論文97-34C-2-5

# Field Programmable Circuit Board를 위한 위상 기반 회로 분할

## (A Topology-Based Circuit Partitioning for Field Programmable Circuit Board)

崔 然 景 *, 林 鐘 錫 **

(Yhon Kyong Choi and Chong Suck Rim)

요 약

본 논문에서는 ASIC 시제품 제작을 위하여 회로를 프로그래밍이 가능한 FPCB 상의 다중의 칩으로 분할하는 새로운 방법을 제안한다. FPCB는 로직을 위한 FPGA 칩 영역과 연결 요소로 구성되어 있으며 칩들은 서로간을 연결하기 위한 배선 위상이 정해져 있다. FPCB에서의 회로 분할 문제는 기존의 분할 문제와는 달리 칩들간의 배선을 위하여 사용하는 연결선의 수에 대한 제한이 존재한다. 본 논문에서는 먼저 이런 제한을 포함하는 위상 기반 분할 문제를 소개하고 이 문제를 해결하기 위한 휴리스틱으로 시뮬레이티드 어닐링 기법을 이용한 분할 방법을 제안한다. 또한 적은 수행시간에 좋은 회로 분할을 얻기 위한 방법으로 다단계 트리 클러스터링 방법을 제안한다. 다수의 회로에 대하여 실험한 결과 FPCB를 위한 제한을 모두 만족하였으며 큰 회로에 대해서는 기존의 K-way 방법을 수정한 방법과 비교하여 약 두 배 정도의 시간에 분할이 가능하였다.

**Abstract**

In this paper, we describe partitioning large circuits into multiple chips on the programmable FPCB for rapid prototyping. FPCBs consist of areas for FPGAs for logic and interconnect components, and the routing topology among them are predetermined. In the partition problem for FPCBs, the number of wires for routing among chips is fixed, which is an additional constraints to the conventional partition problem. In order to deal with such a constraint properly we first define a new partition problem, so called the topology-based partition problem, and then propose a heuristic method. The heuristic method is based on the simulated annealing and clustering technique. The multi-level tree clustering technique is used to obtain faster and better partition results. In the experimental results for several test circuits, the restrictions for FPCB were all satisfied and the needed execution time was about twice the modified K-way partition method for large circuits.

### I. 서 론

현재 빠른 시제품 제작을 위하여 FPGA(Field Programmable Gate Array)와 같은 소자들이 널리 사용되고 있다. 그러나 수십만이 넘는 게이트로 구성된 VLSI 칩의 경우 FPGA 하나로는 시제품 제작이 불가능하다. 이를 해결하기 위하여 최근 여러 개의 FPGA 칩과 이들간의 연결을 위한 칩을 장착한 PCB(Programmable Circuit Board)가 사용되고 있다. FPCB (Field Programmable Circuit Board)라고 하는 보드를 사용하여 시스템을 시험할 경우, 먼저 시스템을 하나의 FPGA에 구현할 수 있는 크기의 여러 블럭들로 분할한 후 이들을 FPCB의 각 FPGA에 구현하고

* 正會員, 警敏專門大學 電子計算科
(Dept. of Computer Science, Kyung Min College)
** 正會員, 西江大學校 電子計算學科
(Dept. of Computer Science, Sogang University)
※ 이 논문은 1995년도 한국전자통신연구소의 연구 조성비에 의하여 연구되었음.
接受日字:1996年8月13日, 수정완료일:1997年2月5日

이들간을 스위치 기능을 하는 칩을 통하여 연결한다. 이의 한 예로 상용중인 Aptix사의 AXB-AP4 보드[21]가 있으며 그 보드에서 사용되는 스위치 칩으로는 FPIC(Field Programmable Interconnect Component)™가 있다.

그런데 FPCB에는 사용할 수 있는 FPGA와 스위치 칩들이 고정되어 있으며 각 칩간을 연결하기 위한 연결선들의 집합인 채널(channel)이 존재할 수 있다. 채널들은 칩간의 배선 위상(topology)을 형성하여 각 FPGA간의 가능한 연결 형태를 나타낸다. 각 채널에서 사용할 수 있는 연결선의 수는 정해져 있으며 이를 채널의 용량(capacity)이라고 한다. 여기서 주어진 회로를 FPCB 상의 칩으로 분할하기 위해서는 다음의 두 가지 사항들이 고려되어야 한다. 첫째는 주어진 FPGA 칩의 용량과 I/O 핀의 수이며 둘째는 각 채널을 사용하는 네트의 수이다. 두번째 항목은 스위치 칩을 통하여 FPGA 칩간을 100 % 배선하기 위하여 필요한 고려사항이다. 본 논문에서는 앞의 두 항목을 모두 고려하는 새로운 배선 문제인 위상 기반 분할 문제(topology-based partition problem)를 제안한다.

이미 분할을 위한 연구가 많이 이루어졌으나 기존의 분할 방법은 오직 첫번째 항목만을 고려하거나[5,14,15,17,25] 서로 다른 블럭에 속한 네트의 수(컷(cut))를 최소화하기 위한 것[1-4,8-9,11,13,18,19,21, 23,26]이며, 네트의 배선시 주어진 레이아웃 형태에서 핀의 기하학적(geometric)인 위치를 고려하기 때문에 칩간을 100 % 배선하기가 힘들다. 그러므로 본 논문에서는 위상 기반 분할 문제를 위한 시뮬레이티드 어닐링 기법(simulated annealing technique)을 이용한 새로운 분할 방법을 제안한다.

시뮬레이티드 어닐링 기법은 비록 수행시간이 많이 요구되는 기법이나 최적의 해를 얻을 수 있는 방법으로 새롭게 제안한 문제나 알고리즘의 성능 분석을 비교하기 위한 자료를 위해서 이용되고 있으며[10,12,20,24] 회로를 클러스터링하여 분할의 수행시간을 개선한 결과가 있다[17]. 그러므로 본 논문에서는 다단계 트리 클러스터링(multi-level tree clustering)에 의한 클러스터 트리를 사용하는 새로운 시뮬레이티드 어닐링 분할 방법을 제안한다. 이는 기존의 클러스터링을 이용한 분할 방법과는 달리 클러스터 트리를 사용하여 위의 레벨에서 부터 아래 레벨로 클러스터들을 교체해가며 분할한다. 그 결과 큰 회로의 경우 K-way 분할 방법의 약 두 배 정도의 시간이면 좋은 분할을 얻을 수 있음을 보인다.

본 논문은 서론을 포함하여 모두 여섯 장으로 구성된다. 2 장에서 위상 기반 분할 문제를 정의하고 3 장에서는 분할 방법을 좀 더 빠른 시간에 수행하기 위한 다단계 트리 클러스터링 방법을 제안한다. 4 장에서는 시뮬레이티드 어닐링 기법을 이용한 분할 방법에 대하여 기술하고 5 장에 새로운 분할 방법을 실험한 결과를 보인다. 마지막으로 6장에서 결론을 낸다.

