데이터패스 합성에서의 버스와 레지스터의 최적화 기법

신 관 호† · 이 근 만††

요 약

본 논문은 데이터패스 합성에서의 버스 스케듈링 문제와 레지스터의 최적화 방법을 다룬 것이다.

스케듈링은 DFG(Data Flow Graph)의 연산을 제어스텝(control step)에 할당하는 과정으로서, 주어진 조건을 만족하는 범위 내에서 비용함수(cost function)의 최소화에 목적을 둔다. 이를 위해 본 논문에서는, 연산자 배치를 위한 하드웨어 할당(hardware allocation) 과정에서의 설계비용을 최소화시키기 위해, 연산결과를 저장하는 레지스터(register)와 연산간의 이동통로인 버스(bus)의 최적화 기법을 논하였다. 특히, 하드웨어 할당과정의 중요한 과제인 버스와 레지스터의 최소화 기법을 논하였으며, 레지스터의 최적화는 스케듈링이 완료된 후의 결과를 이용하였다.

실험대상으로는 벤치마크 모델인 5차 디지털 웨이브필터(5th-order digital wave filter)를 사용하였으며, 본 논문의 결과를 기존결과와 비교함으로서, 본 논문의 효용성을 입증하였다.

모든 실험결과는 구조적형태의 선형정수계획법(ILP:Integer Linear Programming)을 이용함으로서, 모든 경우에 언제나 최적의 결과를 얻을 수 있도록 하였다.

Bus and Register Optimization in Datapath Synthesis

Kwan-Ho Shinn† · Keun-Man Yi††

ABSTRACT

This paper describes the bus scheduling problem and register optimization method in datapath synthesis.

Scheduling is a process of operation allocation to control steps in order to minimize the cost function under the given circumstances. For that purpose, we propose some formulations to minimize the cost function for bus assignment to get an optimal and minimal cost function in hardware allocations. Especially, bus and register minimization technique are fully considered which are the essential topics in hardware allocation. Register scheduling is done after the operation and bus scheduling.

Experiments are done with the DFG model of fifth-order digital wave filter to show its effectiveness.

Structural integer programming formulations are used to solve the scheduling problems in order to get the optimal scheduling results in the integer linear programming environment.

1. 서 론

디지털 시스템 자동설계 과정의 중요한 영역 중의 하나인 데이터패스 합성(datapath synthesis)은 스케듈링(scheduling)과 하드웨어 할당(hardware allocation), 모듈 바인딩(module binding)으로 이루어지며, 이들 중, 스케듈링과 하드웨어 할당이 주된 관심 분야이다.

스케듈링의 목적은 DFG(Data Flow Graph)상의 연산을 주어진 조건이 만족하는 제어스텝에 할당하는 과

† 정 회 원 : 제2건국위원회 위원, 대한변리사회 회장
†† 정 회 원 : 청주대학교 전자공학과 교수
논문접수 : 1997년 9월 24일, 심사완료 : 1997년 12월 1일

정으로서, 스케듈링 알고리즘은 스케듈링 방식에 따라, 반복/구축(iterative/constructive)형 스케듈링[1,2]과 변형(transformation)형 스케듈링[3,4]으로 구분된다. 변형형 알고리즘은 직렬과 병렬변환을 반복적으로 수행하여 주어진 사양에 맞는 설계를 점차적으로 구하는 방식이고, 반복/구축형 방식의 알고리즘은 정해진 우선순위 함수에 따라 한번에 하나씩 스케듈링을 행하는 과정을 반복하는 방법이다.

또한, 스케듈링은 스케듈링의 목적에 따라, 성능제약 스케듈링[2]과 자원제약 스케듈링[1]으로 구분된다. 성능제약 스케듈링은 주어진 성능하에서 최소의 자원으로 이루어지는 스케듈링을 말한다. 여기서의 자원은 데이터패스 합성 시 요구되는 연산자, 버스 및 레지스터의 구현에 소요되는 모든 비용을 의미한다. 자원제약 스케듈링은 가용자원의 범위 내에서 최대의 성능을 얻기 위한 스케듈링 방법을 의미한다.

또한, 스케듈링은 연산자의 연산유형 및 데이터패스의 여러 조건에 따라, 크게 파이프라인 스케듈링[7]과 비파이프라인 스케듈링[2]으로 구분된다.

