제18회 한국정보처리학회 추계학술발표대회 논문집 제9권 제2호 (2002.11)

# 통합설계에서 파이프라인을 지원하는 분할 알고리즘에 관한 연구

·오주영, ··박도순
·경인여자대학, ··홍익대학교
email: ·odid080@kic.ac.kr, ··dspark@cs.hongik.ac.kr

# A partitioning algorithm that apply pipeline architecture in codesign

·Ju-Young Oh, ··Do-Soon Park
·Kyung-in woman's college
··Dept. of Computer Engeering, Hongik University

요 약

본 논문에서는 하드웨어/소프트웨어 시스템의 파이프라인 실행을 지원하는 알고리즘을 제안한다. 파이프라인 실행을 지원하기 위해 시간제약과 면적제약조건을 만족하는 분할 결과를 찾는 기존의 방법은 하드웨어/소프트웨어 분할과 파이프라인 스케줄링을 독립적으로 실행하였으며 최소시간의 파이프라인 입력간격으로부터 최적의 분할 결과를 얻기위해 점진적인 방법을 사용하기 때문에 많은 알고리즘 실행시간을 가진다.

본 논문에서는 분할 단계에서 스케줄링을 함께 고려하면서 최소 입력 간격을 갖는 파이프라인 실행을 지원하는 낮은 복잡도의 알고리즘을 제안한다. 이를 위해 최소입력간격에서의 파티션에 분포하는 노드와 종속성을 찾아서 하드웨어 구현과 프로세서에서의 분포 그래프를 생성하고, 상대적 스케줄 긴박도[8]를 구할 때는 노드별 실행시간과 구현비용을 고려하며, 분할 이후에 발생하는 통신 지연 시간을 힘 에 반영한다. 논문은 최소 입력 간격내에서 구성되는 파티션에 존재하는 노드의 파이프라인 스케줄과 시스템 제약시간을 만족하면서 구현비용을 저하시키기 위한 낮은 실행시간을 갖는 분할 알고리즘을 제안한다.

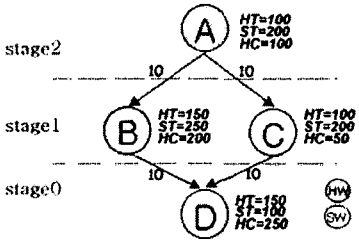
## 1. 서론

하드웨어/소프트웨어 통합설계에서 복합적인 제약 사항을 만족하면서 최적의 결과를 찾는 분할 문제를 해결하기 위하여 그리디(greedy), 클러스터링, 반복 개선, 수학적 프로그래밍 방법뿐만 아니라 설계 대상 시스템의 환경과 목적함수에 따른 다양한 휴리스틱 알고리즘들[1]이 개발되어 왔다. Kalavade[2]등은 하드웨어 자원과 실행시간을 최적화 하는 두 가지 목적함수를 정의하였는데, 전역시간이 임계점에 도달하면 목적함수는 실행시간의 감소가 되며, 그렇지 않은 경우에는 하드웨어 자원의 감소가 목적함수가 되도록 하여 분할을 한다. 이러한 기존의 분할 알고리즘들은 분할과 스케줄링 작업단계를 독립적으로 진행함으로써 분할 이후의 스케줄링과 자원할당으로 인하여 시스템 시간제약을 위배할때는 재분할하는 비용을 유발하게 되었다. 주어진 시간제약내에서 스케줄 대상 노드를 각 제어구간에 균등하게 분포시킴으로써 하드웨어 비용을 최소화하기 위한 FDS를 응용한 방법[3,4]은 분할과 스케줄링을 함께 고려하는데, Rousseau[3]등은 FDS를 응용할 때에 비용함수(실행시간 또는 설계비용)에 따라 계산되는 힘값을 갖고 분할단계에서 노드들중의 하나를 스케줄하며 분할하고, Choi[4]등은 시스템 실행시간과 구현 비용을 최소화하기 위하여, 처음에 모든 노드를 소프트웨어로 배정하고 시간제약을 만족할 때까지 한번에 하나의 노드(노드의 전체 모빌리티 구간에 걸쳐서 병행해서 실행될 수 있는 다른 소프트웨어 노드들의 분포비율을 계산하여, 하드웨어로 선택할 때 유발되는 비용상승을 반영한 힘을 감한 값이 가장 큰 노드를 선택)를 선택하여 하드웨어로 보내는 것을 반복한다. 이러한 알고리즘들의 대상 아키텍처가 하나의 프로세서와 하나의 ASIC 구조로 제한되었는데, Liu[5]는 두 개의 프로세서와 $k$개의 하드웨어 자원을 갖는 구조에 대해 처음에 모든 타스크를 소프트웨어로 분할하고 타스크들의 상대적 모빌리티 정보를 이용하여 하드웨어와 소프트웨어가 최대로 병렬 실행이 가능하도록 $k$개의 타스크들을 하드웨어로 이동하며 분할한다. 논문[6,7]은 파이프라인 스케줄링을 고려한 하드웨어 소프트웨어 분할을 연

