論文95-32A-4-10

여러 개의 FPGA 칩을 위한 대규모 회로의 분할

(Partitioning of Large-Circuits for Multiple FPGAs)

金 衷 希*, 申 鉉 哲**

(Chung Hee Kim, and Hyun Chul Shin)

요 약

여러 개의 FPGA 칩을 이용하여 대규모 회로를 구현하기 위한 분할 알고리즘을 개발하였다. 기존의 분할 알고리즘은 크기의 제약조건을 만족시키는 범위 내에서 최소의 연결선을 가지도록 하는 것이 목적이지만, FPGA 칩을 위한 분할은 조합논리 회로의 크기, 메모리의 크기, 입.출력 핀의 수 등 여러 가지 새로운 제약조건을 만족시켜야 한다. 이는 분할 후 각각의 분할된 부분회로를 하나의 칩에 구현 하기 위한 것이다. 여러 가지 제약조건 하에서 좋은 분할을 얻기 위하여, 대규모 회로의 분할은 전체적인 최적화를 위한 초기의 분할과 제약조건을 만족시키기 위한 반복적인 분할의 개선 단계로 나누어 수행하였다. MCNC 벤취마크 예제에 대하여 실험한 결과, 본 분할 방법은 기존의 방법에 비하여 우수한 결과를 보여주었다.

Abstract

A new partitioning algorithm has been developed to implement a large circuit by using multiple field programmable gate array (FPGA) chips. While the conventional partitioning is to minimize the number of nets cut under size constraints, partitioning for multiple FPGAs has several additional constraints so that each partitioned subcircuit can be implemented in a FPGA chip. To obtain satisfactory results under the constraints, the partitioning is performed in two steps which are the initial partitioning for global optimization and the iterative partitioning improvements for constraint satisfaction. Experimental results using the MCNC benchmark examples show that our partition method produces better results than those of other recent approaches on the average.

I. 서 론

FPGA (Field Programmable Gate Array)를 이용한 설계는 미리 설계된 칩에 프로그램을 함으로써 원하는 기능의 회로를 구현하므로, IC 설계 제작에 비하여 소요되는 비용과 시간을 절감할 수 있으며, 원하는 회로에 대한 신속한 실험 제작 (prototyping)이 가능하기 때문에, 널리 사용되고 있다. FPGA 칩은 조합논리 회로와 메모리를 구현할 수 있는 CLB (Configurable Logic Block)와 입.출력을 위한 IOB (Input/Output Block), 그리고, CLB들과 IOB들을 서로 연결하여 회로를 구현하기 위한 배선 구조로 이루어져 있다.

* 學生會員, ** 正會員, 漢陽大學校 電子工學科

(Dept. of Elec. Eng., Hanyang Univ.)

※ 이 논문은 1994년도 교육부 학술 연구 조성비에 의하여 연구되었음.

接受日字 : 1994年 8月 6日

구현하고자 하는 회로의 규모가 작아서 하나의 FPGA 칩 (chip)으로 구현 가능할 때에는 간단히 구현이 가능하다. 그러나, 회로의 규모가 커서 하나의 FPGA 칩을 사용하여 전체 회로를 구현할 수 없을 경우에는, 회로를 적절히 분할하여 여러 개의 FPGA 칩으로 구현하여야 한다. 여러 개의 FPGA 칩을 이용하여 규모가 큰 회로를 구현하기 위해서는, FPGA 구조에 적합한 여러 부분 회로로의 분할이 필요하다. 분할된 각 부분 회로는 하나의 주어진 FPGA 칩 내에서 구현될 수 있도록 여러 가지 제약조건을 만족시켜야 한다. 또한 칩간의 배선을 최소화 하기 위하여 부분 회로 사이의 연결도를 최소화하여야 한다. 칩 간의 배선이 적어질수록 설계의 복잡도를 줄일 수 있고, 지연시간을 감소시켜 성능을 개선시킬 수 있다. 분할의 제약조건은 FPGA 칩이 가지는 CLB 수와 핀의 수 등이다. 특히 핀의 수가 고정되어 있고, 많은 경우에 핀의 수가 가장 심각한 제약조건이 되므로, 핀의 수를 제한하는 분할이 필요하다.

대부분의 기존의 분할 방법[1,2,3]들은 입력된 회로를 부분집합의 크기의 제약조건만을 만족시키며 연결도에 의한 비용을 최소화 하는 것을 주된 목적으로 하였다. 대표적인 예로 [2]의 방법은 전체 그래프를 두 부분으로 분할하는 알고리즘을 제안하였고, 이 방법을 효율적으로 개선한 분할 알고리즘들[1,3,12]이 개발되었다. 최근에는 클러스터 (cluster)를 이용한 분할 방법들[4,5]이 개발되어 우수한 결과를 보여 주고 있다.

