GAP와 진화 하드웨어를 이용한 State Machine설계

Design of State machine using Evolvable Hardware and Genetic Algorithm Processor

김태훈, 선흥규, 박창현, 이동욱, 심귀보
중앙대학교 전자전기공학부

Tae-Hoon Kim, Heung-Kyu Sun, Chang-Hyun Park, Dong-Wook Lee, and Kwee-Bo Sim
School of Electrical and Electronic Engineering, Chung-Ang University
E-mail : kbsim@cau.ac.kr

ABSTRACT

GA(Genetic Algorithm)는 자연계 진화를 모방한 계산 알고리즘으로서 단순하고 응용이 쉽기 때문에 여러 분야에 전역적 최적해 탐색에 많이 사용되고 있다. 최근에는 하드웨어를 구성하는 방법의 하나로서 사용되어 진화하드웨어라는 분야를 탄생시켰다. 이와 함께 GA의 연산자체를 하드웨어로 구현하는 GA processor(GAP)의 필요성도 증가하고 있다. 특히 진화하드웨어를 소프트웨어상에서 진화 시키는 것이 아닌 GAP에 의해 진화 시키는 것은 독립된 구조의 진정한 EHW 설계에 필수적이 될 것이다. 본 논문에서는 GAP 설계 방법을 제안하고 이를 이용하여 진화하드웨어로 State machine을 구현하고자 한다. State machine의 경우 구조상 피드백이 필요하기 때문에 가산기나 멀티플렉서보다는 훨씬 복잡하고 설계가 까다로운 구조이다. 제안된 방법을 통하여 명시적 설계가 어려운 하드웨어 설계에 GAP를 이용한 하드웨어의 진화에 적용함으로써 그 유용성을 보인다[1].

Key words : 진화하드웨어(EHW), 유전자 알고리즘 프로세서(GAP), State-Machine

I. 서 론

자연계의 각종 생물은 진화의 과정에서 환경에 적응하도록 그 형태나 구조를 진화시킨다. 진화연산은 자연계의 진화과정을 모방하여, 그 구조를 나타내는 코드나 구조를 변경하여 새로운 기능을 가지도록 하는 것이다. 이것은 소프트웨어의 진화만이 아니라, 하드웨어를 진화시키는 것도 자율성, 창조성이 풍부한 정보처리 시스템의 창출에 있어서 중요한 연구 과제이다.

본 연구는 2001년도 서울시·중기청 산학연 공동기술개발 컨소시엄사업의 연구비에 의하여 수행되었습니다. 연구비 지원에 감사 드립니다.

소프트웨어 진화와 유사하게 표현하면, 하드웨어 진화는 환경의 변화나 오차를 이용하여 전자회로인 하드웨어가 그 구조를 자율적으로 바꾸어 그 구조는 물론 기능까지 복잡화하고 다양화하는 것이다. 따라서 그 표현형이 프로그램이 아니라 전자회로인 점을 제외하면, 방법론적으로는 소프트웨어 진화와 기본적으로 차이는 없다. 단 회로소자 그 자체는 진화하지 않지만, 예를 들어 CPLD(Complex Programmable Logic Device)나 FPGA(Field Programmable Gate Array) 등 소자의 결선이나 조합이 재구성 가능한 구조를 가진 하드웨어 디바이스를 전제로 한다.

이와 관련된 연구 중에는 재구성 가능한 디바이스로서 셀룰라 오토마타 머신(CAM: Cellular Automata Machine)을 이용하여 뇌

신경계의 기본구조 모델인 신경망을 하드웨어로서 발생, 성장, 진화시키고자 하는 연구와 미지의 혹은 끊임없이 변동하는 환경에 자율적으로 적응하여 목적의 기능을 획득하는 유연한 하드웨어의 구축을 목표로 하드웨어기술언어(HDL: Hardware Description Language)를 이용한 하드웨어 진화시스템의 연구도 있다[2].

하드웨어를 재구성하는 방법에는 Extrinsic method와 intrinsic method가 있다. Extrinsic method는 컴퓨터(소프트웨어로)에서 하드웨어를 진화 최종진화 후에 하드웨어에 download하는 것이고, intrinsic method는 하드웨어를 구성하여 적합도를 평가하고 다시 재구성하는 방법이다. 이러한 방법은 하드웨어를 재구성하기 위해 부피가 큰 데스크탑 컴퓨터가 필요하다. 본 논문에서는 재구성에 필요한 GAP와 EHW를 하나의 칩으로 설계함으로써 개체 생성에 필요한 환경만 설정하면 적합도가 높은 개체를 생성하는 하드웨어 설계방법을 제안한다[3].

