논문 2009-46SD-4-5

Sequence-Pair 기반의 플로어플랜을 위한 개선된 Simulated-Annealing 기법

(Improved Simulated-Annealing Technique for Sequence-Pair based Floorplan)

성 영 태*, 허 성 우*

(Young-Tae Sung and Sung-Woo Hur)

요 약

Sequence-Pair(SP) 모델은 모듈간의 위상 관계를 표현하는 방법으로써, 일반적으로 SP 모델에 기반한 플로어플래너(floorplanner)는 Simulated-Annealing(SA) 알고리즘을 통해 해를 탐색한다. 다양한 논문에서 SP와 SA 기반 배치 알고리즘의 성능 향상을 위해 SP의 평가 함수의 개선, SA의 스케줄링 기법 향상과 변형 함수의 개선 등을 모색하였다. 제안 기법은 기존의 SA 프레임웍을 수정한 2단계 SA 알고리즘으로써, 전 단계에선 SP로부터 구한 플로어플랜에 압축기법을 적용하여 모듈 사이에 존재하는 빈 공간을 가능한 최소화시켰다. 압축기법이 적용된 플로어플랜으로터 SP를 얻고, 이를 변환함으로써 해 공간을 탐색해 간다. 해가 기준 값에 수렴되었다고 판단되면 전 단계의 SA 기반 검색을 중단하고 압축기법을 사용하지 않은 기존의 SA 프레임웍을 이용하여 최적 해를 계속 탐색해 간다. MCNC 벤치마크 회로를 이용한 실험을 통해 제안 기법이 SA의 해 탐색 과정에 끼치는 효과를 보이며, 제안 기법을 통해 얻은 결과가 기존의 SA 기반 알고리즘으로 구한 결과보다 우수함을 보인다.

Abstract

Sequence-Pair(SP) model represents the topological relation between modules. In general, SP model based floorplanners search solutions using Simulated-Annealing(SA) algorithm. Several SA based floorplanning techniques using SP model have been published. To improve the performance of those techniques they tried to improve the speed for evaluation function for SP model, to find better scheduling methods and perturb functions for SA. In this paper we propose a two phase SA based algorithm. In the first phase, white space between modules is reduced by applying compaction technique to the floorplan obtained by an SP. From the compacted floorplan, the corresponding SP is determined. Solution space has been searched by changing the SP in the SA framework. When solutions converge to some threshold value, the first phase of the SA based search stops. Then using the typical SA based algorithm, ie, without using the compaction technique, the second phase of our algorithm continues to find optimal solutions. Experimental results with MCNC benchmark circuits show that how the proposed technique affects to the procedure for SA based floorplainning algorithm and that the results obtained by our technique is better than those obtained by existing SA-based algorithms.

Keywords : VLSI, Floorplan, Optimization, Sequence-pair, Simulated-annealing.

I. 서 론

VLSI 설계 과정에서 중요한 단계 중의 하나가 플로어플랜(floorplan) 설계 단계이다. 플로어플랜은 평면상에서 직사각형 모듈들을 중첩되지 않게 배치하면서 원하는 목적함수를 최적화 시키는 것을 목표로 한다. 목적함수로는 플로어플랜의 총 면적, 배선 길이 또는 면적과 배선 길이에 가중치를 부여한 합 등이다.

플로어플랜 문제를 위해 블록간의 위상 관계를 표현

* 정회원, 동아대학교 컴퓨터공학과
(Department of Computer Engineering, Dong-A University)
접수일자: 2008년9월26일, 수정완료일: 2009년3월12일

하는 다양한 모델이 제시 되었으며, 특히 비분할 구조(non-slicing structure)를 표현 할 수 있는 모델로써 SP(Sequence-Pair)[1], BSG(Bounded-Sliceline Grid) 구조[2], O-tree[3], CBL(Corner Block List)[3] 등이 있다.

SP 모델은 n개 모듈의 위상적 관계를 나타내기 위해 두 순열을 이용하며, 순열의 변형(perturbation)를 통해 얻을 수 있는 해 공간의 크기는 $O((n!)^2)$이고 한 쌍의 SP 순열로 부터 이에 대응하는 배치를 얻는데 $O(n^2)$ 복잡도를 가진다[1]. BSG 구조에서는 가상의 $n \times n$ 격자를 이용하며, 격자내의 모듈을 교환하는 방식으로 새로운 해를 구한다. 이때 해 공간의 크기는 $O(n!C(n^2, n))$이며 배치를 얻는데 걸리는 시간은 $O(n^2)$ 이다. O-tree 모델에서는 해 공간이 $O(n!2^{2n-2}/n^{1.5})$로써 다른 모델에 비해 상대적으로 해 공간의 크기가 적으며, 한 트리로부터 배치를 얻는 시간은 $O(n)$이다. CBL 모델을 사용하는 기법의 해 공간의 크기는 $O(n!2^{3n-3}/n^{1.5})$이며, 주어진 CBL로 부터 배치를 구하는데 필요한 시간은 $O(n)$이다.