그러나, 이러한 기존의 분할 방법들은 FPGA 칩의 여러 제약조건들을 효과적으로 다루기 어렵기 때문에, 그대로 FPGA 칩을 위한 분할 방법으로 사용하기는 곤란하다. 따라서, FPGA 칩을 위한 몇 가지 분할 방법이 제안되었다. 먼저 Bottom-up 방법에 의한 FPGA 분할 방법 [6]이 제안되었다. 이 방법은 한 번에 하나의 노드 (node) 또는 함수를 디바이스 (device)에 할당하는 방법을 반복하여 분할을 수행하였다. 그러나, 이 방법은 bottom-up 방법으로 구현되었기 때문에, 큰 회로의 분할시 전체적인 최적화에서 멀어질 수 있다는 단점이 있다.

[7]의 방법은 여러 가지 종류의 FPGA 칩들을 위한 분할 방법이다. 분할은 칩을 선택하고, 회로를 두 개의 그룹으로 분할하는 것을 반복하여 수행한다. 두 개의 그룹으로 분할하는 경우, 하나의 그룹은 반드시 선택된 칩을 이용하여 구현 가능하도록 핀과 CLB 수의 제약조건을 만족하여야 한다. 분할 후, 제약조건이 만족되는 그룹을 하나의 칩에 할당하고, 나머지 그룹은 필요한 경우 분할을 반복하여 수행한다. 회로의 분할은[11] 방법을 개선하여 사용하였다. 그러나, 이 방법에서도 전체 회로의 일부분씩을 떼어 내어 분할하므로, 전체적인 최적화에서 멀어질 수 있다. [7]의 방법을 개선한 방법[8]도 제안되었다. 새로운 방법은 칩 간의 배선을 용이하게 하기 위하여 연결선의 최소화에 중점을 두었으며, 또한 논리의 복제를 이용하여 회로의 구현에 필요한 칩의 비용을 줄였다.

[9]의 방법은 대규모 회로의 에뮬레이션 (emulation) 시스템을 위한 분할 방법이다. 대규모 회로의 분할을 위하여, bottom-up 클러스터링과 top-down 분할 방법을 이용하였다. 먼저 ratio-cut 클러스터링 방법을 사용하여 회로의 규모를 줄인 후, 필요한 FPGA 칩의 수를 줄이기 위한 set covering 알고리즘을 수행하였다.

본 논문에서 제안한 새로운 분할 방법은 클러스터를 이용한 분할 기법[5]을 FPGA 칩을 이용한 분할에 적합하게 개선한 방법이다. [5]의 방법은 초기 분할에 중점을 두어 좋은 초기 분할을 얻도록 하였으며, 효율적인 분할을 위하여 두 단계의 분할 방법을 이용하였다. 먼저 서로 밀접하게 연결되어진 셀들을 클러스터로 만든다. 클러스터를 구성한 후, 이들을 각각 몇 가지의 서로 다른 임의의 초기 분할로부터 분할하여 그중에서 가장 우수한 결과를 선택한다. 가장 우수한 클러스터의 분할 결과를 셀 단계에서의 초기 분할로 이용하여 최적에 가까운 결과를 얻도록 하였다. 이때 클러스터의 분할과 셀 단계의 최종 분할은 [1]의 방법을 개선하여 사용하였다. [1]의 방법이 크기의 제약조건을 미리 결정하여 주고 그 범위 내에서 분할하는데 비하여, 크기의 제약조건을 동적으로 조정하여 결과를 개선하도록 하였다. 이러한 분할 방법은 실험결과 [1]의 방법에 비하여 평균 60% 이상 우수한 결과를 보여 주었다.

FPGA 칩을 이용한 본 분할 알고리즘은 전체적인 최적화를 위한 분할과 여러 제약조건을 만족시키기 위한 반복적인 분할의 개선으로 이루어진다. 전체적인 최적화를 위한 분할 방법은 기존의 방법 [5]를 개선하여 사용하였다. 먼저, 제약조건을 만족시키기 위하여 비용 함수에 이를 고려하였다. 또한 [5]의 방법에서는 크기 조건이 만족하는 범위 내에서 hill climbing을 위하여 음의 이득을 가지는 셀의 이동을 모두 허용하지만, 본 방법은 각 단계에서 이득이 최대가 되는 곳까지만 셀들을 이동하여 결과를 개선하였다.

