종분화를 이용한 다품종 하드웨어의 진화

황금성[O] 조성배
연세대학교 컴퓨터과학과
yellowg@candy.yonsei.ac.kr, sbcho@csai.yonsei.ac.kr

Diverse Hardware Evolution using Speciation

Keum-Sung Hwang[O] Sung-Bae Cho
Dept. of Computer Science, Yonsei University

요 약

진화 하드웨어(Evolvable Hardware: EHW)는 환경에 적응하여 스스로 하드웨어 구성을 변경할 수 있어서 근래에 많은 관심을 모으고 있는 분야이다. EHW는 목표 하드웨어를 탐색하기 위해 일반적으로 진화 알고리즘을 사용하는데, 진화 알고리즘은 하나의 목표 하드웨어 탐색 기능만을 수행한다. 본 논문에서는 종분화(Speciation) 알고리즘을 EHW에 적용하여 더욱 다양한 회로들을 얻을 수 있음을 보인다. 종분화 알고리즘은 동시에 여러 종의 해를 발견하게 해주고, 기존 진화 알고리즘에 비해 후반 탐색범위도 넓게 유지된다. 이를 6멀티플렉서의 진화에 적용한 결과, 다양한 품종의 하드웨어를 동시에 얻었고, 기존 진화 알고리즘에 비해 35%정도 빠른 세대에 해를 발견할 수 있었다.

1. 서론

진화 하드웨어는 환경에 적응적이고, 고장 복구 기능을 가지는 등 유용한 점이 많아서 활발한 연구가 이루어지고 있는 연구 분야이다[1]. EHW는 보통 진화 알고리즘(Evolutionary Algorithm: EA)을 이용한 학습으로 목적 하드웨어 구성을 탐색하는데, 일반적으로 유전자 알고리즘(Genetic Algorithm: GA)을 이용한다[2,3]. GA는 탐색 대상을 염색체로 표현한 다음, 여러 염색체를 세대를 거듭해 진화시켜 해를 탐색하는 진화 알고리즘이다. GA는 넓은 해 영역에 대해서 빠른 탐색 속도를 보이기 때문에, 하드웨어 구성방식이 다양한 진화 하드웨어에서 유용하게 쓰인다.

기존의 진화 하드웨어는 하나의 해만을 집중적으로 탐색하는데, 알고리즘의 특성상 우수한 염색체가 나타나면 그 염색체 중심의 해 영역으로 탐색 범위가 축소되는 것이다. 따라서 해 영역 전체를 동시에 고려하지 않고 하나의 해만을 얻게 된다. 하지만 종분화 알고리즘은 여러 종의 좋은 해를 찾는 것을 목적으로 하기 때문에, 발견된 우수한 염색체를 모두 중요시하여 좋은 유전자들을 유지하면서 그 주변을 탐색하며, 교차에 의해 새로운 영역에 대한 탐색도 계속한다. 그리고 진화의 후반까지 탐색 범위가 넓게 유지되는 특성 때문에 여러 해를 가진 문제에서 유용하다.

EHW에서 얻을 수 있는 목표 기능을 수행하는 하드웨어는 여러 개 존재한다. 같은 기능을 하는 하드웨어를 구성하는 방법이 다양하기 때문이다. 따라서 다양한 성질을 지닌 여러 하드웨어를 가지고 있다고 볼 수 있다. 종분화를 이용하면 EHW의 이러한 특성을 고려하여 탐색할 수 있다. 넓은 범위에 있는 다양한 종의 하드웨어 개체들을 모두 탐색하기 때문이다. 그리고 동시에 얻어진 여러 개의 하드웨어 구성들은 각각 독특한 특성을 가지고 있기 때문에, 예상치 못한 기능을 발견할 수도 있다.

본 논문에서는 제안한 방법의 유용성을 보이기 위하여 종분화를 적용시켜서 여러 종의 하드웨어를 동시에 얻고, 각 하드웨어의 특성을 분석하였다. 그리고 넓은 탐색범위를 유지하는 종분화 알고리즘과 기존 진화 알고리즘의 탐색 능력도 비교하였다.

