# 유전자 알고리즘의 유전 연산자 구현

유 명 근, 송 기 용
충북대학교 컴퓨터공학과

# Implementation of a Genetic Operator for Genetic Algorithm

Myoung-Keun You, Gi-Yong Song
mkyou77@chungbuk.ac.kr, gysong@chungbuk.ac.kr
Dept. of Computer Engineering, Chungbuk National University

## 요 약

유전자 알고리즘(genetic algorithm, GA)은 자연적 진화과정에서 생존 경쟁 측면의 가장 적합한 메커니즘이다. GA를 소프트웨어로 수행하는데 큰 지연시간은 필수적이기 때문에 하드웨어 설계를 이용하여 알고리즘 실행 속도를 증가시키기 위한 많은 연구가 진행되어 왔다. 본 논문에서는 염색체의 임의의 유전인자를 기준으로 입력 받은 염색체에 대하여 GA 연산을 수행하는 유전 연산자를 설계한다. 설계된 디자인을 ARM 코어와 PLD로 구성된 Altera사의 Excalibur칩에 구현하여 동작을 검증하였다.

## I. 서 론

GA[1-5]는 범용의 최적화 기술로 간주되어오고 있으나, 크고 복잡한 문제에 대하여 GA를 적용하는 것은 최적화 과정에 소요되는 큰 지연 시간 때문에 소프트웨어 구현 측면에서 어려운 과제이다. 예로 탐색 공간이 크고 실시간 처리가 필요한 곳에 GA를 적용하는 경우를 들 수 있다. GA에는 선택, 교배, 돌연변이 연산이 있으며, GA를 소프트웨어로 수행하는데 큰 지연 시간은 필수적이기 때문에 하드웨어 설계를 이용하여 알고리즘 실행 속도를 증가시키기 위한 많은 연구가 진행되어왔다. 즉, 소프트웨어 구현이 복잡하여 GA를 적용하기 힘든 문제에 하드웨어로 구현하는 것은 바람직할 것이다.

반도체 산업 및 집적회로 기술의 발전 및 칩 집적도가 높아짐에 따라 과거에 보드로 만들던 시스템 기술이 이제는 하나의 반도체에 시스템을 집적화할 수 있게 되었다[7]. 인터넷 시대의 등장과 함께 전자회로의 임베디드 시스템화, 소형화, 저전력화, 인터넷화되고 있다. 또한 반도체 공정 기술의 발전은 하나의 칩 안에 들어갈 수 있는 회로 소자의 개수를 크게 증가시켜 주었다. 과거에 별개의 칩으로 존재하던 메모리, 마이크로프로세서, 코프로세서 등이 요즘에는 하나의 칩 안으로 들어갈 수 있게 되었다. 이와 같이 모든 구성요소를 하나의 칩에 집적하는 것을 SoC라고 하며, 하나의 칩 안에 들어가는 개개의 구성요소를 IP(intellectual property)라고 부른다. 시스템 구성에 필요한 핵심적인 기능을 하나의 칩 위에 구현하는 SoC는 더 이상 메모리, 비메모리라는 낡은 영역 구분을 허용하지 않는다. 지금까지는 IP 검증을 위하여 하드웨어 기술자가 설계를 하였지만 SoC가 시작되면서 하드웨어 및 소프트웨어를 동시에 설계하고 회로를 어떻게 검증하느냐가 중요한 문제로 대두되고 있다.

본 논문에서는 염색체의 임의의 유전인자를 기준으로 입력 받은 염색체에 대하여 GA 연산을 수행하는 유전 연산자를 설계한 후, 설계된 디자인을 SoC 설계 및 IP 검증 시스템인 SoCMaster XP-100에 구현하였다. SoC Master XP-100은 SoC 설계 칩으로 ARM 코어와 PLD로 구성된 Altera사의 Excalibur칩[8-9]을 내장하고 있다.

## II. 유전자 알고리즘 및 유전 연산자

### 1. 유전자 알고리즘

GA[1-5]는 진화 과정에서 유도된 탐색 방법으로, 염색체와 유사한 자료구조를 사용하여 해 공간을 부호화하며, 부호화한 자료 구조에 GA 연산자를 적용하여 염색체들을 진화시킨다. GA는 자연 세계의 진화 과정을 컴퓨터 상에서 시뮬레이션 함으로써 복잡한 실세계의 문제를 해결하고자 하는 계산모델이며, 구조가 간단하고 방법이 일반적이어서 응용범위가 매우 넓으며, 특히 적응적 탐색과 학습 및 최적화를 통한 공학적인 문제의 해결에 많이 이용되고 있다.

