論 文

자원제약조건 하에서의 데이타패스 스케듈링

正會員 李 近 萬* 正會員 林 寅 七*

A Datapath Scheduling Under Resource Constraints

Keun Man Yi* In Chil Lim* *Regular Members*

要 約

본 논문에서는 고위영역합성(High-level synthesis)의 가장 중요한 과제인 스케듈링 문제를 다루었다. 스케듈링 문제에 대한 접근방식으로서, IP(integer programming)을 이용한 방식을 택하였다.

본 논문에서는 특히, 가용(可用)자원이 제한된 상태에서의 스케듈링이 효율적으로 수행될 수 있는 방법을 연구하여, 임의의 스케듈링방식에 의해 구해진 스케듈링 결과로부터, 주어진 조건을 만족하는 스케듈링을 행할 수 있는 방법을 연구하였다.

멀티싸이클 연산의 연산자할당 및 구조적 파이프라이닝을 위한 스케듈링에 중점을 두어, 가능한한 최대의 성능과 최대의 자원공유가 이루어 지도록, 연산자의 특성을 세밀히 분석하였다.

ABSTRACT

This paper deals with the scheduling ploblems, which are the most important subtasks in High-level synthesis. IP(integer programming) formulations is used as the scheduling problem approach.

This paper descibes a new resource-constraints scheduling algorithm.

We have concentrated our attentions on the multicycle operations and the structural pipelining, and we fully analyze the characteristics of operators to achieve the maximal performance and the maximal resource sharing.

For experiment results, we choose the 5-th order digital wave filter as a benchmark and do the schedule. Finally, we can obtain near-optimal scheduling results.

I. 서 론

데이타패스 합성(Datapath Synthesis)과정 중, 가장 중요한 것이 스케듈링(Scheduling) 과정과 하드웨어할당(Hardware Allocation) 과정이다[1]. 스케듈링 과정에서는 DFG(Data Flow Graph) 상의 연산

*漢陽大學校 電子工學科
Dept. of Elec. Eng., Hanyang Univ.
論文番號 : 92-43(接受1992. 3. 12)

을 적절한 제어스텝에 할당시키고, 하드웨어할당 과정에서는 연산자(operator, functional unit), 레지스터 및 데이타이동로(communication path)를 데이타패스에 할당시킨다. 일단, 연산이 해당 제어스텝에 할당되고 나면, 연산자의 종류와 갯수, 변수의 생존시간(life time) 및 시간적인 제약조건(timing constraints)이 고정되므로, 스케듈러(Scheduler)의 성능은 이후의 데이타패스 합성과정에 큰 영향을 미치게 된다. 이러한 스케듈링과 하드웨어할당은 DFG의 본래 특성을 그대로 유지함은 물론, 다음과 같은 조건, 즉,

(1)데이타 의존성(data dependency)
(2)시간간격(stage time)
(3)면적비용(area-cost)
(4)성능(performance)

이 만족될 수 있도록 수행되어야만 한다.

비용제약 스케듈링은, 자원의 수가 제한된 조건 하에서 주어진 제약조건을 만족하는 최대성능의 스케듈링을 말한다. 사용가능한 자원의 수가 제한된 조건하에서, 동일유형의 연산을 전 제어스텝에 걸쳐 균일하게 분포시키기 위해서는, 이들 연산이 균일하게 분포될 수 있는 제어스텝 수의 상하한 값을 결정한 후, 제어스텝의 수를 최소화 시키는 방향으로 스케듈링을 행하면 된다.

자원의 제약이 존재하지 않을 경우, 임의의 연산이 할당될 수 있는 상.하한 제어스텝은 ASAP 스케듈링[2][3] 및 ALAP[4] 스케듈링에 의해 결정된다.

그러나, 자원의 제약이 존재하는 경우는 연산이 할당될 수 있는 제어스텝의 폭이 필연적으로 늘어나게 된다. 이때에 발생되는 문제는 그 폭이 얼마만큼 확장되며, 확장된 제어스텝 만큼의 시간적 공간에 어떤 연산을 어디에 할당시켜야만, 주어진 자원의 제약조건을 만족할 수 있는가 하는 문제이다.

이러한 문제는 기존의 list 스케듈링[5][7] 기법으로 해결할 수 있다. List 스케듈링은 용어가 뜻하는 바와 같이, list에 할당 시키고자 하는 연산을 우선순위에 따라 열거시킨 후, 모든 선행연산(predecessor)이 旣할당된 연산만을 대상으로 이들을 해당 제어스텝에 할당시키는 방법이다. 자원의 제약에 의해 할당시키고자 하는 연산의 수가 사용가능한 자원의 수를 초과할 때에는, 초과분에 대한 할당을 다음 제어스텝으로 미룬다. 연산의 할당에 사용되는 우선순위로서는 연산의 'freedom'[5][6], 'force'[7], 'urgency'[14]와 같은 우선순위함수(priority function)가 이용된다. 이와 같은 list 스케듈링 기법에 의한 스케듈링은, branch-and-bound 방식의 스케듈링과 거의 동일한 결과를 얻을 수 있으나, 모든 경우에 대하여 언제나 최적의 결과를 보장하지는 못한다[1].

