충돌수를 이용한 파이프라인 데이타패스 합성 스케줄링 알고리즘

유 동 진[†] · 유 희 진[††] · 박 도 순[†††]

요 약

본 논문은 상위 수준 합성시의 자원 제약 조건하에서 파이프라인 데이타패스 합성을 위한 스케줄링 알고리즘으로, 제안된 휴리스틱 알고리즘은 자원의 충돌수에 근거한 우선순위 함수를 사용한다. 자원 제약하에서 파이프라인 데이타패스 합성을 할 때 자원 충돌을 해결하기 위해 충돌수를 정의하고 충돌수, 자원 갯수, 그리고 연산의 모빌리티를 고려한 우선순위 함수를 정의하여 스케줄링 한다. 제안 알고리즘은 실질적인 하드웨어 설계를 위해 체이닝, 멀티사이클링, 구조적 파이프라인이 지원되도록 한다. 제안 알고리즘에 의한 16 포인트 FIR 필터와 5차 엘립틱 웨이브 필터 합성 결과에 의해 다른 시스템들과의 성능을 비교하였으며, 대부분의 경우에 최적의 해를 찾을 수 있었다.

A Scheduling Algorithm for the Synthesis of a Pipelined Datapath using Collision Count

Dong-Jin Yu[†] · Hee-Jin Yoo[††] · Do-Soon Park[†††]

ABSTRACT

As this paper is a scheduling algorithm for the synthesis of a pipelined datapath under resource constraints in high level synthesis, the proposed heuristic algorithm uses a priority function based on the collision count of resources. In order to schedule the pipelined datapath under resource constraints, we define the collision count and the priority function based on the collision count, a number of resource, and the mobility of operations to resolve a resource collision. The proposed algorithm supports chaining, multicycling, and structural pipelining to design the realistic hardware. The evaluation of the performance is compared with other systems using the results of the synthesis for a 16 point FIR filter and a 5th order elliptic wave filter, where in most cases, the optimal solution is obtained.

1. 서 론

파이프라인은 일련의 입력을 병렬 처리하는 루프 구조로서 반복적인 데이타 처리를 하는 디지털 신호처리, 화상처리등에서 효과적인 방법이다. 입력은 일정한 시간 간격 또는 가변적인 시간 간격으로 파이프라인에 공급되는데 전자를 고정 데이타 입력 간격, 후자를 가변 데이타 입력 간격이라 한다. 이상적인 경우의 데이타 입력 간격은 1 클럭 사이클이다. 즉 매 클럭 사이클마다 데이타가 파이프라인에 입력되고, 파이프라인에 데이타가 모두 채워지면 매 클럭 사이클마다 출력이 발생하여 파이프라인의 성능을 최대화한다.

† 정 회 원 : 현대정보기술 연구소
†† 정 회 원 : 전주대학교 컴퓨터공학과 객원교수
††† 종신회원 : 홍익대학교 컴퓨터공학과 교수

논문접수 : 1998년 2월 19일, 심사완료 : 1998년 8월 28일

설계 자동화에서 상위 수준 합성(HLS: High Level Synthesis)은 알고리즘 수준의 하드웨어 동작 기술을 입력으로 받아들여 레지스터 전송 수준에서 데이타패스와 컨트롤러로 구성되는 하드웨어를 설계하는 과정이다. 상위 수준 합성에서 스케쥴링이란 하드웨어 동작 연산들을 주어진 제약조건을 만족하는 범위내에서 최적의 제어단계에 할당하는 과정을 말한다. 파이프라인 스케쥴링은 연산 장치의 파이프라인을 허용하는 구조적 파이프라인과 그렇지 않은 기능적 파이프라인으로 구분되는데, 기능적 파이프라인은 동작기술을 나타내는 DFG(Data Flow Graph)를 동시에 실행될 수 있는 스테이지로 나누어야 한다. 비파이프라인 스케쥴링은 시간제약 또는 면적 제약 조건하에서 각각 시스템의 면적과 성능을 최적화하지만, 파이프라인 스케쥴링은 합성되는 하드웨어 성능에 중요한 변수가 되는 데이타 입력 간격을 고려해야 한다. 시간 제약 파이프라인 스케쥴링은 제어단계의 수를 제한하고 데이타 입력 간격과 면적을 최적화한다. 자원 제약 파이프라인 스케쥴링은 연산 모듈의 수 또는 연산 모듈의 면적을 제한하고 데이타 입력 간격과 제어단계를 최적화한다. 그러나 기능적 파이프라인 스케쥴링은 NP complete[1]이므로 최적의 해를 찾기위해 리스트 스케쥴링, 확률기반 스케쥴링등 여러 휴리스틱 알고리즘이 개발되어 왔다.

