자원제약하에서의 지연 스케쥴링

신 인 수† · 이 근 만††

요 약

본 연구에서는 자원의 수가 한정된 상태에서 동작 알고리즘의 수행을 마치기 위한 자원제약 스케쥴링 방법을 다루었다. 특히, 제한된 자원제약하에서 연산이 할당되는 제어스텝의 최하한값을 구하기 위한 지연 스케쥴링 방법을 제안하였다. 스케쥴링 문제에 대하여 연산의 멀티싸이클과 기능적 파이프라이닝을 고려하였으며 스케쥴링 문제에 대한 최적의 결과를 얻기 위해 선형정수계획법을 이용하였다. 5차 디지털 웨이브 필터 DFG를 실험 대상으로 하여 본 연구의 효용성을 입증하였다.

Delayed Scheduling under Resource Constrains

In-Soo Shin † · Keun-Man Yi ††

ABSTRACT

In this paper, we deal with the resource constrain scheduling to execute behavior algorithm under resource limit. Especially, we proposed a scheduling algorithm, called delayed scheduling, which finds the lower bound control step to assign operation under resource limit.

We take in account the actual scheduling problems including multicycle operation and functional pipelining. Integer Linear Programing formulations are used to the scheduling problems in order to get optimal scheduling result. Experiment was done on the DFG model of fifth-order digital wave filter, to show it's effectiveness.

1. 서 론

동작기술(behavior description)과 생성된 하드웨어의 면적, 시간 등의 조건을 입력으로 하여 제한된 제약조건들을 만족하면서 주어진 동작의 하드웨어 구조를 찾아내는 합성(synthesis) 과정중 특히, 알고리즘 수준(level)의 동작기술(behavior description)로 부터 Register Transfer 수준(RT Level)의 데이터 패스(data path)와 제어패스(control path)로 구성된 구조적 기술(structure description)로 변환하는 과정을 상위수준 합성(High Level Synthesis)이라 한다.

상위 수준 합성(High Level Synthesis) 과정은 스케쥴링(scheduling), 모듈 할당(module allocation), 모듈 바인딩(module binding)의 과정으로 세분화된다.

본 연구에서는 상위수준 합성과정중에서도 중요한 데이터 패스 합성을 위한 스케쥴링 과정을 다루었다. 스케쥴링의 목적은 제한된 제약조건들(자원, 속도, 연결구조, 메모리등)을 만족하면서 주어진 목적함수(object function)를 최소화하는 것이다. 즉, 연산의 스케쥴링은 시스템의 비용(cost)과 성능(speed)의 trade-off를 결정하기 때문에 매우 중요하다[2][3].

동작기술로부터 생성된 DFG(Data Flow Graph)상의 연산을 주어진 제약조건들을 만족하는 범위 내에

† 준 회 원:청주기능대학 전자기술학과 전임강사
†† 정 회 원:청주대학교 전자공학과 교수
논문접수:1996년 11월 13일, 심사완료:1997년 7월 30일

서 제어스텝으로 이동시키는 것으로서 ASAP/ALAP (As Soon As Possible/As Late As Possible) 스케쥴링과 자원제약 스케쥴링, 성능제약 스케쥴링으로 크게 구분된다[1][9].

제어스텝의 수에 제약이 가해진 상태에서 행해지는 성능제약 스케쥴링 방법으로는 freedom-based 스케쥴링[4]과 force directed 스케쥴링[5]이 있다.

데이터 패스(data-path) 합성시 요구되는 연산자(operator), 레지스터(register), 버스(bus) 등의 연결구조(interconnection)와 같은 자원의 수에 제약이 가해진 상태에서 행해지는 자원제약 스케쥴링 방법으로는 리스트 스케쥴링 기법이 있다. 대부분의 스케쥴링의 알고리즘은 정해진 우선순위 함수(priority function)에 따라 한 번에 하나씩 연산을 스케쥴링 하는 반복/구조(iterative/constructive) 알고리즘[4]을 사용하는데, 이러한 스케쥴링 방법은 각각의 연산을 한 번에 하나씩 할당하기 때문에 임의의 연산이 할당되고 나면 이후의 연산은 이미 스케쥴링된 연산의 결과에만 전적으로 의존하기 때문에 결과는 결코 최적화 된 것이라고 할 수 없다.

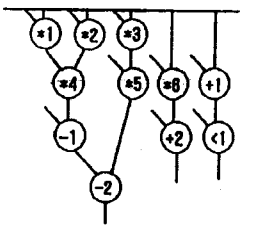
본 연구에서는 이러한 스케쥴링 문제를 해결하기 위해서 스케쥴링시 사용되는 관계식의 변수를, 조건을 만족하면 1의 값을, 만족하지 않으면 0의 정수값(integer value)을 갖는 형태로 간주하여, 다원 1차식으로 전개시켜 해를 구하는 선형 정수계획법(Integer Linear Programming)[2][9]을 이용하였다. 또한 자원제약(resource constraint) 스케쥴링에 중점을 두어 기존의 자원제약 스케쥴링에서 발생하는 문제점들을 개선한 새로운 스케쥴링(지연스케쥴링) 방법을 다룬다. 2장에서는 스케쥴링 이론을 설명하고 3장에서는 본 연구의 주제인 지연스케쥴링에 대해서 설명한다. 4장에서는 스케쥴링 결과를 나타내었으며 결론은 5장에서 다룬다.

2. 스케쥴링 및 ILP 형식

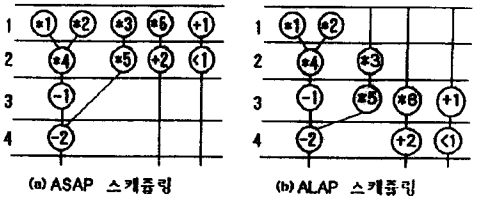
스케쥴링은 DFG상의 연산을 제어스텝에 할당하는 과정으로, 그 목적에 따라 자원제약 스케쥴링과 성능제약 스케쥴링으로 구분된다.

