FPGA 기반 설계의 온도 센서 최적 배치 알고리즘

현철환[1] 남형욱[1] 김용주[2] 김태환[3]
[1]한국과학영재학교
[2]KAIST, 과학영재교육연구원
[3]서울대학교, 전기컴퓨터공학부
[1]big0725@hanmail.net, [1]hwnam831@gmail.com
[2]kyjgifted@kaist.ac.kr, [3]tkim@ssl.snu.ac.kr

Thermal Sensor Allocation and Placement Algorithm on FPGA Based Design

Cheolhwan Hyeon[1] Hyoungwook Nam[1] Yongju Kim[2] Taewhan Kim[3]
[1]Korea Science Academy
[2]Institute for Gifted Students, KAIST
[3]School of Electrical Engineering and Computer Science, Seoul National University

요 약

본 논문은 FPGA 기반 설계에서 주변보다 급격한 온도 변화를 보이는 hotspot들을 탐지하기 위한 열 감지 센서 수를 정하고, 센서의 놓여야 할 배치 장소를 결정하는 알고리즘을 제안한다. 열 감지 센서로는 동적으로 설계가 가능한 ring oscillator 센서 기술을 사용한다는 가정 하에, 센서의 사용 개수를 최소화함과 동시에 최적의 센서 배치 위치 찾는다. 기존의 연구의 단점은 센서가 감지하는 영역 범위를 적당한 크기의 정사각형으로 간주하였기에, 실제 원형의 관측 범위를 보이는 센서 감지 영역의 현실을 올바로 반영하지 못하였으며, 또한 잘 알려진 회로 분할(partition) 기법에 의존한 휴리스틱으로 최적의 결과를 보장하지는 못하였다. 이와는 달리 본 연구에서는 센서의 관측 범위를 원형으로 할 수도 있게 함과 동시에 최적의 해를 보장하는 센서 할당 및 배치 알고리즘을 제안한다. 구체적으로 본 제안 알고리즘에서는 소위 "Candidate Coloring 기법"을 통해 센서가 놓여야 할 모든 후보 영역을 표시하며, "Candidate Filtering 기법"을 통해 불필요한 후보 영역들을 완전히 삭제하여 탐색 공간을 줄이게 되며 (해의 최적 해는 항상 유지 되도록 하면서), 마지막으로 Branch-and-Bound 알고리즘을 적용해 최적의 센서 할당 및 배치 결과를 찾아내었다.

1. 서 론

반도체 칩을 만드는 공정 기술이 발전하면서 더 많은 양의 일을 더 작은 공간 안에서 처리하게 되었고 좁은 공간에 많은 전류가 흐르게 됨에 따라 전력 밀도(power density)가 자연스레 증가한다. 전력 밀도의 증가는 필연적으로 국소적인 온도 증가로 귀결되고 이는 해당하는 논리 회로 영역에서의 주파수 저하(지연시간 증가로 인해) 반도체 구조 파괴 등에 직접적으로 연결되면서 시스템 안전성 유지에 심각한 위협이 되고 있다. 따라서 지엽적인 급격한 온도 상승은 회로 작동의 성능을 방해하는 가장 큰 요인으로 되며 이를 해결하기 위한 연구들도 최근 많이 진행되고 있다 [참고문헌 5,6,7]

방열판과 팬을 이용한 물리적인 냉각 방법은 오래 전부터 쓰이던 방법으로, 발열 문제를 해결하는 가장 기본적인 방법으로 쓰이고 있다. 하지만, 이는 기판 위의 국부적인 평균 온도를 낮추는 방법으로, 조금씩 한계를 보이고 있으며, 전체적인 온도뿐만 아니라 국소적인 온도 분포를 분산시키기 위해 작업 스케줄 관리나 동작 주파수 관리 등의 시스템 운영에서의 접근도 온도 문제 해결의 중요한 열쇠가 되고 있다.

작업(task) 스케줄링이나 주파수 조절 등의 동적 온도 관리를 위해서는, 현재의 논리 회로 상의 온도를 측정해서 실시간으로 받아오는 정보가 필수적이다. 이를 위해 FPGA 상에 ring oscillator sensor [참고문헌 3,4]를 배치하여 온도를 측정하는 것이 일반적이다. Ring oscillator 센서는 부가적인 기능 없이 자체만으로도CLB array상에서 최소 4칸을 사용하게 되며, 측정을 위해 각 센서마다 일정량의 오버헤드(overhead)를 필요로 하게 된다. 이에 따라 센서의 개수가 많아질수록 사용하는 자원의 양 (즉, 논리설계 양)이 늘어나며, 센서 자체가 사용하는 자원 때문에 기판의 열 분포에 영향을 끼치기도 한다. 온도를 측정하는 기기의 특성 상 최대한 적은 센서를 사용하여 측정 시 생기는 오차와 자원 낭비 문제를 해결할 필요가 있다, 즉, 센서의 수를 최대한 줄이는 것이 합리적이다.

