FDS 응용에 의한 하드웨어 소프트웨어 분할 알고리즘의 시간 복잡도 개선

오 주 영° 박효선 박 도 순
홍익대학교 컴퓨터 공학과
{jyoh, hspark, dspark}@cs.hongik.ac.kr

Improvement of time complexity of Hardware-Software partitioning algorithm using FDS

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

요 약

본 논문에서는 FDS를 응용한 하드웨어 소프트웨어 분할 방법을 강 제약 조건을 만족하면서 FDS를 응용하는 방법보다 낮은 복잡도의 분할 알고리즘을 제안한다. 기존의 FDS 응용 방법은 힘값 계산에서 종속성에 의해 후위 연산이 받는 영향값을 계산하여야 하므로 이로 인한 시간 복잡도가 가중되었다. 본 논문에서는 이러한 복잡도를 저하시키기 위해 노드의 분포 그래프와 구현에 소요되는 비용, 그리고 해당 파티션에서의 실행시간 등에 의해 상대적 긴박도를 정의하여 분할을 수행하지만, 종속성 검사는 종속성 제약조건에 의한 분포그래프의 변화와 스케쥴에 대해서만 고려되며 힘값 계산에는 고려하지 않는다. 또한, 분할 단계에서 스케쥴링을 함께 고려함으로써 합성 이후에 재 스케쥴링의 부하를 경감할 수 있도록 하였다. 제안 알고리즘 결과는 ILP 결과와 비교 분석하였다.

1. 서론

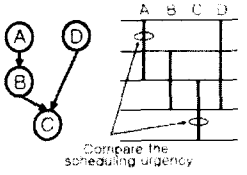
시스템 통합 설계는 내장형 시스템을 위한 일반적인 설계 방법이다. 통합 설계에 의해 합성되는 시스템의 성능을 좌우하는 최대 요소는 시스템 기능을 서술하는 각각의 행위 기술들을 하드웨어 부분과 소프트웨어 부분에 할당될 영역을 설정하는 것이며, 이러한 일련의 작업 과정이 분할이다. 내장형 시스템의 하드웨어/소프트웨어 분할을 위한 다양한 제약 사항과 목적 함수를 갖는 많은 알고리즘들이 개발되어왔다. Gupta[1]에서는 데이터 의존 지연 연산을 제외한 모든 연산을 처음에 모두 하드웨어로 설정하고, 주어진 시간 제약 조건을 만족하는 범위에서 한번에 하나의 연산을 greedy 알고리즘으로 선택하여 소프트웨어로 보내는 방법을 사용한다. 이러한 방법은 탐색된 해가 국부 최적치로 제한될 수 있으므로 Olokutun[2]는 하드웨어 기술언어에 의해 기술된 명세로부터 기본 스케쥴링 블록을 추출하고, 각각의 기본 블록에 대해 list 스케쥴링으로 계산되는 면적에 의한 하드웨어로의 합성 타당성과 하드웨어 구현시 부과되는 통신비용에 의해 대략적인 분할을 수행하고, 시간제약을 만족하는 최소면적의 블록을 빠른시간에 탐색할 수 있도록 이분검색 등의 방법을 취하였다. Kalavade[3]는 임계경로상의 시간적 부담이 큰 노드에 대한 하드웨어 매핑비율인 GC값과 구현상의 난이도에 따라 결정되는 각 노드의 특성 값인 LP값을 더하여 이를 주어진 임계값과 비교하여, GCLP값이 크면 하드웨어로 작으면 소프트웨어로 각각 매핑하는 함수를 사용하여 분할한다. Liu[4]는 상위수준 합성에서 사용된 두 개의 프로세서에 대한 최적 스케쥴 기법을 사용하여 분할 단계에서 스케쥴링을 함께 고려한다. Choi[5]는 FDS[6]를 응용하여 하드웨어로 매핑할 노드를 선택할 경우에 소프트웨어 노드와의 병렬 실행 가능성을 힘값으로 하여 분할을 수행하였으며, Rousseau[7]는 하드웨어 또는 소프트웨어 제어 구간에 스케쥴할 때 자신의 구현 비용이 반영된 힘값과 종속성에 의해 여타 노드가 받는 힘값을 반영하여 분할을 수행하였다. 상위의 알고리즘들은 최적해를 탐색하기 위한 분할 알고리즘 복잡도를 최소화하기 위하여 휴리스틱[1] 방법을 적용하였으며 분할 단계에서 스케쥴링을 함께 고려[2-6]함으로써 분할 이후에 스케쥴 불가능으로 인한 재분할의 문제를 해결하는 방법으로 진행되었다. 본 논문에서는 기존의 FDS 응용 알고리즘[7]의 시간 복잡도를 낮출 수 있도록 하고 강시간 제약시간(HardDeadline)을 갖는 내장형 시스템에 적용될 수 있도록 하기 위하여 개선된 FDS 응용 방법을 제안한다. 2절에서 제안 분할 알고리즘을, 3절에서 알고리즘 성능평가를 4절에서 결론을 각각 기술하였다.

