyama*

*東北大学大学院情報科学研究科

*Graduate School of Information Sciences, Tohoku University

キーワード： ヘテロジニアスアーキテクチャ(Heterogeneous architecture), 高性能計算 (high-performance computing), スーパーコンピューティング (supercomputing), FPGA(FPGA)

連絡先： 〒980-8579 仙台市青葉区荒巻字青葉 6-6-05
東北大学大学院情報科学研究科
張山昌論, Tel.: (022)795-7153, Fax.: (022)263-9167, E-mail: hariyama@ecei.tohoku.ac.jp

1. はじめに

次世代シーケンサーでは，数百万個以上のショートリード (DNA を分断した短い DNA 鎖) を生成することができる．そのような，膨大な数のショートリードをマッピングすることが DNA 解析の重要な基本処理となっている．ショートリードのマッピング用ツールとしては，PC クラスタ上でのソフトウェアとして，Maq [1], Bowtie [2], BWA [3] などのが存在する．ショートリードのマッピングは，図 1 に示すように，数 10 から数 100 個の塩基もつショートリードを，1 塩基だけが異なる場合 (SNP)，塩基が挿入される場合 (Insertion)，および，塩基が欠損する場合 (Deletion) といった場合を考慮して，参照配列と呼ばれる既に解析済みの配列のどの箇所に対応するかを決定する．それにより，個人毎の遺伝子の違いを特定することが可能となる．ソフトウェアツールでのショートリードのマッピングの問題は，その膨大な処理時間である．大規模な PC クラスタでの並列処理を用いたとしても数日〜1 週間程度かかる．

この問題を解決するために，FPGA(field programmable gate array) を用いたアクセラレータを提案する [4]．FPGA では回路構造を，コンフィグレーションと呼ばれるプログラムにより変更することにより，アプリケーション処理に適した専用プロセッサを構築できる．近年の FPGA は，半導体集積回路技術の進展により，数百万以上のプログラマブルロジック，DSP ユニット，内蔵メモリ，DDR2 および DDR3 などの外部メモリインタフェースを備えており，高性能な専用プロセッサ開発にも使えるようになってきた．実装するアルゴリズムとしては，ソフトウェアとして現在最も高速な手法としてしら

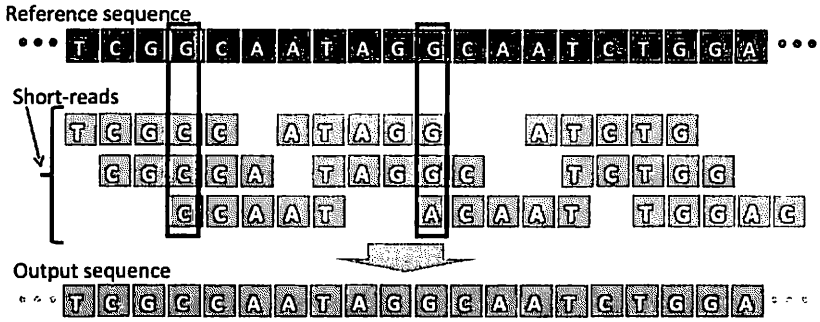


Fig. 1　ショートリードのマッピング

れる Burrows-Wheeler Alignment (BWA)[3] を用いる．BWA はショートリード毎の並列処理も可能でため，FPGA 実装にも適する．提案の FPGA ベースアクセラレータでの処理性能は，ソフトウェア実装と比較して 10 倍以上高いという結果を得た．

2. BWA アルゴリズム

簡単に BWA アルゴリズム[3] について説明する．BWA アルゴリズムは，文献[5] で提案されているマッチング方法を用いている．aW は文字列 $\{a, W\}$ を表すとする．もし文字列 W が，文字列 X の部分文字列で，かつ $k(aW) \leqq l(aW)$ であれば，文字列 aW も X の部分文字列であるという性質がある．変数 k および l は，式 (1) および (2) で与えられ，それぞれ X の suffix array (SA) インターバルの下限と上限を与える．

$$k(aW) = C(a) + O(a, k(aW) - 1) + 1 \quad (1)$$

$$l(aW) = C(a) + O(a, l(aW)) \quad (2)$$

辞書式順序において a より小さい文字の数は，$C(a)$ で与えられる．文字列 $B[0, i]$ において a の生起する数は $O(a, i)$ で与えられる．文字列 X の BWT 配列は B で与えられる．文字列 B の先頭から i 個の文字を含む部分文字列は，$B[0, i]$ で与えられる．

