2003 정보 및 제어 학술회의 (2003.11.21-22)

효율적인 DNA 서열 생성을 위한 진화연산 프로세서 구현

Implementation of GA Processor for Efficient Sequence Generation

전 성 모*, 김 태 선**, 이 종 호***
(Sung Mo Jeon, Tae Seon Kim, Chong Ho Lee)

* 인하대학교 전기공학과(전화:(032)860-7396), ** 가톨릭대학교 정보통신전자공학부, *** 인하대학교 정보통신공학부

Abstract : DNA computing based DNA sequence is operated through the biology experiment. Biology experiment used as operator causes illegal reactions through shifted hybridization, mismatched hybridization, undesired hybridization of the DNA sequence. So, it is essential to design DNA sequence to minimize the potential errors. This paper proposes method of the DNA sequence generation based evolutionary operation processor. Genetic algorithm was used for evolutionary operation and extra hardware, namely genetic algorithm processor was implemented for solving repeated evolutionary process that causes much computation time. To show efficiency of the proposed processor, excellent result is confirmed by comparing between fitness of the DNA sequence formed randomly and DNA sequence formed by genetic algorithm processor. Proposed genetic algorithm processor can reduce the time and expense for preparing DNA sequence that is essential in DNA computing. Also it can apply design of the oligomer for development of the DNA chip or oligo chip.

Keywords : FPGA, genetic algorithm processor, DNA computing

Ⅰ. 서론

Adleman이 1994년 DNA 분자를 이용하여 헤밀토니안 경로문제를 해결한 이후, DNA 분자의 병렬성을 이용한 다양한 DNA 컴퓨팅 기법들이 개발되고 있다.

DNA 컴퓨팅은 생체분자인 DNA를 계산 및 저장의 매체로 사용하고, 실제 생물학 실험에서 사용되는 여러 가지 실험들을 연산자로 이용함으로써, 상보성과 같은 유용한 특성을 이용하게 되었지만 이들의 화학적 특성에 의한 문제점이 발생하게 되었다. 이러한 어려움을 극복하기 위하여 많은 연구들이 진행되고 있고 그 중에서도 오류가능성을 최소화하는 방향으로 DNA 서열을 생성하고자 하는 연구가 활발히 이루어지고 있다. 또한 그 연구 결과로서 DNA 서열의 적합도를 판별하고자 하는 적합도 함수들이 다양하게 밝혀지고 정의되고 있다.[2][7]

본 논문에서는 DNA 서열의 적합도를 판별하는 함수들을 이용하여 여러 적합도 함수를 사용한 유전 알고리즘(genetic algorithm)의 기술들을 하드웨어에 적합하게 적용하고자 시도하였다.[5]

많은 계산 문제들은 굉장히 많은 수의 가능한 해들을 탐색하여야 하는 것을 필요로 한다. 더욱이 해의 수가 작을 때는 이들을 낱낱이 열거해 나가는 단순한 방식으로 찾을 수 있지만, 대단히 클 때는 현실적으로 불가능하게 된다. 다시 말하면 DNA 염기개수와 서열개수가 증가함에 따라 계산해야할 데이터가 기하급수적으로 늘어나고 많은 연산시간을 유발한다. 그러나 대부분의 연구는 기존의 컴퓨터를 이용하여 소프트웨어적으로 DNA 서열을 생성하였다.[3][4] 하지만 실시간 응용, 빠른 연산처리속도가 필요한 분야, stand-alone으로 동작해야 하는 분야 등에서는 기존의 폰노이만 컴퓨터로는 구현하기가 어렵다. 그래서 이러한 분야에서는 반드시 하나의 전용 칩으로 인공생명의 여러 모델을 하드웨어로 집적화 하여 구현하는 것은 필수적이다.

Ⅱ. DNA 서열 최적화 방법

DNA 컴퓨팅의 성공 여부는 문제 해결에 사용된 DNA 서열의 종류와 성질에 크게 의존하므로, DNA의 화학적 특징을 수치화 하여 실험에서의 적합성을 고려하는 일이 중요하다.

적합도 함수의 자세한 표기를 위해 몇 가지 기호를 정의한다.