Sehwa[2]는 우선순위 함수로 순방향 긴급도 또는 역방향 긴급도를 사용하는 리스트 스케쥴링인데, 순방향 긴급도는 노드로부터 출력까지의 가장 긴 길의 길이로 정의하고 역방향 긴급도는 입력으로부터 노드까지의 가장 긴 길의 길이로 정의되며, 스케쥴링된 연산의 수가 자원의 수를 넘으면 긴급도가 낮은 연산을 지연시킨다. HAL 시스템[3,4]은 확률기반 시스템이며 연산의 병행정도를 측정하기 위해 훅크의 법칙에 의한 힘을 이용하는데, 시간 제약 조건에서는 FDS를, 자원 제약 조건에서는 우선순위 함수가 힘인 FDLS를 사용하여 지연시키기 위한 연산을 선택하여 파이프라인 스케쥴링을 한다. IFDS[5]는 FDS의 시간 복잡도를 감소시키기 위한 연구로 각 연산의 할당 가능한 제어단계 중에서 ASAP과 ALAP 제어단계에 그 연산을 각각 고정하였을때 힘의 차이가 최대인 연산을 선택하여 힘값에 따라 ASAP 제어단계, 또는 ALAP 제어단계를 하나씩 삭제하는 방법을 사용하였다. Choi[6]는 임계경로를 조사하여 자원제약에 의한 충돌이 없도록 지연 연산자를 미리 삽입한 후에 연산의 종류별로 연산의 분산정도와 자원의 사용정도에 근거한 우선순위 함수를 사용하여 스케쥴될 연산의 종류를 먼저 결정하고, 선택된 연산 종류의 밀집도가 최소인 파티션을 선정하여 그 파티션에 할당 가능한 연산들중에서 힘값을 최소로 하는 연산을 선택하는 방법이다. EBS[7][8][9]는 연산의 균등분포를 측정하는 확률기반의 엔트로피를 우선순위 함수로 하는 리스트 스케쥴링으로 고정 데이타 입력간격과 가변 데이타 입력간격을 갖는 문제를 해결하였다. PLS[10]는 LCD(Loop Carried Dependency)가 있는 경우에도 스케쥴링이 가능하며, Sehwa처럼 경로길이에 의한 우선순위 함수를 사용하지만 전진 스케쥴링과 후진 스케쥴링을 사용하여 이미 스케쥴된 연산을 재 스케쥴하는 방법이다. Lee등[11]은 자원 제약 조건에서 LCD를 고려하는 기능적 파이프라인 스케쥴링에 관한 연구로 데이타패스 설계의 중요한 성능인 루프의 반환시간을 최소화하기 위하여 우선 ASAP 스케쥴링을 수행하고, 자원 충돌을 해결하기 위하여 자원 제약과의 확률 편차를 사용한 우선순위 함수를 적용하였다. Hwang등[12]은 자원 공유의 최대화를 위하여 연산을 균등 분포시키도록 하는데 정확한 스케쥴링을 위하여 멀티플렉서의 지연까지도 고려하였으며 리스트 스케쥴링을 변형하여 연산의 할당 우선순위로 모빌리티를 사용하고 밀집도가 최소인 파티션을 선정하여 그곳에 모빌리티가 가장 적은 연산을 배정하도록 하며 가변 데이타 입력 간격을 허용하였다.

