https://doi.org/10.5392/JKCA.2021.21.07.164

회로 분할을 위한 어댑티드 유전자 알고리즘 연구

A Study of Adapted Genetic Algorithm for Circuit Partitioning

송호정*, 김현기**
충북대학교 컴퓨터공학과*, 강동대학교 소방안전과**

Ho-Jeong Song(hjsong@chungbuk.ac.kr)*, Hyun-Gi Kim(king@gangdong.ac.kr)**

요약

VLSI 설계에서의 분할(partitioning)은 기능의 최적화를 위하여 설계하고자 하는 회로의 그룹화(grouping)하는 단계로서 레이아웃(layout)에서 면적과 전파지연의 최소화를 위해 함께 배치할 소자를 결정하는 문제이다. 이러한 분할 문제에서 해를 얻기 위해 사용되는 알고리즘은 Kernighan-Lin 알고리즘, Fiduccia Mattheyses heuristic, 시뮬레이티드 어닐링, 유전자 알고리즘 등의 방식이 이용된다. 본 논문에서는 회로 분할 문제에 대하여 유전자 알고리즘과 확률 진화 알고리즘을 결합한 어댑티드 유전자 알고리즘을 이용한 해 공간 탐색(solution space search) 방식을 제안하였으며, 제안한 방식을 유전자 알고리즘 및 시뮬레이티드 어닐링 방식과 비교, 분석하였고, 어댑티드 유전자 알고리즘이 시뮬레이티드 어닐링 및 유전자 알고리즘보다 더 효과적으로 최적해에 근접하는 것을 알 수 있었다.

■ 중심어 : | 유전자 알고리즘 | 회로 분할 | 시뮬레이티드 어닐링 | 어댑티드 유전자 알고리즘 |

Abstract

In VLSI design, partitioning is a task of clustering objects into groups so that a given objective circuit is optimized. It is used at the layout level to find strongly connected components that can be placed together in order to minimize the layout area and propagation delay. The most popular algorithms for partitioning include the Kernighan-Lin algorithm, Fiduccia-Mattheyses heuristic and simulated annealing. In this paper, we propose a adapted genetic algorithm searching solution space for the circuit partitioning problem, and then compare it with simulated annealing and genetic algorithm by analyzing the results of implementation. As a result, it was found that an adaptive genetic algorithm approaches the optimal solution more effectively than the simulated annealing and genetic algorithm.

■ keyword : | Genetic Algorithm | Circuit Partition | Simulated Annealing | Adapted Genetic Algorithm |

I. 서 론

VLSI(Very Large Scale Integration) 시스템 설계 과정에서 칩의 규모나 계산 복잡도 규모 제한 등으로 설계하고자 하는 회로를 여러 개의 부회로(sub-circuit)로 분할하여 설계하는 경우가 많이 있다. 이러한 회로의 분할은 on-chip 연결과 off-chip 연결(cut)을 발생시키게 되며, off-chip 연결은 연결 배선에 의한 성능감소 및 신뢰도 저하 등의 결과를 가져오게 된다.

본 논문에서는 VLSI의 physical design[1] 과정 중

접수일자 : 2021년 03월 26일
수정일자 : 2021년 06월 08일
심사완료일 : 2021년 06월 08일
교신저자 : 김현기, e-mail : king@gangdong.ac.kr

부회로 사이의 연결 배선을 최소로 하는 회로 분할에 대하여 유전자 알고리즘[2]과 확률 진화 알고리즘[3]을 결합한 해 공간 탐색방식을 제안하였으며, 이 방식을 유전자 알고리즘(Genetic Algorighm; GA) 및 시뮬레이티드 어닐링(simulated annealing; SA)[4] 방식과 비교하여 분석하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 회로 분할 문제의 정의와 시뮬레이티드 어닐링 알고리즘에 대하여 알아보고, Ⅲ장에서는 본 논문에서 제안한 어댑티드 회로 분할 유전자 알고리즘의 데이터 표현, 연산자, 수행 내용에 대하여 설명한다. Ⅳ장에서는 제안한 어댑티드 유전자 알고리즘(Adapted Genetic Algorithm; AGA)을 유전자 알고리즘 및 시뮬레이티드 어닐링 방식을 비교하였고, 마지막으로 Ⅴ장에서는 결론과 향후 연구 방향에 대하여 기술한다.