- 염기집합 $\Lambda = \{A, T, G, C\}$, 서열들의 전체집합 Λ^*
- $a \in \Lambda$, $x, y \in \Lambda^*$, 서열의 길이 $|x|$,
  서열 x의 i번째 염기 $a : x(i) = a$
- 서열 x를 k만큼 이동시킨 서열 $\lambda_k(x)$

1. Similarity

Similarity는 한 서열이 다른 서열과 다른 고유한 정도를 측정하기 위한 방법이다. 기본적으로 임의의 두개의 서열에서 동일한 위치에 같은 종류의 염기가 있는 정도를 측정한다. 부가적으로 보다 정확한 고유성을 보장하기 위해서

한 서열을 한 염기만큼 이동시키며 다른 서열과의 고유성도 비교하도록 하였다. 아래의 그림1에서 좌측의 Similarity는 3이고, 한 염기만큼 이동한 우측의 경우는 1인 결과를 보여주고 있다.[4]

$a, b \in \Lambda,\ x, y \in \Lambda^*$, 비교길이 $l = [1, min\{|x|, |y|\}]$

$$d(a,b) = \begin{cases} 1, & \text{if } a = b \\ 0 & otherwise \end{cases}$$

$$s(x,y) = \sum_{i=1}^{l} d(x(i), y(i))$$

$$S(x,y) = \sum_{k=-|y|}^{|x|} [\chi_S(x, \lambda_k(y))] \qquad (1)$$

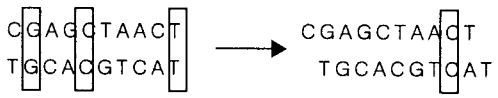

그림 1. Similarity
Fig 1. Similarity

2. H-measure

Similarity는 방향이 동일한 서열간의 고유성만을 점수화하는 기준인 반면, H-measure는 서로 염기 서열 방향이 상보적이어서(5'-3') 두 DNA 서열이 서로 결합하여 duplex를 형성하는 경우에 내가 원하는 염기 서열들간에 결합이 생성되는 지의 여부를 판단하는 기준이다. 각 서열이 서로 상보 서열이라는 가정 하에서 각 염기가 서로 상보 결합을 형성하는 횟수를 계산한다. 역시 Similarity와 마찬가지로 한 염기씩 이동을 하면서 계산하는 과정도 포함되어 있다. 이 H-measure는 실험과정에서 발생하는 잘못된 결합과 위치이동 결합을 미리 예방하는 효과가 있다. 아래의 그림 2에서 좌측의 H-measure는 2이고, 한 염기만큼 이동한 우측도 2인 결과를 보여주고 있다.[4]

$$d(a,b) = \begin{cases} 1, & \text{if } a \text{ and } b \text{ } is\ complementary \\ 0 & otherwise \end{cases}$$

$$h(x,y) = \sum_{i=1}^{l} D(x(i), y(i))$$

$$H(x,y) = \sum_{k=-|y|}^{|x|} [\chi_h(x, \lambda_k(y))] \qquad (2)$$

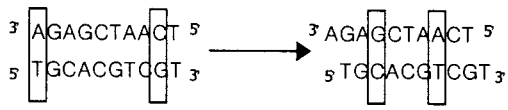

그림 2. H-measure
Fig 2. H-measure

전체 적합도는 각 목적함수의 값을 가중합(weighted sum)하여 그 값이 최소화 하도록 하는 것이다.

$$F = \sum_{i=1}^{n} w_i f_i \rightarrow \min \qquad (3)$$

III. 하드웨어 구현

1. 제안된 진화연산 프로세서의 구조

진화연산프로세서의 효율적인 구현은 하드웨어의 복잡성과 성능을 동시에 고려한 구조여야 한다. 따라서 본 연구에서는 이를 위해 프로세서의 내부의 모든 동작은 handshaking protocol에 의해 각 모듈들이 동작하도록 설계하였다. 그림6은 제안된 유전알고리즘 프로세서의 내부 블록 다이어그램을 나타낸 것이다. 각 모듈은 내부적으로 알고리즘 수행 상태에 따라 하나 혹은 두개가 동시에 독립적으로 실행되어진다.