2. 종분화된 진화 하드웨어의 구성

EHW를 구현하기 위해 OLMC 1개만을 포함한 간단한 GAL16V8을 설정하였다. 그리고 OLMC의 기능은 회로조합 기능만을 선택하여 사용하였다. 그림 1은 실험 모델이 된 간단한 GAL 회로를 나타낸다.

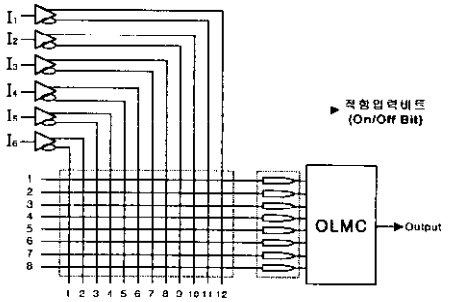


그림 1. 단순화한 GAL 회로

2.1 염색체 설계

본 논문의 염색체는 단순화한 GAL 칩에 맞게 설계하여 사용하였다. 그림 1의 점선영역을 보면, 히니의 OLMC에 들어가는 8개의 입력선의 조합 휴즈 제어부와 입력 허가 여부를 결정하는 8개의 적합입력 제어부가 있다. 이를 제어하는 비트를 유전자 코드로 구성하였다. 이 때, 휴즈비트가 'I_k AND not I_k'와 같은 조합일 때에는 값이 항상 '0'이 되어 의미가 없으므로, 이를 배제하기 위해 휴즈 2비트를 3치 숫자로 바꿔 표현하였다. 그림 2는 이렇게 표현한 염색체를 보여준다. 여기서 가장 오른

쪽의 8비트는 적항입력 제어비트이다.

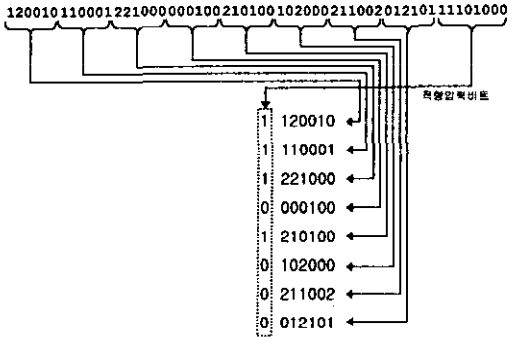


그림 2. 염색체 예제

2.2 적합도와 공유 적합도

목표 하드웨어로 6멀티플렉서를 선정하고, 다음과 같이 진화된 하드웨어와 목표하드웨어의 입출력 패턴을 비교하여 적합도로 사용하였다. 다음 수식은 적합도를 계산하는 방법을 나타낸다.

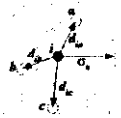
$$\text{적합도} = \frac{\text{매칭된패턴의수}}{\text{전체패턴의수}} \times 100$$

종분화를 적용하기 위해서 적합도 공유(Fitness Sharing) 방법을 사용하여 공유적합도(Shared Fitness)를 계산하였다[4,5]. 공유적합도 sf_i는 다음과 같이 계산한다.

$$sf_i = \frac{f_i}{\sum_{j=1}^{n} sh(d_{ij})}$$

여기서 f_i는 개체에 대한 적합도를 의미하며, n은 개체수를 의미한다. 그리고 $sh(d_{ij})$는 각 개체간 공유 상태를 나타내는데, 개체 i와 j의 차이점을 나타내는 d_{ij}를 이용해서 각 개체가 공유거리 내에 들어와 있는 정도를 아래와 같이 합산한 값이다. 아래 그림은 d_{ij}와 공유거리의 관계를 보여준다.