본 논문에서는 DFG와 자원제약으로 자원의 갯수가 입력인 경우에 이 자원 조건하에서 최소의 데이타 입력 간격을 구하고 이 입력 간격에서 기능적 파이프라인 데이타패스의 실행 시간이 최소가 되도록 연산의 충돌수와 자원수, 그리고 모빌리티에 의한 우선순위 함수를 정의하여 해를 찾는 휴리스틱 스케쥴링 알고리즘을 제안한다. 또한 실질적인 하드웨어 합성을 위해 멀티 사이클링, 체이닝, 연산 모듈의 구조적 파이프라인을 지원하도록 한다. 제안된 휴리스틱 스케쥴링 알고리즘은 반복적(iterative) 알고리즘이며, 성능 평가를 위해 벤치 마크 회로인 16 포인트 FIR 필터와 5차 엘립틱 웨이브 필터를 사용하여 실험하였다.

2. 파이프라인 합성 스케쥴링

2.1 데이타 입력 간격과 파티션

자원 제약하의 비파이프라인 스케쥴링은 각 제어단

계에 할당되는 연산의 수만을 고려하지만 파이프라인 스케쥴링은 동시에 병행 수행될 수 있는 제어단계들에 할당되는 연산의 수를 고려해야 한다. 동시에 병행 수행되는 제어단계들을 파티션이라 하는데, 제어단계 번호를 데이타 입력 간격으로 나눌 때 나머지가 같은 제어단계들이 하나의 파티션이 되어 파티션의 수는 데이타 입력 간격만큼 발생한다.

기능적 파이프라인에서 데이타 입력 간격(DII: Data Initiation Interval)은 식(1)과 같이 자원 종류별로 실행되어야 하는 연산의 수 op_i와 이용가능한 자원의 수 fu_i에 의해 결정되며, 동시에 수행되는 제어단계들의 집합인 파티션 P_k는 식(2)와 같다[6][13].

$$DII \geq \lceil \frac{op_i}{fu_i} \rceil \quad (1)$$

$$P_k = \{ \text{제어단계} CS_n \mid n \bmod DII = k,\ 0 \leq k < DII \} \quad (2)$$

여기서 데이타 입력 간격이 자원의 종류에 따라 다르게 계산될 경우는 큰값이 데이타 입력 간격이 된다.

2.2 초기 상태와 자원 충돌

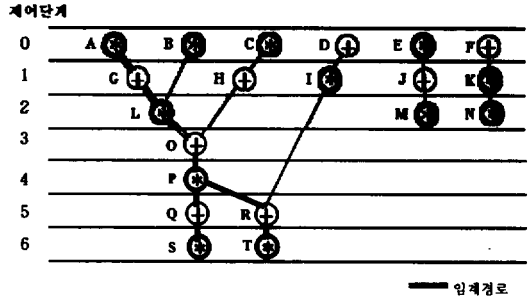
하드웨어 동작기술을 그래프 형태로 표현한 DFG를 제한된 자원을 가지고 파이프라인으로 동작하도록 스케쥴하기 위해 ASAP과 ALAP 스케쥴링 결과를 초기 상태로 설정하는데, 이것으로부터 DFG상의 임계 경로와 연산들의 모빌리티를 결정할 수 있기 때문이다. 자원 충돌은 각 파티션에 배정되는 연산의 수가 이용가능한 자원의 수를 초과 사용하려는 경우에 발생한다. 초기상태하에서 자원 충돌없이 스케쥴링될 가능성이 있는지 알아보기 위해 임계경로상에 있는 연산들이 이용가능한 자원의 수를 초과하는지 계산한다. 임계경로 밖의 연산들은 그들의 모빌리티에 의해 자원 충돌없이 스케쥴링 될 가능성이 있으므로 고려하지 않는다.

초기상태의 임계경로상에서 제한된 자원에 의한 충돌때문에 스케쥴링이 불가능하면 현 제어단계 수로는 스케쥴링이 불가능하므로 제어단계 수를 하나 증가시키고 초기 상태를 다시 설정한다.

