내장형 시스템 설계를 위한 FDS 분할 알고리즘

*오 주 영° **박 도 순
*경인여자대학 전산정보과
**홍익대학교 컴퓨터 공학과
odid080@kic.ac.kr
dspark@cs.hongik.ac.kr

A partitioning algorithm for embedded system design using FDS

*Ju-Young Oh° **Do-Soon Park
*Dept. of Computer Science, Kyungin Woman's College
**Dept. of Computer Engineering, Hongik University

요 약

통합설계를 위한 대부분의 분할 알고리즘들은 분할과 스케줄링을 독립적으로 실행하므로 스케줄 결과에 따른 재분할의 잠재적인 오버헤드가 있다. 분할 단계에서 스케줄링을 함께 고려하는 FDS를 응용하는 방법은 분할할 노드를 선택하면서 동시에 그 노드가 스케줄 되어야 하는 제어구간을 함께 결정한다.

본 논문에서는 기존의 FDS 응용에 의한 분할 알고리즘[7]을 확장하는데, 목적 시스템으로서 하드웨어의 경우 여러 가지 구현 방법을 지원하고 소프트웨어의 경우 여러 개의 프로세서를 수용함으로써 다양한 하드웨어 구현 모듈과 프로세서에 의해 구성되는 내장형 시스템 설계에 적용될 수 있도록 하였다. 이를 위해 각각의 하드웨어 구현들과 여러 가지 프로세서들에서의 분포 그래프를 생성하고, 상대적 스케줄 긴박도를 구할 때 각 노드에 대해 해당 분할 영역에서의 실행 시간과 구현 비용을 고려하며 분할 영역간에 발생하는 통신 지연 시간을 힘 값에 반영하였다. 상대적 스케줄 긴박도를 이용한 분할은 스케줄과 분할이 동시에 이루어져서 기존의 분할 알고리즘[9]보다 낮은 시간 복잡도를 보인다.

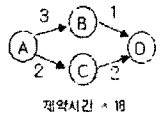
1. 서론

일반적인 내장형 시스템은 여러 가지 구현 방법에 의한 하드웨어부분과 여러개의 프로세서로 구성되며 이러한 내장형 시스템 설계에 있어서 복합적인 제약 사항을 만족하면서 최적의 결과를 찾는 분할 알고리즘의 복잡도는 NP-Complete[2] 문제이다. 따라서, 낮은 복잡도를 갖는 효율적인 휴리스틱 알고리즘을 개발하는 것이 필요하다. 주어진 제약 시간을 만족하는 최소비용의 시스템 분할을 위해 FDS를 응용하는 기존의 휴리스틱 알고리즘[4-8]은 주어진 제약 시간 내에서 스케줄 대상 노드를 각 제어 단계에 균등하게 분포시킴으로써 하드웨어 비용을 최소화하는 방법이다. FDS를 응용하는 알고리즘에서는 한 노드가 분할 영역의 특정 제어 단계에 분할되는 경우에 자신이 부과하는 설계 비용 상승분과 종속성을 갖는 다른 노드들의 제어 단계 감소를 반영하여 시스템에 대한 영향 값이 가장 적은 노드를 분할할 노드로 선택하여 분할한다. 논문[6]는 하나의 프로세서와 하나의 ASIC 구조에 대해 FDS를 응용하여 노드가 스케줄될 수 있는 분포 그래프의 모든 구간에서 스케줄에 미치는 힘 값으로 논문[5]에 정의된 상대적 스케줄 긴박도를 계산하여 가장 큰 값을 가지는 노드 하나를 분할 대상 영역으로 분할한다. 논문[6]의 알고리즘은 한 노드가 분할 영역의 분포 그래프 중에 특정 제어단계에 분할될 확률 값과 실행 시간, 구현 비용을 반영하여 자신의 힘 값을 계산하고 동일한 제어단계에 분할되려는 다른 노드의 힘 값들을 계산하여 상대적 스케줄 긴박도로 스케줄될 노드를 선택하였다. 논문[7]은 각 노드에 대한 힘 값의 계산을 분포 그래프의 최초 제어 단계(ASAP)와 최후 제어 단계(ALAP)로 제한함으로써 제약시간에 의해 부과되던 알고리즘 복잡도를 낮추었다. 이 방법은 노드의 개수를 n, 제어구간의 수를 c라고 하면 알고리즘이 한 번 반복되면 하나의 노드만이 분할되고 최초 제어 단계와 최후 제어 단계에 스케줄 될 수 있는 힘 값이 계산되며, 유도되는 힘 값은 최대 n-1개가 계산되므로 알고리즘의 복잡도는 $\theta(2\ n^2)$이 된다.