제약조건을 만족시키기 위한 분할의 개선 단계에서는, 각 반복 단계마다 만족되지 않은 제약조건이 있을 경우, 그 조건에 대한 가중치 (weight)를 증가하여 많은 비용을 가지도록 함으로써 다음 단계에서 우선적으

로 그 제약조건이 만족되도록 하였다.

제 II장에서는 클러스터를 이용한 기존의 분할 방법 [5]에 대하여 간략히 기술하고, 제 III장에서는 FPGA 칩을 이용한 새로운 분할 알고리즘에 대하여 자세히 기술한다. 그리고, IV장에서는 실험결과에 대하여 기술하고, 끝으로 V장에서 결론을 기술한다.

II. 클러스터를 이용한 분할 방법

본 분할 방법에서는 FPGA의 구조에 맞게 효과적으로 제약조건을 만족시킬 수 있도록 기존의 분할 방법을 개선하였다. 본 장에서는 먼저 클러스터를 이용한 기존의 분할 방법 [5]에 대하여 간단히 기술하며, FPGA를 위한 분할에 대하여는 다음장에서 기술한다.

1. 클러스터 생성 방법

클러스터를 만드는 목적은 서로 밀접하게 연결된 셀들을 하나의 클러스터로 만들어 처리함으로써, 전체 수행 시간을 단축시키고 분할 결과를 개선하는데 있다. 두 셀 또는 클러스터 간의 친밀도는 그들에 연결된 모든 네트의 웨이트 (weight)의 합, 공통된 네트의 웨이트의 합, 두 셀의 크기 등의 함수로 정의되며, 친밀도를 이용하여 클러스터링 (clustering) 여부를 결정한다. 클러스터링을 위하여 반복적인 top-down ratio-cut 분할[13]을 이용하는 방법[4]도 제안되었으나 이는 여러 번의 분할을 수행하여야 하므로 복잡하다.

본 방법에서 처음에는 각각의 셀이 하나의 클러스터가 된다. 그리고 모든 클러스터 쌍에 대하여 클러스터 간의 친밀도를 계산하고, 친밀도가 가장 큰 클러스터 쌍을 통합하여 하나의 클러스터를 형성한다. 통합된 클러스터에 연관된 친밀도는 통합 후 다시 계산한다. 친밀도가 가장 큰 클러스터 쌍을 찾아서 이들을 통합하여 하나의 클러스터를 형성하는 과정을 필요한 회수만큼 반복하여 클러스터링을 완성한다.

두 클러스터 C와 D의 친밀도는 식 (1)을 이용하여 계산하였다.

$$\text{closeness}(C,D) = \alpha * (\text{num\_cnet}(C,D) / \text{MIN}(\text{num\_net}(C), \text{num\_net}(D)) - \beta * (\text{cl\_size}(C,D) / \text{avg cell size}) \quad (1)$$

여기서 MIN(a,b)는 a와 b중에 작은 값을 나타내며, num_cnet(C,D) 는 C와 D에 공통으로 연결된 네트의 수이고 num_net(C)는 클러스터 C에서 다른 클러스터와 연결된 네트의 수를 나타낸다. cl_size (C,D) 는 C와 D를 결합하여 하나의 새로운 클러스터로 만들었을 때의 클러스터 크기이다. avg_cell_size 는 클러스터 생성 전의 전체 셀의 평균 크기이다.

위 식의 첫 번째 항은 C와 D간에 얼마나 서로 밀접하게 연결되어 있는지를 나타내고, 두 번째 항은 클러스터의 크기를 조절하기 위한 항으로 하나의 클러스터의 크기가 너무 커지는 것을 막기 위한 것이다.

2. 분할 방법

분할은 초기 분할과 이 초기 분할을 개선시키는 반복적인 개선 단계로 나눌 수 있다. 임의의 초기 분할을 이용한 방법들은 초기 분할에 의해 최종 결과가 크게 달라질 수 있다는 단점이 있다. 이러한 초기 분할의 영향을 줄이고 다양한 분할 문제에 대하여 일관성 있게 우수한 결과를 얻기 위하여, 클러스터 단계에서 여러 가지 초기 분할에 대하여 평가하여 좋은 클러스터의 분할을 찾도록 하였다. 클러스터의 분할을 수행한 후, 이를 이용하여 셀의 분할을 실시하며, 셀의 분할은 한 번의 분할로 완료된다. 셀을 클러스터로 만들어 처리하면, 셀의 수에 비하여 적은 수의 클러스터를 분할하게 되고, 네트의 수도 클러스터 간을 연결하는 네트만을 고려하므로, 복잡도가 크게 줄어든다. 이 때 전체 클러스터의 수는 초기의 셀의 수의 10% 정도로 실험적으로 결정하였다.

