자원 제약하에서 가변 데이터 입력의 파이프라인 데이터 패스 합성을 위한 스케줄링 알고리즘

오 주 영° 박 도 순
홍익대학교 컴퓨터 공학과
{jyoh, dspark}@cs.hongik.ac.kr

A Scheduling algorithm for pipelined data path synthesis with variable initiation intervals under resource constraints

Ju-Young Oh° Do-Soon Park
* Dept. of Computer Engineering, Hongik University

요 약

상위 수준 합성 과정에서 스케줄링은 하드웨어 동작을 표현한 연산들이 주어진 제약 조건을 만족하며 최적의 제어단계에 배정되도록 하는 과정이며 스케줄 결과는 목적 하드웨어의 면적과 실행속도에 많은 영향을 준다. 파이프라인은 순차적인 데이터 입력을 중첩 수행하여 실행 속도와 자원 이용률을 동시에 증가시키는 방법이다.

상위 수준에서 파이프라인 데이터 패스를 합성하기 위한 기존의 스케줄링 알고리즘들은 고정된 데이터 입력 간격열을 기반으로 제안된 것이 대부분이며, 가변 데이터 입력 간격을 지원하는 스케줄링 알고리즘으로는 시간 제약하의 자원최소화 알고리즘[5]이 제안되었다. 본 논문에서는 가변데이터 입력 간격을 지원하는 자원 제약하의 실행시간 최소화 알고리즘을 제안한다. 이를 위해 연산의 스테이지 인덱스가 초기에 고정되는 시간제약하의 스케줄링 알고리즘[5]을 응용하여 자원제약하의 스케줄 진행과정에서 증가되는 제어단계에 따라 스테이지 인덱스가 변경될 수 있도록 하고 점진적인 모빌리티 축소에 의해 스케줄한다. 제안된 스케줄링 알고리즘의 실험 결과는 다양한 자원제약과 입력 간격렬에 대하여 제약조건을 만족하는 효과적인 스케줄 결과를 유도한다.

1. 서론

상위수준 합성에서의 스케줄링은 제약 조건을 만족하는 범위 내에서 주어진 목적 함수를 최대로 하는 제어순열을 탐색하는 것이다. 복합적인 제약조건을 만족하는 파이프라인 시스템 설계를 위한 스케줄링의 탐색공간은 NP-Hard이므로 다양한 방법에 의한 휴리스틱 알고리즘들이 개발되었다. Sehwa[1]는 자원 제약 조건에서 실행 시간을 최소화하는 리스트 스케줄링 알고리즘으로 우선순위 함수인 긴급도를 사용한다. 긴급도에 따라서 연산의 리스트를 작성하고 제어단계에 순서대로 연산을 배정하며, 리스트내의 연산을 제어단계에 배정하는 과정에서 체이닝 연산들의 지연 시간이 클럭 사이클을 초과하거나 자원 제약을 초과하면 연산을 다음 제어단계로 연기하며 스케줄 한다. Hwang[2]은 각 파티션에 연산을 균등히 분배하여 자원공유를 최대화함으로써 하드웨어 비용을 최소화 할 수 있는 방법을 제안하였다. 스케줄링은 연산들에 대한 자유도를 우선순위 함수로 하여 리스트를 작성한 후 연산의 밀집도가 최소인 파티션에서 자유도가 작은 연산을 제어단계에 배정하고, 리스트에서 그 연산을 제거하는 방법으로 한번에 하나의 연산을 스케줄링 한다. PLS[3]는 자원 제약 조건하에서 루프 스케줄을 위하여 우선순위 함수인 긴급도에 의하여 전진 스케줄링과 후진 스케줄링을 반복 사용하였으며 이미 배정된 연산을 재 배정하는 방법으로 루프의 반환시간을 줄였다. PaCh99[4]는 임계경로의 지연 시간을 우선 순위 함수로 사용하는 리스트 기반 스케줄링이며 자원의 이용률을 개선함으로써 합성될 시스템의 성능을 최대화하였다. 이러한 기존의 대부분의 알고리즘들은 고정 입력 간격을 지원하는 정적 파이프라인에 국한되었다. Hwang[5]은 가변 데이터 입력 간격열에 대해 시간 제약 조건하에서 동적 파이프라인 스케줄링을 위한 알고리즘을 제안하였다. 스케줄링은 특정 연산을 한 파티션에 스케줄함으로써 전체 연산이 각 파티션에 얼마나 균등하게 분포하게 되는지의 정도값인 엔트로피를 이용한다. 엔트로피는 자원 이용률을 높이고 시스템 설계 비용을 최소화할 때 크게된다. 따라서, 파티션에 노드들이 가장 균등하게 분포될 수 있도록 하는 엔트로피가 최대인 노드를 우선 선택하여 스케줄한다.