이들 기존의 스케듈링 방법은, 스케듈링 단계에서 연산의 최적화에 우선순위를 두어 연산의 최적화를 위한 스케듈링을 행하였으며, 또한, 선형정수계획법(ILP: Integer Linear Programming)에 의해 스케듈링 문제를 풀고자 한 기존의 논문에서는 파이프라인 스케듈링[7,9]이나 조건부 자원공유의 스케듈링[1,2,11,15] 및 Loop Folding[6,13,14,15]등 만을 다루었다. 이러한 기존의 선형정수계획법 스케듈링 방법[7,12,16]은, 첫째, 정수계획법을 비선형적인 형태로 기술한 후, 이를 선형적인 형태로 변환하기 때문에 수식이 난해함은 물론 수식의 생성과정이 복잡하고, 둘째, 정수계획법의 형태가 비구조적(nonstructural)이기 때문에 모델의 규모가 큰 경우에는 수식을 생성하고 분석하거나 변형하는데에 상당한 시간이 소요되며, 셋째, 비용제약 스케듈링의 경우, 전 스케듈링 단계에 걸쳐 자원의 제약조건이 불필요하게 중복되므로, 스케듈링의 효율이 좋지 않다는 단점을 갖고 있다.

따라서, 본 논문에서는 이러한 단점의 보완과 아울러, 데이터전송에 사용되는 버스와 래지스터의 비용을 감안한, 연산과 버스의 가용자원에 제약이 가해진 상태에서의 스케듈링 문제를 다루고자 한다.

제2장에서는 데이터패스 스케듈링의 기본이론을 다루었고, 제3장에서는 버스와 레지스터의 스케듈링을 논하였으며, 제4장에서는 버스의 최적화 방법에 대해 기술하였다. 실험결과 및 고찰은 제5장에서, 결론은 제6장에서 다루었다.

2. 데이터패스 스케듈링

스케듈링은 DFG 상의 연산을 한번에 하나씩 제어스텝에 할당하기 때문에, 임의의 연산이 할당되고 나면 이후의 연산은 이미 할당된 연산의 결과에 전적으로 의존하게 되므로, 이와 같은 방식의 스케듈링 결과는 결코 최적화된 것이라고 할 수 없다. 이러한 문제를 해결할 수 있는 방법으로서는 제약조건을 개별적인 변수로 간주하여 이들 변수사이의 관계에 따라 스케듈링 문제를 다원 1차방정식으로 전개하여 이로부터 해를 얻고자하는 ILP 표현법이 이용된다. 따라서, 본 논문에서 다룰 레지스터 공유 문제를 ILP 표현법으로 표기하여 최적의 결과를 얻도록 하였다.

스케듈링 문제에 대한 ILP 표현법은, 기본적으로, 연산간의 선후관계(precedence relation)식, 범위관계(range relation)식, 자원관계(resource relation)식 및 시간관계(time relation)식과 같은 4개의 기본 관계식으로 기술된다.

이후의 수식 표현에 사용되는 기호의 정의는 다음과 같다.

n_t : 연산유형이 t인 연산의 개수.

t_T : 싸이클타임(cycle time)－제어스텝의 시간 간격(time slot)

d_i : 연산 O_i의 지연시간

D_i : 싸이클타임으로 표시되는 연산 O_i의 수행 시간

S_i, L_i : ASAP 스케듈링과 ALAP 스케듈링에 의해 결정되는 연산 O_i의 상한 및 하한 제어스텝

$O_i(k)$: 연산 O_i의 k번째 요소(element)

[$O_i = \{O_i(1), O_i(2), \cdots, O_i(r_i)\}$]

p : 연산 $O_i(k)$가 할당되는 제어스텝

($p = S_i + K - 1$)

r_i : 연산 O_i의 스케듈링 범위

($r_i = L_i - S_i + 1$)

$X_i(k)$: 연산 O_i의 k－번째 요소가 제어스텝－p_{ik}에 할당되었을 때, 그 값이 1이 되는 0－1 정수변수 (0－1 integer variable)

$B_p(k)$: 제어스텝 p에 할당된 연산 $O_i(k)$의 입력

$V_{ip}(k)$: 제어스텝 p에 할당된 연산 $O_i(k)$의 입력변수

$LT_i(j)$: 연산 O_i와 연산 O_j 사이에서 연산 O_i의 출력변수가 저장되는 레지스터의 라이프타임

T_i : 연산 O_i가 할당되는 제어스텝

2.1 선후관계식

연산간의 데이터 의존관계를 나타내는 선후관계식은, 연산 O_i의 최근접 후행연산을 O_j라 할 때, 연산 O_i와 연산 O_j 사이에는 다음의 관계식 (1)이 성립한다.

$$\sum_{p=S_i}^{L_i}[p\times X_i(p-S_i+1)] - \sum_{q=S_j}^{L_j}[q\times X_j(q-S_j+1)] \le -D_i \quad (1)$$

관계식 (1-1)은 연산 O_j의 각 요소가 할당될 수 있는 제어스텝이, 연산 O_i가 할당된 제어스텝 보다도 최소한 연산 O_i의 수행시간인 D_i-싸이클타임 이상 이격된 제어스텝 이어야만 한다는 것을 의미한다.