필요한 센서의 개수를 줄이기 위해서 회로 상의 모든 영역을 조사하는 센서 배치보다는, 다른 논리 구역보다 높은 온도 변화율을 기록한 hotspot들을 정의하여, 이들의 온도를 측정하고, 이들을 중심으로 온도를 관리하는 방법을 택하는 방법이 있다. 이 hotspot들을 모두 조사하면서 최소의 센서를 사용하게 하는 센서 배치 알고리즘

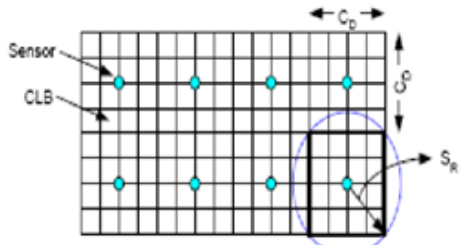

그림 1.1 사각형 센서의 단점

은 자원 낭비 방지와 측정 오차 문제 해결에서 중요한 역할을 담당하게 된다.

기존의 센서 배치 알고리즘들은 소위 Recursive Bisection algorithm [참고문헌 1,2]을 통해 FPGA 상의 hotspot을 포함하는 사각형의 부분들로 나눈 뒤, 모든 사각형이 센서가 측정할 수 있는 물리적 범위보다 작을 때 그 사각형의 중심에 온도 측정 센서를 배치하였다. 하지만, 이 센서 배치 알고리즘에는 다음과 같은 주요 단점이 존재 한다 :

(i) 정해진 범위 내의 사각형을 해로써 찾는 Bisection 알고리즘의 특징 상, 원형 센서에 적합하지 않은 센서 할당 알고리즘이다.

(ii) Bisection 알고리즘은 최적의 해를 찾는다는 보장이 없다. 따라서 많은 연구에서 후-보정을 통해 이 문제를 해결하려 하였으나, 아직 그 알고리즘들이 최적의 해를 보장한다는 사실을 증명하지 못하고 있다.

본 연구에서는, 원형의 측정 범위를 가지고 있는 센서의 가장 효율적인 배치를 찾는 알고리즘을 개발하였다이 알고리즘은 크게 3 단계로 나뉜다. (1) 불러온 FPGA 수행에 따른 온도 분포 데이터를 이용하여hotspot을 커버하기 위한 센서의 배치 위치 후보를 모두 파악하며(2) 후보 영역들의 포함 관계를 조사해 필요 없는 후보 영역들을 완전히 제거하며, (3) Branch-and-bound 알고리즘을 적용해 최적의 센서 할당 및 배치 해를 구하는 것이다. 연구의 핵심은 (3)의 알고리즘이 큰 크기의 문제에서도 적용이 가능하도록 하기 위한 문제 해를 찾기 위한 탐색 공간을 (1)과 (2) 과정을 통해 최대한 줄이는 것에 기술에 있다.

2. 연구 관련 배경 지식

2.1 정의와 가정

Hotspot은 작업 실행 시 문턱(threshold) 온도 이상의 온도를 가지는 논리 설계 영역으로 정의한다. 동일 작업 실행 시 hotspot의 출현 위치는 바뀌지 않는다고 가정한다. (즉, 임베디드 설계 처럼 수행되는application 이 이미 정해져 있는 응용을 가정하고 있다) 기판은 96X64 에서 192X128의 크기를 가지는 불연속CLB array로 정의한다. 센서는 원형의 측정 범위를 가지는 ring oscillator 온도 센서로 정의하며, 반경 '*r*' 이내의 모든 hotspot들을 조사할 수 있다.

2.2 Ring Oscillator Sensor

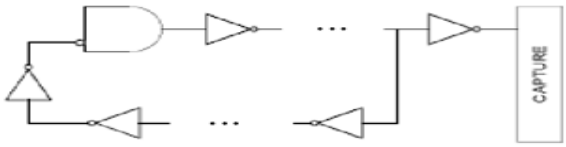

그림 2.1.1 Ring oscillator sensor

Ring oscillator sensor는 위의 그림과 같은 홀수개의 Inverter가 ring을 구성하여, 온도를 측정하는 센서이다. 온도가 높아지면 switching speed가 감소하는 반도체 inverter의 특성을 이용, 동작 클럭(clock)을 측정하여 실험적으로 온도를 측정한다. 대개의 경우 사각형이나, 원형 범위 내의 회로의 온도를 측정한다. 본 논문에서는 원 내부의 모든 영역의 온도를 조사한다고 가정한다. (사각형 형태도 물론 본 알고리즘의 적용에는 가능하다.)