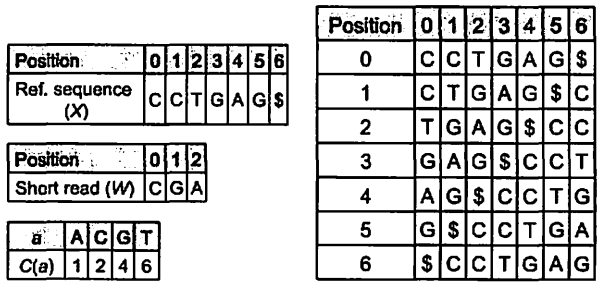
図 2 は BWA アルゴリズムを用いたショートリードのマッピングの方法[3] を示している．アルゴリズムを説明するために，図 3 に示す例を考える．図 3(a) に示すような参照配列 X と，ショートリード W を考える．マッピング処理の入力は，X の BWT，X の反転 (X') の BWT，生起配列 $O(.,.)$, $O'(.,.)$ および $C(.)$ である．配列 $C(.)$ および X の Rotation を，それぞれ，図 3(a) および 3(b) に示す．ソートした結果を図 3(c) に示す．図 3(c) に示すように，BWT B および生起配列 $O(.,.)$ を得る．同様のやり方で，X' に対しても B' および $O'(.,.)$ を得る．紙面の都合により，図 2 のアルゴリズム中の，Calculated 処理の説明は省略する，ここで，Calculated 処理は W 中のミスマッチ数の下限 $D(.)$ を与える処理である (より詳しくは文献[3] を参照)．

図 4 はショートリード W のマッピングを示している．処理 InexRecur(W, i, z, k, l) は，ミスマッチの下限 $D(i)$ の計算の後に，呼び出される．W の探索位置，許容されるミスマッチ数 (SNP，挿入，欠損の合計) , suffix array の下限，および suffix array の上限は，それぞれ，i, z, k , および l で与えられる．図 4 に，InexRecur 処理の実行の様子を示す．ラベル "(i, z, k, l)" は処理 "InexRecur(W, i, z, k, l)" を表す．マッピングが終了した後に，参照配列中のショートリードの位置，SNP，挿入および欠損 (Indel) に関する情報を得る．この例では，図 5 において SNP および Indel をミスマッチとして考慮した結果を得ている．1 個の挿入を持つ結果は SA インターバル (5,5) を持っている (図 4)．図 3(c) に

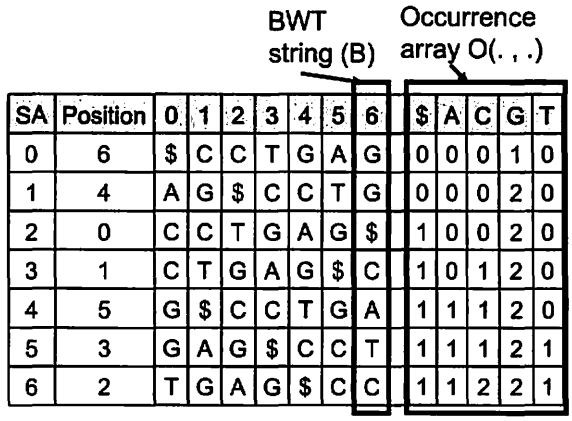
```
Input: BWT string B for reference string X
       Array C(.) and O(.,.) from B
       BWT string B' for the reverse of reference X
       Array O'(.,.) from B'
Output: Suffix-array intervals
Procedures:
Calculated(W)
begin
  | Calculate D(i) that gives a lower bound for the
  | number of mismatches in W
end
InexRecur(W,i,z,k,l)
begin
  | if z < D(i) then
  |  | return φ
  | end
  | if i < 0 then
  |  | return [k,l]
  | end
  | I = φ
  | I = I ∪ InexRecur(W,i-1,z-1,k,l)
  | for each b ∈ {A,C,G,T} do
  |  | k_b = C(b) + O(b,k-1) + 1
  |  | l_b = C(b) + O(b,l)
  |  | if k_b ≤ l_b then
  |  |  | I = I ∪ InexRecur(W,i,z-1,k_b,l_b)
  |  |  | if b = W[i] then
  |  |  |  | I = I ∪ InexRecur(W,i-1,z,k_b,l_b)
  |  |  | else
  |  |  |  | I = I ∪
  |  |  |  | InexRecur(W,i-1,z-1,k_b,l_b)
  |  |  | end
  |  | end
  | end
  | return I
end
main(W,z)
begin
  | Calculated(W)
  | InexRecur(W,i,z,k,l)
end
```