대부분의 플로어플랜 기법들은 Simulated-Annealing(SA) 알고리즘에 기반 한 탐색(search) 구조를 가지며[1~3, 5~7], 수행 시간의 대부분을 현재 플로어플랜으로 부터 새로운 플로어플랜(neighbor solution)을 찾고 후보 플로어플랜을 평가(evaluation)하는데 사용한다. 따라서 좋은 평가 값을 갖는 플로어플랜을 빠른 시간에 찾기 위해서는 플로어플랜의 변형 연산을 효율적으로 계산하거나 평가 함수의 성능을 개선하는 것이 필요하다. Tang과 Wong[6~7]은 Murata[1]가 제안한 $O(n^2)$ 평가 함수를 효율적인 $O(n^2)$ 함수와 $O(nlogn)$ 함수로 개선시켰으며, Pang[7] 등은 O-tree 모델의 변형 함수를 개선하였다. 또한 Yan과 Chu[4] 등은 SA 해 탐색의 비효율성을 제거하기 위해 단일 분할 트리(single slicing-tree)를 이용한 효과적인 고정 영역 배치 기법을 소개 하였고, Chen과 Chang[5]은 B*-tree에 기반한 SA 개선 기법을 소개하였다.

본 논문에서는 SA알고리즘을 사용하는 SP 기반의 개선된 플로어플랜 기법을 제안한다. 제안기법은 기존[6]의 $O(n^2)$ 평가 함수와 변형 함수(perturb function)를 그대로 사용하는 대신 SA 프레임웍을 수정함으로써 해 탐색 능력을 개선한다. 본 논문의 구성은 다음과 같다. Ⅱ장에서 SP모델에 대한 일반적인 내용을 소개하고, 제안 기법을 설명한다. Ⅲ장에서는 제안 기법의 효과를 실험을 통해 보이며, 마지막으로 Ⅳ장에서 결론 및 향후 과제를 말한다.

Ⅱ. 본 론

1. Sequence-Pair 기반의 Simulated Annealing 기법

가. 플로어플랜 표현을 위한 SP 모델

Murata[1]는 n개 블록들의 상대적인 위치 관계를 n개 블록에 대한 순열(permutation)의 쌍으로 표현 할 수 있음을 보였다. 두 순열을 각각 $\Gamma+$, $\Gamma-$라 하며, ($\Gamma+$ = <.., x_i, .., x_j, ..>, $\Gamma-$ = <.., x_i, .., x_j, ..>) 일 때, x_i 는 x_j의 왼쪽에 위치하며 ($\Gamma+$ = <.., x_j, .., x_i, ..>, $\Gamma-$ = <.., x_i, .., x_j, ..>) 일 때, x_i는 x_j의 아래쪽에 위치한다. 이 때, x_i, x_j $\in$ *block*, $1 \le i, j \le n$ 이다. 따라서 두 순열은 블록들의 수평, 수직 관계를 보여주며 이 관계를 통해 수평 제약 그래프 $G_h(V, E)$와 수직 제약 그래프 $G_v(V, E)$를 구성 할 수 있다. $G_h(V, E)$의 생성 과정은 다음과 같다.

(1) $V = \{s_h\} \cup \{t_h\} \cup \{v_i \mid i = 1, .., n\}$,

(2) $E = \{(s_h, v_i) \mid i = 1, .., n\} \cup \{(v_i, t_h) \mid i = 1, .., n\}$
$\cup \{(v_i, v_j) \mid$ block i is left to block $j\}$,

(3) Vertex weight = width of block i for v_i, but 0 for s_h and t_h.

수직 제약 그래프도 수평 제약 그래프와 유사한 방법으로 구성한다. 두 그래프 $G_h(V, E)$, $G_v(V, E)$는 각 정점이 가중치를 가지며 방향성이고 또한 사이클이 없는 DAG(Directed Acyclic Graph)이다. 주어진 SP로부터 이에 대응하는 플로어플랜을 구하는 과정, 즉 각 모듈의 위치를 결정하는 과정을 패킹(packing)이라 한다. SP 모델에서 각 블록의 위치 좌표 (x, y)는 각각 $G_h(V, E)$, $G_v(V, E)$에서 최장 경로(longest path)를 구함으로써 결정할 수 있다. 이때 블록의 좌표 (x, y)는 해당 블록의 왼쪽 아래 모서리 위치이다. 주어진 SP로부터 $G_h(V, E)$, $G_v(V, E)$를 구성하는데 $\Theta(n^2)$ 시간을 소요하고 n과 m을 각 정점과 간선의 수라 할 때, 두 그래프로 부터 최장경로를 구하는데 $\Theta(n+m)$ 시간을 소요하므로 임의의 SP로 부터 이에 대응하는 배치로 구하는데 $\Theta(n^2)$ 시간이 필요하다.

나. SA 알고리즘을 이용한 해 탐색

기존의 SP 모델을 이용한 SA 탐색 알고리즘은 그림 1과 같다. SA 알고리즘은 입력으로 초기 플로어플랜

$SP_{initial}$과 환경 변수 T_0, α, β, N_0, N_{max}를 받는다. $SP_{initial}$는 랜덤(random) 알고리즘을 통해 생성 하며, T_0는 초기 온도, α는 쿨링(cooling) 비율, β는 반복 제어값, N_0는 초기 반복 횟수, N_{max}는 최대 반복 횟수를 의미한다. SA 알고리즘은 $SP_{initial}$로부터 해 탐색을 수행해 가면서 T_0가 일정 온도 이하가 되거나 총 반복 횟수가 N_{max}가 되었을 때 종료한다.

