자가적응 유전자 알고리즘 프로세서의 VLSI 구현

허 인 수, 이 주 환, *조 민 석, **정 덕 진
인하대학교 전자전기컴퓨터공학과 집적회로연구실
전화 : 032-874-1663 / 핸드폰 : 017-354-7316

VLSI Implementation of Adaptive mutation rate Genetic Algorithm Processor

Insu Hur, Juhwan Lee, Minsok Cho, Duckjin Chung
Intergrated Circuit Reserch Lab. School of Electrical & Computer Eng.
INHA University
E-mail : insher@dreamwiz.com

Abstract

This paper has been studied a Adaptive Mutation rate Genetic Algorithm Processor. Genetic Algorithm(GA) has some control parameters such as *the probability of bit mutation or the probability of crossover. These value give a priori by the designer.* There exists a wide variety of values for for control parameters and it is difficult to find the best choice of these values in order to optimize the behavior of a particular GA. We proposed a Adaptive mutation rate GA within a steady-state genetic algorithm in order to provide a self-adapting mutation mechanism. In this paper, the proposed a adaptive mutation rate GAP is implemented on the FPGA board with a APEX EP20K600EBC652-3 devices. The proposed a adaptive mutation rate GAP increased the speed of finding optimal solution by about 10%, and increased probability of finding the optimal solution more than the conventional GAP.

I. 서론

현대 공학의 여러 적용분야에서 인공생명의 연구모델인 신경회로망, 진화 알고리즘 및 Evolvable Hardware등을 사용하여 자율성, 적응과 학습, 진화 등의 우수한 특징을 시스템에 적용 및 구현하려는 시도가 활발히 이루어지고 있다. 이것은 인간 두뇌를 모방하는데 있어서 궁극적으로는 두뇌의 기능을 부분적으로 행하는 지능적 프로세서를 구현하는 것일 것이다. 그러나 대부분의 연구는 기존의 컴퓨터를 이용하여 인공생명의 연구모델들의 학습에 관한 내용을 시뮬레이션을 통하여 확인하는 일이 대부분이었다. 특히, 실시간 응용 및 환경의 변화에 민감한 응용분야, 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 필수 불가결하다.

진화 알고리즘(Evolutionary Algorithm) 중 유전자 알고리즘(Genetic Algorithm, GA)이 적당한 연산 시간 안에 그들을 풀 수 있는 주요한 메커니즘으로써 사용된다. 이러한 유전자 알고리즘은 반복적인 진화과정과 적응과정을 거치게 되므로 많은 연산시간을 필요함에 따라 별도의 프로세서, 즉 유전자 알고리즘 프로세서(Genetic Algorithm Processor, GAP)라고 불리는 하드웨어가 요구된다. 본 연구에서는 유전자 알고리즘의 하드웨어 구현을 위하여 기존 알고리즘의 장·단점을 분석하고, 기존 알고리즘의 단점을 해결하기 위해 수정된 개체 생존 결정 방법과 수정된 토너먼트 선택 방법의 장점을 접목한 알고리즘을 MATLAB을 이용한 시뮬레이션으로 알고리즘의 타당성을 검증하였다. 또한, 제안된 알고리즘을 VHDL을 이용하여 설계하여 FPGA가 내장된 Agent2000 Design Kit에서 유전자 알

고리즘 프로세서를 제작한 후 동작을 확인하였다. 제안된 하드웨어의 성능 평가를 위해 De-Jong 함수와 같은 수학적 최적화 문제, Set Covering Problem 및 Royal-Road 함수를 적합도 함수로 적용, 하드웨어로 구현하여 평가하였다.

본 논문의 자가적응 유전자 알고리즘 프로세서는 양질의 해의 탐색에의 연산시간을 현저하게 줄이고 또한 해의 탐색의 확률을 증가시키기 위한 알고리즘을 고안하였다.

다음 본론의 1장에서는 기존의 유전자 알고리즘과 제안된 유전자 알고리즘을 설명하고 2장에서는 자가적응 유전자 알고리즘의 하드웨어 구현 및 제안된 프로세서를 설명한다. 그리고 3장에서는 구현된 prototype의 결과 및 분석에 대해 설명하고 마지막으로 결론을 맺고자 한다.

II. 본 론

2.1 자가 적응 유전자 알고리즘 프로세서

Hardware 기반의 GAP를 실시간으로 응용하기 위해서는 최적의 해를 찾는 수렴속도의 단축이 중요한 고려사항 임과 동시에 양질의 해를 찾는 확률의 증가도 크게 고려해야 사항이다. 그래서 본 연구팀은 Adaptive Mutation Rate 적용시켜 Pre-mature convergence에 강한 Adaptive GAP 개발하고 거기에 맞게 하드웨어로 구현하게 되었다. 알고리즘의 고속화 방법으로 유전자 알고리즘 프로세서의 Genetic Operator의 대부분의 연산시간을 점유하는 부분인 Selection 부분을 수정함으로써 고속화가 가능하였다.

