백트랙킹 방법을 이용한 하드웨어/소프트웨어 분할

이면재[0] 박도순
홍익대학교 컴퓨터 공학과
{mjlee[0], dspark}@cs.hongik.ac.kr

Hardware/Software Partitioning Using Backtracking Method

Myoun-Jae Lee[0] Do-Soon Park
Dept. of Computer Engineering, Hongik University

요 약

본 논문에서는 백트랙킹 알고리즘을 이용한 하드웨어/소프트웨어 분할 방법을 제안한다. 최적의 해를 찾을 때에 효율적인 가지치기 함수를 정의하여 불필요한 탐색 단계를 제거하므로써 계산 시간이 단축될 수 있도록 하였다. 또한 제약조건에 따라 트리의 검색 순서에 변화를 주어 효율적인 검색이 되도록 하였다. 제안된 알고리즘의 성능평가를 위해 시뮬레이티드 어닐링 방법의 결과와 비교 분석하였다.

1. 서 론

내장형 시스템은 크게 주문형 반도체(ASIC) 또는 FPGA등으로 구성되는 하드웨어와 DSP 또는 CPU등으로 구성되는 소프트웨어 부분으로 이루어진다. 이러한 내장형 시스템 설계는 주어진 제약 조건을 만족하면서 최소의 비용으로 시스템을 구현해야 한다. 내장형 시스템의 설계는 통합설계로 행해지며, 이러한 통합 설계 과정중에서 시스템의 성능을 좌우하는 최대 요소는 시스템의 기능을 서술하는 각각의 행위 기술들을 하드웨어 부분과 소프트웨어 부분에 할당될 영역을 설정하는 것이며, 이러한 일련의 작업 과정이 분할이다.

2.관련 연구

내장형 시스템의 하드웨어/소프트웨어 분할을 위해서 다양한 알고리즘과 목적함수들이 개발되어왔다. Gupta[1]에서는 모든 연산을 처음에 하드웨어로 설정하고, 주어진 시간 제약 조건 범위 안에서 한번에 하나의 연산을 greedy 방법으로 선택하여 소프트웨어로 보내는 방법을 사용한다. 이 방법은 탐색된 해가 국부 최적치가 될 수 있는 단점이 있다. COSYMA[2,3]는 시뮬레이티드 어닐링을 사용하여 분할을 수행하며, 초기에는 모든 연산들을 소프트웨어로 설정하고 제약 조건이 만족되는 범위 안에서 연산을 하드웨어로 보내는 방법을 사용한다. 이 방법은 최적의 해를 찾을 가능성은 높아지나 계산 시간이 많이 소요되며, 최적의 해를 찾을 수 있도록 초기 온도 설정, 종료 조건등의 값을 설정하는데 많은 시간이 소요된다. BCS[4]는 이진 검색 방법과 시뮬레이티드 어닐링 방식을 동시에 사용하는 알고리즘으로 최적의 해를 찾을 가능성은 높으나 시뮬레이티드 어닐링과 같이 계산시간이 많이 소요된다. 유전자 알고리즘[5]은 최적의 해를 찾을 가능성은 높아지나 최적의 해를 찾기에 필요한 초기 설정값, 즉 교배율(cross over rate), 돌연 변이율(mutation rate)등을 설정하는데 어려움이 있다. 자원 제약 조건하에서 실행시간을 최소화하는 확장된 Kernighan/Lin 휴리스틱[6,7]은 최적의 해를 빠른 시간안에 찾기위한 휴리스틱 함수를 이용하지만 최적의 해를 찾지 못할 가능성이 있다.

본 논문에서는 최적의 해를 빠른 계산 시간안에 찾을 수 있도록 백트랙킹 알고리즘을 하드웨어/소프트웨어 분할에 적용함을 보인다. 본 논문의 내용은 3절에서 제안된 분할 알고리즘을 기술하고, 4절에서 알고리즘 성능평가를 기술하였다.

3. 제안 분할 알고리즘

일반적으로 백트랙킹 방법은 깊이 우선 탐색을 이용하여 최적의 해를 찾는 경우에 많이 사용된다. 이때 최적의 해를 찾기 위해 소요되는 시간은 트리 구성 방법과 불필요한 탐색과정을 줄이는데 사용되는 가지치기 함수(Pruning Function)에 따라 좌우된다.