### 2. GA 연산자

(1) 선택

자연에서 환경은 더 적합한 생물들이 더 높은 생존 기회를 갖도록 이들의 생존을 조절하는 기능을 가진다. 즉, 선택은 적자생존 또는 자연도태 현상을 모방하려는 인위적인 메커니즘이다. 이를 알고리즘으로 구현한 선택 연산자는 적합도 값을 기반으로 모집단 내의 개체들 중에 교배를 위해 부모가 될 개체

를 선택한다. 이러한 선택은 전체 집단을 강하게 해주는 특징을 가진다.

룰렛 휠 선택은 확률적인 선택 방법으로 룰렛 휠 둘레의 임의의 위치에 하나의 표지를 위치시키고, 룰렛 휠의 슬롯 면적을 개체집단내의 각 개체의 적합도에 비례하게 할당한 후, 룰렛 휠을 회전시켜 정지했을 때 표지가 가리키는 슬롯의 개체를 선택한다. 룰렛 휠 선택은 한번 룰렛 휠의 회전으로 하나의 개체를 선택하게 된다.

(2) 교배

교배는 탐색공간 상의 가능한 새로운 점을 찾기 위하여 개체 집단으로부터 부모 염색체 쌍을 임의로 선택하고, 교배점을 이용하여 비트들을 서로 교환 결합함으로써 자손을 생성한다. 즉, 교배란 부모가 갖는 유효한 정보를 결합시켜 새로운 개체를 만들어내는 것으로, 개체를 다양화하기 위하여 사용된다. 유전자를 절단하는 교배점은 염색체 내에서 임의로 선택된다. 이러한 연산은 교배된 자손의 수가 부모 집단의 크기와 같을 때까지 반복된다.

(3) 돌연변이

진화가 계속되는 동안 선택과 교배 연산자는 집단을 더욱 강하게 해주고 이로 인하여 염색체들은 서로 닮아가게 된다. 이러한 현상은 세대 말기에는 바람직하지만 세대 초기에 발생하게 되면 유전자의 다양성 결핍으로 지역해에 빠지게 되는 요인이 된다. 선택과 교배 연산자는 지역해로부터 벗어나게 하는 메커니즘을 가지고 있지 않다. 다시 말하면 집단 내 염색체들의 특정위치 비트가 모두 같게 되면 선택과 교배 연산자는 이를 변경할 수 없게 된다. 이러한 원치 않는 해로부터 벗어나기 위한 메커니즘 중의 하나가 돌연변이이다. 초기 세대에서 모든 염색체의 특정 비트가 고정되는 것을 방지해주고 또한 탐색영역을 확대해 주기도 한다.

## Ⅲ. 유전 연산자 설계

### 1. GA 시스템

GA 시스템의 구조는 그림 1과 같이 3개의 구성요소인 코어, GA연산을 수행하는 PLD, 그리고 메모리로 이루어진다.

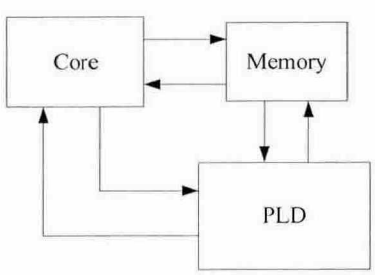



그림 1. GA-SoC 시스템 구조

코어는 적합도 평가 작업을 수행하는 범용의 프로세서이며 GA 연산을 수행하는 PLD는 코프로세서로 동작한다. 메모리는 코어에 대한 지원 및 염색체 저장을 위해 사용된다. 그림 2는 GA의 흐름도이며, GA 연산에 필요한 데이터를 PLD 내에 설계된 레지스터 파일에서 읽어들여 룰렛 휠 선택 연산, 교배 연산, 돌연변이 연산을 수행한 후, 생성된 데이터를 레지스터 파일에 기록한다.

