1999년도 대한전기학회 하계학술대회 논문집 1999. 7.19-21

# EHW를 위한 Genetic Algorithm Processor 구현

김진정, 김용훈, 최윤호, 정덕진
인하대학교 전자재료공학과 집적회로연구실

# Hardware Implementation of Genetic Algorithm Processor for EHW

Kim Jin Jung, Kim Yong Hun, Choi Yun Ho, CHung Duck Jin
Dep. of Electronic Materials & Devices Eng. INHA Univ.

**Abstract** - Genetic algorithms were described as a method of solving large-scaled optimization problems with complex constraints. It has overcome their slowness, a major drawback of genetic algorithms using hardware implementation of genetic algorithm processor (GAP). In this study, we proposed GAP effectively connecting the goodness of survival-based GA, steady-state GA, tournament selection. Using pipeline, parallel processing, handshaking protocol effectively, the proposed GAP exhibits 50% speed-up over survival-based GA which runs one million crossovers per second(1MHz). It will be used for high speed processing such of central processor of EHW, robot control and many optimization problem.

## 1. 서 론

최근 공학의 여러 가지 응용 분야 중 복잡한 제약성을 가진 대규모의 최적화 문제들이 많다. 일반적인 수학적인 프로그램 혹은 최적화 알고리즘의 조합을 이용하여 짧은 연산 시간 안에 그러한 문제들에 대해 최적의 해를 얻기는 매우 어렵다. 그 때문에 진화 알고리즘(Evolutionary Algorithm, EA) 중에 유전자 알고리즘(Genetic Algorithm, GA)이 일반적으로 그러한 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(GAP)가 요구되어 진다.

기존의 유전자 알고리즘은 소프트웨어 기반의 응용 분야에 한정되어 이용되었지만 본 연구의 genetic algorithm processor(GAP)는 VLSI 구현하는데 있어 하드웨어 지향의 유전자 알고리즘을 기반으로 연산 속도가 크게 향상되어 Evolvable Hardware(EHW)와 같은 고속 처리가 요구되어지는 응용 분야에 적합하다.

본 연구는 유전자 알고리즘의 하드웨어 구현을 위하여 기존의 알고리즘의 장· 단점 분석 및 연구 후에 수정된 survival-based GA를 바탕으로 기존의 tournament selection 방법의 장점을 접목한 알고리즘을 MATLAB을 이용한 시뮬레이션으로 검증하였다. 이를 VHDL coding을 하여 FPGA가 내재된 PCIGEN10K board 상에서 GAP를 제작하였다.

## 2. 본 론

### 2.1 Evolvable hardware(EHW)

EHW는 그 주변의 환경과 상호 작용하여 동적이면서도 자율적으로 구조와 반응을 변하게 할 수 있는 하드웨어로서 언급되어 왔다.[1] EHW는 90년대 초기에 FPGA와 같은 쉽게 재구성 할 수 있는 하드웨어의 발달로 많은 관심을 끌어 왔다. 또 EHW는 패턴 인식으로부터 적응 제어와 같은 넓은 범위의 일을 수행할 수 있다는 것이 증명되어 왔다.[2] EHW 속의 중요한 motivation의 하나는 외부의 힘 없이도 인간과 같이 진화를 하기 때문에 자연으로부터 배운다는 것이다. 현재의 거의 모든 EHW는 주요 적응 메커니즘으로서 진화 알고리즘(Evolutionary Algorithm, EA)을 사용한다. 본 연구에서는 EHW의 구현을 위해 EA 중에서 유전자 알고리즘을 사용하였다. EHW는 보통 FPGA와 같은 programmable logic devices(PLD) 상에서 구현되었으며 이는 PLD의 구조와 기능이 변할 수 있는 architecture bit들의 집합으로 되어있기 때문이다. 즉 EHW는 재구성 하드웨어의 구성에 필요한 architecture bit을 유전자 알고리즘의 개체들로 보는 것이다. 따라서 원하는 결과가 나오지 않거나 외부환경의 변화가 발생하면 유전자 알고리즘이 수행됨으로서 하드웨어의 구조가 적합하게 재구성되는 것을 말한다. 즉 EHW는 재구성 하드웨어와 진화 알고리즘과의 결합체라고 할 수 있다.

