임베디드 하드웨어 유전자 알고리즘을 위한 실시간 처리 시스템

박세현* · 서기성**

Real-time processing system for embedded hardware genetic algorithm

Se-hyun Park* · Ki-sung Seo**

본 연구는 2003년도 안동대학교 연구비의 지원에 의하여 이루어진 연구로서, 관계부처에 감사드립니다.

요 약

임베디드 하드웨어 유전자 알고리즘을 위한 실시간 처리 시스템을 설계하였다. 제안된 시스템은 유전자 알고리즘의 기본 모듈인 selection, crossover, 및 mutation과 evaluation을 병행적으로 동작시키기 위해서 이중 프로세서로 구현하였다. 구현된 시스템은 두개의 Xscale 프로세서와 진화 하드웨어가 내장된 FPGA 로 구성되었다. 또한 본 시스템은 유전자 알고리즘의 기본 모듈 수행이 두 개의 프로세서에 자동으로 균등 배분되는 구조를 지니고 있어, 유전자 알고리즘 처리의 효율성을 극대화 할 수 있다. 제안된 임베디드 하드웨어 유전자 알고리즘 처리 시스템은 임베디드 리눅스 운영체제에서 수행되며 진화 하드웨어에서 실시간으로 처리된다. 또한 제안된 이중 프로세서의 각 프로세서 모듈은 동일한 구조로 가지고 있으므로 여러 개의 모듈을 직렬 연결하여 빠른 하드웨어 유전자 알고리즘 실시간 처리에 그대로 사용될 수 있다.

ABSTRACT

A real-time processing system for embedded hardware genetic algorithm is suggested. In order to operate basic module of genetic algorithm in parallel, such as selection, crossover, mutation and evaluation, dual processors based architecture is implemented. The system consists of two Xscale processors and two FPGA with evolvable hardware, which enables to process genetic algorithm efficiently by distributing the computational load of hardware genetic algorithm to each processors equally. The hardware genetic algorithm runs on Linux OS and the resulted chromosome is executed on evolvable hardware in FPGA. Furthermore, the suggested architecture can be extended easily for a couple of connected processors in serial, making it accelerate to compute a real-time hardware genetic algorithm. To investigate the effect of proposed approach, performance comparisons is experimented for an typical computation of genetic algorithm.

키워드

진화하드웨어, 하드웨어 유전자 알고리즘, 유전자 알고리즘, FPGA, 무어 머신

*안동대학교 **서경대학교

접수일자 : 2004. 10. 29

Ⅰ. 서 론

유전자 알고리(Genetic Algorithm)즘의 최대 단점은 알고리즘 처리 시간이 많이 요구되는 것이다. 유전자 알고리즘의 수행 속도를 증대시키기 위해 하드웨어 유전자 알고리즘이 등장하게 되었다.[1]-[4]

일반적으로 하드웨어 유전자 알고리즘은 주로 하드웨어나 FPGA 등으로 구현하여 처리 속도의 증대를 가져온다. 그러나 하드웨어로 구현되어 있기 때문에 다양한 유전자 알고리즘의 적용이 어려우며, 특히 적합도 함수의 변경이 힘든 단점을 가지고 있다. 따라서 하드웨어 유전자 알고리즘은 설계 초기에 특수 목적에 한정시켜 구현하며 범용으로 사용하기가 힘든 단점을 가지고 있다.

본 논문에서는 이중 프로세서를 이용하여 임베디드 하드웨어상에서 유전자 알고리즘을 처리하는 시스템을 설계하고자 한다. 설계된 시스템은 유전자 알고리즘이 기본 모듈인 selection, crossover, mutation 및 evaluation 처리를 두 개의 프로세서에 균등하게 배분함으로써 처리 속도를 효율적 높이는 한편, 기존의 하드웨어 유전자 알고리즘에서 상존하고 있는 하드웨어 적합도 평가를 소프트웨어로 구현할 수 있게 함으로서 하드웨어 유전자 알고리즘의 적용 범위를 확장하려 한다. 또한 설계된 이중 프로세서의 각 모듈은 동일한 구조로 된 다중 프로세서로 구성하여 임베디드 하드웨어 유전자 알고리즘 실시간 처리에 사용될 수 있게 설계하였다.

