제19회 한국정보처리학회 춘계학술발표대회 논문집 제10권 제1호 (2003.5)

FDS를 응용한 분할의 성능을 위한 힘 계산방법

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

Force calculation method for Performance of partitioning that apply FDS

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

요 약

통합설계의 분할 단계에서 스케줄링을 함께 고려하기 위해 FDS(Force-Directed Scheduling)를 응용하는 기존의 방법들은 분할될 노드를 선택하면서 그 노드가 스케줄 되어야 하는 제어구간을 결정한다. 그러나, 노드를 선택하기 위해 힘을 응용하는 기존의 분할 방법들은 알고리즘 실행시간은 현격하게 줄이는 반면 성능은 저하되는 결과를 보인다. FDS응용 기법은 노드의 모빌리티와 구현비용에 따라 분할할 대상 노드를 결정하고 분할 이후에 다른 노드들의 모빌리티를 변경하게 된다. 따라서, 분할 대상 노드들 중에서 어떤 노드를 최초로 선택하여 분할하는가에 따라서 전체적인 분할의 결과가 영향을 받게 된다.

본 논문에서는 기존의 힘 응용에 의한 분할 알고리즘의 실행시간 측면에서의 장점을 유지하면서 성능을 개선할 수 있도록 하기 위하여 효과적인 처음 분할 대상 노드 선택 방법을 제안한다. 제안 논문에 의한 분할 결과는 기존 방법의 알고리즘 실행시간을 초과하지 않으면서 개선된 성능을 보인다.

1. 서론

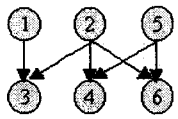
통합설계의 목적함수는 최소의 비용으로 최적의 성능을 갖는 시스템을 얻는 것인데, 이때 가장 큰 영향을 미치는 작업단계가 분할이다. 분할 문제의 복잡성(NP-hard)을 해결하기 위한 많은 휴리스틱 알고리즘들[2]이 개발되었는데, 대부분의 휴리스틱 알고리즘들은 분할과 스케줄링 작업을 독립적으로 진행하기 때문에 시스템 시간제약을 위배할 수 있고, 시간제약 위배는 시스템 재분할의 비용을 유발하게 된다. 재분할 비용과 알고리즘 실행시간 최소화를 위해 FDS를 응용하는 방법[3-5]은 분할과 스케줄링을 함께 고려하는데, Rousseau[3]등은 설계 대상 구조를 하나의 프로세서와 하나의 ASIC 구조로 하여 FDS를 응용할 때에 비용함수(실행시간 또는 설계비용)에 따라 계산되는 힘값을 갖고 분할단계에서 노드들중의 하나를 스케줄하며 분할한다. Choi[4]등은 시스템 실행시간과 구현 비용을 최소화하기 위하여, 처음에 모든 노드를 소프트웨어로 배정하고 시간제약을 만족할 때까지 한번에 하나의 노드를 선택하여 하드웨어로 보내는 것을 반복한다. 하드웨어로 분할될 노드의 선택을 위한 힘은 그 노드의 모빌리티 구간에 걸쳐서 소프트웨어에서 병행해서 실행될 수 있는 노드들과 하드웨어로 선택할 때 상승하는 비용을 반영한다. 논문[5]는 상대적 스케줄 긴박도를 정의하여 기존의 FDS 응용 방법들의 유도힘 계산에 따른 알고리즘 실행 시간을 개선하였다. 기존의 FDS응용 기법은 노드의 모빌리티와 구현비용에 따라 분할할 대상 노드를 결정하고 분할 이후에 다른 노드들의 모빌리티를 변경하게 된다. 따라서, 최초에 선택되는 노드에 의해 전체적인 분할 결과에 큰 영향을 미치게 된다. 본 논문에서는 기존의 힘 응용방법의 알고리즘 실행시간 효율을 유지하면서 성능을 개선할 수 있도록 하기 위하여 최초의 분할 노드를 선택하기 위한 효과적인 방법을 제안한다. 2장에서 제안알고리즘을 3장에서 실험 및 결과를 4장에서 결론을 각각 기술하였다.

