분산된 VLIW 구조에서의 최대 전력 최소화 방법

(Peak Power Minimization for Clustered VLIW Architectures)

서 재 원 † 김 태 환 †† 정 기 석 †††
(Jaewon Seo) (Taewhan Kim) (Ki-Seok Chung)

요 약 VLIW 구조는 다량의 데이터를 처리하는 멀티미디어 애플리케이션에 매우 적합한 구조로서, 이 같은 종류의 애플리케이션에 대해 높은 수준의 병렬 처리를 가능케 한다. 이러한 병렬성을 더욱 증대시키기 위하여 시스템을 확장하는 경우에 있어, 분산된 VLIW 구조는 그렇지 않은 구조에 비해 큰 강점을 갖는다. 하지만 여러 개의 분산된 클러스터를 하나의 구조 속에 포함하는 것은 필연적으로 적지 않은 양의 하드웨어를 요구하고, 이로 말미암아 전체 시스템에서 소모되는 전력 문제가 중요한 이슈로 대두된다. 본 논문에서는 분산된 VLIW 구조에서 전체 시스템의 성능 제한 조건을 만족시키는 동시에 최대 전력 소모량을 줄이는 효과적인 알고리즘을 제시한다. 일련의 실험을 통해 제시된 알고리즘이 최대 30.7%의 최대 전력 소모 감소 효과를 얻을 수 있음이 확인되었다.

키워드 : VLIW, 최대 전력, 스케줄링

Abstract VLIW architecture has emerged as one of the most effective architectures in dealing with multimedia applications. In multimedia applications, there is ample potential for parallelizing the execution of multiple operations because such applications typically have data intensive processing which often has limited data and/or control dependencies. As the degree of instruction-level parallelism increases, non-clustered VLIW architectures scale poorly because of the tremendous register port pressure. Therefore, clustered VLIW architecture is definitely preferred over non-clustered VLIW architecture when a higher degree of parallelizing is possible as in the case of multimedia processing. However, having multiple clusters in an architecture implies that the amount of hardware is quite large, and therefore, power consumption becomes a very crucial issue. In this paper, we propose an algorithm to minimize the peak power consumption without incurring little or no delay penalty. The effectiveness of our algorithm has been verified by various sets of experiments, and up to 30.7% reduction in the peak power consumption is observed compared with the results that is optimized to minimize resources only.

Key words : VLIW, peak power, scheduling

1. 서 론

오늘날의 고성능 내장형(embedded) 시스템에서는 멀티미디어 데이터와 디지털 신호에 대한 처리가 매우 커다란 부하로 자리 잡아 가고 있으며, 이들을 얼마나 효율적으로 처리할 수 있느냐가 시스템의 성능을 결정짓는 중요한 잣대가 되고 있다. 이에 맞추어 현재 매우 다양하고 전문화된 멀티미디어 프로세서와 디지털 신호 프로세서가 등장하였는데, 이들 프로세서들은 고성능을 추구하기 위하여 VLIW(Very Long Instruction Word), 수퍼스칼라(super scalar) 혹은 SIMD(Single Instruction Multiple Data)등의 구조를 채택하고 있다. 이중

· 이 연구는 첨단정보기술 연구센터(AITrc)를 통하여 과학재단의 지원을 받았음.
† 비 회 원 : 한국과학기술원 전자전산학과
jwseo@jupiter.kaist.ac.kr
†† 종신회원 : 한국과학기술원 전자전산학과 교수
tkim@cs.kaist.ac.kr
††† 종신회원 : 홍익대학교 컴퓨터공학과 교수
kchung@sc.hongik.ac.kr
논문접수 : 2002년 9월 27일
심사완료 : 2003년 1월 28일

VLIW 구조는 하드웨어 구축에 드는 비용이 상대적으로 적음에도 불구하고 발전된 컴파일러 기술의 도움에 힘입어 매우 훌륭한 성능을 보여주고 있기 때문에, 근래에 들어 가장 선호 받는 시스템 구조 중의 하나로 여겨지고 있다.

