1999 年度 大韓電子工學會 夏季綜合學術大會 論文集 第 22 卷 第 1 號 99/6

유전자 프로그래밍 기반의 하드웨어 진화 기법

석 호 식
서울대학교 컴퓨터공학과
전화 : (02) 880-7300
팩스 : (02) 875-2240

이 강
한동대학교 전산전자공학부
전화 : (0562) 260-1387
팩스 : (0562) 260-1149

장 병 탁
서울대학교 컴퓨터공학과
전화 : (02) 880-1833
팩스 : (02) 883-3595

Hardware Evolution Based on Genetic Programming

Ho-sik Seok
Dept. of Computer Engineering
Seoul National University
E-mail : hsseok@scai.snu.ac.kr

Kang Yi
School of Computer Science & Electronic Engineering
Handong University
yk@handong.edu

Byoung-Tak Zhang
Dept. of Computer Engineering
Seoul National University
btzhang@comp.snu.ac.kr

Abstract

We introduce an evolutionary approach to on-line learning for mobile robot control using reconfigurable hardware. We use genetic programming as an evolutionary engine. Control programs are encoded in tree structure. Genetic operators, such as node mutation, adapt the program trees based on a set of training cases. This paper discusses the advantages and constraints of the evolvable hardware approach to robot learning and describes a FPGA implementation of the presented genetic programming method.

I. 서론

진화 하드웨어 혹은 재구성 하드웨어 (reconfigurable hardware)는 실행 시간 중 하드웨어의 구성을 변경할 수 있는 FPGA (field programmable gate array)이다.

실행 시간 중 재구성이 가능하다는 특징은 진화 하드웨어를 결함 허용 시스템, 저전력 시스템, 적응 시스템, 회로 설계 등에 대한 새로운 접근 방법으로 제시하고 있다. [Layzell, 98]

진화 하드웨어는 환경의 변화에 맞추어 최적의 회로를 재구성할 수 있기 때문에, 적응 시스템에의 응용 가능성이 기대되고 있다. [Kajitani, 98], [Liu, 96]

유전자 프로그래밍을 진화 하드웨어 상에서 구현하는 것은 인자들의 미세 조정에는 불리하다. 그러나 개체에 대한 진화 연산과 적합도 평가를 병렬적으로 수행할 수 있으므로 병목 지점으로 작용하는 적합도 평가에서 소모되는 시간을 단축시킬 수 있다. [Koza, 98]

본 논문에서는 진화 하드웨어를 이용한 로봇 컨트롤러 설계에 대하여 소개한다. 로봇 컨트롤러가 진화 하드웨어 상의 회로 형태로 구성되어 있을 경우, 환경 변화에 따라 최적의 제어 회로를 구성하는 것이 가능해진다. 환경에 대한 학습을 통하여 제어 회로를 재구성할 수 있다는 것은 로봇의 활용 범위를 다양하게 하는 것을 가능하게 해준다.

본 논문에서는 센서 입력에 대한 최적의 운동을 찾는 로봇 컨트롤러를 설계하였다. 로봇은 운동을 하면서 받는 센서 정보와 기존의 입력 정보 해석 논리를 통해 새로운 컨트롤 로직을 구성하는 것을 목표로 한다. 이를 위한 적합도 평가, 진화 연산은 모두 진화 하드웨어상에서 이루어진다.

논문은 다음과 같은 구성으로 이루어져 있다. II장은 로봇 제어를 위한 유전자 프로그램의 구조를 설명한다. III장은 GP기반의 제어 프로그램 진화를 위한 연산자 및

적합도 함수에 대한 설명이다. IV장에서는 GP의 하드웨어 구현에 대하여 설명한다. V장에서는 연구의 의의와 앞으로의 연구에서 추가해야 할 사항을 담고 있다.

이번 연구에서는 XILINX사의 xc6216칩을 사용하여 설계를 하였다.

II. 로봇 제어 프로그램

2.1 운동 학습

로봇은 장애물을 피해가며 광원으로 표시되는 목적지를 찾아가는 것을 목표로 한다. 로봇에게 입력으로 주어지는 정보는 표 1에 나타나 있다.

- 초음파 센서: 장애물과의 거리 탐색용 - 2개
- 광센서: 목표의 방향 탐색용 - 3개
- 센서정보 인코딩 형식: $a_0a_1a_2a_3a_4a_5$

  a_0a_1: 목표까지의 방향

  a_2a_3: 앞 장애물까지의 거리

  a_4a_5: 뒷 장애물까지의 거리

표 1. 로봇의 센서 시스템

그러나 로봇이 운동을 시작하는 시점에서의 로봇 컨트롤러는 어떤 입력 정보에 대하여 어떤 운동을 해야 할지 전혀 결정되어 있지 않다. 로봇의 컨트롤러는 그림1과 같은 트리 구조로 구성되어 있으며, 최적의 트리 구조를 찾는 것 로봇의 학습 목표이다.