II. 진화 하드웨어(EHW)

유전자 알고리즘은 어떤 세대(generation)를 형성하고 있는 개체(individual)의 집합 가운데 환경에 대한 적응도(fitness)가 높은 개체가 높은 확률로 살아 남도록 재생되도록 하고 교차(crossover)나 돌연변이(mutation)에 의해서 다음 세대의 개체군이 형성되어 가도록 하는 것이다[4].

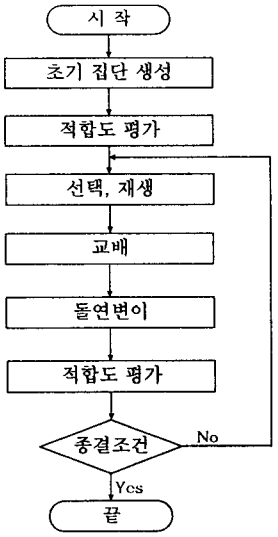

그림 1. 유전자 알고리즘의 기본적인 동작

이 알고리즘을 하드웨어에 적용하기 위해서는 생성된 개체의 bit string에 따라 변형될 수 있는 하드웨어가 필요하다. 이런 하드웨어의 대표적인 것은 Xinlinx의 XC6200 시리즈이다[5]. 이 칩의 구조는 그림 2와 같이cell이 상하좌우로 연결되어 있다[6].

일반적으로 조합회로를 설계할 때는 순차적으로 진행되지만 state-machine은 피드백이 필요하기 때문에 단순히 순차적구조로는 구성할 수 없기 때문에 그림2와 같은 구조를 지녀야 한다.

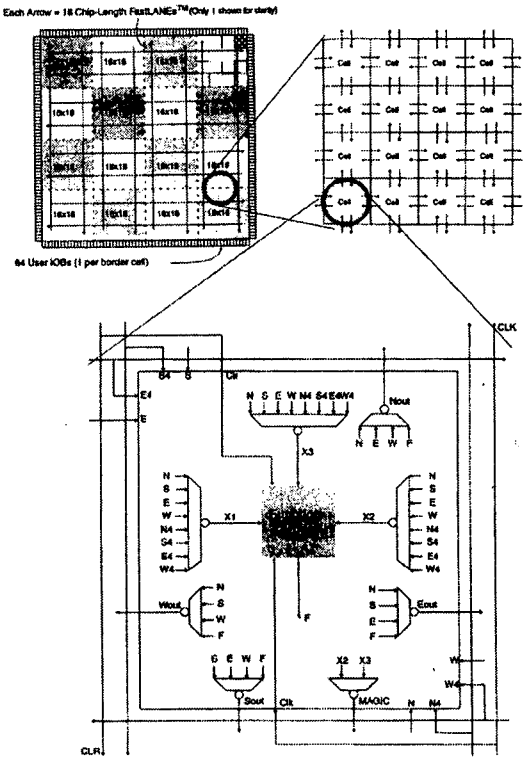

그림 2. XC6200시리즈의 구조

하지만 XC6200시리즈는 intrinsic method를 사용하기 때문에 외부에서 재구성하는 장치가 필요하다. 그리고 이 제품은 단종되었기 때문에 더 이상 사용할 수 없는 기종이다. 본 논문에서 사용한 FPGA는 VertexE(모델명: XCV2000E)이다. 이 FPGA의 cell은 CLB (Configurable Logic Block)으로 구성되어 있다. 이는 XC6200와는 달리 LUT(Look Up Table)과 플립플롭으로 구성되어 있기 때문에 논리 게이트의 구성방법이 다르고 각 cell을 게이트 단위의 조작이 힘들다. XC6200시리즈와 비슷한 기능을 하기 위해서 XC6200시리즈와 같은 기능을 하는 function level의 cell을 VHDL을 이용하여 작성한 후 이를 component로 사용하여 각 부분을 연결하여 사용한다. 그림 3은 이와 같은 구조를 사용한

것이다. 그리고 여기에 적합도(fitness)를 구할 수 있도록 다른 모듈을 덧붙였다. 적합도를 구하기 위한 평가벡터를 메모리에 저장하고, 각 개체가 발생 혹은 재생될 때마다 이를 진화하드웨어에 입력하고 출력을 비교하여 적합도를 구한다. 이를 위해서는 GAP와 EHW사이의 데이터전송을 위한 동기화가 필요하므로 EHW의 동작, 평가벡터 저장, 하드웨어재구성, 적합도 평가는 GAP의 컨트롤에 의해 이루어진다.