멀티미디어 애플리케이션은 일반적으로 종속관계가 많지 않은 방대한 양의 데이터에 대한 처리를 포함하는데, 이 때 종속관계에 놓여 있지 않은 데이터에 대한 연산들은 이론적으로 동시 수행이 가능하다. VLIW 구조는 이와 같이 동시 수행 가능한 연산이 많은 애플리케이션에 매우 적합한 구조로서 등장하였다. VLIW 구조의 명령어 수준 병렬성(instruction level parallelism)을 더욱 증대시키기 위하여 시스템을 확장하는 경우, 내장된 각 연산 유닛의 레지스터 파일(register file) 입출력 포트(port)에 대한 과도한 요구로 말미암아 기존의 단일 구조는 한계점을 지니고 있다. 분산된(distributed 또는 clustered) VLIW 구조는 이를 효과적으로 해결하기 위하여 제시된 구조로서, 동일한 혹은 동일하지 않은 여러 개의 개별 VLIW 구조를 하나의 구조 안에 포함하고 있다 (그림 1). 각 VLIW 클러스터(cluster)는 지역 레지스터 파일과 연산 유닛을 내장하고 있으며, 이 연산 유닛의 입력 값과 출력 값은 같은 클러스터 내의 레지스터 파일에서 읽혀지며 저장된다. 따라서 레지스터 파일의 크기와 입출력 포트의 수를 항상 일정 수준으로 유지할 수가 있게 된다. 이러한 분산 구조에서 어떠한 연산에 필요한 입력 값이 다른 클러스터의 레지스터 파일에 저장되어 있는 경우가 생길 수 있는데, 이 때는 클러스터 간 데이터 전송을 통해 이 값을 읽어오게 되며, 이를 위해 각 클러스터는 버스(bus)에 의해 연결이 되어 있다.

오늘날의 대부분의 고성능 프로세서들은 심각한 전력 문제를 겪고 있는데, 이는 프로세서 회로의 복잡도가 증가함에 따른 많은 수의 논리 게이트들의 집적과 높은 성능을 내기 위한 빠른 클락 속도에 기인한다. 분산된 VLIW 구조의 경우 여러 개의 클러스터를 하나의 구조 안에 포함하고 있으므로 그 집적도가 매우 크며, 일반적으로 고성능 애플리케이션에 적합한 빠른 속도의 클락을 사용하므로 전체 시스템에서 소모되는 전력 문제가 더욱 중요한 이슈로 대두된다. 특히 평균적으로 소모되는 전력뿐만 아니라 순간 소모되는 최대 전력(peak power)를 줄이는 것 또한 중요해 지는데, 이는 고성능 프로세서 구조에서는 최대 전력 소모가 칩의 수명과 시스템을 안정성을 결정짓는 핵심 요소로 작용하고 있기 때문이다.

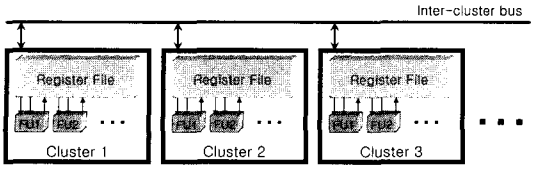

그림 1 분산된 VLIW 구조

본 논문에서는 분산된 VLIW 구조에서의 최대 전력 소모를 줄이는 문제를 다루기로 한다. 여기에서 제안된 알고리즘은 초기 해를 생성하는 단계와 이를 반복적으로 개선시키는 단계로 이루어져 있으며, 이 때 각 연산의 스케줄링과 클러스터 할당을 동시에 고려함으로써 효율적으로 최대 전력을 줄이게 된다. 일련의 실험을 통하여, 제안된 알고리즘이 일반적인 스케줄링 방법에 비하여 최대 30.7%만큼 최대 전력을 줄일 수 있으며, 최적 해를 찾는 소모적인 방법에 비하여 수행 속도는 매우 빠르면서 결과로 얻은 최대 전력 소모량은 평균적으로 2.3%라는 작은 차이를 보인다는 사실을 확인할 수 있었다.

논문은 다음과 같은 구성되었다. 2장에서는 이 문제를 해결하기 위한 우리의 접근 방법의 동기를 예제로서 설명하고, 3장에서는 VLIW 구조에서 전력 문제를 해결하기 위한 기존의 방법들을 간략하게 소개하며, 4장과 5장에서는 제안된 알고리즘을 설명한다. 6장에서는 실험 결과를 제시하고, 마지막으로 7장에서는 전체적인 결론을 내린다.

2. 동 기

이 장에서는 두 가지 예제를 통하여 소프트웨어적 관점에서 분산된 VLIW 구조에서의 최대 전력 소모를 줄이는 방법을 이야기 하고자 한다.

