2000년 한국정보처리학회 추계 학술발표논문집 제7권 제2호

# 노드의 상대적 스케쥴 긴박도 분석에 의한 하드웨어 소프트웨어 분할

오 주 영*, 박 도 순*
*홍익대학교 컴퓨터 공학과
email : jyoh@cs.hongik.ac.kr

# Hardware-Software partitioning by analysis of node's relative scheduling urgency

Ju-Young Oh*, Do-Soon Park*
* Dept. of Computer Engineering, Hongik University

요 약

통합설계에서 제약사항을 만족하는 최적의 시스템을 구현하기 위해 시스템을 기술하는 각 부분을 하드웨어부와 소프트웨어부로 나누어 매핑의 권역을 찾는 분할은 중요한 문제이다. 기존의 분할 알고리즘들[1]은 파티션과 스케쥴링을 2단계로 분리하여 분할 단계의 결과를 스케쥴링하는 과정에 의해 진행되었다. 이러한 작업과정은 스케쥴링 결과 스케쥴이 불가능한 경우 시스템을 재설계 해야하는 문제점을 가진다. 본 논문에서는 분할 단계에서 스케쥴링을 함께 고려하는 낮은 복잡도의 알고리즘을 제안한다. FDS를 응용한 기존 논문[4]이 고려하지 못한 자원제약에 의한 힘값 변이를 고려할 수 있도록 하였고 알고리즘 복잡도를 개선하기 위하여 종속성 제약 조건에 의해 받는 다른 노드의 힘값 계산 방법을 수정하였다. 수정된 계산 방법에서는 특정 노드와 경쟁 노드들의 제어구간별 상대적 스케쥴 요구값의 크기에 의해 분할 대상 노드를 선택하게 된다. 제안된 논문의 실험결과는 시스템 제약시간을 만족하면서 구현비용을 저하시키고 알고리즘 실행시간 측면에서 효과적임을 보인다

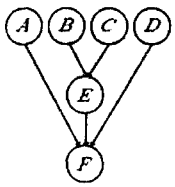
## 1. 서론

통합설계에서 입력 기술을 하드웨어부와 소프트웨어부로 매핑될 권역을 설정하는 분할 알고리즘은 시스템의 성능에 가장 큰 영향을 미친다. 이러한 분할을 위하여 시스템 표현모델, 분할 대상 노드의 크기, 비용함수 계산 방법, greedy 휴리스틱, 클러스트링, 반복 개선, 수학적 프로그래밍 방법 등의 설계 대상 시스템 환경과 목적함수에 따른 다양한 알고리즘들[1]이 개발되어왔다. 하지만, 대부분의 기존 알고리즘들은 분할 단계와 스케쥴링 단계를 독립적인 작업과정으로 분리하여 진행하였다. 분할과 스케쥴링 작업단계의 분리는 스케쥴링 결과가 시간제약을 위반할 경우 적합한 해를 탐색하기 위해 재분할 해야하는 문제점을 가진다. 본 논문에서는 이러한 문제점을 해결하는 낮은 실행시간의 알고리즘 개발을 위해 상위수준 합성에서의 FDS[2]를 응용하여 분할 단계에서 스케쥴링을 함께 고려한다. Choi[3]는 FDS를 응용하여 하드웨어로 매핑할 노드를 선택할 경우 소프트웨어 노드와의 병렬실행 가능성을 힘값으로 하여 분할을 수행하였으며, Rousseau[4]는 하드웨어 또는 소프트웨어 제어 구간에 스케쥴할 때 자신의 구현 비용이 반영된 힘값과 종속성에 의해 여타 노드가 받는 힘값을 반영하여 분할을 수행하였다. 본 논문에서는 기존의 FDS 응용의 힘값 계산에 의한 복잡도를 개선하기 위하여 힘값 계산방법을 수정하였다. 2절에서 상대적 스케쥴 긴박도와 제안 알고리즘을, 3절에서 알고리즘 성능평가를 4절에서 결론을 각각 기술하였다.

## 2. 상대적 스케쥴 긴박도

### 2.1 FDS 응용

논문 [3][4]는 FDS 방법을 응용하여 분할 단계에서 스케쥴링을 함께 고려하였다. 이를 위하여 파티션 결과에 미치는 영향 값으로 한 노드가 특정 파티션으로 분할되어 특정 제어구간에 스케쥴될 경우 시스템에 부과하는 자신의 힘값과, 스케쥴링 이후 종속성에 의해 삭제되는 다른 노드의 영향값인 Induced 힘값을 함께 고려하여 분할을 진행하였다.