2.2 로봇 컨트롤러

로봇 컨트롤러는 그림 1과 같은 트리 형태로 구성되어 있으며 트리 하나하나는 유전자 프로그래밍에서 개체에 해당한다. 표2에 개체를 구성하는 노드에 대한 설명이 나와 있다.

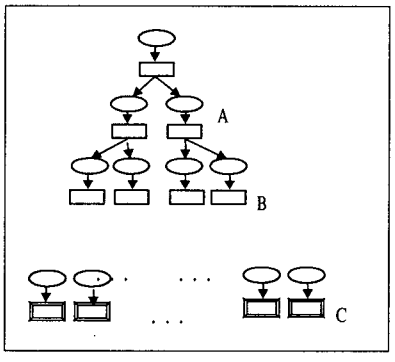

그림 1. 개체의 예

Terminal set에 해당하는 것은 로봇의 운동 정보에 해당한다. 전진, 후진, 오른쪽 전진, 왼쪽전진, 오른쪽 후진, 왼쪽 후진을 나타낸다. Non terminal set으로 설명되는 것은 그림 1에서 노드A에 해당하는것들이다. 세 개의 광센서 중 어떤 것이 목적지를 향하고 있는지의 정보를 $\text{if}_{LL}, \text{if}_{LC}, \text{if}_{LR}$을 통하여 나타낸다.

$\text{if}_{F*}, \text{if}_{B*}$는 정면 장애물과의 거리 및 뒷면 장애물까지의 거리를 4단계로 구분하여 나타낸다.

- Terminal set:
  - MF, MB
  - MLF, MRF
  - MLB, MRB
- Non-terminal set:
  - $\text{if}_{LL}, \text{if}_{LC}, \text{if}_{LR}$
  - $\text{if}_{FX}, \text{if}_{FY}, \text{if}_{FZ}, \text{if}_{FW}$
  - $\text{if}_{BX}, \text{if}_{BY}, \text{if}_{BZ}, \text{if}_{BW}$

표 2. 개체의 노드

표2에는 노드B에 대한 설명이 주어지지 않았는데, 노드 B는 진화에는 참가하지 않는 노드로 정보의 흐름을 제어하는 역할을 한다.

III. 제어 프로그램의 진화

3.1 진화 연산 파라미터

컨트롤러 진화를 위한 유전 연산 파라미터가 표3에 나타나 있다. 개체군의 크기는 4이다. 진화 연산에서 일반적으로 사용하는 개체군 크기에 비하여 작다고 할 수 있는데 이것은 진화 하드웨어에서 제공하는 게이트 수의 제약에 의한 것이다.

- 개체의 수: 4
- 개체의 모습: 깊이 8의 이진 트리
- 선택 정책: $(\mu+\lambda,\lambda)$, $\lambda=2$, $\mu=2$
- 유전 연산: mutation, extended mutation
- 종료 정책: 한정된 세대수 동안 진화 수행

표 3 . 진화 연산 파라미터

개체의 진화, 적합도 평가, 진화 연산 모든 것을 진화 하드웨어상에 표현하고 있으므로 개체군의 크기가 작아지게 된다. 4개의 개체중 2개를 부모 세대로 선택하게 되는데 선택 정책은 $(\mu+\lambda,\lambda)$를 따른다. 개체는 깊이 8의 이진 트리이며 유전자 프로그래밍의 개체로서는 작다고 할 수 있는데 8이상이 될 경우 하드웨어 자원을 과다하게 요구하기 때문에 개체군의 크기를 더 줄이게 된다. 이 때문에 8로 고정될 수 밖에 없다. 한정된 세대수가 지나면 진화를 종료시키는데, 로봇이 실제 상황에서 운동하는 경우를 감안했기 때문이다.

3.2 유전 연산자

개체 진화를 위하여 사용되는 연산자는 mutation과 extended mutation이다. mutation은 그림 1의 노드 중 A노드나 C노드 중 한 노드를 선택하여 다른 노드로 변화시키는 것이다. extended mutation은 mutation을 확장한 것으로 crossover 효과를 보기 위한 것이다. 어느 한 노드를 선택하면 그 노드에 주의의 모든 노드에 대하여 mutation을 실행한다.