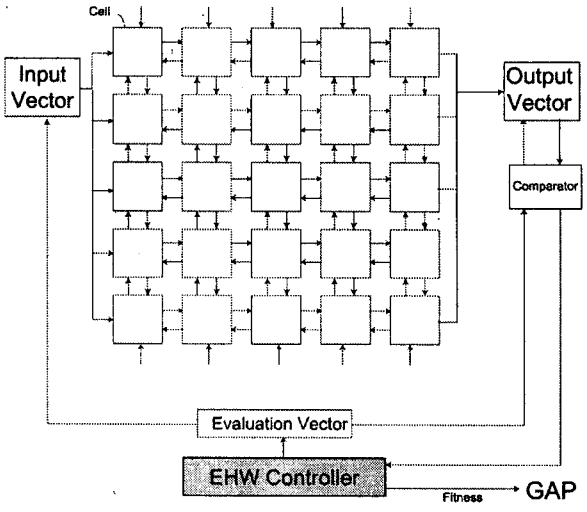

그림 3. EHW 구조

EHW의 각 cell은 네 개의 입력을 각각 다른 논리회로로 네 군데의 출력에 연결되어 있다. 이 논리회로의 역할을 결정하기위해 상하좌우에 2bit, 한 cell에 8bit의 configuration bit을 설정했다. 이 논문에서 사용한 cell의 개수는 36개이기 때문에 총 288bit가 필요하다.

III. 유전자 알고리즘 프로세서(GAP)

3.1 GAP의 구조

유전자 알고리즘을 하드웨어로 구현하기 위해서 필요한 것이 생성된 bit string을 발생, 선택, 교차, 돌연변이등을 시키는 프로세서가 필요하다.

그림 3의 진화하드웨어를 재구성하는데 필요한 bit는 288bit이다. 그림 4의 각 버스를 288bit로 사용하는 것은 많은 리소스를 낭비하고 또한 이런 방법으로는 VertexE의 FPGA가 지원하지 않는다. 그래서 각 버스는 32bit로 하고 이를 9번 반복하여 수행함으로써 288bit를 생성하고, 교차 돌연변이 시킨다. 그림 5는 이와 같은 과정이 EHW에서 동작하는 것을 logic simulator로 확인한 것이다.

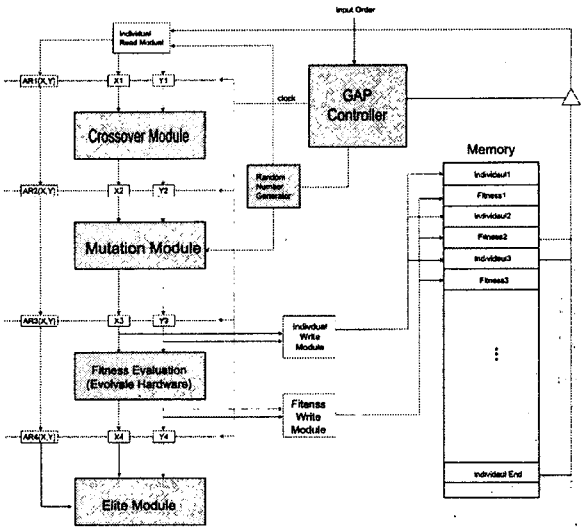

그림 4. GAP 구조

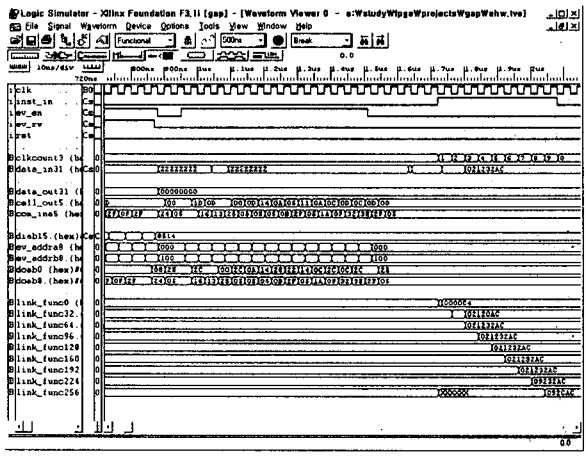

그림 5. logic simulator

교차는 32bit를 기준으로 2점 교차를 수행하였고, 돌연변이는 각 bit에 대해서 적용한 것이 아니라 32bit중 하나의 bit에만 적용한 것이므로 교차율은 줄이고, 돌연변이율은 높여야만 일반적인 유전 알고리즘과 비슷한 확률로 설정된다.

