SoC를 위한 다단 HW/SW 분할 알고리듬

안 병 규, 신 봉 식, 정 정 화
한양대학교 정보통신대학원
전화 : 02-2290-0558 / 핸드폰 : 019-538-9296

A Multi-Level HW/SW Partitioning Algorithm for SoCs

Byunggyu Ahn, Bongsik Sihn, Jongwha Chong
Dept. of Information & Communications, Hanyang University
E-mail : mon809@chol.com

Abstract

In this paper, we present a new efficient multi-level hardware/software partitioning algorithm for system-on-a-chip design. Originally the multi-level partitioning algorithm are proposed to enhance the performance of previous iterative improvement partitioning algorithm for large scale circuits. But when designing very complex and heterogeneous SoCs, the HW/SW partitioning decision needs to be made prior to refining the system description. In this paper, we present a new method, based on multi-level algorithm, which can cover SoC design. The different variants of algorithm are evaluated by a randomly generated test graph. The experimental results on test graphs show improvement average 9.85% and 8.51% in total communication costs over FM and CLIP respectively.

I. 서론

오늘날 많은 내장형 시스템(embedded system)들이 반도체 공정 기술의 발전에 힘입어, 하나의 칩 내부에 시스템을 모두 포함하는 SoC(System-on-a-Chips)로 제작 되고 있다. 특히, 현재 전자 시스템을 설계하는 과정에서 요구되고 있는 성능, 전력 소비, 비용, 신뢰도, 크기 등 주요한 조건들을 모두 만족시키기 위해서 SoC로의 구현이 필수적이기 때문이다. SoC가 가지는 여러 가지 장점들로 인해 수요가 폭발적으로 증가하게 되었고, 그로 인해 많은 사람들이 SoC에 대해 관심을 가지게 되었으며 관련 기업체들 역시 설계 기술을 확보하기 위해 많은 투자를 하고 있다.

현재 SoC를 제작할 경우 크게 프로세서(DSP, general purpose CPU)에서 구동되어지는 소프트웨어 부분과 ASIC 또는 FPGA로 구현되는 하드웨어 부분으로 나눌 수 있다. HW/SW 분할(HW/SW partitioning)은 정의된 시스템의 각 기능들(functionality)을 하드웨어 혹은 소프트웨어로 구현할지 결정하는 과정이며, 각 구현에 따르는 이득과 손실의 기회비용(trade-off)을 잘 생각하여, 한정된 개발 비용에 대해서 시스템의 성능이 최대가 되도록 분할 결과를 유도하여야 한다. 하지만, 통합설계(co-design)의 설계 공간(design space)은 하드웨어나 소프트웨어만의 단일 설계 공간에 비해 훨씬 크고 불규칙적이므로, 이러한 설계 공간을 탐색하여 최적의 해를 찾는 것은 NP-hard 문제로 알려져 있다.

본 논문은 HW/SW 분할의 결과가 로컬 미니멈(local minimum)에 빠지지 않고 최적의 해에 근접하게 하기 위해, 회로 분할(circuit partitioning)에 적용되는 다단 알고리듬(multi-level algorithm)을 변형하여 적용한다. 또한 각 레벨에서는 통신 채널(communication channel)의 형성을 최소화하고, 서로 간의 의존도가 높

은 태스크들이 서로 다른 영역으로 분할되는 것을 막기 위해서 수정된 CLIP 알고리즘을 적용한다.

2장에서는 기존의 대표적인 그래프 이용한 분할 방법들에 대해서 알아보고, 3장에서는 기존의 회로 분할 방법을 HW/SW 분할로 적용하는 방법에 대해서 제안한다. 4장에서는 제안된 분할 방법의 성능을 입증하기 위하여 랜덤 태스크 세트를 대상으로 기존의 분할 방법들과 분할결과를 비교 분석하고 효율성을 입증한다. 마지막으로 5장에서는 제안된 분할 방법에 대한 결론을 맺는다.

II. 기존의 분할 기법

2.1 기존의 회로 분할 알고리듬

대규모 회로의 설계 자동화를 위해서는 적절한 회로 분할 알고리듬이 반드시 필요하다. 그러나 이를 위한 회로 분할 방법은 목적 함수의 최적화와 연관되면서 최적의 해를 합리적인 시간 안에 구하는 것은 불가능한 것으로 증명되었다. 따라서 목적 함수에 부응하는 최적의 분할 결과에 최대로 근접한 결과를 합리적인 시간 안에 구하기 위한 많은 연구가 진행되어 왔다. 분할에 관한 연구는 크게 반복적 분할 개선(Iterative Improvement Partition(IIP)) 방법과 구조적 분할(constructive partition) 방법으로 나뉘어 발전되어 왔고, 이들 방법을 조합한 다단 분할(multi-level partition) 방법도 다양하게 제안되고 있다.