2.2 자가적응 유전자 알고리즘 프로세서의 구조

기존의 유전자 알고리즘은 소프트웨어에 주로 사용된 알고리즘으로써 하드웨어의 제약성 때문에 실제 칩상의 구현은 거의 전무한 상태이다. 이에 본 연구는 이 알고리즘을 하드웨어 구현에 맞게 수정 보완하여 효율적이고 빠른 연산을 수행 할 수 있도록 설계하였다. 기존의 GA 프로세서가 tournament selection을 선택하여 비교개체를 증가시킴으로써 selection pressure를 높이는 장점이 있었으나 Population Variance가 낮아져 Pre-Mature Convergence에 취약한 면을 보이게 되었다. 이에 제안된 프로세서는 유전자 알고리즘의 중요한 Parameter인 Mutation Rate을 자기적응화 함으로써 Local Minimum에 강한 프로세서를 설계하였다.

제안된 유전자 알고리즘 프로세서는 효율적인 연산을 위한 병렬 처리와 pipeline 적용 및 handshaking 프로토콜 사용과 더불어 개체군의 크기, 교차 확률, 돌연변이 위치 및 확률, 최대 세대수 및 난수발생기의 초기값 등의 파라미터 값을 쉽게 변경할 수 있도록 설계하였다. 그림 1의 각 모듈의 구성 및 구현 방법은 다음과 같다.

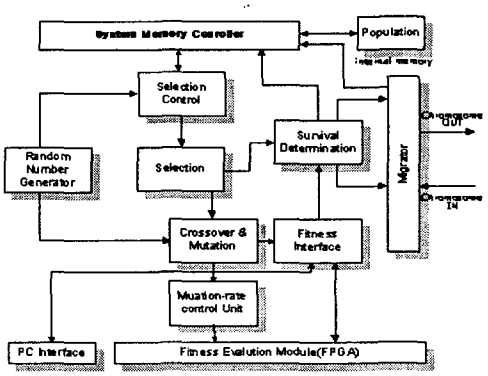

그림 1 Block Diagram of modified GAP

a. Memory Module: 유전자 알고리즘 프로세서는 큰 크기의 개체군를 저장하기 위하여 많은 메모리들을 필요로 한다. 초기 메모리 모듈은 개체에 해당하는 적합도 값과 무작위로 생성된 개체를 저장하며 이들 개체와 적합도 값은 유전자 알고리즘 프로세서가 정상상태(steady-state) 모델에 기반한 제안된 알고리즘을 실행함에 따라 반복적으로 갱신되게 된다.

b. Selection Module: 선택된 개체들 중에 다음 세대로 진화시킬 대상을 임의로 선택하는 부분으로써 수정된 간단한 토너먼트 선택 방법을 이용하여 기존의 확률선택 모델에 비해 하드웨어 구현시 면적의 감소 및 속도의 향상을 얻을 수 있다. 토너먼트 선택 방법은 개체 선택 방법에 있어 무작위선택을 하는 survival-based 유전자 알고리즘에 비해 선택 압력을 더 높임으로서 가능한 더 빠르게 최적의 해에 수렴이 가능하도록 설계되었다. 이 방법은 세대 모델에 기반한 알고리즘에서는 성능 향상이 뚜렷하지 않지만, 정상상태 모델에서는 매우 효율적이다. 그림 2는 앞서 설명한 선택 메커니즘을 도식적으로 나타낸 것이다.

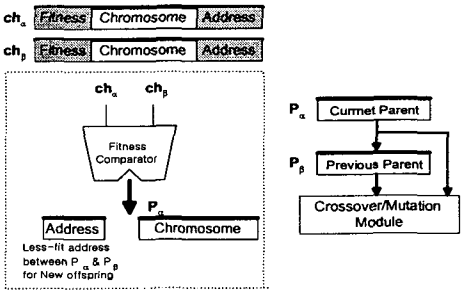

그림 2 Selection method of two parents

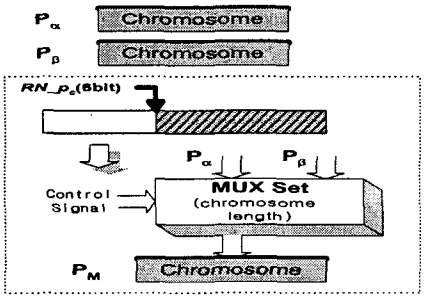

그림 3 Simple crossover(1-point crossover)

c. Mutation & Crossover Module: 여러 가지 문제에 대한 일반성을 위해, 개체의 교차 방법에 있어서는 그림 3에 나타낸 단순 교차(1-point crossover) 및 두 점 교차(2-point crossover)와 균일 교차(uniform crossover) 등의 방법이 모두 수행될 수 있도록 설계되었다. 이를 바탕으로 개체의 생존 결정 모듈에서 결정된 일정한 기간동안에 진화의 정도에 따른 변화를 도입하여, 다양한 해의 생성을 위해 전반부는 균일 교차를 통해 전체 해 공간을 탐색하고, 좋은 해의 유지를 위해 후반부의 두 점 교차를 이용하여 좋은 schema를 보존하는 동적인 적응 교차 조작도 문제의 특성에 따라 사용되었다.