설계된 시스템은 진화 하드웨어(Evolvable Hardware)가 적용 대상이며, 임베디드 하드웨어 유전자 알고리즘 수행 과정에서 생산된 chromosome은 FPGA 에 있는 진화 하드웨어에 제공된다. 진화 하드웨어는 환경의 변화에 적응하여 스스로 최적의 회로로 재구성이 가능하여 결함 허용 시스템, 저전력 시스템, 적응 시스템, 회로 설계 등에 대한 새로운 접근 방법으로 제시되고 있다.[4]-[10]

유전자 알고리즘에서 수행되는 selection, crossover, mutation 및 evaluation 처리 함수는 유전자 알고리즘의 적용 대상에 따라 각 함수의 수행 시간이 다를 수가 있다. 따라서 본 논문에서는 이들 함수의 수행 속도를 모니터링 하여 이중 프로세서가 최대의 처리 효율이 가지도록 스스로 작업 분배가 되도록 설계하였다. 제안된 하드웨어 유전자 알고리즘 시스템은 임베디드 리눅스 상에서 수행되며 진화 하드웨어에 실시간으로 처리가 되며, 향후 3개 이상의 프로세서에서도 적용 가능하게 프로세서 내부의 구성을 동일하게 설계하였다.

Ⅱ. 임베디드 하드웨어 유전자 알고리즘 처리 시스템 설계

본 논문에서 제안한 임베디드 하드웨어 유전자 알고리즘 처리 시스템의 블록다이어그램은 그림 1과 같다.

시스템의 구현은 2 대의 Xscale 프로세서와 FPGA(EPF1 0K100ARC240))을 사용하여 설계하였다. Xscale 프로세서에는 임베디드 리눅스를 포팅하였으며 FIFO와 진화 하드웨어는 FPGA에서 구현하였다.

그림 1에서 보는 바와 같이 2 대의 Xscale uP#1과 uP#2는 내부 구조가 완전히 동일하나 uP#1에서는 FIFO의 입출력 요구 선을 모니터를 하는 기능과 uP#2 에서는 유전자 알고리즘을 적용하려는 진화 하드웨어 기능이 부가되어 있다. 2개의 FIFO 장치는 chromosome과 명령 flag을 전달하기 위한 장치이며 각각의 Xscale 프로세서 출력을 상대 프로세서로 전달하기 위해 사용된다. 전체 시스템의 동작은 다음과 같다.

초기에 uP#2에서 buffer에 랜덤 chromosome을 발생시키며 이것을 MUX를 통하여 evaluation부에서 적합도를 평가하게 한다. 이때 uP#2의 selection, crossover와 mutation 처리부는 동작 하지 않는다. uP#2의 chromosome과 적합도를 출력단의 MUX와 FIFO를 통해 uP#1에 전달한다. uP#1은 chromosome의 적합도에 의해 개체들을 selection한 후 crossover와 mutation을 수행한다. 이때 uP #1의 evaluation 부는 출력단의 MUX을 이용해 동작시키지 않는다. uP#1에서 처리된 chromosome은 FIFO를 통해 Xscale uP#2에 전달한다.

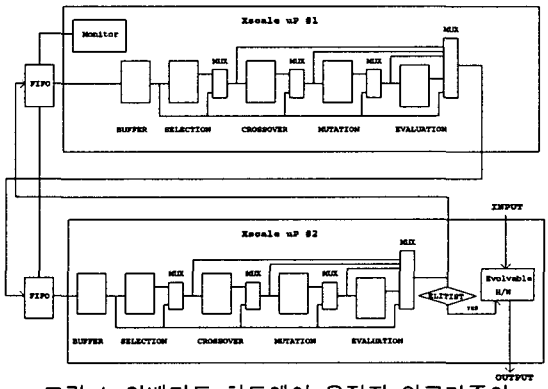

그림 1. 임베디드 하드웨어 유전자 알고리즘의 블록다이어그램

Fig. 1 Block diagram of Embedded Hardware Genetic Algorithm

같은 방법으로 uP#2는 전달 받은 chromosome을 evaluation한 후 이를 다시 FIFO을 통하여 uP#1에 보낸다. 이 과정은 2개의 uP의 부하의 균형을 잡는 절차이다. 이 과정을 초기에 반복하면서 2개의 uP의 부하의 균형을 잡기 위해 2개의 FIFO의 접근 속도를 비교한다.