2.1 예제 1 : 연산 스케줄링

표 1은 본 논문에서 사용된 RTL 라이브러리의 대표적인 몇 가지 컴포넌트에 대한 최대 전력 소모량을 나타낸다[1]. 각 컴포넌트에서 소모되는 최대 전력은 몇

표 1 컴포넌트 최대 전력 소모량 [1]

컴포넌트	최대 전력 소모
add_cla_16	0.475 mW
sub_cla_16	0.549 mW
mul_wal_16	1.238 mW
reg_16	0.068 mW
lt_cmp_16	0.113 mW
gt_cmp_16	0.107 mW
eq_cmp_16	0.037 mW

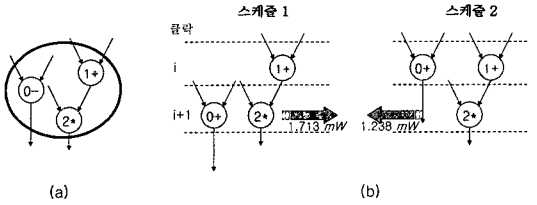

그림 2 연산 스케줄링의 효과를 보여주는 예제

가지 알려진 논리 및 트랜지스터 레벨에서의 측정 방법을 사용하여 구할 수 있다[2,3]. 각 연산의 최대 전력은 그 연산이 수행되는 컴포넌트의 최대 전력과 같으며, 클러스터간 전송 연산은 두 개의 레지스터 컴포넌트에서 수행된다고 하자.

그림 2(a)는 주어진 연산들 간의 데이터 종속성을 나타내는 데이터 흐름 그래프(DFG, Data Flow Graph)를 보여주며, 그림 2(b)는 이 그래프에 따른 종속성을 만족시키는 두 가지의 서로 다른 스케줄을 보여준다. (각 연산은 1 클락 안에 수행을 종료한다고 가정한다.) 여기서 VLIW 구조는 분산되어 있지 않으며, 하나의 덧셈 유닛과 하나의 곱셈 유닛을 갖는다고 하자. 이 때, 스케줄 1은 1.713mW의 최대 전력 소모를 갖게 된다. 만약 우리가 연산 '0+'를 i번째 클락에서 i+1번째 클락으로 이동시킨다면 (스케줄 2), 전체 수행 속도는 스케줄 1과 같지만, 이에 비해 최대 전력 소모는 0.475mW만큼 줄어들게 된다. 이 간단한 예제는 최대 전력 소모를 줄이는 데 있어서 연산을 어떻게 스케줄링 하느냐가 적지 않은 영향을 미치고 있음을 말해준다.

2.2 예제 2 : (클러스터로의) 연산 할당

그림 3은 분산된 VLIW 구조에서의 서로 다른 두 가지 스케줄을 보여준다. 그림 3(a)에서 만약에 클러스터 2에 할당되어있는 '0*' 연산을 클러스터 1으로 재할당한다면 (그림 3(b)), 우리는 최대 전력 소모를 0.136mW만큼 줄일 수 있게 된다. 이 예제는 각 연산을 어느 클러스터로 할당하느냐 역시 최대 전력 소모에 지대한 영향을 미치며, 최대 전력 소모를 줄이기 위해서는 세심한 주의를 기울여 연산을 할당할 필요가 있음을 말해준다.

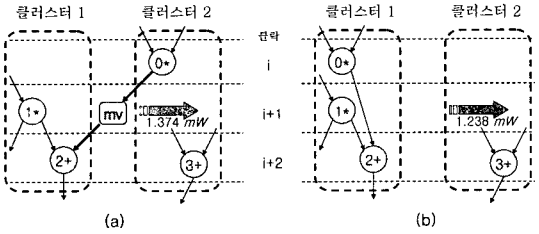

그림 3 연산 할당의 효과를 보여주는 예제

3. 관련된 연구들

VLIW 구조에서의 최적화된 코드 생성에 관한 기존의 많은 연구들이 있는데, 이 연구들은 대부분 수행 속도의 최대화, 필요한 레지스터 개수의 최소화, 레지스터 개수의 제한으로 인한 부가적 메모리 사용의 최소화 등의 이슈에 초점을 두고 있다[4-9]. 근래에 들어 VLIW 구조와 기타 내장형 시스템에서의 전력 소모를 줄이기 위한 여러 가지 시도들이 활발하게 행해지고 있다.

각 명령어의 수행 빈도를 고려하여 평균적으로 소모되는 전력의 양을 간단한 명령어 수준에서 측정하는 방법이 제안되었으며[10], CPU 코어, 레지스터 파일, 캐쉬에 대한 매크로모델(macro-model)을 기반으로 VLIW 구조를 모델링하고 여기에서의 전력 소모를 측정하는 방법이 고안되었다[11].

