Proceedings of KFIS Fall Conference, 2003. 5. 3

FPGA를 이용한 진화 하이브리드웨어

Evolvable Hybrid-ware using FPGA

김태훈, 이동욱, 심귀보
중앙대학교 전자전기공학부

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

ABSTRACT

진화하드웨어는 하드웨어 스스로 진화하여 필요한 회로를 구성한다. 회로를 재구성하기 위해서 유전자 알고리즘을 사용한다. 유전자 알고리즘(Genetic Algorithm)은 전역적 탐색을 통하여 해를 구한다. 하지만 유전자 알고리즘은 많은 개체의 평가를 통하여 이루어지기 때문에 수행하는데 시간이 많이 소요된다. 이전의 연구에서 유전자 알고리즘 프로세서를 이용하여 진화하드웨어를 구성했다. 유전자 알고리즘 프로세서는 유연성이 떨어지고 범용적으로 사용하기 어렵다. 본 논문에서는 CPU를 이용하여 유전자 알고리즘 프로세서를 소프트웨어로 제어하는 방법을 제안한다. 소프트웨어로 합성한 신호로 GAP의 동작을 제어하기 때문에 유연성을 가질 수 있다. FPGA에 CPU와 유전자 알고리즘 프로세서를 구현하여 one-chip 하드웨어를 구현한다.

Key words : 진화하드웨어(EHW), 유전자 알고리즘 프로세서(GAP), FPGA, CPU

I. 서 론

자연계의 진화과정을 모방하여 최적화 알고리즘으로 사용하는 기술이 진화연산이다. 진화연산의 기본원리는 자연계의 생물처럼 적합도가 높은 개체들을 남기고 적합도가 낮은 개체들을 줄여 나가는 것이다. 진화연산을 적용하기 위해서는 개체를 설정하여 적합도 평가 방법에 따라 여러 분야에 쓰일 수 있다.

진화하드웨어는 스스로 진화하여 회로를 구성하는 것으로 진화연산 알고리즘을 하드웨어에 적용시킨 것이다. 진화하드웨어에 적용되는 알고리즘은 유전자 알고리즘인데 이는 하드웨어의 크기가 고정되어 있기 때문이다. 유전자 알고리즘은 존 홀랜드에 의해 제안되었다[1]. 유전자 알고리즘은 복잡한 최적화 문제에 해를 찾는데 유용하다[2][3]. 유전자 알고리즘의 개체와 평가함수를 어떻게 정의하느냐에 따라 여러 분야에 적용시킬 수 있다. 하지만 유전자 알고리즘은 여러 개체를 생성하여 적합도에 따라 진화를 함으로써 실행속도가 느리다는 단점이 있다. 특히 유전자 알고리즘의 개체를 하드웨어에 적용하여 하드웨어의 구조나 기능을 바꾸게 할 경우 많은 비트가 필요하므로 속도문제는 더욱 심각 하게 된다.유전자 알고리즘의 속도문제를 극복하기 위해서는 병렬 프로세싱을 사용하는 방법이 있다[4]. 이전의 연구에서는 파이프라인 구조의 유전자 알고리즘 프로세서를 제안했다[5]. 유전자 알고리즘 프로세서의 controller는 하드웨어로 구성되어 있기 때

본 연구는 과학기술부의 뇌신경정보학연구사업의 '뇌정보처리 메커니즘에 기반한 인간행동 시스템연구'의 연구비 지원으로 수행되었습니다. 연구비 지원에 감사드립니다.

문에 이후에 프로세서를 수정하기 위해서는 전체를 새로 구성해야 한다.

본 논문에서 일반적인 CPU를 이용하여 소프트웨어로 유전자 알고리즘 프로세서를 제어하는 방법을 제안한다. 이는 유연성을 가지고 더 범용적으로 사용할 수 있다.

II. 진화 하드웨어(EHW)

하드웨어에 진화알고리즘을 적용하기위해 유전자 알고리즘을 사용하는데 이는 하드웨어의 크기는 정해져 있고 이것의 기능을 정의할 때 bitstring을 사용하기 때문이다. 특히 Xilinx의 XC6200모델은 게이트 단위로 재구성이 가능하여 연구에 많이 이용되는데[6], 범용성이 적어 단종되었다. 또한 Xilinx의 JBit을 이용하여 LUT의 기능을 정의하여 진화시키는 연구도 있다[7].

