FDS 응용 분할을 위한 개선된 DG 생성방법

오주영
경인여자대학
email: *odid080@kic.ac.kr

A refined DG generation method for FDS-applied partitioning

Ju-Young Oh
Kyung-in woman's college

요 약

하드웨어/소프트웨어 통합설계에서 기존의 FDS(Force-Directed Scheduling)를 응용하는 모든 방법들은 분포그래프를 기반으로 분할을 수행하는데, 이들의 문제점은 입력테이블의 특정한 설계방법에서 구현이 불가능한 노드가 존재할 때에는 분할의 해가 존재함에도 불구하고 분할의 해를 찾지 못하는 경우나 다양한 해가 존재함에도 불구하고 탐색공간을 충분히 고려하지 못하는 경우가 발생한다. 본 연구에서는 입력테이블의 여러 가지 설계방법에서 구현이 불가능한 노드가 복합적으로 존재하는 경우에도 분포그래프를 생성하고 생성된 분포그래프의 노드별 힘을 계산하여 분할이 가능하게 하였다. 제안 방법은 비용테이블에서 특정한 구현방법으로의 매핑이 불가능한 경우에는 최소 실행시간을 갖는 구현방법을 임의로 선택해서 분포그래프를 생성하여 분할할 수 있도록 하였다. 제안방법의 실험 결과는 기존의 FDS 응용 방법들보다 개선된 알고리즘 실행시간과 더불어 여러개의 노드에 대해 불가능한 구현 방법이 복합적으로 존재할 때에도 분할이 가능함을 보인다.

1. 서론

분할과 스케줄링을 함께 고려하기 위해 FDS를 응용하는 방법[1][2]은, 설계 대상 구조를 하나의 프로세서와 하나의 ASIC 구조로 하여 비용함수(실행시간 또는 설계비용)에 따라 계산되는 힘값을 갖고 한번에 하나의 노드를 스케줄하며 분할한다. 이 방법들은 노드를 분할할 때 증가되는 설계비용을 자신의 힘으로 정의하고 분할 이후에 그 노드의 스케줄로 인해서 삭제되는 다른 노드들의 모빌리티를 유도힘으로 정의하여 두 개의 힘의 합을 하드웨어 부분과 소프트웨어 부분에 대해 구하여 이 값이 가장 작은 노드를 선택하여 분할하였는데, 논문[3-6]에서는 기존의 FDS 응용방법의 유도힘 계산으로 인해 가중되던 알고리즘 복잡도를 개선하기 위해 상대적 스케줄 긴박도를 정의하여 여러개의 프로세서와 다양한 형태의 구현방법을 갖는 하드웨어 구조에 대해서도 분할할 수 있도록 확장하였다. 그러나 FDS를 응용한 기존의 모든 분할 방법들[1-6]은 특정한 구현 방법에서 구현이 불가능한 노드가 존재할 때에는 분포그래프가 그려지지 않음으로 인해 분할이 불가능하게 되거나 탐색공간을 충분히 고려하지 못하는 문제점이 있다.

본 연구는 복합적인 타겟아키텍처를 지원하기 위해 상대적 스케줄 긴박도를 확장하여 분할을 수행하는데, 분할에 따른 통신지연과 스케줄 가능성 및 구현비용을 함께 고려한다. 그리고 소프트웨어 영역에 분할되는 노드의 실행을 위한 프로세서와 하드웨어 영역에 분할되는 노드의 구현상의 다형성은 각 구현의 경우를 다중의 프로세서로 취급하여 문제를 단순화한다. 알고리즘의 성능평가는 상위수준 합성에서의 스케줄링을 위한 벤치마크들인 M-DHRC, 3×3 DET, 16-FIR Filter, AR-Lattice Filter와 통합설계에서의 분할을 위한 벤치마크들인 DCT, jam, GMDF-Alpha, 그리고 Genetic-based[12]를 사용하며 기존의 FDS 응용 분할 알고리즘의 결과와 비교한다. 2장에서 제안 방법을 3장에서 실험 및 결과를 4장에서 결론을 각각 기술하였다.