Fig. 2　BWA アルゴリズム

従うと，SA インターバル (5,5) は参照文字列 X における位置 3 に対応している．したがって，W は位置 3 にマッピングされることになる (図 5(a))．1 個の SNP を持つ結果は，SA インターバル (6,6) を持っている．したがって，W は位置 2 にマッピングされることになる (図 5(b)) 同様に，1 個の欠損はは SA インターバル (3,3) を持ち，位置 1 にマッピングされる (図 5(c)) 図 4 に示すように，InexRecur 処理は，深さ優先探索，幅優先探索，そのハイブリッドにより行うことができる．



(a) Reference sequence X, short-read W and $C(a)$ of X　　(b) Rotation of X



(c) Sorting result

Fig. 3　Mapping example

2.1 FPGA ベースアクセラレータのアーキテクチャ

ショートリード配列マッピング問題においては，参照ゲノム配列は通常同じものが用いられる．そのため，本稿では，Burrows-Wheeler 変換と生起配列をあらかじめオフラインで生成しておき，マッピング処理の際には FPGA に転送するだけにする．図 6 に FPGA を用いたアクセラレータアーキテクチャを示す．FPGA アクセラレータは，シンプルな 128 個の PE と 2 個の DDR2 メモリから構成されている．各 PE では，異なるショートリードに対するマッピング処理が並列に行われる．生起配列とショートリードは，それぞれ DDR2 メモリと FPGA 内のメモリに転送される．

図 7 に PE の構成を示す．PE は，32 ビット加算器，比較器，レジスタから構成され，式 (1)，および (2) を計算する．1 回の InxRecur 処理

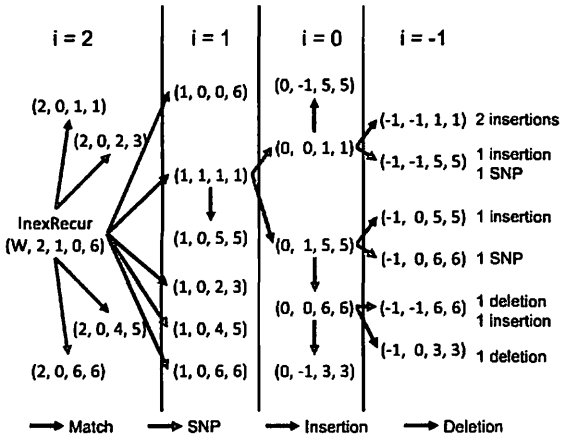


Fig. 4　ショートリード W のマッピング

Position	0	1	2	3	4	5	6
X	C	C	T	G	A	G	
W				C	G	A	
Result	C	C	T	C	G	A	G

(a) SA interval : (5,5), position : 3
Insertion @ X(3)

Position	0	1	2	3	4	5	6
X	C	C	T	G	A	G	
W			C	G	A		
Result	C	C	C	G	A	G	

(b) SA interval : (6,6), position : 2
SNP @ X(2)

Position	0	1	2	3	4	5	6
X	C	C	T	G	A	G	
W		C	G	A			
Result	C	C	G	A	G		

(c) SA interval : (3,3), position : 1
Deletion @ X(2)