2.2 범위관계식

연산이 할당될 수 있는 스케듈링 범위를 나타내는 것으로서, ASAP/ALAP 스케듈링에 의해 산출된 연산의 스케듈링 범위 내에서, 연산 O_i는 상한 제어스텝 S_i와 하한 제어스텝 L_i의 범위에 걸쳐 오직 한번만 할당되어야 하기 때문에, 연산의 각 요소 사이에는 다음의 관계식 (2)가 성립한다.

$$\sum_{k=1}^{r_i} X_i(k) = 1, \quad (i=1,2,\cdots,n_t) \quad (2)$$

2.3 자원관계식

특정 제어스텝에 할당될 수 있는 연산의 개수를 나타내는 자원관계식은, 제어스텝-p에 할당될 수 있는 연산유형 t인 연산의 총 개수는 사용가능한 자원의 개수 N_t를 초과해서는 안된다.

$$\sum_{i=1}^{n_t} X_i(p-S_i+1) \le N_t, \quad (p=1,2,\cdots,T_{max}-d_i+1) \quad (3)$$

2.4 시간관계식

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} 사이에는 다음의 관계식 (4)가 성립한다.

$$\sum_{p=S_i}^{L_i}(p\times X_i(p-S_i+1)) \le T_{max} - D_i + 1, \quad (i=1,2,\cdots,n_t) \quad (4)$$

3. 버스와 레지스터

연산간의 데이터 전송에 사용되는 상호연결선(interconnection)과 저장장치(storage device)는 스케듈링 과정에서 필히 고려하여야할 사항으로서, 상호연결선은 와이어(wire), 멀티플렉서(multiplexer) 및 버스(bus)로, 저장장치는 레지스터, ROM 또는 RWM 등으로 구성된다. 자원공유(resource sharing)가 별로 필요치 않은 소규모 시스템에서는 단지 와이어 만을 사용하여 변수를 전달할 수 있으나, 규모가 큰 시스템에서는 버스와 멀티플렉서를 이용하여 연산간의 데이터 전송을 행한다.

임의의 제어스텝(p-1)에서 제어스텝 p로 이동하는 연산 $O_i(k)$의 각기 다른 입력을 $B_{ip}(k)$라 할 때, 제어스텝 p에서 요구되는 버스의 개수 N_{bus}는 그 스텝에 할당된 모든 연산의 입력변수 합과 같아야만 한다. 따라서, 스케듈링 과정에서 요구되는 버스의 개수는 다음의 관계식 (5)를 만족하여야만 한다.

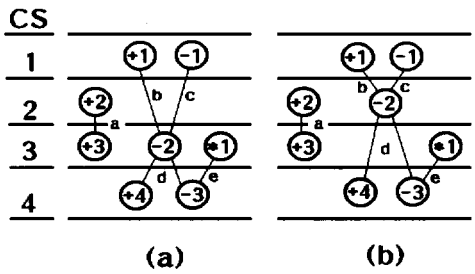
$$\sum_{i=1}^{n_t} B_{ip}(k) \le N_{bus}, \quad (p=1,2,\cdots,T_{max}-D_i+1) \quad (5)$$

입력 $B_{ip}(k)$에 할당된 변수 $v_{ip}(k)$는 하나 또는 복수개의 제어스텝을 경유하여 연산 $O_i(k)$의 입력으로 사용된다. 변수 $v_{ip}(k)$가 단지 하나의 제어스텝 만을 경유한 후, 그 다음 제어스텝에서 연산 $O_i(k)$의 입력으로 사용되는 경우, 해당 변수는 반드시 하나의 버스를 할당받아야만 한다. 그러나, 여러개의 제어스텝을 경유할 경우, 변수는 연산 $O_i(k)$의 입력으로 사용되는 제어스텝에 도달하기 전까지 임의의 레지스터에 저장된다.