2.3 충돌수를 고려한 우선순위 함수

자원제약 파이프라인 스케쥴링시에 각 파티션에 배정되는 연산의 수가 이용가능한 자원의 수를 초과할 경우에는 자원 충돌이 발생하게 된다. 본 논문은 제한된 자원의 수를 초과하는 연산들을 충돌수를 고려한 우선순위 함수를 이용하여 모든 파티션에서 자원 충돌이 발생하지 않도록 스케쥴링하는 방법을 제안한다. 충돌수는 자원 충돌이 발생할때에 그 연산과 관련된 자원을 사용하려는 연산들의 수의 합으로 정의한다. 자원 충돌 발생없이 DFG에 있는 연산들이 각 파티션에 스케쥴되는 경우에 충돌 연산수는 0이 된다. 자원 충돌이 발생한 경우에는 모빌리티가 있는 모든 연산노드를 각각 다음 제어단계로 지연시키는 경우의 우선순위 함수값을 계산하고 이 값이 최소인 연산노드를 선택하여 지연시킨다. 우선순위 함수는 모빌리티가 있는 연산이 다음 제어단계로 지연될 경우에 충돌 연산수의 비율이 적어지도록, 그리고 연산들의 모빌리티가 커질 수 있도록 다음과 같이 정의 한다.

$$\text{우선순위 함수} = \sum_{i \in \text{충돌된 연산}} \left(\frac{\text{충돌 연산수}_i}{\text{전체 자원수}_i} * \frac{1}{\text{모빌리티}_i} \right)$$



(a) 데이타 흐름 그래프
(a) Data Flow Graph

파티션 \ 연산	덧셈연산	곱셈연산
P0	D F	A B C E P
P1	G H J Q R	I K
P2		L M N S T
P3	O	

(b) 각 파티션에 배정되는 연산
(b) Operation allocation in each partition

(그림 1) DFG와 파티션의 예
(Fig. 1) Example of DFG and Partition

여기서 충돌한 연산 i 는 각 파티션에서 이용가능한 자원을 초과하는 경우에 초과되는 자원 종류별로 해당하는 모든 연산을 의미한다.

예를들어 그림 1(a)의 DFG에서 자원 제약이 덧셈기 2개, 곱셈기 3개로 주어진다면, 식(1)에 의해 최소 데이타 입력 간격은 4가 되고, 이때의 파티션 수는 4개가 된다. 또한 각 파티션에 배정되는 연산들은 그림 1(b)와 같이 된다. 그런데 덧셈 연산의 경우 이용가능한 덧셈기 2개를 초과하는 파티션 P_1에서 자원의 충돌이 발생하는데, 충돌 연산수는 5가 된다. 곱셈 연산의 경우 이용가능한 곱셈기 3개를 초과하는 파티션 P_0, P_2에서 자원의 충돌이 발생하며 충돌 연산수는 각각 5가 된다. 현재의 상태에서 전체 충돌 연산수는 충돌이 발생하는 파티션 내의 모든 연산을 누적하여 15가 된다.

2.4 합성 스케쥴링 알고리즘

제안된 알고리즘은 표 1과 같으며, 하드웨어의 동작에 관한 서술과 이용가능한 자원의 갯수를 입력으로 하여, 이 입력을 바탕으로 최소 데이타 입력 간격을 계산한다. ASAP과 ALAP 스케쥴링으로 초기상태를 설정한 후에 임계경로 분석에 의해 충돌이 발생하지 않을 가능성이 있는 최소 제어단계 수를 찾는다. 단계 5는 모든 파티션내에 자원 충돌이 발생하지 않을 때까지 반복하며, 단계 5-1에서는 현재의 제어단계 수로 스케쥴링이 가능한지 검사한다. 자원 충돌이 있는 상황에서 모든 연산 노드의 모빌리티가 고정되면 현재의 제어단계 수로는 스케쥴링이 불가능하다. 이러한 경우에는 제어단계의 수를 하나 증가 즉 모든 연산 노드의 모빌리티를 증가시켜 스케쥴링이 가능하도록 한다. 단계 5-2와 5-3에서는 모빌리티가 있는 모든 연산노드를 각각 지연하는 것을 가정하여 충돌수에 의한 우선순위 함수값을 계산한 다음 이 값이 최소인 연산노드를 선택하여 지연한다.