본 논문에서는 가변 입력을 지원하는 파이프라인 처리기 합성을 위한 자원제약하의 스케줄링 알고리즘을 제안한다. 이를 위해 가변 입력을 지원하는 시간제약 알고리즘에서 고정되었던 스테이지 인덱스를 제어단계 증가에 따라 재구성 할 수 있도록 하였다. 또한, 스테이지 인덱스의 변화에 따라서 연산의 스케줄 우선 순위 값이 달라질 수 있으므로 우선순위 함수를 사용하지 않고 배정가능 범위 축소[6] 방법에 의해 스케줄한다. 2장에서는 가변 입력을 지원하는 파이프라인 처리기 합성을 위한 스케줄링에서 필요한 제반 변수를, 3장에서 제안 알고리즘을, 4장에서 알고리즘에 대한 성능 평가를, 5장에서는 결론을 각각 기술하였다.

2. 파이프라인 스케줄링

파이프라인 방식에서 시스템 성능을 결정하는 가장 중요한 요소는 데이터 입력간격이며, 데이터 입력은 고정간격 또는 가변간격으로 파이프라인에 공급될 수 있다. 가변 입력에 의한 파이프라인 처리 방식은 자원 이용률과 합성 시스템의 성능면에서 고정 입력이나 지연연산 삽입에 의한 처리방식보다 우수하다[5].

2.1 가변 입력의 파이프라인

가변 데이터 입력에 의한 파이프라인 처리기 합성에서 스케줄링을 위해 정의된[5] 식(1)은 파이프라인으로 유입되는 입력 데이터의 시간 간격을, 그리고 식(2)는 시간 간격에 의해 부여되는 시간순열을 각각 의미한다.

$$IS=(I_0, I_1, \ldots, I_{L-1}) \quad \ldots\ldots(1)$$

$$IT=(t_0, t_1, \ldots, t_L), t_0=\sum_{j=0}^{i-1} I_j \quad \ldots\ldots(2)$$

식(2)는 비순환 입력 간격열을 가지는 가변 데이터 입력간격 파이프라인 구조에서, 시간 t_i에서 중첩되는 스테이지 집합을 의미하며 최근에 발생한 데이터 입력을 수행하는 최소 스테이지 번호로부터 이전에

발생한 데이터 입력의 시간차를 누적함으로써 계산할 수 있다.

P_k=[*stage whose index is*

q_n, $q_n = q_{n-1} + I_{(i-n)\%L}$, $q_0 = q_{\min,k}$, $0 \le q_n \le N_{stage} - 1$]....(3)

식(2)에서 입력 시간 순열의 합 t_L은 파이프라인 실행시에 동시에 활성화되는 파티션의 개수이며 파티션은 제어단계의 집합으로 구성된다. 파이프라인 파티션의 수가 결정되면, 식 (3)에 의해 각 파이프라인 파티션에 속하는 스테이지들의 인덱스를 구하고, 스테이지 별로 연산을 할당하여 스케줄링을 위한 초기 파이프라인 파티션을 구한다. 가변 입력 간격에 의한 파이프라인 파티션의 개수는 입력간격열의 합만큼 생성되고 스케줄 대상노드는 입력 DFG에 분포하는 노드수에 입력 간격열의 개수배 만큼 존재한다. 따라서, 입력그래프 이차미분 방정식에 대해 입력 간격열을 (2,4)로 하였을 경우에는 그림[1]과 같이 파티션의 수는 6이 되고 스케줄 대상 노드는 입력 DFG에 존재하는 노드수의 두배가 된다.

