Forward 알고리즘을 이용한 Chaining 연산에 관한 연구

백 남우† , 송 정영‡
† 청주기능대학 생산자동화과 , ‡ 배재대학교 컴퓨터공학과

A study on Chainnig operation using Forward algorithm

Back Nam-Woo† , Song Jeong-Young‡
† Dept. of Automated Production Technology, Chougju Polytechnic College
‡ Dept. of Computer Science, PaiChai University
E-mail: bnw13@kopo.or.kr

요 약

본 연구에서는 자원제약이 존재할 때의 스케듈링 문제인 자원제약에서 리스트 스케듈링 방법을 이용했으며, 기존의 리스트 스케듈링 기법이 모든 우선순위 함수를 산출 한 후, 이에 근거에 스케듈링을 하는 반면, Forward 스케듈링 기법은 스케듈링 과정 내에서 필요한 경우에만 연산의 우선순위 함수를 산출하여 이를 이용하는 방식을 택했다.

1. 서론

리스트 스케듈링은 연산의 우선순위(Priority)에 의해 스케듈링을 하는 기법이며, 자원을 가한 경우에는 제어구간이 확장된다. 이때 확장된 제어구간의 수를 추출하기 위해서는, 여러 단계의 스케듈링 과정에 수식을 사용해야 하는 것이 문제점으로 지적되고 있다. 본 논문에서는 주어진 자원의 수에 따라 확장되는 제어구간의 하한 값을 추출하기 위해서 수식을 사용하는 문제점을 개선하기 위해 새로운 Forward 스케듈링 기법을 제안하고자 한다. 제안된 기법의 타당성을 검토하고자 표준 벤치마크 모델인 5-order elliptic 웨이브 필터와 16-point FIR filter를 선택하고, FDS(Force- Direct Scheduling)와 (Integer Linear Programming) 스케듈링[1] 및 제안된 스케듈링을 비파이프라인 데이터패스 내에서 체이닝 방법을 수행하여 비교 평가하고자 한다.

II. 기본이론

2.1 스케듈링 이론

스케듈링은 입력 기술문에서 나타나는 연산들을 특정 제어구간에 할당시키는 과정으로서 제한된 하드웨어와 속도의 범위 내에서 수행에 필요한 제어구간의 수를 최소화하는 과정을 말한다. 이중 ASAP 스케듈링은 DFG상 모든 연산 선후관계(Precedence relation)가 유지되는 범위에서, 자원제약 조건 없이, 연산간의 선후순위로 연산을 위에서 차례로 제어구간 내에 최대로 할당시키는 기법이며, ALAP 스케듈링은 ASAP 스케듈링과 반대로 DFG상의 모든 연산을 아래에서 역으로 제어구간에 최대로 할당시키는 기법이다. 그러나, 이러한 ASAP 스케듈링과 ALAP 스케듈링은 자원 제약조건 없이 연산을 위 또는 아래에서부터 순차적으로 연산을 할당하므로 연산 처리속도는 빨라지는 점에 비해 자원은 많이 필요로 하는 점이 발생된다. 따라서, 이러한 점들을 상호의존적으로 고려할 수 있는 스케듈링이 필요하다.

2.2. 속도제약 스케듈링 및 자원제약 스케듈링

1] 자원제약 스케듈링 (Resource-Constrained Scheduling)은 제어구간의 수를 줄이는 것을 목적으로 스케듈링을 수행한다. 자원 제약시 확장되는 하한 값을 추출하기 위한 스케듈링 기법(ILP:Integer Linear Programming)이 제안되었다.

제안된 논문의 스케듈링은 1) 리스트 스케듈링 2) 수정된 ASAP 스케듈링 3) 수정된 ALAP 스케듈링 4) ILP 스케듈링으로 구성하는 4단계 스케듈링을 하였으며 수행과정을 설명하며, 리스트 스케듈링 기법에 의해 스케듈링을 수행하여 연산의 상하한 제어구간의 값을 구한 후 이로부터 얻어진 결과로 제어구간의 연산자를 전체 제어구간에 균등하게 분배시키기 위해 스케듈링을 한다. 또한 균등하게 분배된 연산자를 제약조건으로 하여 제어구간의 수를 최소화하는 스케듈링을 하고 이후 구해진 제어구간의 최하한 값과 연산의 상,하한 제어구간 값으로부터 선후관계식, 길이 관계식, 자원 관계식, 시

간 관계식의 선형정수프로그램(ILP : Integer Linear Programming)을 생성하여 주어진 자원에서의 최적화를 얻고자 하였다.

