제주교육대학교 논문집 (2003)
제32집 pp. 319~332

파이프라인이 고려된 하드웨어 소프트웨어 분할 개선 방안

An Improved Method for Hardware/Software Partitioning considering Pipelining

김 종 훈* · 이 면 재**

〈목 차〉

〈요 약〉

비용 또는 성능 제약 조건을 만족시키기 위해 내장형 시스템은 일반적으로 ASIC과 같은 하드웨어 부분과 DSP와 같은 소프트웨어 부분의 조합으로 구현된다. 소프트웨어 부분은 하드웨어 부분에 비해 설계 시간이 적게 소요되고, 설계 비용이 저렴하고 성능이 낮지만 프로그램 변경이 용이하다. 하드웨어 부분은 설계와 제조과정이 어렵지만

* 제주교육대학교 컴퓨터교육과 조교수
** 홍익대학교 대학원 전자계산학과 박사과정

성능이 높다. 통합 설계는 성능이 중요한 태스크들은 하드웨어 부분에서 수행되고 그렇지 않는 부분은 소프트웨어 부분에서 실행되는 일을 다룬다. 통합 설계 과정 중에서 주어진 시스템 사양 명세를 하드웨어 부분과 소프트웨어 부분으로 분할하는 것을 하드웨어/소프트웨어 분할이라고 하며, 하드웨어/소프트웨어 분할은 시스템의 성능을 결정하는 중요한 단계이다. 성능이 중요한 시스템인 경우 모든 태스크들을 하드웨어 부분으로 수행하여도 성능이 만족되지 않는 경우가 발생될 수 있으므로 파이프라인 방법이 필요하다. 본 논문에서는 기존의 파이프라인을 고려한 하드웨어/소프트웨어 분할 방법을 살펴본 후 계산시간을 단축할 수 있는 개선된 알고리즘을 제안한다.

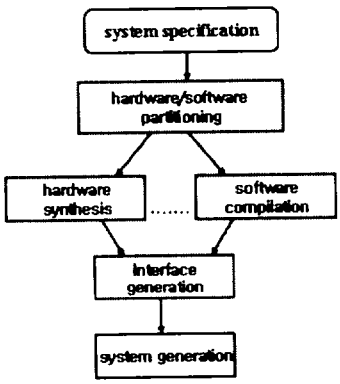
〈ABSTRACT〉

In order to cost and performance requirements, Embedded systems generally consist of application specific hardware parts and software parts, i.e processors like DSPs. In comparison to hardware parts, software parts offer high-programmability, lower design time, and comparatively lower cost and lower performance. On other hand, hardware parts are more expensive to design and fabricate, but offer comparatively higher performance. However, hardware parts provide better performance. Thus, for a given system, these component are selected such that the performance critical tasks are performed rapidly in hardware part, and the less critical parts that requires higher programmability are performed in software parts. Hardware/Software codesign deals with these problem of designing embedded systems. The co-design phase, during which the system specification is partitioned onto the hardware part and software part, is called hardware/software partitioning. This phase represents one key issue during codesign process because system this phase determines system performance. For performance critical design, it is not sufficient to only implement the critical tasks as hardware parts, but it is also necessary to pipeline the system.

In this paper, after examining the existing hardware/software partitioning considering pipelining, we present an improved hardware/software partitioning to be shorten computation time.

Ⅰ. 서 론

디지털 이미지 프로세싱이나 통신에 사용되는 디지털 시스템은 복잡하다. 이러한 시스템의 복잡도는 점점 증가하는 동시에 해당 제품을 시장에 적절한 때에 출시해야 하는 부담감은 높아져 가고 있다[1]. 특히 이러한 이유로 주어진 성능을 만족할 수 있는 제품을 빠른 시간에 설계해야 하기 때문에 상위 수준의 설계 자동화는 필요하게 되었다. 상위 수준의 설계 자동화 중에서 DSP(Digital Signal Processor) 같은 프로세서로 구성되는 소프트웨어 부분과 ASIC(Application Specific Integrated Circuit) 또는 FPGA로 구성되는 하드웨어 부분을 동시에 고려하여 설계하는 방법을 통합 설계 또는 시스템 수준의 설계라고 한다. 이러한 통합 설계는 여러 단계로 나뉘어 지고, 특히 하드웨어/소프트웨어 분할 과정은 시스템의 기능을 서술하는 각각의 행위 기술들을 하드웨어 부분 또는 소프트웨어 부분에 분할하는 과정으로 시스템 성능을 결정하는 중요한 단계이다[1,2].