2.3. 문제 개요

본 연구에서 사용된 센서 할당 및 배치 알고리즘(이후로 OPT-SA 지칭함)은 우선 hotspot들의 개수, 이들을 조사할 ring oscillator들이 미치는 가능 반경 그리고 각 hotspot들의 FPGA 상 x좌표와 y좌표가 입력되어 있는 입력 데이터가 주어진다. 이 데이터는 랜덤 함수를 통해 만든 무작위적인 값일 수도 있고, power model 패키지를 이용한 시뮬레이션 데이터일 수도 있다. 이를 통해 우리가 얻고자 하는 데이터는 이 hotspot들을 모두 조사하고, 최소 개수의 센서를 사용할 때 각 센서들의 위치 데이터이다. 최적의 해를 찾는 가장 확실한 방법은 branch-and-bound 알고리즘이기 때문에 우선 이 알고리즘에 적용할 수 있는 후보지들을 선정할 필요가 있다. 후보지에 대한 데이터는 후보지들의 개수, 각 후보지의 위치와 각 후보지들이 덮는 hotspot의 번호들의 집합으로 이루어질 것이다.

2.4 센서 할당 및 배치 알고리즘 (OPT-SA)

2.4.1 Step 1: Coloring Candidates

Hotspot을 조사하는 최적의 센서 배치 조합을 찾기 위해서는, 우선 센서를 배치할 수 있는 후보지를 선정해야 한다. 본 알고리즘에서 '후보지'는 'CLB array 내에서 센서를 배치했을 때, 하나 이상의 hotspot을 조사하며, 센서가 조사하는 hotspot의 조합이 동일한 점들의 집합이라 정의한다. 센서의 반경 이내의 모든hotspot들을 조사할 수 있으므로, hotspot에서 센서의 반경을 가지는 원 이내에 위치한 모든 센서들은 해상하는hotspot을 조사할 수 있다. 따라서, 주어진 array의 hotspot들에 대해 후보지들은 각 hotspot을 중심으로 한 반경 *r*인 원들과, 그들의 교집합이 된다.

불연속적인 array들의 점을 '색칠'하기 위해 주사선 채움 알고리즘을 응용한다.

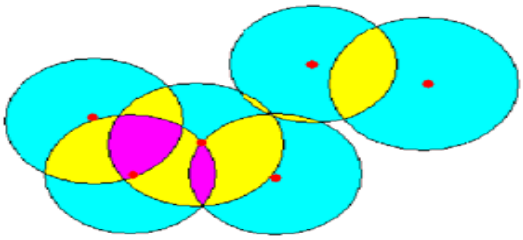

그림 2.4.1.1 후보지 색칠 모식도

(1) *array 상에 hotspot을 중심으로 하는 원을 그린다.*
(2) *해당 원의 경계선에 해당하는 점들에게 색깔로써 해당 hotspot의 번호를 지정한다.*
(3) *Array의 1행부터 interlacing을 시작해, 처음 경계선을 만났을 때 경계선에 해당하는 색깔을 만나는 점들에 대입한다.*
(4) *또 다른 경계선을 만났을 때, 그 경계선이 이미 지나온 경계선과 같은 색깔을 가질 경우 색칠하는 색깔의 정수에 해당 경계선의 색깔을 뺀다. 만약 다중 교집합을 색칠하고 있었다면, hotspot의 개수만큼을 더 뺀다.*
(5) *만난 경계선이 다른 원의 경계선일 경우, 그 다음부터 칠하는 색깔의 숫자를 (현재 색깔 + 해당하는 경계선의 색깔 + hotspot의 개수)로 바꾼다. 다른 영역과 구분하기 위하여 반드시 hotspot의 개수를 더해줄 필요가 있다.*
(6) *(5)단계까지 진행하면, 우리가 정의한 후보지들은 각자 다른 색을 지니는 영역들로 나타나게 된다. 하지만, 실제 센서의 중심은 하나의 점으로 표현되기 때문에 이들 중 하나의 점을 택하여 대표 점으로 사용할 필요가 있다. 센서의 중심에서 멀어질수록 측정 정확도가 떨어지는 센서의 특징을 고려하여 영역에 포함되는 점들 중 가장 위, 가장 아래, 가장 오른쪽, 가장 왼쪽의 점들을 찾아 그들이 이루는 사각형의 중심을 대표점으로 선발하였다.*

2.4.2 중간 값 포함 정리(Theorem)

이 정리는 "*다각형이 원들의 교집합 영역에 포함되는가?*" 와 "*그들의 중심점은 이 영역 내부에 존재하는가?*" 라는 것을 증명하는 데에서 출발한다.