3.3 적합도 평가

적합도 함수는 식(1)과 같이 표현된다.

$$f(a_1, a_2, a_3, a_4) = \alpha I_1(a_1) + \beta I_1(a_2) + \gamma I_2(a_1, a_2)a_3 + \delta a_4 \quad \cdots(1)$$

식 1에 적합도 함수가 나타나 있다. 각 항의 계수들은 $\alpha > \beta > \gamma > \delta$의 관계를 이루며 $I_1(x)$와 $I_2(x, y)$는 다음과 같이 해석된다.

$$I_1(x) = \begin{cases} x \geq 2 & \eta \\ 1 \leq x < 2 & \lambda \\ x < 1 & \chi \end{cases}, \quad (\ 1 < \eta < 2,\ \eta < \lambda, \chi > 2\lambda\ )$$

$$I_2(x, y) = \begin{cases} x \geq 2 \text{ and } y \geq 2 & \frac{1}{\varepsilon} \\ others & \varepsilon \end{cases}, \quad (\ 0 < \varepsilon < 1\ )$$

식(1)을 직접 진화 하드웨어상에서 계산하지는 않는다. 센서 입력 정보가 한정되어 있으므로 식(1)에 따라 센서 입력 정보를 계산한 후 적합도 테이블을 구성하고 이 테이블에서 적합도를 부여하는 방식으로 적합도를 평가한다. $a_1 \sim a_4$는 개체의 입력 정보 및 운동 정보를 10진수로 변환한 값이다. I_1 과 I_2는 장애물과의 거리에 따라 목표까지의 거리와 장애물과의 거리 사이의 가중치를 다르게 하기 위한 함수이다. α, β, γ, δ의 크기 관계는 장애물과의 거리와 목표까지의 거리, 목표까지의 방향 사이의 가중치를 의미한다. α, β, γ, δ를 이용하여 장애물과의 거리, 목표까지의 거리, 목표까지의 방향등의 적합도 계산에 대한 기여도를 조정한다.

IV. 유전자 프로그래밍의 하드웨어 구현

4.1 하드웨어상에서의 진화

그림2에 하드웨어상에서의 진화 모습이 개략적으로 보여지고 있다. VHDL을 이용하여 제어 회로를 표현한다. 생성된 개체들에 대하여 Mutation과 Extended mutation을 행한다. 진화 연산을 적용한 후 적합도 평가를 통하여 다음 세대의 부모로 작용할 개체들을 선정한 후 다시 진화 연산을 적용한다. 이 과정을 정해진 세대 수만큼 반복하는 것이다.

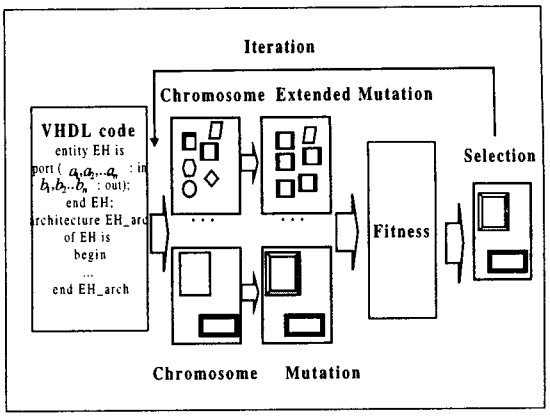

그림2 . 하드웨어 상에서 진화

4.2 개체의 표현

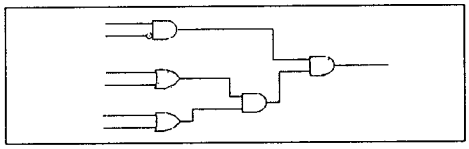

그림 3. 노드 A의 예

그림3는 개체를 이루는 노드중 노드A의 예를 보이고 있다. 그림2와 같이 개체들은 게이트를 이용하여 표현되는데, VHDL을 이용하여 진화 하드웨상에 표현된다.

개체에 입력으로 들어오는 것은 센서의 입력 정보 6비트이다. A라고 표시된 노드 입력 정보의 부분 해석을 담당하는데 센서 정보가 참일 경우 출력으로 1을 내보낸다. A 노드 표2의 Nonterminal set에 포함되어 있는 if 함수의 진위 여부를 판단하는 역할을 한다.

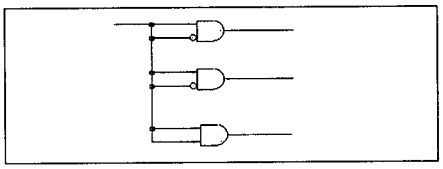

그림 4. 노드 C의 예

그림 4에서는 노드 C를 이루는 함수 중 하나를 보이고 있다. 입력은 항상 1이 들어 온다. 이 입력 정보에 대하여 불린 연산을 수행하여 가능한 6가지 운동 중 하나를 표현하도록 한다.