Branch-and-bound 스케듈링 범주에 속하는 IP 수식에 의한 스케듈링의 목적은, 모든 경우에 대하여 최적의 결과를 보장하지 못하는 list 스케듈링의 단점을 보완하기 위한 것이다. 이러한 이유로 인하여, list 스케듈링 기법과 ASAP 스케듈링 및 ALAP 스케듈링으로 부터 얻어지는 제어스텝의 최하한값 및 연산의 상.하한 제어스텝 값을 기본으로 하여, 제어스텝의 최하한 값을 최소화 하기 위한 스케듈링 기법이 제안되었다[8][10]. 제안된 논문에서는 list 스케듈링 기법에 의해 연산의 스케듈링을 행하여 제어스텝의 최하한 값을 구한 후, 이로부터 얻어진 결과에 대하여 재차 ASAP 스케듈링과 ALAP 스케듈링을 행하여, 연산이 할당될 수 있는 상.하한 제어스텝의 값을 산출한다. 이후, 구해진 제어스텝의 최하한 값과 연산의 상.하한 제어스텝 값으로 부터 범위관계식(range relation)과 자원관계식(resource relation) 및 시간관계식(time relation)의 IP 수식을 생성하여 주어진 자원에서의 최적해를 얻고자 하였다.

그러나, 이러한 기법은 다음과 같은 단점을 내포하고 있다. 즉, list 스케듈링에 의해 구해진 제어스텝의 최하한 값을 상한(upper limit)으로 하여, 재차, 자원의 제약이 가해진 상태하에서의 ASAP 스케듈링－변형형 ASAP 스케듈링－과 ALAP 스케듈링－변형형 ALAP 스케듈링－을 행하기 때문에, 스케듈링의 효율이 나빠진다. 다시 말해서, 3개의 스케듈링 과정이 모두 자원의 제약이 가해진 상태하에서의 스케듈링이기 때문에 불필요한 조건의 중첩이 발생된다. 이러한 단점을 해결하기 위하여, 본 논문에서는 자원의 제약이 가해지지 않은 상태에서의 스케듈링 결과로 부터, 주어진 자원의 제약조건을 만족하는 비용제약 스케듈링을 행함으로서, 자원의 제약이 스케듈링의 전 단계에 걸쳐 중복되는 것을 방지한, 지연스케듈링(Delayed scheduling) 기법을 제안하였다.

Ⅱ. IP 모델링

1. 기본이론

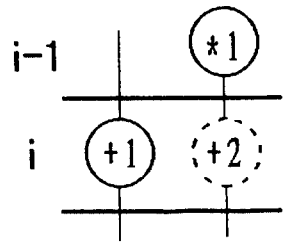
제한된 자원으로서 스케쥴링을 행하는 경우, 특정 제어스텝 내에서는 자원의 충돌이 필연적으로 발생하게 된다. 이 경우, 초과된 만큼의 연산을 뒷쪽 제어스텝에 할당시킴으로서 해당 제어스텝의 자원의 충돌을 방지할 수 있다. 그러나, 임의의 제어스텝에 할당된 연산이 뒷쪽 제어스텝으로 이동되게 되면, 새로이 할당된 제어스텝 내에서 이동된 연산에 의해 필연적으로 연산의 갯수가 증가하게 된다. 해당 제어스텝에 기 존재한 연산과 앞쪽 제어스텝으로부터 이동된 연산의 총량이 사용가능한 연산자의 수를 초과하는 경우는, 앞서와 마찬가지로, 또 다시 초과로 할당된 연산의 이동을 행하여야만 한다.

따라서, 이러한 연산의 연속적인 이동을 효과적으로 다루기 위해서는, 연산이 초과로 할당된 제어스텝 내의 연산 중, 어떤 연산을 선택하여 이를 뒷쪽 제어스텝에 할당시키는 것이 가장 효과적인가 하는 것을 결정할 수 있는 방법이 모색되어야만 한다.

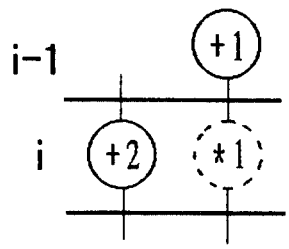
그림 1은 제어스텝-(i-1) 내의 연산이 제어스텝-i로 이동됨에 따라, 제어스텝-i에 할당되는 연산의 변화를 나타낸 것이다. 그림 1(a)는 제어스텝-(i-1) 내의 '*1' 연산이 뒷쪽 제어스텝으로 이동됨에 따라, 제어스텝-i의 '+' 연산의 갯수가 감소하는 경우를 나타낸 것이고, 그림 1(b)는 '+1' 연산의 이동에 의해 오히려 뒷쪽 제어스텝의 특정유형의 연산이 증가하는 경우를 나타낸 것이다. 그림 1(c)는 '+1' 연산의 이동에도 불구하고 제어스텝-i의 '+' 연산의 갯수에는 아무런 변화가 없는 경우를 나타낸 것이다. 그림의 연산 중, 점선으로 표시된 연산은 해당연산이 존재하지 않더라도 동일한 효과가 나타남을 뜻한다.