Ⅱ. 회로 분할 문제

1. 문제 정의

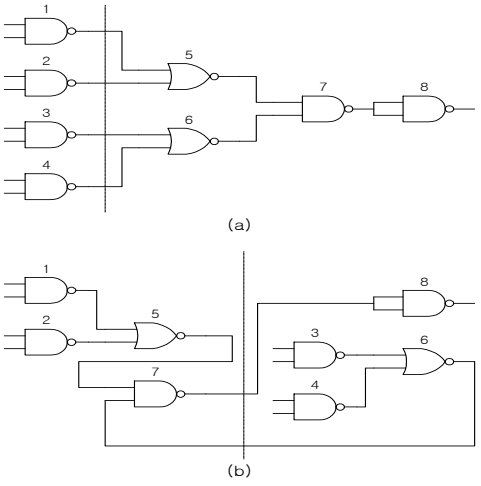

그림 1. 분할 문제

VLSI 설계에서의 분할 문제는 기능의 최적화 및 연결의 최소화를 위하여 설계하고자 하는 회로의 그룹화로 회로의 면적과 전파지연의 최소화를 위해 함께 배치할 소자를 결정하거나 계산 복잡도를 낮추기 위해서나 또는 칩 규모 등의 제한을 만족시키기 위해 회로를 분할하는 문제이다. 즉, 회로 분할의 목적은 분할로 분리된 소자들 사이의 연결이 최소가 되도록 각각의 소자들을 그룹화 하는 것이다[1][5].

[그림 1]의 (a) 회로 분할에는 두 개의 그룹 사이에 4개의 연결이 있으나, 소자 5,7과 3,4를 교환 하면 [그림 1]의 (b)에서와 같이 연결을 2개로 줄일 수 있다. 이와 같이 회로 분할 문제는 두 그룹 사이의 연결을 최소화하는 해를 구하는 것이다.

회로 분할 문제에서 최적해를 얻기 위한 알고리즘은 Kernighan-Lin 알고리즘[6], Fiduccia Mattheyses heuristic[7], 시뮬레이티드 어닐링등의 방식들이 있다.

Ⅲ. 어댑티드 회로 분할 유전자 알고리즘

유전자 알고리즘은 자연의 진화(evolution) 과정에 기초한 계산 모델에서 유도된 탐색방식으로, 생물의 진화를 모방한 진화 연산의 대표적인 기법으로, 실제 진화의 과정에서 많은 부분을 차용한 알고리즘이다.

유전자 알고리즘은 염색체와 유사한(chromosome-like) 자료구조를 사용하여 해 공간을 부호화하고, 이러한 자료구조에 교배, 돌연변이 등의 재조합 연산자(recombination operator)를 적용하여 염색체들을 진화시킨다.

유전자 알고리즘은 처음에 임의로 선택된 염색체 집단(population of chromosome)을 기반으로 시작하며, 염색체 집단 중에서 일정한 방식으로 부모 염색체를 선택하고 이들 부모 염색체를 교배(crossover)시켜 자식 염색체를 생성한다. 새로 생성된 자식 염색체는 평가함수(evaluation function)에 의해 평가되며 높은 평가 결과를 가지는 염색체가 다음 세대에 살아남을 확률이 높게 된다. 이와 같은 방식으로 유전자 알고리즘은 염색체 집단의 진화를 통하여 최적해에 근접할 수 있으므로[8-10], 최적해를 구하기 어려운 여러 NP-문제에 적용될 수 있다[11][12].

1. 염색체의 표현(Representation)

회로 분할 문제는 회로를 구성하는 소자들을 최소 비

용으로 그룹화 하는 것으로 유전자 알고리즘을 회로 분할 문제로 표현하기 위해서는 각 소자들이 속한 그룹을 나타낼 수 있어야 한다.

회로 분할 문제에서 각 그룹에 포함되는 소자의 수가 n으로 같다고 가정할 때, 각 그룹에 속해있는 소자들을 다음과 같은 염색체로 표현할 수 있다. [그림 2]는 소자 2, 5, 8, 1 및 3이 그룹 A에 속해있고, 소자 6, 9, 10, 7 및 4가 그룹 B에 속해있는 것을 나타내고 있다.