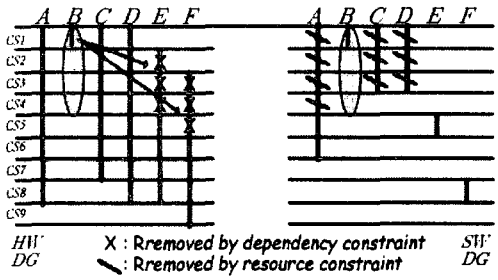



| | $HT_i$ | $ST_i$ | $HC_i$ | $SC_i$ |
|---|---|---|---|---|
| A | 1 | 2 | 2 | 1 |
| B | 1 | 4 | 2 | 1 |
| C | 1 | 2 | 4 | 1 |
| D | 1 | 5 | 2 | 1 |
| E | 1 | 3 | 2 | 1 |
| F | 1 | 2 | 5 | 1 |

(a) 입력 DAG (b) 노드의 비용

그림[1] 입력 DAG와 비용테이블

그림[2]는 그림[1]과 같은 입력 조건과 제어단계 9의 제약시간에 대한 ASAP과 ALAP 스케쥴에 의해 구성된 하드웨어 소프트웨어 각각의 분포그래프이다. 알고리즘은 분할 대상 노드를 선택하기 위하여 각 노드의 분포그래프별, 제어단계별 힘값을 계산하게 된다. 노드 B가 소프트웨어로 분할되어 제어단계 1에 스케쥴되는 힘값을 계산하는 경우 입력그래프의 종속선상에 있는 모든 노드의 영향 값을 계산해야 하며, 소프트웨어 분포그래프의 경우 자원 제약조건에 의해 삭제되어야 하는 다른 노드의 영향 값은 배제되었다. 자원제약에 의해 삭제되는 노드의 제어구간은 소프트웨어로의 분할 가능성을 줄여서 결국 시스템 설계비용을 증가시킬 수 있다.

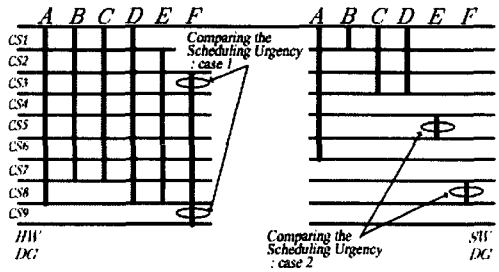



그림[2] 종속성 제약조건과 자원제약 조건

이 방법의 알고리즘 복잡도는 제어구간의 수가 c이고 노드 개수가 n일 경우, 알고리즘이 한번 반복될 때에 하나의 노드만이 분할되고 각 제어구간에 배정될 수 있는 모든 노드의 힘값이 계산되며, Induced 힘값은 최악의 경우에 n-1개에 대하여 계산되므로 알고리즘 복잡도는 $\theta(c^2 n^3)$가 된다.

### 2.2 상대적 스케쥴 긴박도

본 논문에서는 시스템의 설계시간을 개선하기 위하여 자신의 스케쥴로 인해 다른 노드들이 받는 영향값[4]의 계산 방법을 수정하여, 스케쥴을 위해 특정 제어 단계에서 경쟁하는 노드들의 모빌리티에 대한 상대값과 해당 파티션에 대한 구현 비용만을 반영하여 전체적인 복잡도를 개선할 수 있도록 하였다. 그림[3]의 case1에서 F노드가 제어단계 3에 스케쥴 되는 경우와 제어단계 9에 스케쥴 되는 경우 스케쥴 이후 다른 노드의 모빌리티에 부과하는 영향값은 차이가 있으며 결과적으로 해당 제어 단계에 대한 스케쥴 기회를 줄이게 되므로 특정 제어단계에서 경쟁하는 다른 노드의 빈수가 적은 노드가 선택될 수 있도록 하였다. 또한, 자신의 힘값은 설계비용이 낮은 노드가 선택될 수 있도록 하였다. case2에서 노드 E와 F의 자신의 스케쥴 확률은 동일하지만, 자원 제약조건에 의해, E노드는 A노드의 스케쥴 가능성을 줄이게 된다. 시간제약과 노드의 모빌리티간의 관계는 하드웨어 분포그래프와 소프트웨어 분포그래프 각각이 비례하며 따라서 해당 제어 단계에 대한 스케쥴 요구 경쟁 분석은 분포그래프 상호간에 나타날 수 있는 영향값을 반영할 수 있도록 한다. 노드간 종속성 제약 조건은 분할 노드 선택이후 해당 제어단계에 스케쥴 이후 모빌리티 수정에 따라 반영되며 이러한 방법은 각각의 노드별, 제어단계별, 분포그래프별로 수행된다.