SA 알고리즘을 이용한 해 탐색 과정에서 최종 해의 질(quality)과 알고리즘의 수행 속도는 그림1의 *Metropolis* 함수에서 결정된다. *Metropolis* 함수의 3번째 줄 *Neighbor* 함수는 현재 해에서 다음 해를 구하는 변형(perturbing) 과정으로써 Murata[1]와 Tang 그리고 Wong[6~7]은 SP의 한 순열에서 임의의 한 쌍의 모듈을 교환하거나, 두 순열에서 각각 한 쌍의 모듈을 교환, 또는 임의의 모듈 회전을 고려하는 방식으로 변형 함수를 구현하였다. 4번째 줄의 *Cost*함수는 변형된 SP로부터 각 모듈의 위치 좌표를 구하여 전체 플로어플랜의 평가값을 구하는 패킹(packing) 과정이다. 수평, 수직 제약

```
Algorithm Simulated-Annealing
1. INPUT: SP_initial, T0, α, β, N0, Nmax
2. OUTPUT: A solution SP
3. Begin
4.     set N := N0, SP := SP_initial, T := T0, N_total :=0;
5.     while ((T>0.0)  and (N_total <= Nmax)) do {
6.         Call Metropolis(SP, T, N);
7.         N_total := N_total + N;
8.         T := T * α;
9.         N := N * β;
10.    }
11.    return SP;
12. End
Algorithm Metropolis(SP, T, N)
1. Begin
2.     while (N > 0) do {
3.         SP_new := Neighbor(SP);
4.         Δh := Cost(SP_new) - Cost(SP);
5.         if ((Δh < 0) or (random < e^(-Δh/T))) then
6.             SP := SP_new;
7.             N := N - 1;
8.     }
9. End
```

그림 1. 기존의 SP 모델을 이용한 SA 기반 알고리즘
Fig. 1. Existing SA based algorithm using SP model.

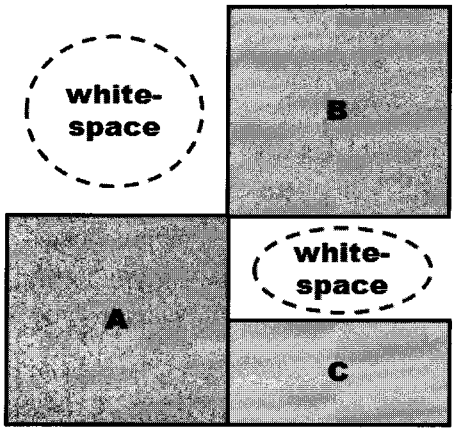

그림 2. 빈 공간을 가진 SP_{new}의 예
Fig. 2. An example of SP_{new} with white-space.

그래프를 이용한 Murata[1]의 경우 $O(n^2)$ 시간 알고리즘을 제안하였고, Tang[6~7]등은 효율적인 $O(n^2)$알고리즘과 더불어 이진 탐색 트리를 이용한 $O(nloglogn)$ 복잡도 알고리즘을 제안하였다.

기존의 *Metropolis* 알고리즘에서 사용한 변형 함수는 새로운 SP_{new}를 구함에 있어 패킹 상에 빈 공간(white-space)을 허용하는 단점을 가지고 있다. 예를 들어, SP_{new}의 $\Gamma+$와 $\Gamma-$를 각각 <*A*, *B*, *C*>, <*A*, *C*, *B*>라 하고 모듈의 너비와 높이를 각각 *A*(*w*:2, *h*:2), B(*w*:2, *h*:2), C(*w*:2, *h*:1)라 할 때, 그림 2와 같은 위상 적 관계를 가지며 이때 제거 가능한 빈 공간을 중간 해인 SP_{new}가 포함하게 된다. 따라서 SP_{new}가 가지는 빈 공간을 제거하고 이것에 대응하는 SP를 다시 찾음으로써 SA의 탐색 능력을 향상 시킬 수 있다.

2. 제안 기법

본 절에서는 기존의 SA 알고리즘의 초기 반복 시점에서 기존의 알고리즘보다 빠른 시점에 좋은 해를 구할 수 있는 기법을 제안한다. 제안 기법의 개요와 제안 기법의 두 가지 주요 알고리즘 *Compact*와 *Reverse*를 소개하고 SA의 초기 반복에서 두 기법의 효과에 대해 설명한다.

가. 개요

본 논문에서 제안한 기법은 효율적인 $O(n^2)$ 평가 함수[6]를 사용하고, 플로어플랜 배치 면적의 최소화(area minimization)를 목표 함수로 한다. 제안 기법의 알고리즘을 그림 3에 나타내었다.