2. 제안 방법

하드웨어/소프트웨어 분할을 위해 제안한 스케줄긴박도[3]는 각 노드의 분포그래프별로 계산되는 확률, 실행 시간, 구현 비용, 그리고 종속성이 있는 노드들이 서로 다른 영역으로 분할될 때 발생하는 통신 지연 시간을 반영하여 계산한다. 노드간의 통신 지연시간은 종속 노드들이 분할되는 프로세서가 상이할 경우에만 반영하고, 같은 프로세서에 분할될 경우에는 통신 지연시간이 없다고 가정한다. 힘 값은 한 노드가 특정 분할 영역의 특정 제어단계에 스케줄 되어야 하는 필요성의 정도를 의미한다. 즉, 실행시간이나 구현비용은 작으면서 스케줄할 수 있는 제어구간이 짧고 상대적으로 스케줄 경쟁 노드가 적어서 다른 노드의 스케줄을 방해하는

○ 본 연구는 2005년도 경인여자대학 교내연구지원 연구비에 의해 수행되었음

힘이 가장 작은 노드가 우선 스케줄될 수 있도록 한다. 한 노드의 분포 확률 값은 특정 분할 영역에서 노드가 갖는 배정 가능 범위로서 특정 분할 영역에서 갖는 실행 시간에 따라 다르게 결정되며 소프트웨어 분할영역 j에서 노드 i가 갖는 분포 확률은 식(1)로 계산하고 하드웨어 분할영역 j에서 노드 i가 갖는 분포 확률은 식(2)로 계산한다. 분할을 위한 힘인 스케줄 긴박도 계산은 시간제약을 만족한다면 구현 비용이 싸고 실행 시간이 빠르며 상이한 분할간에 발생할 수 있는 통신 지연 시간을 최소화 할 수 있도록 계산한다. 즉, 구현 비용과 실행 시간이 작을수록 큰 힘 값을 가지게 하고 종속성이 있는 노드들인 경우에는 동일한 영역으로 분할을 유도하기 위해서 통신 지연 시간이 없는 경우에 큰 힘 값을 가지게 하며 식 (3)과 (4)로 계산한다.

$$distr_{soft\ j}(i) = 1/(n_{soft_j}(i)) \quad (1)$$

$$distr_{hard\ j}(i) = 1/(n_{hard_j}(i)) \quad (2)$$

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

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

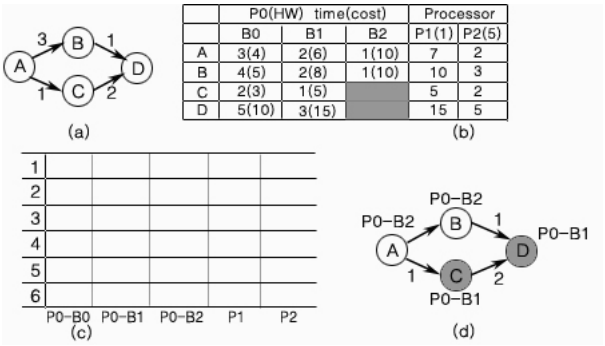
분할 영역별 상대적 스케줄 긴박도는 노드 i를 제어단계 j에 분할할 때에 노드 i의 스케줄 긴박도로부터 제어단계 j에 모빌리티를 갖는 다른 노드들의 스케줄 긴박도의 합으로 감하여 계산하며 분포그래프의 처음 제어단계와 마지막 제어단계에 대해서 계산한다. 분할 대상 노드는 하드웨어 분할 영역의 노드별 상대적 스케줄 긴박도를 식(5)와 같이 계산하고 소프트웨어 분할 영역의 노드별 상대적 스케줄 긴박도를 식(6)과 같이 계산하여 이들중에서 최대값을 갖는 노드를 선택하는데 식(7)과 같다. 따라서 시간제약을 만족한다면 구현 비용이 싸고 실행 시간이 빠르며 상이한 분할간에 발생할 수 있는 통신 지연 시간이 최소화될 분할을 선택하게 된다.