(그림 1)은 스케듈링이 완료된 DFG의 일부분을 나타낸 것이다. (그림 1)의 (a)와 (b)는 동일한 알고리즘을 수행하는 DFG 모델로, 요구되는 연산자의 개수는 ALU 2개, 승산기 1개로서 요구되는 연산자의 개수는 동일하다. 그러나, 연산 간의 변수이동을 고려할 경우, (그림 (a))는 제어스텝-2로부터 제어스텝-3으로 이동하는 데이터의 개수가 3개인 반면, (그림 (b))에서는 전 구간에 걸쳐 변수이동이 2개로 균일하게 분포되어 있다.

스케듈링 단계에서 연산과 버스의 최적화와 함께 고려할 사항으로, 데이터 저장을 위한 레지스터의 최적

화를 들 수 있다. 연산의 입력으로 사용되는 변수의 임시 저장장소인 레지스터에는 각기 다른 데이터가 하나 또는 여러 제어스텝에 걸쳐 기억된다. 임의의 제어스텝에서 연산의 수행이 완료되면 후행연산에 그 결과를 전달하기 위한 변수가 생성되며, 그 변수가 후행연산의 입력으로 사용된 후 소멸하게 된다. 이 기간을 변수의 라이프타임(life time)이라 부른다.



(그림 1) DFG 모델

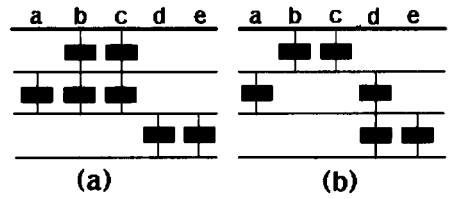
회로가 요구하는 레지스터의 수는, 변수의 이동이 가장 많은 제어스텝에서의 변수합으로 구해진다. 그러므로, 레지스터의 최적화를 이루기 위해서는 DFG 상의 연산을 제어스텝에 균일하게 분포시킬 필요가 있다.

임의의 변수는 라이프타임 동안 반드시 하나의 레지스터를 점유하여야만 한다. 또한, 레지스터는 변수의 라이프타임이 중복되지 않는 범위 내에서는 다수의 변수가 공유할 수 있다.

변수는 최소한 하나 또는 여러 제어스텝에 걸쳐 존재하게 되므로, 연산 O_i의 출력인 임의의 변수가 연산 O_j의 입력으로 사용되기 까지의 시간간격인 변수의 라이프타임 LT는 다음의 관계식 (6)으로 표현된다.

$$LT_I(j) = Max(L_i - S_j + D_i - 1) \qquad (6)$$

(그림 2)는 (그림 1)의 DFG 모델에 대한 변수의 라이프타임을 나타낸 것이다. (그림 1)의 (a)와 (b)는 소요되는 자원의 갯수와 제어스텝의 수는 동일하나 요구되는 레지스터의 갯수는 서로 다르다. 즉, '-2' 연산이 제어스텝-2나, 또는 제어스텝-3 어느쪽에 할당되는가에 따라 변수의 라이프타임이 달라지게 된다. (그림 1(a))에서는, 연산 '+1'과 연산 '-1'의 출력이 연산 '-2'의 입력이 되기 까지 2개의 제어스텝을 경유하여야만 한다. 그러므로, 변수 'b'와 'c'가 연산'-2'의 입력으로 사용되기 까지, 라이프타임의 합은 4-싸이클타임이 된다. 그러나, (그림 1(b))에서는, 변수 'b'와 'c'가 연산 '-2'로 이동하기 위해서는 단 하나의 제어스텝을 경유하므로 라이프타임의 합은 2-싸이클타임이 된다. 따라서, (그림 1)의 'a'와 'b'에서, 연산 '-2'의 선행연산과 후행연산의 라이프타임의 합은 각각 5-싸이클타임과 6-싸이클타임이 되며, 전체적인 라이프타임의 합은 (그림 2)와 같이 각각 7-싸이클타임과 6-싸이클타임이 된다.



(그림 2) 변수의 라이프타임

<표 1>과 <표 2>는 (그림 2)의 변수에 대한 레지스터 할당과 자원공유를 나타낸 것이다. 라이프타임이 중복되지 않을 경우, 임의의 변수는 또 다른 변수와 동일한 레지스터를 공유할 수 있으므로, <표 1>과 <표 2>에서 요구되는 레지스터의 개수는 각각 3개와 2개가 된다. 따라서, (그림 1(a))의 스케듈링 결과 보다는 (그림 1(b))의 스케듈링 결과가 보다 이상적임을 알 수 있다.