분산된 VLIW 구조에서 레지스터 파일이 차지하는 면적과 입출력 속도 그리고 여기에서의 전력 소모를 최적화하기 위해서는 중앙 집중형의 레지스터 파일 구조가 아닌 분산된 레지스터 파일 구조가 더욱 유리하다는 것이 입증되었다[12].

유전 알고리즘(genetic algorithm)을 사용하여 메모리 액세스 횟수를 최소화하고 더불어 전력 소모를 줄이는 코드를 생성하는 기법이 연구되었으며[13], 수직적 스케줄링과 수평적 스케줄링을 사용하여 연속된 두 명령어의 인코딩에 대한 해밍 거리(hamming distance) 최소화를 통해 명령어를 읽어오는 과정에서 버스에서 발생하는 스위칭의 횟수를 최소화함으로써 전력을 줄이는 방법이 제안되었다[14]. 각 파이프라인 단계(pipeline stage) 모델을 확장하여 명령어 수준의 내장형 VLIW 코어의 에너지 모델링을 하는 기법이 연구되었다[15].

지금까지 언급한 기존의 연구들은 각자 나름의 관점에서 VLIW 구조에서의 여러 최적화 문제에 대한 해결책을 제시하였는데, 우리의 연구는 다음의 두 가지 점에서 이들과 차별화된다. 첫째는 우리는 분산된 형태의 VLIW 구조를 다루고 있다는 점이고 둘째는 평균 전력보다는 순간 최대 전력을 줄이는 것에 초점을 두고 있다는 것이다.

VLIW 구조에서 시스템의 안정성을 증가시키고 누설 전류를 최소화하기 위해 최대 전력을 줄이기 위한 방법이 [16]에서 제시되었지만, 이 방법은 오직 분산되지 않은 VLIW 구조만을 다룬다는 점에서 우리의 연구와는 다르다.

4. 전력 모델과 문제 정의

4.1 전력 모델

O_i를 i번째 클락에서 수행을 시작한 연산들의 집합이라고 하자. 전체 연산의 집합에서 가장 많은 클락이 걸리는 연산이 d_{max} (클락) 단계의 수행을 한다고 하자. 예를 들어, 나누기 연산이 전체 연산의 집합에서 가장 많은 클락이 걸리는 연산이며 수행을 마치기까지 6 클락이 걸린다면 $d_{max}=6$ 이다. 임의의 연산 op에 대해 $p(op, j)$가 이 연산의 j번째 단계에서 소모되는 전력량을 나타낸다고 하자. 프로그램의 수행이 T 클락에서 끝난다고 했을 때, 최대 전력 소모는 다음과 같이 표현 된다 :

$$\max_{1\le i\le T} \sum_{j=1}^{d_{max}} \sum_{op\in O_{i-j+1}} p(op, j)$$

여기에서 제시된 전력 모델은 다소 간단하며 다른 연산과의 상호 작용, 입력 값에 따른 전력 변화 등을 고려하지 못한다는 몇 가지 약점을 가지고 있다. 하지만 우리의 목적은 정확한 전력 모델을 구축하는 것이 아니라, VLIW 구조에서의 전력 소모를 줄이는 것이므로 본 논문에서는 위의 모델을 그대로 사용하도록 한다. 그렇지만 제안된 알고리즘은 특정 전력 측정 방법에 의존적이지 않으므로 [15]에서 제안된 것과 같은 보다 정확한 모델을 사용하게끔 쉽게 확장이 가능하다.

4.2 문제의 정의

우리가 풀고자 하는 문제를 보다 정확히 기술하자면 다음과 같다 : *K개의 클러스터로 구성된 주어진 분산 VLIW 구조에 대해, 주어진 데이터 흐름 그래프 안의 각 연산을 적절한 클러스터의 적절한 연산 유닛에 할당하며 동시에 스케줄링을 하되, 최대 전력 소모를 최소화하라.*

K=1인 경우, 주어진 VLIW 구조는 분산되지 않은 구조를 의미한다.

5. 전력을 고려한 연산의 스케줄링과 할당 방법

이 장에서는 앞서 정의한 문제를 해결하기 위한 알고리즘이 제안된다. 먼저 K=1 인 경우를 풀기 위한 알고리즘 $PSCH_1$이 제안되며, 이를 바탕으로 분산된 VLIW 구조에서 스케줄링과 할당을 하는 방법인 $PSCH_K$가 제안된다.