본 논문에서는 DAG(Directed Acyclic Graph), 각 노드의 하드웨어 실행시간, 소프트웨어 실행시간, 그리고 시간 제약이 주어질 때 시간 제약 조건을 만족하는 범위내에서 최소의 비용으로 구성되는 하드웨어/소프트웨어 분할 결과를 얻기 위해 백트랙킹 방법을 사용함을 보이며, 이를 위해 3개의 가지치기 함수를 정의하였다. 또한 트리를 구성할 때에 소프트웨어에서의 실행시간과 하드웨어에서의 실행시간의 차가 큰 순서대로 노드의 단계를 결정하도록 하여 무작위로 트리의 단계를 정하는 경우보다 평균 탐색시간을 줄이도록 하였다.

3.1. 비용함수

본 논문에서 제안한 비용함수는 식1,2와 같은데, 전체 실행시간 t, 즉 성능은 하드웨어에 할당되는 노드들의 실행시간의 합과 소프트웨어에 할당되는 노드들의 실행시간의 합이며, 전체 비용 c는 하드웨어에 할당되는 노드들의 비용의 합과 이와 관련된 노드들의 통신 비용의 합이며, 소프트웨어의 구현 비용은 0으로 가정한다.

$$t = \sum_{i \in 0..n\_HW} HT(i) + \sum_{i \in 0..n\_SW} ST(i) \quad \text{.....(1)}$$

$$c = \sum_{i \in 0..n\_HW} [HC(i) + COM(i,j)] \quad \text{.....(2)}$$

3.2 트리 구성

노드들의 소프트웨어에서의 실행시간과 하드웨어에서의 실행시간의 차를 기준으로 노드들의 단계를 결정하며 이 차가 큰 노드일수록 상위 단계에 놓인다. 즉 루트 노드는 모든 노드들 중에서 소프트웨어에서의 실행시간과 하드웨어에서의 실행시간의 차가 가장 큰 노드가 된다. 트리에서 간선은 해당 단계에 있는 노드를 하드웨어에 또는 소프트웨어에 분할할 것인지를 나타낸다.

3.3 가지치기 함수

백트랙킹 방법에서 가지치기 함수는 불필요한 서브 트리의 생성을 막기 위해 사용한다[8]. 본 논문에서 제안한 가지치기 함수는 식(3),(4),(5)와 같으며, 함수에 표현된 기호는 아래의 의미이다.

1) $T_{constraint}$: 입력으로 주어진 시간제약 조건
2) n_{part}:현재까지 분할이 결정된 노드들의 집합
3) t_{part} : 현재까지 분할이 결정된 노드의 실행시간의 합
4) C_{part}: 분할이 결정된 노드들의 구현 비용 합
5) $n_{no_{part}}$:현재까지 분할이 결정되지 않는 노드들의 집합
6) C_{best}: 시간 제약 조건을 만족한 분할 상태중에서 가장 낮은 구현 비용

$$t_{part} + \sum_{i=0} \min[HT(i), ST(i)] < T_{constraint} < t_{part} + \sum_{i=0} \max[HT(i), ST(i)]$$

$$\text{for } i \in n_{no_{part}} \quad \text{.....(3)}$$

$$C_{best} \geq C_{part} + \sum_{i=0} [\min(HC(i) + COM(i,j), SC(i) + COM(i,j)]$$

$$\text{for } i \in n_{no_{part}}, j \in Predecessor(i) \text{...(4)}$$

$$HT(i) \leq T_{constraint},\ ST(i) \leq T_{constraint} \quad \text{.....(5)}$$

식(3)은 성능과 관련된 함수로 현재 분할이 결정된 노드들의 실행시간의 합과 해당 노드의 서브 트리안에 있는 노드들의 실행시간의 합이 시간 제약 조건을 만족할 가능성이 있다면 해당 노드에 대한 서브 트리를 생성하고 그렇지 않다면 서브 트리를 생성하지 않는다. 식(4)는 비용과 관련된 함수로 현재까지 수행한 여러 분할 상태중에서 가장 낮은 비용을 갖는 분할 상태보다 더 낮은 비용의 분할 상태가 나올 가능성이 있다면 해당 노드에 대한 서브 트리를 생성하고 그렇지 않다면 서브 트리의 생성을 중지한다. 식(5)는 해당 노드의 하드웨어 실행시간 또는 소프트웨어 실행시간이 시간 제약 조건을 위배하지 않아야 한다는 것을 의미한다. 해당 분할 영역의 실행시간이 시간 제약 조건을 위배하는 경우에는 서브 트리의 생성을 중지한다. 위의 3개의 가지치기 함수를 모두 만족하는 경우에만 해당 노드에 대한 서브 트리를 생성하고 그렇지 않다면 서브 트리를 생성하지 않는다.