어떠한 형태의 DFG라 할지라도, 임의의 연산이동은 그림 1의 (a) 및 (b)와 같은 이유형(異類型) 연산간의 이동과 (c)의 경우와 같은 동일유형 간의 연산이동으로 구분되고, 각각은 이동되는 제어스텝의 연산의 수를 증감시키기 때문에, (a)와 같은 형태의 연산이동을 I-유형 연산이동, (b)와 같은 형태의 연산이동을 E-유형 연산이동, (c)와 같은 형태의 연산이동을 D-유형 연산이동으로 구분지운다.

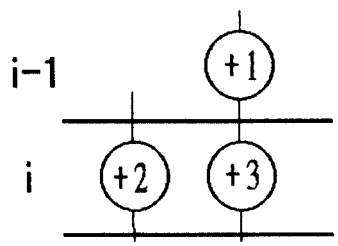
사용가능한 연산자가 승산기 하나와 가산기 하나인 경우에, 제어스텝-(i-1)과 제어스텝 i 간에 그림 1 (a)와 같은 D-유형 연산이동이 가능하다면 이러한 연산이동이 가장 바람직 하나, D-유형 연산이동과 같은 이동형태가 불가능한 경우에는 I-유형 연산이동, 또



(a) D-유형 연산이동



(b) I-유형 연산이동



(c) E-유형 연산이동

그림 1. 연산의 이동형태
Fig. 1. Deferring types.

는, E-유형 연산이동 행하면서 자원의 제약조건을 만족할 수 있도록, 제어스텝-i 내의 연산의 일부, 또는, 전부를 제어스텝-(i+1)로 이동시켜야만 한다.

그림 1에서 알 수 있는 바와 같이 임의의 연산이동은 이동되는 제어스텝의 특정 연산유형의 연산의 갯수를 증감시킨다. 임의의 연산 O_i의 연산이동에 의해 증감되는 연산유형 t의 연산 O_i의 갯수를 부가상수(負加常數)라 정의하고, 이를 σ_{it}로 표기 할 때, 연산 O_i가 단일출력인 경우, σ_{it}는 그림 1과 같이 -1과 +1

및 0의 값을 갖는다. 그러나, 그림 2와 같이 연산 O_i가 복합출력인 경우의 σ_{ti}는 +1 부터 -N까지의 값을 갖게 된다 .

이러한 각 연산의 부가상수는 이후 언급할 지연스케듈링에서 특정 제어스텝의 일부 연산을 뒷쪽 제어스텝으로 이동시키고자 할 때, 이동시킬 연산을 선택하는 기준으로 사용된다.

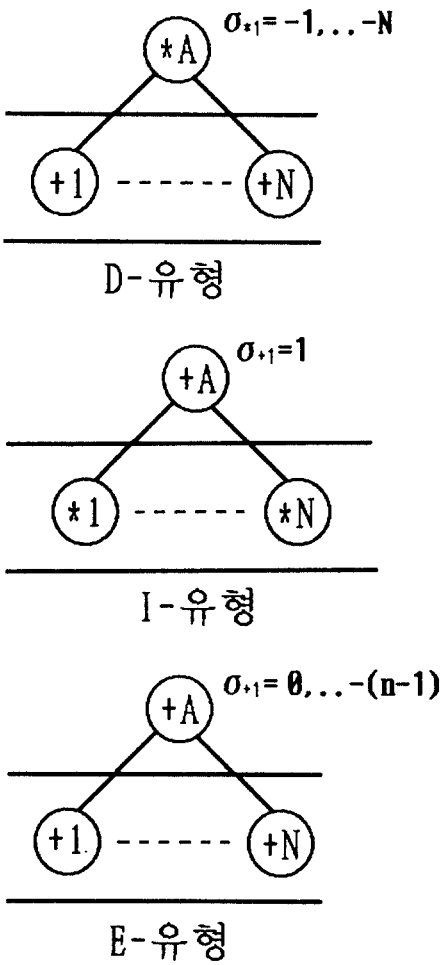


그림2. 복합출력연산의 부가상수

Fig.2. Imposing constants of multiple-output operations.

2. 지연(遲延)스케듈링

자원의 제약조건 하에서, 연산이 할당되는 제어스텝의 최하한 값 T_{max}을 구하기 위한 지연스케듈링 기법에 관해 논한다.