uP#1의 모니터 처리부는 FIFO read 요구 신호와 FIFO write 요구 신호를 비교하여 2개의 uP의 부하를 알아낸다. 만약 uP#1의 부하가 uP#2의 부하보다 크다면 uP# 1의 mutation 부나 혹은 mutation과 crossover의 일부를 기능을 정지 시키고 그 기능을 uP#2에서 처리하게 할 수 있다.

이것은 각 프로세서 모듈이 selection, crossover, mutation 및 evaluation 처리부를 모두 가지고 있고 각 처리부의 출력부에 MUX에 의해 그 출력을 다음 처리부로 진행 시킬 수 있거나 혹은 MUX에 의해 다음 처리부를 수행하지 않고 생략하여 할 수 있기 때문이다. 따라서 각 프로세서 모듈의 내부의 처리부는 자유로이 활성화 시킬 수도 비활성화 시킬 수 있다. uP#1의 부하가 uP#2의 부하보다 작을 경우 같은 방법으로 부하 조정이 가능 하다.

2개의 uP가 균등 부하가 되었을 경우 이중 프로세서는 최대의 효율을 갖게 되고 하드웨어 유전자 알고리즘을 처리하게 된다. 최대 적합도를 가진 chromosome를 Evolvable H/W에 인가하여 실시간으로 진화하도록 한다.

그림 2.은 이중 프로세서의 하드웨어 블록도 나타낸 것이다.

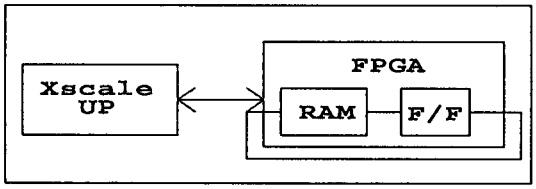

그림 2. 임베디드 하드웨어 블럭도
Fig. 2 Embedded Hardware Block

본 논문에서 selection, crossover, mutation 및 evaluation 처리부를 하나의 블록으로 나타냈지만 보다 세밀하게 나누어 구현할 수가 있다. 예를 들어 selection 부의 경우 많은 부분으로 나누어 MUX 화 하거나 병행 처리 할 수 있다. 그리고 사용된 uP는 동일한 구조를 가지고 있으므로 여러 개의 uP모듈을 직렬 연결하여도 그대로 수행된다. 차후 이 모듈을 개선시켜 모듈을 병렬 연결하여도 효과적인 시스템으로 확장될 수 있을 것으로 기대된다.

III. 실험 및 고찰

그림3은 유전자 알고리즘 수행 처리과정에서 2개의 uP가 균등 부하가 되었을 경우 이중 프로세서의 내부 처리도이다. 2 개의 프로세서가 균등 부하가 되는 내부 처리 구성 도는 유전자 알고리즘의 적용 경우에 따라 다르지만 본 논문에서는 uP#1에서 selection, crossover, mutation이 수행되고 uP#2에서는 evaluation 함수가 수행 되는 것을 알 수 있었다.

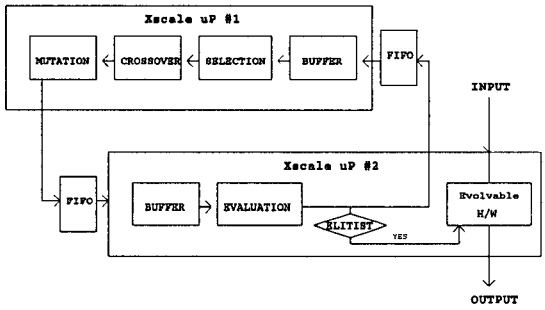

그림 3. 유전자 알고리즘 처리에 있어서 프로세서가 균등 부하를 가질 때 이중 프로세서에의 내부 구성
Fig. 3 Internal Configuration of Dual Processor when each processor has Equal Load in Genetic Algorithm Processing

2개의 uP가 균등 부하가 되었을 때 최대의 수행 속도가 나오는 것을 알기 위해서 유전자 알고리즘을 단일 프로세서로 구현 했을 때와 이중 프로세서로 구현 했을 때의 성능 평가를 하여 보았다.