자원제약(resource constraint) 스케쥴링은 사용 가능한 자원(데이터 패스 합성시 사용되는 연산자, 연결자, 레지스터등)의 범위 내에서 최대의 성능(제어스텝의 수를 최적화)을 얻기 위한 스케쥴링을 의미하고 성능제약((performance constraint) 스케쥴링은 제어스텝의 수에 제한이 가해진 상태하에서 최소의 자원을 사용하기 위한 스케쥴링 방법을 의미한다. 여기서는 연산자만을 고려한 자원제약 스케쥴링 기법에 대해서만 다루었다.

스케쥴링의 첫단계는 연산의 할당범위를 결정하기 위하여 ASAP스케쥴링과 ALAP스케쥴링을 수행해야 한다. ASAP스케쥴링은 자원제약이 가해지지 않은 상태에서 연산간의 선후 관계(precedence relation)가 유지되는 범위 내에서 DFG상의 모든 연산을 가능한 제어스텝의 앞쪽으로 할당시키는 방법이고, ALAP 스케쥴링은 가능한 뒤쪽으로 할당시키는 방법이다.



(그림 1) HAL 모델
(Fig. 1) HAL model



(그림 2) ASAP/ALAP 스케쥴링 결과
(Fig. 2) ASAP/ALAP scheduling result

(그림 1)의 HAL시스템 모델[5]에 대한 ASAP/ALAP 스케쥴링 결과는 (그림 2)와 같다.

Branch-and-bound 탐색 알고리즘을 이용하여 최적의 스케쥴링 결과 해를 찾는 선형정수계획법(ILP)[2]의 수식표현에 사용된 기호에 대한 정의는 〈표 1〉과 같다.

〈표 1〉 ILP 정의
〈Table 1〉 ILP notations

n_t : 연산유형이 t인 연산의 갯수
tT : 제어스텝의 시간간격(cycle time)
d_i : 연산 O_i의 지연시간
D_i : 싸이클타임 수로 표시되는 연산 O_i의 지연시간
S_i, L_i : ASAP 스케쥴링과 ALAP 스케쥴링에 의해 결정되는 연산 O_i의 상하한 제어스텝
O_i : $O_i(k)$: 연산 O_i의 k번째 요소(element) $[O_i = \{O_i(1), O_i(2), ..., O_i(r_i)\}]$
P_{ik} : 연산 $O_i(k)$가 할당되는 제어스텝 ($P_{ik} : S_i + k - 1$)
$O_i \rightarrow O_j$: 연산 O_i와 O_j의 선후관계
n : 연산 O_i의 스케쥴링 범위($n = L_i - S_i + 1$)
X(i, k) : 연산 O_i의 k-번째 요소가 제어스텝-P_{ik}에 할당되었을 때, 그 값이 1이 되는 0-1 정수형 변수 (0-1 integer variable)

아무런 제약이 가해지지 않은 상태에서 수행하는 ASAP/ALAP스케쥴링 결과로부터 〈표 1〉에서 정의된 기호들을 이용하여 제약조건을 만족하는 스케쥴링을 위해서는 다음과 같은 관계식을 만족해야한다.

연산 O_i와 가장 가깝게 인접한 후행연산(nearest successor)를 O_j라 할 때, 연산 O_i와 O_j의 선후관계식(precedence relation)은 식(1)과 같다.

$$\sum_{k=1}^{r_i} \{P_{ik} \times X(i, k)\} - \sum_{h=1}^{r_j} \{P_{jh} \times X(j, h)\} \leq -D_i \qquad (1)$$

연산 O_i는 상한 제어스텝 Si와 하한 제어스텝 L_i의 범위에 걸쳐 오직 한번만 할당되어야 하기 때문에, 연산의 각 요소 사이에는 식(2)와 같이 범위관계식(range relation)을 만족해야한다.

$$\sum_{k=1}^{r_i} X(i, k) = 1 \ (i = 1, 2, ..., n_t) \qquad (2)$$

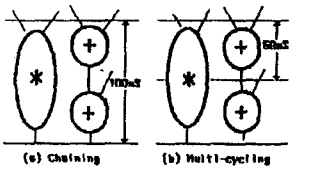
연산 O_i의 각 요소가 할당되는 제어스텝-P_{ik}는 제어스텝의 최대치(T_{max})를 초과해서는 안된다. 따라서, 후행연산이 존재치 않는 모든 연산 O_i의 요소 $O_i(k)$와 제어스텝의 최대치(T_{max}) 사이에 식(3)과 같은 시간관계식(time relation)을 만족해야한다.

$$\sum_{k=1}^{r_i} \{P_{ik} \times X(i, k)\} \leq -D_i + 1, \ (i = 1, 2, ..., n_t) \qquad (3)$$

특정 제어스텝 P에 할당될 수 있는 연산유형 t인 연산의 총 갯수는 사용 가능한 자원의 갯수 N_t를 초과해서는 안된다. 즉, 식(4)와 같은 자원관계식(resource relation)을 만족해야 한다.

$$\sum_{i=i}^{n_t} X(i, P - S_i + 1) \leq N_t (P = 1, 2, ..., T_{max} - D_i + 1) \qquad (4)$$

(그림 3(a))와 같이 복수개의 연산이 1-싸이클타임 이내에서 순차적으로 수행되는 경우를 연산의 체이닝(chaining)이라 한다. 체이닝된 복수개의 연산은 그 수행시간의 합이 1-싸이클타임보다 크지 않아야 한다. (그림 3)은 제어스텝의 시간간격이 100nS, '*' 연산의 지연시간이 80nS, '+' 연산의 지연시간이 40nS인 연산의 체이닝에 대한 예를 나타낸 것이다.