5.1 분산되지 않은 VLIW 구조에서의 스케줄링과 할당 방법

순간 최대 전력을 줄이기 위해서 모든 연산은 특정 시간에 집중되지 않도록 분산되어야 한다. FDS(Force Directed Scheduling)[17]은 같은 종류의 자원에 대한 요구가 특정 시간에 집중되지 않도록 이를 분산시키는 가장 효과적인 방법 중의 하나로 알려져 있다. 전력 소모를 줄이기 위해서는 자원의 종류와는 상관없이 각 클락 단계에서 소모되는 최대 전력을 되도록 균등하게 분산시켜야 하므로 FDS의 비용 함수에서 DG의 값을 다음과 같이 수정한다면 FDS는 우리가 원하는 균등 분산 작업을 수행하게 된다. (여기에서 O는 전체 연산의 집합, $prob(op, i, j)$는 연산 op가 클락 i에서 j번째 단계의 수행을 하고 있을 확률이다.) 이렇게 수정된 알고리즘이 $PSCH_1$이다.

$$DG(i) = \sum_{op\in O} \sum_{j=1}^{d_{max}} p(op, j) * prob(op, i, j)$$

$PSCH_1$은 FDS와 같이 기본 블락(basic block) 내에서의 스케줄링만을 담당하므로 블락 내 스케줄링의 한계를 넘는 최적화는 전처리 혹은 후처리 작업으로 수행된다. 루프(loop) 내의 명령어 수준 병렬성을 최대화하는 루프 변환 방법인 소프트웨어 파이프라이닝(software pipelining)의 경우 [5]에서 제안된 방법을 전처리 과정에 포함함으로써 수행된다.

5.2 분산된 VLIW 구조에서의 스케줄링과 할당 방법

분산된 VLIW 구조에서는 각 연산을 어느 클러스터에 할당하는가를 정하는 추가적인 문제가 생기게 된다. 이 때 한 가지 유의할 점은 가능한 클러스터간 데이터 전송 연산의 횟수가 최소가 되도록 각 연산을 할당할 필요가 있다는 것이다. 클러스터간 데이터 전송 연산은 그 자체로서 이미 전력 소모를 유발하기 때문에 이러한 연산이 많을수록 시스템 전체의 최대 전력 소모는 늘어날 확률이 높아지며, 이 연산 역시 수행하는데 시간을 요구하므로 전체 시스템의 성능이 저하되거나 스케줄링에 있어서 제약이 가해지게 된다.

우리의 알고리즘은 이러한 점을 충분히 고려하여 다음의 두 단계를 통해 주어진 문제를 해결하게끔 설계되었다.

단계 1. 초기 해의 생성 : 초기 해는 우선 모든 연산을 K개의 그룹으로 나눈 후, 여기에 $PSCH_1$을 적용시켜서 구할 수 있다. 서로 다른 두 클러스터간의 연산에 종속관계가 있는 경우에는 $PSCH_1$을 적용하기에 앞서 적절한 클러스터간 데이터 전송 연산을 데이터 흐름 그래프에 추가해 주어야 한다. 연산을 K개의 그룹으로 나누는 방법으로서, 연결된 다른 연산의 개수가 많은 연산 순으로, 즉 가장 종속적인 연산 순으로 할당하는 시점에서의 클러스터간 데이터 전송 연산의 개수가 최소화되는 그룹에 할당하는 직관적인 방법을 사용한다.

단계 2. 반복적 개선 : 이 단계에서는 같은 연산 유

닛에 할당되어 있지 않은 서로 다른 두 연산을 선택하여 이 둘의 할당을 맞바꾸는 과정을 통해 초기의 해를 점진적으로 개선시켜 나아간다. 선택된 두 연산이 수행되는 각 연산 유닛은 동일한 클러스터 혹은 서로 다른 클러스터에 존재하는데, 후자의 경우 연산의 종속성에 따라 적절한 클러스터간 데이터 전송 연산이 추가되어야 한다. 맞바꿈이 성공적으로 이루어지기 위해서는 각 연산이 상대방의 연산 유닛에서 수행이 가능해야 하며, 이 새로운 할당에 대하여 유효한 스케줄이 존재해야 한다. 유효한 스케줄이 존재하는지를 검사하는 것과 더불어 새로운 할당에 대한 새로운 저전력의 스케줄을 얻기 위하여 스케줄링 과정을 다시 한 번 수행해야 할 필요가 있는데, 이 때 전체 연산에 대하여 매번 새로이 $PSCH_1$을 적용하는 것은 적지 않은 시간을 요하므로 우리는 다음의 방법을 사용하여 시간을 줄이도록 한다.