먼저 균등 부하 상태로 uP#1과 uP#2를 고정 시킨 후 더 이상의 부하의 이동이 없게 한다. 그 상태에서 uP#2의 evaluation 처리 시간을 임의로 가변 하였다. 그 결과 단일 프로세세와 이중 프로세서에서 각각 유전자 알고리즘을 처리한 수행 시간을 비교한 자료가 그림 4에 나와있다.

x축은 유전자 알고리즘의 수행 시간을 나타낸 것이고 y축은 selection, crossover, mutation의 수행 속도를 고정 시킨 상태에서 evaluation 처리 시간을 가변 하였을 때 상대 부하를 나타내었다.

따라서 설계된 시스템은 evaluation 수행 시간이 selection, crossover, mutation의 수행 시간의 합의 2 배가 되었을 때 이중 프로세서 처리 성능이

단일 프로세세 성능에 2 배가 되어 최대의 성능을 보이고 있다.

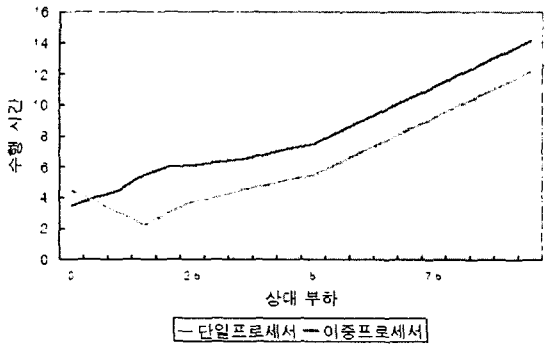

그림 4. 유전자 알고리즘 처리에서의 단일 프로세서와 이중 프로세서의 성능
Fig. 4 Performance of Single Processor and Dual Processor in Genetic Algorithm Processing

evaluation 수행 시간이 매우 작거나 클 경우 즉 두 개의 프로세서의 부하가 균등하게 배분 되지 않고 한 쪽으로 극단적으로 치우치면 이중 프로세서 구조가 오히려 단일 프로세서 성능 보다 못하다는 것을 알 수 있었다.

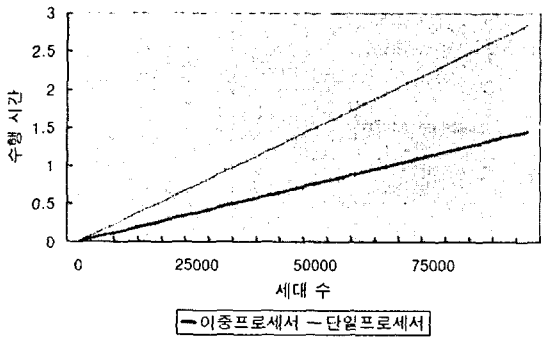

그림 5. 세대수에 따른 수행 시간
Fig. 5 Processing Time according to number of generation

그림 5는 설계된 시스템을 정상적으로 동작 시켰을 때 즉 이중 프로세서 처리 성능이 단일 프로세세 성능에 비해 최대로 되게 균등 부하일 때의 이중 프로세서와 단일 프로세서의 세대수에 따른 수행 시간을 나타낸 것이다. 이중 프로세서로 유전자 알고리즘을 수행 했을 때 세대수에 따라 수행 시간의 차가 커지는 것을 알 수 있다.