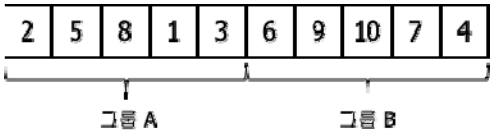

그림 2. 염색체의 표현

[그림 2]는 하나의 해에 대하여 표현한 방법이지만 [그림 3]에 보인 바와 같이 여러 가지 표현이 가능하므로 좋은 표현 방법이 아니다. 본 논문에서는 [그림 2]의 표현방식을 수정하여 [그림 4]와 같이 각 그룹에 포함된 소자(이후로는 유전인자[gene]로 표현)들을 오름차순으로 정렬하여 하나의 해는 하나의 표현방식으로 나타낼 수 있도록 정의하였다.

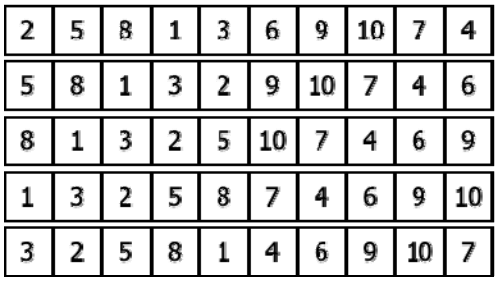

그림 3. 같은 해의 여러 표현 들

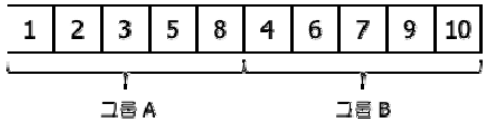

그림 4. 수정된 염색체의 표현

2. 평가함수(Evaluation Function)

유전자 알고리즘이 진행되는 동안 현재 모집단(population)의 개체들은 특정 평가함수에 의해 평가된다. 회로 분할 문제에서의 평가함수는 분할된 두 그룹 사이의 연결이 가지는 가중치(weight)의 합을 나타내며, Cost(A,B)로 표시한다[식 1]. 여기서 가중치 W는 회로의 중요도에 따라 설계자에 의해 정해지는 고유의 값으로 높은 가중치를 가지는 연결에 속한 소자는 같은 그룹에 포함 될 확률을 높이는 역할을 한다. 또한 가중치 W를 모두 1로 설정하면 두 그룹 사이의 연결의 개수가 Cost를 의미하게 된다.

$$Cost(A,B) = \sum_{n \in cutset} W_n \qquad \text{(식 1)}$$

3. 교배 연산자(Crossover Operator)

본 논문에서 사용한 TGX(Two-way Grouped Crossover) 연산자는 PMX(Partially Mapped Crossover)[13] 의 한 형태로 볼 수 있으며, 구체적인 연산 방식은 다음과 같다[그림 5].

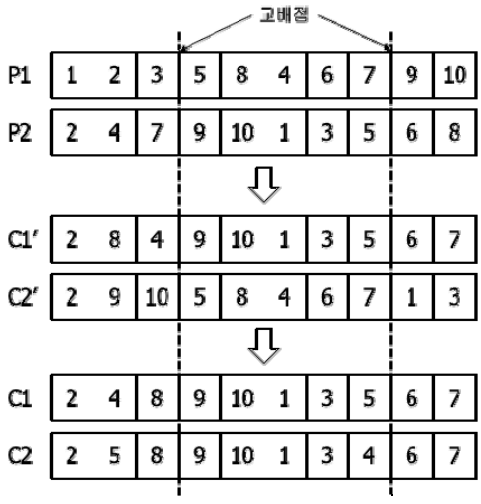

그림 5. 교배 연산

Step 1 : 부모 염색체 P1과 P2에서 각 그룹에 속하는 교배점(cross point)을 하나씩 랜덤하게 선택한다.

Step 2 : 부모 염색체 P1에서 두 개의 교배점에 의해 만들어진 부염색체(sub- chromosome)를

자식 염색체 C2'의 같은 위치에 복사한다. 자식 염색체 C1'도 같은 방식으로 복사한다.

Step 3 : 부모 염색체 P2에서 자식 염색체 C2'에 있는 유전인자를 삭제하고 P2의 나머지 유전인자를 C2'의 빈 곳에 순서대로 채운다. 자식 염색체 C1'도 같은 방식으로 빈 곳을 채운다.