대표적인 IIP 방법으로는 KL(Kernighan-Lin) 방법과 그 뒤를 이어서 지금까지도 널리 사용되고 있는 FM(Fiduccia-Mattheyses)방법이 있다. 근래 S. Dutt의 CLIP(CLuster-oriented Iterative Improvement Partitioner)방법[1]에서는 분할 개선 시에 즉각적인 컷사이즈(cutsize) 감소량(total gain)에 따른 셀 선택 방식이 로컬 미니멈에 머무를 가능성이 높다는 사실을 지적하고, 즉각적인 컷사이즈 감소량 보다는 이전 이동 셀에 의한 컷사이즈 변화량(updated gain)을 셀 선택의 판단 근거로 이용하는 방식을 제안하였다. CLIP 방법은 FM 방법과 거의 대등한 수행 속도를 유지하며 놀라운 분할 성능 향상을 가져왔다.

CLIP에서 제안된 방식은 하나의 셀 이득(gain)을 초기 이득(initial gain)과 갱신 이득(update gain)으로 분류하여 초기 이득은 초기 분할 직후에 그 셀의 이동에 따른 즉각적인 컷사이즈 감소량을 의미하고 갱신 이득은 다른 셀의 이동에 의해 영향 받은 그 셀의 이득 변화량을 의미한다. 따라서 갱신 이득이 높은 셀은 이전에 이동한 셀에 의해 끌리는 힘이 크다고 해석할 수 있고 CLIP 방법에서는 이와 같이 끌리는 힘 즉, 갱신 이득이 높은 셀을 최적 셀로 선택하여 우선 이동 한다. 이러한 선택 방식은 하나의 셀의 이동이 이와 연결된 또 다른 셀의 이동을 불러일으킨다는 사실을 직관적으로 알 수 있다. 이러한 사실은 분할 개선 시에 한 클러스터내의 임의의 셀 u가 이동하면 다른 클러스터내의 셀에 우선하여 u가 속해 있는 클러스터내의 다른 셀들의 이동을 연쇄적으로 촉진시키는 효과를 얻을 수 있고, 이는 한 클러스터가 분할된 양 부분에 의해 나뉘어 지는 것을 방지(cluster-Removal)할 수 있다.

2.2 기존의 HW/SW 분할 기법

HW/SW 분할을 자동화하는 대표적인 방법들로 VULCANII[3]와 COSYMA[4]가 있다. 두 가지 방법은 모두 하위 단계(fine-grain)에서 수행되는 분할 방법이며, VULCANII는 하드웨어로 COSYMA는 소프트웨어로 편향된 상태에서 분할을 시작한다.

하위 단계 분할을 사용한 다른 접근 방법은 [5, 6]에서 소개되고 있으며, 반면에 상위 단계(corse-grain) 방법을 사용한 [7, 8]등이 있다.

위 방법들의 공통적인 특징은 분할 대상의 단위가 고정적이라는 점이다. 제안하는 접근 방법은 분할 개체를 레벨 단위로 클러스터링을 수행함으로써, 상위 단계에서 하위 단계까지 효과적으로 분할하는 것을 목적으로 하고 있다.

III. 제안하는 HW/SW 분할 알고리듬

3.1 접근 방법

HW/SW 분할은 탐색공간이 매우 넓어서 한정된 시간 내에 최적의 해(global minimum)를 찾는 것은 거의 불가능 하다. 그래서 빠른 결과를 얻을 수 있는 greedy 알고리듬을 사용하기도 하나, 로컬 미니멈에 빠질 확률이 높아서 개선된 방법인 시뮬레이티드 어닐링(simulated annealing) 알고리듬이 많이 사용되는 편이다. 하지만 이 또한 상대적으로 매우 큰 수행 시간이 필요하며, 초기 분할 조건에 따라서 결과에 큰 차이를 보이게 된다.

본 논문에서는 수행 속도를 개선시키고 의존성이 높은 태스크들이 다른 영역으로 분할되는 것을 막기 위해 근접도가 높은 태스크를 클러스터링하고, 이를 그림 3과 같이 다단(multi-level)으로 구성한다. 여기서 클러스터링 할 때 사용되는 근접도 함수는 두 태스크 사이에 의존성(dependency)이 있는 변수들의 집합으로 정의된다.

