진화 시스템을 위한 유전자 알고리즘 프로세서의 구현

論 文
53D-4-8

Implementation of an Adaptive Genetic Algorithm Processor for Evolvable Hardware

鄭 哲 宇* · 金 賢 植* · 金 東 淳** · 鄭 德 鎭***
(Seok-Woo Jung · Hyun-Sik Kim · Dong-Sun Kim · Duck-Jin Chung)

Abstract - Genetic Algorithm(GA), that is shown stable performance to find an optimal solution, has been used as a method of solving large-scaled optimization problems with complex constraints in various applications. Since it takes so much time to execute a long computation process for iterative evolution and adaptation. In this paper, a hardware-based adaptive GA was proposed to reduce the serious computation time of the evolutionary process and to improve the accuracy of convergence to optimal solution. The proposed GA, based on steady-state model among continuos generation model, performs an adaptive mutation process with consideration of the evolution flow and the population diversity. The drawback of the GA, premature convergence, was solved by the proposed adaptation. The performance improvement of convergence accuracy for some kinds of problem and condition reached to 5~100% with equivalent convergence speed to high-speed algorithm. The proposed adaptive GAP(Genetic Algorithm Processor) was implemented on FPGA device Xilinx XCV2000E of EHW board for face recognition.

Key Words : Genetic Algorithm, Adaptive Mutation Rate, GAP, 진화 시스템

1. 서 론

진화 알고리즘(Evolutionary Algorithm; EA)은 자연세계의 진화과정을 컴퓨터상에서 시뮬레이션 함으로써 복잡한 실세계의 문제를 해결하고자 하는 계산모델이다. 진화 알고리즘은 구조가 간단하고 방법이 일반적이어서 응용범위가 매우 넓으며, 특히 적응적 탐색과 학습 및 최적화를 통한 공학적인 문제의 해결에 많이 이용되고 있다. 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 하나의 전용 칩으로 인공생명의 여러 모델을 하드웨어로 집적화하여 구현하는 것은 필수 불가결하다[1,2]. 진화 알고리즘은 염색체를 표현하는 방법과 연산자의 종류 및 특성에 따라 여러 가지가 있으나 대표적으로 4가지로 나누어 볼 수 있다. 유전자 알고리즘(Genetic Algorithms, GA)[2,3]은 고정된 길이의 이진 스트링을 염색체로 사용하며 진화전략(Evolution Strategies, ES)은 실수의 값을 취하는 유전자들로 구성된 벡터를 사용한다. 그 밖에도 그래프와 트리를 염색체 표현에 사용하는 진화 프로그래밍(Evolutionary Programming, EP)[4]과 유전자 프로그래밍(Genetic Programming, GP)등이 있다. 진화적 탐색에 사용되는 연산자로 EP와 ES는 돌연변이(Mutation)를 GA와 GP는 교배(Crossover)를 주로 사용한다. 역사적으로 이들은 서로 독립적으로 발전되어 오다가 1980년대에 와서야 서로 간의 관련성에 대한 관심이 높아져 현재는 이 모델들을 모두 포함한 연구 분야를 가리킬 때 진화 연산(Evolution Computation) 또는 진화 알고리즘이라 한다. 흔히 이 분야를 유전자 알고리즘 또는 GA라고도 하는데 이는 위의 네 가지 모델 중 Genetic Algorithm이 가장 널리 알려져 있기 때문이다[1]. 그러나 이러한 유전자 알고리즘은 반복적인 진화과정과 적응과정을 거치게 되며 양질의 해를 얻는 데는 많은 연산시간을 필요하게 된다는 문제점을 가지고 있다[5,6]. 그리고 최적의 해로 접근하지 못하고 국부해(Local optima)로 접근해 최적이 아닌 해를 찾는 경우가 발생하기도 한다.

이와 같은 GA의 성능을 평가하는 기준으로는 수렴 속도와 수렴 정확도가 있다. 수렴 속도는 문제 해결할 수 있는 해를 찾는데 걸리는 수렴 시간을 의미하고, 수렴 정확도는 얼마나 정확히 가장 좋은 해를 찾는지를 나타내는지를 나타낸다. 기존의 GA의 경우 단일 파라미터를 사용하거나, 사용자가 시스템에 최적화 되는 파라미터를 미리 찾아 주어야 하는 문제점이 있으며, 이 경우 빠른 수렴 속도를 보이나 수렴 성공률에서의 성능 저하를 나타내는 문제점을 가지고 있다. 또한 이러한 유전자 알고리즘은 반복적인 진화과정과 적응과정을 거치게 되므로 많은 연산시간이 필요하기 때문

* 正 會 員 : 仁荷大 情報通信工學科 碩士課程
** 正 會 員 : 電子部品硏究院 先任硏究員
*** 終身會員 : 仁荷大 情報通信工學科 敎授 · 工博
接受日字 : 2003年 12月 9日
最終完了 : 2004年 1月 29日

에 별도의 프로세서, 즉 유전자 알고리즘 프로세서(Genetic Algorithm Processor)라고 불리는 하드웨어의 연구의 필요성이 요구되어 지고 있다.

본 연구에서는 하드웨어에 적합한 적응형 유전자 알고리즘 프로세서를 위한 별도의 알고리즘을 제안하였다. 고속의 연산처리를 요구하는 시스템 및 실시간 처리가 요구되는 복잡한 최적화 문제들에 적용시키기 위한 목적으로 수렴 정확도 및 수렴 속도 면에서의 성능향상을 위한 알고리즘을 고안하였다. 돌연변이의 발생확률을 세대 및 진화중인 개체들의 상태에 따라 적응적으로 변화시키는 알고리즘을 제안하였다. 이 알고리즘의 타당성을 검증하기 위해 유전자 알고리즘의 성능 검증에 널리 사용되고 있는 Dejong function[7]과 같은 여러 최적화 문제에 적용하여 기존의 속도지향의 GA인 Modified Survival-based GA[6]와 비교를 통해 성능의 우수성을 입증하였다. 또한, 제안된 알고리즘을 Verilog HDL을 이용하여 하드웨어로 구현하여 FPGA Device인 Xilinx사의 XCV2000E chip이 탑재된 얼굴인식을 위한 EHW board에서 성공적으로 동작함을 확인하였다.

본 논문의 구성은 2장에서는 돌연변이 연산자의 효율성을 높인 적응형 유전자 알고리즘의 제안 및 성능을 검증하고, 3장에서는 적응형 유전자 알고리즘 프로세서의 구현에 관한 내용을 언급하고, 마지막으로 4장에서 결론을 맺고자 한다.

2. 적응형 유전자 알고리즘

2.1. 유전자 알고리즘

유전자 알고리즘은 1960년대에 Holland의 "진화가 유기체에서 잘 동작한다면 컴퓨터 프로그램 상에서 구현하는 것은 왜 안 될까?" 라는 의구심을 계기로 "인위적인 적자생존"과 같은 비자연적 선택방법과 교배, 돌연변이 등의 방법을 기반으로 해서 창안되었다. 그는 1975년 그의 저서를 통해 생물학적 진화의 추상적 개념으로써 유전자 알고리즘을 소개하고, 알고리즘 하에서 적응의 이론적인 토대를 제시하였다.[3,8]

유전자 알고리즘은 두 부모의 유전자로부터 그들 자손의 유전자를 형성하는 유성생식과 자연환경에서 일어나는 진화원리를 흉내 내고 있다. 이러한 과정이 문제 해결에 이용될 때, 기본적으로 요구되는 전제는 인위적인 진화 현상을 일으킬 초기집단을 구성하는 것이다. 집단은 문제 상에서 잠정적인 해를 뜻하는 다수의 개체로 형성되고 이들은 유전자의 역할을 반영하도록 흔히 비트 스트링의 형태로 표현된다.

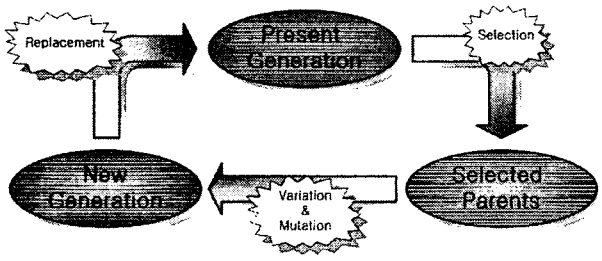


그림 1. 기본적인 진화의 흐름
Fig. 1 Basic Evolution Cycle

이 집단은 유성생식과 진화원리를 모방한 유전 연산자에 의해 점진적으로 개선되게 된다. 각 개체는 집단의 구성원으로 더 적합하고 덜 적합한 어떤 특징을 가지고 있기 때문에 적합도 함수에 의해 개량화 될 수 있다. 따라서 적합도 값이 큰 개체들은 더 많이 선택된다. 선택된 개체들 간에 유전정보가 교환되고, 무작위적인 유전자 변경이 도입됨으로써 다음 세대의 집단이 형성된다.