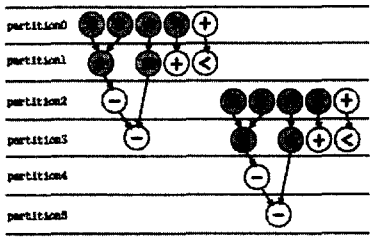


그림 [1] 입력 간격(IS=(2,4))에 의한 초기 파티션 분포

입력 간격이 IS=(2, 4)에 대한 입력 시간 순열은 IT=(0, 2, 6)로 구성되고, 파이프라인 파티션에 속하는 스테이지 인덱스는 식 (3)에 의해 $P_0=\{s_0\}$, $P_1=\{s_1\}$, $P_2=\{s_2, s_0\}$, $P_3=\{s_3, s_1\}$, $P_4=\{s_2\}$, $P_5=\{s_3\}$와 같이 구성된다.

3. 제안 알고리즘

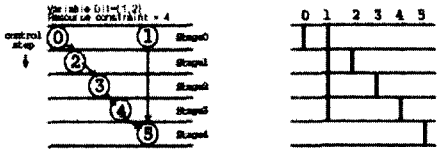
자원제약 하에서 가변 입력을 갖는 파이프라인 데이터 패스를 합성하기 위한 알고리즘은 [표 1]과 같다.

```
단계1 입력 간격열, DFG, 자원 제약 입력
단계2 파이프라인 파티션과 스테이지 인덱스 구성
단계3 초기 배정 가능한 제어단계 범위 계산
단계4
    단계4-1  연산의 ASAP 제어단계에 자원을 임시 배정
    단계4-2  if ( 자원 배정 가능 판단( ) )
             then op를 자원에 배정
              else 연산의 ASAP 제어단계 증가
               if (연산의 ASAP > 연산의 ALAP)
               then 제어단계 증가후 스테이지 인덱스 변경
                        ASAP, ALAP 스케줄링 후 단계 4로
    단계4-3  연산의 ALAP 제어단계에 자원을 임시 배정
    단계4-4  if ( 자원 배정 가능 판단( ) )
             then op를 자원에 배정
            else 연산의 ALAP 제어단계 감소
             if (연산의 ASAP > 연산의 ALAP)
              then 제어단계 수 증가하고, 스테이지 index구함
                   ASAP, ALAP 스케줄링 후 단계 4로
단계5  if( 파이프라인 파티션의 크기 < 연산의 Mobility 감소 )
       then 파티션 증가후 단계 2로
       else ASAP 또는 ALAP이 변경된 것이 존재하면 단계 4로
단계6  if(ASAP==ALAP) then 종료
          else 각 연산의 ASAP 제어단계가 결과
```

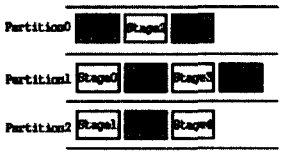
[표 1] 제안 알고리즘

알고리즘은 단계 1과 2에서 가변 데이터 입력 간격열, DFG, 자원 제약 등을 입력받아서 파티션과 파티션에 분포하는 스테이지 인덱스를 구성하며 단계 3에서 스테이지 인덱스에 대해 각 노드가 가지는 최초의 모빌리티 구간을 계산하며 이는 임계경로에 의한 초기의 배정 가능 범위를 나타낸다. 단계 4에서 스테이지 인덱스 변화에 따른 우선순위 함수의 계산 시간을 최소화하고 최적 해의 탐색 가능성을 높이기 위해 논문[6]의 스케줄링 방법을 적용한다.