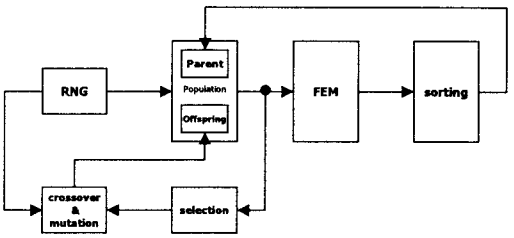

그림 3. 진화연산 프로세서의 블록 다이어그램
Fig 3. Block diagram of GA processor

2. Random Number Generator(RNG)

난수 발생기는 개체군의 배열로부터 개체의 선택, 교차의 발생 가능성 및 교차와 돌연변이의 발생위치를 결정하기 위해 사용되었다. 난수의 평균이 피연산자의 반이 되지 않아서 정확한 계산을 기대하기 어려운 일반적인 LFSR(linear feedback shift register)방법의 단점을 보완한 cellular automata(CA) 방법을 이용하여 설계하였다.[6]

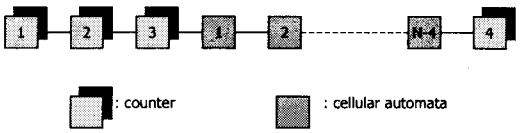

그림 4. 난수 발생기(RNG)
Fig 4. Random number generator(RNG)

주기 및 초기 값의 문제를 개선하기 위해 4bit counter를 이용하여 CA의 경계조건을 변화시켜 maximum length cycle를 증가시켰다. 난수발생기에 사용된 CA는 시뮬레이션을 통해 주기가 가장 긴 rule 90(식(4))에 의해 설계되었다.

$$a_i(t+1) = a_{i-1}(t) \otimes a_{i+1}(t) \qquad (4)$$

3. 적합도 평가 모듈(FEM)

적합도 측정을 위한 하드웨어 구현은 아래 그림6와 같이 각각의 적합도 측정 모듈(fitness evaluation module, FEM)을 병렬로 설계하여 적합도 평가에 소요되는 시간을 줄였다. 우선 들어오는 입력의 데이터 비트를 하위비트부터 개체의 서열길이에 맞게 각각의 레지스터에 저장하게

된다. 레지스터에 저장된 값과 입력 데이터를 2비트씩 우측으로 이동시키면서 각 서열들 간의 적합도를 계산하게 된다. 2비트씩 이동시키는 이유는 각 서열들간의 위치이동 결합, 잘못된 결합을 측정하기 위함이다. 입력으로 염기서열을 나타내는 데이터가 들어오면 Fitness #1은 Similarity을 측정하고, Fitness #2는 H-measure을 측정한다.

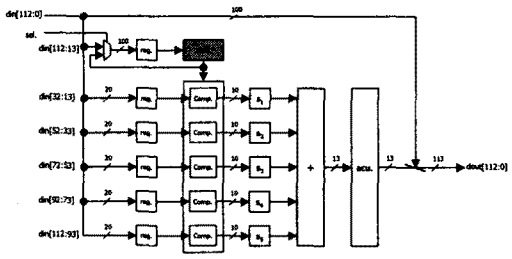

그림 5. Similarity, H-measure 측정 모듈
Fig 5. Similarity, H-measure evaluation module

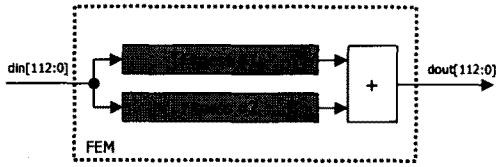

그림 6. 적합도평가 모듈
Fig 6. Fitness evaluation module

4. Sorting 모듈

제안한 유전알고리즘에서 새로운 부모개체(parent)를 만들기 위해서는 적합도가 가장 좋은 순서대로 부모개체의 수만큼이 필요하다. 처음에 메모리에서 하나의 개체를 출력하여 가장 좋은 적합도를 가지는 개체라고 가정하고 레지스터에 저장하고 그 이후 메모리에서 출력하는 개체를 차례로 비교하여 더 나은 적합도를 가지고 있으면 갱신한다. 이 과정을 메모리에 저장되어 있는 모든 개체에 대하여 행한 이후 중간에 저장한 가장 좋은 적합도를 가진 개체를 가장 상위에 위치하도록 메모리에 다시 저장한다. 그 다음은 메모리의 두 번째에 저장되어 있는 개체부터 다시 비교해나가 적합도가 둘째로 가장 좋은 개체를 찾아낸다. 찾아낸 개체는 다시 메모리의 다음 번지에 저장한다. 이런 방식으로 메모리에 적합도가 좋은 순서대로 저장하면서 정렬을 한다. 메모리에 정렬이 완료되면 부모개체(parent)의 개수만큼 부모개체를 갱신 한다.[5]