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

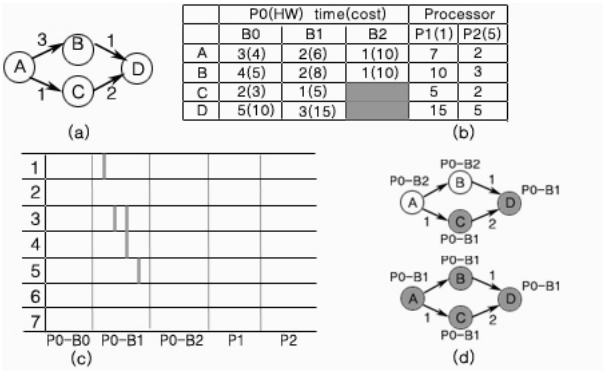
$$Rel-Urgency^{p}_{soft_j}(i) = Urgency^{p}_{soft_j}(i) - \sum_{k=1(k\neq i)}^{n} Urgency^{p}_{soft_j}(k) \quad (6)$$

$$Rel-Urgency^{p}_{j}(i) = \max\{Rel-Urgency^{p}_{soft_j}(i), Rel-Urgency^{p}_{hard_j}(i)\} \quad (7)$$

이러한 계산방법에 의한 분할방법[6]과 기존의 모든 FDS 응용 방법들은 분포그래프를 기반으로 분할을 수행한다. 이러한 분포그래프 기반의 분할 알고리즘들의 문제점은 입력 테이블의 특정 설계방법에서 구현이 불가능한 노드가 존재할 때에는 분할의 해가 존재함에도 불구하고 해를 찾지 못하는 경우나 다양한 해가 존재함에도 불구하고 탐색공간을 충분히 고려하지 못하는 경우가 발생한다. 예를 들어, [그림 1]과 [그림 2]의 입력환경에서 노드 C와 노드 D가 하드웨어 P0의 설계방법 B2로의 구현이 불가능하다고 가정할 때, 시간제약이 6인 [그림 1]은 [그림 1(d)]와 같은 분할의 해가 존재함에도 불구하고 [그림 1(c)]와 같이 시간제약내에서 모든 구현 방법에 대한 분포그래프가 그려지지 않아서 분할이 불가능하게 된다. 또한 시간제약이 7인 [그림 2]는 여러 형태의 구현 방법이 존재함에도 불구하고 하드웨어 P0의 설계방법 B1의 분포그래프만 그려지고 분할의 진행과정에서도 [그림 2(c)] 이상의 분포그래프가 생성되지 않는다. 이 때문에 분할을 위한 충분한 탐색이 이루어지지 못하게 되는데 이러한 현상은 각각의 구현 방법에서 구현이 불가능한 노드가 많을수록 크게 된다.