지연스케듈링 기법은 ASAP 스케듈링이나, 또는, 기타의 성능제약 스케듈링 결과를 이용하여, 자원의 제약이 가해진 상태에서의 스케듈링을 행하는 방법이다.

ASAP 스케듈링, 또는, 성능제약 스케듈링에 의해 각 제어스텝에 할당된 연산은 자원의 제약조건이 부합되지 않을 경우, 하나 또는 그 이상의 연산이 뒷쪽 제어스텝에 할당되어야만 한다. 즉, 제어스텝 T_i에 할당된 연산 O_i가 자원의 제약에 의해 제어스텝 T_j에 할당되어야만 하는 경우, 연산 O_i는 (j-i) 스텝 만큼 연산의 할당이 미루어야만 한다. 이와같이, 임의의 연산이 자원의 제약이 가해지지 않은 상태하에서 할당될 수 있는 제어스텝과 자원의 제약이 가해진 상태하에서 할당될 수 있는 제어스텝과의 차를 지연상수라 정의할 때, 지연스케듈링은 이러한 지연상수를 최소화 시키기 위한 스케듈링이라고 볼 수 있다.

앞서 정의한 연산의 지연상수와 가용자원 간에는 다음과 같은 관계식이 성립한다. 연산 O_i의 지연상수를 A_i, 연산 O_i의 최근접후행연산으로서 연산 O_i가 할당된 제어스텝의 바로 뒤 제어스텝에 할당된 연산 O_j의 지연상수를 A_j라 할 때, A_i는 최소한 A_j보다 크거나 같은 값을 가져야만 한다. 다시 말해서, 연산 O_i의 할당이 유예된 만큼 또는 그 이상의 시간 만큼, 연산 O_j의 할당이 유예되어야만 한다. 따라서, A_i와 A_j 사이에는 식(1)과 같은 관계식이 성립하여야만 한다.

$$A_i \leq A_j \tag{1}$$

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

$$\sum_{i=1}^{N_k} A_i \geqq (N_k - M_t) \tag{2}$$

또한, 제어스텝-(k-1)에서 연산의 이동이 존재할 경우는 이동된 연산 만큼의 연산이 추가로 후행 제어스텝으로 이동되어야만 한다. 그림 3과 같이 제어스텝-(k-1)에 존재하는 연산의 지연상수와 부가상수를 각각, a1, a2, a3 및 σ_1, σ_2, σ_3라 할 때, 제어스텝(k-1)로 부터의 연산이동에 따른 식(2)의 변화는 다음과 같다.

$$(1-a_1)*\sum_{i=1}^{\ell} A_{1i}+(1-a_2)*\sum_{i=1}^{m} A_{2i} \geqq (N_k-M_1)+\sum_{i=1}^{3} \sigma_i*a_i \tag{3}$$

식을 정리하면,

$$\sum_{i=1}^{\ell} A_{1i}+\sum_{i=1}^{m} A_{2i} \geqq (N_k-M_1)+(\sum_{i=1}^{\ell} A_{1i}+\sigma_1)*a_1 + \left(\sum_{i=1}^{m} A_{2i}+\sigma_2\right)*a_2+\sigma_3*a_3 \tag{4}$$

또한, $\sigma_1=-\ell$, $\sigma_2=(1-m)$이므로, 식(4)는 다음과 같이 표현된다. 즉,

$$\sum_{i=1}^{\ell} A_{1i}+\sum_{i=1}^{m} A_{2i} \geqq (N_k-M_1)+(a_2+a_3) \tag{5}$$

식(5)의 좌변은 제어스텝-k에 존재하는 동일유형 연산의 지연상수의 총합을 나타내고, 식의 우변 중, 제1항은 제어스텝-k에 존재하는 동일유형 연산의 갯수와 가용자원과의 차를 나타내는 연산의 초과분을 나타내고, 제2항은 제어스텝-(k-1) 내의 연산으로서 제어스텝-k의 연산과 유산유형이 동일한 연산의 지연상수의 총합으로서, 연산의 이동에 의해 증가되는 제어스텝-k의 연산의 갯수를 나타낸다.

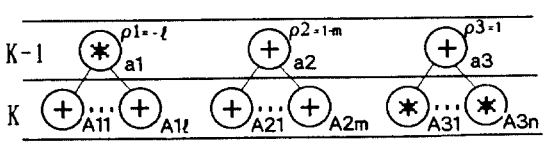


그림3. 복합출력연산의 이동형태
Fig.3. Deferring types of multiple-output operations.

수식의 논리적인 의미는 다음과 같다. 즉, 좌변은 제어스텝-k에서 제어스텝-(k+1)로 이동되어야만 하는 연산의 갯수를, 우변의 제어스텝-(k-1)에서 제어스텝-k로 이동되는 연산의 갯수를 각각 의미한다.