(그림 3) 체이닝과 멀티싸이클링
(Fig. 3) Chaining & Multicycling

체이닝을 고려한 스케쥴링에서는 연산 O_i, O_j, O_k의 지연시간을 각각 d_i, d_j, d_k라 할 때, $(d_j + d_k) \leq tT$인 조건과 $(d_i + d_j + d_k) > tT$인 조건이 동시에 만족되는 경우, 식(5)의 (a)와 (b)와 같은 선후관계식을 만족해야한다.

$$\sum_{h=1}^{r_j} \{P_{jh} \times X(j, h)\} - \sum_{m=1}^{r_k} \{P_{km} \times X(k, m)\} \leq 0 \qquad (5a)$$

$$\sum_{h=1}^{r_j} \{P_{ih} \times X(i, h)\} - \sum_{m=1}^{r_k} \{P_{km} \times X(k, m)\} \leq -1 \qquad (5b)$$

(그림 3(b))와 같이 하나의 연산이 복수개의 제어스텝에 걸쳐 존재하는 멀티사이클링(multicycling)은 제어스텝의 시간 간격을 50nS로 설정한 경우, * 연산의 수행을 위해서는 2 싸이클타임이 필요하다는 것을 알 수 있다. 이 경우, 3개 연산의 수행에 소요되는 제어스텝의 수는 2가 되며, 연산의 수행에 걸리는 시간은 100nS 로서 체이닝의 경우와 동일하다. 그러나, 멀티

사이클링은 지연시간이 짧은 연산자가 서로 다른 제어스텝에 할당되므로, 이들 간에는 자원의 공유가 가능하다는 장점을 갖고 있다. 따라서, (그림 3(b))의 예에서는 하나의 가산기만으로 2개의 '+' 연산을 수행할 수 있게 된다.

멀티사이클 연산의 스케줄링은 사용되는 연산자의 유형에 따라, 비파이프라인(nonpipeline) 스케줄링과 파이프라인(pipeline) 스케줄링으로 구분된다.

멀티사이클 연산은 비파이프라인형 연산자(nonpipelined functional unit)나 또는 파이프라인형 연산자(pipelined functional unit)에 의해 수행될 수 있다.

어느 경우나 멀티사이클 연산을 수행할 수 있다는 점은 동일하나, 비파이프라인형 연산자는 일단 연산의 수행이 시작되면, 해당 연산을 마칠 때까지 다른 연산의 수행에 할당될 수 없는 반면, 파이프라인형 연산자는 연산의 수행 도중에도 다른 연산의 수행에 할당될 수 있기 때문에, 보다 많은 자원공유를 이룰 수 있다. 지연시간이 D_i인 연산 A가 제어스텝 P에 할당되는 경우, 연산유형 t인 연산 O_i의 요소 $O_i(k)$가 제어스텝 P에 할당되기 위해서는 다음의 조건식(6)이 만족되어야만 한다.

$$\sum_{i=1}^{n_t} X(i,(P-S_i+1))-(D_i-1)+\sum_{i=1}^{n_t} X(i,(P-S_i+1)-(D_i-2)) +\cdots\sum_{i=1}^{n_t} X(i,(P-S_i+1)-1)+\sum_{i=1}^{n_t} X(i,(P-S_i+1))\le N_t \quad (6)$$

이를 정리하면, 다음과 같은 관계식(7)로 요약된다.

$$\sum_{j=0}^{D_i}\sum_{i=1}^{n_t} X(i,(P-S_i-1)-j)\le N_t \quad (7)$$

파이프라인형 연산자는, 연산자 내에서 이전 데이터의 연산이 수행되고 있는 동안에도 새로운 데이터가 동일 연산자 내에 유입될 수 있는 기능을 보유하고 있다. 이미 연산이 진행되고 있는 입력 데이터와 새로이 유입되는 데이터간의 시차인 회전 지연시간(latency)이 1 싸이클타임이고, 지연시간이 D_i인 연산자는, $1\le D_i$인 조건하에서 매 1 싸이클타임마다 새로운 연산을 수행할 수 있다.

연산유형 t인 연산의 수행에 필요한 파이프라인형 연산자의 갯수는 연산의 공유가 가능한 것끼리 묶어, 다음 식(8)과 같이 구분 지울 수 있다.

$$N_t=N_{t1}+N_{t2}+\ldots+N_{t2} \quad (8)$$

여기서, Nt_p (p=1, 2,…, L)는 제어스텝-P와 제어스텝 (P+L) 이후의 제어스텝에 존재하는 연산유형 t인 연산이 공유할 수 있는 파이프라인형 연산자의 최대치를 나타낸다. 따라서, 파이프라인형 연산자에 의한 멀티사이클 연산의 스케줄링에서는 자원관계식이 다음의 식(9)와 같이 표현된다.

$$\sum_{j=1}^{n_t} X(i,P-S_i+1)\le Nt_p,\ (p=1,2,\ldots,L) \quad (9a)$$

$$\sum_{i=1}^{n_t} X(i,q-S_i+1)\le Nt_p,\ldots(q=p+L,L+1,\ldots,T_{\max}-D_i+1) \quad (9b)$$

3. 자원제약하의 지연스케줄링

자원의 제약이 존재하는 경우, 제약 조건을 만족하기 위해서는 연산이 할당될 수 있는 제어스텝의 수가 늘어날 수 있는데, 이때에 발생되는 문제는 그 제어스텝의 수가 얼마만큼 확장되며, 확장된 제어스텝 만큼의 시간적 공간에 어떤 연산을 어디에 할당시켜야만, 주어진 자원의 제한조건을 만족할 수 있는가 하는 문제이다.

