Placement 확률 진화 알고리즘의 설계와 구현

Design and Implementation of a Stochastic Evolution Algorithm for Placement

송호정, 송기용
Ho-Jeong Song, Gi-Yong Song
충북대학교 컴퓨터공학과 · 컴퓨터정보통신연구소

요 약

배치(Placement)는 VLSI 회로의 physical design에서 중요한 단계로서 회로의 성능을 최대로 하기 위하여 회로 모듈의 집합을 배치시키는 문제이며, 배치 문제에서 최적의 해를 얻기 위해 클러스터 성장(cluster growth), 시뮬레이티드 어닐링(simulated annealing; SA), ILP(integer linear programming)등의 방식이 이용된다. 본 논문에서는 배치 문제에 대하여 확률 진화 알고리즘(stochastic evolution algorithm; StocE)을 이용한 해 공간 탐색(solution space search) 방식을 제안하였으며, 제안한 방식을 시뮬레이티드 어닐링 방식과 비교, 분석하였다.

Abstract

Placement is an important step in the physical design of VLSI circuits. It is the problem of placing a set of circuit modules on a chip to optimize the circuit performance. The most popular algorithms for placement include the cluster growth, simulated annealing and integer linear programming. In this paper we propose a stochastic evolution algorithm searching solution space for the placement problem, and then compare it with simulated annealing by analyzing the results of each implementation.

Keyword : stochastic evolution algorithm, simulated annealing, slicing structure, placement

I. 서론

배치는 VLSI 회로의 설계 설계(physical design)에서 중요한 단계로써 회로의 성능을 최적화하기 위해 주어진 제약을 고려하여 칩상에 회로 모듈들을 배치시키는 문제이다[1,2,3].

본 논문에서는 VLSI의 설계 설계 과정 중 회로 모듈의 집합을 상호연결의 길이와 칩 면적을 고려하여 배치시키는 배치 문제에 대하여 확률 진화 알고리즘[4,6]을 이용한 해공간 탐색 방식을 제안하였으며, 이 방식을 시뮬레이티드 어닐링 방식과 비교하여 분석하였다.

본 논문의 구성은 다음과 같다. Ⅱ장에서 배치 문제에 대하여 알아보고, Ⅲ장에서는 본 논문에서 제안한 placement 확률 진화 알고리즘의 데이터 표현 방법과 알고리즘에 대하여 설명한다. Ⅳ장에서는 제안한 배치 확률 진화 알고리즘과 시뮬레이티드 어닐링[5,6] 방식을 비교하였고, 마지막으로 Ⅴ장에서는 결론과 향후 연구 방향에 대하여 기술한다.

II. 배치 문제

1. 문제정의

배치는 칩 면적과 상호연결의 길이를 고려하여 n개의 모듈을 정해진 영역에 겹치지 않게 배치시키는 문제이다. 즉, 최적의 배치는 배치된 모듈들 사이의 상호연결의 길이와 칩 면적이 최소가 되도록 각각의 모듈들을 배치시키는 것으로서, 모듈들을 어떻게 배치하느냐에 따라 칩 면적과 모듈들 사이의 상호연결의 길이가 바뀔 수 있다[7].

배치 문제에서 최적의 해를 얻기 위하여 클러스터 성장, 시뮬레이티드 어닐링, ILP등의 알고리즘들이 기존에 연구되었다.

기존에 연구된 각 알고리즘을 살펴보면, 클러스터 성장은 계산량이 적어 속도가 빠른 반면 지역해에 빠지기 쉬운 단점이 있으며, 시뮬레이티드 어닐링은 최적해에 가까

운 해를 찾을 수 있으나, 온도에 따라 지역해를 벗어날 수 있는 가능성을 부여하므로 저온에서 해탐색 범위가 급격히 축소되는 경향을 보인다. 또한 ILP 알고리즘은 최적해를 반드시 찾을 수 있는 반면에 계산량이 많아 아주 작은 문제에만 적용 가능하고 큰 문제에서는 적용이 불가능한 단점이 있다.

그림 1은 배치의 간단한 예로서 (a),(b),(c)는 같은 칩 면적 안에 배치 되었지만, (d),(e)는 다른 결과를 얻는 것을 볼 수 있다.

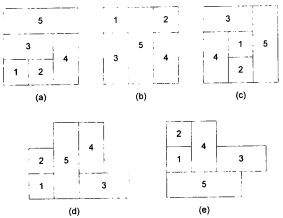


그림 1. 배치 예
Fig. 1. Example of placement.