2] 동작속도제약 스케듈링 (Time-Constrained Scheduling)은 자원의 개수를 최소화하는 목적으로 스케듈링을 수행한다. 스케듈링을 할 때 연산을 선택하는 방법과 선택된 연산이 할당되는 제어구간을 결정하는 방법으로 Freedom-based스케듈링 기법과 Force-directed 스케듈링 기법을 들 수 있다. 1) Freedom-based 스케듈링 기법에서 임계경로(Critical path)의 연산을 할당시킨 후, 비 임계 경로(non-Critical path : Freedom)의 연산을 이동도에 따라 하나씩 할당시켜 스케듈링을 수행한다. 2) Force-directed 스케듈링에서 각각의 연산이 할당될 수 있는 모든 제어구간에 대한 장력(Force)을 계산한 후, 제어구간에 해당하는 장력의 값이 가장 작은 연산을 그 제어구간에 할당하여 스케듈링을 수행한다.

3] 제약요소가 속도와 자원인 스케듈링(Feasibile Schduling)으로 구분한다.

2.3. 리스트 스케듈링

DFG상의 임의의 연산이 할당될 수 있는 제어구간의 상하한(Upper-down)값을 결정하기 위한 방법으로 ASAP 스케듈링 기법과 ALAP 스케듈링 기법을 조화시킨 리스트 스케듈링 기법이 사용된다. 리스트 스케듈링은 용어가 뜻하는 바와 같이 리스트에 할당시키고자 하는 연산을 우선 순위와 임의의 기준에 따라 분류시킨 후 이에 따른 스케듈링을 행하는 기법이다. 이를 사용한 시스템으로는 BUD-DAA시스템, BSI (Behavioral Synthesis of Interfaces)시스템, SLICER시스템[8]등이 있다. 이 시스템들이 사용한 우선 순위는, BSI 시스템에서는 시간제약(Time Constraint)을 사용하였고, SLICER 시스템과 BUD-DAA 시스템에서 이동도(Mobility)를 우선 순위로 사용하였다.

2.4 체이닝 연산

연산 체이닝 기법은 연산들을 하나의 제어구간에 체인처럼 연결시켜 할당 해 주는 기법이다. 체이닝 연산은 연결된 연산이 하나의 제어구간 내에서 순차적으로 수행하며, 체이닝 된 복수 개 연산은 그 수행시간의 합이 제어구간보다 크지 않아야 한다. 또한 체이닝 연산을 하는 경우 각 할당되는 시간간격이 크면 클수록 보다 많은 연산을 하나의 제어구간 안에서 수행할 수 있다. 체이닝 연산의 장점으로는 체이닝 된 연산 간에는 데이터의 저장을 위한 레지스터가 필요 없으며 연산의 수행에 필요한 제어구간의 수가 적다는 점이다.

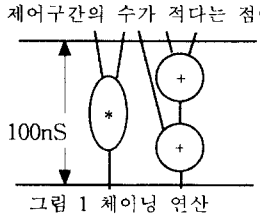

그림 1 체이닝 연산

그림 1의 연산은 체이닝에 대한 예를 나타낸 것으로, 제어구간이 100nS, 승산기 연산의 지연시간이 80nS, 가산기 연산의 지연시간이 40nS인 경우를 나타낸 것이다.

Ⅲ. FORWARD 스케듈링

3.1. 스케듈링의 알고리듬

DFG상의 모든 연산의 이론전개에 편리하도록 기호를 아래와 같이 정의하여 사용한다.

(1) Ofn : 선행 연산이 없는 연산
(2) Obn : 후행 연산이 없는 연산
(3) Of : 선행 연산이 존재하는 연산
(4) Ob : 후행 연산이 존재하는 연산
(5) Oc : 임계 경로에 존재하는 연산
(6) Ocn : 임계 경로와 무관한 연산
(7) Ocp : 연산의 출력이 Oc의 입력으로 사용되는 연산 및 이들의 모든 선행(predecessor) 연산
(8) Ocs : 연산의 입력으로 Oc의 출력을 사용하는 연산 및 이들의 모든 후행(successor)연산

논문의 Forward 스케듈링은 기존의 스케듈링과 다른 스케듈링 방식을 사용한다. 즉 기존의 모든 방식들은 먼저 모든 연산에 대한 우선 순위 함수를 계산한 후 스케듈링을 행하는데 비해, 본 방식은 주어진 자원수(N)과 제어 구간수(T)로 구성하는 T.N martrix 표를 작성하여 이 표를 이용하여 스케듈링을 수행한다.