5. Selection 모듈

일정한 크기의 집단을 가지는 유전 알고리즘에서 발생하는 중요한 문제점은 유전적 다양성 유지와 선택압력의 저하이다. 먼저 유전 알고리즘이 최적해나 근사의 해를 찾기도 전에 상대적으로 적합도가 높은 개체가 발생할 경우가 생기게 된다. 이런 경우를 짧은 시간동안데 모든 개체에 이 개체가 영향을 주어 조기수렴에 빠지게 된다. 다음은 개체들이 적합도 관점에서 매우 가까워져 더 나은 개체들이 선택될 기회가 많지 않음으로써 발생하게 된다. 이에 본 실험에서는 토너먼트 선택방법을 사용하여 비교되는 개체를 증가시킴으로써 선택 압력을 높여 확률적으로 최적의 해에 도달할 가능성을 높였다. 토너먼트 선택방법은 확률에 기반한 모델에 비해 개체들의 적합도 총합이나 적합도에 따른 개체의 분류를 위한 추가적인 정보를 요구하지 않기 때문에 면적과 속도의 향상을 얻을 수 있을 뿐만 아니라 하드웨어적인 면에서 그 성능을 향상시킨다.

메모리에서 두개의 개체(chromosome)를 선택하여 적합도를 비교한 후 우성인 개체를 자식개체(offspring)로 만든다.[6]

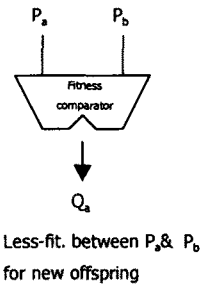

그림 7. 두 부모개체의 선택 방법
Fig 7. Selection method of two parents' chromosome

6. Crossover & Mutation 모듈

개체들간의 유전 정보의 교환을 가능케 해주는 교차(crossover)는 두 부모 염색체를 조합하여 일부를 바꾸어 자식의 염색체를 생산하는 조작이다. 난수 발생기에서 교차점을 받아 개체(chromosome)의 각 bit의 index와 비교하여 index가 작은 경우는 각 개체를 교환하고, 그렇지 않은 경우는 그대로 자손을 생성함으로써 교차가 이루어진다. selection이 끝난 후 난수발생기로부터 받은 임의의 확률과 돌연변이율을 비교하여 임의의 확률이 크면 '1'이라는 bit를 생성하고, 그렇지 않은 경우는 '0'을 생성한다.[6] 이러한 반복과정을 통해 자손개체(offspring)를 생성하게 된다.[6]

Ⅳ. 실험 결과

본 논문에서 제안된 DNA 서열 생성을 위한 진화연산 프로세서를 실제 하드웨어 구현시 출력과 소프트웨어 시뮬레이션으로 나온 출력과의 비교를 통하여 어느 정도의 신뢰성을 가지는지를 보여주고 있다.

표1의 적합도 평가에 사용된 서열은 무작위로 생성된 CGCCGAACTT, CGCTTAACTG, CGCTAAAGTG, TGGTTTTTTT, CTATATACTG 이다.

표1. 적합도평가 결과 비교

	소프트웨어	하드웨어
Similarity	682	665
H-measure	600	603

표2. 무작위 서열과 진화연산 서열의 적합도 비교

	무작위 서열	진화연산 서열
Similarity	676	507
H-measure	576	503

위의 표1에서와 같이 결과가 차이가 나는 것은 하드웨어에서 한 염기씩 이동시키며 적합도를 측정할 때 처음과 마지막의 데이터 이동시 예상치 못한 적합도 값이 추가 되는 것을 막기 위해 처음과 마지막 서열에서는 한 염기씩 이동하는 것에 제한을 가했기 때문이다. 10mer의 DNA 서열 5개가 하나의 개체라고 했을 때 개체 하나의 적합도를 평가하는데 400ns의 시간이 걸리게 된다. 표2에서는 무작위 서열과 진화연산 후 서열의 Similarity와 H-measure값을 비교한 것이다.