[그림 1] 통합 설계 과정

[그림 1]은 통합 설계 과정이다[9]. 시스템은 태스크 또는 프로세스의 집합으로 명시된다. 그리고 이러한 태스크들이 하드웨어 부분에서 실행될 것인지 또는 소프트웨어 부분에서 실행될 것인지를 결정하는 하드웨어/소프트웨어 분할 과정이 수행된다. 분할 결과는 주어진 제약 조건을 만족해야만 한다. 하드웨어 부분으로 분할이 결정된 태스크들은 상위 수준 합성 단계를 통해 ASIC이 생성되고, 소프트웨어 부분으로 분할이 결정된 태스크들은 소프트웨어 컴파일 과정을 통해 최적화된다. 하드웨어 부분으로 분할이 결정된 태스크들과 소프트웨어 부분으로 분할이 결정된 태스크들간에는 데이터

또는 제어 의존도가 존재하므로 상호 통신을 위한 인터페이스 생성 과정이 수행된다. 최종적으로 생성된 ASIC과 소프트웨어 부분, 그리고 인터페이스를 통합하는 시스템 통합 과정을 통해 설계자가 원하는 내장형 시스템이 생산된다.

하드웨어/소프트웨어 분할을 수행할 때 ASIC은 설계와 제조 비용이 많이 들지만 비교적 높은 성능을 제공하므로 성능이 중요한 태스크들은 빠른 시간에 수행하도록 하고 그렇지 않는 태스크들은 프로세서에서 수행하도록 하는 것이 분할의 기본 원리이다. 또한 하드웨어/소프트웨어 분할을 수행할 때의 고려 사항은 다양하다. 시스템의 제약조건이 성능이거나 면적인 경우 주어진 제약 조건을 만족하면서 시스템에 소요되는 비용을 최소화해야 한다. 또한 시스템의 아키텍처는 주어진 자원에 따라 1개의 하드웨어 부분과 1개의 소프트웨어 부분 또는 여러 개의 소프트웨어 부분과 여러 개의 하드웨어 부분 등이 될 수 있으며 시스템의 성능을 높이기 위하여 파이프라인을 고려할 수도 있다.

본 논문에서는 내장형 시스템을 설계할 때 입력으로 주어진 모든 태스크들을 성능이 빠른 하드웨어 부분으로 분할하여도 성능이 만족되지 않는 경우 성능을 높이기 위해 파이프라인이 고려된 하드웨어/소프트웨어 분할 알고리즘을 살펴본다[1,2]. 또한 기존 파이프라인이 고려된 논문의 계산 시간이 오래 소요될 수 있는 단점을 보완하는 개선된 하드웨어/소프트웨어 분할 알고리즘을 제안한다. 본 논문은 2절에서 관련 연구를 기술하고, 3절에서 파이프라인을 고려한 기존 분할 알고리즘과 이에 대한 문제점을 기술하고, 4절에서 제안된 하드웨어/소프트웨어 분할 알고리즘을 기술하고, 5절에서 결론 및 향후 연구방향을 기술하였다.