2. 제안 하드웨어/소프트웨어 분할

FDS 응용[7]에 의한 분할에서는 모든 입력 노드가 하드웨어/소프트웨어로 각각 구현될 경우를 고려한다. 이를 위하여 ASAP스케쥴과 ALAP스케쥴에 의해 구성되는 각각의 분포그래프에서 한 노드가 특정 제어구간에 스케쥴되어 특정 파티션으로 분할될 경우에 시스템이 받는 영향과 스케쥴 이후에 종속성에 의한 시간 지연을 반영하여 스케쥴링을 함께 고려할 수 있도록 하였다. 따라서, 특정 제어구간에 스케쥴될 확률 값에 구현비용을 반영한 자신의 힘값과 스케쥴 이후에 종속성에 의해 제거되는 노드의 힘값을 합한다. 각 분포그래프의 힘값들 중에서 최소 값을 갖는 노드를 선택하여 해당 제어단계의 특정 파티션으로 분할을 수행하였다. 이 방법의 시간 복잡도는 제어구간의 수가 c이고 노드 개수가 n일 경우, 알고리즘이 한번 반복될 때에 하나의 노드만이 분할되는데, 각 제어구간에 배정될 수 있는 모든 노드의 힘값이 계산되며, 종속성에 의한 영향은 최악의 경우에 n-1개에 대하여 계산되므로 복잡도는 $\theta(c^2 n^3)$이다.

2.1 제안 방법

본 논문에서는 주어진 시간 제약 조건하에서 시스템 설계비용을 최소화하는 목적 함수를 만족시키기 위해, 각 노드의 상대적 스케쥴 긴박도를 정의하고 이에 기반 하여 강한 제약 시간을 갖는 내장형 시스템 합성을 위한 분할알고리즘을 제안한다.



그림[1] 노드의 제어단계별 상대적 스케쥴 긴박도

노드의 제어단계별 상대적 스케쥴 긴박도는 그림[1]에 의해 설명된다. 각 노드의 힘은 입력 그래프의 종속성을 기반으로 한 분포 그래프의 확률값에 비례하므로, 모빌리티가 적고 비용이 적은 노드가 큰 힘을 갖게 한다. 또한, 같은 제어 구간에 스케쥴 되려는 여타의 노드에 미치는 스케쥴 방해값이 상대적으로 적은 노드가 큰 힘값을 갖게 한다. 그림[1]의 노드 A와 C의 경우에 두 제어 구간에서 자신의 힘은 같지만, A 노드의 제어 구간의 경쟁 노드 수가 C 노드의 경쟁 노드 수보다 많으므로 상대적인 긴박도는 C 노드의 네 번째 제어 구간이 더 크게 된다. 긴박도에 의한 스케쥴 우선순위 부여는 스케쥴되지 않은 다른 노드의 모빌리티를 유지시킴으로써 전체적인 노드의 스케쥴 가능성을 향상시킬 수 있도록 하며, 이는 소프트웨어 할당의 가능성을 재고함으로써 설계비용의 저하를 기할 수 있도록 한다.