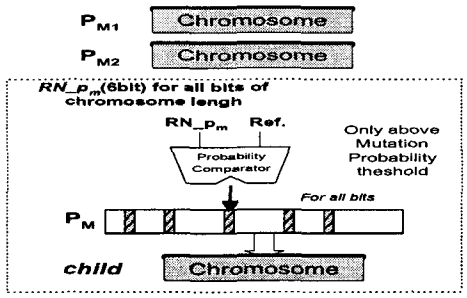

그림 4 Mutation of string (Multi-point mutation)

뿐만 아니라, 돌연변이 방법에 있어서도 단순 돌연변이(single-point mutation) 및 그림 4와 같은 스트링에 의한 돌연변이(multi-point mutation)가 발생하도록 설계되었으며 쉬운 문제에 대해 면적은 작고 빠르게 진화할 수 있는 교차와 돌연변이가 결합된 형태의 유전 연산자도 함께 설계되었다.

d. Fitness Evaluation Module: 교차와 돌연변이와 같은 유전 연산자에 의해 생성된 새로운 개체에 대해 해의 적합도를 평가하는 부분으로서 주어진 문제에 대한 사상 함수와 평가 함수에 절대적으로 의존적이며, 더 복잡하고 어려운 문제에 대해 적합도 평가 과정의 연산시간이 전체 성능에 지배적인 영향을 미친다. 또한 그 연산시간으로 인하여 병목 현상이 발생하는 부분으로서 효율적인 병렬처리를 위해 적합도 함수에 따라 두 개 이상의 모듈로 구성되어 유전자 알고리즘 프로세서에 사용되었다.

e. Random Number Generator(RNG): 난수 발생기는 개체군의 배열로부터 개체의 선택, 교차의 발생 가능성 및 교차와 돌연변이의 발생 위치를 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 설계되었다. 주기 및 초기 값의 문제를 개선하기 위해 4 bit counter를 이용하여 CA의 경계조건을 변화시켜 초기 값에 관계없이 난수를 발생하도록 설계되었으며 maximum length cycle를 증가시켰다.

f. Survival Determination Module: 현재의 개체보다 더 적합한 자손만이 생존하도록 결정하는 부분으로서 모든 개체 중 가장 적합한 개체와 현재의 두 부모 개체 중 적합도가 나쁜 개체 사이의 차이에 따른 개체 생존 조건 변화 및 방법의 변화를 꾀할 수 있도록 설계되었으며 survival-based 유전자 알고리즘에 비해 비교하는 개체의 수를 늘임으로서 빠르게 최적의 값에 도달하도록 구성되었다.

g.. Mutation rate controller : 집단의 개체의 종류가 급격하게 줄어듬에 따라 Pre-mature Convergence에 빠질 수 있으므로, 집단의 Divergence를 증가시켜 주기위해 적합도의 변화에 따라서 가장 적합한 개체와 현재의 두 부모 개체중 적합도가 좋은 개체와의 적합도의 거리에 따라 Mutation rate이 스스로 변화[1]하게 설계되었다.

h. Migrator Module: GAP간의 데이터를 주고 받을 수 있는 인터페이스 기능을 하는 모듈로써 한번의 유전 연산자를 수행 후에 생성되는 두개의 개체들 중에 그 형질이 우수한 자손 개체를 선택하여 migrator를 통해 보내고, 다른 GAP에서 받은 개체는 생존결정 모듈에서 보내온 열성인 부모 개체와 비교하여 적합도가 높을 경우 개체군(memory)에 쓰여지게 된다. migrator는 다른 GAP 로부터 이동된 개체에 대해 무조건 사용하는 것이 아니라 자신의 GAP의 개체의 열성 개체와 비교하여 좋은 개체만 사용되도록 함으로써 개체군 전체의 평균을 증가시킴으로써 최적의 해에 수렴하는 속도 및 효율성을 증가시키게 되는 장점을 지닌다.

i. External Interface Module: Adaptive GAP는 일종의 co-processor로써 적합도 평가 모듈과 함께 구성되어 동작하는데 적합도 평가 모듈은 문제에 따라 가변적이기 때문에 다양한 문제에 적용 가능하기 위해서는 FPGA로 구현되어 진다. 제안된 프로세서는 제작된 칩을 이용하여 PC Interface를 통해 PC에서 볼 수 있도록 Adaptive GAP안에 interface 모듈이 추가적으로 필

요하다.