2.1.1 유전자 알고리즘의 기본 구조

유전자 알고리즘의 구조는 크게 초기화, 적합도 평가, 재생산, 교배와 돌연변이 5단계로 구분된다.

초기화 단계에서는 주어진 문제의 해가 될 가능성이 있는 개체들로 집단이 구성된다. 초기 집단의 구성방법에는 무작위로 균등하게 분포되도록 하거나 경험적으로 구성하는 방법이 있다. 다음단계에서는 개체들의 적합도가 평가된다. 각 개체는 환경에 따라 우성과 열성이 결정되는데, 이와 같은 성질이 수치화되어 선택 및 도태의 기준이 되는 것이다. 재생산과정은 일명 선택연산 과정으로써, 적합도를 기준으로 양질의 개체를 더 높은 비율로 선택하여 개체를 재생산한다. 선택된 개체들은 교배를 통해 재결합되는데 개체 내에 보유하고 있는 현 집단에 대한 유전정보를 교환함으로써 집단에 새로운 개체를 도입하게 된다. 돌연변이는 선택된 개체의 하나 이상의 유전정보를 임의로 변경하여 집단에 새로운 정보를 도입하는 수단이다. 그러나 이러한 변화는 매우 낮은 확률로 이행된다. 그리고 돌연변이를 이용한다는 것은 탐색공간상의 어떤 점에 도달할 수 있는 확률이 결코 0이 아니라는 사실을 말해준다. 이렇게 해서 새롭게 형성된 집단은 다시 평가되고 최적의 해가 발견될 때까지 앞서 수행한 일련의 연산과정이 반복된다. 그림 2는 유전자 알고리즘의 기본적인 흐름을 나타내고 있다.

2.1.2 유전자 알고리즘의 연산자 특성

Holland의 이론에 의하면 강한 스트링 사이에서 발견되는 유사성(similarity)이 탐색을 유도하는 데 중요한 역할을 해주기 때문에 유전자 알고리즘은 병렬적인 형태로 해의 좋은 구성부(building blocks)를 찾아 이들을 재결합함으로써 동

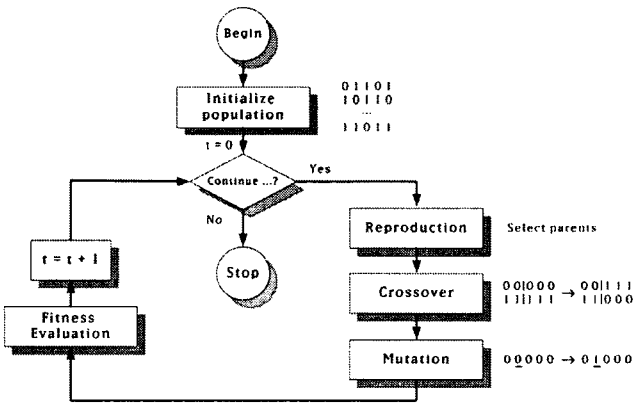


그림 2. 유전자 알고리즘의 동작 순서
Fig. 2 A flowchart of working principle of a genetic algorithm

작하는 것으로 가정하였다. 이런 스트링의 유사성을 나타내주는 하나의 템플릿을 정의하였고, 이의 공식화를 위해 스키마(schema)의 개념이 도입되었다. [2,3]

스키마 이론에 사용되는 용어의 정의는 다음과 같다.

□ 스키마(H) : 스키마란 스트링 위치의 유사성을 효과적으로 표시할 수 있는 템플릿으로 정의된다. 이진스트링의 경우 스키마는 이진 알파벳에 추가로 심볼 '*'을 추가하여 {0, 1, *}로 구성된다.

□ 정의길이 ($\delta(H)$) : 어떤 스키마 H의 정의길이(length)는 첫번째 특이점의 위치와 마지막 특이점의 위치 사이의 거리로 정의 된다.

□ 차수($o(H)$) : 어떤 스키마 H의 차수(order)는 스키마에 포함된 특이점의 수로 정의된다.

□ 빈도($m(H,k)$) : k 세대에서 어떤 스키마 H의 빈도(frequency)는 한 집단 내에서 이와 일치하는 스트링의 수로 정의된다.

□ N, l : 개체군의 크기 및 개체길이

□ P_c, P_m : 교차 및 돌연변이 확률

□ $f_{sum}(k)$: k 세대에서의 개체군의 전체 적합도의 합

□ $\dot{f}(H,k)$: k 세대에서 스키마 H와 일치하는 평균 적합도

k 세대 집단에서 어떤 스키나 H와 일치하는 스트링의 수는 재생산, 교배 및 돌연변이 연산의 영향을 받아 증가(성장) 또는 감소(쇠퇴)할 수가 있다. 스키마의 생성과 파괴는 집단의 구성과 연산자들의 확률적 특성에 따라 매우 복잡한 양상을 띠면서 일어나기 때문에 빈도를 정확히 예측하는 것은 대단히 어렵다. 그러나 근사적인 한계치를 유추해내는 것은 가능하다.

룰렛휠 선택으로 구현되는 일반적인 경우 스키마 H와 일치하는 스트링에 대한 선택확률은 평균적으로 $\dot{f}(H,k)/f_{sum}(k)$와 같게 되고, 집단의 평균 적합도가 $\dot{f}(k) = f_{sum}(k)/N$이므로, 재생산 수행 후 예상되는 빈도는 다음과 같다.

$$\overline{m}(H,k+1) = m(H,k)\frac{\dot{f}(H,k)}{f_{sum}(k)} = m(H,k)\frac{\dot{f}(H,k)}{\dot{f}(k)}$$

이것은 집단의 평균적합도 이상의 적합도를 가지는 스키마타(schemata)는 그 수가 증가하게 되고, 반대의 경우는 스키마타의 수가 감소하게 된다는 것을 의미한다.

교배연산을 거치는 동안 스키마 H의 파괴는 두 확률적 요인에 기인한다. 하나는 교배 확률이고 다른 하나는 교배점이 스키마의 특이점들을 절단할 확률이다. 일반적으로 교배점은 $[1, l-1]$의 범위에서 균등하게 선택되므로 이점이 스키마 내의 첫 번째와 마지막 특이점 사이에 놓일 확률은 $\delta(H)/(l-1)$이 된다. 파괴확률(disruption probability) P_{dist}는

$$P_{dist} = P_c\frac{\delta(H)}{l-1}$$

이 되고, 생존확률(survival probability) P_{surv}는 다음과 같다.

$$P_{surv} = 1 - P_{dist} \geq 1 - P_c\frac{\delta(H)}{l-1}$$

따라서 재생산과 교차연산이 이루어진 후 빈도는

$$\tilde{m}(H,k+1) \geq \overline{m}(H,k+1)[1-P_c\frac{\delta(H)}{l-1}]$$

$$= m(H,k)\frac{\dot{f}(H,k)}{\dot{f}(k)}[1-P_c\frac{\delta(H)}{l-1}]$$

이 된다. 이는 전체 평균 적합도 이상의 적합도를 가지면서 정의 길이가 짧은 H의 수가 증가하게 됨을 의미한다.

돌연변이 연산자는 집단 내의 모든 유전자들을 같은 확률로 변경시키기 때문에 스키마 특이점 중 하나가 돌연변이 되면 이로 인해 스키마가 파괴될 것이다. 스트링 내 각 비트가 변경될 확률이 P_m 이므로 비트의 생존 확률은 $1-P_m$이 된다. 스키마 H에 포함된 특이점의 수(차수)가 $o(H)$이므로 스키마의 생존확률은

$$P_{surv} = (1-P_m)^{o(H)}$$

이 된다. 일반적으로 $P_m \ll 1$ 이므로 Taylor 급수를 얻고 고차항을 무시하면 생존확률은 다음과 같이 근사화된다.

$$P_{surv} \square 1 - P_m o(H)$$

따라서 재생산, 교차, 돌연변이의 영향을 모두 고려한 스키마 H의 생존확률은

$$m(H,k+1) \geq \tilde{m}(H,k+1)[1-P_m o(H)]$$

$$= \overline{m}(H,k+1)[1-P_c\frac{\delta(H)}{l-1}][1-P_m o(H)]$$

$$= m(H,k)\frac{\dot{f}(H,k)}{\dot{f}(k)}[1-P_c\frac{\delta(H)}{l-1}][1-P_m o(H)]$$

가 되며 $P_c\frac{\delta(H)}{l-1}P_m o(H) \ll 1$ 이므로 더욱 간략화 되어 최종적으로 다음과 같은 식으로 정리된다.

$$m(H,k+1) \geq m(H,k)\frac{\dot{f}(H,k)}{\dot{f}(k)}[1-P_c\frac{\delta(H)}{l-1} - P_m o(H)]$$

위 식은 스키마 H의 정의길이와 차수의 함수로 표시되는데 진화가 진행됨에 따라 평균적합도 이상의 적합도를 가지면서 정의길이가 짧고 낮은 차수의 스키마는 개체군 내에서 여전히 그 수가 증가하게 된다는 것을 의미한다. 이러한 스키마 이론을 바탕으로 어느 한 해로의 집중이 되는 진화과정을 반복하여 수행하면 정의길이가 짧고 낮은 차수의 스키마들이 결합하여 보다 정의길이가 길고 높은 차수의 새로운 스키마를 발생시켜 그것을 존중하면서 탐색의 과정이 이루어진다.