### 2.2 유전자 알고리즘(Genetic Algorithm)

#### 2.2.1 유전자 알고리즘

유전자 알고리즘은 복잡한 최적화 작업 또는 학습을 이용한 구조에 자주 사용되는 자연도태의 유전적인 메커니즘에 기초한 탐색 알고리즘이다. 그 동작원리를 간단히 설명하면 유전자 알고리즘은 임의의 값으로 초기화된 개체는 상대적인 문제해결 능력에 따라 그 적합도(fitness)가 평가되며 적합도에 따라 다음 세대(generation)에 부모의 유전자가 복제(reproduction)되는 정도를 달리 함으로써 우성형질을 지닌 개체들은 열성 형질을 지닌 개체들에 비하여 더욱 많은 자식을 생성할 수 있도록 유도된다. 이러한 선택(selection)은 제공된 요소들 중에 임의로 다음 세대로 진화시킬 대상을 선택하게 되고, 이 선택 복제된 개체들은 선택된 두 인자들이 합쳐서 새로운 인자를 생성하게 되는 교차(crossover), 돌연변이(mutation) 등의 여러 가지 유전 연산자들에 의한 재결합(recombination)되어 다음 세대의 개체군을 생성한다. 마지막으로 개체군의 통계(population statistics)과정에서는 그 결과를 평가해 기록하고 최적화 된 결과를 판단하게 된다. 이와 같은 세대 교체는 원하는 수준의 해가 개체군 내에 존재하거나 또는 다른 종료 조건이 만족할 때까지 반복한다.

#### 2.2.2 제안된 유전자 알고리즘

제안된 유전자 알고리즘은 기존의 속도 지향의 survival-based GA의 무작위 선택에서 벗어나 steady-state GA에 가장 적합한 tournament selection을 통해 selection pressure를 높임으로서 빠른 해의 수렴을 지향하였다. 또 기존의 개체군의 통계 과정에서 개체 교체를 위해 비교하는 대상의 수를 증가시킴으로써 다양한 해의 생성 및 유지하도록 하였다.

그림1은 set coverage problem에 대한 survival GA

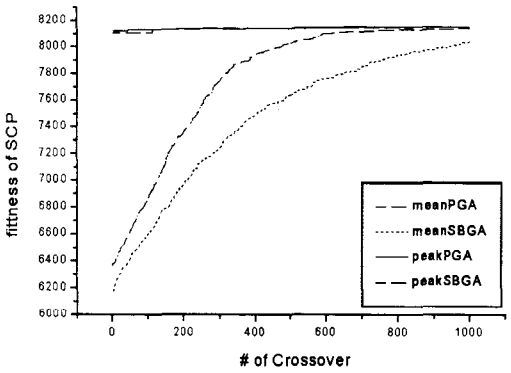


그림 1. Comparison of Survival-based GA and proposed GA

와 제안된 GA의 각 개체군의 적합도 평균과 각 개체의 적합도의 최대값을 나타내고 있다. 그림에서 보는 바와 같이 각 개체의 적합도의 최대값은 비슷하지만 그 평균에서 큰 차이가 나는 것을 알 수 있다. 이는 survival-based GA의 경우 selection pressure가 낮아 빠르게 수렴 가능성이 확률적으로 낮아지게 된다. 이에 비해 제안된 방법은 수렴할 가능성을 확률적으로 매우 증가시켰다.