$$sh(d_{ij}) = \begin{cases} 1 - \dfrac{d_{ij}}{\sigma_s}, & for\ 0 \le d_{ij} < \sigma_s \\ 0, & for\ d_{ij} \ge \sigma_s \end{cases}$$



개체간 거리는 조합 입력식 별로 계산한 거리를 유클리드 거리(Euclidean distance)로 합산한 값이다. 조합 입력식 단위 계산은 유전자형 거리의 하나인 해밍거리(Hamming distance)를 이용하였으며, 전체 거리로의 합산은 아래 수식과 같이 유클리드거리를 이용하였다[5].

$$d_{ij} = \sqrt{(d_{ij}^1)^2 + (d_{ij}^2)^2 + (d_{ij}^3)^2 + (d_{ij}^4)^2 + (d_{ij}^5)^2 + (d_{ij}^6)^2 + (d_{ij}^7)^2 + (d_{ij}^8)^2}$$

수식에서 d_{ij}^k 는 k번째 조합식에 대한 개체 i와 j의 해밍거리이다. 적항입력 제어신호가 다른 경우에는 조합입력식이 다르다고 간주할 수 있으므로, 거리를 최대값으로 주었다.

2.3 유전자 연산

선택은 다음 수식과 같이 각 개체의 공유적합도에 비례하는 확률로 수행하였다. 세대의 최대 공유적합도에 대한 각 개체의 공유적합도를 선택확률로 하여, 최대 적합도를 가진 개체는 무조건 선택될 수 있도록 하였다.

$$\text{선택확률} = \frac{\text{개체별공유적합도}}{\text{해당세대의최대공유적합도}} \times 100$$

교차 방법으로는 1점 교차, 일정교차, 조합 입력식 단위의 일정교차(그림 3)를 사용하였다. 일정교차는 두 개의 염색체에 대해 비트단위 교차여부를 확률적으로 정하는 방법인데[2], 유전자 교환이 빠르게 일어나도록 하기 위해서 사용하였다. 그리고 염색체 구조상 조합 입력식 단위별로 독립적인 특성을 지니기 때문에, 특성을 유지하면서 교차가 이루어질 수 있도록 하기 위해, 조합 입력식 단위의 일정교차를 고안하여 사용하였다. 실험에서는 이 세 가지 교차 방식을 특별한 조건없이 같은 확률로 수행되도록 하였다. 그림 3은 조합 입력식 단위의 일정교차를 보여준다. 그림에서처럼 적항입력 제어비트도 조합식에 맞춰 각 비트별로 교차를 수행하였다.

```
121011200101100012210011001000210100102000211002    00100011
120010110001221000000100021010010200021100201210    11110100
1.....0.....1.....1.....0.....0.....0.....1.....    10110001
120010110001221000000100021010010200021100 201210  11110100
120010110001221000000100 021010010200021100 201210  11110100
```

그림 3. 조합 입력식 단위의 일정교차의 예

돌연변이는 각 유전자에 대해 돌연변이 확률에 의해 일어나도록 하였다. 임의의 적항비트 하나의 값을 바꾸는 돌연변이와 임의의 적항비트 하나의 값을 바꾸되, 바뀐 값이 '1'이면 그에 해당되는 조합회로의 비트열을 모두 임의의 값으로 바꾸는 돌연변이를 같은 확률로 적용하였다.

3. 실험 결과

진화를 위한 환경변수는 개체수 50, 돌연변이율 0.02, 교차율 0.4, 공유거리 3.0, 결과 생성률 0.3으로 하였고, 필요한 경우 몇몇 환경변수를 바꿔가며 실험하였다. 여기서 결과 생성률이란 본 논문에서 종분화 알고리즘의 종료조건으로 제시한 것이다. 한 세대의 개체수에 대한 해의 수 비율이 결과 생성률에 도달하였을 때 진화가 멈추도록 하였다.