Forward 스케듈링 알고리듬 조건

[1] DFG의 모든 연산을 Ofn, Obn, Of, Ob, Oc, Ocn, Ocp, Ocs로 구분한다. [2] 연산 Oc 와 가장 먼 연산 Ocp를 먼저 표에 할당한다. 1)주어진 조건을 연산 Ofn부터 연산 Obn까지 연산들을 위해서 아래로 순차적으로 스케듈링을 한다. 2)연산 Ofn 연산들은 첫 제어구간(1-STEP) 에 반드시 할당 되야 하고, 마지막 제어구간 전에 할당을 할 수 있다. 3)연산 Obn 연산들이 할당할 수 있는 제어구간은 마지막 제어구간에 할당이 되야 하고, 첫 제어구간 이후에 할당을 할 수 있다. 4)연산 Of와 연산 Ob는 첫 제어구간과 마지막 제어구간만 할당하지 못하고, 어느 제어구간에도 할당할 수 있다. 5) 연산 Of와 연산 Ob가 동일한 제어구간에 여산이 여러개 있을 때에는, 입력까지 제일 긴 경로길이의 연산을 기준으로 삼아서 Forward 스케듈링을 해야한다. 6) 스케듈링시 제약된 자원을 넘은 연산이 할당되어 있으면, 이 연산자는 자원의 선후관계가 변하지 않는 범위 내에서 다음 제어구간에 할당시킨다. [3] 스케듈링은 왼쪽에서 오른쪽으로 수행하여 도표에 할당시킨다. [4] 연산 Oc,연산 Ofr와 연산 Obr의 할당이 모두 완료될 때까지 [2]과[3]를 반복 수행하다.

[5] 연산 Oc, 연산 Ofr와 연산 Obr로 구성되는DFG상의 부분 연산을 DFG로 부터 제거시킨다. [6] 연산 Oc, 연산 Ofr와 연산 Obr가 적절한 제어구간에 스케듈링이 되어 있지 않으면 [2]부터[5]까지 다시 수행한다. [7] 자원제약으로 제 Ocs어구간에 적절히 연산이 할당되었으면 스케듈링을 끝

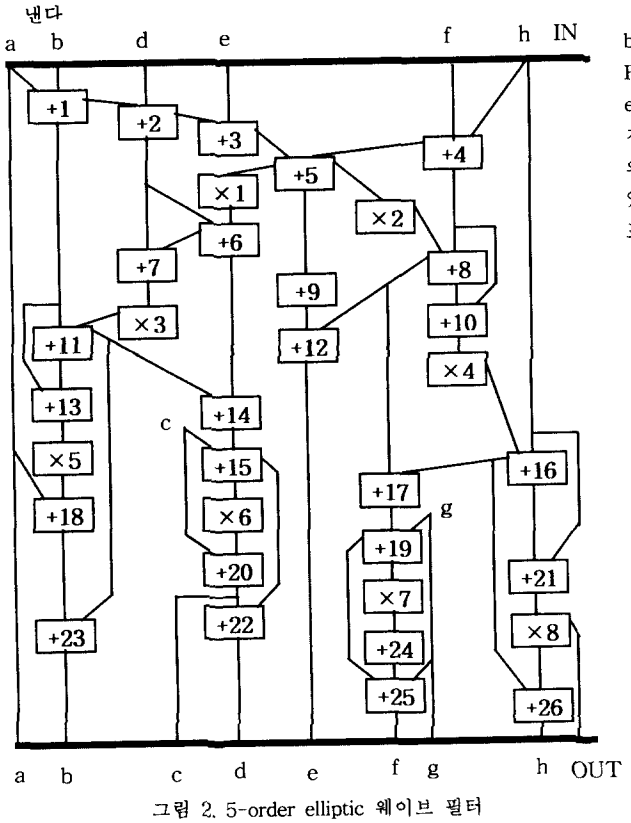

그림 2. 5-order elliptic 웨이브 필터

표1은 주어진 자원의 수(N)와 제어구간의 수(T)로 구성하는 T · N matrix 표를 작성하여, 이 표를 이용하여 제어구간의 하한 값을 추출한다. 이때 사용하는 기능 단위는 승산기(*) 1개와 가산기(+) 2개로 하고, 승산기 지연시간은 80nS, 가산기 지연시간은 40nS, 제어구간 시간 간격은 100nS 로 가정한다. 표1에서 알 수 있듯이, 1개의 승산기와 2개의 가산기로는, 수행시간이 1700nS인 5-order elliptic wave filter를 구현할 수 있다. 표2는 동일조건에서 chaining 방법으로 한 Forward 스케듈링의 결과이다.