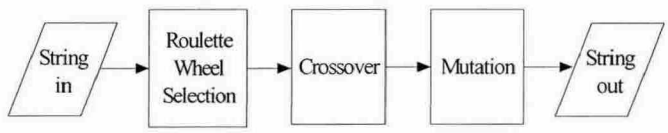



그림 2. GA 연산을 수행하는 회로의 데이터 흐름

염색체는 그림 3과 같이 크게 유전자 필드와 적합도 필드로 구성된다. 적합도 필드는 코어에서 염색체의 적합도 평가 작업에서 사용되는 적합도 값을 포함하고 있으며 염색체 필드에 연결된다.


그림 3. 염색체 구조

### 2. 룰렛 휠 선택 연산자 설계

룰렛 휠 선택 연산자는 [6]에서 설계한 것 같이 그림 4와 그림 5와 같은 구조를 가진다.

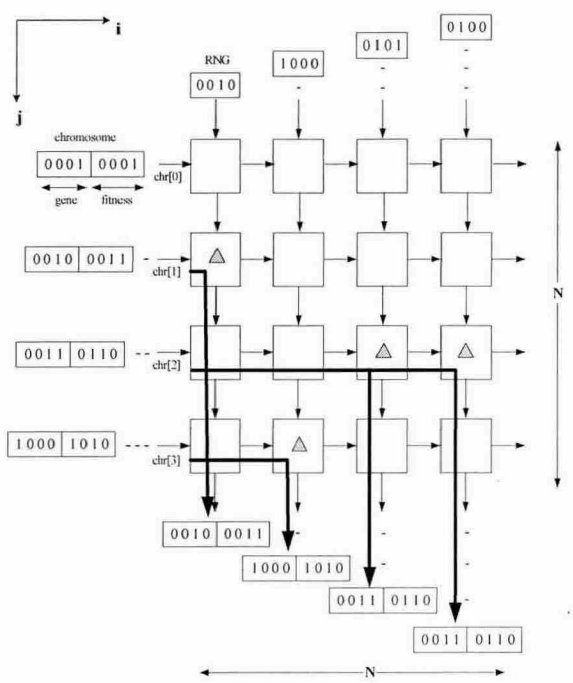



그림 4. 룰렛 휠 선택을 위한 시스톨릭 어레이

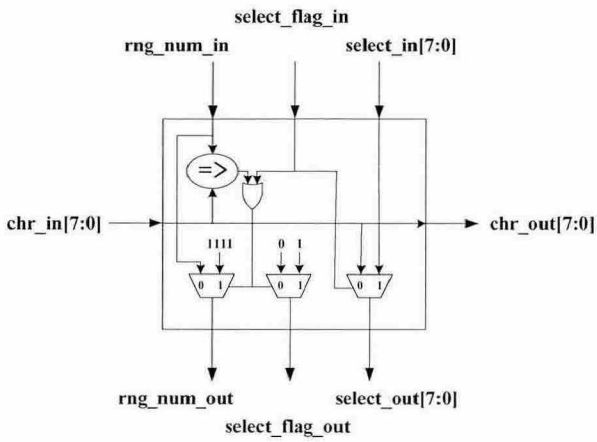



그림 5. 셀의 내부 구조

그림 4에서 $N$개의 열은 $N$번의 룰렛 휠 회전을 의미한다. 각 열의 어레이[7]는 룰렛 휠의 회전에 해당하는 임의의 수를 입력으로 받는다. 선택된 염색체의 경로는 굵은 선으로 표시되었으며 4개의 회색 삼각형은 염색체가 선택된 시점을 나타낸다.

## 3. 교배 연산자 설계

4개의 염색체 입력에 대하여 룰렛 휠 선택 연산을 수행하면 2쌍의 부모 염색체가 생성된다. 교배 연산자는 2쌍의 부모 염색체와 1-점 교배 연산의 교배점에 해당하는 CP(cut-point)를 입력으로 받으며, 교배 연산자를 구성하는 셀은 0부터 $n$(유전인자 수) 사이의 임의의 교배점에 대하여 병렬적으로 교배 연산을 하는 회로와 생성되는 자식 염색체들 중 올바른 자식 염색체를 선택하는 멀티플렉서로 구성된다.

교배 연산자와 셀의 내부구조를 그림 6과 그림 7에 각각 보인다. 그림 7은 염색체의 유전자 필드를 구성하고 있는 유전인자 수가 6인 경우이다.