Step 4 : 새로 생성된 자식 염색체 C1'과 C2'를 각 그룹 별로 오름차순으로 정렬을 하여 최종 자식 염색체 C1과 C2를 생성한다.

4. 돌연변이(Mutation)

유전자 알고리즘에서 각 세대의 모집단은 진화를 진행하면서 얻고자 하는 해에 가까운 염색체들로 구성된 모집단으로 수렴하게 되지만, 그 결과가 최적해가 아닌 국부해로 수렴할 수도 있으며, 이러한 국부해로의 수렴을 막기 위하여 돌연변이(mutation) 연산을 수행하게 된다.

본 논문에서 사용한 돌연변이 연산 방식은 다음과 같다[그림 6].

Step 1: 염색체에서 각 그룹에 속하는 돌연변이 위치(mutation point) m1과 m2를 랜덤하게 선택한다. ($1 \le m1,m2 \le n$)

Step 2: 선택된 돌연변이 위치에 속하는 유전인자를 서로 교환한다.

Step 3: 각 그룹을 오름차순으로 정렬한다.

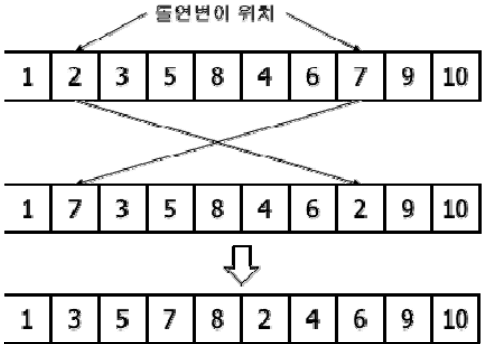

그림 6. 돌연변이 연산

5. 어댑티드 회로 분할 유전자 알고리즘

알고리즘 1은 앞에서 기술한 방식을 사용한 어댑티드 회로 분할 유전자 알고리즘을 나타낸다.

알고리즘 1. 어댑티드 회로 분할 유전자 알고리즘

Step 0: 파라미터들의 설정
개체의 수를 나타내는 pop_size, 돌연변이 율을 나타내는 Pm, 그리고 최대 생성 횟수를 나타내는 max_gen을 설정한다.

Step 1: 초기 모집단의 생성
각기 다른 pop_size 만큼의 객체 Si(i=1, …, pop_size)를 랜덤하게 생성한다.

Step 2: 모든 모집단의 비용을 계산하고 최대, 최소 객체를 기억한다.

Step 3: 교배

3.1: 모집단으로부터 두 개의 부모 염색체 P1과 P2를 랜덤하게 선택한다.

3.2: PMX 연산자를 사용하여 새로운 자식 염색체 C1과 C2를 생성

Step 4: 새로운 세대 구성

4.1: 자식 염색체 C1 또는 C2가 모집단 내의 최대 비용을 갖는 염색체보다 비용이 작다면 해당 염색체를 자식 염색체로 대체하고, 비용이 크면 확률값에 의해서 대체한다.

4.2: 자식 염색체 C1 또는 C2가 모집단 내의 최적의 해와 같다면 국부해에 도달한 것으로 간주하고 돌연변이 연산을 수행한다.

Step 5: 새로운 염색체의 생성 횟수가 max_gen이 될 때까지 단계 1부터 4까지를 반복한다. 최적해는 모집단 내에서 가장 작은 비용을 갖는 염색체이다.

Ⅳ. 시뮬레이션

본 논문에서 제안한 어댑티드 유전자 알고리즘을 시뮬레이션하기 위하여 각각 10, 12, 14, 16, 18 및 20개의 노드를 갖고, 랜덤 에지(edge)와 랜덤 가중치를 갖

는 그래프를 각 4종류씩 자동으로 생성하고 동일 그래프에 유전자 알고리즘과 시뮬레이티드 어닐링, 어댑티드 유전자 알고리즘을 각 100번씩 수행시켰고, 또한 회로분할 문제에 사용되는 벤치마크 회로인 fract와 primary1에 대하여 동일 알고리즘을 각 100번씩 수행시켜 그 결과를 분석하였다.