각 단계에서의 분할 알고리즘은 [1]의 분할 방법을 개선한 방법인 점진적으로 크기 제약조건을 만족시키는 방법을 사용하였다. 이 방법은 [1]의 linear한 복잡도를 가지는 특성을 이용하면서, 단계적으로 크기의 균형을 맞추어 주는 알고리즘이다. 즉 [1]의 방법은 크기의 조건에 벗어나는 셀의 이동을 허용하지 않으나, 본 알고리즘은 크기의 제약조건을 동적으로 조정한다. 초기에는 크기의 차이가 크게 벗어나는 것도 허용하며, iteration이 반복됨에 따라 크기 조건에서 벗어나는 폭을 감소 시켜서, 마지막 단계에서는 나누어진 두 그룹의 크기의 제약조건이 모두 만족되도록 하였다. 초기 단계에서 셀의 이동을 크게 허용하면 부분적 최적화 (local optimum)에서 벗어나 전체적인 최적화 (global optimum)를 얻을 수 있는 가능성이 커져서 최적에 가까운 결과를 얻을 수 있다.

III. FPGA 칩을 위한 분할

본 장에서는 FPGA 칩의 구조에 적합하게 회로를 분할하는 알고리즘에 관하여 기술한다. 대부분의 경우에 설계자는 같은 종류의 FPGA 칩들을 사용하여 시스템을 구현하므로, 본 분할 알고리즘은 같은 종류의 FPGA 칩을 위한 분할을 수행하며, 주어진 칩의 수에 맞게 분할을 수행하였다. 분할은 회로를 구현하는데 필요한 칩들의 전체 비용을 최소화 하도록 수행하였다.

알고리즘의 구성은 전체적인 최적화를 위한 클러스터를 이용한 초기 분할 과 만족되지 않은 제약조건을 만족시켜 주기 위한 반복적인 분할의 개선 단계로 나누어진다. FPGA를 위한 초기 분할은 전체적으로 우수한 분할을 얻기 위하여 II장에서 기술한 방법을 개선하여 사용하였고, 분할의 개선 단계에서는 만족되지 않은 제약조건을 만족시켜 주기 위하여 가중치 (weight)를 점진적으로 조정하는 방법을 사용하였다.

알고리즘 1은 전체 알고리즘을 나타낸다.

```
Algorithm 1: Partitioning for FPGAs
input_data();
/* Clustering-Based Initial Partitioning */
make_cluster();
several_cluster_partitioning();
flatten_clusters();
cell_level_partitioning();
/* Partition Improvement */
partition_improvement();
output();
```

1. FPGA 칩을 위한 분할에서의 제약조건

FPGA 칩을 위한 분할에서는 분할된 그룹이 지정된 칩을 사용하여 구현 가능하도록 여러 가지 제약 조건을 만족시켜야 한다. 분할시 고려하여야 하는 제약조건들은 다음과 같다. 먼저, 조합논리 (combinational logic) 회로의 크기이다. RAM에 기초한 FPGA 칩은 조합논리 회로를 구현 하기 위한 LUT (LookUp Table)들을 포함하고 있다. 하나의 FPGA 칩은 수십 개에서 수백 개의 LUT를 포함하고 있으므로, 주어진 LUT 수에 맞게 회로를 분할 하여야 한다. 순차논리 (sequential logic) 회로를 구현하는데 사용되어지는 메모리의 양도 하나의 제약조건이다. 순차논리 회로는 조합논리 회로와 flipflop으로 나누어 구현한다. LUT와 마찬가지로 칩에 포함된 flipflop 수도 제한 되어 있으므로, flipflop 수를 제약조건으로 고려하여 분할하여야 한다. 또한, 각 칩에는 사용 가능한 입.출력 핀이 한정되어 있다. 핀은 분할에 의하여 다른 칩들과의 연결을 위해 사용되어지는 핀과, 외부 입.출력으로 사용되는 핀으로 나누어 진다. 분할시 핀의 수를 줄이기 위해서는 연결도를 최소화 하여, 다른 칩과 연결을 위하여 사용되어지는 핀의 수를 줄여야 한다. 또 하나의 제약 조건은 지연시간이다. 회로를 분할하여 여러 개의 칩으로 구현하면 칩간의 연결에 의하여 지연시간이 증가한다. 그러므로, 우수한 성능을 가지도록 분할하기 위하여 지연시간을 고려할 필요가 있다.