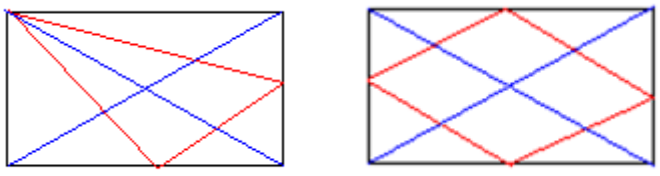

그림 2.4.2.1 사각형에서의 중간값 포함 모식도

전자를 증명하기 위해 삼각형인 경우와 사각형인 경우를 고려한다. 왜냐하면 볼록 다각형의 왼쪽, 오른쪽, 위, 아래 끝 값을 가진 좌표들로 사각형을 만들면 그 것은 다각형보다 작은 면적을 가지게 되는데 사각형의 면적 안에만 들어가 있음을 증명하면, 더 큰 면적에서 들어가 있음은 증명은 자연스럽게 가능하다. 삼각형일 경우엔 한 점이 구석에 있을 경우이고, 사각형일 경우엔 네 점이 각각 다른 변에 있을 경우이다. 이 때, 파란 선의 교점이 우리가 sample point로 삼고자 하는 곳인데 일단 파란변이 빨간 변 사이에 있으므로 그 안에 들어가 있음은 쉽게 보일 수 있다. 이것을 이용하면 원이 만나는 부분을 교점으로 한 도형이 볼록 다각형임을 보이면 증명이 끝나게 된다.

두 번째 단계로, 원이 만나는 곳을 꼭지점으로 한 도형이 볼록 다각형임을 증명하기 위해 귀납법을 사용한다. 두 개의 원이 만났을 경우는, 위와 아래 교점을 잡고 왼쪽 원의 아무 점, 오른쪽 원의 아무 점을 잡고 이은 것은 볼록 다각형이 됨은 자명하다.

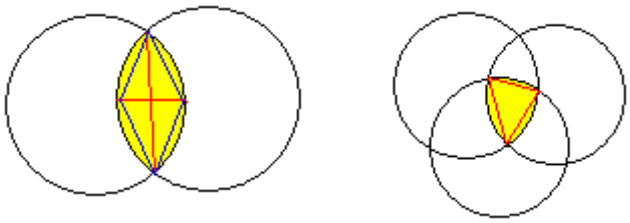

그림 2.4.2.2 원에서의 중간값 포함 모식도

왜냐하면, 두 개의 원이 만났을 경우에는 위 그림과 같은 경우밖에 없게 되는데, 이는 만나는 두 점과 오른쪽 변의 어떤 점, 왼쪽 변의 어떤 점을 잡으면 그것에 따라 그려진 사각형(그림 2.4.2.2참조)에서 왼쪽, 오른쪽, 위, 아래 끝 값을 잡을 수 있고, 이를 통하여 사각형을 그리면 이는 볼록 다각형이기 때문이다. 이를 통하여 세 개일 경우에는, 이 도형에 하나의 원이 더 첨가 되게 되는 셈인데, 이 때는 두 원에서 오른쪽 왼쪽 점을 잡을 때 그 것을 교점으로 잡게 되면, 그 점의 사잇각은 180도를 넘지 않음을 알 수 있기 때문이다. 따라서, 중간 값이 sample point 중에 포함된다.

2.4.3 Step 2: Candidates Filtering

앞의 단계에서 모든 후보지에 대한 데이터를 얻었으니 해당하는 데이터를 branch-and-bound알고리즘에 적용하여 최적의 후보지 조합을 찾을 것이다. 하지만, branch-and-bound 알고리즘은 사용하는 데이터의 숫자에 따라 실행 시간의 차이가 크게 나는 알고리즘이다. 따라서 불필요한 선택지와 필수적으로 선택해야 하는 후보지를 미리 제외시킨 후 branch-and-bound알고리즘에 대입할 필요가 있다. 본 연구에서 정의하는 해는 "모든 hotspot을 조사하는 후보지의 집합이며, 최적의 해는 "원소의 수가 가장 적은 해"이다. 후보지를 선정할 때, 최적의 해의 결과에 영향을 주어서는 안된다.

후보지들이 조사하는 hotspot들의 집합을 '조사 집합'이라 정의하자. 우선 이 조사 집합들의 포함 관계를 조사한다. 한 조사 집합에 대하여, 이 조사 집합이 어떤 조사 집합의 부분집합이라면 이 조사 집합을 부분 조사 집합, 이를 포함하는 조사 집합을 포함 조사 집합이라 정의한다.