## 2.3 Genetic Algorithm Processor

### 2.3.1 기존의 GA 하드웨어 연구

유전자 알고리즘의 병렬 분산 처리 및 하드웨어 구현 등의 방법 등을 통해 진화 연산에 있어서의 심각한 문제인 연산 시간을 줄이기 위한 방법이 시도되어 왔다. 기존의 하드웨어의 구현은 간단한 유전자 알고리즘(standard simple generation algorithm)에 기초한 Hardware-based Genetic Algorithm(HGA)를 제안한 Scott 등

| | Generational Steady-state | Selection | R.N.G. | Cross-over | Note |
|---|---|---|---|---|---|
| Stephen D. Scott(HGA) *Washington Univ.* | Steady-state | Roulette | CA | 1-point | BORG board |
| Paul Graham *Brigham young Univ.*[4] | Generational | Roulette | RNG | 1-point | Splash2 |
| M. Salami(GAP) *Victoria Univ.*[5] | Generational | Roulette | CA | 1-point | NeoCAD FPGA |
| *Tommoska & Vuori Helsinki Univ.*[6] | Steady-state | Round-robin fashion | LFSR | 1-point | Altera's FPGA |
| B. Shakleford (S-GA) *Mitsubishi E.C.* | Steady-state | Survival | CA | 1-point | Aptix AXB-MP3 FPGA |
| T. Higuchi, I. Kajitani[8] *Tsukuba* | Steady-state | Elitist Recombination | CA | Uniform | EHW chip |
| N. Yoshida(GAP) *Kyushu Univ.* | Steady-state | simplified tournament | CA | 1-point | SFL(HDL) |
| S. Wakabayashi (GAA) *Hiroshima Univ.* | Generational | Roulette & elitist strategy | CA | 2-point Uniform adaptive | 0.5㎛ CMOS technology |
| Our developed GAP | Steady-state | tournament | CA | 2-point, Uniform, adaptive | PCIGEN10K (Alteras FPGA) |

표 1. Previous & our developed work for hardware implementation of GA

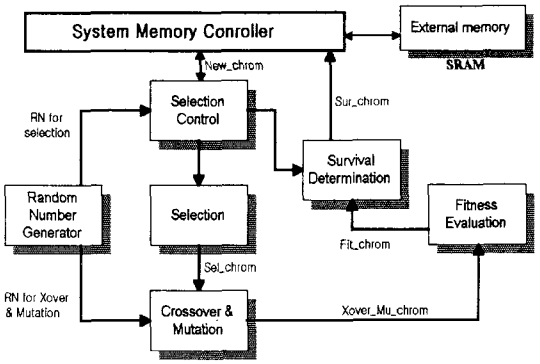


그림 2. Block diagram of GAP

에 의해 처음 시도되었다. Scott[3]의 연구에 뒤이은 유전자 알고리즘의 하드웨어 구현은 표1과 같다. Yoshida[9] 등은 GA Processor(GAP)라고 불리는 하드웨어 지향의 GA를 제안하였으며 이는 실질적으로 세대라는 개념을 가지지 않는 steady-state GA이다. Shakleford[7] 등은 steady-state GA와 어느 정도 비슷한 Survival-based GA를 제안하였으며 Kajitani 등은 LSI chip 상에서 EHW를 위한 GA 하드웨어를 제시하였다. 또 Wakabayashi[10] 등은 범용 GA 하드웨어로서 GA Accelerator(GAA) chip이라 불리는 GA의 VLSI를 구현하였다. 그러나 이들 연구는 FPGA 상에서 간단히 합성한 GA의 유전자 조작(genetic operation)과 소프트웨어와 비교한 속도의 향상에 의해 평가되어 왔다.

### 2.3.2 GAP의 하드웨어 구현

본 연구에서 제안된 GAP의 block diagram은 그림2와 같다. 제안된 GAP는 다음과 같이 크게 7개의 모듈로서 구성되었다. 이 중 메모리와 적합도 평가 모듈은 그 용량과 문제 의존성에 의해 외부에 따로 구성된다.

a. *Memory Module*: 개체군의 데이터(population)를 저장하는 부분으로서 외부의 메모리를 이용하여 구성되었으며 steady-state GA의 사용을 통해 세대 모델(generation model)의 유전자 알고리즘에 비해 사용되는 메모리의 사용을 반으로 줄일 수 있었다.