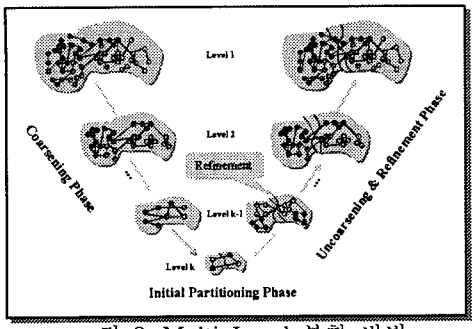

그림 3. Multi-Level 분할 방법

제한되어 있는 매칭율과 레벨 한계치까지 클러스터링을 레벨 단위로 수행하고, 마지막 레벨에서는 초기 분할을 수행한다. 그리고 각 레벨에서는 HW/SW 간의 통신 채널(communication channel)이 최소한으로 되면서 전체적인 cost가 낮아지는 방향으로 분할을 수행한다. 이를 위해 기존의 CLIP 알고리듬을 HW/SW 분할에 적합하도록 초기 이득(initial gain)이 크기(area)와 시간(time) 부분(component)을 가진 비용 함수(cost function)가 되도록 수정하고, 갱신 이득(update gain)은 의존성이 높은 태스크간의 분할을 막기 위해 통신 채널의 변화량으로 수정한다.

3.2 근접도와 비용 함수의 정의

(1) 근접도(Closeness)

클러스터링을 수행하기 위한 두 노드간의 근접도는 다음과 같이 정의한다.

$$closeness = \sum_{k=0}^{n} | e(v_i, v_j)_k | \cdot c_{e_k} \qquad (1)$$

두 노드 v_i와 v_j간의 간선(edge), e_k의 비중(weight)과 해당 간선에서 단위 시간동안 통신이 일어날 확률의 곱으로 정의하며, 만약 클러스터간의 간선일 경우는 내부에서 외부로 나가는 모든 간선들의 비중의 합이다.

(2) 비용 함수(Cost Function)

CLIP 방법을 HW/SW 분할에 적합하도록 초기 이득(initial gain)과 갱신 이득(update gain)을 다음과 같이 정의한다.

$$initial\_gain_{SW->HW} = k_1 \cdot (t_{SW} - t_{HW}) + k_2 \cdot (-A_{HW}) + k_3 \cdot \Delta\sum closeness \qquad (2)$$

$$initial\_gain_{HW->SW} = k_1 \cdot (t_{HW} - t_{SW}) + k_2 \cdot (A_{HW}) + k_3 \cdot \Delta\sum closeness \qquad (3)$$

초기 이득 값은 *SW->HW*와 *HW->SW*로 이동하는 두 가지로 나누어서 각각 식(1)과 식(2)로 계산한다. 그 이유는 각 영역에서 얻을 수 있는 이득이 다르기 때문이며, 식은 각 영역으로 구현시의 시간차와 HW로 구현되는 크기, 그리고 근접도의 변화량의 합을 나타낸다. 다른 차원을 동시에 고려하기 위해 비중 값인 k를 사용한다.

$$update\_gain = \Delta gain = gain_{after} - gain_{before} \qquad (4)$$

갱신 이득 값은 가장 큰 초기 이득 값을 가지는 노드를 이동한 후에 해당 노드의 이동에 의해 영향을 받은 노드들의 이득 변화량이며, 식(4)의 갱신 이득 값이 크다는 것은 끌리는 힘이 크다고 해석할 수 있다

3.3 제안된 HW/SW 분할 알고리듬

본 논문에서 제안하는 알고리듬은 다음과 같다.

```
/* 계층적인 클러스터 형성 */
For (l = 0 upto MAX_LEVEL)
  G_{l+1}(V_{l+1}, E_{l+1}) = Cluster(G_l(V_l, E_l))
  Set l = l + 1
Endfor
/* 레벨 단위 분할 */
P_0 = Partition(G_MAX_LEVEL, Null)
For (l = MAX_LEVEL-1 downto 0)
  Pl = Partition(G_l, P_{i+1})
Endfor
Return P_0
/* HW/SW CLIP 알고리듬 */
InitialGain(G_l, P_i)
While (there exists free node)
  n = PickBestNode(G_l)
  Move(n)
  Lock(n)
  For (each node n_i connected with n )
    UpdateGain(n_i)
  Endfor
Endwhile
```