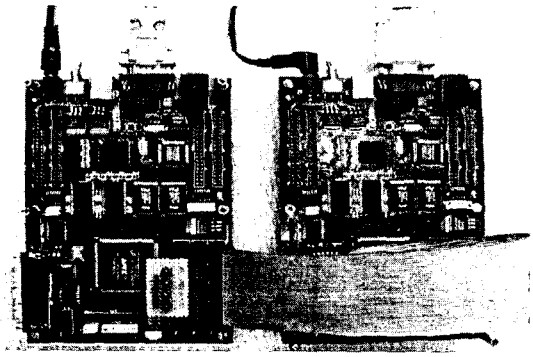

그림 6. 하드웨어 유전자 알고리즘 보드
Fig.6 Hardware Genetic algorithm Board

그림 6은 실험에 사용된 자체 제작한 하드웨어 유전자 알고리즘 보드이다. 2 대의 Xscale 프로세서와 FPGA(EPF1 0K100ARC240))을 사용하여 설계하였다.

IV. 결 론

임베디드 하드웨어 유전자 알고리즘을 위한 실시간 처리 시스템을 개발하였다. 제안된 임베디드 하드웨어 유전자 알고리즘 시스템은 2개의 프로세서 시스템과 2 개의 FIFO 및 진화 하드웨어로 구성되었다.

설계된 시스템은 유전자 알고리즘의 처리 부 selection, crossover, 및 mutation 과 evaluation의 처리부의 부하가 2 개의 프로세서에서 스스로 균등 배분되게 설계하였다.

하드웨어 유전자 알고리즘은 임베디드 리눅스 상에서 수행되며 진화 하드웨어에 실시간으로 처리된다. 실험 결과 설계된 시스템은 selection, crossover, 및 mutation의 수행이 자동으로 2 개의 프로세서에 배분되며 selection, crossover, 및 mutation의 수행시간의 합이 evaluation 함수의 처리 시간의 두 배가 될 때 효과적인 수행을 하는 것을 알 수 있었다.

또한 제안된 이중 프로세서의 각 프로세서 모듈은 동일한 내부 구조를 가지고 있음으로 여러 개의 모듈을 직렬로 연결하여도 그대로 동작 되는 특성을 가지고 있다. 차후 본 모듈을 병렬 연결 구조를 채용할 수 있게 함으로서 보다 빠른 임베디드 하드웨어 유전자 알고리즘 실시간 처리에 사용될 수 있다.

참고문헌

[1] Paul Layzell, The 'Evolvable Motherboard' A Test Platform for the Research of Intrinsic Hardware Evolution, Cognitive Science Research Paper 479, 1998

[2] Koza, John et al, Evolving computer programs using rapidly reconfigurable field programmable gate arrays and genetic programming, Proceeding of the ACM Sixth International Symposium on Field Programmable Gate Arrays. New York, NY:ACM Press. pp. 209-219, 1998

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

[4] Shin'ichi Wakabayashi et al., "GAA:A VLSI genetic algorithm accelerator with on-the-fly adaptation of crossover operators", ISCAS 98, 1998

[5] Jin Jung Kim, Duck Jin Chung, "Implementation of Genetic Algorithm based on Hardware Optimization", TENCON '99 1999;

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

[7] Hiroaki Kitano, IDEN TEKI ALGOLITHM, SANGYO TOSHO, 1993.

[8] E. Vonk, L. C. Jain, and R. P. Johnson, Automatic Generation of Neural Network Architecture Using Evolutionary Computation, World Scientific Publishing, 1997.

[9] L. C. Jain, R. K. Jain, HYBRID INTELLIGENT ENGINEERING SYSTEMS, World Scientific Publishing, 1997.

[10] I. Kajitani, T. Higuchi, "A gate-level EHW chip: Implementing GA operations and reconfigurable hardware on a signal LSI", Evolvable System: From Biology to Hardware, Lecture Notes in Computer Science 1478, pp. 1-12., Springer Verlag, 1998

저자소개

박세현(Se-Hyun Park)

현재 안동대학교 전자정보산업학부 교수
※관심분야 : 하드웨어 유전자 알고리즘, 임베디드 시스템, FPGA, Evolutionary Hardware Design

서기성(Ki-sung Seo)

현재 서경대학교 전자공학과 부교수
※관심분야 : 유전 알고리즘, 유전 프로그래밍, Evolutionary design