본 논문에서 사용한 FPGA는 VertexE(모델명: XCV2000E)이다. 이것은 XC6200와 다르게 구성되어 있기 때문에 게이트 단위의 조작할 수 없다. 그래서 XC6200과 비슷한 구조의 module을 만들어 게이트단위의 변화를 가질 수 있도록 하였다[5].

III. 유전자 알고리즘 프로세서(GAP)

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

그림 1는 유전자 알고리즘을 적용하기 위하여 만든 component이다. 이 component 내부에는 교차와 돌연변이를 위한 여러 module로 구성되어 있다. 적합도 평가를 하기 위한 module은 여러 module로 사용할 수 있도록 일정한 인터페이스로 이용한다.

이전의 논문에서는 controller를 이용하여 module를 제어했기 때문에 GAP내에서 모든 기능을 제어했다. 하지만 본 논문에서는 GAP 외부 메모리에서 데이터를 가져오기 때문에 I/O module이 필요하다.

GAP는 bitstring이 길 경우pipeline으로 동작하는데 crossover module에 필요한 신호는 한 유전자가 crossover module에서 처리되고 mutation module에 넘겨진 후 연속적으로 다른 유전자가 들어오므로 같은 신호가 필요하다. 반복되는 회수는 유전자의 길이에 따라 다르다. 또한 crossover module과 mutation module에 필요한 신호의 반복횟수는 같지만 신호의 동기는 mutation module이 1 clock 느리기 때문에 이것을 제어해야만 한다.

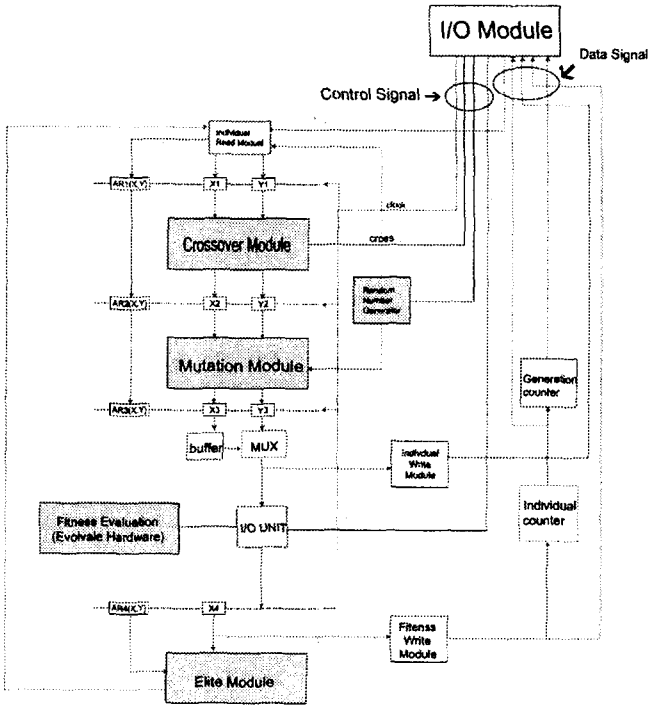

그림 1. GAP 구조

적합도 평가(Fitness evaluation)부분은 앞에서 설명한 진화하드웨어가 GAP의 module로 사용한다. EHW의 bit수와 마찬가지로 적합도 평가에 걸리는 시간은 어떤 적합도 함수이냐에 따라 평가하는데 걸리는 시간이 다르므로 적합도 평가를 하는 동안은 프로세서의 동작을 멈추고 평가가 끝나도록 대기하다가 적합도 평가가 끝나면 다시 다른 과정을 수행하도록 한다. 이를 위해서 상태 플래그를 사용하여 진화하드웨어와 유전자 알고리즘 프로세서가 어떤 상태인지 확인한다.

엘리트 module은 발생한 개체의 재생에 있어 영향을 준다. 적합도가 높은 개체의 재생확률을 높임으로써 수렴을 빨리 할 수 있도록 했다. 개체가 288bit이므로 이를 저장하기 위해서는 많은 메모리가 필요하므로 메모리 주소와 적합도만을 저장하여 module의 크기를 줄였다. 개체은 역시 32bit 버스에서 사용하기 위해서는 9번의 clock이 필요하므로 EHW에 configuration 하는 동시에 메모리에 저장하여 저장을 위한 clock를 줄인다.