부분 조사 집합을 원소로써 갖는 최적의 해를 가정하자. 이 최적의 해에 이 부분 조사 집합의 포함 조사 집합이 원소로써 포함되어있다고 가정하자. 이 경우, 이 부

분 조사 집합을 최적의 해에서 제외해도 모든 hotspot을 조사하므로, 해로써 인정될 수 있다. 하지만, 이 경우 앞서 정의한 최적의 해보다 부분 조사 집합을 제외한 해가 원소를 적게 가지므로, 최적의 해에서 부분 조사 집합과 포함 조사 집합이 함께 원소로 존재할 수 없다 라는 명제가 성립하게 된다. 그렇다면, 이 부분 조사 집합을 포함하는 최적의 해에서 해당하는 부분 조사 집합을 그것의 포함 조사 집합으로 치환하자. 이 경우, 최적의 해가 가지는 두개의 정의 중 모순되는 부분이 없으며 이 집합도 최적의 해 중 하나가 된다. 따라서, 부분 조사 집합을 고려 대상에서 제외하여도 최적의 해를 찾을 수 있다. 속도 증진을 위해, 모든 부분 조사 집합을 고려 대상에서 제외하도록 한다.

후보지들 중 "원소가 하나이며, 포함 조사 집합을 가지지 않는 조사 집합을 가진"후보지를 찾는다. 이를 필수 후보지라 정의하며 필수 후보지의 조사 집합의 원소를 Lone spot이라 정의한다. 필수 후보지가 제외된 후보지들의 집합을 생각하자. 정의에 의해 이 후보지들의 집합들 중 Lone spot을 조사하는 후보지가 존재하지 않게 된다. 그러므로 후보지들의 집합이 해의 조건을 만족하기 위해서는 반드시 필수 후보지를 포함할 필요가 생긴다. 해가 되지 못하면 최적의 해도 될 수 없으므로, 최적의 해도 반드시 필수 후보지를 포함한다. 앞에서 부분 조사 집합들을 모두 제외하였기 때문에 모든 후보지들의 조사 집합은 포함 조사 집합을 가지지 않는다. 따라서 이들 중 원소가 하나인 조사 집합을 가지는 후보지들은 필수 후보지이며, Branch-and-bound에서 이들을 고려하지 않고 우선적으로 최적의 해에 포함시킨다.

2.4.4 Step 3: Branch-and-Bound 알고리즘의 적용

앞의 알고리즘들을 통해 후보지들의 위치 좌표와, 그들의 조사 집합들이 정보로써 주어지게 된다. 그 후, 후보지들을 조사 집합의 원소 개수가 적어지는 순으로 배열한다. 조사 집합의 원소 수가 많은 후보지들부터 고르는 greedy알고리즘을 통해 우선적으로 upper-bound를 설정한다.

그 후, 탐색 함수의 재귀 호출을 실행한다. 탐색 함수는 후보지를 조사할 때, 자신이 찾고 있는 해에 해당하는 후보지를 추가하며, 이때 자신이 만들고 있는 해가 현재 설정된 upper bound의 원소 개수보다 많거나 같은 개수의 후보지를 가지고 있으면 이전 단계로 되돌아간다. 반복문을 통해 포함될 최적의 해를 찾게 되는데 만약에 자신이 필수 후보지일 경우, 바로 최적의 해에 대입시킨다. 만약에 현재까지 조사한 후보지의 집합이 해의 조건을 만족시킬 경우, 곧바로 upper bound에 자신의 해를 대입한다. 이를 모든 후보지를 조사할 때까지 반복한다. 그림 2.4.4.1은 탐색 함수가 처리하는 작업의 Pseudo-code를 보여준다.

```
Input : n=hot spot 개수, m=후보지 개수, spotloca=후보지 좌표, spot=후보지가 커버하는 hot spot 집합
Output : 모든 hot spot을 커버하는 sensor의 최소 개수와 그 위치
num=지금까지 선택한 hotspot 개수, now=지금 수행하고 있는 후보지, count=upper bound

초기 upper bound 설정 - 조사되지 않은 hotspot을 가장 많이 조사하는 후보지를 무조건 선택하여 모두 조사될 때의 sensor 개수
void recur(int num,int now){
 int i,j,k;
 if (num<count) {
   hot spot이 모두 조사되는지 검사
   if (hot spot이 모두 조사된다) {
      count=num; //upper bound 새로이 설정
      hot sensor 집합 저장
   }if (다 조사되지 않았다)
     for (i=now+1; i<m; i++){
      for (j=i+1; j<m; j++){
           i번째 후보지가 j번째 후보지에 포함되는 지 검사(조사되지 않은 hot spot만으로)
      }
      if (i번째 후보지가 다른 후보지에 포함되지 않는다){
           hot sensor로 선택
           recur(num+1,i);
           hot sensor로 선택한 것을 취소함
           i번째 후보지가 어떤 hot spot을 유일하게 조사하는지 조사
           if (i번째 후보지가 어떤 hot spot을 유일하게 조사한다){
                break;
           }
           }
 }
}
```