〈표 1〉 제안 스케쥴링 알고리즘
〈Table 1〉 Proposed Scheduling Algorithm

```
단계 1.  데이타 입력 간격 결정
단계 2.  ASAP, ALAP 스케쥴링
단계 3.  DFG에서 임계경로 발견
단계 4.  if (임계경로상 충돌 발생)
           제어단계 수 1 증가 및 ASAP, ALAP 스케
           쥴링
단계 5. 모든 파티션내 충돌 발생이 없을때 까지 반복
단계 5-1. if (모든 연산노드가 고정) {
              ASAP 스케쥴링
              제어단계 1 증가
            }
단계 5-2. 모빌리티가 있는 모든 연산노드를 지연하
           여 충돌수에 의한 우선순위 함수값 계산
단계 5-3. 충돌수에 의한 우선순위 함수값이 최소인
           연산노드를 선택 지연
```

제안된 알고리즘의 수행시간은 $O(n^2)$인데, DFG의 연산 노드의 수가 n개인 경우에 단계 5가 한번 반복될 때마다 스케쥴될 연산 노드가 하나 선택되고, 이때에 최대 n번의 계산이 이루어지며, n개의 연산노드 만큼 반복 수행하여야 하기 때문이다.

3. 알고리즘의 확장

제안된 알고리즘은 자원 제약 파이프라인 데이타 패스 합성을 위한 스케쥴링으로 비파이프라인 데이타 패스 합성에도 사용될 수 있다. 여기에 실질적인 하드웨어 설계에서 사용될 수 있도록 체이닝, 멀티사이클링, 연산 모듈의 구조적 파이프라인등이 지원될 수 있도록 확장하였다.

3.1 체이닝

체이닝을 지원하기 위해서는 우선 ASAP, ALAP 스케쥴링이 변경되어야 하는데, 기존의 ASAP, ALAP 스케쥴링에서 종속성이 있는 연산들이 동일한 제어단계에 할당될 수 없었으나, 체이닝을 할 경우에 이를 지원하도록 해야한다. 이를 위해 연산의 지연 시간과 제어단계의 클럭 길이를 사용하여 선행 연산과 배정하려는 연산의 지연시간의 합이 클럭 길이보다 작으면 같은 제어단계에 배정한다.

또한 알고리즘 단계 5-2, 5-3에서 연산노드를 지연할 때 지연되는 연산과 종속성을 갖는 후행 노드들에 할당되는 제어단계를 변경해야 하는데, 이때도 역시 종속성이 있는 연산들이 동일한 제어단계에 할당되도록 한다.

3.2 멀티사이클링

멀티사이클링을 지원하기 위해서는 우선 ASAP, ALAP 스케쥴링이 변경되어야 하는데 서로 종속성이 있는 연산들에 있어서 후행 노드의 ASAP, ALAP 값은 선행 노드의 ASAP, ALAP 값뿐만 아니라 선행 노드의 지연시간까지도 고려하여야 한다. 즉 ASAP 스케쥴링할 때, 만약 선행 연산이 멀티사이클링 연산이라면 연산은 선행 연산의 제어단계 번호와 선행 연산의 지연시간을 더한 값의 제어단계에 배정한다.

또한 충돌수를 계산할 때 멀티사이클링 연산은 연산 시작 제어단계에서 지연시간 만큼의 제어단계에 포함시켜 충돌수를 계산한다

그리고 알고리즘 단계 5-2, 5-3에서 연산노드를 지연할 때 지연되는 연산과 종속성을 갖는 후행 연산들에 할당되는 제어단계를 선행 연산의 제어단계 값과 선행연산의 지연시간까지도 고려하여 변경한다.

3.3 구조적 파이프라인

구조적 파이프라인을 지원하기 위해서는 역시 ASAP, ALAP 스케쥴링이 변경되어야 하는데 서로 같은 연산장치를 사용하는 연산에 있어서 선행 연산이 구조적 파이프라인 연산이라면 후행 연산의 ASAP, ALAP 값은 구조적 파이프라인이 허용하는 입력 간격에 의해 결정된다.

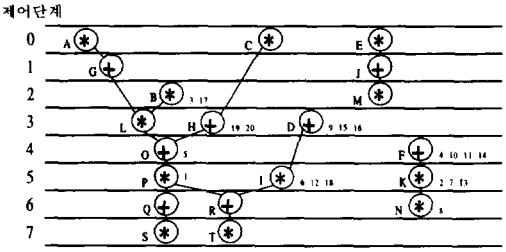
또한 충돌수를 계산할 때, 구조적 파이프라인 연산장치는 연산 시작시간에서 입력간격 만큼의 제어단계에 포함시켜 충돌수를 계산한다.