2.2 적응형 유전자 알고리즘

유전자 알고리즘은 문제에 대한 가능한 해들을 정해진 형태의 자료구조로 표현하여 진화 과정을 반복 수행함으로써 최적의 해를 생성하는 알고리즘이다. 그러나 반복 수행으로 인한 긴 연산 시간이 큰 단점이다. 이런 문제점을 해결하기 위해 빠른 속도로 처리하기 위한 하드웨어 기반의 유전자 알고리즘 프로세서가 설계되었다.[5,6,10-16] 그러나 빠른 수렴에 비해 최적해가 아닌 국부해로 수렴하는 수렴의 정확도상의 문제가 발생하게 되었다. 이를 해결하기 위해 많은 연구가 활발히 진행되어 왔다. 주로 파라미터의 조절하여 성능을 향상시키려 하였으며, 여러 연구가 전역해로의 수렴 가

능성을 높일 수 있는 돌연변이율의 적응에 초점이 맞추어졌다.

경험적인 실험 결과를 토대로 De Jong[16], Grefenstette [17]은 하나의 상수 또는 일정한 범위의 수로 정의되는 돌연변이율을 정의했으나, 이후 적응(Adaptation)의 개념을 적용한 학자들은 여러 파라미터들을 이용한 돌연변이의 적응 방법을 제안하였다. 이들은 현재 세대(t), 최종 세대(T), 적합도(f) 또는 그들의 평균값 등 여러 변수를 이용해 돌연변이율의 수식을 유도하였다. Schaffer[18]는 개체군의 크기와 개체의 길이를 변수로 한 경험적인 수식을 제안하였고, Hesser와 Männer[19]가 추가적으로 time-dependent한 수식을 이론적으로 결정하였다. 현 세대와 최종세대를 변수로 시간 의존적(time-dependent)인 수식은 Holland[3]와 Fogarty[20]를 필두로 Bäck과 Schütz[21], Hinterding[22]에 의해서도 제안되었다. 이 외에도 적합도를 이용한 방법이 연구되기도 하였다.[23]

2.2.1 유전자 알고리즘의 적응 방법의 분류

유전자 알고리즘의 파라미터를 컨트롤하는 방법의 분류에는 몇 가지 방법이 있다. 적응 대상이나 수준의 관점에서 구분할 수 있고, 또한 적응의 직접적인 형태의 관점에서 분류할 수 있다. 먼저 적응 대상, 수준별 구분은 Smith[24]가 제안한 방법으로 다음 세 가지로 정의된다. 개체군(population) 수준, 개체(individual) 수준 및 구성요소(component) 수준에서의 적응으로써, 개체군 수준의 적응은 전역 조작의 고정된 표준을 사용하는 것으로 전형화 할 수 있으며, 시간에 따른 파라미터의 변화가 큰 특징이다. 개체 수준에서의 적응은 개체군의 각 구성원마다 각각의 확률을 적용시키는 것으로, 탐색 영역 내에서 다른 부분에 서로 다른 전략을 허용하고자 함이다. 이는 비균질한 탐색영역에 적합한 경우이다. 적응방법의 가장 낮은 레벨인 구성요소 수준의 적응방법은 각 개체의 분포를 세부적으로 조율할 수 있는 특징을 가지고 있다.

후자의 경우는 어떻게 파라미터를 정하느냐 하는 관점에서의 분류이다. 그림 3과 같이 동작 시점에 따라 파라미터 튜닝과 컨트롤로 구분되고, 파라미터 컨트롤에는 다음 세 가지 방법이 있다. 미리 정해진 규칙에 의해 동작하는 결정론적 방법(deterministic parameter control), feedback이 존재하는 적응형 방법(adaptive parameter control), 그리고 control parameter가 chromosome에 인코딩되는 자기 적응형 방법(self-adaptive parameter control)으로 나누어진다.[25]

2.2.2 전역탐색과 지역탐색

최적화 방법에는 여러 가지 방법이 있다. 구배법[26], 무작위 탐색법[27] 등의 모든 반복법들은 blind 탐색전략(blind search strategy) 또는 경험적 탐색전략(heuristic search strategy) 중 하나로 수행된다.[28] Blind 탐색전략은 함수의 값 이외에 문제공간에 대한 부가적인 정보를 필요로 하지 않는 반면에, 경험적 탐색전략은 탐색을 가장 좋은 방향으로 유도하기 위하여 부가적인 정보를 요구한다. 이러한 탐색전략에는 두 가지 특징, 즉 최선의 해를 찾는 지역탐색(exploitation)과 탐색 공간을 조사하는 전역 탐색(exploration)이 있다.[29] 지역탐색은 탐색 공간의 전역 탐색을 중요시하지 않고 해를 개선하기 위하여 현재의 점 부근을 지역 탐색하는 속성을 가지며, 전역 탐색은 현재의 점 부근에 있을 지도 모르는 더 나은 영역을 지역 탐색하기 보다는 전 탐색 공간을 조사하는 속성을 가진다.

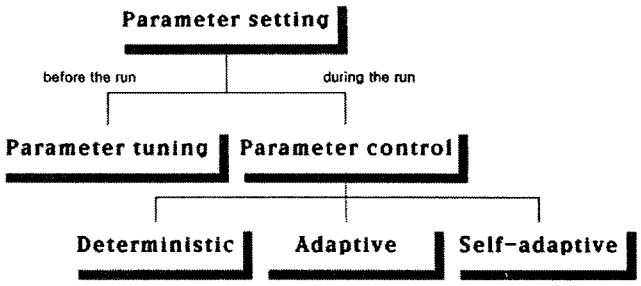


그림 3. Parameter setting 의 분류
Fig. 3 Taxonomy of parameter setting

지역탐색과 전역탐색 전략의 대표적인 방법으로 구배법과, 무작위 탐색법이 있다. 구배법은 현재의 점과 이점에서의 구배로부터 새로운 점을 정하는 일련의 반복동작을 통해 해에 도달하게 된다. 반면에 무작위 탐색법은 무작위로 선택된 새로운 점들을 따라 탐색하기 때문에 지역탐색하는 매커니즘이 없다. 따라서 현재 점 부근에 있을 지도 모르는 더 나은 영역을 탐색하지 못하는 비 방향성 탐색법이 된다. 구배법에서 찾지 목하는 복합한 문제의 최적화에서 무작위 탐색법은 유용하다. 그러나 이 방법은 최적해에 이르기 위해서는 확률적으로 전체 탐색면적의 절반을 커버해야 하기 때문에 아주 큰 해 공간을 탐색하기에는 비능률적이다.

2.2.3 돌연변이율 조정 방법

효과적인 적응 돌연변이율을 위한 알고리즘은 진화 알고리즘의 분석에 기반하여 이루어져야 한다. 본 연구에서는 적응적 돌연변이율은 진화 초기와 후기의 탐색 특징이 전역탐색에서 지역 탐색으로 변화하는 특징을 이용하였다. 즉, 진화 알고리즘에 있어서, 진화 초기에는 개체군이 임의로 구성되었기 때문에 전역탐색의 성질을 띠게 되어 높은 다양성을 갖는 반면, 진화 후기에는 진화를 통해 양질의 해로의 접근이 이루어진 상태이기 때문에 낮은 다양성을 갖는다. 이는 전역 탐색에서 지역탐색으로 변화하게 되는 것으로, 진화 후기의 빠른 수렴을 가능하게 하는 것이다. 그리고 이와 같은 다양성은 임의로 선택되는 두 개체의 유사도에 반영된다. 다음은 탐색 특징의 변화에 따른 돌연변이율 결정을 위한 가정이다.