2. 제안 알고리즘

하나의 일반적인 프로세서와 확장 가능한 하나의 하드웨어 모듈로 구성되는 대상 아키텍처에 대해 그림 1(a)와 같은 입력그래프와 노드 i의 하드웨어에서의 실행 시간, 소프트웨어에서의 실행시간, 하드웨어 구현비용, 그리고 소프트웨어 구현 비용을 의미하는 HT_i, ST_i, HC_i, SC_i등의 그림 1(b)와 같은 노드 정보에 대한 논문[3,4]의 최초의 분할 대상 노드는 힘값 계산에 의해 그림 1(c)와 같이 노드 6번이 선택되고 스케줄은 제어단계 6으로 결정된다. 입력 벤치마크의 대부분의 종속성 구조는 소스와 싱크가 다른 여러개

○ 본 연구는 2003년도 경인여자대학 교내연구지원 연구비에 의해 수행되었음

의 노드가 존재하는데, 소스와 싱크가 상이한 입력그래프 구조에 대한 기존의 힘값 계산에 의한 분할은 4번 노드의 마지막 제어단계와 같이 종속성이 존재하지 않음에도 불구하고 힘값이 작아지는 경우가 발생하여 분할을 성능을 저하시키게 된다.

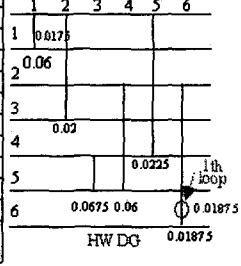


(a) 입력 예

	HT	HC	ST	SC
1	4	6	4	1
2	2	3	2	1
3	2	4	3	1
4	2	2	4	1
5	1	5	5	1
6	1	6	5	1

(b) 노드의 정보

HW	긴박도	처음단계/마지막단계	상대적 긴박도
1	0.06	Asap	0.0175
		Alap	0.0175
2	0.02	Asap	-0.0625
		Alap	-0.06125
3	0.0675	Asap	0.01125
		Alap	0.01125
4	0.06	Asap	-0.00125
		Alap	-0.02625
5	0.0225	Asap	-0.0575
		Alap	-0.05625
6	0.01875	Asap	-0.06575
		Alap	0.01875



(b) 처음 노드 선택

(그림 1) 분할 환경

소스와 싱크가 상이한 노드가 복합적으로 존재하는 입력그래프의 경우 분할 결과는 분할 대상 노드들 중에서 어떤 노드를 최초에 선택하는가에 따라 많은 영향을 받게 된다.

$$distr_{hard}(i) = 1/\ n_{hard_i}\ ,\quad distr_{soft}(i) = 1/\ n_{soft_i} \quad (1)$$

식(1)은 하드웨어와 소프트웨어 분포그래프에서 모빌리티 n_{hard_i}과 n_{soft_i}를 갖는 노드 i가 하나의 제어단계에 스케줄될 확률이다. 식(2)와 (3)은 스케줄 긴박도이며, $Cost-function$은 하드웨어 또는 소프트웨어로 분할할 때의 구현비용과 실행시간을 의미한다.

$$Urgency^{i}_{soft}(i) = (distr_{soft} \times \frac{1}{Cost-function(i,\ sw-implementation)} \times \beta \times HC_i) \quad (2)$$

$$Urgency^{i}_{hard}(i) = (distr_{hard} \times \frac{1}{Cost-function(i,\ hw-implementation)} \times \alpha \times ST_i) \quad (3)$$

일반적으로 하드웨어 분포그래프의 경우에 같은 제어구간에 스케줄되는 노드가 많을수록 설계비용은 증가하고, 소프트웨어 분포그래프의 경우에는 한 노드의 스케줄로 인해 그 제어구간에 모빌리티를 갖던 다른 노드들의 스케줄 긴박도가 증가하게 되므로 같은 제어구간에 모빌리티를 갖는 다른 노드들의 상대적인 힘을 반영해서 스케줄을 결정해야하므로 노드의 한 제어단계에서의 스케줄 긴박도에서 그 제어단계에 모빌리티를 갖는 다른 노드들의 스케줄 긴박도를 감한 값이 상대적 스케줄 긴박도이다. 상대적 스케줄 긴박도가 크다는 것은 스케줄 긴박도가 같을지라도 그 제어단계에서 경쟁하는 다른 노드들에 영향을 주는 정도가 적음을 의미한다. 노드의 상대적 스케줄 긴박도는 식 (4),(5)와 같은데, 노드 i가 제어구간 j에 스케줄되어야 하는 스케줄 긴박도로부터 제어구간 j에 스케줄될 수 있는 즉 모빌리티를 갖는 다른 노드들의 스케줄 긴박도의 합을 감하여 계산한다.

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

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

모든 제어단계에서 구한 상대적 스케줄 긴박도중에서 식 (6)과 같이 최대의 값을 갖는 노드가 우선적으로 분할 및 스케줄 대상이 된다.