식(1)과 식(5)의 관계식이 만족되는 범위하에서 제어스텝의 하한값을 최소로 하기 위한 스케쥴링을 행할 경우, 주어진 자원의 제약조건을 만족하는 비용제약 스케쥴링의 결과를 얻을 수 있다. 또한, 가용자원의 갯수를 나타내는 M_1의 값을 최소로 하기 위한 스케쥴링을 행하는 경우는, 주어진 제어스텝의 범위안에서 모든 연산을 수행시킬 수 있는 최소비용을 구할 수도 있다.

그림4의 DFG 모델에 대해 1개의 가산기와 2개 승산기에 의한 비용제약 스케쥴링을 행하기로 한다. 각 연산의 좌우측 상단에 표시된 기호는 연산의 지연상수를 나타낸 것이다. 수식(1)은 각 연산의 지연상수가 만족하여야만 하는 조건식이므로, 이에 따라, 지연상수 간의 관계식을 기술하면 아래와 같다.

M1 ≦ M4, M2 ≦ M4
M4 ≦ S1≦S2
M3 ≦ M5 ≦ S2
M6 ≦ A2
A1 ≦ C1

수식(5)는 자원의 제약조건을 초과하는 제어스텝에서의 지연상수 간의 관계를 기술한 식이다.

제어스텝-1 : M1+M2+M3+M6 ≧ (4-2)
제어스텝-2 : M4+M5≧(2-2)+M1+M2+M3+M6
A2+C1 ≧ (2-1)+A1
제어스텝-3 : 0 ≧ (0-2)+M4+M5

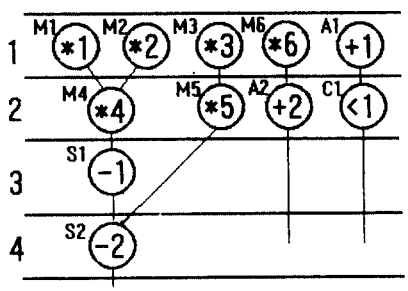


그림4. DFG 모델
Fig.4. A DFG model.

$$S1 \geq (1\text{-}2)+A2+C1$$

제어스텝-4 : $S2 \geq (1\text{-}2)+S1$

이러한 관계식을 만족하면서, 제어스텝의 갯수-데이타패스의 지연시간-을 최소로 하는 지연상수의 값은 M1=M4=M5=M6=A2=S1=S2=1이며, 데이타패스의 지연시간은 5 싸이클타임이다. 따라서, 2개의 승산기와 하나의 가산기로서는 5-싸이클타임 이내에서 모든 연산의 수행이 가능하다.

3. 멀티싸이클연산과 지연스케듈링

멀티싸이클 연산을 비파이프라인 연산자로 구현하고자 하는 경우의 지연스케듈링 방법은 2절의 스케듈링 방법과 기의 동일하다. 단지, 멀티싸이클 연산은 복수개의 제어스텝에 걸쳐 연산이 수행되기 때문에, 연산의 지연시간에 따른 연산의 지연상수와 부가상수의 관계를 추가로 고려할 뿐이다.

그림5와 같이 지연시간이 d인 멀티싸이클 연산이 제어스텝-k 내에 복수개 존재하는 경우를 가정한다. 이 경우, 제어스텝-k의 연산의 수를 감소시키기 위해서는, 4개의 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.

$$\sum_{i=1}^{i} A_i / i \geq N \qquad (6)$$

또한, 제어스텝-(k-1) 이전의 제어스텝으로 부터 I-유형 연산이동이 존재할 때에는 증가된 만큼의, D-유형 연산이동이 존재할 때에는 감소된 만큼의 연산이동이 제어스텝-k에서 발생되어야만 한다.

제어스텝-k에서 연산의 수행을 시작하는 연산의 갯수를 L1, 제어스텝-k에서 r/d [r=1,...,d]만큼 연산의 수행이 진행된 연산의 갯수를 Lr, 제어스텝-(k-1) 내 I-유형연산의 갯수를 N, 이들의 지연상수를 a_i로 표기할 때, 기본적인 지연스케듈링의 관계식 (6)을 다음과 같이 변형시켜, 비파이프라인형 멀티싸이클 연산의 연산이동 관계를 표현한다.

$$\sum_{i=1}^{L1} A_i + \sum_{i=1}^{L2} A_i /2 + \ldots + \sum_{i=1}^{Lr} A_i / r \geq (N_k - M_k) + \sum_{i=1}^{N} {}^{*}a_i \qquad (7)$$

그림4와 같은 DFG 모델에 대하여 식(7)을 적용시켜, 연산의 이동과 자원의 정렬이 행해지는 과정을 다루기로 한다. 그림의 '*' 연산을 지연시간이 2-싸이클타임인 멀티싸이클 연산으로 간주한다.