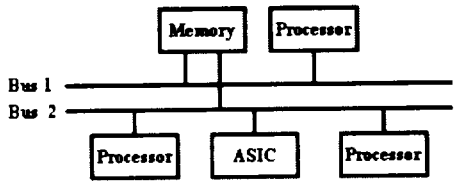
Ⅱ. 관련 연구

지금까지 여러 가지 통합 설계 시스템이 개발되어져 왔으나 각각 고유한 특징을 가지고 있다. 이러한 시스템들 중에서 Cosyma[3, 12, 13], Vulcan[4]와 같은 통합 설계 방법은 입력으로 주어진 태스크나 프로세스들을 하드웨어 또는 소프트웨어로 구현할 경우의 비용, 면적, 실행 시간 등의 정보를 예측하는 것과 하드웨어와 소프트웨어로 분할하는 일에 주안점을 두었다. Ptolemy[5, 6]는 하드웨어/소프트웨어 통합 시뮬레이션과 코드 생성에 초점을 두었다. Chinnook[7]은 하드웨어/소프트웨어 인터페이스 합

성에 주안점을 두었다. BCS(Binary Constraint Satisfaction)알고리즘은 이진 검색 방법과 시뮬레이티드 어닐링 방식을 동시에 사용하는 알고리즘이다[14]. 위에서 사용된 하드웨어/소프트웨어 분할 알고리즘들은 하드웨어/소프트웨어 분할을 수행할 때 파이프라인을 고려하지 않았다. 루프와 파이프라인이 고려된 하드웨어/소프트웨어 분할 알고리즘의 대상 아키텍처는 1개의 하드웨어와 소프트웨어 부분으로 구성되고 시간 또는 면적 제약조건을 만족하면서 MII(Minimum Initiation Interval)에 근접하는 파이프라인 분할 상태의 결과를 구했는데, 분할에 사용된 방법은 branch and bound이다[10,11,13]. 본 논문은 n개의 소프트웨어 부분과 m개의 하드웨어 부분으로 구성되고 파이프라인을 고려된 하드웨어/소프트웨어 분할을 수행하는 알고리즘을 살펴본 후 계산 시간이 많이 소요될 수 있는 단점을 보완하는 개선된 하드웨어/소프트웨어 분할 알고리즘을 제안한다.

Ⅲ. 기존 하드웨어/소프트웨어 분할

대상 아키텍처는 [그림 2]이다. 1개 이상의 소프트웨어 부분과 하드웨어 부분, 그리고 메모리로 구성되고 이들 간의 통신은 버스를 통하여 수행된다. 전역적인 데이터와 파이프라인을 수행하는데 필요한 데이터는 메모리에 저장된다. 본 논문의 대상 아키텍처는 1개의 하드웨어 부분과 3개의 소프트웨어 부분으로 구성된다고 가정한다.



[그림 2] 대상 아키텍처

입력으로 태스크들과 각 노드들에 대한 하드웨어 실행시간 테이블과 소프트웨어에서 실행시간을 나타내는 프로세서 실행시간 테이블과 프로세서 할당 비용 테이블, 그리고 스테이지 지연 시간 즉 성능 제약 조건이 주어진다.

[그림 3]은 입력으로 주어진 정보이다. [그림 3.1]은 주어진 시스템 사양 명세이고

[그림 3.2]는 소프트웨어 실행시간 테이블로서 Beh는 태스크, processor는 태스크를 실행할 수 있는 프로세서, 그리고 Exec는 해당 프로세서의 실행시간이다. [그림 3.3]은 프로세서 할당 비용 테이블로써 임의의 태스크가 해당 프로세서에 할당되었을 때 소요되는 프로세서의 비용이다.

input

```
Behavior TOP type sequential is
Begin
  A,B: (TOC,true,C);
  C:(TOC,true,D);
  D:(TOC,true,STOP);
Behavior AB tuype concurrent is
begin
  A:;
  B:;
  Behavior A type leaf is
    begin
      -- contain VHDL code
    end A;
  Behavior B type leaf is
    begin
      -- contain VHDL code
    end B;
End AB;
Behavior C type leaf is
  begin
    -- contain VHDL code
  end C;
Behavior D type leaf is
    begin
      -- contain VHDL code
    end D;
End TOP;
```

[3.1] Specification

성능 제약 조건=10

Beh	Processor	Exec
A	P1	7
A	P2	9
A	P3	11
B	P1	5
B	P2	8
B	P3	8
D	P1	2
D	P2	8
D	P3	4

