배정 가능 범위 축소에 의한 스케쥴링을 위한 연산 선택 방법

○
서 영 진, 유 희 진, 박 도 순*
*홍익대학교 컴퓨터 공학과

A method of operation selection in scheduling with decreasing mobilities

○
Young-Jin Seo, Hee-Jin Yoo, Do-Soon Park*
*Dept. of Computer Engineering, Hong-ik University

요 약

자동화 설계의 합성 과정 중에서 스케쥴링은 설계되는 하드웨어의 면적과 실행 시간을 결정하는 중요한 단계이다

본 논문에서는 논문[8]의 방법으로 모든 연산의 배정 가능 범위를 축소하였으나 스케쥴링이 종료되지 않을 때 현재의 스케쥴링 상황을 매개변수로 표현하여 임의의 연산 배정 범위를 축소하기 위한 선택 함수를 제안한다 제안한 방법에서는 배정 가능 범위가 가장 큰 연산의 ASAP 또는 ALAP중의 하나를 항상 선택하는데, 그러한 연산이 2개 이상인 경우에는 임의의 연산의 ASAP 또는 ALAP을 선택하여 축소하는 경우에 모든 연산의 배정 가능 범위의 변화량, 임의의 연산을 ASAP 또는 ALAP에 고정하였을 때 자원 제약과 그 연산의 종속성에 의한 다른 연산들의 이동 변화량, 그리고 각 파티션에 연산들의 배정을 균등하게 하는 정보를 사용하여 연산의 ASAP 또는 ALAP중에 하나를 선택한다 이 알고리즘의 성능 평가는 5차 엘립틱 웨이브 필터를 벤치마크로 사용하였으며, 실험 결과는 모든 엘립틱 웨이브 필터에 대해 최적이었다.

1. 서론

상위 수준 합성은 알고리즘 수준의 하드웨어 동작 기술을 입력으로 하여 중간 코드인 CDFG(Control/Data Flow Graph)로 번역한 후 스케쥴링과 모듈할당 과정을 수행하여 레지스터 전송 수준의 데이터패스와 콘트롤러 설계표현을 생성하는 과정이다 이러한 합성 과정 중 스케쥴링은 설계되는 하드웨어의 면적과 실행 시간을 결정하는 중요한 단계이다

상위 수준 합성에서의 파이프라인 스케쥴링은 DFG내의 연산들 중에서 동시에 실행될 수 있는 제어단계 집합인 파티션을 고려하여야 한다

파이프라인 스케쥴링[1]은 NP-complete 문제이며[5], 최적의 해를 찾기 위해 리스트 스케쥴링[2], 확률기반 스케쥴링[3][4], 변환 기반 스케쥴링 등 많은 휴리스틱 알고리즘[6]이 개발되었다 논문[8]에서는 자원 제약하에서 파이프라인 데이터패스를 설계할 때에, 각 연산을 그 연산의 배정 가능한 제어단계들 중에서 첫번째 제어단계 그리고 마지막 제어단계에 각각 배정하는 경우에 해에 도달할 수 있는지를 판단하여, 해에 도달할 수 없는 경우, 즉 스케쥴링이 불가능하게 되는 경우에 그 제어단계를 제거하여 배정 가능 범위를 축소하므로써 최종 해에 도달할 수 있도록 하였다 그러한 방법으로 모든 연산의 배정 가능 범위를 축소하였으나 스케쥴링이 종료되지 않은 경우에는 배정 가능 범위가 축소될 수 있는 연산들 중에서 임의로 하나의 연산과 그 연산의 ASAP을 선택하였다

본 논문에서는 현재의 스케쥴링 상황을 정량화하여 이를 기반으로 최적의 스케쥴링 결과를 얻을 수 있는 연산과 그 연산의 ASAP 또는 ALAP 중의 하나를 선택하여 축소하는 방법을 제안한다.