기존의 모든 자원제약 스케줄링의 경우, urgency, freedom, force와 같은 우선순위(priority)를 이용한 방법들은 선행 연산이 이미 할당된 연산만을 대상으로 이들을 해당 제어스텝에 할당시키는 방법으로 최적의 결과를 보장하지 못한다.

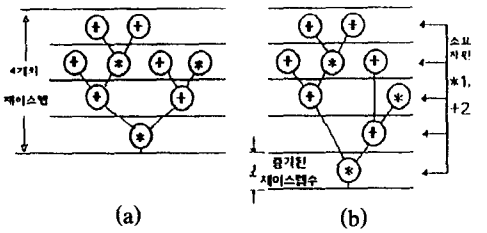
본 연구에서는 자원의 제약이 가해지지 않은 상태에서의 스케줄링 결과로부터, 자원의 제한조건을 만족하는 자원제약 스케줄링을 행함으로서, 자원의 제약이 스케줄링의 전체 단계에 걸쳐 중복되는 것을 방지한 지연(遲延) 스케줄링(delayed scheduling)기법을 제안하였다.

3.1 가용자원과 제어스텝의 확장

성능제약 스케줄링은 (그림 4)의 (a)와 같이 연산이 할당될 수 있는 제어스텝의 수가 4개로 한정된 상태하(성능제약)에서, 최대의 자원공유를 꾀하는 스케줄

링인 반면, 자원제약 스케줄링은 (b)와 같이 사용 가능한 자원의 수가 한정된 상태 하에서, 가장 빠른 시간 내에 동작 알고리즘의 수행을 마치기 위한 스케줄링을 말한다.

DFG의 연산을 제어스텝 내에 주어진 자원제약 조건을 만족하면서 효율적으로 할당시키고자 하는 것이 자원제약 스케줄링의 궁극적인 목표이기는 하나, 스케줄링에 앞서 ℓ 값을 알지 못하고서는 연산이 할당될 수 있는 제어스텝의 범위를 산출할 수 없게 된다. 확장의 정도를 나타내는 ℓ값은 사용 가능한 연산자의 갯수를 나타내는 N_i의 값에 따라 그 크기가 달라지고, ℓ 값에 따라 연산의 할당될 수 있는 하한 제어스텝의 값 L_i가 달라지므로, 정수계획법에 사용되는 수식의 갯수나, 이진변수(二進變數)의 갯수 등이 따라서 변하게 된다.



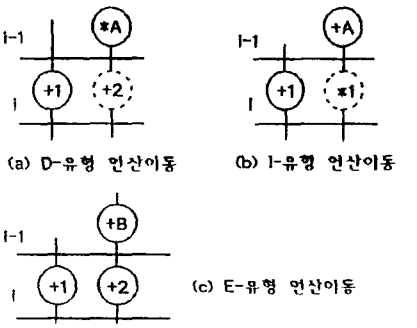
(그림 4) 성능/자원 제약스케줄링
(Fig. 4) Time/Resource constrained scheduling

3.2 연산의 이동유형

제한된 자원으로서 스케줄링을 행하는 경우, 특정 제어스텝 내에서는 자원의 충돌이 필연적으로 발생하게 된다.

임의의 제어스텝에 할당된 연산을 뒤쪽 제어스텝으로 이동시키게 되면, 새로이 할당된 제어스텝 내에는 이동된 연산에 의해 필연적으로 연산의 갯수가 증가하게 된다. 해당 제어스텝에 이미 존재한 연산과 앞쪽 제어스텝으로 부터 이동된 연산의 총량이 사용가능한 연산자의 수를 초과하는 경우는, 앞서와 마찬가지로, 또 다시 초과된 만큼의 연산을 뒤쪽 제어스텝으로 이동시켜야만 한다. 따라서, 이러한 연산의 연속적인 이동을 효과적으로 다루기 위해서는, 연산이 초과로 할당된 제어스텝 내의 연산 중, 어떤 연산을 선택하여 이를 뒤쪽 제어스텝에 할당시키는 것이 가장 효과적인가 하는 것을 결정할 수 있는 방법이 모색되어야만 한다. (그림 5)는 제어스텝 (i-1)내의 연산이 제어스텝 i로 이동됨에 따라, 제어스텝 i에 할당되는 연산의 변화를 나타낸 것이다.

(그림 5)의 (a)는 제어스텝 (i-1)내의 '*A' 연산이 뒤쪽 제어스텝으로 이동됨에 따라, 제어스텝 i의 '+A' 연산의 갯수가 감소하는 경우(D-유형)를 나타낸 것이고, (b)는 '+B'연산의 이동에 의해 오히려 뒤쪽 제어스텝의 특정유형의 연산이 증가하는 경우(I-유형)를 나타낸 것이다. (c)는 '+1' 연산의 이동에도 불구하고 제어스텝 i의 '+' 연산의 갯수에는 아무런 변화가 없는 경우(E-유형)를 나타낸 것이다. 그림의 연산 중, 점선으로 표시된 연산은 해당연산이 존재하지 않더라도 동일한 효과가 나타남을 뜻한다.

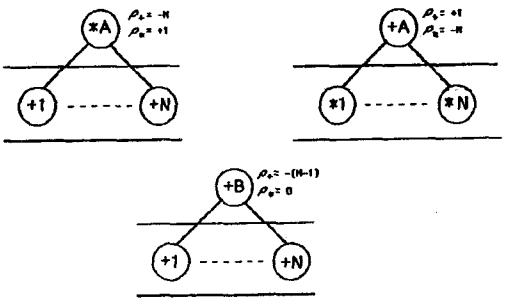


(그림 5) 연산의 이동유형
(Fig. 5) Operation's differing type

임의의 연산 O_i의 연산이동에 의해서 증감되는 연산의 유형인 t인 연산의 갯수를 부가상수(附加常數: additive constant)라 정의하고, 이를 $\rho_t(O_i)$로 표기할 때, (그림 5)와 같이 연산의 팬아웃(fanout)이 1인 경우는, $\rho+(*A)=1$, $\rho+(+A)=+1$, $\rho+(+B)=0$ 및 $\rho*(*A)=+1$, $\rho*(+A)=-1$, $\rho*(+B)=0$인 값을 갖는다. 그러나, (그림 6)과 같이 연산의 팬아웃(fanout)이 N인 경우는 $\rho*(*A)=-N$, $\rho+(+A)=+1$, $\rho+(+B)=-(N-1)$ 및 $\rho*(*A)=+1$, $\rho*(+A)=-N$, $\rho*(+B)=0$인 값을 갖는다.