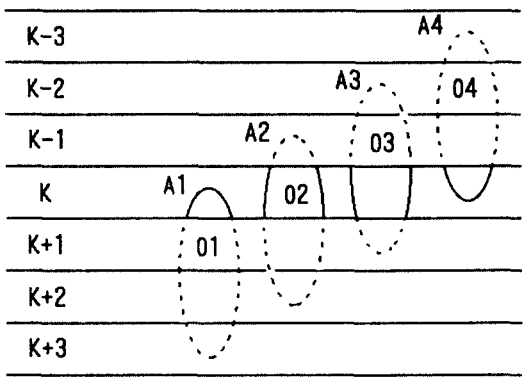


그림5. 비파이프라인형 연산자
Fig.5 Nonpipelined operators.

그림4의 DFG 모델에 대해 1개의 가산기와 2개의 승산기에 의한 비용제약 스케듈링을 행하기 위한 관계식은 다음과 같다.

제어스텝-1 : $M1+M2+M3+M6 \geq (4\text{-}2)$

제어스텝-2 : $(M1+M2+M3+M6)/2 \geq (4\text{-}2)$
$C1 \geq (1\text{-}1)+A1$

제어스텝-3 : $(M4+M5) \geq (2\text{-}2)+M1+M2+M3+M6$
$A2 \geq (1\text{-}1)+C1$

제어스텝-4 : $1/2^{*}(M4+M5) \geq (2\text{-}2)$

제어스텝-6 : $S2 \geq (1\text{-}1)+S1$

이러한 관계식과 지연상수 간의 조건식을 만족하면서, 데이타패스의 지연시간을 최소로 하는 지연상수의 값은 M1=M3=M4=M5=S1=S2=2이며, 데이타패스의 지연시간은 8-싸이클타임이다. 따라서, 그림4의 DFG 모델은 하나의 가산기와 2개의 승산기-지연시간이 2-싸이클타임인 승산기-로서 8-싸이클타임 이내에서 스케듈링을 행할 수 있다.

멀티싸이클 연산을 파이프라인형 연산자로 구현할 때의 지연스케듈링 기법을 기술하기 위하여, 가상연

산자(image operator)를 아래와 같이 정의하여 사용한다.

(정의 1)

임의의 연산자 F_i의 가상연산자는, 연산자 F_i와 동일한 데이타 처리기능을 보유할 수 있는 연산자 집합을 말한다.

(정리 1)

지연시간이 D이고 제어스텝-k에서 연산의 수행을 시작하는 임의의 멀티싸이클 연산 O_i를, latency가 ℓ인 파이프라인형 연산자 F_i로 구현하고자 하는 경우, 연산자 F_i의 n차 가상연산자는 제어스텝-$(k+n\ell)$에서 연산의 수행을 시작하는 복수개의 비파이프라인형 연산자이다($n=1,\cdots,\lfloor D/\ell \rfloor$).

(증명)

연산 O_i에 입력된 i-번째 데이타가 연산의 수행을 시작한 후, $(n*\ell)$-싸이클타임 뒤에는, $(i+n)$ 번째의 데이타가 새로이 연산 O_i에 입력된다. 이경우, 연산 O_i는 i-번째 부터 $(i+n)$-번째 까지의 모든 입력데이타를 동시에 처리하여야만 한다. 따라서, n개의 비파이프라인형 연산자를 시간간격이 ℓ-싸이클타임 되도록 계단형태로 나열시켜 입력된 데이타를 순차적으로 처리시키면, 기능적으로 지연시간이 D인 파이프라인형 연산자와 동일한 기능을 보유할 수 있다. 그러므로, $(i+1)$-번째 입력되는 데이타를 처리하는 가상연산자를 1차 가상연산자, $(i+n)$-번째 입력되는 데이타를 처리하는 가상연산자를 n차 가상연산자라고 명명할 때, 제어스텝-k에서 연산의 수행을 시작하는 파이프라인형 연산자의 n차 가상연산자는 제어스텝-$(k+n\ell)$에서 연산의 수행을 시작하는 비파이프라인형 연산자이다.

그림6은 파이프라인형 연산자의 가상연산자를 도시해 놓은 것이다. 임의의 파이프라인형 연산자는 하나의 비파이프라인형 연산자와 n개의 가상연산자로 구성되어, 비파이프라인형 연산자 F_i가 i-번째 데이타를 처리하는 도중에 유입되는 $(i+n)$-번째의 데이타는 n차-가상연산자에 의해 처리됨을 알 수 있다.