2절에서는 매개변수에 의한 임의의 연산 선택 알고리즘을 제시하고, 3절에서는 알고리즘에 대한 성능 평가를, 4절에서는 결론을 기술하였다

2. 임의의 연산 선택 알고리즘

배정 가능 범위 축소에 의한 데이터 패스 합성 알고리즘은 스케쥴링 알고리즘과 연산의 자원 배정 가능 판단 알고리즘으로 구성된다. 자원 배정 가능 판단 알고리즘은 주어진 제어 단계 안에 자원충돌을 해결하며 스케쥴링이 가능한지를 판단하는 알고리즘이다 스케쥴링 과정에서는 임의의 연산을 그 연산의 ASAP 제어단계 또는 ALAP 제어단계에 임시로 배정한 후 자원 배정 가능을 판단한다. 만일 자원충돌이 존재한다면 그 제어단계에 연산을 배정할 경우에는 스케쥴링이 불가능하게 되므로 그 제어 단계를 삭제하여 배정 가능 범위

축소를 한다 스케쥴링이 종료되지 않았음에도 불구하고 모든 연산의 ASAP/ALAP 제어단계들 중에서 제거할 제어단계를 찾지 못하는 경우에는 최적의 스케쥴링을 위하여 배정 가능 범위를 제거할 연산을 찾는 휴리스틱 알고리즘을 적용한다. 논문[8]에서는 배정 가능 범위가 제일 큰 연산의 ASAP을 선택하여 축소시키는 방법을 사용하였다 휴리스틱 알고리즘을 적용하는 경우에는 휴리스틱에 의한 알고리즘 실행시간이 커진다

본 논문은 스케쥴링 알고리즘의 실행시간이 증가하지 않도록 현재의 스케쥴 상황을 매개변수로 정의하고, 그 매개변수에 의하여 임의의 연산의 ASAP 또는 ALAP중의 하나를 선택하고 선택된 연산의 제어단계를 축소하여 스케쥴링이 계속되도록 하는 연산 선택방법을 제시한다

2.1 매개변수

배정 가능 범위 축소 단계 진행 중에 현재의 스케쥴 상황을 매개변수로 표현하고, 선택 함수는 매개변수의 계수에 의해 정의된다 현재의 스케쥴 상황은 연산의 종속성에 대한 정보 임의의 연산의 ASAP 또는 ALAP을 선택하여 제거할 때 모든 연산의 배정 가능 범위의 변화량, 임의의 연산을 ASAP 또는 ALAP에 고정하였을 때 자원 제약과 그 연산의 종속성에 의한 다른 연산들의 이동 변화량, 그리고 각 파티션에 연산들의 배정을 균등하게 하는 정보로 표현한다.

연산의 종속성에 대한 정보는 그 연산의 ALAP을 제거할 경우에 선행연산의 수 또는 ASAP을 제거할 경우에는 후행연산의 수를 의미한다. 모든 연산의 배정 가능 범위의 변화량은 임의의 연산의 배정 가능 범위 축소와 그 연산의 종속성에 의하여 선행연산 또는 후행연산의 배정 가능 범위가 변하는 것을 의미한다 그리고 임의의 연산을 ASAP 또는 ALAP에 고정하였을 때 자원 제약과 그 연산의 종속성에 의한 다른 연산들의 이동 변화량은 자원 제약을 해결하기 위해 이동해야 하는 모든 연산의 이동횟수로 정의한다

2.2 선택 함수

선택 함수는 연산의 배정 범위를 축소하기 위하여 특정 연산과 그 연산의 ASAP 또는 ALAP중의 하나를 선택하기 위해 정의된다 선택 함수에 영향을 미치는 매개변수의 성격은 다음과 같이 정의한다

연산들 중에서 항상 배정 가능 범위가 가장 큰 연산의 ASAP 또는 ALAP이 선택되도록 하는데, 그 연산이 2개 이상인 경우에는 모든 연산의 배정 가능 범위 변화율이 작을수록, 모든 연산들의 이동 변화율이 클수록, 파티션 내에 각 연산들이 균등하게 배정될수 있는 연산이 선택되도록 한다. 이는 연산의 배정 가능 범위 변화율은 작을 수록 전체 스케쥴에 대한 영향이 적게 미치며, 모든 연산의 이동 횟수는 많을수록 자원 제약에 의한 충돌 가능성이 높기 때문이다.