표1 normal 연산의 Forward 스케듈링

구간수	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
A1	1 +1	2 +2	3 +3	5 +5		7 +6	9 +7	12 +9	14 +11	17 +13	20 +14	22 +15	24 +18	27 +23	30 +20	32 +22	34 +25
A2	4 +4						10 +8	13 +10	15 +12	18 +16	21 +17	23 +21	25 +19	29 +26		33 +24	
M1					6 *1	8 *2		11 *3	16 *4		19 *5		26 *8	28 *6	31 *7		

표2 체이닝 연산의 Forward 스케듈링

구간수	1	2	3	4	5	6	7	8	9	10	11
A1	+1 +2	+3 +5		+6 +7	+8 +10	+11 +3	+16 +17	+15	+18 +23	+20 +22	+24 +25
A2	+4			+9	+12	+14	+21	+19	+26		
M1			*1	*2	*3	*4	*5	*8	*6	*7	

Ⅳ. 검증 및 결과

본 연구에서 제안한 스케듈링 기법을 사용하여, 표준 benchmark 모델인 fifth-order elliptic wave filter와 16-point FIR filter에 적용시킨 결과는 표3과 표4와 같다. 5-order elliptic wave filter와 16-point FIR filter는 " + " 연산자의 지연시간은 40nS, " * " 연산자의 지연시간은 80nS, 제어구간의 시간간격은 체이닝 연산시 100nS으로 실험모델을 선택하였다.

표3. 비파이프라인 데이터 패스(5-order elliptic wave filter)

제어시구간 시간간격 : 100nS						
소요 자원		+	3	2	2	1
		*	2	2	1	1
구간수	ILP[2]	normal	14	16	17	26
		chaining	9	10	11	17
	Forward	normal	14	16	17	26
		chaining	9	10	11	17

표4. 비파이프라인 데이터 패스 16-point FIR filter

제어시구간 시간간격 : 100nS						
FORWARD	소요 자원	+	3	2	2	1
		*	2	2	1	1
	구간수	normal	9	9	10	15
		chainning	6	8	10	12

Ⅴ. 결론

본 논문에서는 상위영역 합성분야에서 가장 중요하게 취급되는 데이터패스 합성을 위한 스케듈링을 수행하는 경우 주어진 자원의 수에 따라 확장되는 제어구간의 하한값을 보다 용이하게 추출하기 위한 새로운 Forward 스케듈링 기법을 제안하였다. 제안된 스케듈링 과정은 Forward 스케듈링 기법을 표준 벤치마크 모델인 5-order elliptic wave filter와 16-point FIR filter를 선택하여 ASAP와 ALAP 스케듈링을 수행한 후, 자원의 수와 제어구간의 수를 가지고 T · N matrix 표를 작성하고 이 표를 이용하여 주어진 자원의 수에 따라 확장되는 제어구간의 하한 값을 용이하게 추출할 수 있어서 효율성을 입증할 수 있었다. 성능평가방법은 비파이프라인 데이터패스 내에서 체이닝 연산인 경우에는 100nS로 제어구간을 분할하며, *ILP* 스케듈링 기법과 비교 분석해서는 동일한 스케듈링 결과를 얻었으며, FDS 기법에서는 비파이프라인 스케듈링에서 주어진 자원의 수에 대해서 제어구간의 수가 적음을 알 수 있었다. 앞으로의 연구과제는 Forward 스케듈링 알고리즘을 프로그램화하는 것과 멀티싸이클링을 하여 파이프라인 데이터패스에서도 스케듈링을 할 수 있도록 지속적인 연구가 요구된다.

참 고 문 헌

1] E.F.Girczyc, "An ADA to standard cell hardware compiler based on graph grammers and scheduling", Proc. of ICCAD-84, pp.726-731, Oct. 1984.

2] C.T.Hwang, J.H.Lee, and Y.C.Hsu, "A formal approach to the scheduling problem in high level synthesis", IEEE Tr.CAD, Vol.10, No.4, pp.464-475, April. 1991.