임의의 파이프라인형 연산자는 계단형태의 복수개 비파이프라인형 연산자와 기능적으로 동일한 것으로 간주할 수 있고, 또한, 가상연산자는 실제로 존재하지 않는 연산자이므로, 자원공유의 관점에서 볼 때,

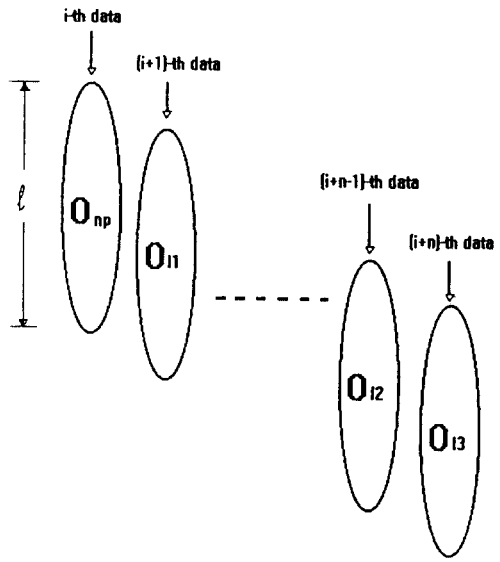


그림6. 파이프라인형 연산자의 가상연산
Fig.6. Image operator of a pipelined operator.

latency가 ℓ인 파이프라인형 연산자는 지연시간이 ℓ인 비파이프라인형 연산자로 간주하여여도 무방하다. 따라서, 멀티싸이클 연산을 파이프라인형 연산자로 구현하고자 하는 경우의 지연스케쥴링 방법은 앞에서 논의한 비파이프라인형 연산자를 사용한 지연스케쥴링 방법과 동일하다. 즉, 데이타 입력 latency가 ℓ인 파이프라인형 연산자를 사용하는 경우는 지연시간이 ℓ-싸이클타임인 비파이프라인형 연산자를 사용하는 것과 동일한 방법으로 지연스케쥴링을 행할 수 있다.

그림4의 '*' 연산이 지연시간이 3-싸이클타임인 연산이라 가정하고, 이를 latency가 2인 파이프라인형 승산기로 구현하는 경우, 1개의 가산기와 2개의 승산기에 의한 지연스케쥴링 방법은 다음과 같다.

Latency가 2이므로, 지연시간이 2-싸이클타임인 비파이프라인형 승산기로 '*' 연산을 구현하는 경우와 동일한 방법으로 지연스케쥴링을 행할 수 있다. 따라서, 지연시간이 3-싸이클타임인 '*' 연산을 지연시간이 2-싸이클타임인 '*' 연산으로 대치시킨 DFG 모델에 대해 비파이프라인형 승산기로 구현되는 지연스케쥴링을 행한다. 이는 앞서 설명한 비파이프라인형 연산자를 사용하는 경우와 동일한 경우이므로, 스케쥴링 결과 역시 앞의 경우와 동일하다. 단지, 지

연시간이 2-싸이클타임인 '*' 연산이 지연시간이 3-싸이클타임인 연산으로 대치되는 것만이 다를 뿐이다.

Ⅲ. 실험 및 결과

본 논문에서 제안한 스케쥴링 기법의 타당성을 입증하기 위하여, benchmark 모델에 대한 스케쥴링을 행하였다. 실험의 대상인 benchmark 모델로서는 1988 High-Level Synthesis Workshop에서 표준 benchmark 모델로 채택된 fifth-order digital wave filter[11]를 택하였으며, 결과의 비교대상으로서는 IP 스케쥴링 기법을 사용한 ALPS 시스템[10]의 스케쥴링 결과를 택하였으며, 그 결과를 최적치로 설정하였다.

모든 IP 수식은 SUN SPARC-Ⅱ의 SunOS 하에서 GINO[12]및 LINGO[13] Optimizer를 이용하여 해를 구하였다. 실험결과는 표1과 같다(표의 '*' 표시는 본 논문에서 제시한 IP 모델에 의한 스케쥴링 결과를 나타내는 행이다). Fifth-order digital wave filter를 실험모델로 선택한 대부분의 논문에서와 마찬가지로, '+'연산의 지연시간은 40nS, '*'연산의 지연시간은 80nS, 제어스텝의 시간간격은 50nS인 것으로 간주하였다.

실험방식으로서는 자원의 제약이 가해지지 않은 상태에서의 성능제약 스케쥴링을 행한 후, 이로부터 얻어지는 결과로 부터 점진적으로 자원의 수를 줄여나가는 방식을 택하였다.

표 1. 비파이프라인형 데이타패스
Table. 1. Nonpipelined datapath.

		비파이프라인형 승산기					파이프라인형 승산기			
소요자원		3+ 3*	3+ 2*	2+ 2*	2+ 1*	1+ 1*	3+ 2*	2+ 2*	2+ 1*	1+ 1*
지연 시간	최적치	17	n/a	18	21	28	17	18	19	28
	*	17	18	19	21	28	17	18	19	28

거의 모든 경우에 공히 최적의 값과 동일함을 알 수 있다. 따라서, 산출된 지연시간을 최하한 값으로 하는 성능제약 스케쥴링을 행하게 되면, 최적치와 실험치와의 편차가 보정된 최적의 스케쥴링 결과를 얻을 수 있다.