3.4 제안 알고리즘

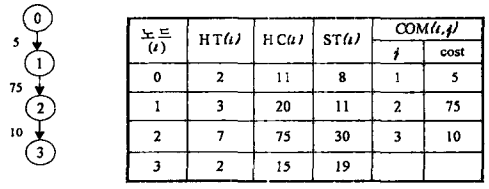
분할을 위한 대상 아키텍처는 소프트웨어 실행을 위한 하나의 프로세서와 확장 가능한 ASIC모듈로 가정하였으며, 제안 알고리즘은 [표 1]과 같다. [표 1]의 promising함수는 식(3),(4),(5)를 검사하여 모든 조건을 만족하면 참을 반환하고 그렇지 않으면 거짓을 반환한다. checknode함수는 promising함수의 반환값이 참이면 서브 트리를 생성하고 그렇지 않으면 서브 트리의 생성을 중지하여 불필요한 탐색과정을 줄이는 함수이다.

```
Step 1: determine node's level
Step 2: C_best=∞;
        determine node's generation order
Step3: checknode(node v) {
  Step 3.1:If (node is leaf and   C_cost is  better than C_best)
             save C_cost,Partiion State;
  Step 3.2: If (promising(v))
             For (each child u of v)
               checknode(u);
```

[표 1] 제안 알고리즘

3.5 제안 알고리즘에 의한 분할

[그림 1]의 입력에서 제약 시간이 14라고 가정한 경우에 제안된 알고리즘에 의한 분할 과정은 그림[2]와 같다.

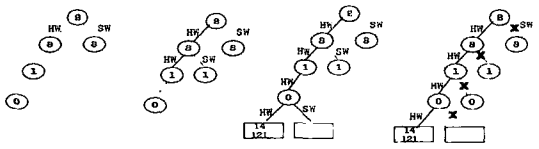


[DAG]

노드 (i)	HT(i)	HC(i)	ST(i)	COM(i,j)	
				j	cost
0	2	11	8	1	5
1	3	20	11	2	75
2	7	75	30	3	10
3	2	15	19		

[노드 정보 테이블]

[그림 1] 분할을 위한 DAG 입력과 노드 정보 테이블



[그림 2] 제안 알고리즘의 분할 과정

[표 1]의 Step1에서 노드들의 단계는 소프트웨어에서의 실행시간과 하드웨어에서의 실행시간의 차가 큰 순서대로 결정된다. [그림1]의 노드 정보 테이블에서 소프트웨어에서의 실행시

간과 하드웨어에서의 실행시간의 차가 가장 큰 노드, 즉 노드 2가 루트 노드가 되고 노드 3,1,0순서로 노드들의 단계가 결정된다. Step2는 루트 노드에 대한 서브 트리를 생성할 때 어떤 분할영역을 먼저 고려할 것인지 결정하는 단계로서 주어진 시간 제약조건에서 모든 노드가 하드웨어로 분할될 경우의 실행시간의 차를 구한 값과 소프트웨어로 분할될 경우의 실행시간의 차를 구한 값을 비교하여 이 차가 작은 쪽에 대한 분할을 먼저 고려하도록 하여 최적의 해를 찾을 때 소요되는 계산시간이 단축될 수 있도록 하였다. 이 예제의 경우 하드웨어로 분할될 경우를 먼저 고려하여 분할을 수행한다. Step3.2는 서브 트리 생성 여부를 결정하는 단계로서 promising 함수를 호출하여 노드 2가 하드웨어로 분할 될 경우 식(3),(4),(5)를 검사하여 모두 참이 되므로 서브 트리를 생성한다. 노드 3,1이 하드웨어로 분할될 경우에 대해서도 promising 함수를 수행하여 모두 참이 되므로 서브 트리를 생성한다. Step3.1은 리프 노드, 즉 입력으로 들어온 모든 노드의 분할이 결정된 상태가 되면 현재 분할 상태의 비용과 현재까지 수행한 여러 분할 상태 중에서 가장 낮은 분할 비용과 비교하여 가장 낮은 비용을 갖는 비용과 분할 상태를 저장하는 단계로, 모든 노드가 하드웨어로 분할된 상태와 비용, 121을 저장한다. 노드 0,1,3,2가 소프트웨어로 분할될 경우에는 식(3),(4)를 만족하지 않으므로 이 노드들이 소프트웨어로 분할 될 경우에 대한 서브 트리는 생성하지 않는다. 제안 알고리즘을 수행한 최종 분할 결과는 1,2,3,4번 노드가 모두 하드웨어로 분할되는 경우이며 이러한 분할 결과는 시간 제약 조건 14에서 최적의 결과이다.