## Ⅱ. 위상 기반 분할 문제 정의

이 장에서는 위상 기반 분할 문제를 정의한다. 서론에서 언급한 바와 같이 FPCB는 사용할 수 있는 FPGA 칩과 스위치 칩이 제한되어 있으며 각 두 칩간에는 연결선들의 집합인 채널이 존재한다. 서로 다른 두 FPGA 칩에 존재하는 로직간을 연결하기 위해서는 한 개 이상의 스위치 칩을 사용하여야 하며, 임의의 두 로직이 서로 다른 FPGA에 놓일 경우에도 스위치 칩과 채널을 사용하여 이들간의 배선이 가능하도록 구성되어 있다. 그러나 각 채널은 용량이 제한되어 있어서 네트의 배선시 그 용량을 초과하여 사용할 수 없다. 이런 FPCB 상의 칩들과 채널은 다음과 같이 그래프로 나타낼 수 있다. 즉 FPCB는 FPGA 칩과 스위치 칩에 대응하는 정점의 집합을 각각 $L$ 과 $I$, 채널에 대응하는 에지의 집합을 $E$ 로 하는 위상그래프(topology graph) $G_T=(L\cup I, E)$로 표현할 수 있다. 각 에지에는 채널의 용량을 나타내는 에지의 용량이 존재한다.

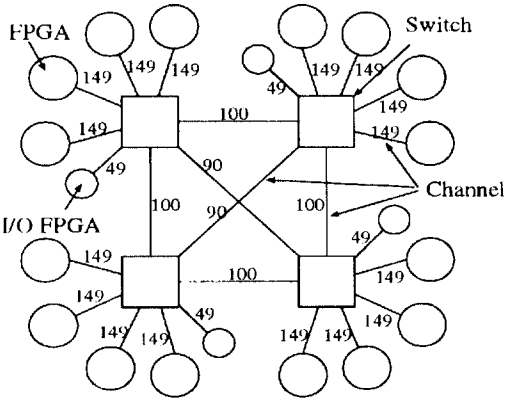



그림 1. 위상그래프의 예
Fig. 1. An example of a topology graph.

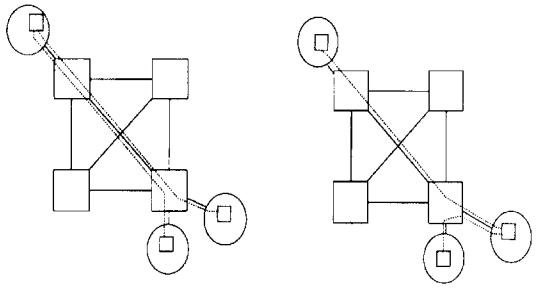
그림 1에 위상그래프의 한 예를 보인다. 그림 1은 FPCB AXB-AP4 보드를 위상그래프로 나타낸 것이다. FPGA 칩과 스위치 칩을 구분하기 위하여 이들을 각각 원과 사각형으로 나타낸다. 그림에 큰 원은 로직 구현용 FPGA 칩을 나타내며 작은 원은 보드 밖으로의 연결을 위해서 사용하는 I/O 구현용 FPGA 칩(이하 "I/O FPGA"라 함)을 나타낸다. 각 에지에 쓰여진 수는 채널의 용량을 나타낸다.

회로를 FPCB로 분할하는 것은 회로를 위상그래프상으로 분할하는 것과 동일하며 이를 위해서는 다음의 두 가지 제약이 만족되어야 한다.

(1) 각 정점을 위한 모듈의 크기가 대응하는 FPGA 칩의 크기를 넘지 않아야 한다.

(2) 분할 후에 FPGA 칩간을 연결하기 위하여 필요한 네트의 수가 각 채널의 용량을 넘지 않아야 한다.

그러므로 FPCB로의 분할 문제는 입력 회로의 모듈의 집합을 $M$, 네트의 집합을 $N$이라고 할 때 위의 두 제약을 만족하고 사용하는 FPGA 칩의 수나 스위치 칩의 수를 최소화하면서 모듈의 집합 $M$을 위상그래프의 FPGA 칩의 집합 $L$ (앞으로 편의상 그냥 블럭으로 부름)로 분할하는 문제이며 이를 *위상 기반 분할 문제* 라고 한다.



(a) (b)

그림 2. 세 모듈의 분할
(a) 분할 예 1. (b) 분할 예 2.

Fig. 2. Partition of three modules.
(a) Partition example 1. (b) Partition example 2.

지금까지 회로를 분할하는 방법에 대해서는 많은 연구가 이루어져 왔으나, 서론에서 기술한 바와 같이 채널에 대한 고려가 없기 때문에 위상그래프 $G_T$의 제약들을 모두 만족하는 좋은 해를 구하기가 힘들다. 결과적으로 기존의 분할 방법을 이용하여 분할을 하고 각 분할된 회로를 FPGA 칩에 넣고 그들간을 연결하였을 경우에 각 채널에서 사용하는 네트의 수가 채널의 용량을 넘을 수 있다.

그림 2에 한 예를 보인다. 그림의 위상그래프에서 정점안의 작은 사각형은 분할하려는 회로의 모듈을 나타내며 점선은 모듈간의 네트를 나타낸다. 기존의 분할 방법에서는 그림 2 (a)와 (b)의 두 분할의 해에서 사용하는 핀의 수가 같기 때문에 이들을 같은 해로 취급한다. 그러나 그림 2 (a)에서는 두 개의 네트가 사용하는 채널이 모두 여섯 개이고 그림 2 (b)에서는 모두 다섯 개를 사용하므로 그림 2 (b)의 해가 FPCB를 위한 분할의 해로 더 적합하다. 그러므로 위상 기반 분할 문제를 위한 새로운 분할 방법이 요구된다.

다른 예로 그림 1과 같은 위상그래프의 경우에는 기존의 사분할 방법[22]을 두 단계로 사용하여 회로를 분할할 수 있겠다. 첫 단계에서는 동일한 스위치 칩과 연결된 FPGA 칩을 하나의 블럭으로 하여 사분할을 수행한다. 두번째 단계에서는 첫번째 단계의 분할된 결과를 가지고 각 FPGA 칩으로 분할하는 사분할을 수행한다. 그러나 이와 같이 할 경우에는 첫번째 단계에서 스위치 칩간의 연결이 최소화되기 때문에 스위치 칩과 FPGA 칩 사이의 연결선들이 많아져서 채널의 용량이 초과될 가능성이 있다. 그러므로 위상 기반 분할 문제를 해결하는 새로운 분할 방법인 시뮬레이티드 어닐링 기법을 이용한 분할 방법을 제안한다. 먼저 시뮬레이티드 어닐링 기법을 이용한 방법의 수행시간을 개선하기 위한 클러스터링 방법을 다음 장에서 설명한다.

## III. 트리 클러스터링

일반적으로 클러스터링은 분할 방법에서 대상이 되는 입력 회로의 크기가 큰 경우에 흔히 사용되고 있다[4,17,19,25]. 시뮬레이티드 어닐링 기법을 이용한 분할 방법에서도 클러스터링된 회로를 이용할 경우에 빠른 시간에 해를 얻을 수 있으나 클러스터링하지 않은 회로를 이용하여 얻은 해에 비하여 결과가 좋지 않을 수 있다. 그리고 만일 회로가 계층구조(hierarchy)를 갖도록 설계된 경우에는 회로의 계층구조를 무시하고 분할하기 보다는 각 모듈을 위한 회로들이 서로 같은 블럭에 속하도록 하는 것이 바람직하다. 이 경우에는 모듈을 분할을 위한 이동의 단위로 사용할 수 있다. 그러므

로 이 절에서는 적은 시간에 좋은 분할을 얻기 위한 계층구조가 포함된 클러스터 트리를 구성하는 방법과 클러스터 트리를 분할에 활용하는 방법을 제안한다.