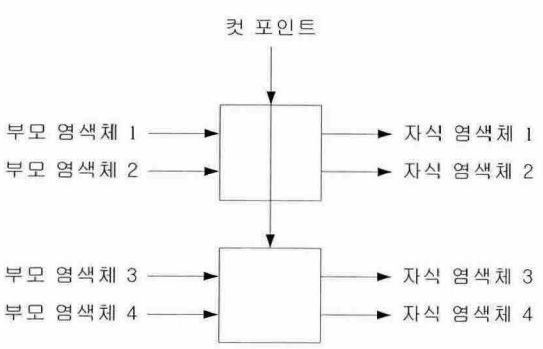



그림 6. 교배 연산자

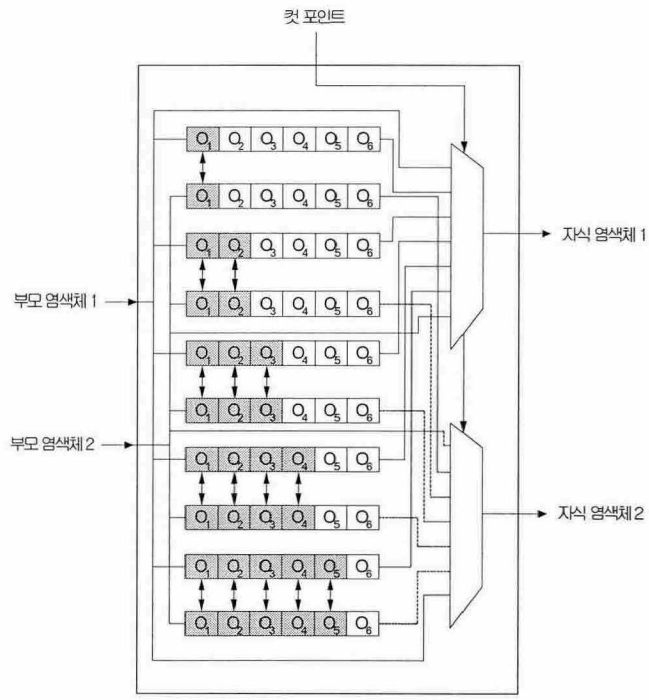



그림 7. 교배 연산자 셀의 구조

## 4. 돌연변이 연산자

교배 연산자로부터 생성된 모든 자식 염색체들에 대하여 정해진 염색체 특정 부분을 invert 값을 취함으로써 돌연변이 연산을 수행한다. 단, 입력으로 받는 돌연변이 확률과 임의의 수를 비교하여 돌연변이 확률이 임의의 수보다 클 경우만 수행된다. 돌연변이 연산자와 셀의 내부 구조를 그림 8과 9에 각각 보인다.

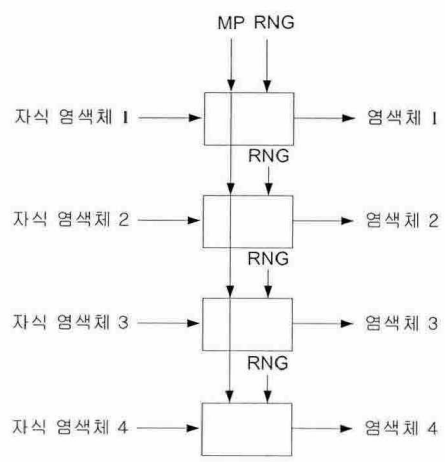



그림 8. 돌연변이 연산자

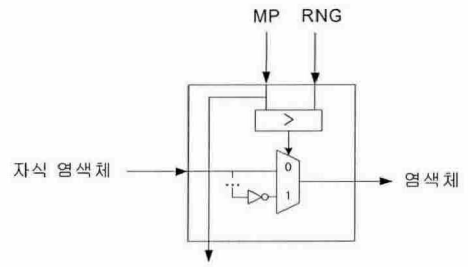



그림 9. 돌연변이 연산자 셀의 내부 구조

## Ⅳ. 시뮬레이션 및 구현

III절에서 설계한 유전인자들을 Verilog-HDL[10]을 이용하여 디자인하였다. *N*, *F*, *G*가 각각 4, 2, 6일 때, 설계된 디자인을 SoC 설계 및 IP 검증 시스템인 SoCMaster XP-100에 구현하였다. *F*와 *G*는 16진수로 표현되었다. EDA 툴인 Quartus를 사용하여 SoC 설계 시스템의 PLD에 구현할 디자인의 스키메틱은 그림 10과 같다.