Fig. 5　一つのミスマッチを許した場合のマッピング結果

を終了した後に，レジスタファイルから新しいデータが読み込まれる．図 2 で示したように，各 InxRecur 処理では，新しい InxRecur 処理が呼ばれる．この再帰的呼び出しのパラメータはレジスタファイルに記憶され，再帰的呼び出しの制御に使われる．PE 内の ADD/SUB ユニットは 式 (1)，および (2) を計算するために使われる．比較器と制御パスが条件分岐の制御を行う．古いショートリードに対する処理が終了した後に，新しいショートリードが PE に入力される．FPGA 内の出力メモリに記憶されたマッピング結果は，CPU から読み出される．出力メモリがオーバフローしないように，CPU は，出力バッファから結果を読み出し，読み出しが終了した

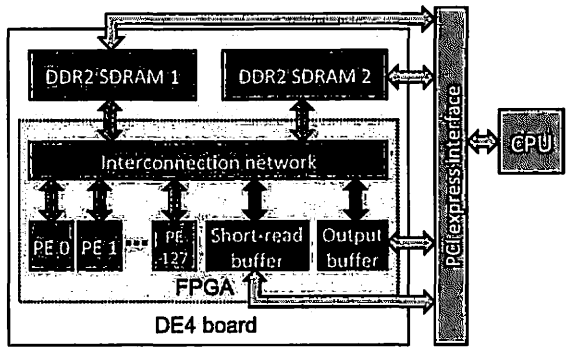


Fig. 6　アクセラレータアーキテクチャ

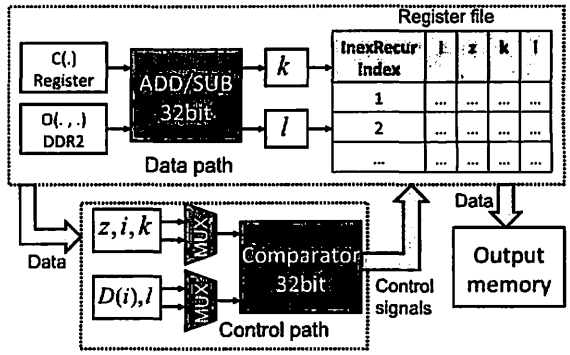


Fig. 7　PE の構成

データを消去するさらに，マッピング処理が行われている間，CPU はショートリードデータを少しずつ FPGA に転送している．FPGA 内の PE は，複雑な浮動小数点演算器や複雑な制御回路を有する CPU とは異なり，ショートリードのマッピングに特化した固定小数点演算のための基づくシンプルな演算器である．そのため，数多くの PE を 1 個の FPGA 上に搭載することが可能となっている．

図 8. に，複数の PE による DDR2 メモリへのアクセスのタイムチャートを示す．本アーキテクチャでは，トータルの性能は，DDR2 メモリへのアクセス性能で決まる．ショートリードマッピングでのメモリアクセスはランダムアクセスであり，そのアクセスは入力されたショートリードに依存する．CPU は理論的には 50GB/s 以上の高い帯域を有しているが，ランダムアクセスに対しては実行メモリ帯域は大幅に低くなる．一方，FPGA では処理に特化したアドレス