회로들은 두 종류의 클러스터링 방법에 의하여 클러스터링된다. 회로들은 먼저 Cong[6]이 제안한 MFFC (Maximum Fanout Free Cone) 단위로 묶고 Local Ratio Cut(LRC) 방법[5]을 수정한 클러스터링 방법을 이용하여 그림 3과 같이 클러스터들이 트리 형태를 형성하도록 반복적으로 클러스터링을 한다. 그림에 삼각형은 MFFC 클러스터를, 원은 시스템 디자이너가 회로를 계층적으로 디자인하였을 경우에 존재하는 모듈을 나타낸다. 사각형은 LRC 분할 방법을 수정한 클러스터링 방법에 의해 생성된 클러스터들로써 부모 클러스터들은 아래 레벨(level)의 자식 클러스터들을 클러스터링하여 생성된 것들이다.

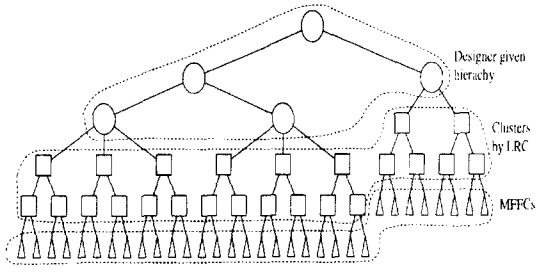



그림 3. 모듈의 계층구조와 클러스터 트리
Fig. 3. Hierarchy of modules and cluster tree.

클러스터 트리의 클러스터들은 다음과 같은 특징을 갖는다. 클러스터 트리의 루트(root)에 인접할수록 클러스터의 크기가 크고 분할을 위한 클러스터의 수가 적기 때문에 이들을 사용하여 분할을 할 경우에는 빠른 시간에 해를 얻을 수 있다. 반대로 클러스터 트리의 리프(leaf)에 인접할수록 클러스터의 크기는 작고 분할을 위한 대상 클러스터의 수가 많기 때문에 앞의 경우보다 더 좋은 해를 기대할 수 있으나 시간이 많이 필요하다. 그러므로 본 논문에서는 이런 특성을 이용하여 루트(root)에 인접한 클러스터에서 부터 분할을 시작하며 분할 도중에 트리의 리프쪽으로 내려오면서 클러스터들을 교체하면서 분할한다. 즉 선택한 클러스터들을 대상으로 해를 구하다가 문제의 제한을 만족하는 해를 얻으면 수행을 중단할 수 있으며, 보다 좋은 해를 얻고 싶은 경우에는 클러스터를 교체하면서 수행을 계속 반복한다.

클러스터링 시에 회로를 먼저 MFFC 단위로 클러스터링하는 것은 분할된 결과들이 주어진 FPGA 칩에 놓이게 되므로 LUT(Look-Up Table)가 적당한 단위이기 때문이며[25], 각 FPGA 칩은 그 칩에 존재하는 CLB의 수와 LUT의 수는 고정되어 있지만 하나의 CLB에 집적할 수 있는 게이트의 수는 고정되어 있지 않기 때문이다. 그러나 기술사상(technology mapping) 과정전에는 입력 회로가 사용하는 LUT의 수를 알 수 없으므로 입력 회로는 MFFC 단위로 클러스터링하며 그의 크기를 LUT의 수로 정의한다. MFFC 단위의 클러스터들은 앞으로 MFFC 클러스터라고 한다.

주어진 회로에서 MFFC 클러스터는 회로의 primary 출력에서 시작하여 primary 입력으로 BFS (Breadth First Search)에 의하여 각 게이트를 방문(visit)하면서 구한다. MFFC 클러스터의 크기도 주어진 MFFC의 입력을 primary 입력으로 하고 MFFC 클러스터의 출력을 primary 출력으로 하여 primary 입력에서 primay 출력으로 BFS에 의해 게이트를 방문하면서 구한다. 이것는 선형 시간(linear time)을 갖는 알고리즘으로 매우 빠른 시간에 그 크기를 구할 수 있다[29].

LRC 분할 방법은 컷/크기 비율을 줄이면서 회로를 빠른 시간에 다중의 칩으로 분할하는 방법으로 본 논문에서는 MFFC 클러스터들을 클러스터링하는데 사용하기 위하여 이를 수정하였다. 즉 LRC 분할 방법에서와 같이 MFFC 클러스터들을 대상으로 컷/크기 비율을 가능한 줄이는 클러스터들을 생성하도록 노력한다. 그러나 클러스터들의 크기의 차이가 너무 클 경우에는 분할 방법에서 클러스터들을 이동시키기가 어려우므로 각 클러스터들의 크기를 제한하도록 수정하였으며 각 클러스터의 크기는 인수로써 사용자가 정의할 수 있도록 하였다. 이때 생성된 클러스터들의 크기는 이 클러스터를 생성하기 위해 포함된 클러스터들의 크기의 합으로 결정한다. LRC 방법을 수정한 클러스터링 방법은 앞으로 LRC 클러스터링 방법으로 부른다.

그림 4에 클러스터링 알고리즘을 보인다. 여기서 MFFC_clustering()는 MFFC 단위로 클러스터링하는 함수를 나타내며 LRC_clustering(Max,Min) 함수는 LRC 클러스터링을 하는 함수를 나타낸다. 이때 Max와 Min은 각각 LRC 클러스터링을 할 때 생성될 클러스터들의 최대, 최소 크기 제한을 나타낸다. LRC 클러

스터링은 생성된 클러스터의 수나 크기가 만족될 때까지 반복한다.

```
MFFC_clustering()
while ( clustering is wanted ) do
begin
  get Max and Min
  LRC_clustering(Max, Min)
end
```

그림 4. 클러스터링 알고리즘
Fig. 4. Clustering algorithm.

## Ⅳ. 시뮬레이티드 어닐링 기법을 이용한 분할

이 장에서는 위상 기반 분할 문제를 위한 시뮬레이티드 어닐링 기법을 이용한 분할 방법에 대하여 설명한다. 이를 위하여 시뮬레이티드 어닐링 기법의 개념을 먼저 설명한다. 시뮬레이티드 어닐링은 많은 최적화 문제에서 좋은 결과를 내는 알고리즘으로 알려져 있다[12]. 반복적인 향상(iterative) 방법은 오직 비용이 적은 해만을 받아들여 지역 최소값(local optimum)을 갖는 해를 찾게 되는 문제가 있는데 시뮬레이티드 어닐링은 이런 지역 최소값에서 탈피할 수 있도록 한 방법이다. 그래서 이 알고리즘이 시간이 많이 걸린다는 문제에도 불구하고 많이 사용되고 있으며 새롭게 제안한 문제를 위한 알고리즘의 성능을 분석하기 위하여 흔히 이용되고 있다[10,12,24]. 또한 Sechen의 분할 방법[17]에서 클러스터링을 이용한 것처럼 수행 방법을 수정하여 많은 시간을 절약하여 해를 얻을 수 있다.