그림 4. 제안된 HW/SW 분할 알고리듬

그림 4와 같이 본 알고리듬은 계층구조를 형성하는 단계와 계층 구조의 최상위레벨에서부터 반복적으로 분할을 수행하는 단계로 구성된다. 분할을 수행하는 각 단계 중 최상위 레벨에서는 임의 초기 분할을 이용

하였고, 이를 제외한 이하 레벨에서는 해당 레벨 보다 한 단계 높은 레벨의 분할 결과를 초기 분할로 사용한다. 각 분할 단계에서는 전단계의 분할 결과를 바탕으로 초기 이득을 구한 후, 가장 이득이 큰 노드를 옮긴다. 그 후 옮긴 노드에 영향을 가장 많이 받는 노드, 즉 갱신 이득이 가장 큰 값을 선택하여 이동하며 이 과정을 반복적으로 적용한다.

Ⅳ. 실험 결과

본 장에서는 앞서 설명한 방법의 성능을 검증하기 위해서 랜덤으로 생성된 태스크 그래프를 이용하여 각 알고리즘의 성능을 비교 분석하였다. 10가지의 랜덤 태스크 그래프에 대해서 10번씩 각 알고리즘을 수행하여 평균값을 구한 후, 기존 알고리즘 대비 제안된 알고리즘의 성능 향상율을 나타내었다. 여기서의 주요 목적함수는 통신비용의 감소를 비교한다.

표 1. 기존 알고리즘 비교 분석

알고리듬	FM	CLIP	Proposed
통신비용 (normalized)	1.0985	1.0851	1.0

표 1을 살펴보면 FM 방법에 비해서 성능이 9.85% 향상하였고, CLIP 만을 적용한 방법에 비해서 8.51% 향상되었다.

Ⅴ. 결론

본 논문은 기존의 회로분할에서 사용되던 다단 분할 알고리즘과 CLIP 알고리즘을 동시에 적용하여, 상위 단계와 하위 단계에서 최적의 이동 노드를 찾아내고, 통신에 걸리는 시간을 최소화 하였다.

HW/SW 분할은 통합설계 과정에서 시스템의 최적의 가격대비 성능을 결정하기 위한 매우 중요한 과제이다. 제안된 방법론은 실제 내장형 시스템(embedded system)을 구현하는 과정에서 더욱 뛰어난 분할 결과를 얻기 위한 하나의 방법으로 사용되어 질 수 있으며, 이를 토대로 전체 시스템의 가격대 성능비가 크게 향상될 것으로 예측된다. 또한 기존에 제안되어 있는 다양한 통합 설계 방법론(methodology)에 선택적으로 적용이 가능하며, 보다 우수한 분할 결과를 유도할 것으로 예상된다.

참고문헌

[1] S. Dutt and W. Deng, "VLSI Circuit Partitioning by Cluster-Removal Using Iterative Improvement Techniques," Proc. Intl Conf. Computer-Aided Design, 1966, pp. 194-200.

[2] C. J. Alpert, J-H Huang and A. B. "Kahng, Multilevel Circuit Partitioning," Proc. ACM/IEEE Design Automation Conf., 1997, pp. 530-533.

[3] R. Gupta and G. De Micheli. "System-level Synthesis Using-Reprogramable Components," Proc. of EDAC, pp. 2-7, March 1992.

[4] J. Henkel, R. Ernst, U. Holtmann, T. Benner, "Adaptation of Partitioning and High-Level Synthesis in Hardware/Software Co-Synthesis," Proc. of ICCAD, pp. 96-100, 1994

[5] Barros,W. Rosenstiel, X. Xiong, "A Method for Partitioning UNITY Language in Hardware and Software," Proc. of IEEE/ACM Proc. of The European Conference on Design Automation (EuroDAC) 1994, pp. 220 - 225, 1994.

[6] A. Jantsch, P. Ellervee, J. Oeberg et. al., "Hardware/Software Partitioning and Minimizing Memory Interface Traffic," IEEE/ACM Proc. of The European Conference on Design Automation (EuroDAC) 1994, pp. 220 - 225, 1994.

[7] F. Vahid, D.D. Gajski, J. Gong, "A Binary - Constraint Search Algorithm for Minimizing Hardware during Hardware/Software Partitioning," IEEE/ACM Proc. of The European Conference on Design Automation (EuroDAC) 1994, pp. 214 - 219, 1994.

[8] F. Vahid,D. D. Gajski, "Clustering for improved system - level functional partitioning," IEEE/ACM Proc. of 8th. International Symposium on System Synthesis, pp. 28 - 33, 1995.