첫째, 돌연변이율은 $1/l$에서 0.5사이의 값을 갖는다. [21]

$$P_m \square [\frac{1}{l}, 0.5]$$

둘째, 돌연변이율은 진화 세대 증가에 따라 감소하는 경향을 보인다.[7]

$$P_m \propto f(t)^{-1}$$

셋째, 돌연변이율은 진화과정의 다양성을 반영하는 값을

통해 재조정되어야 한다.

$$P_m \propto f(S)$$

(S : 두 개체의 구조적인 유사도)

위의 가정을 기반으로 진화에 돌연변이율의 변화를 나타내는 식을 다음과 같이 exponential 함수로 기본 형태를 정의하였다. [19]

$$P_m(t) = k \square exp[-\Delta \square t] \quad (1)$$

식(1)은 t=0일 때 초기값을 갖고, t=∞일 때 0으로 수렴한다. 이 두 조건을 토대로 수식을 전개하면,

$$P_m(0) = \lim_{t \to 0} P_m(t) = P_{m-initial}$$

$$P_m(\infty) = \lim_{t \to \infty} P_m(t) = 0$$

먼저 초기값 $P_{m\text{-}initial}$은 진화 초기에는 개체군의 높은 다양성을 위해 개체의 변화율을 가장 높일 수 있는 비율 0.5로 정하고, 최대세대인 t=T에서는 l의 길이를 갖는 개체의 최소 변화율인 $1/l$을 돌연변이율로 한다. [21]

$$P_m(0) = 0.5,\ P_m(T) = 1/l$$

위의 조건을 이용해 k와 Δ를 결정하면, $k = 0.5$, $\Delta = -\frac{1}{T}\ln\frac{2}{l}$가 되고, 식(1)은 다음과 같이 바꿀 수 있다.

$$P_m(t) = 0.5 \square exp[\left(\ln\frac{2}{l}\right) \square \frac{t}{T}] \quad (2)$$

마지막으로 세 번째 가정을 고려하면, 돌연변이율은 개체군의 다양성을 반영하기 위해 선택된 두 개체의 유사도를 고려하여 조정된다. 유사도의 식은 다음과 같이 유도할 수 있다.

길이가 l인 두 스트링 또는 벡터 a, b사이의 해밍거리(Hamming distance) $d_H(a,b)$는 두 스트링 사이의 다른 비트의 개수를 의미한다. $d_H(a,b)$는 두 스트링이 동일한 경우인 0을 최소값으로 갖고, 모두 상이한 구조인 경우 최대 l의 값을 갖는다. [30]

$$d_H(a,b) \square [0, l]$$

이를 0에서 1사이의 범위를 갖는 차이도(difference)를 D로 표현하면 다음과 같다.

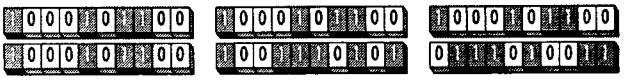
$$D = \frac{d_H(a,b)}{l}$$

반대 개념인 유사도(Similarity) S는 $1-D$가 되어, 다음과 같이 유도된다.

$$S = \frac{l - d_H(a,b)}{l},\ (0 \square S \square 1)$$

($d_H(a,b)$는 두 개체 a, b의 해밍 거리)

그림 4의 (a)는 유사도가 1이기 때문에, 교배연산의 영향이 없게 되어 돌연변이의 역할이 강조된다. 반면에 (c)는 유사성이 없는 상태로써 교배 후 부모와 상이한 구조의 자손이 생성된다. 탐색의 특징을 토대로 한 돌연변이율에 선택된 개체의 유사도를 반영하여 진화 연산의 효율을 높이고자 식(2)에 S^n을 곱하였다. n은 유사도의 반영 정도를 의미하는 상수이다.



(a) (b) (c)

그림 4. 선택된 부모 개체의 구조

Fig. 4 Structures of selected parent chromosomes

$$P_m(t) = 0.5 \square exp[\left(\ln\frac{2}{l}\right) \square \frac{t}{T}] \square S^n \quad (3)$$

2.2.4 돌연변이율 조정 수식의 결정 및 분석

표 1은 Dejong function 등 세 가지 적합도 함수[7,31]를 사용하여 최종세대(T)를 10000세대로 시뮬레이션을 실시한 결과이다. n=2일 때 최고의 성공률을 보이고, 또한 속도지향의 Modified Survival-based GA[6]와 비교하여 성능이 향상되었다. 세 가지 적합도 함수는 다음에 소개하기로 한다.

진화 초기의 유사도 S^n은 작은 값을 갖는다. 진화 후기의 1에 가까운 유사도에 비해 상수 n의 영향을 더 많이 받게 된다. 이는 전역탐색시의 돌연변이율의 차이로 반영된다. n이 클수록 낮은 돌연변이율이 적용되어 더 짧은 전역탐색 기간을 갖게 된다. n이 큰 경우 충분하지 못한 돌연변이로 인해 조기 수렴되었고, n이 작은 경우는 과도한 전역탐색으로 인해 수렴 속도 저하를 초래뿐만 아니라, 높은 돌연변이율로 인한 해의 공간으로의 접근이 오히려 방해받는 결과가 되었다. 아래와 같이 n=2인 돌연변이율을 결정하는 수식을 최종 유도 하였다.

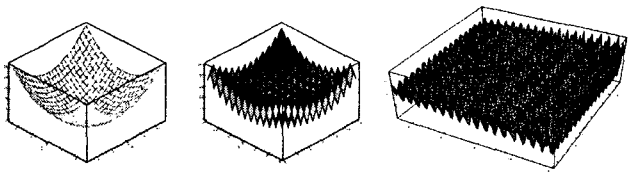
$$P_m(t) = 0.5 \square exp[ln\frac{2}{l} \square \frac{t}{T}] \square S^2 \quad (4)$$

지금까지 연구되었던 기존의 여러 방법[3, 16~22]들을 보면 경험적인 결과를 토대로 돌연변이율을 정하거나, 단순히 시간(세대)에 의존한 돌연변이율을 제안하였다. 적합도를 이용한 방법[23]과 같이 진화중인 개체들의 상태를 고려하여 적절한 돌연변이가 일어나도록 하지 않고 하나의 값 또는 세대에 따른 일률적인 값으로 정한 것이다. 본 논문에서 제안한 적응적인 돌연변이율은 앞서 제안하였던 세 가지의 가정을 만족함과 동시에 세대와 진화중인 개체들의 상태를 고려하여 적절한 돌연변이율을 결정하는 수식을 위와 같이 제

표 1. 제안한 알고리즘의 n에 대한 수렴 세대 및 수렴 정확도 (T=10000)

Table 1. Performance for n of proposed GA

Function \ n		0.5	1	1.5	2	2.5	3	Non-Adaptive
Dejong	세대	8320	7383	6748	**5701**	5449	5077	5796
	정확도	64%	88%	90%	**90%**	86%	77%	56%
Rastrigin	세대	8483	7453	6665	**5694**	5190	4923	5842
	정확도	65%	75%	80%	**82%**	73%	63%	60%
Math	세대	8379	7895	7864	**6552**	6206	5813	6606
	정확도	86%	98%	99%	**99%**	96%	96%	93%



(a) Dejong (b) Rastrigin (c) Mathematical

그림 5. 사용된 세 가지 적합도 함수

Fig. 5 Three kinds of Fitness function

안하였다.

2.2.5 사용된 적합도 함수

1) Dejong function [7] :

$$f(x_1, x_2, x_3) = \sum_{i=1}^{3} x_i^2, \quad (x_i \square [-5.12, 5.12])$$

2) Rastrigin function [31] :

$$f(x_1, x_2, x_3) = 30 + \sum_{i=1}^{3}(x_i^2 - 10 \square cos(2\pi x_i)), \quad (x_i \square [-5.12, 5.12])$$

3) Mathematical function [7] :

$$f(x, y) = 21.5 + x \square sin(4\pi x) + y \square cos(20\pi y), \quad (-3.0 \square x \square 12.1,\ 4.1 \square y \square 5.8)$$

2.3 적응형 유전자 알고리즘의 성능 및 분석

2.2절에서 제안한 적응형 돌연변이율(식(4))을 적용하였을 때, 사용자나 시스템에 의해 결정되는 최종세대(T)에 따른 성능을 표 2에 정리하였다.