FPGA 칩은 보통 몇개의 LUT와 flipflop이 결합되어 하나의 CLB를 구성한다. 그러므로, 입력된 회로가 LUT와 flipflop으로 구분되어진 경우 이를 따로 고려하여 분할하지만, 기술매핑이 CLB 단위로 이루어진 경우에는 CLB 수만을 제약조건으로 사용하면 조합논리 회로와 flipflop에 대한 조건이 모두 만족되므로 구현이 간단해진다.

본 방법에서는 지연시간을 고려하지 않고 분할하였으며, 입력회로는 CLB 단위로 구성되었다고 가정한다. 본 방법에서 제약조건을 만족시키며 효율적인 분할을 수행하기 위하여 사용한 비용함수는 (2)식과 같다.

$$Cost = Net\_cost + W1 \times CLB\_cost + W2 \times Pin\_cost \quad (2)$$

식에서 첫째 항은 분할된 그룹을 연결하는 네트에 대한 비용이다. 둘째 항은 각 그룹에서 칩의 CLB 수보다 더 많이 할당된 CLB 수의 합이다. 그리고 셋째 항은 칩의 핀의 수보다 더 많이 할당된 핀 수의 합이고, W1, W2는 각 제약조건에 대한 가중치 (weight)이다. 분할 후 제약 조건이 모두 만족되면, 둘째 항과 셋째 항의 값은 0이 되므로 비용은 네트에 의한 비용으로만 구성된다.

2. 클러스터를 이용한 초기 분할 (Clustering-Based Initial Partitioning)

클러스터를 이용한 분할은 주어진 칩의 수에 맞도록 회로를 초기 분할하는 과정으로 전체적인 최적화에 중점을 두었다. 그러므로, 전체적으로 칩 간을 연결하는 네트의 수를 최소화하고, 분할된 회로의 크기의 균형에 중점을 두었다. 본 초기 분할은 II장에서 설명한 방법[5]을 개선하여 사용하였다. [5]의 방법에 비하여 본 방법의 주된 개선점은 다음과 같다.

[5]의 방법에서는 각 단계마다 초기 분할된 그룹을 번갈아 가며 선택하고, 선택된 그룹에 속한 셀 중에서 이득이 큰 것부터 다른 그룹으로 그룹의 크기 조건이 만족될 때까지 계속 이동시킨다. 그러나, 본 방법에서는 선택된 그룹에 속한 셀의 크기 조건이 만족되는 범위 내에서 이득이 큰 셀부터 모든 셀을 다른 그룹으로 이동시킨다. 더이상의 이동이 불가능하면 다른 그룹의 셀을 선택된 그룹으로 이동시키는 것을 크기 조건 때문에 더이상의 이동이 불가능할 때까지 반복한다. 속한 그룹에서 크기의 lower bound 까지 셀을 다른 그룹으로 이동시킨 후, 크기의 upper bound가 될 때까지 다른 그룹의 셀을 선택한 그룹으로 이동한다. Upper bound및 lower bound의 계산은 평균 그룹의 크기 (avg_size)에 각 단계에서 허용하는 크기를 더하고 빼서 구한다. 각 단계에서 허용하는 크기는 avg size * bal 이다. 여기서 bal은 허용하는 크기의 비율

이다. 만약 bal 값이 0.5이면 avg_size의 50% 까지 크기의 차이를 허용한다. 현재의 알고리즘에서 bal 값은 실험적으로 결정하였다. 초기에는 0.5에서 시작하여 매 iteration이 수행된 후, reduction ratio 0.9를 곱하여 감소시켰다. 그러나, bal값이 0.2보다 작은 경우에는 0.2를 사용하였다. 그러므로 iteration이 반복되면서 점차로 크기의 제약조건을 만족시키게 되며, 최종에는 avg_size의 20% 까지 크기의 편차가 허용된다.

[5]의 방법에 비하여 또하나의 개선점은 각 iteration에서 가장 우수한 분할을 선택하였다는 점이다. 셀을 이동시킬 경우에는 이를 리스트 (list)에 저장한다. 셀의 이동을 수행하고 그 단계에서 허용하는 크기의 조건을 만족하면서 비용이 가장 작은 곳까지만 채택하고 나머지 이동은 본래의 위치로 재 이동한다. 그러므로 각 단계마다 가장 좋은 결과를 저장하게 된다. 본 초기 분할에서는 비용의 계산 식에서 W1은 0, W2는 1을 사용하였다. W1을 0으로 사용한 이유는 각 단계에서 CLB 크기의 범위를 지정하여 주고 수행하므로 항상 주어진 크기 조건에 만족되기 때문이다.