시뮬레이티드 어닐링은 조합 최적화 문제 해결에 적용되는 대표적 반복 휴리스틱 알고리즘으로 Timber-Wolf[14]등 실용 패키지뿐 아니라 알고리즘의 성능을 비교하는 벤치마크로 주로 사용된다.

본 시뮬레이션에서는 유전자 알고리즘의 돌연변이율 Pm=0.1, 모집단 내의 개체의 수 pop_size=20, 최대 수행 횟수 max_gen=10000, 시뮬레이티드 어닐링의 한 온도에서의 알고리즘 수행 횟수 M=10, 최대 수행 횟수 max_time=10000, 초기 온도 T=10, 냉각 파라미터 α=0.9, 그리고 β=1.0의 값을 사용하였다.

표 1. 생성된 회로의 네트리스트

네트	가중치
N1 = {C1, C5, C8, C4}	3
N2 = {C2, C10}	1
N3 = {C3, C9}	2
N4 = {C4, C5, C6, C9}	4
N5 = {C5, C6, C9}	3
N6 = {C6, C4, C10, C3, C9}	4
N7 = {C7, C10}	2
N8 = {C8, C1}	2
N9 = {C9, C10, C5, C4, C8}	3
N10 = {C10, C4, C9, C1}	5

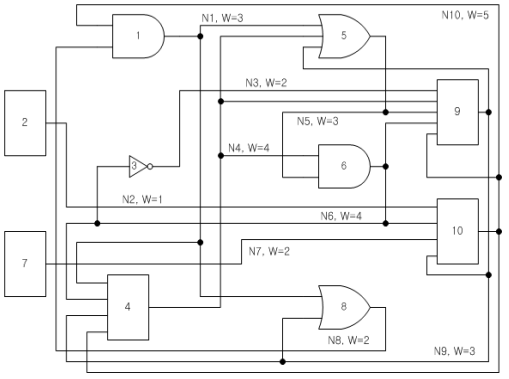

그림 7. 생성된 회로

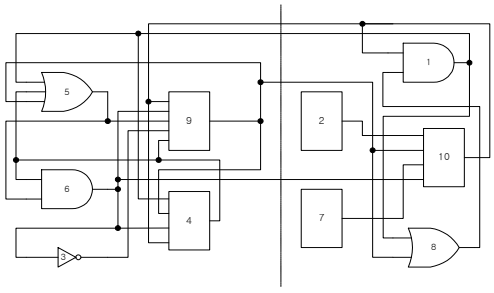

그림 8. 분할 결과

[표 1]은 자동으로 생성된 그래프의 네트 리스트와 가중치의 값을 표시한 예이며, [그림 7]은 [표 1]과 같은 네트 리스트와 가중치의 값을 갖는 그래프를 회로로 표현한 예이다. [그림 7]의 회로를 시뮬레이티드 어닐링과 유전자 알고리즘, 어댑티드 유전자 알고리즘을 각각 적용하여 얻은 최적해는 [그림 8]로 동일하며 비용이 15인 두 개의 그룹으로 분할되었음을 알 수 있다. 본 논문에서는 자동생성 회로와 fract, primary[15]에 대한 시뮬레이티드 어닐링과 유전자 알고리즘 적용 결과의 최적값, 최악값, 평균값을 구하여 각 알고리즘을 비교하였다[표 2][표 3].