이러한 각 연산의 부가상수는 지연스케줄링에서 특정 제어스텝의 일부 연산을 뒤쪽 제어스텝으로 이동시키고자 할 때, 이동시킬 연산을 선택하는 기준으

로 사용된다.



(그림 6) 연산의 부가상수
(Fig. 6) Operation's additive constants

3.3 지연스케줄링

자원의 제약이 가해진 조건하에서의 지연스케줄링 기법은 ASAP 스케줄링이나, 또는, 기타의 성능제약 스케줄링 결과를 이용하여, 자원의 제약이 가해진 상태에서의 스케줄링을 행하는 방법이다.

자원제약이 가해지지 않은 상태하에서 임의의 연산이 할당될 수 있는 제어스텝과 자원제약이 가해진 상태하에서 동일 연산이 할당될 수 있는 제어스텝과의 차를 지연상수(遲延常數 : delay constant)라 정의할 때, 지연스케줄링은 이러한 지연상수를 최소화시키기 위한 스케줄링이라고 볼 수 있다.

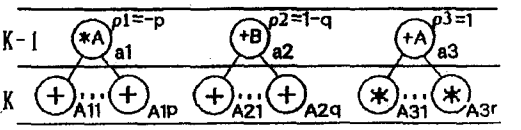
정의된 연산의 지연상수와 가용자원 간에는 관계식(10)이 성립한다.

연산 O_i의 지연상수를 A_i, 연산 O_i의 최근접 후행연산으로써 연산 O_i가 할당된 제어스텝의 바로 뒤 제어스텝에 할당된 연산 O_j의 지연상수를 A_j라 할 때. A_j는 최소한 A_i보다 크거나 같은 값을 가져야만 한다.

$$A_i \le A_j \tag{10}$$

제어스텝 k 내에 할당된 연산유형 t인 연산의 갯수를 n_t, 연산유형 t인 연산을 구현할 수 있는 연산자의 갯수를 $M_t(M_t < N_k)$, 제어스텝 k에 할당된 연산의 지연상수를 $A_i(i=1,...,N_k)$, 지연상수가 A_i인 연산의 선행연산의 지연상수를 a_i라 할 때, 제어스텝-k에 할당된 연산은 최소한 $(n_t - M_t)$개 만큼의 연산이 제어스텝 $(k+1)$로 이동되어야만 제어스텝 k에서의 자원제한조건이 만족된다. 따라서, 이들 연산 사이에서는 다음과 같은 조건식이 만족되어야만 한다.

$$\sum_{i=1}^{n_t} A_i \ge (n_t - M_t),\ (t=1,...,m) \tag{11}$$



(그림 7) 연산의 지연상수와 부가상수
(Fig 7) Delay constants and additive constants

(그림 7)과 같이 제어스텝 $(k-1)$에 존재하는 연산의 지연 상수를 각각 a1, a2, a3라하고 부가상수의 값을 각각 $\rho 1$, $\rho 2$, $\rho 3$ 즉, $\rho+(*A)=\rho 1$, $\rho+(+B)=\rho 2$, $\rho+(+A)=\rho 3$라할 때, 제어스텝 $(k-1)$로 부터, '+' 연산이동에 따른 식(11)의 변화는 다음과 같다.

$$(1-a1)\times\sum_{i=1}^{n} A1_i + (1-a2)\times\sum_{i=1}^{n} A2i \ge (m_t - M_t) + \sum_{i=1}^{3}\rho_i \times a_i \tag{12}$$

식(12)를 정리하면,

$$\sum_{i=1}^{p} A1_i + \sum_{i=1}^{q} A2_i \ge (n_t - M_t) + (\sum_{i=1}^{p}(A1_i + \rho_i)\times a1 \tag{13}$$
$$+(\sum_{i=1}^{q} A2_i + \rho_2)\times a2 + \rho_3 \times a3$$

또한, (그림 7)에서 연산의 팬아웃이 각각 p, q, r이므로 $\rho 1 = -p$, $\rho 2 = (1-q)$, $\rho 3 = 1$이므로, 식(13)은 다음과 같이 표현된다. 즉,

$$\sum_{i=1}^{p} A1_i + \sum_{i=1}^{q} A2_i \ge (n_t - M_t) + (a2 + a3) \tag{14}$$

'*' 연산의 이동에 따른 수식, 역시, 식(14)와 동일하다. (그림 8)의 '*' 연산을 '+' 연산으로 , '+' 연산을 '*' 연산으로 각각 대치시키게 되면, 식(14)와 동일한 결과를 얻을 수 있다.

(그림 1)의 DFG모델에 대하여 1개의 가산기(+),

2개의 곱셈기만(*)을 사용하기 위한 자원제약 상태에서의 지연스케쥴링을 위한 간단한 예를 (그림 8)에 나타내었다. (그림 8)의 연산 왼쪽에 표시된 기호(여기서는 M1, M2,...C1등으로 표시)는 각 연산의 지연상수를 나타낸 것이다. 선후관계가 유지되는 지연상수간의 관계는 관계식(10)으로부터 다음과 같다.