EHW에서 상태레지스터를 이용하여 reconfiguration 중인지 아닌지 확인하듯이 GAP 또한 데이터 전송을 위해서 상태레지스터를 가지고 있어야 한다. 이는 유전자 알고리즘이 적용되어 EHW가 동작을 하는지 안하는지를 통하여 GAP동작을 위한 프로그램을 작

동시키기 위해서이다. 또한 현재 GAP에 필요한 데이터가 무엇인지 확인하여 전해주고 GAP에 필요한 연산을 할 수 있도록 하기 위해서이다.

이러한 GAP는 적용되는 EHW module만 변경하면 다양한 문제에 적용될 수 있다[5].

IV 소프트웨어를 이용한 GAP제어

하지만 최근의 FPGA를 이용한 ASIC에서는 단순히 하나의 기능을 설계하는 것이 아니라, 여러 기능을 하나의 칩에 설계하는 SoC(System on Chip)방식을 사용한다. 그래서 최신의 FPGA 모델은 범용CPU를 칩의 일부분으로 포함시켰다(Altera의 Excalibur, Xilinx의 Vertex II Pro). 본 논문에서는 CPU를 이용하여 GAP를 제어함으로써 GAP의 유연성을 늘리고 칩에 CPU를 구성함으로써 범용성을 늘릴 수 있도록 하였다.

4.1 하이브리드웨어의 구조

GAP는 속도문제를 개선하기 위하여 유전자 알고리즘을 하드웨어로 구성한 것이다. GAP는 EHW와는 달리 하드웨어의 기능을 변경할 수 없기 때문에 이후 다른 방식을 적용하기 위해서는 회로를 다시 설계하여 구성하여야 한다. 이 방식은 유전자 알고리즘의 유연성을 적용하기 어렵다. 본 논문에서는 CPU를 이용하여 GAP의 control signal을 합성하여 이를 이용하도록 한다.

그림 2는 컴퓨터 I/O인터페이스의 한 구조이다. GAP는 메모리의 한 주소로 인식된다. GAP에 상태레지스터를 설정하여 CPU에서 이것을 읽어들인 후 상태에 따라 프로그램을 실행시킨다.

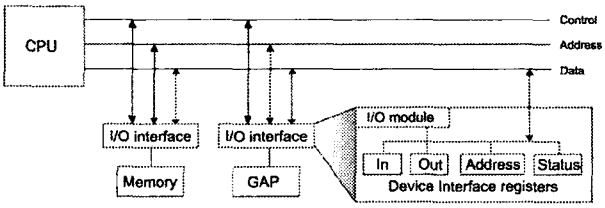

그림 2. Programmed I/O Device Interface Structure

초기상태에서 GAP 속의 진화하드웨어는 설정이 되지 않은 상태이기 때문에 상태레지스터를 이용하여 이를 표시한다. CPU에서 상태레지스터에 해당하는 주소의 데이터를 읽어온 후 상태에 따라 프로그램의 서브루틴을 실행시켜 GAP의 control signal 를 합성할 것인지 GAP를 실행시킬 것인지 결정한다.

GAP의 control signal이 완성되면 GAP에 유전자 알고리즘을 실행하는 control 신호를 보내준다.

GAP의 control signal이 합성된 후에는 CPU에서 GAP의 상태만 확인하게 되어 관여하지 않으므로 일반적인 프로세서로 사용하게 된다. GAP에서는 control signal을 불러오거나 유전자를 메모리에서 불러와야 한다. 이때에는 DMA(Direct Memory Access)방식을 이용한다. DMA는 CPU를 거치지 않고 데이터를 옮기는 방식이다. CPU에서 메모리를 사용하지 않는 동안 메모리 데이터를 가져와서 사용한다. 또한 유전자 알고리즘을 이용한 진화가 끝났을 경우 GAP에서는 메모리를 사용하지 않으므로 이를 상태레지스터를 이용하여 나타낸다.

4.2 CPU를 이용한 control signal합성

이전의 연구에서 GAP는 controller를 이용하여 제어하였다. 진화가 필요한 module이 32bit이상일 때는 데이터를 반복해서 처리하는데 이럴 경우 각 module마다 필요한 신호를 보내주기 위해 플립플롭과 카운터를 이용하여 각 module에 필요한 신호를 조합하였다[5].

이는 일정한 순서와 알고리즘을 적용하도록 되어있기 때문에 적합도 평가 module은 변경이 가능하지만 GAP의 신호는 변경을 할 수 없었다. 본 논문에서는 일반적인 프로세서에서 소프트웨어를 실행시켜 GAP를 control하는 방법을 제안한다.