그림 4는 종분화를 적용한 진화 하드웨어의 진행 과정을 세대별로 보여준다. 그림에서 위 그래프는 최대 적합도를, 아래 그래프는 평균 적합도를 나타낸다. 최초 해 발견은 901세대에 일어났고, 결과 생성률인 30%의 해를 만족하는 세대는 1216세대로 나왔다.

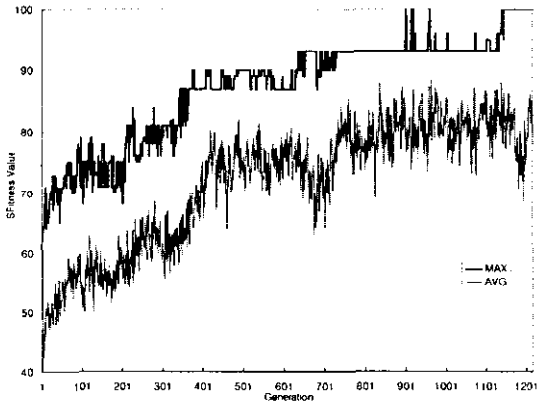


그림 4. 공유적합도 값의 세대별 변화

실험 결과 30%의 결과 생성률에 의해 15개의 해가 나왔으나, 같은 염색체를 제외하여 13개의 최종 염색체를 얻었다. 이 13개의 염색체는 모두 다른 구조를 가지면서도 같은 6멀

티플렉서 기능을 하는 하드웨어를 나타낸다. 해들의 차이를 분석해보기 위해 단일 연결 클러스터링(single linkage clustering)을 하여 덴드로그램(dendrogram)을 작성하여 보았다[6]. 그림 5에서 볼 수 있듯이 13개의 개체간 거리는 3.3~6.7의 거리 분포를 보이고 있다. 조합시 하나의 최대거리가 2.4이므로 적어도 하나의 조합단위 이상의 차이를 가짐을 의미한다.

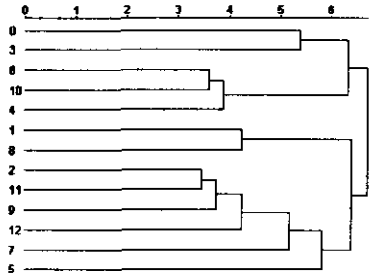


그림 5. 단일거리 클러스터링 결과

덴드로그램의 거리 6지점에서 개체들을 4개의 집단으로 나눈 다음, 대표 개체들을 선택하여 게이트 회로를 그리고 분석하였다(그림 6). 그리고 이들 회로구조의 차이를 분석한 결과는 표1과 같다.

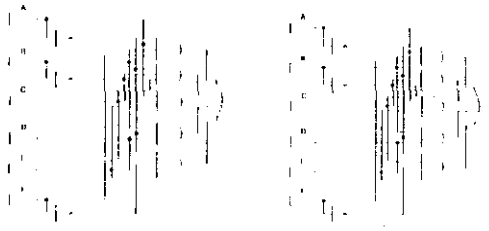


그림 6. 개체 0과 개체 6의 게이트 회로도

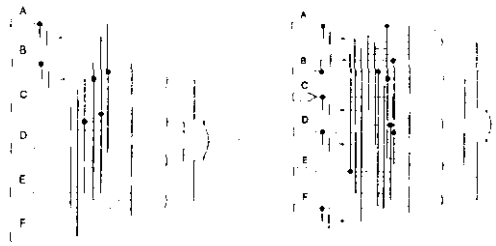


그림 7. 개체 1과 개체 2의 게이트 회로도

	개체 0	개체 6	개체 1	개체 2
AND 게이트 입력수	17	17	12	18
OLMC 입력수	5	6	4	5
사용된 외부입력수	9	9	8	11

표 1. 4개체의 게이트 회로구조 비교

표 1을 보면 개체 1이 가장 적은 수의 게이트와 입력을 사용하고 있다. 그림 7의 회로도를 보아도 가장 간단하고 효율적인 구조를 나타내고 있음을 알 수 있다. 따라서, 가장 효율적인 회로를 선택한다면 개체 1이 적절하다.