2.1.1 사용 함수

비용 함수 계산을 할 때에 사용되는 변수 HT_i, ST_i, HC_i, SC_i는 각각 노드 i의 하드웨어 실행 시간, 소프트웨어 실행시간, 하드웨어 구현비용, 소프트웨어 구현 비용을 의미한다. 식 (1)은 노드 i의 분포 확률을 나타내며, (2)와 (3)은 확률과 비용, 실행 시간을 반영한 소프트웨어와 하드웨어에서의 자신의 힘값을 계산한다. 모빌리티의 범위가 작은 노드일수록 큰 힘을 갖도록 하고, 하드웨어의 경우에는 구현비용을, 소프트웨어의 경우에는 프로세서에서의 실행 시간을 각각 반영하여 성능대비 스케쥴 가능성의 효율성이 반영될 수 있도록 하였다. 식 (4), (5)는 여타 노드에 대한 자신의 스케쥴 긴박도의 상대적인 값이며 α, β의 가중치에 의해 비용과 실행시간을 반영하여 하드웨어 선택의 경우에는 소프트웨어 실행시간이 큰 노드를, 소프트웨어 선택의 경우는 하드웨어 구현 비용이 큰 노드가 각각 선택될 수 있도록 하였다. 이러한 계산 방법에 의해서 각각의 제어단계별, 노드별, 분포그래프별로 계산된 힘값중에서 식 (6)과 같이 최대치를 갖는 노드를 해당 파티션의 특정 제어 구간으로 분할하는 방법을 취하였다.

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

$$Self-Force\ '_{soft}(i) = distr_i \times \frac{1}{Cost-function(i,\ hardware-implementation)} \qquad \text{--(2)}$$

$$Self-Force\ '_{hard}(i) = distr_i \times \frac{1}{Cost-function(i,\ software-implementation)} \qquad \text{--(3)}$$

$$Urgency\ '_{soft}(i) = (Self-Force\ '_{soft}(i) - \sum_{k=\text{first}(i)}^{\text{last}(i)} DistrOfSDG-Force\ '_k) + \beta \times HC_i \qquad \text{--(4)}$$

$$Urgency\ '_{hard}(i) = (Self-Force\ '_{hard}(i) - \sum_{k=\text{first}(i)}^{\text{last}(i)} DistrOfHDG-Force\ '_k) + \alpha \times ST_i \qquad \text{-(5)}$$

$$Urgency_f(i) = \max\{Urgency\ '_{soft}(i),\ Urgency\ '_{hard}(i)\} \qquad \text{-(6)}$$

2.1.2 제안 알고리즘

분할을 위한 가정으로써 타겟 아키텍쳐는 소프트웨어 실행을 위한 하나의 프로세서와 확장 가능한 ASIC 모듈로 하였다. 또한, 프로세서의 실행은 순차 실행으로 가정하였으며 통신에 소요되는 비용은 분할 결과에 크게 영향을 주지 않는 것으로 하였다. 본 논문에서 제안하는 강한 시간제약을 갖는 내장 시스템을 위한 분할 알고리즘은 그림[2]와 같다. Step 1, 2 에서 주어진 시간 제약 조건을 만족하는 하드웨어 분포 그래프를 생성하고, Step 3에서 각 노드별, 분포그래프별, 제어단계별 힘을 구하고 최대값을 갖는 노드를 해당 영역의 제어 구간으로 분할한다. Step 3-4에서 임의의 노드가 할당됨으로써 실행 시간 제약 조건을 만족하면서 소프트웨어로의 구현이 가능한 노드의 소프트웨어 분포 그래프를 생성하고, Step 3을 반복 수행한다.