III. Placement 확률 진화 알고리즘

확률 진화 알고리즘은 조합 최적화 문제를 위한 유용한 알고리즘으로, 확률적인 결정에 의하여 이동(move)을 받아들이는 알고리즘이다. 즉, 이득이 증가하는 이동은 받아들이고, 이득이 감소하는 이동은 0이 아닌 확률값에 의해 수용한다. 또한 지역해에 도달했을 때 확률값을 변경하여 지역해에서 벗어날 수 있는 기회를 제공하며, 이득이 증가하였을 경우에는 반복할 횟수를 증가시켜 정밀 탐색을 수행하는 알고리즘이다.

1. 해 표현 방법

본 논문에서 제안한 배치 확률 진화 알고리즘은 문제의 복잡도를 줄이기 위해 분할가능구조(slicing structure[8]), 모든 모듈의 크기 고정, 그리고 90° 회전 가능한 경우만을 가정하였다.

분할가능구조란 전체 사각형 구조의 모듈을 가로나 세로의 연속 분할만으로 부 모듈로 나눌 수 있는 구조를 말한다(그림 2).

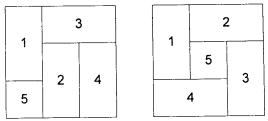


그림 2. (a) 분할가능 배치
(b) 분할불가능 배치
Fig. 2. (a) Slicing placement.
(b) Non-slicing placement.

분할가능 배치는 그림 3과 같이 분할가능트리(slicing tree)로 표현할 수 있고, 분할가능트리는 $E=e_1e_2\cdots e_{2n-1}$, $(e_i \in \{1,2,\cdots n,H,V\},\ 1 \le i \le 2n-1)$의 식으로 표현할 수 있다. 여기서 연산자 H, V는 다음과 같은 의미를 갖는다.

ijH : 모듈 j가 모듈 i의 위에 위치한다.
ijV : 모듈 i가 모듈 j의 왼쪽에 위치한다.

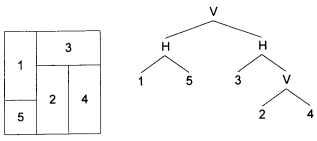


그림 3. (a) 분할가능 배치
(b) 분할가능트리와 분할가능트리의 표현
Fig. 3. (a) Slicing placement.
(b) Slicing tree and slicing tree representation.

하나의 배치는 그림 4와 같이 여러개의 분할가능트리로 표현할 수 있으므로 중복되는 해를 없애기 위하여 정규화된 분할가능트리의 표현을 사용하였다. 정규화된 표현이란 같은 연산자가 연속적으로 나오지 않는 식을 말하며, 정규화된 표현을 사용하여 하나의 배치는 하나의 표현만을 가질 수 있도록 하였다.

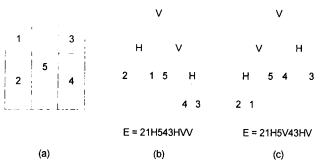


그림 4. 여러 가지 분할가능트리의 표현
(a) 배치 (b) 비정규화된 표현
(c) 정규화된 표현

Fig. 4. several slicing tree representation.
(a) Placement. (b) Not normalized expression.
(c) Normalized expression.

2. 이동 방법

본 논문에서 제안된 확률 진화 알고리즘의 PERTURB 함수에서 사용되는 이동 방법은 다음과 같다. 여기서 모듈 $1,\cdots n$을 피연산자, H,V를 연산자라고 가정한다.

M1 : 두 개의 이웃된 피연산자를 교환한다.
M2 : 임의 개수의 연속된 연산자를 반전시킨다. 즉, $\overline{V}=H, \overline{H}=V$이다.
M3 : 두 이웃된 피연산자와 연산자를 교환한다.
M4 : 피연산자를 90° 회전시킨다. 즉, 12V4H35VH에서 5는 5번 피연산자를 90° 회전시킨 것을 의미한다.

여기서 *M1*, *M2*, *M4* 이동은 항상 정규화된 표현을 얻는 반면, *M3* 이동은 비정규화된 표현을 얻을 수 있으므로 다음의 경우에 해당하는 *M3* 이동은 취소되어야 한다.