시뮬레이티드 어닐링 기법은 기본적으로 Metropolis[24] 절차를 이용한 것으로 어떤 해에서 이웃해를 반복적으로 구할 때 구한 이웃해의 비용 $S'$ 가 현재 해의 비용 $S$ 보다 감소할 경우에는 언제나 이웃해를 채택하고, $S'$ 가 $S$ 보다 클 경우에는 확률로써 이웃해로 채택할지를 결정한다. 초기에는 거의 모든 이웃해를 받아들이도록 매우 높은 초기 온도에서 작업을 시작한다. 한 온도에서는 충분한 평형 상태에 도달할 때까지 새로운 이웃해를 구한다. 여기서 평형 상태란 충분히 많은 이웃해가 받아들여진 상태를 말한다. 평형 상태에 도달하면 온도를 조금 내리고 새로운 온도에서 같은 과정을 되풀이한다. 이 알고리즘에서는 효율적인 이동 방법과 비용함수를 정의하는 것과 온도 변화에 따른 적절한 스케줄을 하는 것이 매우 중요하다. 그러므로 시뮬레이티드 어닐링 기법을 위상 기반 분할 문제에 적용하기 위해서는 아래에 열거한 항목에 대한 연구가 필요하다.

1) 현재 해로 부터 이웃해를 구하기 위한 효과적인 이동 방법,
2) 이웃해를 받아들일 것인가를 결정하기 위한 비용함수,
3) 냉각 스케줄,
4) 중단조건.

먼저 전체 알고리즘을 1 절에서 설명하고 이동방법과 비용함수 및 냉각 스케줄과 중단조건을 2 절과 3 절에 걸쳐 설명한다.

### 1. 분할 방법의 알고리즘

이 절에서는 분할 방법의 전체 알고리즘에 대하여 설명한다. 그림 5에 분할 방법의 전체 알고리즘을 보인다. 먼저 클러스터 트리의 루트(root)에서 부터 적당한 크기의 클러스터들을 선택하여 초기 분할을 하고 초기 온도를 구한다. 만일 초기 분할이나 선택한 클러스터들로 시뮬레이티드 어닐링 방법을 충분히 수행하여 분할 결과가 만족스러울 경우 즉, 위상 기반 분할 문제의 제약을 만족할 경우에는 수행을 중단하여 빠른 시간에 원하는 해를 얻을 수 있다. 그러나 제약을 만족하는 해를 얻을 수 없거나 보다 최적화된 해를 얻고 싶은 경우에는 그 다음 레벨에 존재하는 클러스터들을 이용하여 분할을 계속 진행한다. 이와 같이 클러스터 트리의 맨 아래의 클러스터까지 내려가면서 클러스터를 선택하여 분할을 수행할 수 있다. 그림 5의 중단 조건들은 3절에서 설명한다.

```
get clusters nearest from the root of cluster tree
S = Initial_solution(initial partition)
T = Initial_temperature
while ( finish criteria is not satisfied ) do
begin
        while ( cost is improving ) do
        begin
                while ( not equilibium ) do
                        Find_new_solution(T,S)
              T = T * Ratio;
        end
        if ( finish criteria is not satisfied ) then get
        new clusters
 end
```

그림 5. 분할 방법의 알고리즘
Fig. 5. Algorithm of the partition method.

시뮬레이티드 어닐링 기법을 이용한 분할 방법에서 FPGA 칩간의 네트를 100 % 연결시키기 위해서는 각 채널의 사용률을 알아서 이를 비용에 포함시켜야 한다. 여기서 채널 사용률은 채널의 용량과 그 채널을 사용하는 네트의 수를 백분율로 나타낸 것을 말한다. 그림 1의 보드 형태에서 두 개의 클러스터들이 서로 다른 스위치 칩에 연결된 FPGA 칩안에 존재한다고 가정하자. 두 클러스터들은 그림의 두 칩안에 놓이는 경우에는 배선시 그들간의 네트들의 연결을 위하여 그림 6 (a)와 같이 채널 A를 사용하여 배선하려고 할 것이다. 그러나 스위치 칩 2에 인접한 채널들에서 모두 채널 사용률이 100 %를 초과할 경우에는 채널 사용률이 100 %를 초과하지 않는 다른 채널이 존재하여도 네트를 배선할 수 없게 된다.

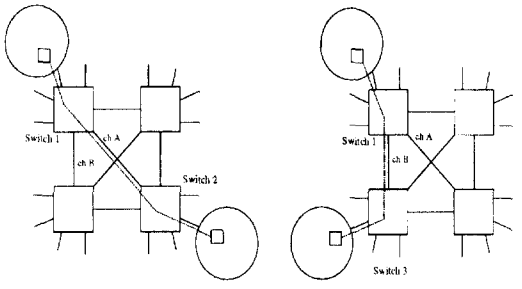



그림 6. 클러스터 위치에 의해 사용되는 채널
Fig. 6. Used channels depending on cluster location.

만일 분할의 과정에서 각 채널을 사용하는 네트의 수를 알고 있다면 채널의 용량을 초과하지 않는 채널을 사용하도록 클러스터들을 이동할 수 있다. 그림 6에서 스위치 칩 2에 인접한 채널들의 채널 사용률이 100 %를 초과하고 채널 B를 사용하는 네트의 수가 채널의 용량을 초과하지 않는다면 그림 6 (a)의 클러스터들은 그림 6 (b)와 같이 네트의 배선시 채널 B를 사용하도록 채널 B와 인접한 블럭에 놓는 것이 바람직하다. 그러므로 분할 과정에서 네트가 사용하는 채널들과 각 채널을 사용하는 네트의 수를 고려하여야 한다. 그러나 실제로 네트에 대한 배선이 이루어지기 전에는 이를 정확히 알 수 없으며 각 채널 사용률을 정확히 측정하는 것이 매우 어려우므로 본 분할 방법에서는 네트가 사용하는 채널들을 예측하기 위하여 실제로 각 네트에 대한 배선을 수행한다.

어떤 네트를 위한 배선은 위상그래프에서 스위치 칩에 해당하는 사각형의 정점에 대하여 미로배선[16]을 응용한 방법을 이용하여 구한다. 그것은 서로 다른 FPGA에 속한 클러스터들간의 네트는 그들간의 연결을 위하여 FPGA에 인접한 채널을 반드시 사용하여야 하므로 배선을 하지 않아도 그 채널이 사용됨을 알 수 있기 때문이다. 네트의 미로배선에서는 각 에지에 가중치를 주어 한 정점에서 다른 정점으로 수를 전파(propagate)할 경우에 가중치를 더하여 가중치의 합이 가장 작은 경로(path)를 선택하여 배선한다. 만일 어떤 채널 사용률이 100 %를 초과할 경우에는 큰 가중치를 주어 다른 채널을 사용하여 배선되도록 한다. 배선을 위한 정점 $I$ 의 크기를 $|I|$라고 하고, $I$ 에 속한 정점에서 $I$ 에 속한 서로 다른 정점을 연결하는 에지 수의 최대값을 $|E_{max}|$라고 할때 배선의 복잡도는 $O(|I| \cdot |E_{max}|)$이다. 그림 1의 위상그래프의 경우에는 $|I| = 4$이고, $|E_{max}| = 3$이므로 복잡도는 O(12)이다.