이러한 매개변수들은 선택 함수에서 정규화하여 사용하는데, 모든 연산의 배정 가능 범위 변화량은 전혀 변하지 않는 경우부터 전체 배정 가능 범위만큼 변하는 경우가 존재할 수 있으므로 정규화를 위해 전체 배정 가능 범위(*TotalMobility*)로 나누며, 이동 변화량도 전혀 변화가 없는 경우부터 전체 배정가능 범위만큼 변할 수 있으므로 전체 배정 가능 범위로 나누어 정규화한다

각 파티션에 연산들의 배정을 균등하게 하는 정보는 다음의 식에 의하여 계산하며 0과 1사이의 값을 가진다

$$\frac{TotalOpsInPar - NumOfSelectedOpsInPar}{TotalOpsInPar \times NumberOfResourcesInThisType}$$

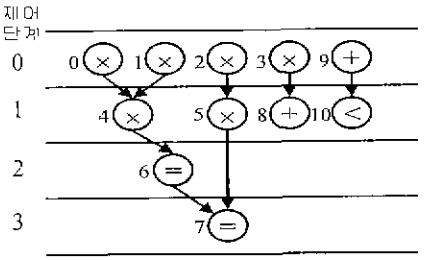
TotalOpsInPar 파티션 내의 전체 연산 수
NumOfSelectedOpsInPar 파티션 내에서 현재까지 선택된 연산의 수
NumberOfResourcesInThisType 관련 연산의 자원 재익 수

선택 함수의 의사코드는 표1과 같다. 모든 연산들에 대해 그 연산이 가장 큰 배정 가능 범위를 갖는 연산이라면, 그 연산의 ASAP, ALAP에 대해 매개변수를 계산한다. 매개변수 *MobilityChangeRate*는 모든 연산의 배정 가능 범위의 변화율을 나타내며 변화율이 작을수록 선택되어야 하므로 1에서 감하여 사용한다 매개변수 *MovingRate*는 모든 연산들의 이동 변화량을 나타낸다 매개변수 *PartitionDistributionRate*는 각 파티션에 연산들의 배정을 균등하게 하는 정보를 나타낸다. 변수 *DecisionValue*는 매개변수에 계수를 곱하여 계산되고 가장 큰 *DecisionValue* 값을 가지는 연산과 그 연산의 ASAP, ALAP 중의 하나가 선택된다.

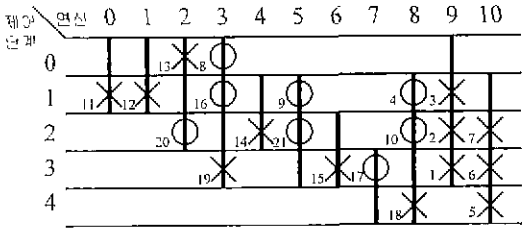
표1 선택 함수

```
Begin
TempValue ← -∞
For op Do
  If Mobility(op) = MaxMobility Then
    For Asap-Alap Do
      MobilityChangeRate ←
        1 - (GetMobilityChange(op,Asap-Alap)/TotalMobility),
      MovingRate ←
          MovingTimesOfOp(op,Asap-Alap)/TotalMobility,
      PartitionDistributionRate ←
          GetOpsInPartition(partition,type)
            /GetTotalOpsInPartition(partition,type)
            /NumberOfResourcesInThisType(type)
      DecisionValue ← α × MobilityChangeRate
                    + β × MovingRate
                    + γ × PartitionDistributionRate,
      If TempValue < DecisionValue Then
        SelectedOp ← op,
        SelectedAsapAlap ← Asap-Alap,
      End If
    End For
  End If
End For
Return SelectedOp, SelectedAsapAlap,
End.
```