4. 실험 결과

제안된 알고리즘은 충돌수를 이용한 우선순위 함수에 의한 파이프라인 데이타패스 합성 스케쥴링 알고리즘으로 제한된 자원 또는 면적하에 자원충돌 발생이 없는 스케쥴링 결과를 구하는 휴리스틱이다.

그림 1의 DFG를 제안된 휴리스틱에 의해 스케쥴링할 때에, 이용가능한 자원을 덧셈기 2, 곱셈기 3개로 제한한다면 그림 2와 같은 스케쥴링 결과를 얻는다. 임계경로에 있는 연산을 보면 파티션 P1에 배정된 덧셈연산 5개가 이용가능한 자원 2개를 초과하기 때문에 제어단계의 수를 하나 증가하고 스케쥴링을 시작한다. 노드의 우측 숫자는 스케쥴링이 되는 순서를 나타낸다. 노드의 우측 숫자가 여러개 있는 것은 스케쥴링 과정에서, 해당 노드가 지연되어 스케쥴링이 되는 것을 나타낸다.

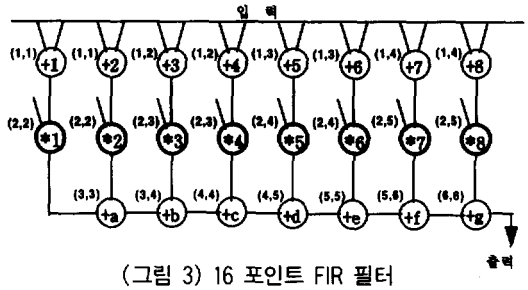
알고리즘은 펜티엄 PC 환경에서 C언어로 구현하였으며, 제안된 알고리즘의 성능 평가를 위해 벤치마크용 16 포인트 FIR 필터와 5차 엘립틱 웨이브 필터를 사용하였다.



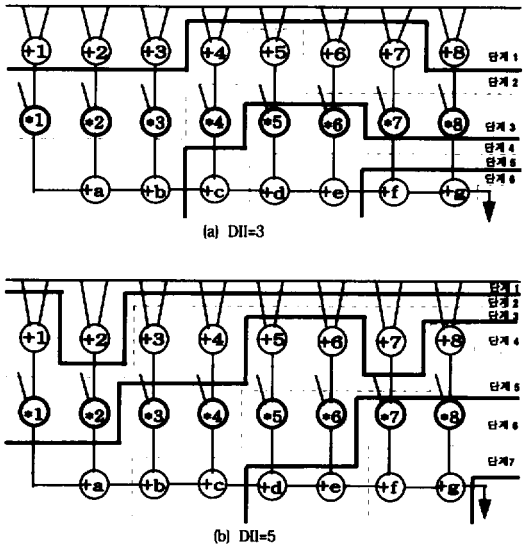
(그림 2) 그림 1에 대한 스케쥴링 결과
(자원제약: 덧셈기 2, 곱셈기 3)
(Fig. 2) Result of scheduling of Fig. 1
(resource constraint : adder 2, multiplier 3)

4.1 16 포인트 FIR 필터

그림 3과 같은 16 포인트 FIR 필터는 15개의 덧셈연산과 8개의 곱셈 연산으로 구성되며 모든 연산이 1 사이클 연산이라면 임계경로상에 9개의 연산이 있기 때문에 적어도 9 사이클의 응답시간이 필요하다. 다른 시스템과의 비교를 위해 지연시간이 적은 덧셈연산 2개가 하나의 제어단계에서 실행가능한 체이닝을 허용하는 것으로하여 초기에 6개의 제어단계를 할당한다. 그림 3에서 연산노드의 좌측 숫자는 노드의 할당가능한 제어단계를 의미하며 ASAP, ALAP 스케쥴링으로 얻을 수 있다. 그림 4(a)는 이용가능한 자원을 덧셈기 5, 곱셈기 3, 즉 데이타 입력 간격 DII가 3일때의 스케쥴링 결과를 보여준다. 그림 4(b)는 이용가능한 자원을 덧셈기 3, 곱셈기 2, 즉 DII가 5일때의 스케쥴링 결과이다.