(1) 같은 연산자의 연속 표현 : 12*V*43*H*5*HV*에서 e_7과 e_8을 교환하였을 경우 12*V*43*HH*5*V*의 비정규화된 결과를 얻게 된다.
(2) Balloting property[8]의 위배 : 피연산자 e_i와 연산자 e_{i-1}을 *M3* 이동하였을 때 $2N_{i-1}\geq i$이면 피연산자의 수보다 연산자의 수가 많아져 비정규화된 표현이 된다(여기서 N_k는 정규화된 표현 $E=e_1e_2\cdots e_k(1\leq k\leq 2n-1)$에서의 연산자의 수로 정의한다). 즉, 12*V*4*H*35*VH*에서 e_7와 e_{7+1}를 *M3* 이동하면 $2N_{7+1}(=6) < 7$이지만, e_4와 e_{4+1}를 *M3* 이동하면 $2N_{4+1}(=4) = 4$이므로 balloting property를 위배하게 된다.

3. Placement 확률 진화 알고리즘

그림 5는 앞에서 제안한 이동방법을 사용한 placement 확률 진화 알고리즘을 나타낸다.

그림 5. Placement 확률 진화 알고리즘
Fig. 5. Placement stochastic evolution algorithm.

```
Begin
  BestE = CurE = E0;
  BestCost = CurCost = Cost(CurE);
  p = P0;
  ρ=0;
  Repeat
    PrevCost = CurCost;

    CurE = PERTURB(CurE, p);
    CurCost = Cost(CurE);
    UPDATE(p, PrevCost, CurCost);

    If (CurCost<BestCost) Then
      BestE = CurE;
      BestCost = CurCost;
      ρ = ρ -R;
    Else
      ρ = ρ +1;
    EndIf
  Until ρ>R
  Return (BestE);
End

Function PERTURB(CurE, p)
Begin
    ForEach (i∈E) Do
      SELECT_MOVE(E, i,M);

      Case M of
        M1 : E' = Swap(CurE, ei, ei+1);
        M2 : E' = Complement(CurE, Ci);
        M3 : If (ei-1 ≠ei+1) and (2Ni+1<i) Then
                E' = Swap(CurE, ei, ei+1);
             EndIf;
        M4 : E' = Rotate(CurE, i);
      EndCase

      Gain(m) = Cost(CurE) - Cost(E');
      If (Gain(i)>RANDINT(-p,0)) Then
        CurE = E';
      EndIf
    EndFor;

    Return(CurE);
End

Procedure UPDATE(p, PrevCost, CurCost)
Begin
    If (PrevCost=CurCost) Then
      p = p + pincr.
```

```
    Else
      p = p0;
    EndIf;
End

Procedure SELECT_MOVE(E, i,M)
Begin
    If (RANDINT(0,1)) Then
      M0 = RANDINT(0,1)+1;

      If (e_i=OPERATOR) Then
        M = M0+1;
      Else
        If (e_i-1=OPERATOR) Then
          M = M0+2;
        Else
          M = M0*3;
        EndIf;
      EndIf;
    Else
      M = -1;
    EndIf;
End
```

여기서 새로운 이동으로 얻은 이득값이 양수이면 이동을 받아들이고, 음수이면 p값의 확률에 따라 받아들이게 되며, 이전 비용과 현재 비용이 같으면 지역해에 도달한 것으로 간주하여 확률값 p를 증가시켜 지역해에서 벗어날 기회를 제공한다. 또한 새로운 이동으로 얻은 해의 비용이 *BestCost*보다 더 좋다면 반복횟수 ρ를 초기반복횟수 R만큼 감소시켜 전체 반복횟수를 증가시키고, 그렇지 않으면 ρ를 1 증가시킨다.

4. 비용함수

배치의 목적은 최소의 칩 면적에 최단의 상호연결의 길이를 갖는 해를 찾는 것이므로, 비용함수는 전체 모듈이 차지하는 영역과 각 모듈들의 상호연결의 길이의 합으로 나타낸다. 모듈이 차지하는 영역은 가로, 세로의 비를 고려하여 차이가 너무 커지지 않도록 할 수 있다(식 1).

$$Cost(F) = \alpha A(y/x) + \lambda W \qquad (식\ 1)$$

여기서 가로, 세로의 길이 중 짧은 부분을 x로 긴 부분을 y로 가정하였으며, α와 λ는 각각 칩 면적과 상호연결의 길이의 중요도를 나타내는 파라미터로서, α와 λ값을 적절히 조절함으로써 다른 해를 얻을 수 있다.

Ⅳ. 시뮬레이션

본 논문에서 제안한 placement 확률 진화 알고리즘을 시뮬레이션 하기 위하여 일정한 모듈의 개수를 갖는 블록 데이터를 생성하고, 동일 블록 데이터에 시뮬레이티드 어닐링과 확률 진화 알고리즘을 적용하여 그 시뮬레이션 결과를 분석하였다. 또한 배치 문제에 사용되는 벤치마크 회로인 MCNC 회로에 대하여 동일 알고리즘을 적용하여 결과를 비교하였다.