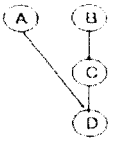
Step 1 : DAG 및 노드특성, 시간 데드라인
Step 2 : 최초 HW DG 생성
Step 3 : for (미분할 노드)
Step 3-1 : Force 계산
Step 3-2 : 최대 Force 노드 선택 후 해당 영역으로 분할
Step 3-3 : HW DG, SW DG의 모빌리티 삭제
Step 3-4 : 구현 가능한 SW DG 생성
Step 3-5 : 분할된 노드를 대상 노드 집합에서 삭제

그림[2] 제안된 분할 알고리즘

2.1.3 시간 복잡도

힘값은 노드별, 제어단계별, 분포그래프별로 계산되고 알고리즘의 매 반복 시점마다 하나의 노드가 분할되므로, 알고리즘의 시간 복잡도는 노드 개수를 n, 제약시간이 c 제어구간일 경우에 $\theta(c\,n^2)$이다.

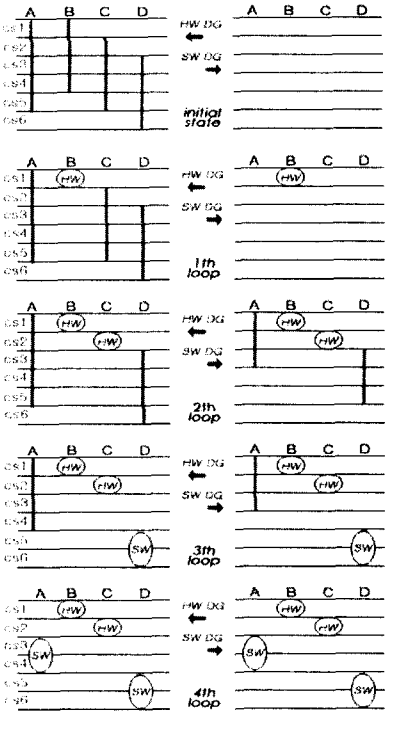
2.2 제안 알고리즘에 의한 분할



노드	HC_i	HT_i	SC_i	ST_i
A	2	1	1	2
B	2	1	1	4
C	2	1	1	3
D	5	1	1	2

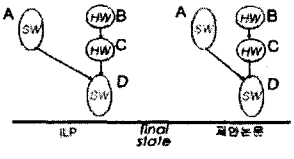
그림[3] 입력 DAG와 각 노드의 비용

그림[3]과 같은 입력조건에 제약시간이 6제어단계라고 가정한 경우에 제안된 알고리즘의 분할 과정은 그림[4]와 같다.



그림[4] 제안된 알고리즘 반복에 의한 분할 과정

입력 DAG와 시간제약에 대한 각 노드의 ASAP, ALAP 스케쥴에 의한 최초 분포그래프 생성에서 소프트웨어 분포 그래프의 경우는 시간제약 조건을 만족하지 못하므로 생성되지 않는다. 알고리즘의 첫 번째 반복이후, B노드의 하드웨어 분할이 결정되고, 두 번째 알고리즘 반복에서 C노드가 하드웨어로 분할 이후에 노드 D가 종속성 제약조건을 만족하는 범위에서 소프트웨어로 실행이 가능하므로 소프트웨어 분포그래프를 생성하고, D의 분포그래프 생성 이후 프로세서 순차실행을 만족하는 범위에서 A의 소프트웨어 분포그래프가 생성된다. 세 번째 반복에 의해 D노드의 소프트웨어 분할 이후 종속성 제약조건에 따라 A노드의 하드웨어 분포그래프를 수정하게 된다. 알고리즘 반복에 의한 최종 분할 결과는 그림[5]와 같다.



그림[5] 분할 결과

3. 알고리즘 성능평가

제안 알고리즘의 성능 평가를 위하여 알고리즘 실행시간, 알고리즘 결과에 의한 구현 비용에 대해 그림[6]과 같은 각각의 벤치마크의 결과를 ILP와 비교분석 하였으며 실험은 PentiumII에서 Visual C++로 수행하였다.