그림 2.4.4.1 OPT-SA의 Pseudo 코드

3. 실험결과

원형의 센서 범위를 상정하고 만든 최적 배치 알고리즘이 사각형으로 센서 범위를 상정한recursive bisection algorithm[참고문헌 1]에 비해 가지는 이점과 효율 증가를 확인하기 위하여 [참고문헌 1]의 제안한 알고리즘, 즉 bisection algorithm 중 몇 가지 후-보정과 내부 알고리즘 개선을 통해 가장 적은 수의ring oscillator 센서를 배치하는 것으로 알려진 longest edge neighbor detect(lg-nb-det)알고리즘과 우리의 제안한 최적 배치 알고리즘에서의 센서 개수를 비교하였다.

3.1 실험 set-up

OPT-SA 와 recursive-bisection[1](랜덤데이터 사용):
두 알고리즘의 실행 결과를 비교하는 실험을 하기 위해서는 알고리즘 외의 다른 요건들을 모두 고정시킬 필요가 있다. 사용하는 센서의 여건을 같게 하기 위하여[참고문헌 1]의 Recursive-bisection algorithm 중 최적의 해를 찾는다고 알려진 lg-nb-net 알고리즘을 원형 알고리즘에 맞도록 수정하였으며, OPT-SA 프로그램과 같은 입력 값을 통해 알고리즘을 실행하도록 하였다. CLB array 사이즈는 96*64, 128*86, 160*110, 192*128로 상정하였으며, 각 array사이즈마다 5개에서 100개의

hotspot위치를 JAVA랜덤 클래스를 통해 무작위로 생성하였다.

벤치마크 데이터를 통한 데이터에서의 결과 비교 무작위로 만든 데이터는 실제 FPGA에서 논리 소자의 분포와 작업 밀도에 의해 발생하는 온도 변화와 hotspot의 분포 경향을 제대로 반영하지 못한다. 따라서 실제 상황에서의 본 알고리즘의 효용성을 알아보기 위하여 power model 패키지의 ace, t-vpack VPR, Hotspot 패키지들을 이용, netlist 파일로부터 시뮬레이션 된 핫스팟의 위치들을 구하여(50*50 clb array)위와 같은 방식으로 실험을 진행할 것이다.[참고 문헌 1,3]

탐색 후보영역 최소화 알고리즘의 적용 유무에 따른 속도(run rime) 비교: Branch-and-bound 알고리즘은, backtracking 알고리즘을 기반으로 하기에 반드시 최적의 해를 찾는다는 장점이 있지만 기본적으로 지수승의 시간 복잡도를 가지는 단점이 있다. 따라서 본 알고리즘의 Branch-and-bound 알고리즘의 실행 속도도 후보지 색칠 알고리즘에서 나온 후보지들의 개수에 크게 좌우된다. 따라서 실제로 사용할 후보지를 추려내는 알고리즘의 효용성을 실행 시간 비교를 통해 알아보고자 한다