[그림 1] 분포그래프가 생성되지 않는 경우의 예

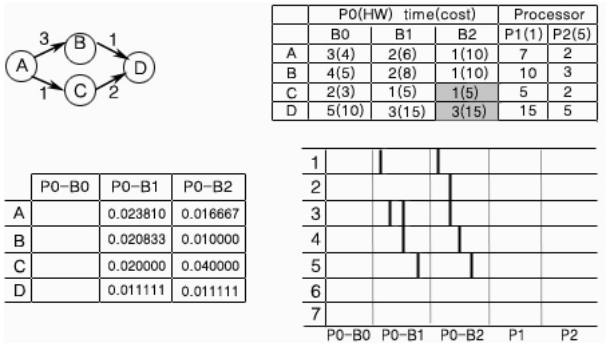


[그림 2] 구현 방법을 충분히 탐색하지 못하는 경우의 예

따라서, FDS를 응용하는 기존의 분할 알고리즘에서 특정한 구현방법으로 구현이 불가능한 노드가 존재할 경우에 발생하는 문제점을 해결하기 위해 구현이 불가능한 노드에 대한 실행시간을 설정하여 분포그래프를 생성하는 방법이 요구되는데, 제안 방법에서는 구현이 불가능한 노드의 실행시간은 시간/비용 테이블에서 그 노드의 다른 구현방법들 중에서 실행시간이 최소가 되는 항목의 구현방법을 선택하며 식(8)과 같다. 만약에 노드 i가 구현 방법 j에서 구현이 불가능한 경우에는 다른 구현 방법들(p)중에서 실행시간이 최소인 항목의 시간을 실행시간으로 설정한다. 최소의 실행시간을 선택하여도 분포그래프가 생성되지 않을 때에는 분할의 해가 존재하지 않는 것이며 빠른 실행시간을 갖는 구현을 선택함으로써 상승되는 비용은 스케줄 긴박도 계산에서 비용 상승분에 의해 고려된다.

$$impossible_i^j = \min_{k \neq j, k \in p(possible implementation)} | Time_i^k | \quad (8)$$

구현이 불가능한 노드의 분포그래프는 최소 실행시간을 갖는 다른 구현 방법을 선택하여 생성하는데, 이때 다른 구현방법에 할당함으로써 발생하는 통신상의 지연시간을 함께 고려하여 생성한다. [그림 2]의 시간/비용 테이블에서 하드웨어 P0의 설계방법 B2로의 구현이 불가능한 노드 C와 노드 D는 하드웨어 P0의 설계방법 B1의 실행시간을 선택하여 분포그래프를 생성하며 최초의 분포그래프는 [그림 3]과 같다.



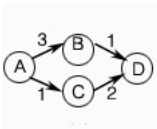
	P0(HW) time(cost)			Processor	
	B0	B1	B2	P1(1)	P2(5)
A	3(4)	2(6)	1(10)	7	2
B	4(5)	2(8)	1(10)	10	3
C	2(3)	1(5)	1(5)	5	2
D	5(10)	3(15)	3(15)	15	5

	P0-B0	P0-B1	P0-B2
A		0.023810	0.016667
B		0.020833	0.010000
C		0.020000	0.040000
D		0.011111	0.011111

[그림 3] 수정된 시간/비용 테이블에 의한 분포그래프

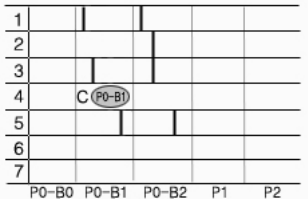
3. 실험 및 결과

알고리즘 반복에서 분할 노드 선택을 위하 알고리즘은 논문[12]와 동일하며 최초의 시간/비용 테이블에서 구현이 불가능한 노드가 존재할 때에 식 (8)에 의해 선택된 구현방법의 실행시간과 통신지연시간을 반영하여 분포그래프를 생성한 후에 알고리즘을 진행한다. 알고리즘의 처음 반복에서는 분할 대상 노드들 중에서 구현비용이 제일 작고 같은 제어단계에서 스케줄을 경쟁하는 노드가 없어서 상대적 스케줄 긴박도가 가장 큰 노드 C가 하드웨어 P0의 구현방법 B1으로 제어단계 4에 분할되며 [그림 4(a)]와 같다. 알고리즘의 두 번째 반복에서는 구현비용이 작고 모빌리티가 짧으며 같은 제어단계에서 경쟁하는 노드의 힘이 상대적으로 적은 노드 A가 하드웨어 P0의 구현방법 B1으로 제어단계 1에 분할된다. 분할결과는 [그림 4(b)]와 같으며, 충분한 탐색을 위해 하드웨어 P0의 구현방법 B2의 분포그래프를 생성하였지만 구현비용이 적은 방법으로 분할되었으며 최적의 분할결과와 같다.