표 2. SA와 GA 및 AGA의 자동생성 회로 테스트 결과

노드 개수	그래프 종류	시뮬레이티드 어닐링			유전자 알고리즘			어댑티드 유전자 알고리즘		
		최적	최악	평균	최적	최악	평균	최적	최악	평균
10	1	15	15	15	15	15	15	15	15	15
	2	19	19	19	19	19	19	19	19	19
	3	17	17	17	17	17	17	17	17	17
	4	19	20	19	19	19	19	19	19	19
12	5	20	21	20.3	20	20	20	20	20	20
	6	22	24	22.1	22	22	22	22	22	22
	7	20	24	20.1	20	20	20	20	20	20
	8	17	21	17.3	17	17	17	18	17	17.5
14	9	31	32	31	31	31	31	31	31	31
	10	13	13	13	13	13	13	13	13	13
	11	27	29	27.5	27	27	27	27	27	27
	12	18	19	18.2	18	18	18	18	18	18
16	13	18	26	19.6	18	18	18	18	18	18
	14	20	25	21.6	20	23	20	20	23	20
	15	24	29	24.6	24	24	24	24	24	24
	16	11	11	11	11	11	11	11	11	11
18	17	27	29	27.6	27	27	27	27	27	27
	18	25	29	26.1	25	25	25	25	25	25
	19	15	19	16.5	15	17	16.3	16	17	16.3
	20	28	33	29.4	28	32	28	28	32	28
20	21	35	41	37.3	35	39	35.1	35	39	35.1
	22	25	33	25.9	25	26	25	25	26	25
	23	31	37	31.8	31	31	31	31	31	31
	24	17	20	17.2	17	17	17	17	17	17

표 3. SA와 GA 및 AGA의 벤치마크 테스트 결과

노드 개수	그래프 종류	시뮬레이티드 어닐링			유전자 알고리즘			어댑티드 유전자 알고리즘		
		최적	최악	평균	최적	최악	평균	최적	최악	평균
125	fract	45	59	51.9	41	55	46.6	41	55	46.4
752	primary1	183	207	200	175	205	188	174	204	187.6

[표 2]와 [표 3]에서 노드 개수는 그래프의 노드 개수를 의미하며 그래프 종류는 동일한 노드 개수를 갖는 서로 다른 그래프를 의미한다. 또한 최적, 최악, 평균은 각 그래프에 대해 SA, GA 및 AGA 세 개의 알고리즘을 100번씩 수행한 100개의 결과 중 최적 비용, 최악 비용 그리고 비용의 평균값을 의미한다.

테스트 결과를 분석해보면 자동생성 회로 및 테스트 벤치 회로 모두 본 논문에서 제안한 AGA가 가장 좋은 결과를 보인다. 다만 노드 수가 적은 회로의 경우 최적값의 경우 큰 차이가 없어 보이며, 최악값의 경우 GA 및 AGA에서 조금 더 좋은 결과를 얻는 것을 확인할 수 있으며, 노드 수가 많은 primary1 회로의 경우 최적값은 AGA가 SA대비 약 5%, GA 대비 약 1% 정도 좋은 결과를 얻은 것을 확인할 수 있다. 이는 노드 수가 적은 회로에서는 경우의 수가 적어 쉽게 최적해에 도달할 수 있으므로 세 가지 알고리즘 모두 크게 차이가 없지만 노드 수가 많은 회로에서는 SA 보다는 GA가, GA 보다는 AGA가 좀 더 효과적으로 최적해에 수렴할 수 있기 때문이다[그림 9-11].