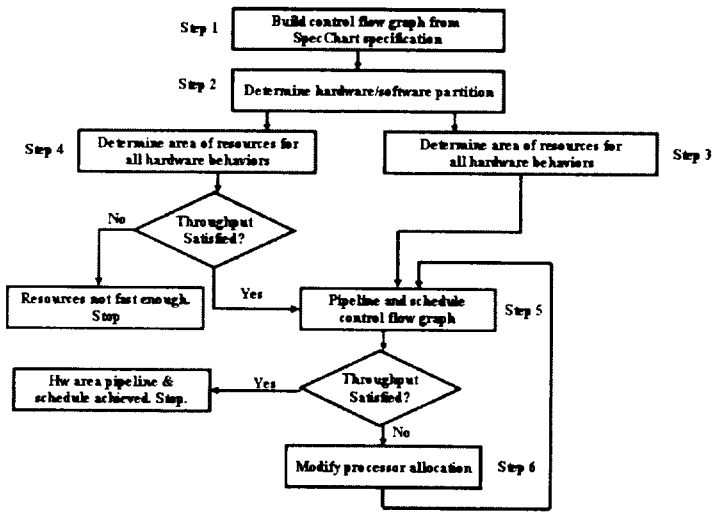
[3.2] 소프트웨어 실행시간 테이블

Allocation	Cost
P3	10
P2	30
P1	50
P3P3	20
P3P2	40
P2P2	60
P2P1	80
P1P1	100
P3P3P3	30
.....	.....
P1P1P1	150

[3.3] 프로세서 할당 비용 테이블

[그림 3] 입력으로 주어진 정보

[그림 4]는 기존 하드웨어/소프트웨어 분할 알고리즘이다.

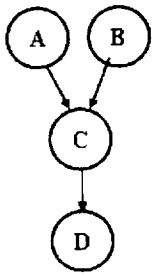


[그림 4] 파이프라인을 고려한 기존 하드웨어/소프트웨어 분할 알고리즘

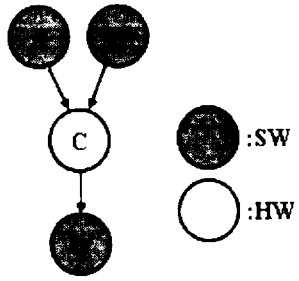
Step 1은 주어진 사양 명세를 분석하여 CFG(Control Flow Graph)를 만드는 과정으로 [그림 5]는 [그림 3.1]의 사양 명세의 입력과 출력을 분석하여 산출된 CFG이다. 노드 A, B, C는 입력으로 주어진 태스크이고 간선은 데이터 또는 제어 의존도이다.

Step 2는 입력으로 주어진 모든 노드들에 대한 초기 분할 상태를 결정하는 단계이다. CFG의 각 노드에 대해 프로세서 실행 시간 테이블을 참조하여 성능 제약

조건을 만족하고 해당 노드를 실행할 수 있는 프로세서가 있다면 소프트웨어로 분할되고 그렇지 않다면 하드웨어로 분할된다. 노드 A는 프로세서 P1과 P2의 실행시간이 성능 제약 조건 10을 만족하므로 소프트웨어로 분할되고 노드 B는 프로세서 P1, P2, P3의 실행시간이 성능 제약 조건 10을 만족하므로 소프트웨어로 분할된다. 그러나 노드 C는 소프트웨어 부분에서 실행 가능한 프로세서가 없으므로 하드웨어로 분할된다.



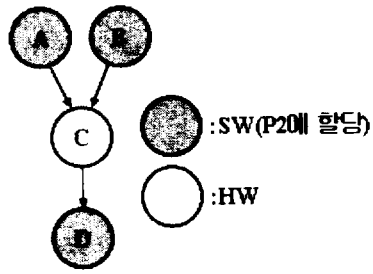
[그림 5] CFG



[그림 6] 초기 하드웨어/소프트웨어 분할 결과

[그림 6]은 [그림 5]의 CFG에 대한 초기 하드웨어/소프트웨어 분할 결과이다.