*b. System memory Controller*: 외부의 메모리 모듈과 내부의 연산 모듈과의 인터페이스 및 데이터의 입출력을 관리한다.

*c. Selection Module*: 제공된 요소들 중에 임의로 다음 세대로 진화시킬 대상을 선택하는 부분으로서 simplified tournament selection 방법을 이용하여 기존의 확률선택 모델에 비해 하드웨어 구현시 면적과 속도를 향상시켰다. 개체 선택 방법에 있어 무작위선택을 하는 survival-based GA에 비해 선택 pressure를 높임으로서 빠른 수렴이 가능하도록 설계되었다.

*d. Mutation & Crossover Module*: 여러 가지 문제에 대한 generality를 위해 개체의 교차 방법에 있어서는 다양한 해의 생성을 위해 전반부는 uniform crossover와 좋은 해의 유지를 위해 후반부의 2-point crossover와 같은 동적인 적응 교차 operation과 1-point, 2-point, uniform 등의 각각 교차가 모두 수행될 수 있도록 설계되었으며 돌연변이 방법에 있어서도 single point 및 multi point에서 돌연변이가 발생하도록 설계되었다. 또 쉬운 문제에 대해 면적은 작고 빠르게 진화할 수 있는 교차와 돌연변이가 결합된 형태의 유전자조작도 함께 설계되었다.

*e. Fitness Evaluation Module*: 교차와 돌연변이에 의해 생성된 새로운 개체에 대해 해의 적합도를 평가하는 부분으로서 절대적으로 문제에 의존적이다. 또한 그 연산 시간으로 인하여 병목 현상이 발생하는 부분으로서 제안

된 GAP에서는 두 개의 모듈로서 구성되었다.

*f. Random Number Generator(RNG)*: 개체군으로부터 개체의 선택 및 교차와 돌연변이의 발생가능성과 발생 위치를 결정하기 위해 사용된다. 난수의 평균이 피연산자의 반이 되지 않아서 정확한 계산을 기대하기 어려운 LFSR(linear feedback shift register)방법의 단점을 보완한 cellular automata(CA) 방법을 이용하여 설계되었으며 주기 및 초기값의 문제를 개선하기 위해 4 bit counter를 이용하여 CA의 경계를 만들어 초기값에 관계없이 난수를 발생하도록 설계되었다.

*g. Survival Determination Module*:현재의 개체보다 더 적합한 자손만이 생존하도록 결정하는 부분으로서 Most/Least-fit의 적절한 관계에 따른 survival condition 변화 및 method의 변화를 꾀할 수 있도록 설계되었으며 survival-based GA에 비해 비교하는 개체의 수를 늘림으로서 빠르게 최적의 값에 도달하도록 구성되었다.

GAP 내부에서의 효율적인 연산을 위해 내부적인 병목지점에 병렬 처리와 빠른 연산 처리를 위해 파이프라이닝을 사용하였으며 전체적인 데이터의 입출력은 handshaking 프로토콜을 기반으로 구성되었다. 또 개체군의 크기, 교차 확률, 돌연변이 확률, 최대 세대수 및 RNG의 초기값 등의 파라미터 값을 쉽게 변경할 수 있도록 설계하였다.

### 2.4 Fitness function

적합도 함수의 회로는 각 문제에 따라 그 구성이 달라지므로 일반적으로 FPGA와 같이 재구성이 가능한 하드웨어 상에서 구현이 된다.