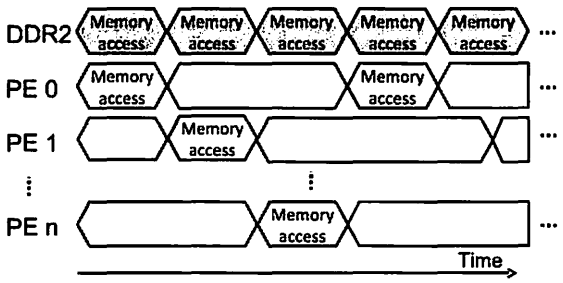


Fig. 8　メモリアクセスのタイムチャート

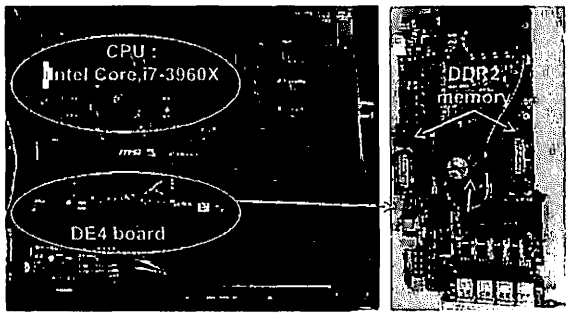


Fig. 9　構築したショートリード配列マッピングシステム

生成回路により，ランダムアクセスであっても高いバンド幅を得ることができる．

3. 評価

評価には，DE4 ボード[6] を用いた．DE4 ボードには，Altera EP4SGX530KH40C2 FPGA と 4GB DDR2-SDRAM が 2 個搭載されている．図 9 に構築したシステムを示す．このシステムでは，core i7-3960x CPU を搭載したマザーボードと，DE4 ボードが PCI express ポートを経由で接続されている．アクセラレータの動作周波数は，100MHz である．表 1 に，1 個の PE のリソース使用量を示す．この表から分かるように，PE サイズはレジスタ数により制限されており，1 個の FPGA に 400 個以上の PE を搭載できることが分かる．

表 2 は，CPU ベースシステムと，FPGA ベースシステムのメモリ帯域の比較を示している．CPU は理論的には高いバンド幅を持っているが，ランダムアクセスに対しての実行メモリ帯域は大幅に低下することが分かる．ショートリード配列マッピングにおけるメモリアクセスはランダムであるため，実行メモリ帯域が大幅に低下し，それにより CPU ベースシステムの性能が大幅に低下する．一方，FPGA ベースのシステムでは，対象とする処理に特化した無駄のないメモリアドレス生成ユニットを設計することができるので，ランダムアクセスであっても，5.15GB/s という理論的な上限値に近い高いバンド幅を得ることができる．FPGA には，2 個の DDR2 メモリを接続できるため，10GB/s 以上のバンド幅を得ることができる．その結果，FPGA ベースシステムでは，CPU ベースシステムと比較して 10 倍以上の性能を達成できる．さらに，3 個の FPGA ボードを一つのホスト PC に PCI express ポートを接続できるため，それに比例して性能を向上できる．また，そのようなシステムを複数用いたり，FPGA ボード同士を低レイテンシな専用 I/O により接続することにより，さらなる性能向上が期待できる．

我々は，3000 文字程度の参照配列に対して 100 個のショートリードのマッピングが可能なシステムの構築した．ソフトウェアでの処理と比較

Table 1　1 個の PE のリソース使用量

リソース名	使用量	(使用量)/(全利用可能リソース量) ×100%
LUTs	844	0.20
Registers	1015	0.24
Onchip memory	1.25kB	0.05

Table 2　メモリ帯域の比較

	Bandwidth (GB/s)	
	理論値	ランダムアクセス時の値
CPU (i7-3960x)	51.2	1.06
FPGA (EP4SGX 530KH40C2)	6.4 × 2	5.15 × 2

して，10 倍程度の高速化が達成された．現在，より大規模な参照配列に対応できるシステムを開発中である．

4. まとめ

本稿では，BWA アルゴリズムを高速化するための FPGA ベースアクセラレータを提案した．ソフトウェアでの処理と比較して 10 倍程度の高速化が達成された．今回の開発で用いた FPGA は，中規模の FPGA で高いグレードのものではない．現在利用可能な最新の FPGA(Altera 社 Stratix V など) を用いることにより，さらなる高速化・高並列化が可能できることが期待される．

謝辞

本研究の一部は文部科学省科研費 12020735 の助成を受けたものです．

参考文献

1) H. Li, “Maq: Mapping and assembly with qualities”, http://maq.sourceforge.net/, 2008.

2) B. Langmead, C. Trapnell, M. Pop, and S. Salzberg, “Ultrafast and memory-efficient alignment of short dna sequences to the human genome”, Genome Biology, Vol.10, No.3, p.R25, 2009.

3) Heng Li and Richard Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform”, Bioinfomatics, Vol.25, No.14, pp.1754-1760, 2009.

4) Hasitha Muthumala Waidyasooriya, Masanori Hariyama and Michitaka Kameyama, “”Implementation of a Custom Hardware-Accelerator for Short-read Mapping Using Burrows-Wheeler Alignment“”,? 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS) ,? pp.651-654,?2013.

5) P. Ferragina and G. Manzini, “Opportunistic data structures with applications”, Proc. of 41st Symp. on Foundations of Computer Science, pp.390-398, 2009.

6) http://www.altera.com/education/univ/materials/boards/de4/unv-de4-board.html