본 논문에서는 내장형 시스템 설계에 적용될 수 있도록 여러 구현방법을 갖는 하드웨어 부분과 여러 프로세서로 구성되는 타겟 아키텍처를 선택할 수 있는 낮은 복잡도를 갖는 FDS 응용 분할 알고리즘을 제안한다. 이를 위해서 논문[5]의 상대적 스케줄 긴박도와 논문[7]의 낮은 복잡도의 힘 값 계산 방법을 이용한다. 논문의 구성은 2절에서 제안 방법을 설명하고, 3절에서 실험 및 결과를 4절에서 결론을 각각 기술하였다.

2. 제안방법

본 논문의 목적함수는 시간 제약 조건하에서 설계 비용을 최소화하는 것으로 하였다. 이를 위해 논문[5]에 정의된 상대적 스케줄 긴박도를 사용하여 분할을 수행하는데, 상대적 스케줄 긴박도를 구할 때의 분할 영역에 따른 통신 지연 시간과 스케줄 가능성, 그리고 구현 비용을 함께 고려한다. 입력은 [그림 1]과 같이 방향성 비순환 그래프와 비용테이블이며 입력 그래프의 노드는 연산이나 작업을 나타내고 노드간의 간선은 연산이나 작업간의 종속성을 나타내며 간선에 부과되는 중량값은 통신상의 지연시간을 의미한다. 대상 구조에 대한 제한은 없으며 소프트웨어 영역에 분할되는 노드의 실행을 위한 프로세서와 하드웨어 영역에 분할되는 노드의 구현상의 다양성은 각 구현의 경우를 다중의 프로세서로 취급하여 문제를 단순화한다.



	P0(분할영역)			P1(1)	P2(5)
	B0	B1	B2		
A	3(4)	2(6)	1(10)	7	2
B	4(5)	2(8)	1(10)	10	3
C	2(5)	1(5)		5	2
D	5(10)	3(12)		15	5

시간(비용)

(a)입력그래프 (b) 비용 테이블

[그림 1] 입력 환경

노드간의 통신 지연시간은 종속 노드들이 분할되는 프로세서가 상이할 경우에만 반영하고, 같은 프로세서에 분할될 경우에는 통신 지연시간이 없다고 가정한다. 논문[5]에서 정의한 힘 값은 한 노드가 특정 분할 영역의 특정 제어단계에 스케쥴 되어야 하는 필요성의 정도를 의미한다. 즉, 실행시간이나 구현비용은 작으면서 스케쥴되는 제어구간이 짧고 상대적으로 스케쥴 경쟁 노드가 적어서 다른 노드의 스케쥴을 방해하는 힘이 가장 적은 노드가 우선 스케쥴될 수 있도록 한다. 자신의 힘 값은 특정 노드가 특정 제어 단계에 분포될 확률 값과 실행 시간, 구현 비용, 그리고 종속성이 있는 노드들이 서로 다른 영역으로 분할될 때 발생하는 통신 지연 시간을 반영하여 계산한다. 식(1)은 소프트웨어 분할영역 j에서 노드 i가 갖는 분포 확률을 나타낸 것이고 식(2)는 하드웨어 분할영역 j에서 노드 i가 갖는 분포 확률을 나타낸 것이다. 노드의 힘 값은 식(3), 식(4)와 같은데, 구현 비용과 실행 시간이 작을수록 큰 힘 값을 가지게 되고 통신 지연 시간이 없는 경우에 큰 힘 값을 가지게 된다. 따라서 제약 시간을 만족한다면 구현 비용이 싸고 실행 시간이 빠르며 상이한 분할간에 발생할 수 있는 통신 지연 시간이 최소화될 분할을 선택할 수 있도록 한다.