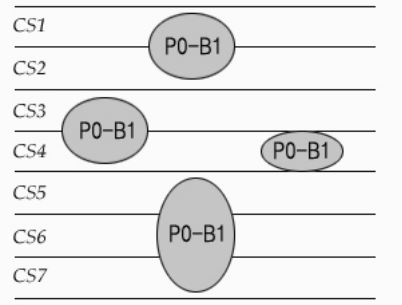


	P0(HW) time(cost)			Processor	
	B0	B1	B2	P1(1)	P2(5)
A	3(4)	2(6)	1(10)	7	2
B	4(5)	2(8)	1(10)	10	3
C	2(3)	1(5)	1(5)	5	2
D	5(10)	3(15)	3(15)	15	5

	P0-B0	P0-B1	P0-B2
A		0.023810	0.016667
B		0.020833	0.010000
C		0.020000	0.040000
D		0.011111	0.011111



(a) 노드 C를 분할한 이후의 분포그래프



(b) 분할 결과

[그림 4] 수정된 분포그래프에 의한 분할과정

3.1 구현 불가능이 복합적으로 존재할 때의 분할

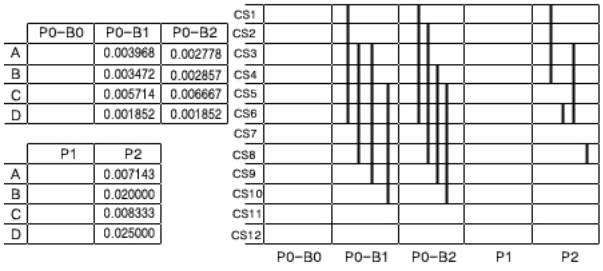
[그림 5(a)]는 각 노드에 대한 구현이 불가능한 경우가 복합적으로 존재하는 경우의 시간/비용 테이블이다. 이와 같은 입력환경에서 기존의 FDS 응용방법으로 분할하면 모든 노드가 하드웨어 P0의 구현방법 B1으로 분할되며 구현비용이 34가 된다. 본 연구에서 제안하는 실행시간 선택방법은 [그림 5(b)]와 같이 시간/비용 테이블을 수정하고 [그림 5(c)]와 같은 최초의 분포그래프를 생성한다. 처음에는 구현비용이 작고 모빌리티가 짧으며 같은 제어단계에서 스케줄을 경쟁하는 노드가 없는 노드 D가 프로세서 P2의 제어단계 8에 분할되며 [그림 5(c)]와 같다. 노드 D의 분할 이후에는 모빌리티가 존재하는 분포그래프들 중에서 실행시간이 짧고 같은 제어단계에서 스케줄을 경쟁하는 노드가 없는 노드 B가 하드웨어 P0의 구현방법 B2의 제어단계 6에 분할된다. 최종 분할 결과는 [그림 5(g)]와 같으며 시간/비용 테이블에서 구현이 불가능한 경우가 복합적으로 존재하는 경우에도 제안방법에 의한 실행시간 선택방법으로 분포그래프를 생성하고, 생성된 분포그래프의 노드별 힘을 계산하여 분할이 가능함을 보였다. 제안한 실행시간 선택방법에 의한 설계비용은 24인데 충분한 탐색공간을 고려함으로써 분포그래프 수정 이전의 결과에 비해 설계비용이 크게 줄어든다.