2.3 실험 및 결과

Adaptive GAP의 성능을 평가하기 위하여 NP 문제의 일종인 non-unicost set coverage problem(SCP)[2]를 사용하였다. 성능 평가는 Matlab를 이용한 소프트웨어 상에서 시뮬레이션과 Or-CAD를 이용한 시뮬레이션, 두 가지를 수행하였으며, 비교대상은 같은 조건에서의 single GAP와 Adaptive GAP 간의 최적의 해의 수렴 속도 및 효율성의 두 가지 측면에서 비교되었다.

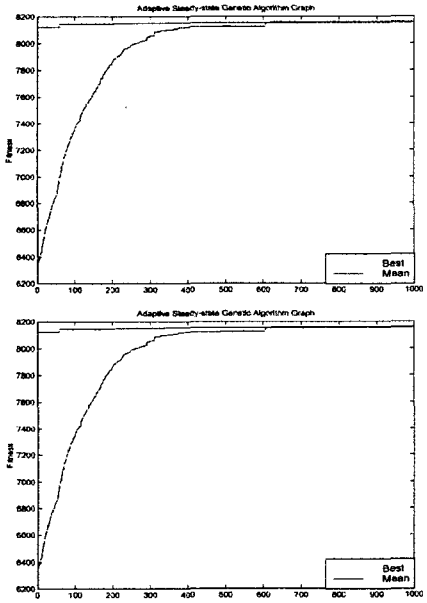

그림 5 SGAP vs. AGAP in set covering problem(1-point crossover)

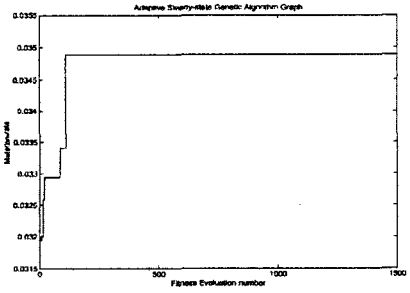

그림 6 Mutation Rate in set covering problem

제안된 방법이 SGAP 보다 수렴속도가 빠르며 Population 전체 평균도 높아짐을 알 수 있다. 또한 그림 8에서는 개체군의 개체의 종류가 줄어듬에 따라 Mutation Rate이 향상됨을 알 수 있다.

◇ OrCAD를 이용한 시뮬레이션

: 사용된 함수는 Set Coverage Problem으로서 최적의 값은 "1FDF" 이며 crossover의 방법에 따라 시뮬레이션을 수행하였다.

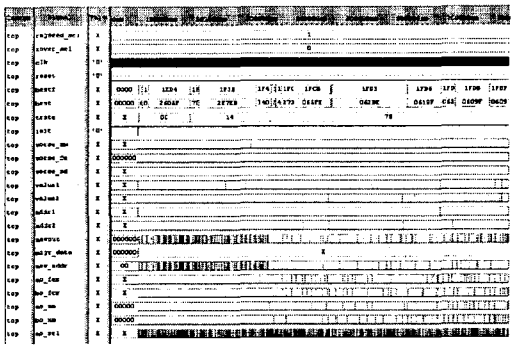

그림 7 Hardware Simulation using OrCAD

Ⅲ. 결 론

구현된 자가 적응 유전자 알고리즘은 최적해 탐색에 있어 기존의 유전자 알고리즘 프로세어에 비해 10%의 속도 향상과 월등한 최적해 탐색확률의 증가를 보였다.

자가적응 유전자 알고리즘 프로세서는 제작된 칩을 이용한 보드 구성을 통해 구현될 것이며, 빠른 연산 시간을 바탕으로 차후의 EHW의 중앙연산처리 장치 및 실시간 처리가 요구되는 최적화 문제, Robot Control, 음성 및 문자인식, 컴퓨터 비전 분야 등의 다양한 문제의 최적의 해를 얻기 위한 도구로써 수많은 응용 분야에 적용될 것이다. 본 연구는 2000년도 뇌연구개발사업의 지원으로 수행되었음.

참고문헌(또는 Reference)

[1] Smith, J.E. & Fogarty, T.C. (1996a) " Self Adaptation of Mutation Rates in a Steady State Genetic Algorithm ".
pp 318 - 323 Proceedings of IEEE International Conference on Evolutionary Computing 1996.IEEE Press

[2] Barry Shackleford, Etsuko Okushi et al., "A High-performance Hardware Implementation of a Suvival-based Genetic Algorithm", ICONIP'97, pp 686-691, Nov, 1997.