4. 실험결과

제안 알고리즘의 성능평가를 위하여 논문 [5]의 6개의 노드를 가진 예제를 사용하여 시뮬레이티드 어닐링과 제안된 알고리즘을 비용과 성능을 중심으로 비교하였다. [표 2]와 [표 3]은 시뮬레이티드 어닐링과 제안된 알고리즘에 대한 비용과 성능 비교이다.

시간 제약조건	시뮬레이티드 어닐링[5]	제안 알고리즘
35	296	296
50	296	266
60	285	256
65	285	169
70	285	169
90	250	129
95	146	121
100	146	64
105	111	50
110	135	50
115	100	50
125	150	24
130	150	0

[표 2]비용 비교

시간 제약조건	시뮬레이티드 어닐링[5]	제안 알고리즘
35	32	32
50	32	32
60	51	57
65	51	62
70	51	62
90	85	87
95	85	93
100	85	97
105	104	103
110	104	103
115	114	103
125	121	122
130	121	128

[표 3]성능 비교

[표 2]와 [표 3]은 국부 최적치에서 종료되는 경우가 많은 시뮬레이티드 어닐링보다 제안 알고리즘이 비용과 성능 비교에서 최적임을 보여준다.

참고 문헌

[1].R.K. Gupta,C.Coelho, and G. De Micheli,"Synthesis and Simulation of Digital Systems Containing Interacting Hard ware and Software Components",29th ACM,IEEE Design A utomation Conference ,pp. 225-230,1992.

[2]. J.Henkel,R .Ernst,U.Holtman, T.Benner, "Adaptation o f partitioning and high-level synthesis in hardware/softw are co-synthesis" , in proceedings,ICCAD-94,pp.96-100,I EEE Computer Society Press,1994.

[3].D.Henkel,J.Herrmann and R.Ernst, "An Approach to t he adaptation of cost parameters in the COSYMA System international,Workshop on Hardware/SoftwareCodesign,pp. 100-107,1994.

[4]Frank Vahid, Jie Gong and Daniel D. Gajski, "A Binary -Constraint Search Algorithm for Minimizing Hardware du ring Hardware Software Partitioning", In EURO-DAC 94, pp.214-219, 1994.

[5]J.I.Hidalgo, J.Lanchares, "Functional Partitioning for Ha rdware-Software Codesign Using Genetic Algorithm", Pro ceedings of the 23rd EUROMICRO conference, Budapest (Hungary), 1-4 IEEE Press. pp.631-638, 1997.

[6] F. Vahid,"Modifying Min-Cut Hardware and Software for Functional Partitioning",International Workshop on Har dware/Software Codesign, pp. 43--48, March 1997

[7]F. Vahid and T.D.M."Extending Keringhan and Lin Heu ristics for Hardware and Software for Functional Partition ing", Kluwer Journal on Design Automation of Embedded Systems, Vol. 2, No. 2, pp.237-261, March 1997

[8].Richard Neaplitan, Kumas Naimipour,"FUNDATIONS O F ALGOARITGMS",D.C.Health and Company,1996

[9]P. Eles, Z. Peng, K. Kuchcinski, A. Doboli, "System Le vel Hardware/Software Partitioning Based on Simulated A nnealing and Tabu Search", Journal on Design Automation for Embedded Systems, vol. 2, pp.5-32, 1997.