제안 기법은 그림 3에서 보이는 바와 같이 기존 알고

```
Algorithm Metropolis_Proposed(SP, T, N)
1. Begin
2.     while (N > 0) do {
3.         SP_new := Neighbor(SP);
4.         if (CPR < Area_total/Cost(SP_new)) {
5.             Comp_SP_new := Compact(Packing_SP_new);
6.             SP_new := Reverse(Comp_SP_new);
7.         }
8.         Δh := Cost(SP_new) - Cost(SP);
9.         if ((Δh < 0) or (random < e^(-Δh/T))) then
10.            SP := SP_new;
11.            N := N - 1;
12.    }
13. End
```

그림 3. 2 단계 SA 기반 탐색을 위한 수정된 *Metropolis* 알고리즘

Fig. 3. Modified *Metropolis* algorithm for 2 phase SA search mechanisms.

리즘(그림 1의 Metropolis)에 조건부 루틴(4번째 줄~7번째 줄)을 추가하였다. 조건부 루틴을 수행하기 위한 조건은 $CPR \leq Area_{total}/Cost(SP_{new})$ 이며, 이때 *CPR*(Completion Ratio)은 블록의 총 면적에 대한 현재 해의 평가 값 비율이다. 예를 들어, 블록의 총 면적의 합이 120일 때, Neighbor 함수로 생성된 SP_{new} 의 평가 값이 125 라면 *CPR*=0.96 이다. 이것은 현재 해의 배치 완성도가 96%임을 의미한다. 따라서 수정된 *Metropolis_proposed* 알고리즘은 SA 해 탐색 과정 중 해의 배치 완성도가 *CPR* 값 이하일 때 추가된 조건부 루틴을 수행하고 그 이상일 때 기존의 *Metropolis* 알고리즘과 동일한 방법으로 동작한다.

조건부 루틴은 *Compact*와 *Reverse* 알고리즘으로 구성된다. *Compact* 알고리즘은 *Neighbor* 함수로 생성된 SP_{new} 에 대응하는 플로어플랜의 모든 블록이 각각 왼쪽과 아래쪽 방향으로 최소한 하나 이상의 다른 블록 또는 왼쪽 아래 영역 경계선과 인접하도록 배치의 위상을 변형하는 과정이다. *Compact* 과정에서 생성된 새로운 배치를 $Comp\_SP_{new}$ 라 할 때, *Reverse* 알고리즘은 $Compact_SP_{new}$로 부터 이에 대응하는 SP를 생성하는 과정이다. *Compact*와 *Reverse* 알고리즘은 다음과 같은 성질을 가진다. $Cost(Rev\_SP_{new}) = Cost(Comp\_SP_{new}) \leq Cost(SP_{new})$. $Comp_SP_{new}$는 SP_{new}가 가진 white-space를 제거 한 형태로써 $Cost(Comp_SP_{new}) \leq Cost(SP_{new})$은 명백하다. *Compact* 과정의 예를 그림 4에 보인다.

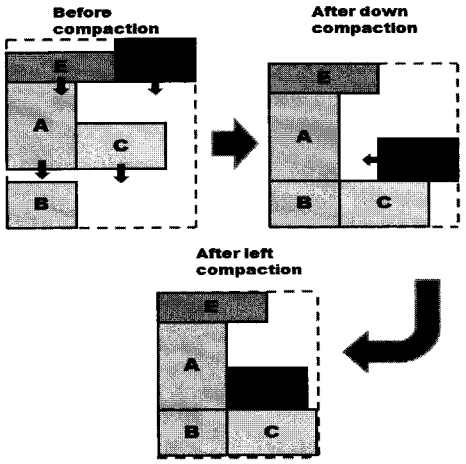

그림 4. 압축과정을 보여 주는 예

Fig. 4. An example to show compaction procedure

그림4에서 왼쪽 배치의 블록 A, C, D, E(A블록의 이동결과 아래쪽으로 이동가능)는 아래쪽 방향으로 인접한 블록을 가지지 않는다. 따라서 아래쪽 방향으로 압축이 가능하며 오른쪽 그림이 아래쪽 방향 압축 결과이다. 아래 그림은 마지막으로 블록 D의 왼쪽 압축을 수행한 결과이다.

추가된 조건부 루틴의 목표는 가능한 빠른 시간에 *CPR*값에 대응하는 SP_{new} 를 찾기 위함이다. 이것은 SA 알고리즘의 특성상 많은 해를 하나씩 탐색해야 하는 단점을 극복하기 위한 방안이며, 초기 반복 시점에서 좋은 해를 빠르게 찾음으로써 제한된 SA 환경에서 더 좋은 해를 구할 수 있는 방법이다.

나. Compact 알고리즘

Compact 알고리즘의 개요를 그림 5에 기술하였다.

그림 5에서 함수 *LeftFirstDownSecond*는 주어진 배치의 각 블록에 대해 왼쪽 방향으로 최소한 하나의 인

```
Algorithm Compact(Packing)
1. Begin
2.     Packing_comp1 := LeftFirstDownSecond(Packing);
3.     Packing_comp2 := DownFirstLeftSecond(Packing);
4.     if(Cost(Packing_comp1) < Cost(Packing_comp2))
5.         return Packing_comp1;
6.     else
7.         return Packing_comp2;
8. End
```

그림 5. *Compact* 알고리즘

Fig. 5. Algorithm *Compact*

접한 블록을 가지도록 압축하고 아래쪽으로 다시 압축하며, 함수 *DownFirstLeftSecond*는 아래쪽 방향으로 먼저 압축한 후 왼쪽 압축을 수행한다. 이 두 배치 중 평가 값이 더 좋은 배치를 반환한다. 압축 알고리즘은 Guo[3]가 소개한 윤곽선(contour-map) 기법을 이용하여 구현하였으며 *O(nlogn)* 시간에 수행된다.