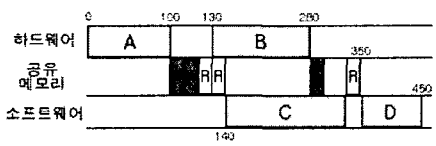
구하였는데, Bakshi[6]등은 파이프라인 스케줄을 위해 list 기반의 스케줄링 기법을 사용하였으며, Chatha[7]등은 시간제약 조건을 만족하면서 파이프라인의 입력 간격을 최소화하는 목적함수를 만족시키기 위해 branch-and-bound 기법을 사용하는데, 알고리즘의 복잡도가 높은 단점이 있다. 본 논문에서는 하드웨어/소프트웨어 통합시스템의 파이프라인 실행을 지원하기 위해 기존[8]의 힘 계산방법을 수정하여 분할에 따른 통신지연시간을 함께 고려하는 낮은 복잡도의 분할 알고리즘을 제안한다. 2절에서 제안 알고리즘을 설명하고, 3절에서 실험 및 결과를 4절에서 결론을 각각 기술하였다.

## 2. 제안 알고리즘

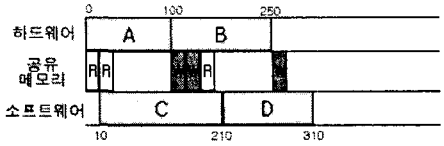
그림[1](a)와 같은 입력 DAG환경에 대해 시간제약 450으로 순차실행을 지원하는 최적의 분할 및 스케줄 결과는 그림[1](b)와 같다. 자료종속성에 대한 데이터의 읽고 쓰기는 상호 배타적으로 수행된다고 가정할 때 순차 실행에서의 최대 공유 메모리 요구량은 30개 변수가 된다. 그림[1](a)와 같이 노드 A, B가 하드웨어로 분할되고 3개의 파이프라인 스테이지로 각각 구분될 때 동일한 시간제약내에서 어떻게 분할되는가에 따라서 그림[1](c), 그림[1](d)와 같이 스케줄 결과가 달라지므로써 설계비용과 파이프라인 입력간격이 달라질 수 있다. 따라서, 본 논문에서는 최소 입력 간격하에서의 파이프라인 실행을 지원하기 위해 논문[8]의 힘 계산방법을 응용한 분할 알고리즘을 제안한다.