Step 3은 [그림 3.3]의 프로세서 할당 비용 테이블을 참조하여 주어진 성능 제약조건을 만족하는 실행시간을 갖는 프로세서들 중에서 가장 비용이 낮은 프로세서를 선택하여 소프트웨어로 분할이 결정된 노드들을 해당 프로세서에 할당하는 단계이다. 프로세서 P3의 비용이 가장 낮지만 프로세서 P3은 A 노드의 실행 시간이 성능 제약 조건 10을 위배하므로 프로세서 P3 다음으로 비용이 낮은 프로세서 P2에 소프트웨어로 분할이 결정된 모든 노드들이 할당된다. [그림 7]은 소프트웨어에 분할이 결정된 노드들이 프로세서 P2에 할당됨을 보여준다.



[그림 7] 프로세서 P2에 소프트웨어로 분할이
결정된 노드들이 할당

Step 4는 하드웨어로 분할된 노드들을 하드웨어로 구현했을 경우의 정보들, 즉 면적, 파이프라인 스테이지 개수, 구성 요소 개수 등을 하드웨어 합성 툴을 이용하여 하드웨어 실행 시간 테이블을 완성하는 단계이다[8]. [표 1]은 [그림 7]의 하드웨어로 분할이 결정된 노드가 C이므로 노드 C를 하드웨어로 구현했을 경우 면적과 실행시간을 나타내는 하드웨어 실행시간 테이블이다. 하드웨어 부분의 가능한 파이프라인 스테이지 수는 2라고 가정한다.

Beh	Area	Exec
C	1000	8

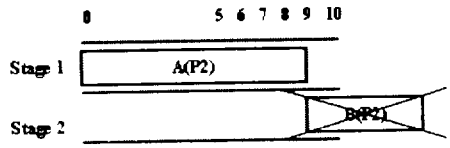
[표 1] 하드웨어 실행시간 테이블

Step 5는 소프트웨어로 분할이 결정된 노드들이 할당된 프로세서와 하드웨어 실행시간 테이블 정보를 가지고 파이프라인이 고려된 스케줄링을 수행한다. 이때 [표 2]는 리스트 스케줄링(List Scheduling)이다[16]. [그림 8]은 프로세서 P2를 사용하여 리스트 스케줄링을 수행한 결과이다. 노드 A가 Processor 2에 배정되고 스테이지 1에 할당되어 0부터 9까지의 시간을 사용한다. 이후에 노드 B를 스테이지 1 또는 2에 할당하려고 했으나 성능 제약 조건 10을 만족시킬 수 없으므로 성능 제약 조건을 만족하지 못하는 스케줄링 결과가 된다.

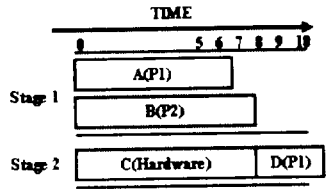
```
1. For every node in the CFG, determine the longest compilation time from
that node till any output node, and assign node priorities.
2. Initialize the utilization list of all processors.
3. Loop
4.      form a new ready list.
5.      LOOP
6.        If (node is software)
7.              Find processor and corresponding time slot that gives earliest
completion time.
8.              If(no avaliable processor)
9                       Stop. No feasible solution
10.             Else
11.                     Assign node to processor & time slot.
12.                     Update utilization list of processor.
13.             Endif.
14.       Else if (node is type hardware)
15.             Assign hardware to earliest feasible time.
16.       End if
17.       Mark node as scheduled and remove from ready list
18.     Until (ready list empty)
19. Until (all node in CFG are scheduled)
```