표 2. Adaptive vs. Non-adaptive GA의 최종세대(T)별 성능 (100회 평균)

Table 2. Performance between Adaptive vs. Non-adaptive GA for final generation(T)

Func	구분		최종세대 (T)								
			4000	6000	8000	10000	12000	14000	16000	18000	20000
Dejong	Adaptive	세대	3455	4904	5411	5701	6575	7133	7648	7907	8412
		정확도(%)	22	57	85	90	93	96	98	100	100
	Non-adapt	세대	3294	4867	5330	5769	6138	6138	6138	6286	6286
		정확도	12	30	47	56	60	60	60	61	61
	성능 (%)	세대	95	99	99	102	93	86	80	80	75
		정확도	183	190	189	160	155	160	163	164	164
Rast-rigin	Adaptive	세대	3562	4895	5362	5691	6536	7386	7905	8128	8746
		정확도	20	60	71	82	89	90	92	93	96
	Non-adapt	세대	3665	4909	5450	5842	6092	6092	6092	6092	6092
		정확도	8	31	50	60	63	63	63	63	63
	성능 (%)	세대	103	100	103	103	93	83	77	75	70
		정확도	250	194	142	137	141	143	146	148	152
Math	Adaptive	세대	3710	5310	6209	6552	7271	8155	9350	10270	10614
		정확도	9	75	97	99	100	100	100	100	100
	Non-adapt	세대	3708	5571	6284	6606	6684	6684	6684	6684	6684
		정확도	4	38	82	93	95	95	95	95	95
	성능 (%)	세대	100	105	101	101	92	82	72	65	53
		정확도	225	197	118	107	105	105	105	105	105

표 2를 보면 모든 경우에 걸쳐 수렴 정확도 면에서의 성능의 향상이 뚜렷하다. 적응형과 비적응형 두 가지 GA 모두 시스템 또는 사용자가 정하는 최종세대에 따라 성능의 차이가 나타났다. 진화 시간이 짧은 경우는 빠른 수렴과 낮은 수렴율을 보였지만 적응형 GA가 더 우수함을 보였고, 충분한 진화 시간이 주어지는 경우는 적응형 GA는 수렴율이 100%에 가깝도록 높아지지만 비적응형 GA는 수렴속도와 수렴정확도의 변화가 없다. 이는 긴 진화시간이 주어지더라도 비적응형 GA는 충분한 시간을 이용하지 못함을 보이는 것으로, 적응형 GA보다 빠른 수렴을 보이나 수렴정확도가 지역해(local optima)로 수렴하는 경우가 더 많음을 보인다. 적응형 GA는 주어진 진화시간에 따라 수렴속도와 수렴 정확도면에서의 최적의 성능을 이끌어냄을 확인할 수 있다.

III. 적응형 유전자 알고리즘 프로세서의 구현

3.1 적응형 유전자 알고리즘 프로세서의 구조

유전자 알고리즘 프로세서의 효율적인 구현은 하드웨어의 복잡성과 성능을 동시에 고려한 구조여야 한다. 따라서 본 연구에서는 이를 위해 프로세서의 내부의 모든 동작은 handshaking protocol에 의해 각 모듈들이 pipelining으로 작동하도록 설계 되었다. 이 handshaking protocol 기반으로 구성되어진 pipelining은 각 모듈사이에 연산 시간이 많은 차이를 보일 때 기존의 pipelining 방법 보다 효과적이다. 또한 각 병렬 그룹들은 클럭에 의해 독립적인 연산이 가능하게 설계되었기 때문에 효율적인 연산뿐만 아니라 빠른 연산처리가 가능하다. 본 연구에서 제안한 적응형 유전자 알고리즘 프로세서의 블럭도는 그림 6과 같다.

3.1.1 Memory module & Memory controller(MC)

유전자 알고리즘 프로세서는 큰 크기의 개체군을 저장하기 위하여 많은 메모리를 필요로 한다. 이는 FPGA의 내부 메모리 블록이나 외부 SRAM등의 메모리를 이용하여 구현

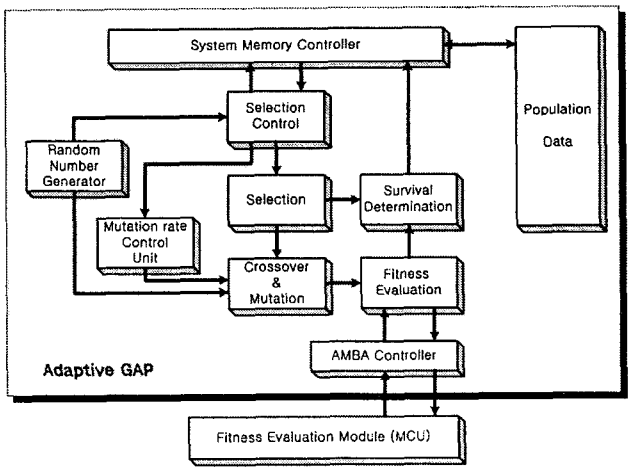


그림 6. Adaptive GAP의 block diagram

Fig. 6 Block diagram of Adaptive GAP

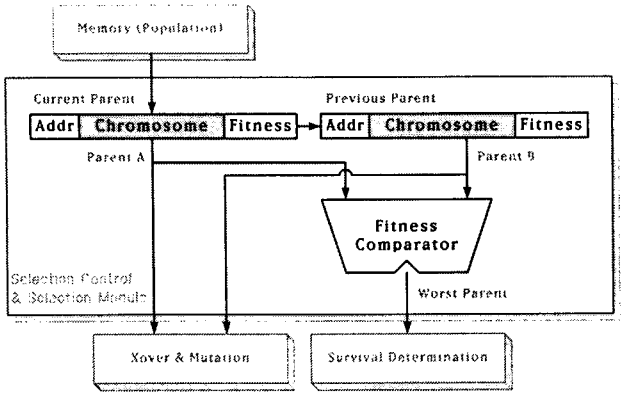


그림 7. 두 부모 개체의 선택 메커니즘
Fig. 7 Selection method of two parents

될 수 있다. 초기의 메모리 모듈을 무작위로 생성된 개체군과 이들 개체의 적합도를 저장하며 이 값들은 진화에 따라 반복적으로 갱신된다. 메모리 컨트롤러(MC)는 개체군이 저장된 메모리와 GA 연산 모듈간의 인터페이스 및 데이터의 입출력을 담당한다.

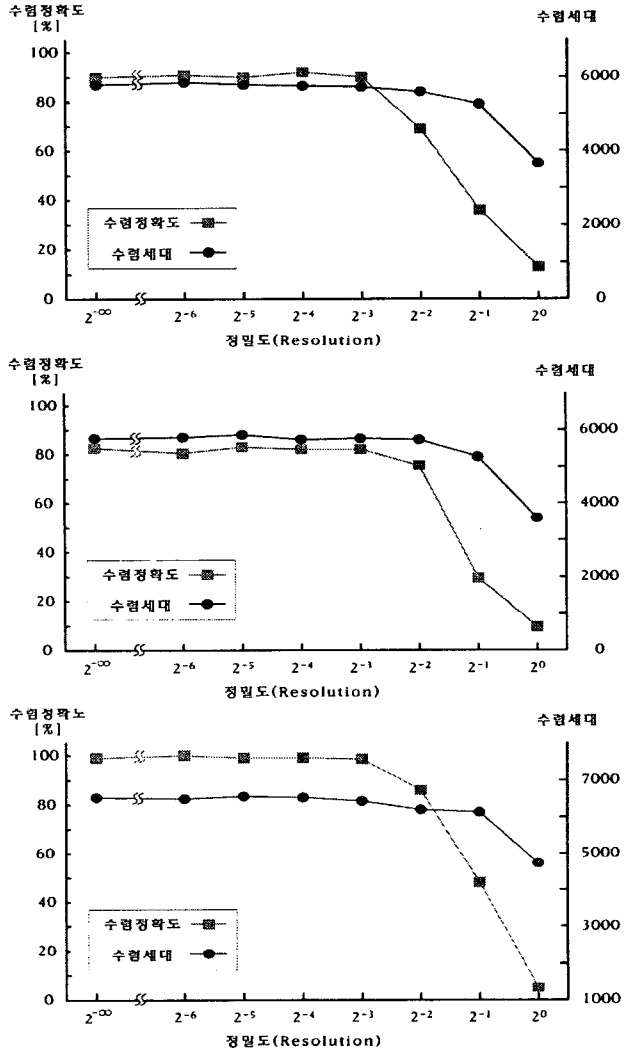
3.1.2 Selection Control & Selection module

선택 연산을 위한 부분으로 크게 두 모듈로 구성되어 있다. Selection Control 부분은 random number generator (RNG)로부터 선택할 부모의 메모리 어드레스를 받아 MC에 요청하는 부분으로 선택된 개체를 Selection module과 MRCU로 전달한다. Selection module은 수정된 간단한 토너먼트 선택 방법(Modified Tournament Selection method)이 적용된 부분으로 기존의 확률 선택 방법에 비해 하드웨어 면적의 감소 및 속도의 향상을 얻을 수 있다.[6]

3.1.3 Mutation rate control unit(MRCU)

기존의 돌연변이율에 관한 수식은 복잡한 계산 과정으로 인한 시간 지연의 문제로 인해 하드웨어로의 구현에 많은 어려움이 있었다. 제안한 수식의 하드웨어 구현에는 지수함수의 처리, 곱셈연산 등이 문제가 된다. 이와 같은 문제를 해결하기 위하여 본 연구에서는 하드웨어 자원을 고려하여 성능의 저하가 없는 범위에서 수식의 여러 값과 계산 중간의 수치들의 정밀도(분해능, resolution)의 단순화하는 방법을 사용하였다. 수식의 계산 과정의 값들의 정밀도를 조절하여 복잡한 곱셈 연산을 단순화 하였고, 지수 함수의 경우는 입력 부분이 일정한 범위에 한정됨을 이용하여 ROM을 이용한 Lookup table로 처리하였다. 이와 같은 단순화 작업은 충분한 실험을 통해 결정하였으며, 단순화로 인한 성능저하는 없었다. 뿐만 아니라 정밀도의 조절로 인한 ROM access 빈도 감소의 이점도 이끌어 낼 수 있다. 그림 8은 세 가지 Test 함수의 정밀도에 따른 성능의 변화를 나타낸 것이다.