L 단계 재스케줄링: 데이터 흐름 그래프의 어느 한 노드(node)에 대한 스케줄을 변경한 경우, 그 영향력은 조정된 노드로부터 멀어질수록 줄어들게 된다. 이러한 사실을 바탕으로 우리는 재스케줄링할 두 연산과 이 두 연산을 기준으로 위, 아래 L 층의 연산들에 대해서만 $PSCH_1$을 적용한다. (여기에서 L은 사용자가 정의하는 상수이며, 우리의 실험에서는 L=3으로 충분히 좋은 결과를 얻을 수 있었다.)

앞서 설명한 맞바꿈의 방식은 각 연산 유닛에 할당된 연산의 개수를 일정하게 유지시키므로, 예를 들어 어느 한 클러스터에서 다른 클러스터로의 일방적인 연산의 이동과 같은 경우는 일어나지 않는다는 단점을 가지고 있다. 이를 극복하기 위하여 우리는 각 연산 유닛에는 항상 더미(dummy) 연산이 할당되어 있다고 가정한다. 이 더미 연산과의 맞바꿈이 일어나는 경우 실제로는 연산의 한방향 이동이 일어나게 된다.

전체적으로 이 단계에서 우리의 알고리즘이 수행하는 작업은 다음과 같다 : 앞서 설명한 맞바꿈을 통해 얻을 수 있는 최대 전력 소모의 감소량이 가장 큰 (감소하는 경우가 없다면, 증가량이 가장 작은) 두 연산을 선택하여 그 할당을 맞바꾸고, 한 번 할당이 바뀐 두 연산은 그 위치가 고정된다. 이러한 과정을 더 이상 맞바꿀 두 연산을 고를 수 없을 때까지 진행시킨다. 그리하여 이 과정을 통해 처음의 해에 비하여 더 나은 해를 발견했을 경우는, 이 해를 초기 해로 설정한 후 똑같은 과정을 다시 한 번 수행하고, 그렇지 않은 경우는 이를 종료하며 여태까지 발견된 최적의 해를 최종 해로 보고한다. (그림 4)

$PSCH_K$: 분산된 VLIW 구조에서 최대 전력 최소화 알고리즘

```
01: 연산들의 집합을 K개의 그룹으로 분할한다;
02: Scurrent = PSCH1을 사용하여 얻은 초기 해;
03: COSTmin = Scurrent의 최대 전력;
04: Sbest = Scurrent;
05: while ( Sbest가 변경되었다 )
06:     /* 현재의 스케줄 Scurrent에 대해 */
07:     while ( 고정되지 않은 두 연산이 있다 ) {
08:         for ( 고정되지 않은 두 연산 ni, nj에 대해) {
09:             ni와 nj의 할당을 맞바꾼다;
10:             if ( 유효한 스케줄이 존재한다 ) then
11:                 ΔC(ni, nj) = 최대전력 감소량;
12:             else
13:                 ΔC(ni, nj) = -∞;
14:             ni와 nj의 할당을 맞바꾼다; /* 원위치로 */
15:         }                              /* 되돌림   */
16:         ΔC(ni、nj)가 최대인 ni와 nj를 선택한다;
17:         if ( ΔC(ni, nj) == -∞ ) then
18:             Sbest를 최종 해로 보고하고 종료한다;
19:         ni와 nj의 할당을 맞바꾼다;
20:         Scurrent를 이에 맞게 변경한다;
21:         COSTcurrent = Scurrent의 최대 전력;
22:         if ( COSTcurrent < COSTmin ) then
23:             COSTmin = COSTcurrent;
24:             Sbest = Scurrent;
25:     }
26:     모든 연산들을 풀어준다;
27:     Scurrent = Sbest;
28: }
29: Sbest를 최종 해로 보고한다;
```

그림 4 제안된 알고리즘

6. 실험 결과

제안된 알고리즘은 C++로 구현하였으며, 실험은 펜티엄4 1.5GHz 시스템에서 수행하였다. 우리의 알고리즘을 검증하기 위해서 몇 가지 잘 알려진 전형적인 설계에 대해 실험을 구성해 보았다[18,19]. 각 설계의 성능 제한 조건은 하향한계와 상향한계의 중간으로 설정하였다. 여기서 하향한계라 함은 그 이하로 수행 시간을 줄이는 경우에는 유효한 스케줄이 존재하지 않는 점을 말하며, 상향한계는 그 이상으로 수행 시간을 늘리는 경우에는 더 이상의 이득이 없는 점을 의미한다. 실험 대상 VLIW