[표 2] 리스트 스케줄링

소프트웨어로 분할이 결정된 노드들을 프로세서 P2에 배정한 스케줄링 결과가 성능 제약 조건을 만족하지 못하므로 step 6에서는 [그림 3.3]의 프로세서 할당 비용 테이블에서 P2 프로세서보다 비용이 한 단계 높은 프로세서, 즉 성능이 빠른 프로세서인 P1에 소프트웨어로 분할이 결정된 노드들을 배정하여 step 5와 6의 과정을 수행한다. 이와 같이 step 5와 6의 과정을 성능 제약 조건이 만족될 때까지 반복 수행한다. [그림 9]는 최종 스케줄링 결과로써 성능 제약 조건 10을 만족하고 사용된 소프트웨어 프로세서는 P1과 P2이다. [그림 9]의 결과는 프로세서 P2부터 P2P1 프로세서 그룹까지 Step 5와 6의 과정을 6번 반복 수행한 결과이다.



[그림 8] 프로세서 P2를 사용한 경우의 리스트 스케줄링 결과

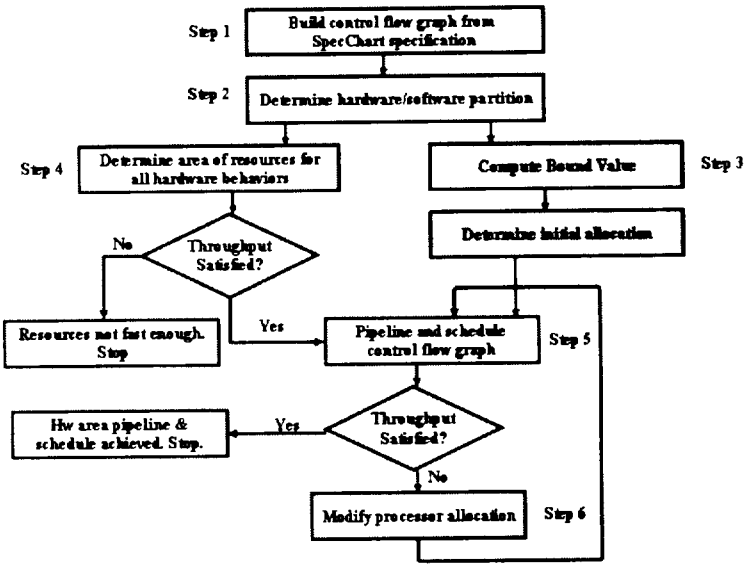


[그림 9] 최종 스케줄 결과

Ⅳ. 제안된 분할 알고리즘

기존의 하드웨어/소프트웨어 분할 알고리즘은 소프트웨어 부분에 분할이 결정된 태스크들의 독립적인 실행 시간만 고려하여 성능제약 조건을 만족하는 범위 내에서 무조건 비용이 낮은 프로세서부터 할당을 하였다. 그러므로 만약 n개의 프로세서가 있다면 최악의 경우 1개의 프로세서부터 n개로 조합 가능한 프로세서 개수까지 step 5부터 6을 반복 수행해야 하므로 계산시간이 많이 소요될 수 있다. 또한 이 계산 시간은 태스크의 개수와 파이프라인 스테이지 수, 그리고 프로세서 개수에 비례한다. 그러므로 제안된 알고리즘에서는 해당 노드들을 프로세서 그룹 중에서 가장 빠른 프로세서에 할당 시켰을 경우의 실행시간을 하한 값으로 계산하여 이를 만족하는 적절한 프로세서 그룹부터 step 5부터 6을 수행하여 계산 시간이 단축되는 알고리즘을 제안한다. [그림 10]은 제안된 하드웨어/소프트웨어 분할 알고리즘이다.