시뮬레이티드 어닐링은 조합 최적화 문제 해결에 적용되는 대표적 반복 휴리스틱 알고리즘으로 Timber-Wolf[9]등 실용 패키지뿐 아니라 알고리즘의 성능을 비교하는 벤치마크로 주로 사용된다.

본 시뮬레이션에서는 확률 진화 알고리즘의 초기 반복횟수 R=5000, 초기 확률값 P_0=2, Pincr=2를 사용하였고, 시뮬레이티드 어닐링의 한 온도에서의 알고리즘수행 횟수 M=100, 최대 수행 횟수 max_time=10000, 초기 온도 T=10, 냉각 파라미터 α=0.9, 그리고 β=1.0의 값을 사용하였다. 또한 비용 계산의 중요한 파라미터인 α=1, λ=0을 사용하여 칩 면적을 최소화하는 해를 구하도록 하였다.

그림 6, 7, 8은 생성된 블록 데이터에 시뮬레이티드 어닐링과 확률 진화 알고리즘 적용한 결과의 최적값, 최악값, 평균값을 가지고 각 알고리즘을 비교한 것을 보이고 있다. 자세한 수치 결과는 Appendix의 표 2에 보였다.

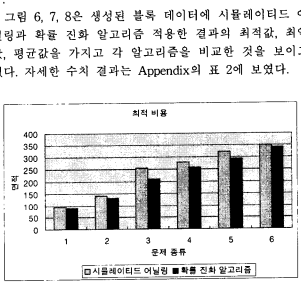


그림 6. 최적 비용 비교

Fig. 6. Comparison of the best cost.

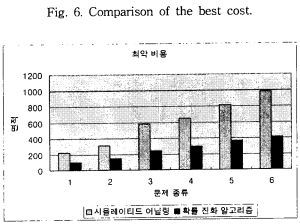


그림 7. 최악 비용 비교

Fig. 7. Comparison of the worst cost.

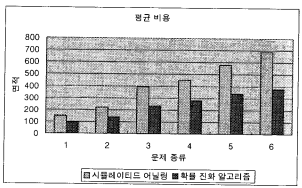


그림 8. 평균 비용 비교

Fig. 8. Comparison of the average cost.

또한, 그림 9는 배치 문제에 사용되는 벤치마크 회로인 MCNC 회로에 대하여 동일 알고리즘을 적용한 결과를 보이고 있다. 자세한 수치 결과는 Appendix의 표 3에 보였다.

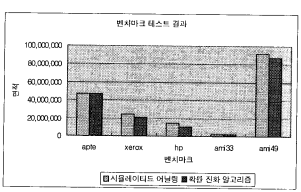


그림 9. 벤치마크 테스트 결과

Fig. 9. Benchmark test result

표 1은 MCNC 벤치마크 회로에 적용하여 얻은 개선도를 나타낸 것이다. 개선도를 얻기 위해 사용된 식은 다음과 같다.

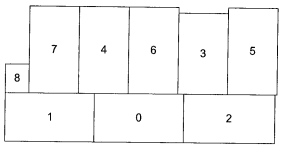
$$개선도(\%) = (SA-StocE) / SA * 100 \quad (식\ 2)$$

표 1. StocE의 개선도

Table 1. Improvement by Percentage of StocE

벤치 마크	모듈 개수	개선도
apte	9	0.45
xerox	10	14.93
hp	11	29.76
ami33	33	0.96
ami49	49	4.74

그림 10는 MCNC 벤치마크 회로중 "apte"의 확률 진화 알고리즘 적용 후의 배치 결과이다.



$E=10\sqrt{2}1\sqrt{87}\sqrt{46}1\sqrt{3}1\sqrt{5}VVH$ Cost = 47313280

그림 10 MCNC "apte" 회로의 배치 결과

Fig. 10. Placement result for MCNC "apte"

두 알고리즘의 시뮬레이션 결과를 분석해보면 placement 확률 진화 알고리즘이 시뮬레이티드 어닐링보다 더 좋은 결과를 얻는 것을 알 수 있다.

시뮬레이티드 어닐링은 초기에는 지역해에서 벗어날 수 있는 확률이 높으나, 냉각과정이 진행되면서 지역해에서 벗어날 수 있는 확률이 낮아져서 일정 횟수를 수행한 후에는 더 좋은 해를 찾을 가능성이 작아지게 된다. 반면에 placement 확률 진화 알고리즘은 지역해에 도달한 것을 감지했을 경우 이후 탐색의 수용 확률을 높여서 지역해에서 벗어날 수 있도록 유도하며, *BestCost*값이 갱신되는 경우에는 반복횟수를 증가시켜 더 많은 탐색을 할 수 있도록 한다.