시스템에서의 각 연산이 소모하는 전력은 표 1의 것을 사용하였다. 실험의 편의를 위하여 곱셈은 2 클락, 나눗셈은 6 클락 그리고 그 외의 모든 연산은 1 클락이 걸리며, 레지스터의 개수는 충분히 많은 것으로 가정하였다. 대상 VLIW 시스템은 2개의 load/store 유닛, 1개의 정수 ALU, 1개의 정수 MPY/DIV, 2개의 FP ALU, 2개의 FP MPY/DIV를 갖는 8-issue 모델로서 MIPS와 유사한 ISA를 가정하였다. 표 2는 분산되지 않은 VLIW 구조에서의 FDS과 $PSCH_1$을 비교한 것이다. 즉, 최대 전력을 고려하지 않은 일반적인 스케줄링 방법과 이를 고려한 우리의 스케줄링 방법을 비교한 것이다. 표의 두 번째 열은 해당 설계의 데이터 흐름 그래프에서의 노드의 개수를 나타낸다. 이 실험을 통해 $PSCH_1$이 FDS에 비해 최대 30.7%, 평균 19.3% 절감된 최대 전력을 갖는 다는 것을 확인할 수 있었다.

표 3은 분산된 VLIW 구조에서, 최적 해를 구하는 소모적인 방법과 $PSCH_K$를 비교한 것이다. 여기에서는 K개의 동일한 8-issue 클러스터를 가정하였다. 최적 해의 경우는 가능한 모든 해를 구한 후, 그 중 최대 전력이 가장 낮은 것을 선택한 것이다. 이러한 방식은 설계의 크기가 커짐에 따라 필요한 계산 시간이 지수 함수적으로 증가하므로 일반적으로 적용하기가 불가능하다. 본 실험에서 사용된 설계들은 모두 24시간 안에 최적 해를 구할 수 있는 것들로만 선정하였다. 표 3을 통해 우리는 제안된 알고리즘에 의해 구한 해가 최적 해에 비해 평균적으로 2.3%의 차이밖에 나지 않는다는 것을 확인할 수 있다. 우리의 알고리즘의 경우 수행시간은 모두 30초를 넘지 않았으나, 최적 해는 경우 최대 9시간이 걸렸다.

표 2 FDS와 $PSCH_1$과의 비교

설계	노드 수	최대 전력		감소량 (%)
		FDS	$PSCH_1$	
Polint	12	1.787	1.238	30.7
Complex	6	1.787	1.238	30.7
Diffeq	10	1.713	1.238	27.7
DHRC	25	2.951	2.476	16.1
IDCT	23	1.713	1.238	27.7
Wavelet	40	3.500	2.951	15.7
FFT	27	3.500	3.025	13.6
FIR	9	0.987	0.950	3.1
Kalman	7	1.713	1.238	27.7
Ellip	37	1.425	1.425	0.0
평균				19.3

표 3 소모적 방법과 $PSCH_K$와의 비교

설계	K	소모적 방법		$PSCH_K$		에러 (%)
		전력	시간	전력	시간	
Polint	2	1.388	4초	1.388	1초	0.0
Complex	2	2.476	1초	2.476	1초	0.0
Diffeq	2	1.787	1초	1.787	1초	0.0
DHRC	3	1.713	5분	1.777	1초	3.6
IDCT	3	2.476	7분	2.476	1초	0.0
Wavelet	4	2.476	9시간	2.476	27초	0.0
FFT	3	2.476	4분	2.951	1초	16.1
FIR	2	0.549	1초	0.549	1초	0.0
Kalman	2	1.238	1초	1.238	1초	0.0
Ellip	4	0.950	1시간	0.987	17초	3.1
평균						2.3

7. 결 론

본 논문에서 우리는 여러 개의 클러스터로 분산된 VLIW 구조에서 최대 전력을 줄이기 위한 알고리즘을 제시하였다. 제안된 알고리즘은 초기 해를 생성하는 단계와 이를 반복적으로 개선시키는 단계로 이루어져 있으며, 이 때 각 연산의 스케줄링과 클러스터 할당을 동시에 고려함으로써 효율적으로 최대 전력을 줄이게 된다. 일련의 실험을 통하여 제안된 알고리즘이 최대 전력 절감에 있어 매우 효과적이라는 사실을 확인할 수 있었다.