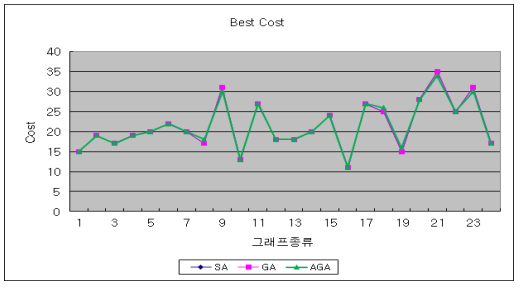

그림 9. 최적 비용 비교

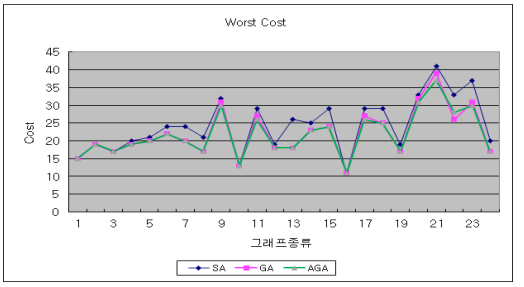

그림 10. 최악 비용 비교

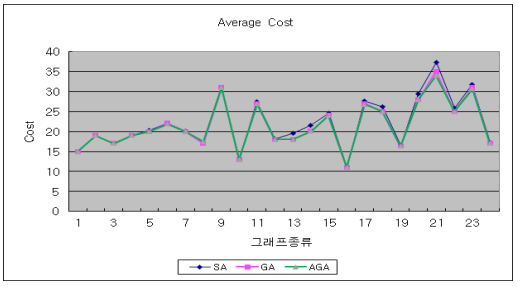

그림 11. 평균 비용 비교

V. 결론

본 논문에서는 VLSI 설계 과정 중 회로 분할 문제에 대하여 어댑티드 유전자 알고리즘을 이용한 회로 분할 방식을 제안하였으며, 제안한 방식을 시뮬레이티드 어닐링 방식 및 유전자 알고리즘 방식과 비교, 분석하였다. 제안한 어댑티드 회로 분할 유전자 알고리즘을 시뮬레이티드 어닐링 방식 및 유전자 알고리즘과 비교, 분석한 결과 어댑티드 회로 분할 유전자 알고리즘이 시뮬레이티드 어닐링 방식뿐만 아니라 유전자 알고리즘보다 더 효과적으로 최적해에 근접하는 것을 알 수 있었다.

앞으로 분할 문제를 위한 보다 더 효과적인 염색체의 표현방식과 연산자들이 연구되어야 할 것이며, physical design 과정의 다른 문제들에 대해서도 어댑티드 유전자 알고리즘의 적용에 관한 연구가 필요하다고 생각된다.

참 고 문 헌

[1] S. M. Sait and H. Youssef, *VLSI Physical Design Automation Theory and Practice*, World Scientific Publishing, 2001.

[2] Z. Michalewicz, *Genetic Algorithm + Data Structure = Evolution Programs,* 3rd edition, Springer-Verlag, New York, 1996.

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

[4] S. kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science, Vol.220, No.4598, pp.671-680, 1983.

[5] J. Cong, L. Hagen, and A. Kahng, "Net partitions yield better module partitions," In 29th Design Automation Conference, pp.47-52, 1992.

[6] K. H. Kernighan and S. Lin, "An Efficient Heuristic Procedure for Partitioning Graph," Bell System Technical Journal, Vol.49, No.2, pp. 291-307, February, 1970.

[7] C. M. Fiduccia and R. M. Mattheyses, "A linear time heuristic for improving network partitiions," in 19th Design Automation conference, pp.175-181, 1982.

[8] M. Gen and R. Cheng, *Genetic Algorithms and Engineering Optimization*, Wiley-Interscience publication, 2000.

[9] T. Bäck, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New Your, 1996.

[10] M. Gen and M. Cheng, *Genetic Algorithms and Engineering Design*, Wiley, New Your, 1997.

[11] S. M. Sait and H. Youssef, *Iterative Computer Algorithms with Applications in Engineering*, Computer Society, 1999.

[12] D. Goldberg, R. Alleles, "loci and the Traveling Salesman Problem," proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum Associates, N. J. Hillsdale, pp.154-159, 1985.

[13] C. Cheng and Y. A. Wei, "An Improved two-way Partitioning algorithm with stable performance," IEEE Transactions on Computer-Aided Design, Vol.10, No.12 pp.1502-1511, 1991.

[14] C. Sechen and A. L. Sangiovanni-Vincentelli, "Timberwolf3.2 : A new standard cell placement and global routing package," Proceedings of 23rd Design Automation Conference, Vol.1, pp.432-439, 1986.

[15] K. Kozminski, "Benchmarks for layout synthesis - evolution and current status," Institute of Electrical and Electronics Engineers, Design Automation Conference, pp.265-270, 1991.

저 자 소 개

송 호 정(Ho-Jeong Song) 정회원

- 1994년 2월 : 배재대학교 물리학과 (이학학사)
- 1996년 8월 : 청주대학교 전자공학과(공학석사)
- 2003년 2월 : 충북대학교 컴퓨터공학과(공학박사)
- 현재 : 원텍(주) 수석연구원

〈관심분야〉 : VLSI 설계, High-level Synthesis

김 현 기(Hyun-Gi Kim) 정회원

- 1986년 : 호서대학교 정보통신과 졸업(공학사)
- 1992년 : 호서대학교 대학원 정보통신과 졸업(공학석사)
- 2002년 : 청주대학교 대학원 전자공학과 졸업(공학박사)
- 1996년 ~ 현재 : 강동대학교 소방안전과 부교수

〈관심분야〉 : VLSI & CAD, 실시간 정보처리, 컴퓨터 네트워크