그런데 배선은 오직 클러스터의 이동에 의하여 네트의 배선 형태가 달라질 경우에만 수행한다. 그림 7과 같이 네 개의 블럭이 각각 다른 스위치 칩에 연결되어 있는 위상그래프에서 블럭 A에 속한 클러스터 a를 이동하려고 할 때 클러스터 a를 블럭 C로 옮길 경우에는 클러스터 a에 연결된 네트에 대한 배선을 수행하여야 한다. 그러나 클러스터 a를 블럭 B나 블럭 D로 이동할 때에는 블럭 B와 D에 같은 네트에 대한 배선이 이미 이루어져 있으므로 다시 배선을 수행하지 않는다. 이와 같이 배선 방법은 알고리즘의 복잡도가 크지 않고 네트를 필요한 경우에만 배선하므로 전체 수행시간에 큰 영향을 주지 않는다.

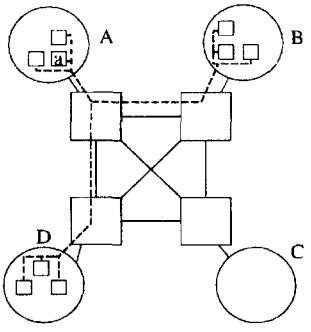



그림 7. 네트의 배선 예
Fig. 7. A routing example.

2. 이동 방법과 비용함수

이 절에서는 위상 기반 분할 문제를 시뮬레이티드

어널링 기법을 이용하여 효과적으로 해결하기 위한 이동 방법과 비용함수를 정의한다. 현재 해로 부터 이웃 해를 구하기 위하여 개발된 이동 방법의 기본 단위는 클러스터이며 개발한 이동 방법은 다음의 두 가지이다. 첫번째 이동은 서로 다른 블럭에 속한 두 클러스터를 원래 속해 있던 블럭에서 다른 클러스터가 속해 있던 블럭으로 이동시키는 것이다. 두번째 이동은 임의의 블럭에 속한 임의의 클러스터 하나를 원래 속해 있던 블럭이 아닌 다른 블럭으로 이동시키는 것이다

두번째 이동은 원래 속해 있던 블럭의 클러스터 수를 줄이므로 이로 인하여 사용하지 않는 블럭이 생기게 되어 결국 전체 사용하는 블럭의 수를 줄이는 데 효과적으로 사용될 수 있다. 반면 첫번째 이동은 두 클러스터를 교체하는 것이기 때문에 서로의 블럭 용량을 많이 변화시키지 않으므로 온도가 낮을 때에도 효과적으로 사용될 수 있다.

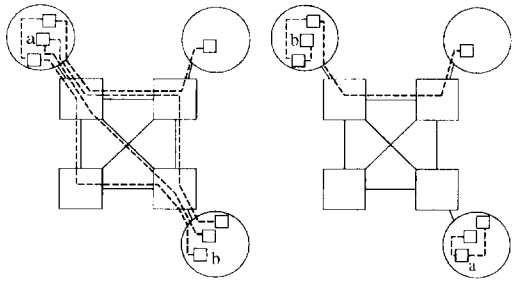



그림 8. 두 클러스터의 교체
Fig. 8. Exchanging of two clusters.

그림 8에 첫번째 이동 방법의 예로 두 클러스터 a와 b의 교체를 통하여 채널을 사용하는 네트가 현저히 줄어드는 것을 보인다. 그림 9에서는 두번째 이동 방법의 예를 보인다. 이 경우 클러스터 a를 이동함으로써 블럭 C가 비게 되므로 채널을 사용하는 네트뿐만 아니라 사용하는 블럭의 수도 줄일 수 있음을 보인다.

임의의 해의 비용 함수는 앞에서 설명한 위상 기반 분할 문제에서의 제한을 만족할 수 있도록 선택되어야 한다. 이를 위하여 정의한 비용 함수는 모두 다섯 가지이다. 첫번째 비용함수 $C_1$은 사용하는 블럭 수이고, 두번째 비용함수 $C_2$는 사용하는 I/O 블럭 수로 두 비용함수는 사용하는 블럭의 수를 최소화하기 위한 비용함수들이다.

세번째 비용 함수 $C_3$은 각 블럭에 들어가는 회로의 크기가 주어진 블럭의 크기를 벗어나는지에 대한 것이다. 이 비용을 분할을 수행하기 위해 $P(l_j)$를 블럭 $l_j$ 안에 놓인 클러스터들의 집합이라고 하고 블럭 $l_j$의 용량을 $S(l_j)$ $(l_j \in L)$, $S(P(l_j))$를 $P(l_j)$의 크기라고 정할때 비용함수는 다음과 같다.

$$C_3 = \sum_{l_j \in L} Y_j^2, \quad Y_j = \begin{cases} S(P(l_j)) - S(l_j) & \text{if } S(P(l_j)) > S(l_j), \\ 0 & otherwise \end{cases}$$

즉, 블럭안의 속하는 클러스터들의 크기의 합이 블럭의 용량을 넘을 경우에는 넘는 양의 제곱을 비용으로 하고 그렇지 않은 경우에는 비용을 0으로 한다.

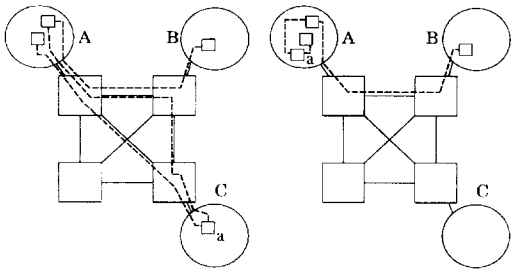



그림 9. 하나의 클러스터의 이동
Fig. 9. Movement of a cluster.

네번째 비용함수 $C_4$는 각 채널에서 네트의 사용률이 채널의 용량을 초과하여 사용하지 않도록 하는 비용함수이다. 이를 위하여 그래프 $G_T$의 각 채널 $e_i$를 위하여 해당하는 채널의 용량인 $C(e_i)$를 유지한다. 그리고 현재 해에서 연결을 위해 채널 $e_i$를 사용하는 네트의 수를 $U(e_i)$라고 하면 다섯번째 비용함수는 다음과 같다.

$$C_4 = \sum_{e_i \in E} Z_i^2, \quad Z_i = \begin{cases} U(e_i) - C(e_i) & \text{if } U(e_i) > C(e_i), \\ 0 & otherwise \end{cases}$$

즉, 만일 채널을 사용하는 네트의 수가 용량을 넘을 때에는 그 초과분의 제곱을 비용으로 한다. 비용함수 $C_3$와 $C_4$에서는 이차 함수를 이용하여 주어진 제한을 만족하는 해에 빨리 도달할 수 있도록 한다.

다섯번째 비용 함수 $C_5$는 각 네트가 사용하는 채널 수에 대한 비용 함수이다. 이 비용함수는 각 네트가 사용하는 채널의 수를 줄여서 각 채널에서 네트의 사용률을 줄이기 위한 것이다. 네트 $n_i$가 사용하는 채널 수를 $H(n_i)$라고 할때 비용 함수는 다음과 같다.

$$C_5 = \sum_{n_i \in N} H(n_i)$$

그러므로, 모든 비용 함수를 포함한 전체 비용은 다음과 같다.

$$Cost = W_1*C_1 + W_2*C_2 + W_3*C_3 + W_4*C_4 + W_5*C_5$$

여기서 $W_1, W_2, W_3, W_4, W_5$들은 각 비용함수를 위한 가중치이다.