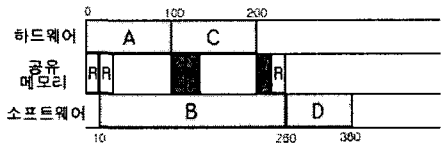

(a) 입력그래프




(b)순차실행



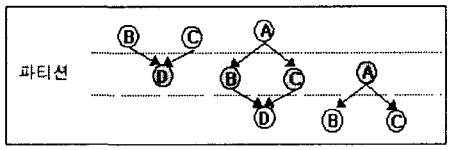

(c) 파이프라인 실행 1




(d) 파이프라인 실행 2

그림[1] 입력그래프와 스케줄링

논문의 목적함수는 최소 시간 의 파이프라인 입력 간격을 만족하면서 설계비용을 최소화하는 분할의 해를 찾는 것이다. 제안 알고리즘은 고정된 타겟 아키텍처를 가정하는데, 하나의 일반적인 프로세서와 하나의 확장가능한 하드웨어 및 통신을 위한 공유메모리로 구성된다고 가정하였다. 공유메모리는 소프트웨어 실행을 위한 프로세서와 하드웨어 모듈간의 통신이나 하드웨어 모듈간의 통신에 사용되며, 배타적 읽기와 배타적 쓰기로 가정한다. 또한, 소프트웨어 실행을 위한 프로세서 내의 통신은 프로세서의 지역메모리를 통해 실행되며, 통신지연시간은 노드의 실행시간을 초과하지 않는다고 가정하며 하드웨어 노드는 순차실행을 하는 것으로 가정한다. 입력은 그림[1](a)와 같은 방향성 비순환 그래프와 비용테이블이며 입력 그래프의 노드는 연산이나 작업을 나타내고 노드간의 간선은 연산이나 작업간의 종속성을 나타내며 간선에 부과되는 값은 통신상의 지연시간을 의미한다. 하드웨어 노드가 병렬로 실행될 수 없다고 가정하면 그림[2]와 같이 입력그래프의 소스노드로부터 종속성을 기준으로 레벨별 파티션을 결정하면, 최소 입력 간격은 같은 파티션을 가지는 노드들의 최소실행시간의 크게된다. 상대적 스케줄 긴박도[8]는 제약시간내에서 설계 비용 최소화를 위한 기법인데, 파이프라인 실행을 지원하는 분할 방법에서는 최소의 입력 간격으로부터 파티션과 파티션에 분포하는 노드간의 종속성을 구분함으로써 분할과 스케줄링을 동시에 수행할 수 있도록 한다.




그림[2] 파이프라인 파티션

파이프라인 스케줄링을 위한 힘 계산은 파티션에 분포하는 노드들중에서 분할과 스케줄의 우선순위를 결정하기 위한 힘이다. 즉, 실행시간이나 구현비용은 작으면서 스케줄되는 제어구간이 짧고 다른 노드의 스케줄을 방해하는 힘이 가장 적은 노드가 우선 스케줄될 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(2))은 소프트웨어(하드웨어) 분포그래프에서 노드 $i$가 갖는 모빌리티 값인 분포 확률을 나타낸 것이다. 스케줄 긴박도는 어떤 노드를 먼저 스케줄해야 하는지를 결정하는 값이며 식(3(4))와 같다. 식(3(4))는 노드 $i$를 제어단계 $j$에 하드웨어(소프트웨어)로 구현하는 경우의 타당성 정도를 나타낸다. $n_{soft_i}$( $n_{hard_i}$)는 노드 $i$의 소프트웨어(하드웨어)분포그래프에서의 모빌리티를 나타내며, $Cost-function$은 하드웨어(소프트웨어)로 분할할때의 구현비용(실행시간)을 의미한다. $cr_i$는 노드 $i$의 하드웨어 실행시간( $HT_i$) 대비 통신시간( $comm_i$)의 비율 값을 나타내며, max($cr$)은 각 노드의 통신 비율값에서 최대값을 의미한다. 따라서, 자신의 힘은 구현 비용과 실행 시간이 작을수록 큰 힘 값을 가지게 되고 통신 지연 시간이 발생하지 않을 때 큰 힘 값을 가지게 된다. 따라서 제약 시간을 만족한다면 구현 비용이 싸고 실행 시간이 빠르며 분할간에 발생할 수 있는 통신 지연 시간이 최소화될 수 있도록 힘을 계산한다.

$$distr_{soft}(i) = 1/(n_{soft_i}) \quad --(1)$$

$$distr_{hard}(i) = 1/(n_{hard_i}) \quad --(2)$$

$$SelfForce^{i}_{hard} = distr_{hard} \times \frac{1}{Cost-function(i, hw-implement)} \times \alpha\ ST_i \times \frac{\delta\ cr_i}{\max(cr)} \quad --(3)$$

$$SelfForce^{i}_{soft}(i) = distr_{soft} \times \frac{1}{Cost-function(i, sw-implement)} \times \beta\ HC_i \times \delta\ \frac{1}{comm_i} \quad --(4)$$

각 분할영역의 상대적 스케줄 긴박도는 노드 $i$를 제어단계 $j$에 분할한다고 가정하고 자신의 힘에서 제어 단계 $j$에 분포되어 있는 다른 노드들의 힘을 합한 값을 감하여 나타낸다. 상대적 스케줄 긴박도의 선택은 식(5)와 같으며 하드웨어 분할 영역의 상대적 스케줄 긴박도와 소프트웨어 분할 영역의 상대적 스케줄 긴박도는 각각 식(6), 식(7)과 같다.