4.3 진화 연산

그림 5에서는 진화 하드웨어상에서 Mutation이 이루어지는 모습을 보이고 있다. 회로도(a)와 회로도(b)의 왼쪽 상단 게이트를 보면 (a)의 게이트에는 NOT (Bubble)

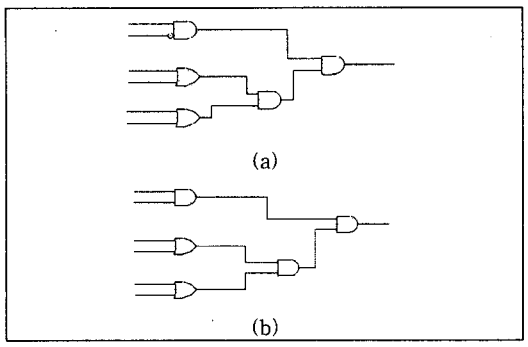

그림 5. Mutation의 모습

이 붙어 있는 반면 (b)에는 NOT이 붙어 있지 않다. 이것은 회로를 구성하는 과정에서 미리 "reconfig" 속성을 부여한 후 제어 프로그램에서 "AND2B1"으로 표현되어 있는 게이트의 함수를 "AND2"로 변경하는 방식으로 이루어진다. Mutation은 하나의 노드를 이루는 게이트에 대하여 적용되어 다른 노드로 변경하는 역할을 한다.

Mutation이외에 Extended mutation을 지원하는데 이것은 하나의 노드에만 적용되는 Mutation의 단점을 보완하기 위한 것이다. Extended mutation node를 선택하면 이노드 주위 노드 n개에 대하여 Mutation을 실시하도록 한다. Extended mutation이 적용되는 노드 수 n은 무작위로 주어진다.

V. 결론

본 연구에서는 진화 하드웨어에 기반한 로봇 컨트롤러를 설계하여 보았다. 진화 하드웨어상에서의 진화의 수행은 다음과 같은 특징을 갖는다. 첫째, 수행하려는 진화에 특화된 진화 연산자를 구성할 수 있어 빠른 진화가 가능하며, 둘째, 개체간의 진화 연산을 병렬적으로 수행할 수 있고, 셋째, 일반 프로세서를 이용했을 경우, 계산의 병목 원인으로 작용하는 적합도 평가를 병렬적으로 수행하여 최적 개체를 진화시키는 과정에서 소요되는 시간을 줄일 수 있다는 장점을 갖는다. 그러나 하드웨어에서 제공하는 게이트수가 한정된 관계로, 소프트웨어적인 방법으로 수행되는 진화 연산에 비하여 개체군의 크기가 작으며, 개체의 성장을 제한해야 한다.

향후 개선되어야 할 사항은 다음과 같다. 첫째, Open architecture의 특징을 이용하여 진화를 수행하여 본다. 이번 연구에서는 게이트를 이용하여 회로를 구성한 후 게이트를 변경하여 진화 연산을 수행하는 방식을 사용하였다. 이것은 configuration bit를 직접 제어할 수 있는 XC6216의 특징을 전혀 이용하지 않은 것이다. 다음 연구에서는 configuration bit의 변경을 통한 진화를 시도해 보아야 할 것이다.

둘째, 적합도 연산의 진화를 시도한다. 실험에서 사용된 적합도 연산기는 센서 입력 정보와 운동의 조합에 따른 기대 효과를 미리 설정된 함수를 통해 구해낸다. 그러나 센서 입력의 잡음 효과와, 설계자가 고려하지 못한 요소가 발생할 수 있다는 점등을 생각한다면 적합도 연산을 계산하는 부분도 동작중에 개선될 필요성이 있다. 다음 연구에서는 이 부분도 실행 시간중에 진화할 수 있도록 할 것이다.

셋째, 실제 로봇의 운동 제어에 응용하여 본다. 이번 연구에서는 실제 로봇에 적용하여 보지는 못하였다. 로봇이 실험에 사용될 수 있도록 정비되는 대로 실제 로봇의 제어에 사용해 보아야 할 것이다.

감사의 글: 본 연구는 과학재단 핵심전문 연구(과제 번호 981-0920-107-2)에 의해 지원되었음.

참고 문헌

[Layzell,98] Paul Layzell, The 'Evolvable Motherboard' A Test Platform for the Research of Intrinsic Hardware Evolution, *Cognitive Science Reseacrch Paper 479,* 1998.

[Liu, 96] Weixin Liu et al, ATM Cell Schedulling by Function Level Evolvable Hardware, *International Conference on Evolvable Systems 1996,* pp. 180-192 ,1996.

[Kajitani, 98] Isamu Kajitani, et al, A Gate-Level EHW Chip: Implementing GA Operations and Reconfigurable Hardware on a Single LSI, *International Conference on Evolvable Systems 1998,* pp. 1-12, 1998.

[Koza, 98] Koza, John et al, Evolving computer programs using rapidly reconfigurable field proramm able gate arrays and genetic programming, *Proceedings of the ACM Sixth International Symposium on Field Programmable Gate Arrays.* New York, NY: ACM Press. pp. 209-219, 1998.