$$distr_{soft\,j}(i) = 1/(n+1-Time_{soft\,j}(i)) \quad --(1)$$

$$distr_{hard\,j}(i) = 1/(n+1-Time_{hard\,j}(i)) \quad --(2)$$

$$SelfForce_{soft\,j}(i) = (distr_{soft\,j} \times \frac{1}{Cost_{soft\,j}(i) \times (Time_{soft\,j}(i) + Time_{com}(i))}) \quad --(3)$$

$$SelfForce_{hard\,j}(i) = (distr_{hard\,j} \times \frac{1}{Cost_{hard\,j}(i) \times (Time_{hard\,j}(i) + Time_{com}(i))}) \quad --(4)$$

각 분할영역의 상대적 스케쥴 긴박도는 특정 노드 i를 특정 제어 단계 j에 분할한다고 가정하고 자신의 힘에서 특정 제어 단계 j에 분포되어 있는 다른 노드들의 힘을 합한 값을 감하여 나타낸다. 상대적 스케쥴 긴박도의 선택은 식(5)와 같으며 하드웨어 분할 영역의 상대적 스케쥴 긴박도와 소프트웨어 분할 영역의 상대적 스케쥴 긴박도는 각각 식(6), 식(7)과 같다.

$$Urgency(i) = Max\{Urgency^{h}_{soft\,j}(i),\ Urgency^{h}_{hard\,j}(i)\} \quad --(5)$$

$$Urgency^{h}_{hard\,j}(i) = SelfForce^{h}_{hard\,j}(i) - \sum_{k=1(k\neq i)}^{n} SelfForce^{h}_{hard\,j}(k) \quad --(6)$$

$$Urgency^{h}_{soft\,j}(i) = SelfForce^{h}_{soft\,j}(i) - \sum_{k=1(k\neq i)}^{n} SelfForce^{h}_{soft\,j}(k) \quad --(7)$$

제안 알고리즘은 [그림 2]와 같다. 단계 1에서 입력그래프와 제약 조건, 분할 영역별 노드의 수행 시간과 구현 비용을 입력으로 받는다. 단계 2에서는 입력된 제약 조건에 따라 각 분할 영역 별로 분포 그래프를 생성하고 단계 3-1에서 힘 값을 계산한다. 제약 시간에 따라 어떤 분할 영역에서는 분포 그래프가 생성되지 않을 수도 있으며 알고리즘 실행과정에서 어떤 노드의 분할 이후 제약 조건을 만족할 때 새로운 분포 그래프가 생성될 수 있다. 힘 값 계산은 분포 그래프가 생성된 모든 분할 영역에서 노드별로 최초 제어 단계와 최후 제어 단계에서만 계산되고 계산된 힘 값으로 단계 3-2에서 상대적 스케쥴 긴박도를 계산하여 가장 큰 상대적 스케쥴 긴박도를 갖는 노드를 선택하여 분할한다. 알고리즘은 한번 반복에 하나의 노드씩 분할되고 모든 노드가 분할될 때까지 반복된다.