이와같이, 동일한 자원이 요구되는 스케듈링 결과를 얻었다 하더러도, 특정 연산이 할당되는 제어스텝의 위치에 따라, 라이프타임 변화로 인한 버스와 레지스터의 개수가 달라지게 된다. 따라서, 스케듈링 과정에서는 버스와 레지스터를 동시적으로 고려한 스케듈링이 요구된다.

〈표 1〉 (그림 1(a))의 레지스터 할당

레지스터	변 수
1	a
2	b, d
3	c, e

〈표 2〉 (그림 1(b))의 레지스터 할당

레지스터	변 수
1	a, b, e
2	c, d

4. 버스의 최적화

스케듈링 과정으로부터 연산의 최적 스케듈링 결과를 얻었다 할지라도, 특정 제어스텝에 할당된 연산의 개수에 따라, 요구되는 버스와 레지스터의 개수가 달라지게 된다. 따라서, 스케듈링의 제약조건을 연산의 최적화에 국한시키지 않고, 연산의 할당에 따른 버스 및 레지스터의 최적화를 고려하여 스케듈링을 행하여야만 한다.

임의의 제어스텝 p에 할당된 연산 O_i의 입력은 언제나 2개이므로, 제어스텝 p에서 요구되는 버스의 개수는, 해당 제어스텝에 할당된 모든 연산의 2배수로 표현된다. 그러므로, 제어스텝 p에서, 변수의 이동에 필요한 버스의 개수는 다음의 관계식 (7)을 만족하여야만 한다.

$$2\times \sum_{i=1}^{n_i} X_i(p-S_i+1) \le N_{bus},\ (p=1,2,\cdots,\ T_{max}-D_i+1) \quad (7)$$

또한, 변수는 최소한 하나 또는 여러 제어스텝에 걸쳐 존재하게 되므로, 연산 O_i의 출력인 임의의 변수가 연산 O_j의 입력으로 사용되기 까지의 시간간격인 변수의 라이프타임은 다음의 관계식 (8)로 표현된다.

$$LT_i(j) = Max(L_j - S_i + D_i - 1) \quad (8)$$

(그림 3)은 26개의 가산기와 8개의 승산기로 구성된 5차 디지털 웨이브 필터의 DFG 모델을 나타낸 것이다 (각 연산의 좌우측 숫자는 각각 ASAP(As Soon As Possible) 스케듈링과 ALAP(As Last As Possible) 스케듈링에 의해 산출된 스케듈링 범위를 의미한다). 그림에서 LT 값이 가장 큰 변수는 '+9'의 출력이며, LT 값이 가장 작은 변수는 '+1'의 출력과 같이, 경우하는 제어스텝 수가 하나뿐인 변수이다. 관계식 (8)에서 '+' 연산의 수행시간을 1로 설정하였을 때, (그림 3)의 연산에 대한 LT의 최소값과 최대값은 다음과 같다.

$$LT_{+1}(+3) = 2-1+(1-1) = 1$$

$$LT_{+9}(+11) = 17-8+(1-1) = 9$$

관계식 (6)에서 기술한 변수의 라이프타임인 LT와 관계식 (1)의 선후관계식을 이용하면 변수의 라이프타임 최소화를 위한 다음의 관계식 (9)를 얻을 수 있다.

$$\sum_{p=S_i}^{L_i}[p\times X_i(p-S_i+1)] - \sum_{q=S_j}^{L_j}[q\times X_j(q-S_j+1)] + LT_i(j) = -D_i \quad (9)$$

스케듈링 단계에서의 가장 중요한 문제는 연산과 버스의 최적화에 있으므로, 레지스터의 최적화로 인하여 연산과 버스의 소요자원이 증가되어서는 안된다.

<표 3>과 <표 4>는 각각 (그림 3)의 5차 디지털 웨이브 필터에 대한 버스의 자원관계식과 레지스터의 라이프타임의 관계식을 나열한 것이다.