3.2. 실험 결과

랜덤 데이터에서의 알고리즘 비교

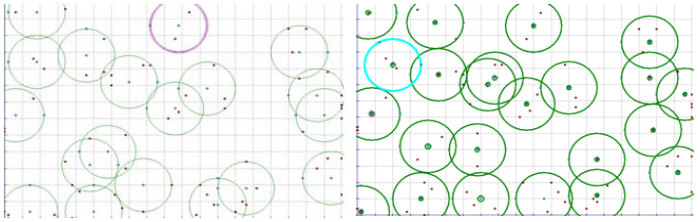

그림 3.2.1 OPT-SA(왼)과 Bisection(오른)의 센서 배치도

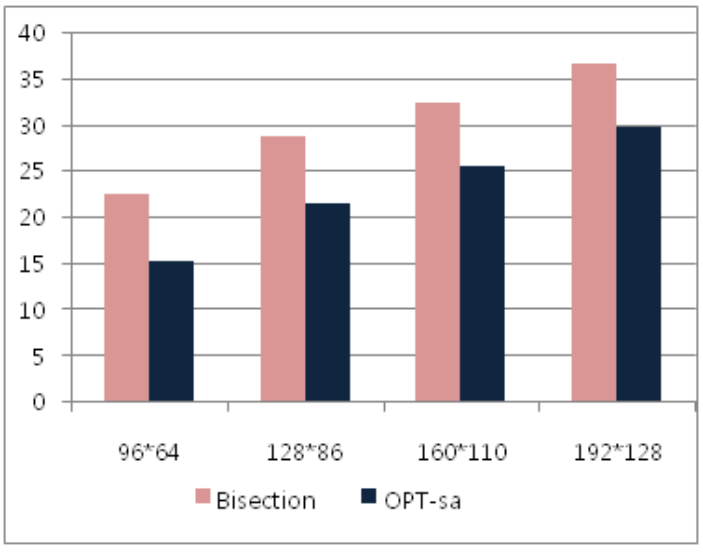

그림 3.2.2 Bisection(왼)과 Opt-SA(오른)의 센서 개수 비교

Bisection algorithm은 조건에 부합하는 사각형의 영역을 찾기 위한 알고리즘으로, 원형 센서를 배치하기 위해서도 원형이 배치될 수 있는 사각형의 영역을 찾는다. 그림 3.2.1에 보이듯이 원형에 적합하지 않은 Bisection algorithm의 단점이 결과에 여실히 나타난다. 실제 센서 배치도에서도 배치할 필요 없는 영역에 센서를 배치한 모습이 눈에 띈다. 그림 3.2.2sms 실험 결과를 요약한 것이다.

벤치마크 데이터를 통한 데이터에서의 결과 비교

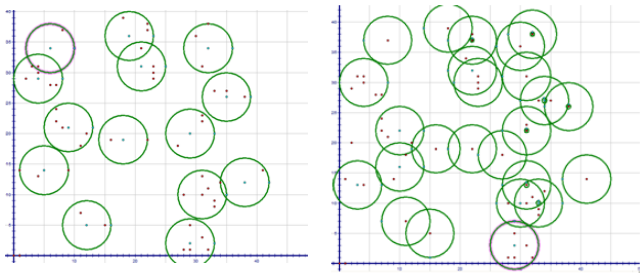

그림 3.2.2.1 OPT-SA(좌)와 Bisection(우)의 결과 (ex5p)

Netlist	사용 sensor 개수			감소율 (%)
	Hotspot 수	Bisection	OPT-SA	
ex5p	29	7	4	42.9
frisc	46	11	9	18.2
s298	17	5	4	20
s38417	46	25	14	44
s38584_1	18	11	10	9.1
s38584_2	28	14	12	14.3
tseng	123	11	7	36.4
평균	43.86	12	8.57	**28.6**

표 3.2.2.1 시뮬레이션 데이터에서 각 알고리즘이 얻은 해

50 * 50의 작은 array size를 고려하여 r=4로 하였다. 전과는 달리, 각 netlist파일에서 도출된 hotspot결과를 기준으로 데이터를 정리하였다. 랜덤 데이터 분포에 비해 hotspot들이 몰리는 경향을 보여, 작은 반지름에도 불구하고 적은 센서로 모두를 조사할 수 있었다. 적은 수의 센서만으로도 조사가 가능한 지역에서 센서를 여러 개 사용하는 모습을 보면 Bisection algorithm이 가지는 태생적 한계를 유추할 수 있다. Hotspot이 집중되면서, OPT-SA가 가지는 효율은 증가하였다

후보영역 최소화 알고리즘의 유무에 따른 속도 비교 실험한 CLB array의 사이즈는 192*128이다. Log를 취했을 때 1차 선형 함수가 나온다는 사실은 시간 복잡도가 $O(x^n)$ 인 branch-and-bound 알고리즘의 특징을 보여준다. 후보지 선정을 통한 최적화가 이루어지지 않을 경우 최적화가 이루어졌을 때에 비해 큰 증가율을 보였다. 후보지의 개수가 Branch-and-Bound의 실행 시간에서 지

수 역할을 하므로, 최적화가 이루어진 경우와 아닌 경우의 실행 시간 차이는 아주 크다. 납득할만한 시간 내에서 프로그램을 돌리기 위해선 위에서 설명한 최적화 알고리즘이 필수적이다.

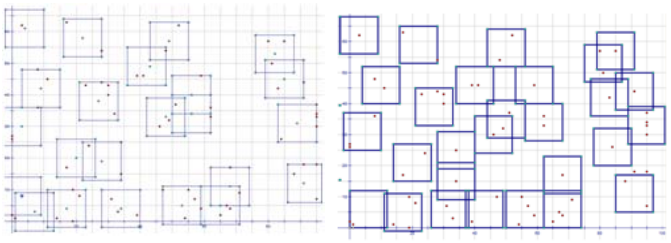

그림 3.2.4.2 정사각형 센서에서 OPT-SA(왼)와 Bisection(오른)의 센서배치

#Hotspot	Non-opt(#Candidate)	Opt(#Candidate)
30	0.016s(46)	0.016s(22)
40	2.968s(68)	0.016s(29)
50	2808.78s(102)	0.016s(34)
55	>1hr(122)	0.015s(40)