그림[3] 상대적 스케쥴 긴박도

#### 2.2.1 수식 정의

수식 (1)은 노드 $i$의 하드웨어 소프트웨어 각각의 분포그래프에서의 확률 값을 나타내며 (2),(3)은 노드 $i$가 제어구간 $j$에 스케쥴되어 시스템에 미치는 자신의 힘값을 의미한다. 해당 제어단계에 대한 스케쥴 확률값과 구현비용을 추가함으로써 전체 시스템의 설계비용과 광역실행 시간을 고려할 수 있도록 정의하였으며, 이미 분할된 노드의 상태값(비용, 시간)을 반영하여 스케쥴 가능성을 높이고 설계비용을 낮출수 있도록 하였다. 수식(4),(5)는 노드의 상대적 스케쥴 긴박도를 나타내며 제어구간 $j$에 노드 $i$가 특정 파티션으로 매핑되어 스케쥴되어야 하는 타당성을 제어구간 $j$에서 스케쥴 될 수 있는 다른 노드들의 분포그래프별 자신의 힘값의 합으로 감하여 계산되는 값이다. 각 분포그래프의 계산 결과에서 긴박도가 최대인 노드를 해당 제어구간의 특정 파티션으로 분할하는 방법을 취한다.

$$distr_{hard}(i) = 1/(n+1-th_i)$$
$$distr_{soft}(i) = 1/(n+1-ts_i) \quad \text{----(1)}$$

$$SForce^{i}_{soft}(i) = distri_i \times \frac{1}{ST_i \times AssignedSWNC} \quad ---(2)$$

$$SForce^{i}_{hard}(i) = distri_i \times \frac{1}{HC_i \times AssignedHWNC} \quad ---(3)$$

$$Urgency^{i}_{soft}(i) = SForce^{i}_{soft}(i) - \sum_{k=1(k\neq i)}^{n} SForce^{i}_{k} \quad ---(4)$$

$$Urgency^{i}_{hard}(i) = SForce^{i}_{hard}(i) - \sum_{k=1(k\neq i)}^{n} SForce^{i}_{k} \quad ---(5)$$

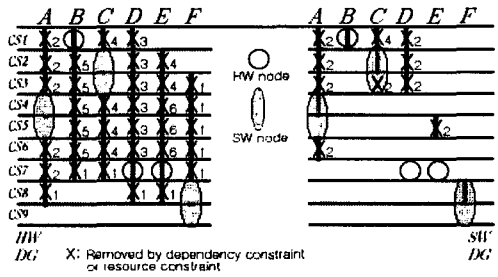
$$Urgency_i(i) = \max\{ Urgency^{i}_{soft}(i),\ Urgency^{i}_{hard}(i)\} \quad ---(6)$$

본 논문에서 제안하는 알고리즘은 그림[4]와 같다. 주어진 시간제약에 대한 분포그래프를 구성하고 각각의 노드별, 분포그래프별, 제어단계별 상대적 스케줄 긴박도를 구하여 최대 값을 가지는 노드를 해당 제어구간의 특정 파티션으로 배정한다. 배정된 노드를 분할 대상집합에서 제거하고 분포그래프를 수정하는 과정으로 알고리즘이 진행되며, 그림[1]과 같은 입력에 대한 분할과정 및 결과는 그림[5]와 같다. 알고리즘 반복에 의한 노드의 분할과 분포그래프의 변환 과정은 첨자로 표시하였다.

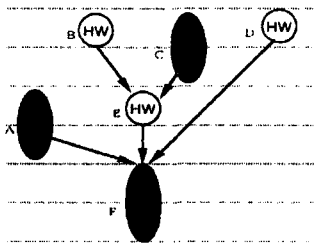
```
for(분할 대상 노드 집합)
 {
   for(분할 대상 노드 집합)
    {
      - 비용함수 계산
      - 최대 비용함수 계산 값의 노드 선택
      - 해당제어구간의 해당 파티션으로 노드 배정
      - 하드웨어·소프트웨어 분포그래프 수정
    }
   -분할 대상 노드집합에서 해당 노드를 삭제
 }
```

그림[4] 분할 알고리즘

최초 분할에 의해 F노드가 제어단계 8에서 소프트웨어로 배정되어 분포그래프에서 첨자 1로 표기된 각각의 제어단계가 삭제되는데 하드웨어 분포그래프의 경우 종속성에 의해 A, B, C, D, E 노드의 제어단계가 삭제된다. 알고리즘의 두번째 반복에 의해 A 노드가 4제어단계에서 소프트웨어로 배정되며 배정 후 소프트웨어 분포그래프의 경우 자원제약과 노드의 실행 시간에 의해 스케줄 될 수 없는 각각의 제어구간들이 삭제된다.




(a) 반복(루프)에 수반되는 노드 분할과정




(b) 분할 결과

그림[5] 분할과정 및 결과

노드 개수가 n인 경우 알고리즘 복잡도는, 각 노드가 가지는 제어 구간에 대하여 하드웨어 소프트웨어 분포그래프 각각에 대한 해당 제어구간에서의 스케줄 긴박도를 계산하고 최대 값을 가지는 노드를 해당 파티션으로 매핑한다. 매핑 이후 해당 노드를 스케줄 대상 노드에서 제거한 후 DAG 내의 모든 대상 노드의 스케줄 완료 시점까지 반복 실행한다. 제어단계가 c라고 가정할 경우 알고리즘 복잡도는 $\theta(c\ n^2)$으로 계산된다.