<그림1>은 배정 가능 범위 축소 및 선택함수에 의한 스케줄링 예를 나타낸다. <그림1>의 (a)는 입력인 DFG를 나타낸 것이며 (b)는 배정 가능 범위의 축소 순서와 임의의 연산 선택 순서를 나타낸 것이다. ×는 배정 가능 범위 축소 알고리즘에 의해 축소된 것을 의미하며, ○는 선택 함수에 의해 선택되어 축소된 것을 의미한다. 숫자는 배정 가능 범위의 축소 순서를 나타낸다. 예를 들어 3번째까지는 배정 가능 범위 축소 알고리즘에 의해 제거되고 더 이상 축소할 연산이 없는 상황이어서, 연산의 배정 범위가 가장 큰 연산3과 연산 8의 ASAP과 ALAP에 대해 선택함수를 적용하여 연산 8의 ASAP이 4번째로 축소되어 스케줄링이 계속된다.



(a) DFG



(b) 알고리즘 실행 순서

<그림1>알고리즘 실행 예

3. 실험 결과

제안된 알고리즘의 성능 평가 실험으로 5차 엘립틱 웨이브 필터를 사용하였다. 선택 함수내의 매개변수들의 계수는 실험에 의해 α=0.8, β=0.6, γ=0.2로 설정하였다. 표2는 실험결과이며 모든 경우에 대해 최적의 결과를 나타내고 있다.

표2 엘립틱 웨이브 필터의 스케줄링 결과(곱셈기의 delay 2, 덧셈기 delay 1인 경우)

덧셈기 수		26	13	9	7	6	5	4	4	3	2	1
곱셈기 수		8	4	3	2	2	2	1	1	1	1	1
레이턴시		1	2	3	4	5	6	7	8	9	13	26
제어 단계 수	ILP[7]	17	17	18	19	19	17	18	20	22	23	33
	Sehwa[7]	17	17	18	19	20	21	20	22	25	24	33
	본논문	17	17	18	19	19	17	18	20	22	23	33

4. 결론

본 논문에서는 논문[8]의 방법을 사용하여 모든 연산의 배정 가능 범위를 축소하여 스케줄링을 할때에 스케줄링이 종료되지 않은 상황에서 축소할 배정 가능 범위가 존재하는 경우에 현재의 스케줄링 상황을 매개변수로 표현하고 이를 임의의 연산 배정 범위를 축소하기 위한 선택 함수에 사용하였다. 선택함수의 매개변수는 배정 가능 범위 축소단계와 자원 할당 가능 판단 단계를 실행할 때에 계산되므로 매개변수 계산에 의한 별도의 부가적인 시간 복잡도가 필요 없다. 그리고 적절한 계수와 함께 선택 함수에 의한 연산의 선택은 항상 최적의 스케줄링 결과를 얻을 수 있었다.

참고문헌

[1] P M Kogge, "The Architecture of Pipelined Computers", McGraw-Hill, 1981.

[2] Park, N and Parker, AC, "Sehwa: A software package for synthesis of pipelines from behavioral specification", IEEE Transaction Computer Aided Design, vol 7 March 1988

[3] Pierre G Paulin, John P Knight, "Force-Directed Scheduling for the Behavioral synthesis of ASIC's", IEEE Trans Computer Aided Design, vol 8, 1989.

[4] Pierre G. Paulin, John P. Knight, "Scheduling and Binding Algorithm for High-Level Synthesis", 26th ACM/IEEE DAC, pp 1-6, 1989

[5] Daniel D Gajski, N Dutt, "High-Level synthesis: Introduction to chip and system design", KAP, 1992

[6] Yuan-Long Jeang, Yu-chin Hsu, "Pipeline scheduling techniques in High-Level Synthesis", 30th ACM/IEEE DAC, pp 396-403, 1993

[7] Cheng-Tsung Hwang, Yu-Chin Hsu, Youn-Long Lin, "PLS A scheduler for pipeline synthesis", proc IEEE trands on CAD/ICAS, vol 12, No 9, pp 1279-1286, 1993.

[8] 윤훈병, 박도순, "배정 가능 범위 축소에 의한 자원 제약 스케줄링 알고리즘", 96 추계 학술 발표 논문집, 한국정보처리학회, 1996