3. 분할의 개선 (Partition Improvement)

분할의 개선은 클러스터를 이용한 초기 분할을 수행한 후, 만족되지 않은 제약조건을 만족시키기 위하여 수행한다. 초기 분할은 전체적인 최적화를 위하여 그룹의 크기의 균형 및 그룹 간을 연결하는 네트의 수, 즉 그룹들의 핀의 수를 줄이는데 중점을 두었다. 그러므로 전체적으로 좋은 분할이 되었더라도 일부 그룹에 CLB 및 핀의 수가 주어진 수보다 많이 할당되어질 수 있다. 분할의 개선 단계에서는 만족되지 않은 제약조건을 만족시키기 위한 셀의 이동을 수행한다.

분할의 개선은 다음과 같다. 먼저 각 셀의 이득을 구하고 이득이 큰 순서대로 다른 그룹으로 이동한다. 셀의 이득은 앞의 (2)식을 이용하여 비용을 계산할 때, 하나의 셀을 다른 그룹으로 이동하기 전의 비용에서 이동한 후의 비용을 뺀 값이다.

이득이 큰 순서로 한 그룹에서 다른 그룹으로 셀을 이동하고 더이상의 개선이 없으면, 다른 그룹으로부터 셀의 이동을 반복한다. 이와 같이 하여 모든 그룹에서 셀의 이동이 수행되면 한 번의 iteration을 마치게 된다. 한번의 iteration을 수행한 후, 핀 및 CLB 수의 제약조건이 만족되지 않았으면, 그 제약조건에 대한 가중치를 크게 함으로써, 다음의 이동에서 우선적으로 그 조건이 만족되도록 하였다.

가중치 W1과 W2의 초기값은 1이며, 가중치는 1씩 증가하였다. 처음 iteration에서 제약조건에 대한 가중치 W1과 W2는 1이지만, 첫 번째의 iteration을 수행한 후에 핀의 제약조건이 만족되지 않았으면 W2는 2로 증가하고, 두 번째 iteration을 수행한다. 그러면 핀의 overflow에 의한 비용이 커지므로 이를 우선적으로 줄이기 위한 이동이 수행된다. 이와 같이 만족되지 않은 제약조건의 가중치를 증가하여 셀의 이동을 수행하면 다른 제약조건이 만족되지 않게 되거나 네트에 의한 비용이 증가할 수도 있다. 그러나 iteration을 반복할수록 점차 제약조건들을 만족시키게 된다. 비용의 감소 없이 3번 이상 iteration을 반복하면 수행을 마치게 된다. 비용의 감소 없이 3번 이상의 iteration이 반복된 경우에는 더 이상 수행하여도 수행 시간만 증가하고 더이상 개선될 가능성이 적기 때문이다. 본 분할의 개선 단계에서는 초기 분할과는 달리 셀의 이동시 양의 이득을 가지는 셀만 이동하였다.

지금까지 기술한 분할의 개선과정을 알고리즘으로 나타내면 알고리즘 2와 같다.

```
Algorithm 2: Partition Improvement
evaluate_initial_gain;
initialize_weight;
move_group = 1;
while( TRUE ) {
    /* Find the cell C with the largest gain in
    move_group */
    C = find_cell;
    if( C != NULL ) {
        /* Move cell and update gain */
        move_cell;
        update_gain;
    }
    else {
        /* Change move_group */
        move_group ++;
        /* If one iteration is finished */
        if( move_group == NUM_GROUP + 1 ) {
            move_group = 1;
            if( all constraints are satisfied )
                break;
            else
                update_weight;
            if( partition is not improved for
                previous 3 iterations )
                break;
        }
    }
}
```

Ⅳ. 실험결과

지금까지 기술한 알고리즘을 C언어를 사용하여 UNIX 환경 하에서 구현하여 DEC 3000 시스템에서

실험하였다. 실험에는 MCNC 벤취마크의 예제를 사용하였으며, 실험에 사용한 예제는 표 1에 나타내었다. 표에서 회로명, CLB 수, IOB 수, 네트 수, pin 수 등이 각 열에 정리되어 있다. 예제 회로는 Xilinx 2000 series (XC2000)와 Xilinx 3000 series (XC3000) 칩의 구조[11]에 맞게 Xilinx 툴 (tool)을 사용하여 기술매핑 (technology mapping) 되어 있다. 그러므로, 예제 회로는 CLB 단위로 구성되어진 회로이다. 사용한 예제는 [7]의 방법에서 사용한 모든 예제를 나타낸다.

표 1. 벤취마크 예제
Table 1. Benchmark Examples.