다. Reverse 알고리즘

함수 *Reverse*는 주어진 배치로 부터 이에 대응하는 SP를 찾는 알고리즘이다. *Reverse* 알고리즘의 특징은 크게 두 가지로 요약된다.

(1) $Cost(Rev\_SP_{new}) = Cost(Comp\_SP_{new})$

(2) $SP_{prev} \neq Rev\_SP_{new}$, 이때, SP_{prev}는 압축 이전의 *Neighbor* 함수의 결과.

특징(1)은 $Comp_SP_{new}$의 평가 값과 Rev_SP_{new}의 평가 값이 같아야 함을 말하는데 $Cost(Rev\_SP_{new}) > Cost(Comp\_SP_{new})$ 일 경우, 이 후의 SA 탐색 과정에서 수락(accept)되는 해가 없을 수 있기 때문에 반드시 만족해야 한다. 본 논문에서는 Murata[1]가 제안한 Gridding 알고리즘을 사용하여 특징(1)을 보장한다. 특징(2)는 Rev_SP_{new}가 SP_{prev} 보다 더 나은 평가 값을 갖는 변형된 형태의 SP임을 말한다. 이것은 *Neighbor* 함수와 더불어 *Reverse* 함수 또한 하나의 변형(perturbation) 함수로 볼 수 있음을 말한다.

그림 6에서 SP의 Γ_+를 구하는 과정을 예로 보이고, *Reverse* 알고리즘을 그림 7에 기술하였다.

Γ_+와 Γ_-는 각각 Murata[1]가 제안한 Gridding 알고리즘을 사용한다. 그림 6에서 점선은 LociLine을 의미

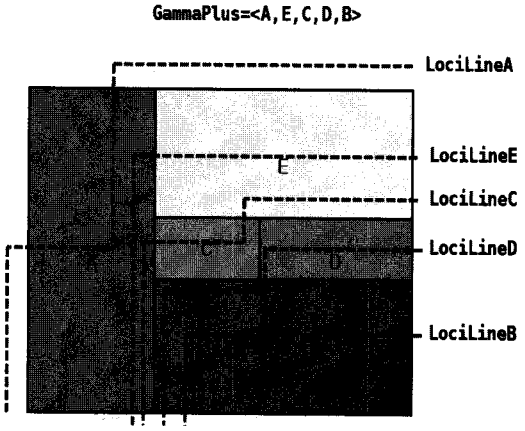

그림 6. LociLine을 이용하여 Γ_+를 구하는 예
Fig. 6. An example to determine Γ_+ using LociLine.

```
Algorithm Reverse(Packing)
1. INPUT: Packing
2. OUTPUT: SP (consists of Γ+ and Γ-)
3. Begin
4.   //find Γ+ from the given packing
5.   for each bi ∈ Packing: 0 ≤ i < n {
7.     Γ+_LociAreai.area := Eval_Γ+_LociLine_Area(bi);
8.     Γ+_LociAreai.id := index of bi;
9.   }
8.   sort array Γ+_LociArea by decreasing order
9.   for each Γ+j: 0 ≤ j < n {
10.    Γ+j := Γ+_LociAreaj.id;
11.  }
12.    /* Γ- can be found from similar fashion
            of the Γ+ routine */
13. return SP;
14. End
```

그림 7. *Reverse* 알고리즘
Fig. 7. Algorithm *Reverse*.

하고 Gridding 기법을 통해 각 블록의 LociLine을 구한 후 이 선이 포함하는 면적의 크기 순서대로 블록을 정렬시킨다. 그림6에서 블록 A의 LociLine 면적이 가장 크므로 Γ_+에서 가장 처음에 위치하고 블록 B의 면적이 가장 작으므로 맨 마지막에 위치한다. Γ_-를 구하는 방법도 Γ_+를 구하는 방법과 유사하다.

그림 7에서 7째 줄의 *Eval_Γ+_LociLine_Area* 함수는 각 블록의 중심 좌표에서부터 왼쪽 아래 이동(LeftDownMove)와 오른쪽 위 이동(RrightUpMove를 반복해 가면서 LociLine이 형성하여 면적을 누적 합산시키는 역할을 한다. 각 블록에 대한 *Eval_Γ+_LociLine_Area*함수는 $O(n^2)$ 복잡도를 가지며 *Reverse* 함수는 $O(n^3)$ 복잡도를 가진다.

III. 실험 및 결과

제안 기법의 실험은 WindowsXP/AMD 1.8G 시스템 상에서 수행 하였으며, C 언어로 구현하고 minw-gcc-3.4.2로 컴파일 하였다. 실험에 사용한 회로는 표준 벤치마크 회로인 apte, xerox, hp, ami33, ami49를 사용 하였으며, 각각 회로의 수가 9, 10, 11, 33, 49개인 hard-block이다.