$M1 \le M4,\ M2 \le M4,\ M4 \le S1 \le S2$
$M3 \le M5 \le S2,\ M6 \le A2,\ A1 \le C1$

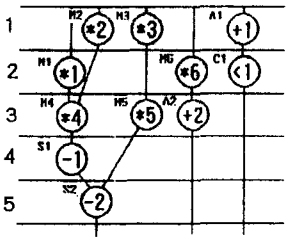
각 제어스텝에서의 지연상수간의 관계는 관계식(14)로부터 〈표 2〉와 같이 표현된다.

〈표 2〉 지연상수간의 관계식
〈Table 2〉 Relations delay constants

제어스텝 1 : $M1 + M2 + M3 + M6 \ge (4-2)$
제어스텝 2 : $M4 + M5 \ge (2-2) + M1 + M2 + M3 + M6$
$A2 + C1 \ge (2-1) + A1$
제어스텝 3 : $0 \ge (0-2) + M4 + M5$
$S1 \ge (1-2) + A2 + C1$
제어스텝 4 : $S2 \ge (1-2) + S1$

지연스케쥴링을 위한 관계식(10~14)을 적용하여 구해진 제어스텝의 수(데이터패스의 지연시간)를 최소로 하는 지연상수의 값은 $M1 = M4 = M5 = M6 = A2 = S1 = S2 = 1$이므로, (그림 1)의 DFG모델에 대하여 ASAP스케쥴링 결과로부터 각 연산을 구해진 지연상수값만큼 제어스텝을 이동시키면 (그림 8)과 같은 결과를 얻을 수 있다.

(그림 8)과 같이 1개의 가산기, 2개의 곱셈기로 제한된 자원을 만족하면서, 모든 연산의 수행이 가능한 데이터패스의 지연시간은 5-싸이클타임이다.



(그림 8) 자원제약하에서의 지연 스케쥴링
(Fig. 8) Delayed scheduling under resource constrained

멀티싸이클 연산을 비파이프라인 연산자로 구현하고자 하는 경우의 지연스케쥴링 방법은 앞절의 스케쥴링 방법과 거의 동일하다. 단지, 멀티싸이클연산은 복수개의 제어스텝에 걸쳐 연산이 수행되기 때문에, 연산의 지연시간에 따른 연산의 지연상수와 부가상수의 관계를 추가로 고려할 뿐이다.

지연시간이 D_i 싸이클타임인 멀티싸이클 연산이 제어스텝 k내에 복수개 존재하는 경우, 제어스텝 k내의 연산의 갯수를 감소시키기 위해서는, N개의 제어스텝 k의 연산 중, 적당량을 제어스텝 (k+1)로 이동시켜야만 한다. 연산 O_i를 제어스텝 (k+1)로 이동시키기 위해서는 각 연산의 지연상수 A_i가 i값을 가져야 하므로, n개 이상의 연산을 이동시키고자 할 때에는 다음과 같은 조건식이 만족되어야만 한다.

$$\sum_{i=1}^{N} (A_i/i) \ge n \tag{15}$$

또한, 제어스텝 (k-1) 이전의 제어스텝으로 부터 I-유형 연산이동이 존재할 때에는 증가된 만큼의 연산이동이 제어스텝 k에서 발생되어야만 한다. 제어스텝 k에서 연산의 수행을 시작하는 연산의 갯수를 L1, 제어스텝 k에서 r/D_i $[r = 1,..., D_i]$ 만큼 연산의 수행이 진행된 연산의 갯수를 Lr, 제어스텝-(k-1)에 존재하는 I-이동유형 연산의 갯수를 qi, 이들의 지연상수를 a_{ij} $(i = 1,..., D_i\ j = 1,..., q_i)$로 표기할 때, 기본적인 지연스케쥴링의 관계식(14)를 다음과 같이 변형시켜, 비파이프라인형 연산자로 구현되는 멀티싸이클연산의 연산이동 관계를 표현한다.

$$\sum_{i=1}^{L_1} A_i + \sum_{i=1}^{L_2} (A_i/2) + \ldots + \sum_{i=1}^{L_r} (A_i/r) \ge (n_t - M_t) + \sum_{i=1}^{Di} \sum_{i=1}^{qi} (a_{ij})/i \tag{16}$$

멀티싸이클 연산을 파이프라인형 연산자로 구현하고자 하는 경우의 지연스케쥴링 방법은 앞에서 논의한 비파이프라인형 연산자를 사용한 지연스케쥴링 방법과 동일하다. 즉, 회전지연시간이 ℓ-싸이클타임인 파이프라인형 연산자를 사용하는 경우는 지연시간이 ℓ-싸이클타임인 비파이프라인형 연산자를 사용

하는 것과 동일한 방법으로 지연스케줄링을 행할 수 있다. 기능적 파이프라이닝(functional pipelining)은 구현하고자 하는 데이터패스로 하여금 일종의 파이프라인형 연산자와 같은 기능을 보유토록 하는 기법을 말한다.

지연시간이 D_i 싸이클타임인 데이터패스를 회전지연시간이 ℓ-싸이클타임($\ell \le D_i$)인 기능적 파이프라이닝 구조로 변환하기 위해서는, 연산유형 t인 연산의 갯수 n_t와 수행할 수 있는 연산자의 갯수 N_t 사이에는 관계식(17)이 성립하여야만 한다.