3. 냉각 스케줄과 중단 조건

분할의 처음 단계에서는 원하는 크기의 클러스터들을 선택하고 초기 해를 얻기 위하여 초기 분할을 수행한다. 클러스터를 선택할 때에는 블럭의 크기 보다는 작은 크기이면서 루트에 가까이 있는 것을 선택한다. 초기 분할 방법은 선택한 클러스터들을 랜덤(random)하게 임의의 블럭에 넣는다. 초기 온도는 90 %의 확률로 초기에 대부분의 이웃해를 받아들일 수 있도록 높은 온도에서 시작하며 초기 분할된 결과로 부터 이웃해를 구한다. 온도는 0.95의 비율로 내리며 한 온도에서 시도되는 이동의 횟수는 현재 사용하고 있는 클러스터의 수가 $|M|$일 때 $35|M|$이다.

한 번 선택한 클러스터들에 대해서는 비용이 감소하는 동안에는 계속해서 다음 해를 구하며 만일 해의 비용이 몇 번의 온도 변화 동안에도 더 이상의 감소가 없거나 대부분의 해가 받아들여지지 않는 경우가 반복되면 현재 사용하고 있는 클러스터들의 클러스터 트리에서 자식에 해당하는 클러스터들로 교체하여 사용한다. 이 과정에서 자식에 해당하는 클러스터들은 그들의 부모에 해당하는 클러스터들이 있던 블럭에 놓이고 이로부터 다음 해를 구하는 작업을 진행한다. 그러나 분할의 대상이 MFFC 클러스터들이면서 해의 비용이 몇 번의 온도 변화 동안에도 그 이상의 감소가 없고 대부분의 해가 받아들여지지 않으며 온도가 정해진 최하 온도까지 낮아진 경우에는 수행을 중단한다.

## V. 실험결과

본 장에서는 본 연구에서 제안한 분할 방법을 여러 회로에 대하여 적용한 실험 결과를 보인다. 제안한 분할 방법은 C 언어를 사용하여 구현하였으며, 실험은 삼보 워크스테이션 SDT-820에서 수행하였다. 실험은 대상 회로를 여덟 개와 열여섯 개의 블럭으로 분할하는 것과 AXB-AP4 보드의 위상과 여기에서 사용할 수 있는 FPGA의 제약 조건을 만족하도록 회로를 분할하는 것으로 나누어 수행하였으며, 비교를 위하여 기존의 K-way 분할 방법을 위상 기반 분할 문제에 적합하도록 수정한 분할 방법을 개발하였다. 수정한 K-way 분할 방법에서는 채널을 사용하는 네트의 수가 채널의 용량을 넘는지를 이득(gain) 계산시 포함시켰다[29]. 회로를 여덟 개와 열여섯 개의 블럭으로 분할하는 것은 제안한 분할 방법을 기존의 분할 방법과 비교하기 위한 것이고, AXB-AP4 보드를 이용하여 분할하는 것은 실제로 시제품을 제작할 경우에 제안한 분할 방법이 효과가 있는지를 보이기 위함이다.

모든 실험에서 분할을 위하여 대상이 되는 위상그래프는 그림 1의 것을 채널의 용량이나 사용하는 블럭의 수를 조정해가며 사용하였다. 예를 들어 여덟 개의 블럭으로 분할할 경우에는 그림 1에 각 스위치 칩과 인접한 네 개의 FPGA 칩들을 두 개로 줄여 사용하였다. 또 여덟 개와 열여섯 개의 블럭으로 분할할 경우에는 각 블럭의 크기를 각 블럭에 대상 회로의 크기가 비교적 균등하게 나뉘어 들어갈 수 있도록 제한하였다.

실험에 사용된 데이타는 ISCAS 벤치마크(benchmark) 회로 중에서 c6288, s5378, s9234, s13207, s15850, s35932, s38584와 설계자가 디자인한 회로 idct8와 comet들이며 이들 정보를 표 1에 보인다.

표 1. 실험용 데이타들(*로 마크한 회로들은 I/O의 갯수가 커서 AXB-AP4 보드에 장착될 수 없어서 I/O 수를 수정함)

Table 1. Experimental data(Data with * have too many I/Os to partition to AXB-AP4).

| 회로 | Primary I/O수 | 예측 크기 |
|---|---|---|
| c6288 | 64 | 1488 |
| s5378 | 87 | 965 |
| s9234 | 44 | 1176 |
| s13207 | 155 | 1931 |
| s15850 | 104 | 1823 |
| s35832* | 145 | 3329 |
| s38584* | 154 | 5555 |
| idct8 | 129 | 2039 |
| comet | 20 | 826 |

표 1에서 Primary I/O는 회로의 primary 입출력 갯수를 의미하고, 예측 크기는 본 연구에서 제안한 예측 방법으로 각 MFFC의 LUT 갯수를 예측하여 이들

을 모두 합한 것이다. 표 1의 *로 나타낸 회로들은 AXB-AP4의 primary 입출력 제한을 넘는 회로이므로 primary 출력을 OR시키는 수정작업을 하였으며 표에 나타난 I/O 수는 수정작업 후에 회로가 사용하는 I/O 수이다. ISCAS 벤치마크들은 .xnf라는 확장자를 갖는 화일들을 사용하였으며 설계자가 디자인한 회로들은 Synopsys 툴에서 FPGA에 적합하도록 컴파일하고 .sxnf라는 확장자를 갖는 XNF 형태의 화일로 출력한 것을 사용하였다.

표 2에 AXB-AP4에서 사용가능한 FPGA 중에서 본 실험에서 사용한 FPGA 칩을 보인다[27]. 로직 구현을 위한 FPGA로는 XC4010을 사용하였으며, I/O FPGA 용으로는 XC4005를 사용하였다. LUT 수와 IOB 수는 각 FPGA 칩에서 사용 가능한 LUT와 IOB의 수이다.

표 2. 실험을 위하여 사용한 FPGA
Table 2. FPGAs used for the test.

| Device | LUT수 | IOB수 |
|---|---|---|
| XC4010 | 800 | 160 |
| XC4005 | 392 | 112 |

모든 입력된 회로에 대해서 MFFC 클러스터링과 LRC 클러스터링의 과정을 수행하고 초기 분할을 수행한 후에 제안한 분할 방법을 수행한다. LRC 클러스터링은 클러스터를 생성할 때 클러스터의 크기를 제한할 수 있으므로 클러스터가 가질 수 있는 최대 LUT의 수와 최소 LUT의 수를 지정하여 수행한다. 회로가 매우 작은 경우에는 LRC 클러스터링을 생략하였다. 임의의 회로에 대하여 LRC 클러스터링을 반복하는 횟수와 그 때의 클러스터의 크기는 실험에 의하여 가장 적합한 것을 사용하였다. 실험의 대부분의 회로에서 LRC 클러스터링은 2~4 회 사용하였으며, 마지막 클러스터링 후에 클러스터의 크기는 40~60 LUT 정도가 되도록 하였다. 클러스터링 후에 클러스터들은 랜덤하게 임의의 블럭에 넣고 이를 초기 분할로 하여 시뮬레이티드 어닐링 분할 방법을 진행하였다.

표 3과 4에 여덟 개의 블럭과 열여섯 개의 블럭으로 분할한 두 분할 방법을 비교한다. 표에 MAX는 각 채널을 사용하는 네트의 수의 최대값이며, Chls는 네트들이 사용하는 채널 수의 합이며, $B_{max}$는 각 블럭에 속한 회로 크기의 최대값을 나타낸다. 각 요소에서 K-way는 수정한 K-way 분할 방법의 결과를 Sim은 시뮬레이티드 어닐링 분할 방법의 결과를 나타낸다. 두 표에서 시뮬레이티드 어닐링 기법에 의한 분할 방법이 수정한 K-way 분할 방법보다 각 채널에서의 네트의 수가 크게 적음을 알 수 있다.