〈표 3〉 버스의 자원관계식

$2\times\{X_{+01}(1)+X_{+02}(1)+X_{+09}(11)+X_{+11}(10)+X_{+19}(2)+X_{+21}(4)+X_{+22}(4)+X_{+23}(3)+X_{+24}(3)+X_{+25}(3)+X_{+26}(2)\} \le N_{bus}$;
$2\times\{X_{+02}(5)+X_{+04}(3)+X_{+05}(2)+X_{+01}(1)+X_{+02}(1)\} \le N_{bus}$;
$2\times\{X_{+05}(2)+X_{+07}(2)+X_{+08}(1)+X_{+09}(1)+X_{+10}(1)+X_{+01}(3)+X_{+02}(3)\} \le N_{bus}$;
$2\times\{X_{+05}(3)+X_{+07}(3)+X_{+08}(2)+X_{+09}(2)+X_{+10}(2)+X_{+11}(1)+X_{+03}(1)+X_{+04}(1)\} \le N_{bus}$;
$2\times\{X_{+08}(3)+X_{+09}(3)+X_{+10}(3)+X_{+11}(2)+X_{+03}(2)+X_{+04}(2)\} \le N_{bus}$;
$2\times\{X_{+09}(8)+X_{+11}(7)+X_{+14}(4)+X_{+17}(4)+X_{+18}(3)+X_{+20}(3)+X_{+21}(1)+X_{+05}(3)+X_{+06}(3)+X_{+07}(2)+X_{+08}(2)\} \le N_{bus}$;
$2\times\{X_{+06}(5)\} \le N_{bus}$;

〈표 4〉 레지스터의 라이프타임 최소화 관계식

$X_{+1}(1)-X_{+3}(1)+LT_{+1}(+3) = -1$;
$3X_{+2}(1)-6X_{+7}(1)-7X_{+7}(2)-8X_{+7}(3)-9X_{+7}(4)+LT_{+2}(+7) = -1$;
$13X_{+5}(1)+14X_{+5}(2)-15X_{+21}(1)-16X_{+21}(2)+LT_{+5}(+21) = -2$;
$4X_{+5}(1)-8X_{+9}(1)-9X_{+9}(2)-10X_{+9}(3)-11X_{+9}(4)-12X_{+9}(5)-13X_{+9}(6)-14X_{+9}(7)-15X_{+9}(8)-16X_{+9}(9)+LT_{+5}(+9) = -1$;
$7X_{+7}(1)+8X_{+7}(2)+9X_{+7}(3)-9X_{+11}(1)-10X_{+11}(2)-11X_{+11}(3)-12X_{+11}(4)-13X_{+11}(5)-14X_{+11}(6)-15X_{+11}(7)-16X_{+11}(8)-17X_{+11}(9)+LT_{+7}(+11) = -1$;
$13X_{+6}(1)+14X_{+6}(2)+15X_{+6}(3)-15X_{+22}(1)-16X_{+22}(2)-17X_{+22}(3)+LT_{+6}(+22) = -2$;

(그림 4)는 멀티싸이클 연산을 파이프라인형 연산자[8]로 구현할 경우의 스케듈링 결과이다. 그림의 우측 수치는 해당 제어스텝에서 요구되는 승산기, 가산기, 버스 및 레지스터의 개수를 각각 나타낸 것이다.

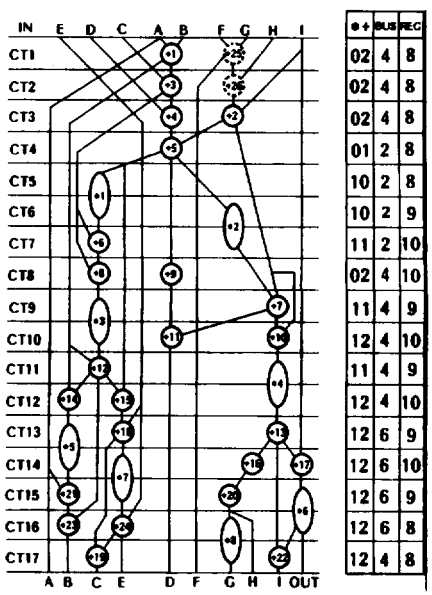
5. 실험 및 고찰

본 논문에서 제안한 스케듈링 방법의 효율성을 검증하기 위해, 벤치마크 모델(benchmark model)에 대한 스케듈링을 행하였다. 사용된 벤치마크 모델로서는 1988 High-Level synthesis workshop에서 표준 벤치마크 모델로 채택된 (그림 3)의 5차 디지털 웨이브 필터를 택하였다.

실험결과는 공개된 자료가 풍부하고 스케듈링 성능이 우수한 것으로 판정된 ALPS 시스템[7]과Spaid 시스템의 결과와 비교하였다.