	Circuit	#CLBs	#IOBs	#NETs	#PINs
XC2000	c499	74	73	123	409
	c1355	74	73	123	408
	c1908	147	58	238	780
	c2670	210	221	450	1255
	c3540	373	72	569	1933
	c5315	535	301	936	3004
	c6288	833	64	1472	3438
	c7552	611	313	1057	3318
XC3000	c3540	283	72	489	1645
	c5315	377	301	699	2409
	c6288	833	64	1472	3438
	c7552	489	313	921	2924
	s5378	381	86	628	2332
	s9234	454	43	716	2671
	s13207	915	154	1377	5309
	s15850	842	102	1265	4977
	s38584	2904	292	3884	17483

표 2. Xilinx 3000 series 칩의 종류[7]
Table 2. Xilinx 3000 series chip types.

Chip Type	Device	Cost ($)	#IOBs	#CLBs
1	XC3020	1.00	64	64
2	XC3030	1.36	80	100
3	XC3042	1.84	96	144
4	XC3064	3.15	110	224
5	XC3090	4.83	144	320

위의 회로를 구현하기 위하여, [7]의 방법에서는 Xilinx 2000 series 칩과 3000 series 칩을 이용하였다. Xilinx 2000 series에서는 64개의 CLB와 58개의 입.출력 핀을 가지는 한 종류의 칩만을 사용하였고 칩의 비용은 1.0으로 계산하였다. Xilinx 3000 series에서 사용한 칩의 종류는 표 2에 나타내었다. 표에서는 CLB와 핀의 수 및 비용도 함께 나타내었다. 본 방법에서는 [7]의 방법에서 사용한 것과 같은 종류의 칩과 같은 칩의 비용을 이용하여 실험하여 결과를 비교하였다.

표 3은 Xilinx 2000 series 칩의 구조에 맞게 기술매핑 된 회로를 본 논문에 기술한 방법 (New)으로 분할하여, [7]의 방법의 결과와 비교한 것이다. 표의 [7] 방법의 결과는 논문 [7]에서 인용하였다. 실험결과 본 방법은 사용한 모든 예제에 대하여 [7]의 방법과 동일하거나 약간 우수한 결과를 얻었다. 표에는 [7]에서 계산한 lower bound도 함께 나타내었다. 회로의 규모가 작기 때문에, 두 방법 모두 lower bound에 근접한 결과를 얻었다.

표 3. XC2000 예제의 분할 결과
Table 3. Partitioning results of XC2000 examples.

Circuit	Cost bound	[7]	New
c499	2.0	2.0	2.0
c1355	2.0	2.0	2.0
c1908	3.0	3.0	3.0
c2670	4.0	6.0	6.0
c3540	6.0	6.0	6.0
c5315	9.0	11.0	10.0
c6288	14.0	14.0	14.0
c7552	10.0	11.0	11.0
Total	50.0	55.0	54.0

벤취마크 예제 중에서 Xilinx 3000 series의 구조로 기술매핑 된 회로를 이용한 분할 실험을 수행하여 [7] 및 [8]의 방법과 비교하여 이를 표 4에 나타내었다. [7]의 방법은 서로 다른 종류의 칩을 사용하여 최소 비용을 가지도록 분할하므로 사용한 칩의 수를 Device distribution에 나타내었다. 그리고 모든 칩의 비용을 더한 총 비용 (total cost) 및 CLB utilization (CLB Util.)을 비교하였다. 그리고 새로운 방법의 수행시간을 제시하였다.

본 논문에서 기술한 새로운 방법에서는 같은 종류의 칩만을 사용하여 주어진 수의 칩에 맞게 회로를 분할하여 성공한 결과이다. 분할 후, 분할된 부분회로 중에서 더 작은 칩을 사용하여 구현 가능한 그룹은 더 작은 칩을 사용하도록 하였다. 실험결과 3개의 예제에서만 각각 하나의 부분회로가 더 작은 칩으로 구현 가능하였다. 전반적으로 본 분할 알고리즘은 여러 개의 FPGA를 위한 분할을 성공적으로 수행하는 우수한 결과를 보여 주었다. 결과의 비교에서 본 방법은 9개의 예제 중에서 6개의 예제에서 가장 우수한 결과를 얻었

표 4. XC3000 예제의 분할 결과
Table 4. Partitioning results of XC3000 examples.