$$Urgency_j(i) = \max\{\ Urgency^{i}_{soft}(i),\ \ Urgency^{i}_{hard}(i)\} \quad --(5)$$

$$Urgency^{i}_{hard}(i) = Self\_force^{i}_{hard}(i) - \sum_{k=1,(k\neq i)}^{n} Self\_Force^{i}_{k}(k) \quad --(6)$$

$$Urgency^{i}_{soft}(i) = Self\_force^{i}_{soft}(i) - \sum_{k=1,(k\neq i)}^{n} Self\_Force^{i}_{k}(k) \quad --(7)$$

제안 알고리즘은 그림[3]과 같다. 단계 1에서 입력그래프와 제약 조건, 분할 영역별 노드의 수행 시간과 구현 비용을 입력으로 받는다. 단계 2에서는 입력그래프의 각 노드의 레벨을 계산하고 파티션에 나타나는 노드와 노드간의 종속성을 찾는다. 단계 3에서 입력그래프의 하드웨어 실행시의 임계경로시간을 파이프라인의 최소 입력 간격으로 설정한다. 입력된 제약 조건에 따라 하드웨어와 소프트웨어의 각 분할 영역 별로 분포 그래프를 생성하고 단계 4-1에서 힘 값을 계산한다. 소프트웨어 분포그래프는 시간제약에 따라 분포 그래프가 생성되지 않을 수도 있으며 알고리즘 실행과정에서 어떤 노드의 분할 이후 제약 조건을 만족할 때 생성될 수 있다. 힘 값 계산은 분포 그래프가 생성된 모든 분할 영역에서 노드별로 계산되고 계산된 힘 값으로 단계 4-2에서 상대적 스케줄 긴박도를 계산하여 최대치를 갖는 노드를 분할 및 스케줄한다. 알고리즘은 한번 반복에 하나의 노드씩 분할되고 모든 노드가 분할될 때까지 반복된다. 제안 분할 알고리즘의 복잡도는 입력 노드의 개수가 $n$, 생성되는 파이프라인 입력 간격의 제어단계수를 $c$라고 하면, 분할을 결정하기 위한 힘값 계산은 모든 노드에 대한 모든 분할 영역과 각 노드의 모든 제어단계에서 계산되며, 한번의 알고리즘 반복에서 하나의 노드를 분할하고 모든 노드가 분할될 때까지 반복하므로 알고리즘의 복잡도는 $o(cn^2)$이다.

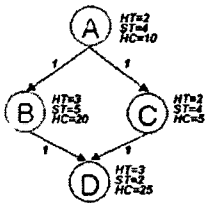
```
단계 1: 입력그래프, 노드의 환경 입력
단계 2: 입력노드의 레벨 계산, 파티션 생성
단계 3: 최소 입력간격 계산
단계 3: 분포그래프 생성
단계 3: for(분할 대상 노드)
  단계 4-1: 모든 분할 영역의 모든 노드에 대한 힘 값 계산
  단계 4-2: 최대의 스케줄 긴박도를 가지는 노드를 분할
  단계 4-3: 분포그래프 수정
  단계 4-4: 분할 대상 집합에서 분할된 노드 삭제
```

그림 [3] 분할 알고리즘

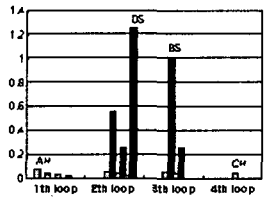
## 3. 실험 및 결과

입력 그래프와 노드의 비용 및 통신지연시간이 그림[4](a)와 같을 때 통신시간을 반영한 하드웨어 실행시의 임계경로 시간은 10이고 알고리즘 반복시점마다 계산되는 힘값 분포는 그림[4](b)와 같다. 알고리즘의 최초 반복에서는 제약시간으로 인해 소프트웨어 분포그래프가 생성되지 않고, 하드웨어 분포그래프의 힘 계산에서 모빌리티가 짧고, 구현비용이 상대적으로 작은 노드 A가 하드웨어로 분