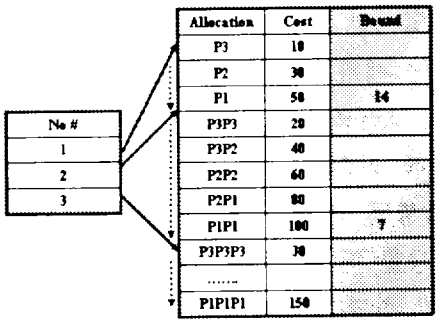
Step 1과 2는 [그림 3]의 기존 알고리즘과 동일하고 step 3에서 초기 프로세서를 사용하여 소프트웨어로 분할이 결정된 노드들을 할당할 때 성능 제약 조건을 만족하는 범위 내에서 무조건 비용이 낮은 프로세서부터 할당하는 것이 아니라 하한 값을 이용하여 이 값을 만족하는 프로세서 그룹부터 할당하도록 개선하였다. 식(1)은 하한 값을 의미한다. n은 소프트웨어 부분의 프로세서 개수이고 는 소프트웨어로 분할이 결정된 노드이고 PS delays는 성능 제약 조건이다. 따라서 하한 값, 즉 Bound(n)는 n개의 프로세서를 가지고 파이프라인을 고려하여 스케줄링을 수행 할 경우 소요되는 최소의 시간을 의미한다.



[그림 10] 제안된 하드웨어/소프트웨어 분할 알고리즘

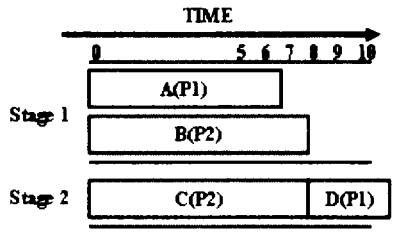
$$Bound(n) = \sum_{i \in SW} \frac{t_{i,fast}}{n} \leq PS\ delay \quad \cdots\cdots\cdots\cdots 식(1)$$

예를 들어 하나의 프로세서를 사용할 경우 n은 1이 된다. 하한 값은 소프트웨어로 분할이 결정된 노드 A, B, D들을 가장 실행이 빠른 프로세서인 프로세서 P1에 할당한 경우의 실행시간의 합 14((7+5+2)를 프로세서 개수인 1로 나눈 값이다. [그림 11]은 변경된 프로세서 할당 비용 테이블인데 하한 값이 추가되었다. No#는 조합 가능한 프로세서 개수이고 에지는 각 프로세서 개수로 조합 가능한 프로세서 그룹 중에서 가장 비용이 낮은 그룹을 지시한다. 그리고 Bound은 해당 프로세서 개수의 하한 값이다. 즉 프로세서 1개로 가능한 최소 실행시간은 14임을 의미한다.

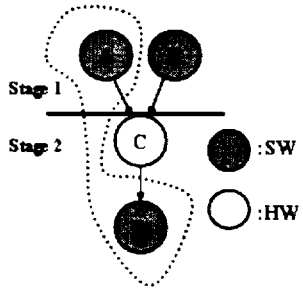


[그림 11] 변경된 프로세서 할당 비용 테이블

Step 3은 입력으로 주어진 성능 제약 조건, 즉 10보다 작은 최초의 하한 값, 즉 7을 검색한 후 해당 자원의 개수로 조합하여 가장 비용이 낮은 프로세서 그룹인 P3P3을 초기 프로세서로 선택한다. 그리고 소프트웨어로 분할이 결정된 노드들을 P3P3 프로세서부터 할당한다. 그리하여 불필요하게 P2부터 프로세서 할당이 시작되는 기존의 하드웨어 소프트웨어/분할 알고리즘의 계산 시간을 단축할 수 있다. 위 예제의 경우 최종 파이프라인을 고려한 스케줄링 결과는 [그림 12]와 같으며 최종 하드웨어/소프트웨어 분할 결과는 [그림 13]이다. 기존 하드웨어/소프트웨어 분할 알고리즘의 Step 5부터 6까지의 반복횟수는 6이지만 제안된 알고리즘은 프로세서 P3P3부터 P2P1까지 4회로 감소된다.