Circuit	[7]			[8]		New			
	Device Distribution	Total cost	CLB Util. (%)	Total cost	CLB Util. (%)	Device Distribution	Total cost	CLB Util. (%)	CPU (sec)
c3540	{0,0,3,0,0}	5.52	86	4.56	85	{0,3,0,0,0}	4.08	94	9
c5315	{2,1,2,0,0}	7.03	73	6.92	73	{0,6,0,0,0}	8.16	62	11
c6288	{0,0,4,2,0}	13.66	96	13.76	82	{0,0,6,0,0}	11.04	96	38
c7552	{0,0,4,0,0}	7.36	85	7.36	85	{0,6,0,0,0}	8.16	81	18
s5378	{0,0,1,0,1}	6.67	82	6.19	94	{0,0,0,2,0}	6.3	85	14
s9234	{0,0,0,1,1}	7.98	77	7.98	86	{0,1,3,0,0}	6.88	85	26
s13207	{3,5,4,0,0}	17.16	72	18.12	83	{1,11,0,0,0}	15.96	78	375
s15850	{0,0,2,2,1}	14.80	83	14.97	81	{0,10,0,0,0}	13.6	84	272
s38584	{0,5,15,4,1}	51.83	75	51.19	82	{0,1,21,0,0}	40.00	92	1308
Total		132.01		131.05			114.16		

다. 사용한 모든 예제를 구현하는데 필요한 전체 비용이 [7]의 방법은 132이며, 논리의 복제를 허용한 [8]의 방법도 131로 거의 유사한 결과를 보여주었지만, 본 논문의 방법은 114.16의 비용으로 구현 가능하여 14.7% 우수한 결과를 보여 주었다. 특히 큰 회로에서 더 나은 결과를 보여 주었다. 본 알고리즘의 결과가 더 우수한 이유는 전체적인 분할의 최적화를 수행한 후, 부분적인 크기의 제약조건을 만족시키는 방법을 사용하였으며, 전체를 한 번에 여러 개의 그룹으로 분할하여, 전체적인 최적화를 비교적 잘 수행하였기 때문으로 생각된다.

V. 결 론

여러 개의 FPGA 칩을 이용하여 대규모 회로를 구현하기 위한 효율적인 분할 알고리즘을 개발하였다. 새로운 분할 방법은, 전체적으로 최적에 가까운 분할을 수행하면서 제약조건들을 효율적으로 만족시켜 주기 위하여, 전체적인 최적화를 위한 초기 분할 과정과 반복적인 분할의 개선 과정으로 나누어 수행하였다. 먼저 기존의 클러스터를 이용한 분할 알고리즘을 효율적으로 개선하여 전체적인 최적화를 위한 초기 분할을 수행하였다. 제약 조건들을 만족시켜 주기 위한 분할의 개선 과정에서는 가중치를 동적으로 조정하여 최종에는 모든 제약조건들이 만족되도록 하였다. MCNC 분할 벤취마크 회로를 이용한 실험결과, Xilinx 3000 series를 이용한 9개의 예제 회로에서, 본 알고리즘은 최근에 발표된 다른 방법 [7], [8]에 비하여 14% 이상 평균적으로 우수한 결과를 보여 주었으며, 특히 s38584 등의 큰 회로에서는 20% 이상 우수한 결과를 보여 주었다.

참 고 문 헌

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

[2] 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.

[3] B. Krishnamurthy, "An Improved Min-cut Algorithm for Partitioning VLSI Networks", IEEE Trans. Comput., vol. C-33, pp. 438-446, May 1984.

[4] C. W. Yeh and C. K. Cheng, "A General Purpose Multiple way Partitioning Algorithm", Proc. 28th Design Automation Conference, pp. 421-426, 1991.

[5] H. Shin and C. Kim, "A Simple Yet Effective Technique for Partitioning," IEEE Trans. on VLSI Systems, Vol. 1, No. 3, pp. 380-386, Sep. 1993.

[6] W. O. McDermith, "A Bottom-Up Approach to FPGA Partitioning," in Proc. IEEE Custom Integrated Circuits Conference, pp. 5.4, 1992.

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

[8] 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.

[9] N.-C. Chou, L.-T. Liu, 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.

[10] Z. Hasan, D. Harrison and M. Ciesielski, "A Fast Partitioning Method for PLA-Based FPGAs," IEEE Design & Test of Computers, pp. 34-39, Dec., 1992.

[11] Xilinx, Inc., The Programmable Gate Array Data Book, Xilinx, San Jose, 1992.

[12] L. A. Sanchis, "Multiple-Way Network Partitioning", IEEE Trans. on Computers. Vol. 38, No 1, Jan, 1989.

[13] Y. C. Wei and C. K. Cheng, "Towards Efficient Hierarchical Designs by Ratio Cut Partitioning", Int. Conf. on Computer Aided Design, pp. 298-301, 1989.

저 자 소 개

金 表 希(學生會員) 第 30卷 A編 第 6號 參照
현재 한양대학교 대학원 박사과정

申 鉉 哲(正會員) 第 32卷 A編 第 2號 參照
현재 한양대학교 전자공학과 부교수