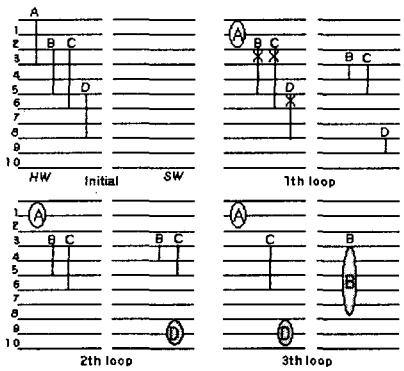
할되며 다른 노드의 스케줄을 상대적으로 가장 적게 방해하게되는 제어단계 1에 스케줄된다.



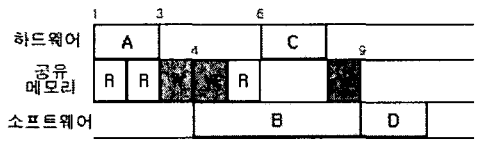

(a) 입력환경




(b) 알고리즘 반복에서의 힘값




(c) 분할과정




(d) 분할결과

[그림 5] 분할 과정 및 결과

알고리즘의 최초실행 이후 A노드의 하드웨어 분할 이후 통신 지연시간 만큼 종속성을 가지는 노드의 모빌리티 그래프를 수정하고, 노드 A의 하드웨어 결정에 의해 다른 노드들의 소프트웨어 분포그래프가 생성되며 두 번째 알고리즘 반복에서 노드 D가 소프트웨어로 분할된다. 알고리즘의 실행결과는 그림[5](d)와 같이 최소입력 간격의 시간제약을 충분히 사용하였으며 분할 결과는 시간제약을 만족하는 최소비용 결과와 동일하다.

## 4. 결론

본 논문에서는 기존의 분할 알고리즘들이 순차실행을 하는 타겟 아키텍처를 대상으로 진행되던 것을 확장하여 파이프라인 실행을 지원하는 타겟 아키텍처를 대상으로 분할과 스케줄링을 동시에 수행하는 낮은 복잡도의 알고리즘을 제안하였다. 이를 위해 최소입력간격을 시간제약으로 하여 분할 및 스케줄을 하기 위해 상대적 스케줄 긴박도[8]를 응용하였다. 현재까지는 하드웨어 부분의 병렬실행을 허용하지 않고 통신지연 시간이 노드의 실행시간을 넘지 않는 것으로 가정하였으며 하나의 하드웨어 모듈과 하나의 프로세서에 대한 파이프라인 실행을 가정하였다. 따라서, 복합적인 현실 시스템에 적용하기 위한 대상 시스템 고려와, 다양한 벤치마크의 실험을 통한 검증을 향후 연구과제로 한다.

### 참 고 문 헌

[1] X. Hu, J. G. D'Ambrosio, B. T. Murray, and D.-L. Tang, "Codesign of architectures for powertrain modules." *IEEE Micro, vol. 14, no. 4,* pp. 48-58, Aug. 1994.

[2] A. Kalavade and E.A. Lee, "A Global Critically/Local Phase Driven Algorithm for the Constrained Hardware/Software Partitioning Problem," 3th *International Workshop on Hardware/Software Codesign,* pp. 42-48, Grenoble, 1994.

[3] F. Rousseau, J. Benzakki, J-M. Berge, M. Israel, "Adaptation of Force-Directed Scheduling Algorithm for Hardware/Software Partitioning," Rapid System Prototyping, 1995.

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

[5] Huiqun Liu. D.F., "Integrated Partitioning and Scheduling for Hardware/Software Co-design," *IEEE Proc. of Int'l Conference on Computer Design : VLSI in Computers and processors,* Pages: 609-614, 1998.

[6] S. Bakshi and D. D. Gajski, " A Scheduling and Pipeline Algorithm for Hardware/Software Systems," *Proceedings of 10th International Symposium on system synthesis.* Antwerp, Belgium. September 1997

[7] "A Tool for Partitioning and Pipelined Scheduling of Hardware-Software System" .

[8] Juyoung Oh, Hyosun Park, Dosoon Park, "Improvement of time complexity of Hardware-Software partitioning algorithm using FDS," *in Proc. 27th KISS, vol. 27, no. 2,* pp. 24-26, Oct. 2000.