또한, placement 확률 진화 알고리즘에서는 *M1, M2, M3, M4* 이동으로 정의된 PERTURB에 의해 시뮬레이티드 어닐링보다 효과적인 탐색이 가능하다.

V. 결론

본 논문에서는 실제 설계 과정중 배치 문제에 대하여 확률 진화 알고리즘을 제안하였으며, 제안한 방식을 시뮬레이티드 어닐링 방식과 비교, 분석하였다.

제안한 placement 확률 진화 알고리즘과 시뮬레이티드 어닐링 방식을 자동 생성한 회로와 벤치마크 회로에 적용하여 비교, 분석한 결과 제안한 placement 확률 진화 알고리즘이 시뮬레이티드 어닐링 방식보다 더 효과적으로 최적해에 근접하는 것을 알 수 있었다.

앞으로 각 모듈간의 상호 연결을 고려한 배치와 각 모듈의 모양이 사각형이 아닌 다각형인 경우에 대한 확률 진화 알고리즘의 연구가 필요하다고 생각된다.

Appendix

표 2. StocE와 SA의 테스트 결과
Table. 2. Test result for StocE, SA.

문제 종류	모듈 개수	시뮬레이티드 어닐링			확률 진화 알고리즘		
		최적	최악	평균	최적	최악	평균
1	10	96	224	151	90	104	96.67
2	12	140	315	223.16	132	148	141
3	14	256	580	394.57	210	246	235.32
4	16	280	648	452.79	258	301	282.02
5	18	324	816	578.17	294	364	337.99
6	20	350	986	689.6	342	416	381.49

표 3. SA와 StocE의 벤치마크 테스트 결과
Table 3. Benchmark test result for SA, StocE

벤치 마크	모듈 개수	면 적	
		시뮬레이티드 어닐링	확률 진화 알고리즘
apte	9	47528748	47313280
xerox	10	24911208	21190050
hp	11	14868168	10442880
ami33	33	2379636	2356704
ami49	49	91773864	87423644

접수일자 : 2001. 12. 31 수정완료 : 2002. 1. 24

참고문헌

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

[2] Naveed A. Sherwani, *Algorithms for VLSI Physical Design Automation. 3rd Edition*, Kluwer Academic Publishers, 2001.

[3] F.Y. Young, D.F. Wong, Hannah H. Yang, ON Extending Slicing Floorplan to Handle L/T-Shaped Modules and Abutment constraints, *IEEE Transaction on Computer-Aided Design*, pp. 800-807. June 2001.

[4] Y.Saab and V.Rao. Stochastic evolution: A fast effective heuristic for some generic layout problems. *17th ACM/IEEE Design Automation Conference*, pp. 26-31, 1990

[5] S. kirkpatrick, C. D. Gelatt and M. P. Vecchi. Optimization by Simulated Annealing, *Science*, vol. 220, no. 4598, pp.671-680, 1983.

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

[7] K.Shahookar and P.Mazumder. VLSI cell placement techniques. *ACM Computing Surveys*, 23(2):143-220, June 1991.

[8] D.F.Wong and C.L.Liu. A new algorithm for floorplanning design. *Proc. of the 23rd DAC*, pages 101-107, 1986.

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

[10] H.Murata, K.Fujiyoshi, S.Nakatake, and Y.Kajitani. VLSI module placement based on rectangle-packing by the sequence-pair. *IEEE Transactions on CAD*, 15:1518-1524, December 1996.

[11] G.Vijayan and R.Tsay. A new method for floorplanning using topological constraint reduction. *IEEE Transactions on CAD*, 10(12):1494-1501, December 1991.

송호정(Ho-Jeong Song)
準會員
1994년 배재대학교 물리학과 (이학학사)
1996년 청주대학교 전자공학과 (공학석사)
2001년 충북대학교 컴퓨터공학과 박사수료
관심분야 : VLSI 설계, High-level Synthesis

송기용(Gi-Youn Song)
正會員
1974~80년 서울대, 동대학원 (전자공학)
1995년 Univ. of Southwestern Louisiana 컴퓨터공학박사
1983년~현재 충북대학교 공과대학 컴퓨터공학과 재직중
관심분야 : 컴퓨터구조, VLSI 설계등