정밀도 $2^{-\square}$부터 2^{-3}까지는 수렴정확도의 성능 저하가 나타나지 않았다. 그러나 2^{-3}이하의 더 작은 정밀도를 갖는 경우는 조기 수렴의 현상이 보이면서 성능의 저하가 뚜렷하게 나타났다. 안정성을 확보하기 위하여 2^{-4}을 최적의 정밀도로 결정하고, 돌연변이율의 계산을 고정소수점(fixed-point) 곱셈기, 나눗셈기 및 Lookup table 등을 이용하여 하드웨어로



(a) Dejong (b) Rastrigin (c) Mathematical

그림 8. Test 함수의 정밀도별 수렴정확도 및 수렴세대
Fig. 8 Performances of test functions vs. resolution

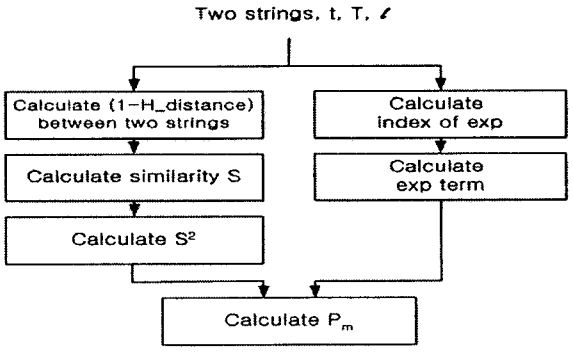


그림 9. MRCU의 연산 순서
Fig. 9 Flowchart of MRCU

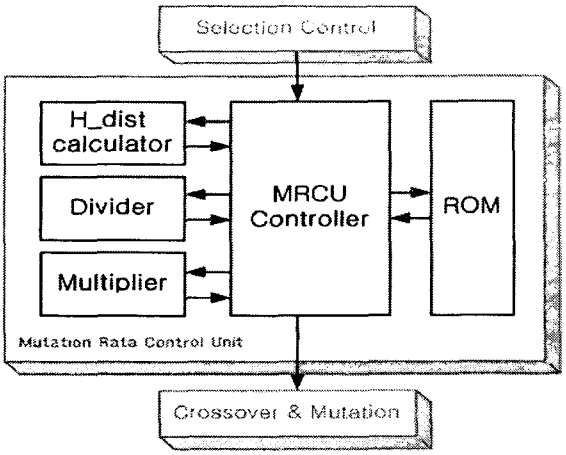


그림 10. MRCU의 Block Diagram
Fig. 10 Block diagram of MRCU

구현하였다. 그림 9는 총 다섯 부분으로 구분되는 식(4)의 연산을 순차적으로 계산하는 MRCU의 동작을 보여주는 순서도이다. 입력으로 선택된 두 부모, 현재 세대 및 파라미터들을 받아 유사도를 계산하는 부분과 exponential term을 계산하는 부분으로 나뉘어 계산된다.

그림 10은 하드웨어로 구현한 블록도이다. 해밍거리를 계산하는 H_dist calculator는 l개의 XNOR 게이트와 이들 출력을 더하는 덧셈기로 구현되었다. 연산량과 하드웨어 면적을 줄이기 위해 $l-d_H(a,b)$을 H_dist calculator에서 두 스트링 사이의 같은 비트의 개수를 더하여 바로 계산하였고, 유사도 S를 구하기 위해 10비트/5비트 크기의 combinational logic 형태의 나눗셈기를 사용하였다. exponential term의 계산은 Lookup table을 이용하여 빠른 속도의 처리가 가능하도록 하였다. 그림 11에 exponential function term $0.5 \square exp[ln\frac{2}{l} \square \frac{t}{T}]$과 lookup table을 이용하기 위해 2^{-5}자리에서 반올림하여 단순화한 값을 그래프로 나타내었다. 나머지 계산과정인 exponential 함수의 index 부분 및 S^2의 계산과 현 세대의 돌연변이율의 계산 과정에 공통적으로 combinational 구조의 5□5비트의 고정소수점(fixed-point) 곱셈기가 이용하였고, 곱셈 연산의 과정이 중복 없이 서로

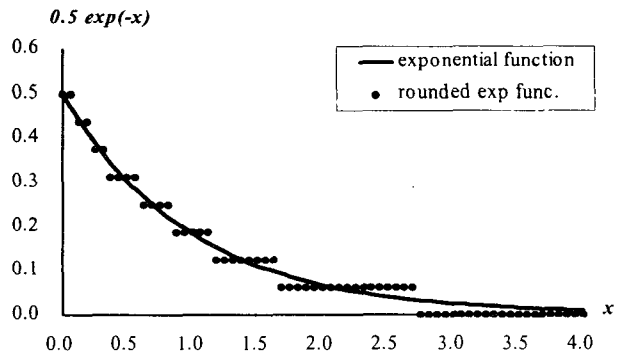


그림 11. 단순화된 exponential function
Fig. 11 Graph of rounded exponential function

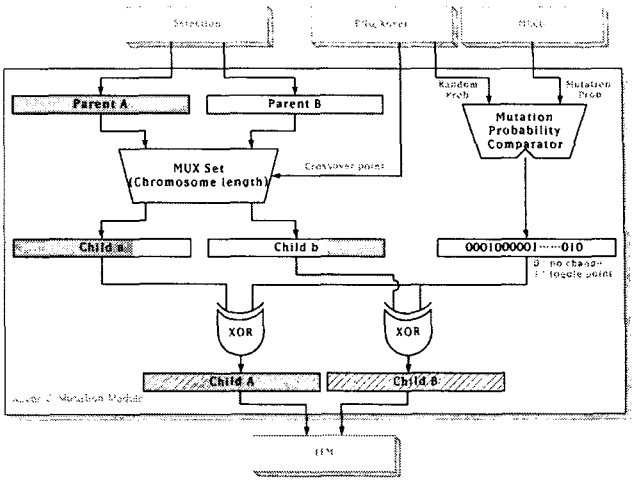


그림 12. Crossover 및 Mutation의 Block Diagram
Fig. 12 Block diagram of Crossover & Mutation Module

순차적으로 실행되도록 하여 동작 속도의 손실 없이 하나의 모듈로 모두 처리하도록 설계하였다.

구현한 MRCU는 빠른 계산 속도와 최소의 자원소모를 위해 정밀도를 조절하여 모든 연산 과정에서 2^{-4}이 유효숫자로 적용되었고, Lookup table, Combinational logic 구조의 5□5 fixed-point multiplier와 10/5 divider가 사용되어 하드웨어 지향의 최적화된 구조로 설계되었다.

3.1.4 Crossover & Mutation module

그림 12는 교차(Crossover)와 돌연변이(Mutation) 연산의 block diagram이다. 이 모듈은 두 가지 유전 연산이 단계적으로 수행되도록 설계되었다.

개체들 간의 유전정보의 교환을 위한 교차 연산은 단순 교차(1-point crossover), 두 점 교차(2-point crossover)와 균일 교차(uniform crossover) 방법 등 세 가지로 구현하였다. 그림 12는 단순 교차인 경우이며, 난수 발생기로부터 교차점을 받아 두 개체의 bit index와 비교하여 index가 작은 경우 bit를 서로 교환함으로써 새로운 개체를 생성하는 것이다.

돌연변이 방법에 있어서도 단순 돌연변이(single-point mutation) 및 다중 돌연변이(multi-point mutation)가 발생하도록 설계하였다. 난수 발생기로부터 받은 임의의 확률과 돌연변이 비율을 개체의 비트 수만큼 비교하여 0과 1로 구성된 스트링을 생성한다. 이 스트링과 교차연산을 마친 개체와의 각 bit를 XOR연산을 통해 최종 자손을 만들어 낸다.