모든 ILP 수식는, branch-and-bound 방법을 사용함으로써 최적의 정수 해를 구하는 선형프로그램(linear program)인 LINGO[17] 패케지를 이용하여 해를 구하였다.

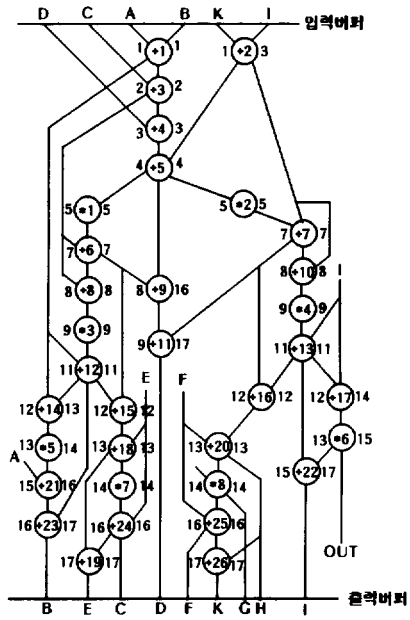
실험결과는 <표 5>와 같으며, 본 논문에서 제시한 방법에 의한 결과를 "TM"이라 표현하였다.



(그림 3) 5차 디지털 웨이브 필터의 DFG 모델

5차 디지털 웨이브 필터를 실험모델로 선택한 대부분의 논문과 마찬가지로 '+' 연산의 지연시간은 40nS, '*' 연산의 지연시간은 80nS, 제어스텝의 시간간격은 50nS인 것으로 간주하였다.

<표 5>로부터 알 수 있듯이, 본 논문의 결과는 ALPS 시스템에서 구현한 결과와 동일한 결과를 얻었으며, Spaid 시스템에서 구현한 결과보다는 더 나은 결과를 얻을 수 있었다.



(그림 4) 스케듈링 결과

〈표 5〉 5차 디지털 웨이브 필터에 대한 스케듈링 결과

시스템 / 소요자원	Spaid	ALPS	TM
가 산 기	3	2	2
승 산 기 (파이프라인형)	2	1	1
버 스	—	6	6
Loop의 길이	17	17	17
DFG 수행시간	—	19	19

6. 결 론

본 논문에서는 데이터패스 스케듈링에서의 비용함수 최소화를 위한 버스와 레지스터의 최적화 문제를 다루었다. 특히, 연산과 버스와의 관계를 고려한 스케듈링 기법과 스케듈링의 결과에 따른 레지스터의 최소화 방법을 제안하였다.

기술된 버스의 최적화 기법의 정확성을 검증하기 위하여 표준 벤치마크 모델인 5차 디지털 웨이브 필터[9]

에 적용시켜 기존의 논문과 비교한 결과, 스케줄링 과정에서 요구되는 버스의 개수가 기존의 논문결과와 일치함으로써, 제시된 모든 ILP 수식이 정확함을 알 수 있었다.

본 논문에서는, 멀티싸이클 연산의 파이프라인형 연산자를 이용할 수 있도록, 연산과 버스의 최적화에 대한 ILP 수식을 기본적으로 다루었다.

본 연구는 스케줄링 과정에서 레지스터의 최소화로 인한 연산과 버스의 개수가 증가되는 것을 피하기 위해 레지스터의 최적화 과정을 별도로 수행함으로써, 연산과 버스 및 레지스터의 스케줄링이 동시에 이루어지지 않았다.

향후 연구과제로서는 연산과 버스 및 레지스터를 동시에 고려한 스케줄링 기법이 요구되며, 스케줄링과 하드웨어 할당을 동시에 수행시킬 수 있는 구조적 ILP 수식표현에 대한 새로운 연구가 필요하다.

참 고 문 헌

[1] C.T.Hwang, Y.C.Hsu, Y.L.Lin, "Optimum and Heuristic Datapath Scheduling under Resource Constraints," in Proc. of 27th DAC, pp.65-70, June. 1990.

[2] J.H.Lee, Y.C.Hsu, Y.L.Lin, "A New Integer Programming Formulation for the Scheduling Problem in datapath Synthesis," in Proc. of ICCAD'89, pp.20-23, Nov. 1989.

[3] P.G.Paulin, J.P.Knight, "Force-directed Scheduling for the Behavioral Synthesis of ASIC's," IEEE Tr.CAD, Vol.8, pp.661-679, June. 1989.

[4] K.S.Hwang, et al., "Scheduling and Hardware sharing in pipelined Datapath," in Proc. of ICCAD'89, pp.24-27, Nov. 1989.