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

상대적 스케줄 긴박도를 사용하는 제안 분할 알고리즘은 그림 2와 같다. 단계 2에서는 입력받은 시간제약 내에서 형성될 수 있는 분포그래프를 하드웨어와 소프트웨어 각각에 대해 생성한다. 단계 3에서 최초에 분할 되어야할 노드를 선택하여 분할한 이후, 단계 4에서 각 분포그래프내의 노드들에 대해 상대적 스케줄 긴박도를 계산하여 이 값이 최대인 노드를 선택하여 분할하고, 분할에 따른 분포그래프를 수정하며 모든 노드가 분할될 때까지 이 단계를 반복한다.

```
단계 1  DAG 및 노드특성, 시간제약 입력
단계 2  ASAP, ALAP 스케줄링으로 DG 생성
단계 3  select_first_node( )
단계 4  repeat until (모든 노드가 분할될 때까지)
 단계 4.1  for (분할 대상 노드)
           모빌리티의 처음 제어단계와 마지막 제어단계에서
           상대적 스케줄 긴박도 계산
           end for
 단계 4.2  최대의 상대적 스케줄 긴박도를 갖는 노드를 선택하여 분할
 단계 4.3  for (선택 노드를 제외한 분할 대상 노드)
           HW DG와 SW DG의 모빌리티 수정
           end for
 단계 4.4  선택 노드를 분할 대상 노드 집합에서 삭제
단계 5  end repeat
```

그림 2 제안 알고리즘

단계 3에서 최초에 분할되어야 할 노드를 선택하기 위한 힘 계산은 수식(2), (3)에서의 긴박도 계산에서 소스와 싱크가 상이한 복합적인 노드들의 종속관계를 반영하기 위해 입·출력 디그리를 반영하여 수식(7),(8)과 같이 결정한다. 식 (9)의 상대적 디그리(RD)의 크기는 각 노드의 종속성의 크기중에서 최대크기의 종속성에 대한 상대적인 크기를 의미하는 값이다. 처음제어 단계의 경우는 입력 디그리의 상대적인 크기를 반영하며 마지막 제어단계의 경우는 출력 디그리의 상대적인 크기를 반영하여 각 노드의 긴박도를 계산한다.

$$Urgency^{i}_{soft}(i) = (distr_{soft} \times \frac{1}{Cost-function(i,\ sw-implementation)}) \times \frac{}{\beta \times HC_i \times (RD \times 0.5)} \quad (7)$$

$$Urgency^{i}_{hard}(i) = (distr_{hard} \times \frac{1}{Cost-function(i,\ hw-implementation)}) \times \frac{}{\alpha \times ST_i \times (RD \times 0.5)} \quad (8)$$

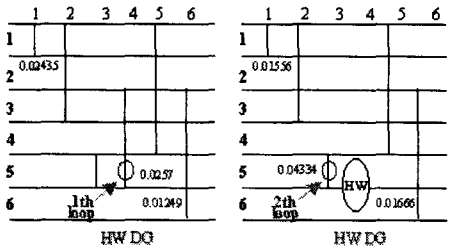
$$RD = \frac{\max(\deg ree)}{1+\deg ree} \quad (9)$$

제안 알고리즘의 시간복잡도는 노드의 개수가 n인 경우에, 처음 노드 선택을 위한 힘계산이 수행되므로 $O(n)$이며, $n-1$개의 각 노드의 처음 배정가능 제어단계와 마지막 배

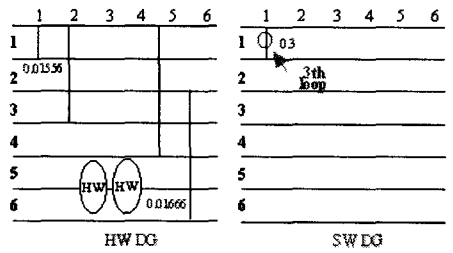
정가능 제어단계에서 상대적 스케줄 긴박도를 계산하므로 $O(2n)$이 되어 전체적인 시간복잡도는 $O(n^2)$이 된다.

3. 실험 및 결과

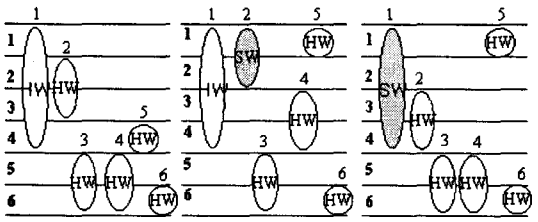
(그림 1)의 분할 환경과 시간제약 6에 대한 제안 알고리즘의 진행과정은 그림 3과 같다. 최초의 분할 노드는 하드웨어 비용은 작으면서 상대적으로 디그리의 크기가 작은 4번 노드의 마지막 제어단계로 결정된다. 이후 3번 노드의 하드웨어로의 분할 결정이후 소프트웨어 분포그래프가 생성된다. 분할 결과는 그림 3(c)와 같이 이전의 FDS 응용방법[3, 5]에 비해 효과적이며 ILP[9]에 의한 결과와 동일하다.