3.1.5 Fitness evaluation module(FEM)

교차와 돌연변이와 같은 유전 연산자에 의해 생성된 새로운 개체에 대해 해의 적합도를 평가하는 부분으로서 주어진 문제에 대한 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 한 개 이상의 모듈로 구성한

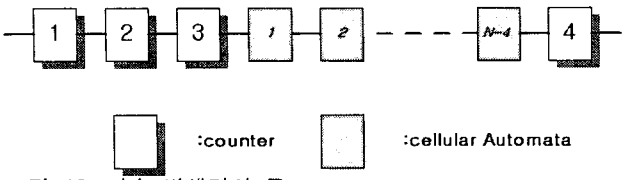


그림 13. 난수 발생기의 구조

Fig. 13 Architecture of pseudo-random number generator

다. 내부 테스트 모듈로 set covering problem[32]이 구현되어 있으며, 복잡한 연산의 경우 외부의 계산 모듈을 이용할 수 있도록 인터페이스가 구현되어 있다.

3.1.6 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.

그림 13에 나타낸 바와 같이 주기 및 초기 값의 문제를 개선하기 위해 4 bit counter를 이용하여 CA의 경계조건을 변화시켜 초기 값에 관계없이 난수를 발생하도록 설계되었으며 maximum length cycle를 증가시켰다. 난수발생기에 사용된 CA는 시뮬레이션의 성능에서 rule 150(식(6))과 거의 비슷한 rule 90(식(5))[33]에 의해 설계되었다.

$$a_i(t+1) = a_{i-1}(t) \oplus a_{i+1}(t) \quad (5)$$

$$a_i(t+1) = a_{i-1}(t) \oplus a_i(t) \oplus a_{i+1}(t) \quad (6)$$

3.2 Set Covering Problem

Set Covering Problem은 잘 알려진 조합의 최적화문제로서 일반적으로 문제 푸는 방법이 존재하지 않는 NP 문제의 일종인 non-unicost set covering problem이다. Set covering problem은 m-row, n-column을 가진 zero-one 행렬의 rows를 최소의 비용의 columns를 이용하여 covering 하는 문제이다.[32] 이는 주로 승무원 스케줄링, 비상 설비의 위치 결정, 조립 라인 평형 및 boolean 표현 간략화 등과 같은 많은 실제적인 응용문제에 적용된다. 본 연구에서는 19

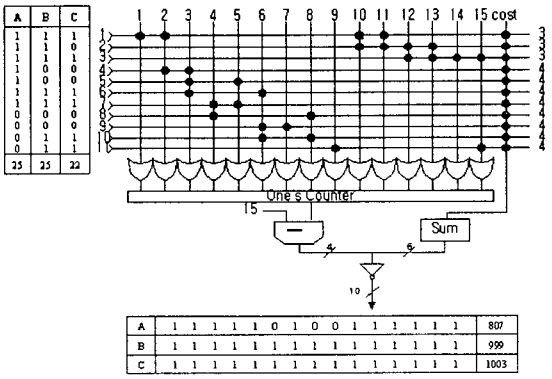
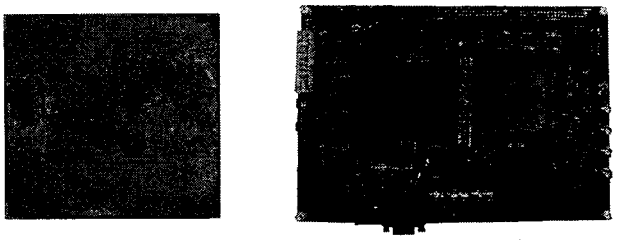


그림 14. Non-unicost set covering problem

Fig. 14 Non-unicost set covering problem



(a) XCV2000E FPGA device (b) EHW board

그림 15. 유전자 알고리즘 프로세서의 원형

Fig. 15 The prototype of Adaptive GAP

rows와 63 columns로 구성된 non-unicost set covering problem이 적용되었고 이는 모든 63 columns를 만족시킬 수 있는 rows의 요소들을 최소 크기의 집합을 찾는 것으로써 적용된 문제의 개념적인 구성의 형태는 그림 19에 나타낸 것과 같다. 이는 FEM의 하위 모듈로 구성되어 Test function으로 사용되었다.

3.3 실험 및 결과

3.3.1 Prototype 및 얼굴인식 진화 시스템

그림 15는 구현된 Adaptive GAP가 download된 Xilinx 사의 XCV2000E FPGA device와 얼굴인식을 위한 EHW board의 모습니다. 사용된 XCV2000E FPGA는 약 200만 게이트급의 device로서, 시스템에서는 Adaptive GAP와 MCU가 함께 FPGA에 탑재되고, 얼굴인식 프로그램과 EHW H/W Filter block이 추가되어 얼굴인식 시스템으로 동작한다.

3.3.2 시뮬레이션 결과

그림 16은 GAP내부 테스트 모듈인 Set Covering Problem을 ModelSim으로 시뮬레이션한 결과이다. 19비트의 염색체 64개로 개체군이 구성되고 적합도는 13비트로 표현되어, 32비트 x 64개 크기의 개체군이 메모리에 저장되었다. 이는 최적해 '0609b'와 최적의 적합도 '1fdf'의 16진수 값을 갖는다.

그림 17은 Adaptive GAP의 합성 결과로서 각 모듈들이 블록으로 표시되어 있고 클럭 게이팅을 위한 AND 게이트가 각 모듈의 입력에 연결 되어 있음을 보여준다.

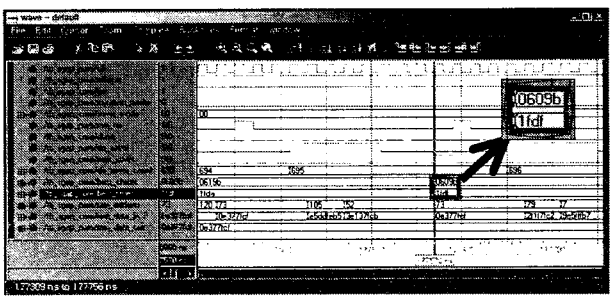


그림 16. Adaptive GAP의 결과 파형

Fig. 16 Waveform of Adaptive GAP

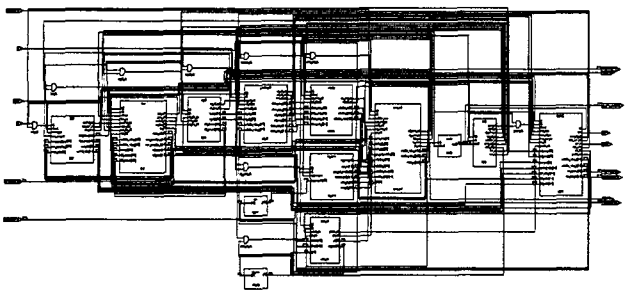


그림 17. Adaptive GAP의 합성도
Fig. 17 Synthesis of Adaptive GAP

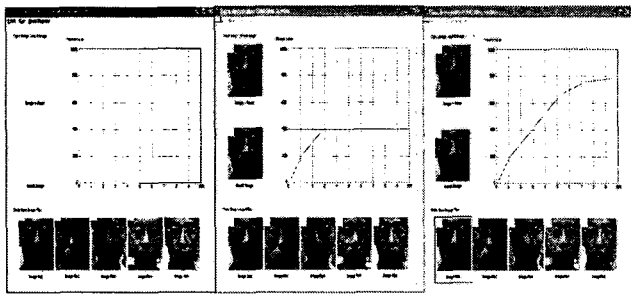


그림 18. 얼굴인식 진화 시스템의 진화 과정
Fig. 18 Evolution process of evolutionary system for face recognition

그림 18은 얼굴인식 진화 시스템이 진화과정을 통해 왜곡된 이미지의 잡영을 제거하여 일치되는 인물을 정확히 찾는 과정을 보여준다. 이 시스템에 사용된 Adaptive GAP는 이미지에 관한 정보의 표현을 위해 240비트의 chromosome이 사용되었고, 전체 시스템은 50MHz의 클럭 속도로 동작하였다.

Ⅳ. 결 론

본 논문에서는 다양하고 종합적인 분야로의 실제적인 적용을 위해 새로운 하드웨어 지향의 적응형 유전자 알고리즘을 제안하여 소프트웨어로 성능을 검증하고 Verilog HDL을 이용하여 하드웨어로 구현하였다.

제안된 알고리즘은 이산세대 모델에 비해 개체군의 저장을 위해 사용되는 메모리를 반으로 줄일 수 있는 정상상태 모델 내에서 보다 복잡한 적용함수에 대해 수렴 세대를 단축시키고, 수렴 정확도를 향상시킬 수 있는 방법을 제안하여, 유전자 알고리즘 내에서의 중요한 파라미터 중 하나인 돌연변이율을 진화에 따라 적응하도록 설계하였다.