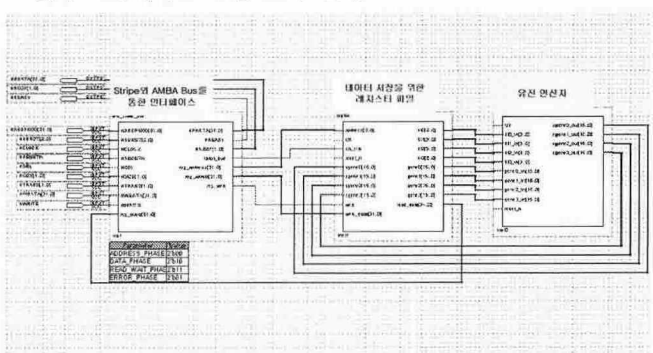


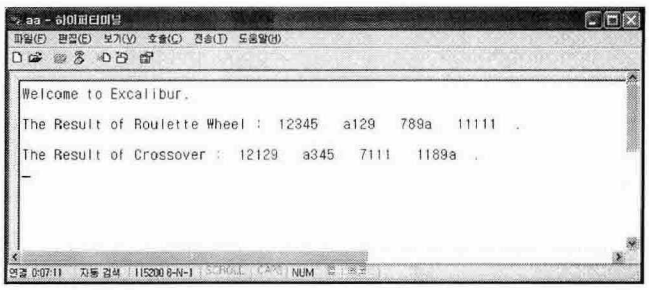
그림 10. SoC 설계 시스템의 PLD에 구현될 디자인 스케메틱

그림 10의 오른쪽 블럭은 룰렛 휠 선택, 교배, 돌연변이 연산을 차례로 수행하는 부분에 해당하며, 중간 블록은 유전 연산자의 입력 및 출력 데이터를 저장하기 위한 레지스터 파일이다. 그리고 왼쪽 블록은 Excalibur 칩의 ARM 프로세서를 포함하고 있는 Stripe와 AMBA 프로토콜을 사용하여 상호 커뮤니케이션 할 수 있도록 AMBA 인터페이스를 설계한 부분이다.

그림 11은 동작이 검증된 디자인을 Excalibur칩의 PLD에 다운로드하여 표 1과 같은 데이터를 주었을 때의 칩 상의 동작을 검증하는 과정을 보여준다.

표 1. 데이터 입력 값

| | | | |
|---|---|---|---|
| 염색체1 | 0x01234507 | rng1 | 0x07 |
| 염색체2 | 0x00789A18 | rng2 | 0x20 |
| 염색체3 | 0x00A12920 | rng3 | 0x16 |
| 염색체4 | 0x0111112F | rng4 | 0x2C |
| | | CP | 6 |




## Ⅴ. 결론

본 논문에서는 염색체의 임의의 유전인자를 기준으로 입력 받은 염색체에 대하여 GA 연산을 수행하는 유전 연산자를 설계한 후, 설계된 디자인을 SoC 설계 및 IP 검증 시스템인 SoCMaster XP-100에 구현하였다. SoC Master XP-100은 SoC 설계 칩으로 ARM 코어와 PLD로 구성된 Altera사의 Excalibur칩 을 내장하고 있다.

향후 연구 과제는 SoC 시스템을 설계시 타겟 보드에 운영체제를 적재한 후, 디바이스 드라이버 및 응용 프로그램을 작성하여 타겟 보드에서 실행시키는 것이다.

## 참 고 문 헌

[1] D.E.Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesely, 1989.

[2] M.Gen, R.Cheng, *Genetic Algorithms and Engineering Optimization*, Wiley-Interscience publication, 2000.

[3] S. M. Sait, H. Youssef, *Iterative Computer Algorithms with Applications in Engineering*, Computer Society, 1999.

[4] M.Gen, M.Cheng, *Genetic Algorithms and Engineering Design*, Wiley, New Your, 1997.

[5] P. Mazumder, E.M. Rudnick, *Genetic Algorithms for VLSI Design, Layout & Test Automation*, Prentice Hall, Inc., 1999.

[6] 이재진, 유명근, 송기용, "*유전자 알고리즘 선택기의 구현*", KISPS, Jul. 2005

[7] S.Y.Kung, *VLSI Array Processors*, Prentice Hall, 1988

[8] Altera Corporation, http://www.altera.com

[9] (주)휴인스, http://www.huins.com

[10] Bob Zeidman, *Verilog Designer's Library*, Prentice Hall, 1999