Ⅳ. 결 론

본 논문에서는 자원의 제약조건 하에서의 데이타패스 스케쥴링을 위한 새로운 IP 모델을 제안하였다. 특히, 본 논문에서는 임의의 스케쥴링 방식으로 부터 구해진 스케쥴링 결과로 부터, 보다 적은 자원에 의한 스케쥴링 결과를 얻기 위한, 지연스케쥴링 이론을 제시하였다.

제안된 IP 모델에는 멀티사이클 연산의 스케쥴링을 위한 가상연산자 개념의 IP 모델과, 구조적 파이프라이닝 스케쥴링을 위한 IP 모델이 포함되었다.

본 논문에서 제시한 IP 모델의 변수는 $O(n)$ 형태로 증가하고, 생성되는 방정식은 $O(s \times m)$로 증가하기 때문에, 복잡도는 기존의 IP 모델에 비해 현저하게 감소한다.

본 논문에서 제시한 IP 모델에 스케쥴링 성능을 평가하기 위하여, 표준 benchmark 모델인 fifth-order digital wave filter에 대한 스케쥴링을 행한 결과, 최적치와 동일하거나, 또는, 거의 최적치에 접근된 결과를 얻을 수 있었다.

참 고 문 헌

1. M.C.McFarland, A.C.Parker, "Tutorial on High-Level Synthesis," Proc.of 25th DA Conf., pp.330-336, June 1988.
2. C.Tseng, D.P.Siewiorek, "Automated Synthesis of Datapath in Digital System," IEEE Tr.CAD, Vol.CAD-5, pp.379-385, July 1986.
3. C.H.Gebotys, M.I.Elmasry, "A VLSI Methodology with Testability Constraints," Proc. 1987 Canadian Conf., pp.271-277, July 1986.
4. S.Y.Kung, H.J.Whitehouse, T.Kailath, VLSI and Modern Signal Processing. Englewood Cliffs, NJ : Prentice Hall, pp.258-264, 1985.
5. E.F.Girczyc, "An ADA to Standard Cell Hardware Compiler based on Graph Grammers and Scheduling," Proc.of ICCD-84, pp.726-731, Oct. 1984.
6. A.C.Parker et al., "MAHA : A Program for Datapath Synthesis," Proc.of 23rd DA Conf., pp.461-466, 1986.
7. P.G.Paulin et al, "HAL : A Multi-Paradigm

Approach to Automatic Datapath Synthesis," Proc. of 23rd DAC., pp.263-270, 1986.
8. J.H.Lee, Y.C.Hsu, Y.L.Lin, "A New Integer Programming Formulation for the Scheduling Problem in Datapath Synthesis," -, pp.20-23, 1989.
9. C.T.Hwang, Y.C.Hsu, Y.L.Lin, "Optimum and Heuristic Datapath scheduling under Resource Constraints," Proc.of 27th DA Conf., pp.65-70, 1990.
10. C.T.Hwang, et al., "A formal approach to the scheduling problem in high-level synthesis," IEEE Tr.CAD, vol.10, no.4, pp.464-475, 1991.
11. S.Y.Kung, H.J.Whitehouse, T.Kailath, VLSI and Modern Signal Processing, Englewood Cliffs, NJ : Prentice Hall, pp.258-264, 1985.
12. "GINO : General Interactive Optimizer," LINDO Systems Inc., 1991.
13. "LINGO : Optimization Modeling Language," LINDO Systems Inc., 1991.
14. M.C.McFarland, "Using Bottom-Up Design Techniques in the Synthesis of Digital Hardware from Abstract Behavioral Descriptions," Proc.of 23rd DA Conf., pp.474-480, June 1986.

李近萬(Keun Man YI) 正會員

1949年 9月 18日生

1973년 2월 : 한양대학교 전자공학과 졸업

1980년 : 한양대학교 대학원 전자공학과 졸업 공학석사학위 취득

1992년 4월~현재 : 한양대학교 대학원 전자공학과 박사과정 재학중

1992년~현재 : 청주대학교 전자공학과 전자공학과 부교수

※주관심분야는 VLSI 설계 등임.

林寅七(In Chil LIM) 正會員

1962년 : 한양대학교 공과대학 졸업

1970년 : 와세다대학 전자공학과 박사학위 취득, 일본 후지쓰(주) 정보처리시스템연구소 연구원, 한국과학기술원 대우 교수 University of Illinois at Urbana-Champaign의 Computer Science학과 Visiting Professor역임.

1964년 : 한양대학교 강사, 조교수, 부교수.

1977년~현재 : 한양대학교 교수 재직, 현재 동대학교 전자계산소 소장.

IEEE Computer Society Korea Chapter Chairman.

※주연구분야는 Computer Architecture, RISC Processor 및 Compiler설계, VLSI Testing / Testable Design, Simulation, Layout, AI기법을 이용한 VLSI 설계 및 Machine-Translation등임.