각 회로에 대한 실험에서 같은 회로의 경우 동일한

SA 스케줄링 환경(동일한 반복수)에서 수행 하였으며, 회로의 특성마다 스케줄링 환경을 달리하였다. 예를 들어, hp 회로의 경우 높은 T_0 온도에서 균등한 a 값 비율에 대해 동일한 β 값의 반복 회수를 주었고, ami49의 경우 낮은 T_0 온도에서 서서히 cooling을 진행하며 높은 비율로 반복회수를 늘렸다.

1. 제안 기법의 효과 비교

hp 회로의 경우 제안 기법으로 수행 한 결과 제안 기법의 평가 값이 9.34인 반복 시점에서 기존 기법은 10.92값을 가지고 있다.

apte 회로의 경우 그림 9에서 보이는 결과 값 47.44를 제안 기법으로 찾은 반면 기존 기법은 동일 반복 시점에 48.47 값을 가지고 있다.

그림 10의 xerox 회로의 경우 예외적으로 기존 기법에서 21.60 결과 값을 찾는 동안 제안 기법에서는 21.97 결과 값을 보여주고 있다. 이것은 제안기법에서 효과적으로 CPR값에 도달하도록 유도하는 알고리즘이 아닌 단순히 SP의 순서쌍 교환을 통해 이웃 해를 구하기 때문으로 분석된다.

그림 11의 ami33 회로의 경우 같이 제안기법이 특정 반복 시점에 결과 값 1.23을 찾는 동안 기존 기법은 결과 값 1.25 상태에 있다.

그림 12의 회로 ami49는 벤치마크 회로 중 가장 크

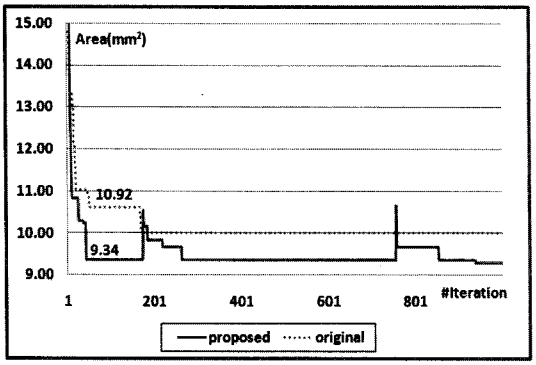

그림 8. hp 회로에 대한 기존기법과 제안기법의 비교
Fig. 8. Comparison of existing SA and proposed SA for hp circuit.

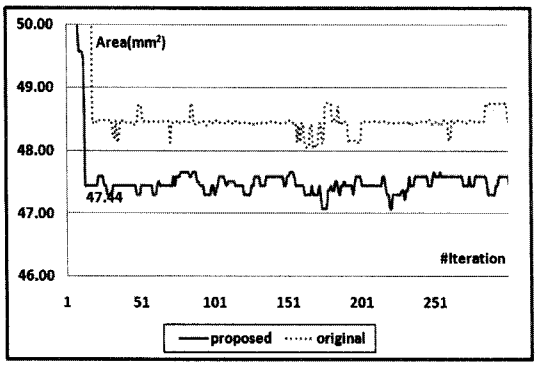

그림 9. apte 회로에 대한 기존기법과 제안기법의 비교
Fig. 9. Comparison of existing SA and proposed SA for apte circuit.

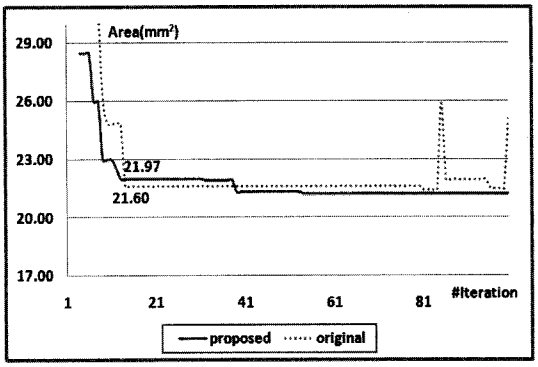

그림 10. xerox 회로에 대한 기존기법과 제안기법의 비교
Fig. 10. Comparison of existing SA and proposed SA for xerox circuit.

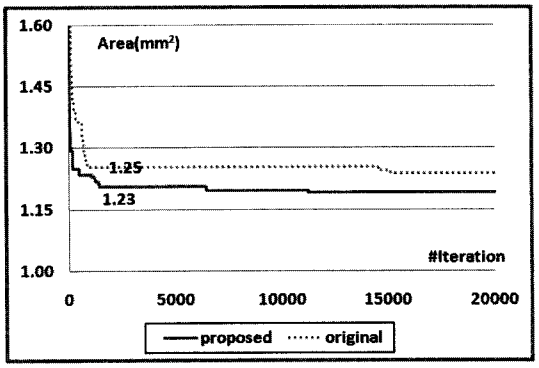

그림 11. ami33 회로에 대한 기존기법과 제안기법의 비교
Fig. 11. Comparison of existing SA and proposed SA for ami33 circuit.