$$N_t \ge \lceil n_t / \ell \rceil \tag{17a}$$

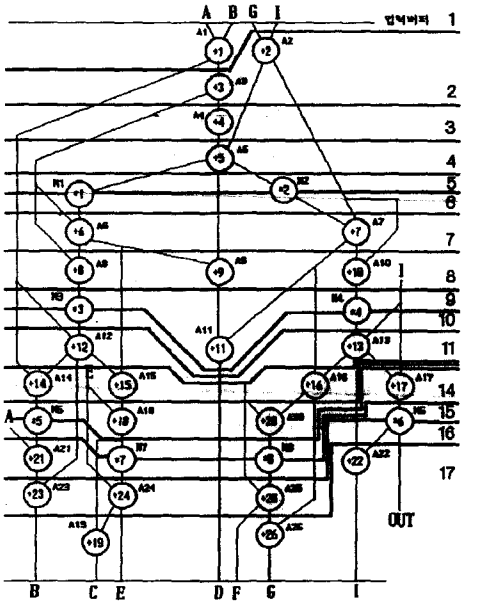
$$N_t \ge \lceil D_i / \ell \rceil \times n_i \tag{17b}$$

식(17a)는 관계식 $2 \times D_i \le i$이 만족되는 경우, 연산유형에 무관하게 적용시킬 수 있는 관계식이며, 식(17b)는 연산유형이 비파이프라인형인 경우에 연산의 지연시간 D_i와 데이터패스의 회전지연시간 은 다음의 관계식을 따른다.

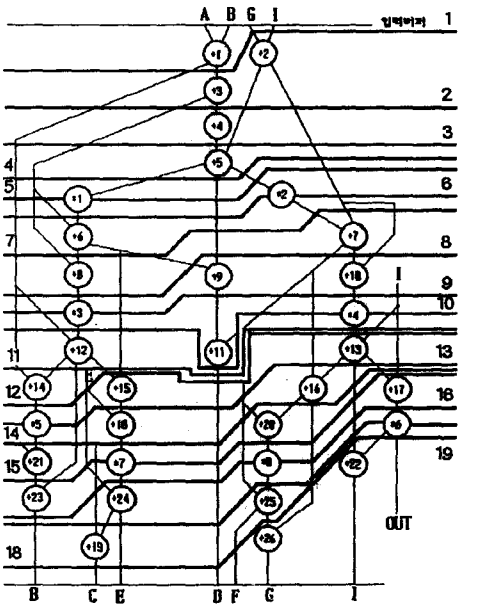
$$\ell = MAX \lceil n_t / M_t \rceil, \ (t = 1, \cdots, m) \tag{18}$$

식(17)은 자원의 제약이 가해지지 않는 상태하에서, 데이터패스 합성 시에 소요되는 연산자의 갯수를 산출할 수 있는 관계식이며, 식(18)은 가용자원(可用資源)이 한정된 상태하에서, 합성된 데이터패스의 데이터 처리속도를 가늠할 수 있는 관계식이다. 따라서, 사용 가능한 연산자의 갯수가 고정되는 비용제약 스케줄링에서는, 식(18)로 부터 구해지는 ℓ값과 가용자원의 조건하에서 지연스케줄링을 행함으로서 기능적 파이프라이닝을 실현할 수 있다.

(그림 9)는 1988년 High-level Synthesis Workshop에서 표준 벤치마크 모델로 채택된 5차 디지털 웨이브필터(fifth-order digital wave filter)의 DFG이다. DFG 모델의 '*' 연산과 '+' 연산의 지연시간은 각각 2 및 1 싸이클타임인 것으로, 멀티싸이클연산인 '*' 연산은 회전지연시간이 1-싸이클타임인 승산기에 의해 수행되는 것으로 간주하였을 때, 그림으로 부터 알 수 있듯이, 지연시간이 17-싸이클타임인 데이터패스가 구성되기 위해서는, 각 제어스텝내에서, 최소한 2개의 파이프라인형 곱셈기와 3개의 가산기가 필요함을



(그림 9) 5차 디지탈웨이브필터 모델
(Fig. 9) Fifth-order digital wave filter



(그림 10) 그림 9의 지연스케줄링 결과
(Fig. 10) Scheduling result of Fig. 9

알 수 있다.

이러한 스케줄링 결과를 이용하여, 하나의 곱셈기와 2개의 가산기로서, 시스템에 입력되는 모든 데이터를 처리할 수 있는 데이터패스 구현을 위한 지연스케줄링을 행한다.

(그림 9)에 대하여 구해진 지연상수의 값에 해당되는 만큼 연산을 뒤쪽 제어스텝으로 이동시키게 되면, (그림 10)과 같이 전체의 제어스텝 내에서 주어진 자원의 제한조건이 만족되고 있는 스케줄링 결과를 얻을 수 있다.

데이터패스의 지연시간이 성능제약 스케줄링 결과에 비해 2-스텝만큼 확장되었다.

4. 실험결과

본 연구에서 제안한 스케줄링 방법의 효용성을 검증하기 위하여 벤치마크 모델에 대한 스케줄링을 수행하였다. 사용된 벤치마크 모델로서는 1988년 High Level Synthesis Workshop에서 표준 벤치마크 모델로 채택된 5차 디지털 웨이브필터(fifth-order digital wave filter)를 대상으로 하였고 스케줄링 문제에 대하여 결과를 구하기 위한 모든 정수계획법은 SUN SPARC-II 하에서 Lingo 패키지 프로그램을 이용하여 해를 구하였다.

실험결과는 공개된 자료가 가장 풍부하고 우수하다고 평가되는 ALPS시스템[9]의 스케줄링 결과와 비교하였다. 5차 디지털 웨이브 필터를 사용한 대부분의 연구에서와 마찬가지로 본 논문에서도, '+' 연산과 '*' 연산의 지연시간을 각각 40nS, 80nS로, 제어스텝의 시간간격은 50nS로 간주하였다.

〈표 3〉은 비파이프라인형 곱셈기를 이용한 비파이프라인 데이터패스에 대한 스케줄링 결과로서 ALPS의 결과와 일치함으로써 제안된 기법에 대한 효용성을 입증할 수 있었다.

〈표 4〉는 회전지연시간이 2싸이클타임이고 지연시간이 3싸이클타임인 파이프라인형 곱셈기로 수행시켰을 때의 결과이다.