표 3. 두 분할 결과의 비교(여덟 블록)
Table 3. Comparision of two partition result (eight blocks).

| 회로 | Max | | Chls | | $B_{max}$ | |
|---|---|---|---|---|---|---|
| | K-way | Sim | K-way | Sim | K-way | Sim |
| c6288 | 85 | 71 | 602 | 743 | 215 | 213 |
| s5378 | 108 | 91 | 672 | 761 | 135 | 136 |
| s9234 | 99 | 80 | 627 | 727 | 171 | 170 |
| s13207 | 130 | 109 | 930 | 1168 | 286 | 286 |
| s15850 | 114 | 93 | 768 | 934 | 261 | 260 |
| s35832 | 150 | 131 | 945 | 1272 | 441 | 440 |
| s38584 | 299 | 199 | 1627 | 1917 | 801 | 799 |
| idct8 | 132 | 124 | 818 | 1114 | 277 | 278 |
| comet | 104 | 88 | 532 | 775 | 117 | 118 |

표 4. 두 분할 결과의 비교(열여섯 블록)
Table 4. Comparision of two partition result (sixteen blocks).

| 회로 | Max | | Chls | | $B_{max}$ | |
|---|---|---|---|---|---|---|
| | K-way | Sim | K-way | Sim | K-way | Sim |
| c6288 | 66 | 56 | 739 | 952 | 117 | 110 |
| s5378 | 79 | 70 | 825 | 982 | 68 | 68 |
| s9234 | 75 | 69 | 768 | 1022 | 81 | 82 |
| s13207 | 111 | 101 | 1109 | 1408 | 144 | 146 |
| s15850 | 86 | 70 | 904 | 1219 | 135 | 136 |
| s35832 | 100 | 99 | 1260 | 1543 | 244 | 246 |
| s38584 | 197 | 138 | 1920 | 2281 | 378 | 378 |
| idct8 | 147 | 90 | 1153 | 1398 | 153 | 154 |
| comet | 70 | 68 | 799 | 994 | 54 | 54 |

표 5. AXB-AP4로의 분할(수정한 K-way 분할방법을 이용한 분할)
Table 5. Partition into AXB-AP4(partition using the modified K-way partition method).

| 회로 | 블럭갯수 | $B_{max}$ | $B_{min}$ | Chls | $C_{max}$ | $C_{min}$ | Time(s) |
|---|---|---|---|---|---|---|---|
| c6288 | (2,2) | 96% | 88% | 286 | 77% | 71% | 51 |
| s5378 | (2,2) | 89% | 32% | 315 | 95% | 58% | 41 |
| s9234 | (2,1) | 95% | 52% | 197 | 58% | 46% | 38 |
| s13207 | (10,4) | 26% | 17% | 1025 | 95% | 26% | 747 |
| s15850 | (3,3) | 100% | 28% | 469 | 100% | 50% | 194 |
| s35832 | (7,3) | 68% | 50% | 887 | 99% | 57% | 798 |
| idct8 | (7,3) | 49% | 10% | 758 | 93% | 17% | 466 |
| comet | (2,1) | 100% | 3% | 82 | 40% | 21% | 40 |

표 6. AXB-AP4로의 분할(시뮬레이티드 어닐링 기법을 이용한 분할)

Table 6. Partition into AXB-AP4(partition using the simulated annealing method).

| 회로 | 블럭갯수 | $B_{max}$ | $B_{min}$ | Chls | $C_{max}$ | $C_{min}$ | Time(s) |
|---|---|---|---|---|---|---|---|
| c6288 | (2,2) | 99% | 85% | 283 | 69% | 63% | 720 |
| s5378 | (2,2) | 67% | 54% | 349 | 73% | 69% | 312 |
| s9234 | (2,1) | 82% | 66% | 183 | 52% | 43% | 422 |
| s13207 | (9,4) | 40% | 18% | 1164 | 82% | 29% | 1274 |
| s15850 | (3,3) | 100% | 34% | 526 | 82% | 77% | 1312 |
| s35832 | (7,3) | 92% | 16% | 885 | 64% | 35% | 2031 |
| s38584 | (15,4) | 81% | 11% | 2348 | 100% | 34% | 8589 |
| idct8 | (4,3) | 84% | 35% | 711 | 85% | 76% | 717 |
| comet | (2,1) | 87% | 16% | 106 | 35% | 22% | 269 |

표 5와 6에 AXB-AP4 보드에서의 수정한 K-way 방법에 의한 분할 결과와 시뮬레이티드 어닐링 방법에 의한 분할 방법의 결과를 보인다. 블럭갯수는 사용한 블럭 수와 I/O 블럭 수를 의미하고 이를 (블럭 수, I/O 블럭 수)로 표현하며, $B_{max}$는 가장 사용률이 높은 블럭, $B_{min}$은 가장 사용률이 낮은 블럭의 사용률을 나타낸다. $C_{max}$는 FPGA와 스위치 칩을 연결하기 위해서 사용하는 채널들 중에서 가장 사용률이 높은 것, $C_{min}$은 이 채널들 중에서 가장 사용률이 낮은 것의 사용률을 나타낸다. Chls는 앞에서 정의한 것과 같으며 Time(s)는 분할 방법의 수행시간을 나타낸다.

표에서 보이듯이 제안한 두 분할 방법이 모두 실험에 이용한 모든 회로에 대하여 AXB-AP4 보드의 채널과 블럭의 용량 제한을 만족하며 분할함을 알 수 있다. 그리고 s13207, s35932, s38584, idct8을 제외한 모든 회로에 대하여 최적의 블럭 수와 I/O 블럭수를 사용하여 분할되었다. s13207, s35932, s38584, idct8은 회로 자체가 많은 네트로 복잡하게 연결되어 있기 때문에 채널을 사용하는 네트의 수가 많아서 많은 블럭으로 분산되었으며 s38584의 경우에는 수정한 K-way 분할 방법으로는 채널 용량의 제한을 만족하는 해를 얻지 못하였다.

대부분의 실험 결과에서 시뮬레이티드 어닐링 기법을 이용한 분할 방법에서 채널 사용률이 훨씬 적었으며 AXB-AP4로의 분할에서는 더 적은 수의 블럭 수를 사용하여 분할되었다. 그런데 시뮬레이티드 어닐링 기법을 이용한 분할 방법의 경우에는 단지 채널을 사용하는 네트의 수를 최소화하기 보다는 빠른 수행시간에 블럭의 수를 최소화하면서 각 채널 사용률을 최소화하려고 하였기 때문에 네트가 사용하는 채널 수가 큰 경우도 있었다. 이상의 실험을 통하여 논문에서 제안하는 분할 방법이 수정한 K-way 분할 방법보다 더 효율적인 분할을 수행하며, AXB-AP4 보드에서는 각 채널과 블럭의 용량 제한 조건을 만족하며 회로를 분할할 수 있음을 입증하였다. 이때 수행에 필요한 시간은 큰 회로에서 최소 1.5 배, 적은 회로에서는 최대 15 배로 나타났다. 비록 적은 회로의 경우 수행시간에서 많은 차이를 보이지만 이는 시뮬레이티드 어닐링 분할 방법을 사용하여 큰 회로를 빠른 시간에 수행하기 위하여 노력하였기 때문이며 전체적으로 비교적 양호하였다.