	P0(HW) time(cost)			Processor	
	B0	B1	B2	P1(1)	P2(5)
A		2(6)	1(10)	7	2
B	4(5)	2(8)	1(10)	10	
C	2(3)	1(5)		5	2
D	5(10)	3(15)			5

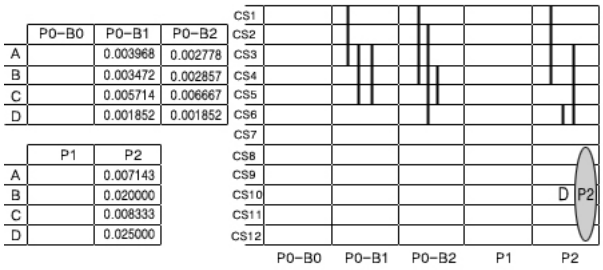
(a) 구현 불가능이 존재하는 시간/비용 테이블

	P0(HW) time(cost)			Processor	
	B0	B1	B2	P1(1)	P2(5)
A	1(10)	2(6)	1(10)	7	2
B	4(5)	2(8)	1(10)	10	1(10)
C	2(3)	1(5)	1(5)	5	2
D	5(10)	3(15)	3(15)	3(15)	5

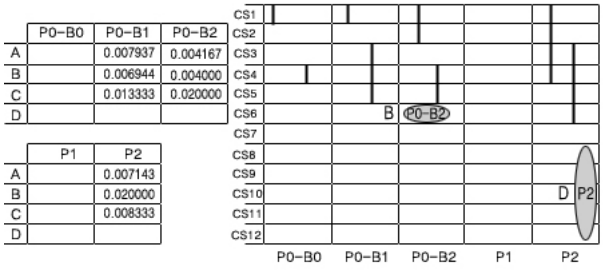
(b) 수정된 시간/비용 테이블



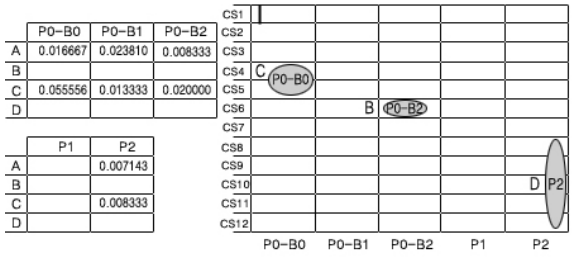
(c) 수정된 시간/비용 테이블에 의한 최초 분포그래프



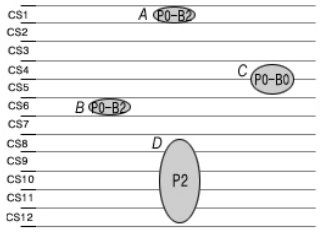
(d) 노드 D의 분할 이후의 분포그래프



(e) 노드 B의 분할이후의 분포그래프



(f) 노드 C의 분할이후의 분포그래프



(g)분할 결과

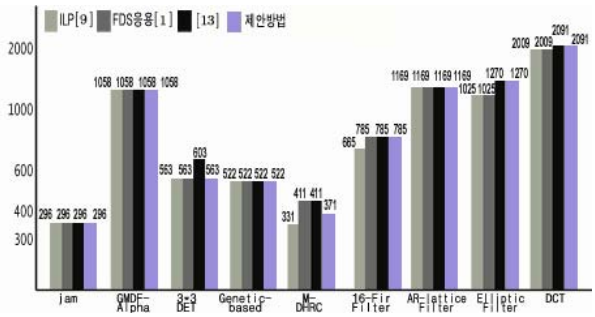
[그림 5] 구현 불가능이 복합적으로 존재할 때의 분할과정

4. 결론

입력 노드의 개수가 n이고, 분할 영역의 개수가 p개, 각 분할영역에서 구현 가능한 방법의 개수를 i, 고려해야할 제어단계의 수를 c라고 가정할 때, 제안 알고리즘의 시간복잡도는 [그림 6]과 같으며, 분할을 결정하기 알고리즘 실행결과는 [그림 7]과 같이 타겟아키텍처의 다형성에 구애받지 않으며 성능 면에서는 [그림 8]과 같이 많은 향상이 있음을 확인하였다.