### 3. 알고리즘 성능평가

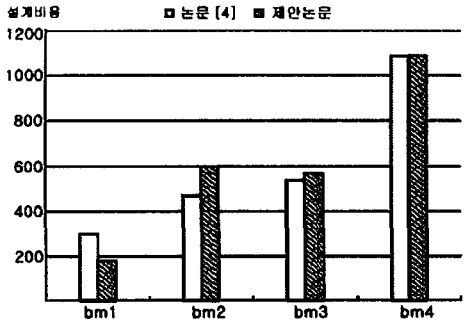
제안된 알고리즘에 의한 분할의 효율성 검증을 위하여 그림[6]과 같은 각각의 벤치마크로 실험하여 논문[4]와 비교하였다.

| 벤치마크 | bm1 | bm2 | bm3 | bm4 |
|---|---|---|---|---|
| 이름 | jam1.chs [5] | GMDF-alpha [4] | [6] | elliptical filter [2] |
| 노드개수 | 7 | 9 | 14 | 34 |

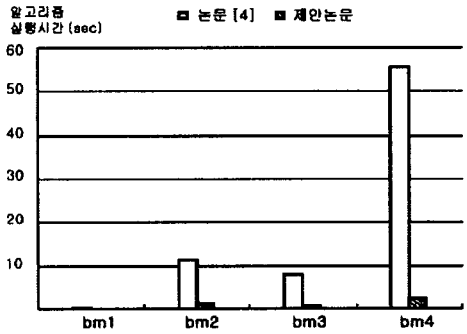
그림[6] 알고리즘 평가를 위한 벤치마크

각각의 벤치마크에 대한 실험결과 제안된 논문은 주어진 시스템 시간 제약을 만족하였으며 분할 이후의 전체 시스템 구현 비용은 그림[7] (a)와 같이 논문[4]의 분할 결과에 상응하며 분할을 위한 알고리

즘 실행시간은 그림[7] (b)와 같이 노드의 개수가 증가함에 따라 급격한 효율을 보임을 확인할 수 있다.




(a) 시스템 구현비용




(b)알고리즘 실행시간

그림[7] 알고리즘 실행시간 및 결과비용 비교

## 4. 결론

본 논문에서는 시스템 수준 자동설계의 중요한 문제인 분할 알고리즘에 대하여 연구하였으며 제약시간 하에서 저비용의 시스템 합성을 가능하게 하는 낮은 복잡도의 알고리즘 개발을 위하여 상대적 스케쥴 긴박도를 계산하는 비용함수를 제안하였다. 제안된 비용함수 계산에 의한 알고리즘 성능은 기존 논문[4]에 비해 분할 알고리즘의 복잡도 저하, 제약사항 만족 하에서의 설계 비용 저하 등의 효율을 보였다. 향후 연구과제는 효율적이고 현실성 있는 내장 시스템 개발을 위해 분할 대상 노드의 기본 블록 크기의 동적 변화, 각 태스크 노드에 대한 시간제약 만족, 인터페이스로 인하여 발생하는 통신상의 지연시간 고려, 다양한 벤치마크를 통한 현실성 검증 등에 대한 지속적인 연구가 필요로 되어진다.

## 참고문헌

[1] Giovanni De Micheli, Mariagiovanna Sami, " Hardware/Software Co-Design." Kluwer Academic Publishers, 1997.

[2] PIERRE G. PAULIN, JOHN P. KNIGHT, "Force-Directed Scheduling for the Behavioral Synthesis of ASIC's," *IEEE Tran. on* CAD Vol. 8, NO. 6. June, 1989.

[3] Jinhwan Jeon, Kiyoung Choi, "An Effective Force-Directed Partitioning Algorithm for Hardware-Software Codesign," on TR report, School of Electrical Engineering, Seoul National Univ. May 30, 1997

[4] F.Rousseau, J. Benzakki, J-M. Berge, M. Israel, "Adaptation of Force-Directed Scheduling Algorithm for Hardware-Software Partitioning," *proc. of Sixth Int'l workshop on Rapid System Prototyping*, pp:33-37, June 1995.

[5] Hidalgo, J.L.; Lanchares, J., "Functional partitioning of hardware-software codesign using generic algorithms," *Proceedings of the 23rd EUROMICRO conference*, Pages:631-638, 1997

[6] J.A. Maestro, "New methodologies for Hardware-Software Codesign Partitioning to Avoid High Communication Overhead", Departmento de informatica y Automatica Universidad complutense de Madrid