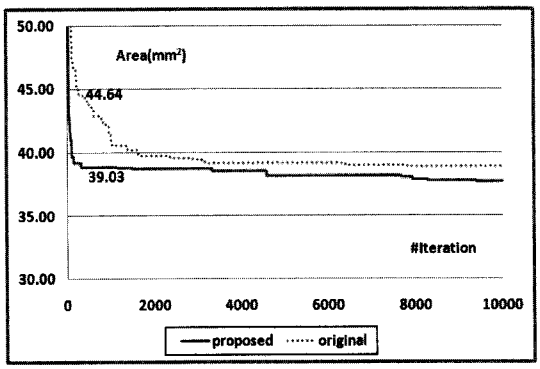

그림 12. ami49 회로에 대한 기존기법과 제안기법의 비교
Fig. 12. Comparison of existing SA and proposed SA for ami49 circuit.

표 1. 기존 기법과 제안 기법의 효과 요약
Table 1. Summary to show the effect of existing SA and proposed SA technique.

Circuit	T(Iteration)	Original (mm^2)	Proposed (mm^2)
apte	23	48.47	47.44
xerox	19	21.60	21.97
hp	80	10.92	9.34
ami33	230	1.25	1.23
ami49	700	44.64	39.03

기가 큰 회로이다. SA 초기 반복 시점에서 제안 기법의 경우 39.03의 해를 찾는데 반해 기존 기법의 경우 동일 반복 시점에서 44.64의 해를 찾고 있음을 알 수 있다. 표 1에 제안 기법의 효과를 요약하였다.

2. 결과 비교

표 2, 3, 4에 각각 최종 결과에 대한 worst, best, average 결과 값과 수행 시간을 요약 하였다. 실험에 사용된 *CPR* 값은 모두 0.90으로써 초기 반복 중 90%의 배치 완성도를 가지기 전 까지 모두 *Compact*와 *Reverse*를 사용하였다. 그리고 표에 기술된 시간(sec)은 해당 결과 값을 찾는데 까지 걸린 총 시간이다.

Worst-case 결과 비교에서 xerox와 hp 그리고 ami33 회로의 경우 기존기법의 해 보다 좋은 결과를 더 빠른 시간에 찾음을 알 수 있다.

Best-case 결과 비교에서 최종 결과 값의 경우 제안 기법에서 우수함을 보였고 수행시간 측면에서 제안기법이 더 많은 시간을 소요했음을 알 수 있다. 이것은 제안 기법이 기존 기법에 비해 SA 알고리즘 내부에 두 가지 알고리즘을 더 사용하기 때문이다. 그러나 동일한 SA 환경(같은 수의 반복)에서 제안기법이 더 좋은 최종 결과를 보임을 알 수 있다.

표 2. Worst-case 결과 비교 (CPR=0.90)
Table 2. Comparison of worst-case results (CPR=0.90).

Circuit	Original		Proposed	
	area (mm^2)	time (sec)	area (mm^2)	time (sec)
apte	48.05	1	48.05	1
xerox	20.42	8	20.47	3
hp	9.52	5	9.27	1
ami33	1.24	53	1.21	8
ami49	38.55	127	37.57	209

표 3. Best-case 결과 비교 (CPR=0.90)
Table 3. Comparison of best-case results (CPR=0.90).

Circuit	Original		Proposed	
	area (mm^2)	time (sec)	area (mm^2)	time (sec)
apte	47.07	1	47.07	1
xerox	19.86	17	19.83	14
hp	9.14	5	9.11	10
ami33	1.21	13	1.19	16
ami49	36.81	76	36.75	148

표 4. Average-case 결과 비교 (CPR=0.90)
Table 4. Comparison of avg.-case results (CPR=0.90).

Circuit	Original		Proposed	
	area (mm^2)	time (sec)	area (mm^2)	time (sec)
apte	47.68	1	47.32	1
xerox	20.23	11	20.12	10
hp	9.25	9	9.20	7
ami33	1.29	26	1.20	49
ami49	37.46	74	36.99	114

표 4에서 보이는 바와 같이 평균적으로 제안 기법이 기존 기법과 비교하여 동일한 SA 환경 하에서 최종 결

표 5. 제안 기법과 Fast-SP, TCG, En.O-tree, ECBL, B*-tree간 시간과 면적 비교.
Table 5. Comparison of runtime and area by Fast-SP, TCG, En.O-tree, ECBL, B*-tree and proposed technique.

	Fast-SP[3]		TCG[3]		En. O-tree[8]		ECBL[3]		B*-tree[3]		Proposed	
	time (sec)	area (mm^2)	time (sec)	area (mm^2)	time (sec)	area (mm^2)	time (sec)	area (mm^2)	time (sec)	area (mm^2)	time (sec)	area (mm^2)
apte	1	46.92	1	46.92	11	46.92	3	N/A	7	46.92	1	47.07
xerox	14	19.80	18	19.83	38	20.21	3	19.91	25	19.83	14	19.83
hp	6	8.94	20	8.94	19	9.16	11	8.91	55	8.94	10	9.11
ami33	20	1.20	306	1.20	119	1.24	73	1.19	3417	1.27	16	1.19
ami49	31	36.50	434	36.77	406	37.73	117	36.70	4752	36.80	148	36.75

과가 우수함을 보여준다. ami33과 ami49 회로의 경우 추가된 함수로 인해 더 많은 수행 시간을 필요로 하는데, 이것은 제안 기법에서 사용하는 *Compact*와 *Reverse* 알고리즘을 개선한다면 수행 시간에서도 개선을 보일 것으로 예상한다.