단계 4-4는 연산의 스테이지 값과 스케줄 과정에서 제어단계 증가에 의해 변경된 노드의 스테이지 값이 같아서 스케줄이 불가능한 경우에는 연산이 속한 파티션 k에서 $k-1$또는 $k+1$중의 하나로 이동시킨다. 이동 후에 알고리즘을 단계 2부터 반복한다. 그림 [2] (b)는 그림 [2] (a)의 DFG가 가변 입력간격 IS = (1, 2)와 자원제약 4를 가질 때 모빌리티에 의해 초기 파이프라인 파티션에 대한 연산의 배정 범위를 나타낸다. 그림 [2] (c)는 그림 [2] (b)에 대해 연산이 가지는 스테이지 인덱스들을 나타낸다.



(a) 입력 DFG와 모빌리티



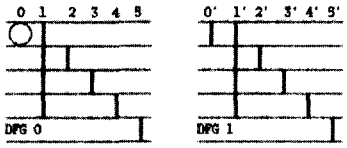
(b) 파이프라인 파티션의 초기 상태

op / Stage	0	1	2	3	4	5
begin	Stage0	Stage0	Stage1	Stage2	Stage3	Stage4
end	Stage0	Stage3	Stage1	Stage2	Stage3	Stage4

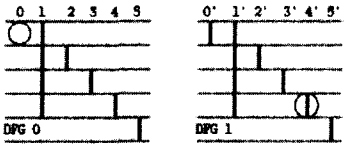
(c) 연산의 배정 가능 범위

그림 [2] 스케줄링을 위한 초기 정보

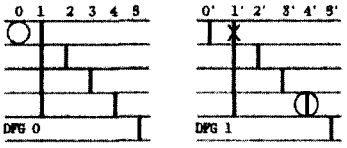
그림[3]은 그림[2]의 각 조건하에서 배정 가능 범위 판단 의한 연산의 스케줄과정을 보이고 있다. 가변 입력 파이프라인에서는 파이프라인 파티션에 분포하는 스케줄 대상 연산은, 입력 DFG내에 존재하는 연산들이 가변 입력 간격열의 개수의 배수만큼 나타날 수 있으므로, 두 개의 DFG 연산에 대한 스케줄링을 진행하는 것과 동일하다. 그림 [3] (a)에서 DFG0의 연산 0번을 파티션 0에 할당할 때는 자원충돌이 없으므로 스케줄이 진행되고 이후 DFG0의 모든 연산의 스케줄 이후 그림[3](b)와 같이 DFG1의 노드순의 스케줄 진행에서 4'을 파티션 1에 배정하는 시점에서 자원충돌이 발생한다. 그러면 그림[3](c)와 같이 모빌리티가 가장 큰 1' 연산의 ASAP에서 자원배정이 불가능하므로 ASAP 제어단계를 삭제하고, 1' 연산의 ALAP에서 자원배정이 불가능하므로 ALAP 제어단계를 삭제하게 되며, 이후 자원충돌 없이 스케줄을 종료한다. 그림 [3] (e)는 스케줄링이 종료된 후 각 연산들이 ASAP 단계에 할당되어 각 스테이지에 분포한 모습이며 모든 파티션에서 자원제약을 만족한다.



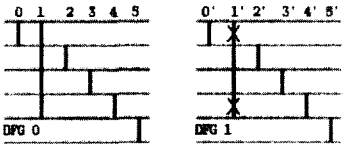
(a) DFG0 의 0번 연산을 ASAP에 배정 가능 판단



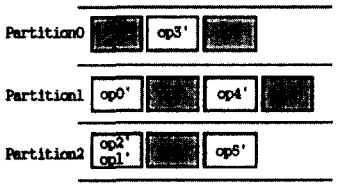
(b) DFG1의 4' 연산의 자원 충돌 발생



(c) 1'번 연산의 ASAP 제어단계 축소 후 스케줄링 가능성 판단



(d) 스케줄링 종료 후 DFG의 모빌리티 그래프



(e) 파이프라인 파티션내의 연산의 위치

그림 [3] 스케줄링 과정

4. 실험

제안 알고리즘은 펜티엄 PC환경에서 C++ 언어로 구현하였으며, 성능 평가를 위하여 벤치마크로 FIR 필터와 5차 엘립티컬 웨이브 필터를 사용하였다. 그리고 임의의 지연 시간을 갖는 연산장치를 허용하기 위해 멀티사이클링, 체이닝, 구조적 파이프라인 방식을 지원한다.