(a) 처음선택 노드 분할과 두 번째 노드 선택



(b) 세 번째 분할 노드 선택



(c) 분할 결과

그림 3 분할과정 및 결과

처음 노드 선택 방법 적용에 의한 분할 결과는 C++로 구현하여 다양한 벤치마크[5]에 대해 시스템 구현비용과 알고리즘 실행시간으로 ILP[5] 및 FDS응용[3,5]과 비교하였으며 그림 4와 같이 알고리즘 실행시간 효율은 유지하면서 분할 결과비용은 많이 감소되었다.

		Jam	GMDF -alpha	3×3 DET	Genetic -based	M-DIIRC	16-FIR Filter	AR-lattice Filter	DCT
시간제약 1	ILP	296	1058	563	522	331	665	1169	2009
	FDS응용[3]	296	1058	563	522	411	785	1169	2009
	논문[5]	296	1058	603	522	411	785	1169	2091
	제안논문	296	1058	563	522	371	785	1169	2091
시간제약 2	ILP	147	591	523	387	233	465	871	1729
	FDS응용[3]	147	591	523	476	439	745	1151	1729
	논문[5]	147	591	625	476	497	745	1151	1857
	제안논문	147	591	545	476	439	745	1151	1857

(a) 시스템 설계비용

		Jam	GMDF - alpha	3×3 DET	Genetic-based	M-DIIRC	16-FIR Filter	AR-lattice Filter	DCT
시간제약 1	ILP	0.3 S	0.4 S	12 S	10 S	12 S	2215 S	4429 S	1200 S
	FDS응용[3]	0.15 S	0.2 S	1.5 S	2.5 S	1.8 S	6.1 S	12 S	4.82 S
	논문[5]	0.05 S	0.01 S	0.1 S	1 S	0.5 S	1.3 S	2.5 S	1.56 S
	제안논문	0.05 S	0.01 S	0.1 S	1 S	0.5 S	1.3 S	2.5 S	1.6 S
시간제약 2	ILP	0.4 S	0.4 S	15 S	12 S	15 S	2240 S	4467 S	1590 S
	FDS응용[3]	0.3 S	0.05 S	2.3 S	3.5 S	3.29 S	6.54 S	16 S	4.9 S
	논문[5]	0.05 S	0.05 S	0.3 S	1 S	0.5 S	1.5 S	3.2 S	1.6 S
	제안논문	0.05 S	0.05 S	0.3 S	1 S	0.5 S	1.5 S	3.5 S	1.8 S

(b) 알고리즘 실행시간

그림 4 실험 결과

4. 결론

본 논문에서는 기존의 FDS 응용 분할 알고리즘의 힘 계산 방법에서 소스와 싱크가 복합적으로 존재하는 입력 벤치마크에서 나타나는 문제점을 해결하기 위해 최초의 분할 대상 노드를 선택하기 위한 방법을 제안하였으며 대부분의 벤치마크 실험결과 알고리즘 실행 시간 효율을 유지하면서 분할 결과를 크게 개선함을 보인다. 현재까지는 소프트웨어부의 병렬실행을 허용하지 않고 통신지연 시간이 노드의 실행시간을 넘지 않는 것으로 가정하였으며 하나의 하드웨어 모듈과 하나의 프로세서에 대한 실행을 가정하였다. 따라서, 복합적인 현실 시스템에 적용하기 위한 다양한 대상 아키텍처 시스템 고려와, 대상 아키텍처에 부합하는 다양한 벤치마크의 실험을 통한 검증을 향후 연구과제로 한다.

참 고 문 헌

[1] S. Edwards, L. Lavagno, E. A. Lee and A. Sangiovanni-Vincentelli, "Design of Embedded Systems: Formal Models, Validation, and Synthesis," *in*

Proceedings of IEEE. vol. 85, no. 3, pp. 366-390, Mar. 1997.

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

[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] 오주영, 이면재, 이준용, 박도순, "하드웨어/소프트웨어 통합설계를 위한 FDS 분할 알고리즘의 성능개선", *한국정보처리학회 논문지,* 2003.