3. 기존의 다른 모델과의 결과 비교

표 5에 기존에 보고된 다른 모델과 제안 기법의 결과를 비교하였다. *O(nloglogn)* 평가 함수를 사용하는 Fast-SP 기법[7]과 효율적인 $O(n^2)$ 평가 함수를 사용한 본 논문의 결과를 비교해 볼 때 ami49 회로의 경우 수행 시간에서 차이를 보이나 비교할 만한 최종 결과를 보임을 알 수 있다.

Ⅳ. 결 론

본 논문에서는 Sequence-Pair(SP) 모델을 기반으로 하는 Simulated-Annealing(SA) 프레임웍에서 동일한 SA 환경이 주어졌을 때, 빠른 반복 시점에서 더 좋은 배치를 찾도록 SA 탐색 과정으로 변형하였다. 기존의 SA 알고리즘보다 더 많은 처리 루틴을 사용하며, 비효율적인 *Reverse* 알고리즘을 사용한다 하더라도 결과를 비교해 볼 때, 비교할 만한 시간에 더 좋은 해를 구할 수 있음을 알 수 있다.

향후 과제로 (1) 제안기법에서 사용하는 *Reverse* 알고리즘의 계산 속도를 개선하는 것과 (2) Kahng[9]이 제안한 것과 같이 현실적인 배치 목적에 맞는 목적 함수로 최적화를 수행하는 것이다. (2)의 경우, 면적의 최소화와 더불어 배선 길이의 최소화[3], 그리고 고정된 영역(fixed-layout) 상에서 배치[10~11], 제약 조건을 가진 배치[12] 등을 들 수 있다.

참 고 문 헌

[1] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangular-packing by the sequence-pair", *IEEE Trans. on CAD of Computer-Aided Design of Integrated Circuits and Systems*, vol.15:12, pp. 1518-1524, 1996.

[2] S. Nakatake, K. Fujiyoshi, H. Murata and Y. Kajitani, "Module Placement on BSG-Structure and IC Layout Applications," *Proc. of ICCAD*, 1996.

[3] H. H.Chan, S. N.Adya, I. L.Markov, "Are Floorplan Representations Important in Digital Design?," *Proc. of the 2005 ISPD*, pp.129-136, 2005.

[4] J .Z.Yan, C. Chu, "DeFer: Deferred Decision Making Enabled Fixed-Outline Floorplanner," *Proc. of the 45th DAC*, pp.161-166, 2008.

[5] T. C.Chen, Y. W.Chang, "Modern Floorplanning Based on Fast Simulated Annealing," *Proc. of the 2005 ISPD*, pp.104-112, 2005.

[6] X. Tang, R. Tian, and D. F.Wong, "Fast evaluation of sequence pair in block placement by longest common subsequence computation", *DATE-2000*, pp. 106-111, 2000.

[7] X. Tang and D. F.Wong, "FAST-SP: a fast algorithm for block placement based on sequence pair", *Proc. of the 2001 conference on ASP-DAC*, pp. 521-526, 2001.

[8] Y. Pang, C. K. Cheng, and T. Yoshimura, "An Enhanced Perturbing Algorithm for Floorplan Design Using the O-tree Representation", *ISPD 2000*, pp. 168-173, 2000.

[9] A. B.Kahng, "Classical floorplanning harmful?," *Proc. of the 2000 ISPD*, p.207-213, 2000.

[10] S. Chen, T. Yosihmura, "A Stable Fixed-Outline Floorplanning Method," *Proc. of the 2007 ISPD*, pp.119-126, 2007.

[11] Y. Zhan, Y. Feng and S. Sapatnekar, "A Fixed-die Floorplanning Algorithm using an Analytical Approach," *Proc. of the 2006 ASP-DAC*, pp.771-776, 2006.

[12] Y. Feng, D. P.Mehta and H. Yang, "Constrained Modern Floorplanning," *Proc. of the 2003 ISPD*, pp.128-135, 2003.

저 자 소 개

성 영 태(정회원)

2002년 동아대학교 컴퓨터공학과 학사 졸업.

2004년 동아대학교 컴퓨터공학과 석사 졸업.

2006년 동아대학교 컴퓨터공학과 박사 수료.

<주관심분야 : 알고리즘, 반도체, VLSI CAD>

허 성 우(정회원)-교신저자

1981년 경북대학교 전자공학과 학사 졸업.

1983년 한국과학기술원(KAIST) 전산학과 석사 졸업.

2000년 UIC Dept. of EECS 박사 졸업.

2001년~2006 Intel사 Physical Design 분야 기술자문위원.

1986년~현재 동아대학교 컴퓨터공학과 교수.

<주관심분야 : 알고리즘, VLSI CAD, Combinatorial Optimization>