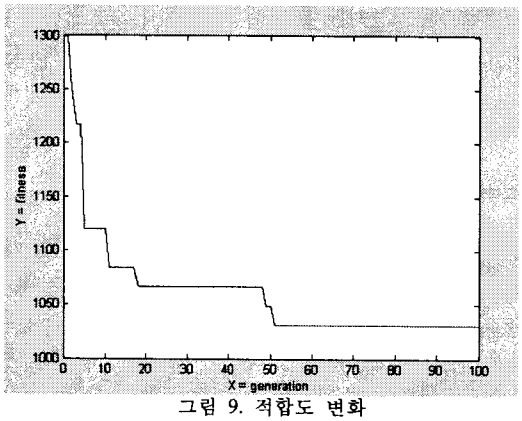

그림 9. 적합도 변화

Fig 9. Graph of the fitness change

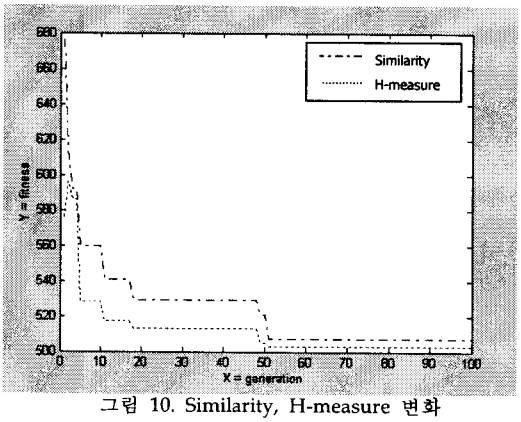

그림 10. Similarity, H-measure 변화

Fig 10. Graph of the Similarity and H-measure change

위 그림9와 그림10은 적합도 평가 시뮬레이션 결과를 보여주고 있다. 그림에서 적합도가 세대를 거듭할수 록 작아지며 좋아짐을 확인 할 수 있다.

V. 결론

이 논문에서는 효율적인 DNA 서열 생성으로 DNA 컴퓨팅에 적용하기 위해 새로운 하드웨어 지향의 유전 알고리즘을 제안하여 소프트웨어로 검증하고 하드웨어 기술 언어인 Verilog을 이용하여 하드웨어로 구현하였다.

제안된 알고리즘은 여러 가지 적합도함수를 DNA 서열 생성에 반영하여 한번에 여러 개의 서열을 생성하도록 하였으며, 실시간 적용을 위한 FPGA구현에 있어서는 하드웨어에서 취할 수 있는 장점인 parallel 방식을 적용하여 설계되었다.

연구된 진화연산 프로세서는 다양한 적합도 함수를 추가하여 보다 더 신뢰할 수 있는 DNA 서열 생성을 위한 도구로써 사용된다.

참고문헌

[1] Melanie Mitchell "An Introduction to Genetic Algorithms", The MIT press, 1997.

[2] Fumiaki Tanaka, Masashi Nakatsugawa, Maasahito Yamamoto, Toshikazu and Azuma Ohuchi "Towards a General-Purpose Sequence Design System in DNA Computing", IEEE, 2002..

[3] U. Feldkamp, S. saghafi, W. Banzhaf, and H. Rauhe, "DNA sequence generator - a program for the construction of DNA sequences", In Jonoska and Seeman [8], pp. 179-188.

[4] S.-Y, Shin, D.-M. Kim, and B.-T. Zhang, "Evolutionary Sequence Generation for Reliable DNA Computing', Proc. of Congress on Evolutionary Computation 2002.

[5] Kalyanmoy Deb, "Multi-Objective Optimization using Evolutionary Algorithms" JOHN WILEY & SONS, pp. 233-240, 2001.

[6] 김진정, "하드웨어 지향의 고속 유전자 알고리즘 프로세서의 구현", 인하대학교 전자재료공학과 석사학위논문, 2000.

[7] 신수용, "DNA컴퓨팅 기법을 이용한 조합 최적화 문제의 해결" 서울대학교 컴퓨터 공학과 석사학위논문, 2000.

[8] N. Jonoska and N. C. Seeman, Eds., 7th international Workshop on DNA-Based Computers, Tampa, U.S.A., 10-13 June 2001. University of South Florida.