본 연구에서는 제안된 하드웨어의 파라미터 추출 및 동작 특성을 파악하기 위해 동작 및 성능을 평가하기 위하여 두 가지 적합도 함수를 하드웨어로 구현하였다. 우선 첫 번째로 다음 식 (1)과 그림 3와 같은 매우 peak 이 많아서 local minima에 빠지기 쉬운 최적화 함수에 대해서 성능을 검증하였다.

$$f(x,y) = 21.5 + x\sin(4\pi x) + y\sin(20\pi y) \quad (1)$$

두 번째는 일반적으로 문제 푸는 방법이 존재하지 않는 NP 문제의 일종인 non-unicost set coverage problem(SCP)이다. SCP는 $m$-row, $n$-column, zero-one matrix 의 rows을 최소의 비용의 columns를 이용하여 coverage 하는 문제이다. 이 때 SCP는 19 rows와 63 columns로 구성되었다. 두 경우 모두 돌연변이 확률은 0.05미만, 세대수 각각 350과 250, 교차 횟수 각각 약 1600, 1000에서 해에 수렴할 수 있었다.

## 3. 결 론

Genetic Algorithm Processor(GAP)의 VHDL Coding에 의한 FPGA 구현은 수정된 survival-based GA를 바탕으로 기존의 tournament selection 및 steady-state GA의 장점을 접목하였다. 연산 시간에 있어 기존의 가장 우수한 survival-based GA의 초당 백만번의 교차를 수행(1MHz)하는데 비해 pipeline, parallel processing, handshaking의 효율적인 구현을 통해 50% 이상(over 1.5 million/second)의 속도 향상을 꾀할 수 있었다. 또 여러 가지 문제에 대한 generality를 위해 1point, 2point, uniform 및 동적인 crossover와 mutation을 통해 다양한 해의 생성, 유지 및 빠른 해의 도달을 하도록 구현되었다. 연구된 GAP는 빠른 연산 시간을 바탕으로 차후의 EHW의 중앙연산처리장치 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 적용 가능하다.

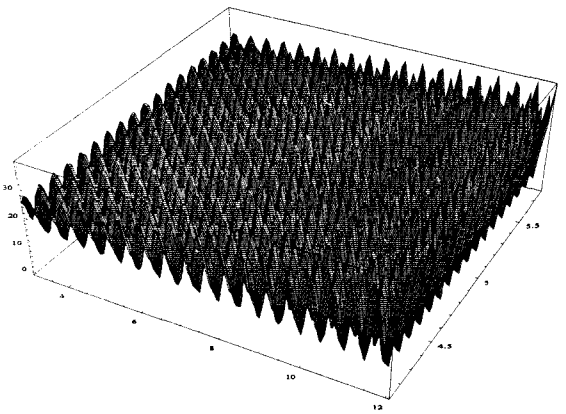

그림 3. Fitness function for GAP

**[참 고 문 헌]**


[1] X. Yao and T. Higuchi, "Promises and challenges of evolvable hardware", Proc. of the First International Conference on Evolvable Systems: From Biology to Hardware (ICES'96), Lecture Notes in Computer Science, Vol. 1259, pp.55--78. 1997.

[2] T. Higuchi *et. al.*, "Evolvable hardware and its applications to pattern recognition and fault-tolerant systems," in Towards Evolvable Hardware: The Evolutionary Engineering Approach, Lecture Notes in Computer Science, Vol. 1062, pp. 118--135, 1996.

[3] 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

[4] 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

[5] 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

[6] M. Tommiska, J. Vuori, "Implementation of genetic algorithms with programmable logic devices", *Proceeding of the 2NWGA*, August 1996

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

[8] 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 reconfigursble hardware on a single LSI", *Evolvable Systems: From Biology to Hardware, Lecture Notes in Computer Science 1478*, pp. 1-12, Springer Verlag, 1998.

[9] N. Yoshida, T. Moriki and T.Yasuoka, "GAP: Genetic VLSI processor for genetic algorithm", *1Second International ICSC Symp. on Soft Computing*, pp.341-345, 1997

[10] S. Wakabayashi *et al.*, "GAA: A VLSI genetic algorithm accelerator with on-the-fly adaptation of crossover operators", *ISCAS 98*, 1998