곱셈기의 수	4	3	3	2	2	1	1
덧셈기의 수	8	6	5	4	3	2	1
최소 레이턴시	2	3	3	4	5	8	15
고정 입력 간격에 의한 최적 결과	6	6	6	6	6	10	15
가변 레이턴시	1,3	2,4	2,4	3,5	4,6	7,9	14,16
제안 알고리즘	6	6	6	6	6	10	15

[표 2] FIR 필터의 가변 입력 간격에 의한 스케줄링

곱셈기의 수	4	3	2			1	
덧셈기의 수	13	9	7			4	
최소 레이턴시	2	3	4			8	
고정 입력 간격에 의한 최적 결과	17	18	19			20	
가변 레이턴시	1,3	2,4	3,5			7,9	
제안 알고리즘	18	18	20			21	

[표 3] 엘립틱 웨이브 필터의 가변 입력 간격에 의한 스케줄링

[표 2][표 3]은 알고리즘 실험을 위한 FIR 필터와 엘립틱 웨이브 필터에 대한 다양한 가변 입력 간격에 의한 파이프라인 스케줄링 결과이다. [표 2][표 3]에서 채색된 각각의 열은 파티션이 확장되어 합성에 대한 추가 비용이 발생하지만, 부합하는 시스템 성능 개선이 가능하게 됨을 보인다. 또한, 평균 레이턴시 값이 작아지게 되면 계산된 스테이지 인덱스 범위에서의 스케줄이 어렵게 된다. 제안 알고리즘은 가변 입력 간격열의 평균값들이 고정 입력간격에 의한 최소 레이턴시보다 작은 경우에도 [표 3]의 채색된 각각의 열과 같이 효과적인 스케줄 결과를 유도한다.

5. 결론

본 논문에서는 가변 입력간격을 지원하는 파이프라인 데이터 패스를 합성하기 위해 자원 제약 조건하의 스케줄링 알고리즘을 구현하였다. 시간 제약하의 자원최소화 알고리즘[5]을 응용하여 자원제약하의 스케줄링 알고리즘의 진행과정에서 증가할 수 있는 제어단계의 크기에 따라 스테이지 인덱스가 변경될 수 있도록 하였고 배정 가능 범위 축소 알고리즘에 의해 스케줄 하였다. 향후 연구과제는 파이프라인 성능의 최대 관건이 되는 효과적인 레이턴시 간격 열의 탐색과 다양한 제약 사항을 만족하는 저 복잡도의 알고리즘 구현, 구현된 알고리즘 검증을 위한 다양한 벤치마크에 의한 실험 등이 필요로 되어진다.

참고문헌

[1] N. Park and A. C. Parker, "Sehwa: A software package for synthesis of pipelines from behavioral specification," IEEE Trans. Computer-Aided Design, vol. 7, pp. 356-370, March 1988.

[2] K. Hwang, A. E. Casavant, C. T. Chang and M. A. d'Abreu, "Scheduling and hardware sharing in pipelined data path," in Proc. Int. Conf. Computer-Aided Design, pp. 24-27, 1989.

[3] C. T. Hwang, Y. C. Hsu and Y. L. Lin, "PLS: A scheduler for pipeline synthesis," IEEE Trans. on CAD/ICAS, vol. 12, no. 9, pp. 1279-1286, Sept. 1993.

[4] S. Park and K. Choi, "Performance-driven scheduling with bit-level chaining," in Proc. 36th Design Automation Conf., pp. 286-291, 1999.

[5] Hong Shin Jun, Sun Young Hwang, 'Automatic Synthesis of Pipeline Structures with Variable Data Initiation Intervals', in Proc. ACM/IEEE 31st Design Automation Conf., San Diego, California, June 1994, pp. 537-541.

[6] Heejin Yoo, Dosoon Park, "A scheduling algorithm for pipelined data path synthesis with gradual mobility reduction," in Proc. AP-ASIC'99, pp. 51-54, Aug. 1999.