표 3.2.3.1 Hotspot 개수에 따른 후보지 개수와 실행시간

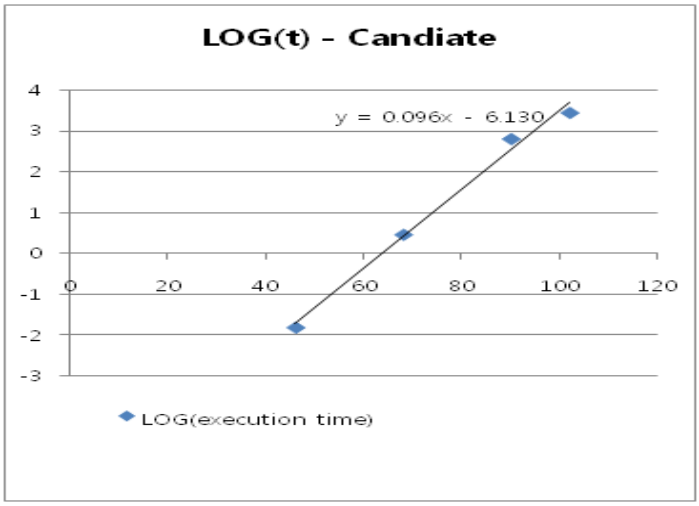

그림 3.2.3.1 후보지 개수와 실행 시간과의 로그 관계

그림 3.2.3.1은 후보지 filtering 기법을 적용하지 않은 경우와 적용한 경우 알고리즘의 수행 속도의 비교를 보여주며, 그림 3.2.3.1은 후보지 개수 증가에 따른 본 제안한 후보지 filtering 이용 시의 알고리즘 수행 시간을 보여 주고 있다. 마지막으로

4. 결론

본 연구에서는, ring oscillator 센서를 통한 온도 관측의 오차를 줄이기 위해, 모든 hotspot을 조사하기 위해 필요한 센서의 개수를 줄이는 과정에서, 실제 센서를 내접사각형으로 근사시켜 bisection algorithm 으로 해를 찾고 후-보정을 통해 최적 솔루션을 찾는 기존의 방법과는 달리, 내접 사각형으로 근사시키지 않고 원의 면적을 모두 사용하는 방법을 통해 근본적인 효율의 증가를 꾀하였다. Branch-and-bound 알고리즘을 이용함으로써 후-보정 포함 최대 O(n3)의 시간 복잡도를 갖는 기존 방법에 비해 매우 느리지만, 증명된 최적의 솔루션을 찾을 수 있었고, 후보지 선정을 통하여 납득할만한 시간 범위 내에서 결과를 도출하는데 성공하였다. 관측 정확도와 정밀도가 중요한 연구의 특성 상 최소의 센서를 이용하는 배치를 찾는 본 알고리즘의 효용성은 매우 높다고 볼 수 있다.

감사의 글: 본 논문은 한국과학재단의 RE 영재 프로그램과 지식경제부가 지원하는 국가 반도체 연구 개발 사업인 "시스템 집적 반도체 기반 기술 개발 사업(시스템 IC 2010)"을 통해 개발된 결과이며, 또한, "서울시 산학연 협력사업"으로 수행한 것이며, 2007년도 정부의 재원으로 한국과학재단의 지원 (과제번호: R01-2007-000-20891-0)을 받아 수행된 연구임.

5. 참고문헌

1. R. Mukherjee, S. Mondal, and S. Memik, "Thermal sensor allocation and placement for reconfigurable systems," *ACM/IEEE International Conference on Computer-Aided Design*, 2006

2. S. Mondal, R. Mukherjee, and S. Memik, "Fine-grain thermal profiling and sensor insertion for FPGAs," *IEEE International Symposium on Circuits and Systems*, 2006.

3. Xilinx, *Answers Database: Virtex/Virtex-E/Virtex-II/Virtex Pro/Virtex-4 - What are temperature sensing diode pins (DXP and DXN, TDN, and TDP)?*, 2005.

4. S. Lopez-Buedo, J. Garrido, and E. Boemo, ""Thermal testing on reconfigurable computers,"" *IEEE Design & Test of Computers*, 2000.

5. S. Gunther, *et al.*, ""Managing the impact of increasing microprocessor power consumption,"" *Intel Technology Journal*, Vol. 5, February 2001.

6. K. -J. Lee, K. Skadron, and W. Huang, ""Analytical model for sensor placement on microprocessors,"" *IEEE International conference on Computer Design*, 2005.

7. S. Lopez-Buedo, E. Boemo, ""Making visible the thermal behavior of embedded microprocessors on FPGAs: a progress report,"" *Proc. International Symposium on Field Programmable Gate Arrays*, 2004.