## VI. 결 론

본 논문에서는 빠른 시제품 제작을 위하여 회로를 위상이 정해진 FPCB 상의 다중의 칩으로 분할하는 새로운 분할 문제인 위상 기반 분할 문제를 정의하였다. 위상 기반 분할 문제는 칩간의 채널을 고려하기 때문에 기존의 분할 문제와는 그 성격이 상이하여 기존의 핀 수만을 최소화하는 분할 방법으로는 최적의 결과를 얻기 곤란하였다. 따라서 본 논문에서는 이를 해결하기 위한 분할 방법으로 시뮬레이티드 어닐링 기법을 이용한 분할 방법을 제안하였다. 또한, 분할 방법을 효율적으로 사용하기 위하여 다단계의 트리 클러스터링 방법을 제안하였다. 실험결과는 주어진 FPCB의 제한을 모두 만족하였고 기존의 결과와 비교하여 거의 대분분의 회로에 대하여 채널 사용률이 크게 적고 더 적은 수의 블럭으로 분할 되었으며 큰 회로에 대해서는 수정한 K-way 분할 방법의 약 두 배 정도의 시간에 분할이 가능함을 보였다. 따라서 본 논문의 분할 방법은 FPCB의 시제품 제작을 위한 회로 분할에 유용하게 사용될 수 있을 것이다. 앞으로 성능을 고려한 분할 방법에 관한 연구가 요구된다.

## 참 고 문 헌

[1] C.J. Alpert and A.B. Kahng, "Multi-Way Partitioning Via Spacefilling Curves and Dynamic Programming", Proc. of 31th Design Automation Conference, pp. 652-

657, 1994.

[2] C.J. Alpert and A.B. Kahng, "Geometric Embeddings for Faster and Better Multi-Way Netlist Partitioning", Proc. of 30th Design Automation Conference, pp. 743-748, 1993.

[3] C.J. Alpert and A.B. Kahng, "A General Framework for Vertex Orderings, With Applications to Netlist Clustering", International Conference on CAD, pp. 63-67, 1994.

[4] P.K. Chan, M.D.F. Schlag, J.Y. Zien, "Spectral K-Way Ratio-Cut Partitioning and Clustering", Proc. of 30th Design Automation Conference, pp. 749-754, 1993.

[5] N.C. Chou, C.K. Cheng. W.J. Dai and R. Lindelof, "Circuit Partitioning for Huge Logic Emulation Systems", Proc. of 31th Design Automation Conference, pp. 244-249, 1994.

[6] J. Cong, Y.Ding, "On Area Depth Trade-off in LUT-Based FPGA Technology Mapping", Proc. 30th ACM/IEEE Design Automation Conference, pp. 213-218, 1993.

[7] J. Cong, L. Hagen, A.B. Kahng, "Net Partitions Yield Better Module Partitions", Proc. of 29th Design Automation Conference, pp. 47-52, 1992.

[8] J. Cong , Z. Li and R. Bagrodia, "Acyclic Multi-Way Partitioning of Boolean Net works", Proc. of 31th Design Automation Conference, pp. 670-675, 1994.

[9] C.M. Fiduccia and R.M. Mattheyeses. "A Linear-Time Heuristic for Improving Network Partitions", Proc. of 19th Design Automation Conference, pp. 175-181, 1982.

[10] L.K. Grover, "A New Simulated Annealing Algorithm for Standard Cell Placement", International Conference on CAD, pp. 378-380, 1986.

[11] B.M. Kerninghan and S. Lin. "An Efficient Heuristic Procedure for Partitioning Graph", Bell System Technical Journal, vol 49, no 2, pp. 297-307, Feb. 1970.

[12] S. Kirkpatrick, C.D. Gelatt, M.P. Vecci, "Optimization by Simulated Annealing", Science, vol 229. no 4598. pp. 671-680, May. 1983.

[13] B. Krishnamurthy, "An Improved Min-Cut Algorithm for Partitioning VLSI Networks", IEEE Trans. on Computers. vol 33. pp. 438-446, May, 1984.

[14] R. Kuznar, F. Brglez and K. Kozminski, "Cost Minimization of Partitioning into Multiple Device", Proc. of 30th Design Automation Conference, pp. 315-320, 1993.

[15] R. Kuznar, F. Brglez and B. Zajc, "Multi-way Netlist Partitioning into Heterogeneous FPGAs and Minimization of Total Device Cost and Interconnect", Proc. of 31th Design Automation Conference, pp. 238-243, 1994.

[16] C. Lee, "An Algorithm for Path Connections and Its Applications", IEEE Transactions on Electronic Computers, VEC-10, pp. 346-365, Sept. 1961.

[17] K. Roy and C. Sechen, "A Timing Driven N-way Chip and Multi-Chip Partitioner", International Conference on CAD, pp. 240-247, 1993.

[18] L.A. Sanchis. "Multiple-Way Network Partitioning", IEEE Trans. on Computers. vol 38. no 1. Jan, 1989.

[19] G. Saucier, D. Brasen, J.P. Hiol. "Partitioning With Cone Structures", International Conference on CAD, pp. 236-239, 1993.

[20] C. Sechen and A. Sangiovanni-Vincentelli, "Timberwolf 3.2 : A New Standard Cell Placement and Global Routing Package", Proc. of 30th Design Automation Conference, pp. 432- 439, 1986.

[21] M. Shih and E.S. Kuh, "Quadratic Boolean Programming for Performance-Driven System Partitioning", Proc. of 30th Design Automation Conference, pp. 761-765, 1993.

[22] P. R. Suaris and G. Kedem, "An Algorithm for Quadrisection and Its Application to Standard Cell Placement", IEEE Trans. on Circuits and Systems, vol. 35, no. 3, pp. 294-303, March, 1988.

[23] Y.C. Wei and C.K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning", International Conference on CAD, pp. 298-301, 1989.
[24] D.F. Wong, H.W. Learg, C.L. Liu, ed, *Simulated Annealing for VLSI Design*, Kluwer Academic Publishers, 1988.
[25] N.S. Woo and J. Kim, "An Efficient Method of Partitioning Circuits for Multiple FPGA Implementation", Proc. of 30th Design Automation Conference, pp. 202-207, 1993.
[26] C.W. Yeh and C.K. Cheng, "A General Purpose Multiple Way Partitioning Algorithm", Proc. of 28th Design Automation Conference, pp. 421-426, 1991.
[27] Xilinx, Inc., The Programmable Gate Array Data Book, Xilinx, San Jose, 1992.
[28] Aptix, Inc., System Data Book, Aptix, 1993.
[29] 임 종석 외, "ASIC Prototyping을 위한 회로 다중 분할에 관한 연구", 한국전자통신연구소, 최종연구보고서, 1995

## 저 자 소 개

**崔 然 景(正會員)**
1991년 2월 서강대학교 전자계산학과 학사. 1993년 2월 서강대학교 전자계산학과 석사. 현재 서강대학교 전자계산학과 박사과정. 경민전문대 전자계산과 전임강사. 주관심분야는 VLSI 설계.

**林 鐘 錫(正會員)** 서강대학교 전자계산학과 부교수