〈표 3〉 비파이프라인형 곱셈기를 이용한 파이프라인형 데이터패스 스케줄링 결과

〈Table 3〉 Pipelined Data-path scheduling result(with nonpipelined multiplier)

회전 지연 시간	1	2	3		4	5	6	7	8	9	13	16	26
소요 자원	+26 *16	+13 *8	+9 *6	+9 *8	+7 *4	+6 *4	+5 *3	+4 *3	+4 *2	+3 *2	+2 *2	+2 *1	+1 *1
지연	17	17	17	na	18	19	19	18	20	21	23	21	33
시간	17	17	na	17	18	19	19	18	20	21	23	21	33

〈표 4〉 파이프라인형 곱셈기를 이용한 파이프라인형 데이터패스 스케줄링 결과

〈Table 4〉 Pipelined Data-path scheduling result(with pipelined multiplier)

회전 지연 시간	1	2	3	4	5	6	7	8	9	13	16
소요 자원	+26 *16	+13 *8	+9 *4	+7 *4	+6 *3	+5 *3	+4 *2	+4 *2	+3 *2	+2 *1	+2
지연 시간	20	20	20	20	21	23	22	23	24	26	24

5. 결 론

본 연구에서는 데이터 패스 스케줄링을 위한 새로운 방법으로 지연스케줄링 기법을 제시하였다. 지연스케줄링은 ASAP/ALAP 또는 성능제약 스케줄링에 의해 생성된 결과를 이용하여 자원의 제약이 가해진 상태에서 효과적인 스케줄링이 가능하도록 하는 방법이다. 지연스케줄링 알고리즘을 이용하여 최적의 결과 해를 구하기 위하여 선형 정수계획법(ILP)[2][11]을 이용함으로써 수식의 생성과 분석 및 재구성이 용이하였다. 기술된 지연스케줄링의 선형정수계획법 수식의 스케줄링 성능을 평가하기 위하여 표준 벤치마크모델인 5차 디지털 웨이브 필터(Fifth-order digital wave filter) 모델에 대하여 스케줄링을 수행하므로 써 결과는 다른 논문에서의 결과와 일치함을 얻을 수 있었다. 본 논문에서는 연산의 스케줄링과 자원의 공유만을 중점적으로 다루었지만 완전한 데이터 패스의 합성을 위해서는 연산자뿐만 아니라 레지스터, 및 데이터 전송로를 데이터 패스에 효과적으로 할당시키

는 하드웨어 할당 과정이 수행되어야 하기 때문에 이에 대한 연구가 지속되어야 하겠다.

참 고 문 헌

[1] Alice. parker, "Automated Synthesis of Digital System", IEEE D & T, Nov. 1984, pp. 71-81.

[2] Jiahn-Hung Lee, Yu-Chin Hau, Youn-Long Lin "A New Integer Linear Programming Formulation for the Scheduling Problem In Data Path Synthesis", Proceedings of International Conference on Computer-Aided Design, 1989, pp. 20-23.

[3] Michael C. McFarland, SJ Alice C. Parker,Raul Camposano "Tutorial on High-Level Synthesis", IEEE Design Automation Conference., 1988, pp. 330-336.

[4] A. C. Parker, et al "MAHA : A Program for Data Path Synthesis", Proc. of 23rd DAC., pp. 461-466, 1986.

[5] P. G. PAULIN, J. P. KNIGHT, E. F. GIRCZYC "HAL : A Multi Paradigm Approach to Automatic Datapath Synthrsis", IEEE Design Automation Conference. 1986, pp. 263-270.

[6] Ki Soo Hwang, Albert E. Casavant, Ching-tang Chang and Manuel A. d'Abreu "Scheduling and Hardware Sharing In Pipelined Data Paths", IEEE. 1989, pp. 24-27.

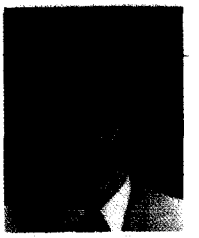
[7] Daniel D. Gajski,Nikil D. Dutt and Barry M. Pangrle "Silicon Compilation(TUTORIAL)", IEEE Custom Integratedcircuits Conference 1986, pp. 102-110.

[8] Linus Schrage, Kevin Cunningham, "LINGO : Optimization Modeling Lanuage", Lindo System Inc., 1991.

[9] Cheng-Tsung Hwang, Jiahn-Hurng Lee, and Yu-Chin Hsu, "A Formal Approach to the Scheduling Problem in High Level Synthesis", IEEE. 1991, pp. 464-475.

[10] S.Y.KUNG, T.KAILATH, "VLSI and Modern Signal Processing", Printice Hall, Inc., 1985.

[11] Daniel D.Gajski, Nikil D.Dutt, Allen C-H Wu, "High-Level Synthesis Introduction to Chip and System Design", Klwer Academic Publishers, 1992, pp. 213-258.



신 인 수

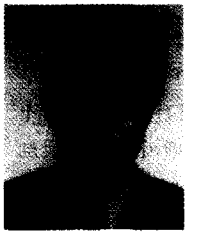
1991년 청주대학교 전자공학과 졸업(학사)

1993년 청주대학교 대학원 전자공학과(공학석사)

1994년~현재 청주대학교 대학원 전자공학과(박사수료)

1997년~현재 청주기능대학 전자기술학과 전임강사

관심분야 : 디지털시스템, VLSI & CAD, High-level Synthesis



이 근 만

1973년 한양대학교 전자공학과 졸업(학사)

1980년 한양대학교 대학원 전자공학과(공학석사)

1992년 한양대학교 대학원 전자공학과(공학박사)

1982년~현재 청주대학교 전자공학과 교수

관심분야 : 디지털시스템, VLSI & CAD, High-level Synthesis