참 고 문 헌

[1] V. Raghunathan, S. Ravi, A. Raghunathan, G. Lakshminarayana, "Transient Power Management Through High Level Synthesis", ICCAD 2001.

[2] C. Y. Wang and K. Roy, "Maximum Power Estimation for CMOS Circuites Using Deterministic and Statistical Techniques", IEEE Trans. on VLSI Systems, 1998.

[3] Y. M. Jiang, A. Krstic, and K. T. Cheng, "Estimation of Maximum Instantaneous Current through Supply Lines for CMOS circuites", IEEE Trans. on VLSI Systems, 2000.

[4] V. S. Lapinskii, M. F. Jacome and G. A. de Veciana, "High-Quality Operation Binding for Clustered VLIW Datapaths", Design Automation Conference, 2001.

[5] C. Akturan and M. F. Jacome, "CALiBeR: A Software Pipelining Algorithm for Clustered VLIW Processors", Proc. of IEEE/ACM International Conference on Computer Aided Design, 2001.

[6] J. Sánchez and A. González, "Modulo Scheduling for a Fully-Distributed Clustered VLIW Architecture", roc. of 33th International Symposium on Microarchitecture, 2000.

[7] J. Zalamea, J. Llosa, E. Ayguade and M. Valero, "Modulo Scheduling with Integrated Register Spilling for Clustered VLIW Architectures", Proc.

of the 33rd Annual International Symposium on Microarchitecture, 2000.
[8] E. Özer, S. Banerjia and T. Conte, "Unified Assign and Schedule: A New Approach to Scheduling for Clustered Register File Microarchitectures", Proc. of the 31st Annual International Symposium on Microarchitecture, 1998.
[9] G. Desoli, "Instruction Assignment for Clustered VLIW DSP Compilers: A New Approach", Technical Report HPL-98-13, HP Laboratories, 1998
[10] M. T.-C. Lee, V. Tiwari, S. Malik and M. Fujita, "Power Analysis and Minimization Techniques for Embedded DSP Software", IEEE Trans. on VLSI Systems, 1997.
[11] L. Benini, D. Bruni, M. Chinosi, C. Silvano, V. Zaccaria and R. Zafanlon, "A Power Modeling and Estimation Framework for VLIW Based Embedded Systems", PATMOS01-IEEE 11th International Workshop on Power and Timing Modeling, Optimization and Simulation, 2001.
[12] S. Rixner, W. J. Dally, B. Khailany, P. Mattson, U. J. Kapasi and J. D. Owens, "Register Organization for Media Processing", Proc. of the 6th International Symposium on High-Performance Computer Architecture, 2000.
[13] M. Lorenz, R. Leupers and P. Marwedel, "Low-Energy DSP Code Generation Using a Genetic Algorithm", Proc. of International Conference on Computer Design, 2001.
[14] C. Lee, J. K. Lee and T.-T. Hwang, "Compiler Optimization on Instruction Scheduling for Low Power", Proc. of the 13th International Symposium on System Synthesis, 2000
[15] M. Sami, D. Sciuto, C. Silvano and V. Zaccaria, "An instruction-level energy model for embedded VLIW architectures", Trans. on Computer-Aided Design of Integrated Circuits and Systems, 2000.
[16] H. Yun and J. Kim, "Power-aware Modulo Scheduling for High-Performance VLIW Processors", International Symposium on Low Power Electronics and Design, 2001.
[17] P. G. Paulin and J. P. Knight, "Force-Directed Scheduling for the Behavioral Synthesis of ASIC's", IEEE Trans. on Computer-Aided Design, 1989.
[18] N. D. Dutt, "High-Level Synthesis Design Repositories", http://www.jcs.uci.edu/~dutt.
[19] W. H. Press, et al, "Numerical Recipes in C", Cambridge University Press, 1988.

서 재 원
한국과학기술원 전산학석사. 현재 한국과학기술원 박사과정중. 관심분야는 임베디드 시스템, 저전력시스템 설계

김 태 환
미국일리노이 주립대 전산학박사. 현재 한국과학기술원 전자전산학과 전산학전공 부교수. 관심분야는 임베디드 시스템, 시스템-온-칩 설계

정 기 석
미국 일리노이 주립대 전산학 박사. 현재 홍익대학교 컴퓨터공학과 조교수. 관심분야는 임베디드 시스템, 저전력시스템 설계