	프로세서 하나와 확장 가능한 하드웨어 모듈	복합적인 구현방법을 지원
[1]	$O(c^2 n^3)$	✖
[12]	✖	$O((npi)^2 S)$
제안방법	$O(n^2)$	$\theta(pi\, n^2)$

[그림 6] 시간 복잡도 비교



[그림 7] 분할결과 비교

[그림 8] 실행 시간 비교

4 결론

본 연구에서는 하드웨어/소프트웨어 통합설계의 분할을 위해 FDS를 응용하는 방법들의 문제점을 해결하기 위해 시간/비용 테이블에서 구현이 불가능한 경우가 복합적으로 존재하는 경우에도 분포그래프를 생성하고, 생성된 분포그래프의 노드별 힘을 계산하여 분할이 가능하게 하였으며 시간제약내에서 타겟아키텍처에 맞는 최소비용의 시스템을 합성하기 위한 분할의 해를 찾는 것을 목표로 하였다. 앞으로는 실질적인 내장형 시스템 합성을 위해서 시간제약뿐만 아니라 면적제약도 함께 지원할 수 있도록 확장할 것이다. 또한, 실시간 내장형 시스템의 특징을 고려하여 프로세서에서 실행되는 각 작업의 주기와 효과적인 작업 스케줄링 및 구조적 파이프라이닝을 함께 고려해야 하며 다양한 벤치마크를 사용하여 결과를 비교 분석하고, 최적의 해를 찾기 위한 분할기법을 연구할 것이다.

참고 문헌

[1] F. Rousseau, J. Benzakki, J-M. Berge and M. Israel, "Hardware/Software Partitioning for Telecommunications systems," RSP, pp. 483-489, August 1995.

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

[3] 오주영, 박도순, "노드의 상대적 스케줄 긴박도 분석에 의한 하드웨어-소프트웨어 분할," 한국정보처리학회 학술발표논문집 제7권, 제2호, 2000.

[4] 박효선, 오주영, 박도순, "FDS 응용에 의한 하드웨어 소프트웨어 분할 알고리즘의 시간복잡도 개선," 한국정보과학회 학술발표논문집, 제27권, 제2호, 2000.

[5] 오주영, 이면재, 이준용, 박도순, "하드웨어/소프트웨어 통합설계를 위한 FDS 분할 알고리즘의 성능개선," 한국정보처리학회 논문지, 제9-A권, 제4호, pp. 491-496, 2002.

[6] 오주영, 박도순, "내장형 시스템 설계를 위한 FDS 분할 알고리즘," 한국정보과학회 학술발표논문집, 제29권, 제1호, pp. 34-36, 2002.

[7] Chuck Monahan, Forrest Brewer, "Scheduling and binding bounds for RT-level symbolic execution," International Conference on Computer Aided Design, pp. 230-235, Nov. 1997.

[8] K. S. Hwang, Albert E. Casavant, Ching-Teng Chang, "Scheduling and hardware sharing in pipelined data paths," In Proceedings of the IEEE International Conference on CAD, pp.24-27, Nov. 1989.

[9] 오주영, 한갑수, 박도순, "하드웨어 소프트웨어 분할을 위한 ILP 구현," 한국정보과학회 학술발표논문집, 제27권, 제2호, pp. 21-23, 2000.

[10] Hidalgo, J. L. Lanchares, J., "Functional partitioning of hardware-software codesign using genetic algorithms," Proceedings of the EUROMICRO conference, pp. 631-638, 1997.

[11] J. A. Maestro, "New methodologies for Hardware-Software Codesign Partitioning to Avoid High Communication Overhead," Departmento de informatica y Automatica Universidad complutense de Madrid, 1997

[12] Hyunok Oh, Soonhoi Ha, "A Hardware-Software Cosynthesis Technique Based on Heterogeneous Multiprocessor Scheduling," 7th International Workshop on Hardware/Software CO-Design, pp. 183-187, May 1999.

[13] 오주영, 박도순, "FDS를 응용한 분할의 성능을 위한 힘 계산방법," 한국정보처리학회 학술발표논문집, 제10권, 제1호, pp. 467-470, 2003.