[그림 12] 최종 스케줄 결과



[그림 13] 최종 하드웨어/소프트웨어 분할 결과

Ⅵ. 결론 및 향후 연구 방향

본 논문은 파이프라인을 고려한 분할을 수행한 기존 논문을 살펴보았다. 기존 논문은 파이프라인이 고려되었지만 하드웨어/소프트웨어 분할 이후 각 태스크의 실행을 독립적으로 간주하여 성능 제약 조건이 만족하는 범위 내에서 비용이 낮은 프로세서를 자원으로 할당하였다. 그러므로 프로세서 개수, 파이프라인 스테이지 개수, 그리고 태스크의 수가 많아질수록 리스트 스케줄링에 대한 반복횟수가 많아져 계산 시간이 많이 소요되는 단점이 발생될 수 있다. 본 논문에서는 소프트웨어로 분할이 결정된 태스크들을 프로세서에 할당할 때 프로세서 개수마다 하한 값을 만족하는 해당 프로세서의 그룹부터 초기 자원 할당 프로세서로 사용하여 계산 시간이 단축될 수 있도록 기존 하드웨어/소프트웨어 분할 알고리즘을 개선하였다. 이러한 계산 시간의 단축은

시스템 사양 명세부터 내장형 시스템의 출시까지 소요되는 시간을 줄일 수 있을 것이다. 향후에는 본 논문을 구현하여 다양한 워크 벤치 데이터를 통하여 비교 할 예정이다. 또한 1개의 하드웨어 부분과 n개의 소프트웨어 부분으로 구성되는 기존 대상 아키텍처를 m개의 하드웨어 부분과 n개의 소프트웨어 부분으로 확장하여 실험할 예정이다.

《참고 문헌》

[1] Smita Bakshi and Daniel D.Gajski,"Partitioning and Pipelining for Performance Constrained Hardware/Software System",December 1999,Vol7,Number 4,IEEE Transaction on VLSI System, pp 419-432.

[2] Smita Bakshi and Daniel D.Gajski, "Hardware/Software partitioning and Pipelining",Proceedings of Design Automation Conference, pp.713-716, June 1997.

[3] R.Ernst,J.Henkel, and T.Benner. "Hardware-Software cosynthesis for micro controllers",pp 64-75,1994.

[4] R.K Gupta and G.De Micheli,"Hardware-software cosynthesis for digital systems", IEEE Design and Test of Computers,10(3):29-41,1993.

[5] A.Kalavade and E.Lee, "A Hardware/software codesign using ptolemy -a case study", In Proc. of the IFIP International Workshop on Hardware/Software Co-Design",1992.

[6] A.Kalavade and E.Lee, "A Hardware/Software codesign methology for DSP applications",In IEEE Design and Test of Computers,1993.

[7] P.H,Chou,R.B.Ortega,and G.Borriello,"The Chinook hardware/software co-synthesis system",In International Symposium on System Synthesis , 1995.

[8] Smita Bakshi and Daniel D.Gajski. "Hierarchical pipelining with hardware/software partitioning",Technical Report 96-38,Dept. of Information and Computer Science University of California,Irivine,1996.

[9] M.D Edwards, J.Forrest, "softwate acceleration using programmable hardware devices", IEE proc-comput. Design Tech, Vol.143, No.1, January 1996.

[10] Karam.S. Chatha and Ranga Vermuri, "A tool for partitioning and Pipelined Scheduling of Hardware-Software System"

[11] Binvoo Srinivasan and ranga Vermuri, "A Retiming Based relaxation Heuristic for Resource-Constrained Loop pipelining"

[12] J.henkel, Th.Benner, R.Ernst, "Hardware generation and partitioning effects in the COSYMA system".

[13] D.herrmannm J.Henkel, R.Ernst, "An Approach to the Adaptation of Estimated Cost Parameters in the COSYMA systems".

[14] Frank Vahid, Jie Gong and Daniel Gajski, "A Binary Constraint Search Algorithm for Minimizing Hardware during Hardware/Software paritioning", European Design Automation Conference, pp214-219 september 1994.

[15] Bakshi and D. D. Gajski, "Partitioning and Pipelining for performance-constrained hardware/software system", IEEE Transaction on VLSI system, pp.419-432,December 1999

[16] Bakshi Daniel Gajski,"Performance-constrained hierarchical pipelining for behaviors, loops, and operations", (1): 1-25 (2001)