제안된 알고리즘은 연산 시간에 있어서 기존의 가장 우수한 속도 지향의 modified survival-based 유전자 알고리즘에 비해 수렴 속도면의 성능 손실 없이 수렴 정확도면에서 작게는 5%, 크게는 100%가 넘는 성능의 향상이 있었고, 파이프라이닝과 병렬 구조를 이용하여 돌연변이율 조절 유닛으로 인한 추가적인 시간 소모는 없도록 구현하였다. 덧붙여 여러 가지 문제에 대한 보편성을 위해 1-point, 2-point 및 균일 교차와 다양한 돌연변이를 갖도록 구현하였고, 이를 통해 다양한 해의 생성, 유지 및 수렴이 빠른 해의 도달이 가능토록 하였다.

연구된 적응형 유전자 알고리즘 프로세서는 빠른 연산 시간을 바탕으로 환경에 민감하고 실시간 처리가 요구되는 분야, 즉 진화형 하드웨어의 중앙연산처리 장치 및 실시간 처리가 요구되는 최적화 문제, Robot navigation, 영상, 음성 및 문자인식, 컴퓨터 비전 분야 등 인공 지능의 핵심 부품으로 사용될 것이며, 앞으로 더 많은 응용 분야가 창출될 것이다.

감사의 글

본 연구는 2003년도 산업자원부 차세대신기술개발사업의 수퍼지능칩 및 응용기술 개발 과제의 지원에 의하여 이루어진 연구로서, 관계부처에 감사 드립니다.

참 고 문 헌

[1] 장병탁, "인공 생명과 진화 알고리즘" , 전자공학회지, 제24권 제3호, pp. 51-60., 1997년 3월.

[2] D. E. Goldberg, Genetic Algorithm in search, Optimization, and Machine Learning, Addison-Wesley, 1989.

[3] J. H. Holland, "Adaptation in Natural and Artificial Systems", Univ. of Michigan Press, Ann Arbor, 1975.

[4] L. J. Fogel, A. J. Owens, and M. J. Walsh, "Artificial Intelligence through Simulated Evaluation", New York: John Wiley & Sons, 1966.

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

[6] J. J. KIM, "Implementation of a High-Performance Genetic Algorithm Processor for Hardware Optimization", IEICE TRANSACTIONS on Electronics, Vol.E85-C, No.1, pp. 195-203, January 2002.

[7] K. Dejong, " An analysis of behavior of a class of genetic adaptive system", Ph.D Thesis, University of Michigan, 1975.

[8] J. H. Holland, "Concerning Efficient Adaptive System", 1962.

[9] S. D. Scott, A. Samal and S. Seth, "HGA :A hardware-based genetic algorithm", Proc. ACM/SIMDA 3rd International Symposium on FPGA, pp. 53-59., 1995.

[10] P. Graham, B. Nelson, "A hardware genetic algorithm for the traveling salesman problem on Splash2" 5th International Workshop on Field-Programmable Logic and its Applications, pp. 352-361., August 1995.

[11] M. Salami, "Multiple genetic algorithm processor for hardware optimization", Proc. First International Conference, ICES96 Evolvable System: From Biology to Hardware, pp. 249-259., October 1996.

[12] M. Tommiska, J. Vuori, "Implementation of genetic

algorithms with programmable logic devices", Proceeding of the 2NWGA, August 1996.
[13] I. Kajitani, T. Hoshino, D. Nishikawa, H. Yokoi, S. Nakaya, T. Yamauchi, T. Inuo, N. Kajihara, M. Iwata, D. Keymeulen, T. Higuchi, "A gate-level EHW chip: Implementing GA operations and reconfigurable hardware on a single LSI", Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science 1478, pp. 1-12., Springer Verlag, 1998.
[14] N. Yoshida, T. Moriki and T.Yasuoka, "GAP: Genetic VLSI processor for genetic algorithm", Second International ICSC Symp. on Soft Computing, pp.341-345., 1997.
[15] Shin'ichi Wakabayashi et al., "GAA: A VLSI genetic algorithm accelerator with on-the-fly adaptation of crossover operators", ISCAS 98, 1998.
[16] K. Dejong, " An analysis of behavior of a class of genetic adaptive system", Ph.D Thesis, University of Michigan, 1975.
[17] J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," IEEE Trans. Systems, Man, Cybern., vol. 16, no. 1, pp. 122 - 128, 1986.
[18] J. Schaffer, R. Caruana, L. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in Proc. 3rd Int. Conf. Genetic Algorithms, J. D. Schaffer, Ed., San Mateo, CA: Morgan Kaufmann, 1989, pp. 51 - 60.
[19] J. Hesser and R. Männer, "Toward an optimal mutation probability for genetic algorithms," in Proc. 1st Conf. Parallel Problem Solving from Nature, (Lecture Notes in Computer Science, vol. 496), H.-P. Schwefel and R. Männer, Eds. Berlin, Germany: Springer-Verlag, 1991, pp. 23 - 32.
[20] T. Fogarty, "Varying the probability of mutation in the genetic algorithm," in Proc. 3rd Int. Conf. Genetic Algorithms, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989.
[21] T. Bäck and M. Schütz, "Intelligent mutation rate control in canonical genetic algorithms," in Foundations of Intelligent Systems (Lecture Notes in Artificial Intelligence, 1079), Z. Ras and M. Michalewicz, Eds. New York: Springer-Verlag, 1996, pp. 158 - 167.
[22] R. Hinterding, Z. Michalewicz, and A. E. Eiben. Adaptation in Evolutionary Computation: A Survey. In Proceedings of the 4th IEEE International Conference in Evolution Computation. IEEE Press, pp.65~69, 1997.
[23] Srinivas M. and Parnaik L. M., " Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms" IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 4, pp. 656~667, April, 1994.
[24] J. E. Smith, "Operator and parameter adaptation in Genetic Algorithm", Soft computing, pp. 81~87, 1997.
[25] A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms", IEEE Transaction on Evolutionary Computation, Vol. 3, No. 2, pp.124~141, July, 1999.
[26] E. K. P. Chong and S. H. Zak, "An Introduction to Optimization", John Wiley & Sons, Inc., N.Y., 1996.
[27] J. Matyas, "Random Optimization", Automation and Remote Control, Vol. 26, pp. 246~253, 1965.
[28] L. Bolc and J. Cytowski, "Search Methods for Artificial Intelligence", Academic Press, London, 1992.
[29] Z. Michalewicz, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlage, berlin Heidelberg, 1996.
[30] Daniel Zwillinger, "Standard Mathematical Tables and Formulae" (31st ed.), Chapman & Hall/CRC Press, 2003.
[31] H. Mühlenbein, D. Schomisch and J. Born. "The Parallel Genetic Algorithm as Function Optimizer ". Parallel Computing, 17, pages 619-632, 1991.
[32] J. E. Beasley and P. C. Chu, "A genetic algorithm for the set covering problem", EUROPEAN JOURNAL OF OPERATIONAL RESEARCH, pp. 392-404, Vol. 94 No. 2, October 1996.
[33] Peter D. hortensius et al., "Parallel Random Number Generation for VLSI System Using Cellular Automata", IEEE Trans. on Computers, Vol.38, No. 10, pp. 1466-1473, October 1989.

저 자 소 개

정 석 우 (鄭 晳 宇)

1976년 8월 7일생. 2002년 인하대 전자전기컴퓨터공학부 졸업. 2002년~현재 동 대학원 정보통신공학과 석사 졸업

Tel : 032-874-1663
Fax : 032-864-1664
E-mail : jsw0807@korea.com

김 현 식 (金 賢 植)

1976년 9월 3일생. 2002년 인하대 전자전기컴퓨터공학부 졸업. 2002년~현재 동 대학원 정보통신공학과 석사 졸업

Tel : 032-874-1663
Fax : 032-864-1664
E-mail : sapper3@hotmail.com

김 동 순 (金 東 淳)

1972년 8월 11일생. 1997년 인하대학교 전자재료공학과 학사 졸업. 1999년 동 대학원 전자재료공학과 석사 졸업. 1999년~현재 전자부품연구원 DMB개발사업단 선임연구원

Tel : 031-6104-264
Fax :031-6104-048
E-mail : dskim@keti.re.kr

정 덕 진 (鄭 德 鎭)

1948년 2월 8일생. 1970년 서울대 전지공학과 졸업. 1984년 미국 Utah State University(석사)(Electrical Eng.) 1988년 미국 Utah State University(공박) (Electrical Eng.) 1989년~현재 인하대 정보통신공학과 교수

Tel : 032-874-1663
Fax : 032-864-1664
E-mail : djchung@inha.ac.kr