이처럼 다양하게 얻어진 하드웨어들을 가지고 고장 상태와 같은 여러 상황에 따른 분석을 하면, 좀더 우수한 하드웨어를 선택할 수 있다. 예를 들어, OLMC 입력부가 고장이 나서 1번째 입력을 받아들이지 못하는 경우, 위 네 개의 개체 중 개체 2를 사용하면 정상적으로 작동을 한다. 개체 2는 구조상 첫번째 AND 게이트를 제거해도 개체 1과 같은 조합의 입력만이 남기 때문에, 제대로 된 기능을 유지하는 것이다. 이처럼 얻어진 여러 하드웨어 구조를 예비하고 있으면 고장상황에 대처할 수 있다.

종분화 알고리즘의 탐색 능력을 알아보기 위해서, 종분화를 한 하드웨어의 해 발견 속도를 기존 방식과 비교해 보았다. 앞서 말했듯이 종분화는 해 탐색 영역이 넓게 유지되기 때문에 해 발견에 유리하다. 속도를 비교하기 위해 무조건 해를 발견할 때까지 50번씩 실험한 다음 비교하였다.

	기존 진화	종분화된 진화
평균 발견 세대	1147.16	747.46
가장 빠른 발견 세대	143	77
가장 늦은 발견 세대	5373	2485
발견 세대 표준 편차	1061.35	584.70

표 2. 해 발견 세대 비교

표 2를 보면 종분화를 한 경우가 그렇지 않은 경우보다 좋은 결과를 보여줌을 알 수 있다. 종분화를 한 경우, 평균 발견 세대가 상당히 앞서고, 표준 편차도 적게 나와 고른 해 발견 성능을 보여주었다. 따라서 EHW는 종분화 알고리즘을 적용하는 것이 빠른 세대의 발견에 유리함을 알 수 있다.

4. 결론

본 논문에서 우리는 진화 하드웨어에 종분화의 특성을 이용하여 다양한 장점을 얻을 수 있었다. 한 번의 진화로 다양한 하드웨어를 얻음으로써 새로운 하드웨어나 우수한 하드웨어의 발견 가능성을 볼 수 있었고, 고장상황에 대한 대비도 생각할 수 있었다. 그리고 종분화를 적용한 진화 하드웨어의 발견 속도가 기존 알고리즘에 비해 빠르고 우수해서, 앞으로 종분화를 적용한 진화 하드웨어의 활발한 연구를 예상할 수 있다.

다만, 아직까지 종분화 방식은 공유적합도 계산에 시간이 많이 소모되어 수행 속도가 느리다. 따라서 하드웨어를 이용한 병렬처리나, 종분화 알고리즘의 개선을 통해 이를 해결할 필요가 있다. 그리고 좀더 복잡한 하드웨어에 대해서도 좀더 구체적으로 연구해 볼 필요가 있다.

참고 문헌

[1] X. Yao and T. Higuchi, "Promises and challenges of evolvable hardware," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 29, pp.87-97, Feb. 1999.

[2] T. Higuchi, 등, "유전자 학습에 의한 하드웨어 진화의 기초실험," *유전자 알고리즘*, 대청정보시스템(주), pp.365-393, 1996.

[3] T. Higuchi, et al., "Evolving hardware with genetic learning," *Proc. of Simulated Adaptive Behvior*, MIT Press, 1993.

[4] P. Darwen and X. Yao (1996b), " Every niching method has its niche: fitness sharing and implicit sharing compared," *Lecture Notes in Computer Science*, Springer-Verlag. Berlin, pp.398-407, 1996.

[5] S. W. Mahfoud, "Niching methods," *Evolutionary Computation 2, Advanced Algorithms and Operators*. Institute of Physics Publishing, pp.87-92, 2000.

[6] A. D. Gordon, "Classification: Methods for the exploratory analysis of Multivariate Data," *Chapman and Hall*, 1981.