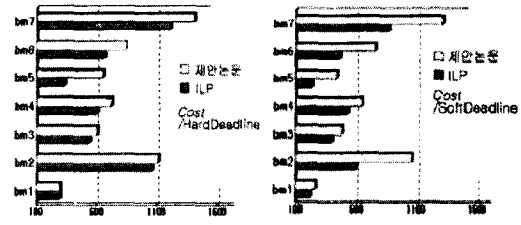
벤치마크	노드개수
jam1.chs[8]	7
GMDF-alpha[7]	9
논문[9]	14
3X3-det[10]	14
dhrc[10]	16
FIR filter[11]	23
AR-lattice filter[12]	28

그림[6] 알고리즘 성능 평가를 위한 벤치마크

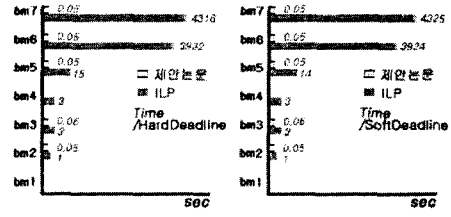
각각의 벤치마크에 대한 시간제약은 전체노드의 하드웨어 구현시 임계경로 시간을 강한 제약시간으로 부여하고, 전체 노드의 소프트웨어 실행시의 임계경로 시간과 강한 제약시간의 평균 시간을 약 제약시간(SoftDeadline)으로 부여하여 실험하였으며, 분할 결과에 의한 시스템 합성 비용과 알고리즘 실행에 소요되는 CPU 시간의 실험 결과는 그림[7]과 같다. 그림[7] (a)의 결과 그래프는 시간제약이 강할수록 ILP결과에 근접해 짐을 보이고, 그림[7] (b)에서 알고리즘 실행 시간은 현저히 낮음을 알 수 있다.

4. 결론

본 논문에서는 FDS를 응용하는 분할 방법에서 기존의 방법보다 낮은 시간 복잡도의 분할 알고리즘 개발을 연구하였다. 이를 위해 노드의 분포 그래프와 노드간 종속성, 그리고 구현시에 소요되는 비용 등에 의해 상대적 긴박도를 정의하여 분할을 수행하였다. 알고리즘은 분할 단계에서 스케쥴링을 함께 고려함으로써 합성 이후에 재 스케쥴링의 부하를 경감할 수 있도록 하였고 기존의 알고리즘 대비 현저한 복잡도 저하를 기할 수 있었다. 향후 연구 과제는 현실성 있는 내장 시스템 개발을 위해서 각 노드 사이의 인터페이스로 인하여 발생하는 통신상의 지연 시간을 고려해야 하며, 알고리즘 실행 결과의 성능향상을 위한 비용함수 개선에 대한 지속적인 연구가 필요로 되어진다.



(a) 시스템 설계비용



(b) 알고리즘 실행시간

그림[7] 벤치마크를 통한 결과 비교

참고문헌

[1] R.K. Gupta, C. Coelho, and G. De Micheli, "Synthesis and Simulation of Digital Systems Containing Interacting Hardware and Software Components," *29th ACM, IEEE Design Automation Conference*, pages 225-230, 1992.

[2] K. Olokutun, R. Helaihel, J. Levitt and R. Ramirez, "A software-hardware cosynthesis approach to digital system simulation," *IEEE Micro*, vol. 14, no. 4, pp. 48-58, Aug. 1994.

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

[4] 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.

[5] 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

[6] 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.

[7] 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.

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

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

[10] Chuck Monahan, Forrest Brewer, "Scheduling and binding bounds for RT-level symbolic execution," *Proc of the International Conference on Computer-Aided Design*, 1997

[11] Ki soo Hwang, Albert E. Casavant, "Scheduling and Hardware Sharing in Pipelined data path," *IEEE Int'l conference on CAD*, Pages:24-27, 1989

[12] Samit Chaudhuri Stepen A. Blythe, Robert A. Walker "A solution methodology for exact design space exploration in a three-dimensional design space," *IEEE transaction on VLSI systems, Vol. 5.1*, 1997