적합도 평가(Fitness evaluation)부분은 앞에서 설명한 진화하드웨어가 GAP의 한 부분으로 들어가서 적합도를 평가한다. 엘리트 모듈은 이렇게 발생한 개체의 재생에 있어 영향을 준다. 적합도가 높은 개체를 관리함으로써 수렴을 빨리 할 수 있도록 했다. 개체가 288bit이므로 이를 따로 관리하기 위해서는 또다시 많은 메모리가 필요하므로 메모리 주소와 적합도만을 저장하여 효과적으로 사용하였다.

GAP를 사용할 때 항상 필요한 RNG (Random Number Generator)는 random number, R_{i+1}을 발생시키는 과정은 다음 (1)을 이용했다.

$$R_{i+1} = (A \times R_i + B)\%M \qquad (1)$$

위의 (1)식에서 A, B, M은 임의의 정수를 대입하는데 특정한 법칙은 없다. 초기 R값에 따라 그 수는 달라지므로 초기값(seed)을 정하는 것이 중요한데 이것은 clock을 count하여 정하였다. 나머지(remainder)연산은 곱셈 후 bit를 줄였다.

개체의 저장 역시 32bit 버스에서 사용하기 위해서는 9개의 clock이 필요하므로 EHW에 configuration 하는 동시에 메모리에 저장하여 저장을 위한 clock를 줄인다.

개체와 적합도를 저장하기 위한 메모리는 VertexE에서 제공하는 Block Memory를 사용하였고 input data는 유전자 알고리즘을 사용하기 위한 최소한의 환경과 적합도를 평가하기 위한 평가벡터를 입력한다. 이후의 진행과정은 GAP Controller를 사용하여 일괄적으로 처리하도록 했고, 종료조건은 세대 혹은 적합도에 따라 유전자 알고리즘을 수행한다.

3.2 GAP를 이용한 State machine

State machine을 설계하기 위해서는 그림 6와 같은 state diagram을 설계한 후 table을 이용하여 카르노프 맵을 작성한 후 논리 회로를 구성하여야 한다. 그러나 EHW에서는 평가벡터를 사용하여 진화시키므로 전문적인 지식 없이 평가 벡터를 사용하여 설계할 수 있다.

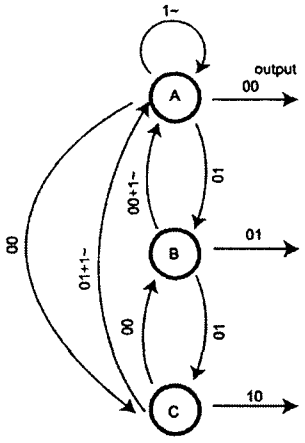

그림 6. State machine Diagram

IV. 결 론

현재의 인간은 많은 기계를 사용하고 있다. 미래사회에서 구현하기 어려운 문제에 봉착했을 때 이를 기계 스스로가 진화하여 문제를 해결한다면 사람 역시 많은 발전을 할 수 있을 것이다. 본 논문에서 주변 환경에 맞춰 진화하는 하드웨어를 구현함으로써 이러한 하드웨어 진화의 가능성을 보였다. 특히 이전 연구에서는 EHW는 컴퓨터에서 configuration bit를 download하였으나 이를 GAP와 함께 설계하여, 하나의 칩으로 구현함으로써 부가적인 장비 없이 독자적으로 진화하는 하드웨어를 구현하였다. 특히 feedback 회로인 state machine을 설계하여 이후 여러 분야에 적용할 수 있을 것으로 기대된다. 본 논문에서는 인터페이스를 제작하지 못하여, VertexE칩의 input/output를 확인하기 위해서는 컴퓨터를 사용하였으나 이후 연구에서는 인터페이스를 제작하여 독립적인 모듈로 동작할 수 있도록 할 것이다.

V. 참고문헌

[1] Tetsuya Higuchi, Hitoshi Iba, Bernard Manderick, *Evolvable Hardware*(Massively Parallel Artifical Intelligence, pp. 398-421), MIT Press, 1994.

[2] 반창봉, 곽상영, 이동욱, 심귀보, "FPGA를 이용한 진화형 하드웨어 설계 및 구현에 관한 연구," *한국퍼지 및 지능시스템학회 논문지*, Vol. 11, No. 5, pp. 426-431, 2001. 10.

[3] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, INC., 1989.

[4] Jin Jung Kim, Daek Jin Chung, "Implementation of genetic algorithm based on hardware optimization," *TENCON 99, Proceedings of the IEEE Region 10 Conference*, Vol. 2, pp. 1490-1493, 1999.

[5] Gordon Hollinworth, Steve Smith, Andy Tyrrell, "The Intrinsic Evolution of Virtex Devices," *Evolvable Systems: From biology to Hardware, ICES2000*, pp. 72-79, Springer, 2000

[6] Adrian Thompson, *Hardware Evolution*, Springer, 1998..