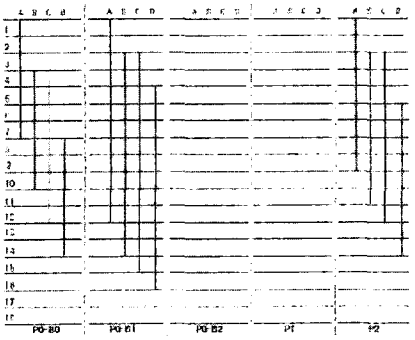
```
단계 1: 입력그래프, 제약조건 입력
단계 2: 분포그래프 생성
단계 3: for(분할 대상 노드)
  단계 3-1: 모든 분할 영역의 모든 노드에 대한 힘 값 계산
  단계 3-2: 최대의 스케쥴 긴박도를 가지는 노드를 분할
  단계 3-3: 분포그래프 수정
  단계 3-4: 분할 대상 집합에서 분할된 노드 삭제
```

[그림 2] 분할 알고리즘

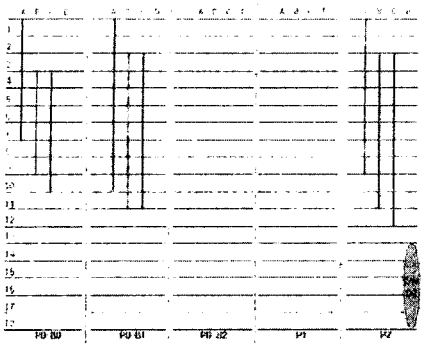
제안 분할 알고리즘의 복잡도는 입력 노드의 개수를 n이라 하고, 분할 영역의 개수를 p, 각 분할영역에서 구현 가능한 방법의 개수를 i라고 가정하면, 분할 결정을 위한 힘값 계산은 모든 노드에 대한 모든 분할 영역 p와 모든 구현 가능한 방법 i에 대해서 최초 제어 단계와 최후 제어 단계에서 계산된다. 따라서 한번의 알고리즘 반복에서 최대 $p \times i \times n$번의 힘 값 계산을 하게 되며, 모든 노드가 분할될 때까지 반복하므로 제안 알고리즘의 복잡도는 $\theta(pin^2)$이다.

3. 실험 및 결과

제안 분할 알고리즘의 실험을 위해 입력 그래프와 비용은 [그림 1]로 가정하고 시간 제약은 18제어단계로 가정하였다. 최초의 분포그래프는 [그림 3]과 같으며, 알고리즘의 첫 번째 루프에 의해 선택되는 노드는 구현 비용이 하드웨어에 비해 저렴하고 실행시간이 프로세서 P1에 비해 짧은 노드 D가 스케쥴 경쟁노드가 없는 P2의 최후제어단계에 분할되며 [그림 4]와 같다.

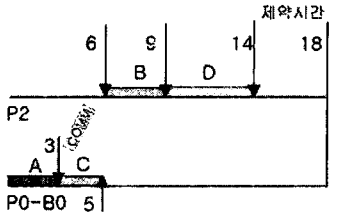


[그림 3] 최초 분포그래프

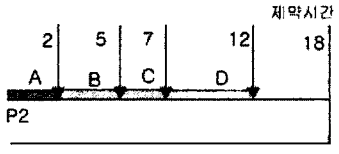


[그림 4] 노드 D 분할 이후 분포그래프

알고리즘 실행 결과 분할 및 스케줄 결과는 [그림 5]와 같다. 설계 결과는 논문[9]의 경우 12단위 비용이 소요되는 반면 제안 논문의 비용은 5단위 비용으로 현격히 줄일 수 있다. 반면, 노드와 분할영역, 노드와 분할영역 쌍에 대해서 가능성을 계산하여 대상 구조를 선택하고 또한 스케줄링 가능성을 따로 계산하는 논문[9]와 달리 제안 알고리즘은 분할영역별로 노드의 최초 제어 단계와 최후 제어 단계에 대한 힘 값을 계산을 하고 이것으로 대상구조 선택과 스케줄링을 동시에 고려하기 때문에 낮은 시간 복잡도를 가지게 된다. 노드의 수를 n, 분할영역의 수를 p, 논문[9]의 스케줄러의 복잡도를 O(S), 논문[9]의 노드-분할영역의 쌍에서 구현 방법을 i라고 하면 논문[9]의 시간 복잡도는 $\theta((npi)^2 S)$이고 제안논문의 복잡도는 $\theta(n p^2 i)$이 된다.