그림 3는 control signal의 합성되는 알고리즘을 보여준다. control_singal은 GAP에 사용될 신호들이고 module_signal은 각 module에서 필요한 신호들로 메모리에 저장되어 있는 데이터를 불러온다.

그림 3. control signal 합성

```
procedure synthesis_signal()
for(i=chromosome/32){
  for(j=signal){
   control_signal[i+j]=
       control_signal[i+j] | module_signal[j]
  }
}
```

각 module은 crossover나 mutation등 필요한 유전자 알고리즘 연산자의 기능을 적용한다. 이 module들은 bitstring의 길이에 따라 알맞은 신호를 보내주어야 한다. 진화가 필요한 module이 여러 번의 동작을 통하여 전체 유전

자를 load할 경우 동기화를 위해 모듈에 데이터가 넘겨지는 순서에 따라 1 clock씩 delay가 되어야 한다. 이는 유전자을 길이에 따라 반복되는 횟수가 변경되므로 각 module에 필요한 신호를 OR연산하여 control_signal이 반복적으로 합성하여 필요한 횟수만큼 동작하도록한다.

그림 3에서 보는 바와 같이 32bit로 구성되어 있기 때문에 각 module은 전체 유전자를 32로 나눈 횟수만큼 반복하면 된다. 각 모듈의 수만큼 1 clock씩 delay하여 control 신호를 저장하여 사용하기 위해서 control_signal에 반복되는 회수만큼 각 module 신호가 control_signal의 하나 위쪽에 저장되도록 한다. 이를 통하여 각기 합성된 신호가 필요한 순간에 동시에 나오도록 한다.

4.3 GAP의 연산

유전자 알고리즘은 초기개체를 random하게 발생시켜 전역적 탐색을 한다. 이때 난수를 발생시키기 위해서 하드웨어에서는 RNG (Random Number Generator)가 필요한데 이곳에 필요한 더하기와 곱하기 연산을 하기 위해서는 이에 맞는 ALU가 필요하다. 하지만 CPU에서 이런 연산이 가능하기 때문에 따로 ALU를 설계하지 않고 필요한 연산을 가능하게 한다. CPU의 인터럽트를 사용하여 GAP에서 random number가 필요할 때 사용할 수 있다.

유전자 알고리즘에서 재생연산자를 사용할 때 또한 룰렛선택이나 토너먼트 선택을 할 때 적합도를 비교해야 하는데, 이때도 역시 가감산이 필요하다. 즉 GAP에서 필요한 ALU의 연산기능을 CPU로 전환함으로써 일반연산을 범용적으로 사용할 수 있다.

V. 결 론

스스로 회로를 구성하는 진화하드웨어는 인간이 알지못하는 상황에 유연하게 대처할 수 있으며 redundancy를 가지고 있기 때문에 자기 오류도 수정할 수 있다. 이러한 진화하드웨어는 재구성할 수 있는 하드웨어의 등장으로 발달하게 되었다. 또한 최근 SoC에 관한 연구가 활발히 진행되어 여러 기능을 하나의 칩으로 구성하는 연구가 활발히 진행 중이다.

본 논문에서는 진화하드웨어를 동작시키는 유전자 알고리즘 프로세서를 CPU로 제어하는 구조를 제안하였다. 이는 유전자 알고리즘을 하드웨어로만 동작하는 것이 아니라 소프트웨어로 control signal을 합성함으로써 유연성을 가지게 되고 CPU가 하나의 chip에 구현되어 다른 범용적인 용도와 함께 EHW가 사용될 수 있다.

VI. 참고문헌

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[2] Z. Michalewicz, *genetic Algorithms + Data Structures = Evolution Programs 3rd*, Springer-Verlag, Berlin, 1996.

[3] R. L. Haupt and S.E. Haupt, *Practical Genetic Algorithms*, John Wiley & Sons, New York, 1998.

[4] H, Mühlenbein, "Parallel genetic algorithms, population genetics, and combinatorial optimization," *in Proc. Third Int. Conf. Genetic Algorithms*, Morgan Kaufmann, SanFrancisco, pp416-421, 1989.

[5] 심귀보, 김태훈, "진화하드웨어를 위한 유전자알고리즘 프로세서," *한국퍼지 및 지능시스템학회 논문지*, Vol. 12, No. 5, pp462-466, 2002.

[6] 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.

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

[8] Vincent P. Heuring, Harry F. Jordan, *Computer Systems Design and Architecture*, Addison Wesley, 2002.