[5] N.Park, A.C.Parker, "SEHWA : A program for Synthesis of Pipelines," in Proc. of 23rd DAC, pp.454-460, June. 1986,

[6] G.Goossens, J.Vandewalle, H. De Man, "Loop Optimization in Register-transfer scheduling for DSP-Systems," in Proc. of 26th DAC, pp.826-831, June. 1989.

[7] 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, April. 1991.

[8] B.M.Pangrle, D.D.Gajski, "State Synthesis and Connectivity Binding for Microarchitecture Compilation," in Proc. of ICCAD'86, pp.210-213, Nov. 1986.

[9] 이근만, "정수계획법을 이용한 상위수준합성의 데이터패스 스케줄링에 관한 연구", 박사학위논문, 1992, 한양대학교.

[10] S.Y.Kung, H.J.Whitehouse, T.Kailath, 'VLSI and Modern Signal Processing,' Englewood Cliffs, NJ: Prentice Hall, pp.258-264, 1985.

[11] M.J.Lim, R.Jain, "Representing Conditional Branches for High-Level Synthesis Applications," in Proc. of 29th DAC, pp.106-111, June. 1992.

[12] C.H.Gebotys, "Optimal Scheduling and Allocation for Embedded VLSI Chips," in Proc. of 29th DAC, pp.116-19, June. 1992.

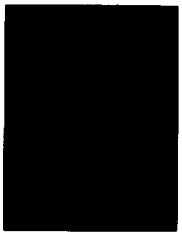
[13] L.F.Chao, A.LaPaugh, "Rotation Scheduling : A Loop Pipelining Algorithm," in Proc. of 30th DAC, pp.566-572, June. 1993.

[14] N.L.Passos, et al., "Loop Pipelining for Scheduling Multi-Dimensional Systems via Rotation," in Proc. of 31st DAC, pp.485-490, June. 1994,

[15] S.Bhattacharya, S.Dey, F. Brglez, "Performance Analysis and Optimization of Schedules for Conditional and Loop-Intensive Specifications," in Proc. of 31st DAC, pp.491-496, June. 1994.

[16] Y.G.DeCastelo, M.Potkonjak, A.C.Parker, "Optimal ILP-based Approach for Throughput Optimization Using Simultaneous Algorothm/Architecture Matching and Retiming," in Proc. of 32nd DAC, pp.113-118, June. 1995.

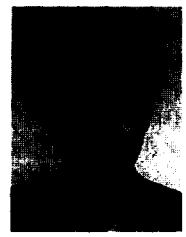
[17] LINGO : 'Optimization Modeling Language,' LINDO Systems Inc., 1991.



신 관 호

e-mail : webmaster@shinnpatent.co.kr

1967년 2월 서울고등학교 졸업(19회)
1971년 2월 한양대학교 공과대학 전자공학과 졸업
1973년 6월 ROTC 9기 육군중위 제대
1973년 6월 중앙특허법률사무소 전자부장근무
1976년 2월 연세대학교 산업대학원 공학석사학위취득
1976년 4월 (주)LG전자(구 금성사) 입사
1979년 12월 제 15회 변리사 시험합격
1982년 7월 (주)LG전자 특허부장승진
1983년 2월 대한검도회 이사
1983년 3월 한양대학교 행정대학원 법학석사학위취득
1986년 3월 신관호특허법률사무소 개업
1990년 2월 대한변리사회 총무이사
1991년 3월 한양대학교 법과대학원 법학박사학위취득
1991년 6월 한국상사중재원 중재위원으로 피선
1992년 2월 대한변리사회 감사
1992년 2월 청주대학교 공과대학원 전자공학과 박사과정입학
1993년 3월 라이센싱 한국협회회장 피선
한양대학교 공업소유권 강의
1994년 2월 대한변리사회 부회장 피선
1994년 6월 AIPPI 한국협회 이사 선임
1994년 12월 WIPO 중재위원회 중재위원으로 피선
1997년 2월 충주대학교 산학협동위원 임명
1997년 2월 특허청 전자출원전문위원 임명
1997년 2월 청주대학교 공과대학원 박사 학위 수여
1998년 2월 대한변리사회 회장 피선
1998년 2월 특허청 정책자문위원 임명
1998년 6월 외교통상부 지적재산권 전문위원 임명
1998년 10월 제 2건국위원회위원 선임
관심분야 : 상위수준 합성시스템의 자원스케듈링



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