(a) 논문[9]의 결과



(b) 제안 알고리즘 결과

[그림 5] 분할 이후 스케줄 결과

성능평가를 위한 벤치마크는 논문[10]의 Yen's random 3이고 30개의 노드로 구성되었다. 제안 알고리즘에 의한 프로그램 실행후의 각 노드의 분할 및 스케줄링 결과는 논문[10]의 구현 비용 1628에 비해 높은 결과 값인 2503으로 나타나며. 구현 비용이 높게 나타나는 원인은 알고리즘의 시간 복잡도를 개선하기 위해 힘 값 계산 시에 노드의 스케줄로 인해 후위 노드에 미치는 영향 값을 반영하지 않았기 때문이다.

4. 결론

본 논문에서는 논문[7]에서 고려한 하나의 하드웨어 분할 영역과 하나의 소프트웨어 분할 영역을 확장하여 여러 구현 방법이 가능한 하드웨어 분할 영역들과 소프트웨어 실행을 위한 여러 종류의 프로세서들을 선택할 수 있도록 하여 내장형 시스템 설계에 적용할 수 있도록 하였다. 또한 다른 분할 영역간에서 발생하는 통신 지연 시간도 함께 고려하여 현실성 있는 하드웨어-소프트웨어 통합 설계가 가능하게 하였다. 논문[5]에서 정의된 상대적 스케줄 긴박도와 논문[7]에서 정의된 낮은 시간 복잡도의 힘 값 계산 방법을 이용하여 다중 이기종 프로세서로 볼 수 있는 여러 가지 구현 방법의 하드웨어와 여러 종류의 프로세서 사용이 가능한 하드웨어-소프트웨어 분할 시의 시간 복잡도를 낮추었다. 본 논문에서는 서로 다른 분할 영역에 있는 종속성이 있는 노드간의 통신 지연 시간은 동일하고, 통신 구현 비용이 없다는 가정 하에서 연구되었다. 그러나 실제로 여러 분할 영역에 따라 통신 지연시간과 통신 구현비용이 상이하므로 이를 반영할 수 있는 연구가 필요하며 노드의 수가 늘어남에 따른 효과적인 통신 지연시간의 고려가 계속 연구되어야 할 것이다.

참 고 문 헌

[1] F. Rousseau, J. Benzakki, J-M. Berge and M. Israel, "Hardware/Software Partitioning for Telecommunications systems", *RSP*, 1995.

[2] F. Vahid and T. D. Be, "Extending the Kernghan/Lin Heuristic for Hardware and Software Functional Partitioning", *Design Automation for Embedded Systems, special Issue : Partitioning Methods for Embedded Systems, Vol. 2, No. 2*, pp237-261, Kluwer Academic Publishers, March 1997.

[3] R. Ernst, J. Henkel, and T. Benner, "The COSYMA Enviorment for Hardware/Software Cosynthesis of small Embedded System", Microprocessor and Microsystem, *Vol. 20*, No. 3, 1996.

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

[5] 오주영, 박도순, "노드의 상대적 스케줄 긴박도 분석에 의한 하드웨어- 소프트웨어 분할", 한국정보처리학회 7권 2호, 2000.

[6] 오수희, "스케줄 긴박도에 의한 하드웨어 소프트웨어 분할", 홍익대학교, 2000

[7] 박효선, 오주영, 박도순, "FDS 응용에 의한 하드웨어 소프트웨어 분할 알고리즘의 시간 복잡도 개선", 한국정보과학회, 27권 2호, 2000

[8] Jinhwan Jeon, Kiyoung Choi, "An Effective Force-Directed Partitioning Algorithm for Hardware-Software Codesign", on TR report, SNU, May 1997.

[9] H. Oh, S. Ha, "A Haradware-Software Cosynthesis Technique Based on Heterogeneous Multiprocessor Scheduling", 7th International Workshop on Hardware /software Co-Design, 1999

[10] Ti-yen Yen, "Hardware-Software Co-Synthesis of Distributed Embedded Systems", *Kluwer Academic Publishers*, 1996.