(그림 3) 16 포인트 FIR 필터
(Fig. 3) 16 point FIR filter



(그림 4) 16 포인트 FIR 필터 스케줄링 결과
(Fig. 4) Result of scheduling of 16 point FIR filter

4.2 5차 엘립틱 웨이브 필터

5차 엘립틱 웨이브 필터는 26개의 덧셈연산과 8개의 곱셈연산으로 구성되며, 다른 시스템과의 비교를 위해 덧셈은 1사이클 연산이고, 곱셈연산은 2사이클에 걸쳐 수행되는 멀티 사이클 연산으로 가정한다. 또한 이 필터는 루프 내부와 외부가 모두 데이타 의존성을 포함하는 필터이지만 외부 의존성은 고려하지 않으며, 멀티 사이클링이 지원되도록 곱셈기는 2 스테이지로 구성되는 것으로 한다.

4.3 스케쥴링 결과

표 2와 표 3은 앞절에서 기술된 각각 16 포인트 FIR 필터와 5차 엘립틱 웨이브 필터의 설계시에 이용가능한 자원의 수가 주어질 때 최소의 데이타 입력 간격과 스케줄된 제어단계의 수를 나타내며, 다른 스케줄링 알고리즘과의 비교 결과이다.

5. 결 론

본 논문에서 상위 수준 합성시의 자원 제약 조건하에서 파이프라인 데이타패스 합성을 위한 새로운 스케줄링 알고리즘을 제안하였다. 최소의 데이타 입력 간격과 제한된 자원하에 충돌 발생이 없는 스케줄링 결과를 충돌수에 의한 우선순위 함수를 이용하여 구하였다. 제안 알고리즘은 비파이프라인 데이타패스 합성은 물론 실질적인 하드웨어 합성을 위해 체이닝, 구조적 파이프라인, 멀티 사이클링이 지원되며, 16 포인트 FIR 필터와 5차 엘립틱 웨이브 필터에 의한 성능 분석으로 알고리즘의 효율성을 보였다.

제안된 알고리즘은 거의 모든 경우에 최적의 해를 구하였지만 표 3의 별표에서와 같이 근사해를 구하는 경우도 있는데, 이는 최적의 경로가 극히 제한되는 경우에 근사해를 찾을 가능성이 높음을 보여주고 있다. 이를 해결하기 위해서는 우선순위 함수 결정시에 다른 요소를 더 고려해야할 것으로 생각된다.

〈표 2〉 16 포인트 FIR 필터
〈Table 2〉 16 point FIR filter

덧셈기		15	8	6	5	4	3	2	1
곱셈기		8	4	3	3	2	2	1	1
레이턴시(L)		1	2	3	3	4	5	8	15
시스템	Sehwa[2]		6		6	6	7		
	HAL[3,4]				6				
	EPLS[10]				6				
	Lee등[11]	6	6	6	6	6	7	10	16
	본논문	6	6	6	6	6	7	10	15

〈표 3〉 5차 엘립틱 웨이브 필터
〈Table 3〉 5th order elliptic wave filter

덧셈기		26	13	9	7	6	5	4	4	3	2	1
곱셈기		8	4	3	2	2	2	2	1	1	1	1
레이턴시(L)		1	2	3	4	5	6	7	8	9	13	26
시스템	Optimum[10]	17	17	18	19	19	17	18	20	22	23	33
	PLS[10]	17	17	18	19	19	17	18	20	22	23	33
	Sehwa[2]	17	17	18	19	20	21	20	22	25	24	33
	본논문	17	17	18	19	19	17	18	21*	22	23	33

참 고 문 헌

[1] P. M. Kogge, 'The Architecture of pipelined computers,' McGraw-Hill, 1981.

[2] 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.

[3] Pierre G. Paulin and John P. Knight, "Force-directed scheduling for the behavioral synthesis of ASIC's," IEEE Trans. Computer-Aided Design, Vol.8, pp.661-679, June 1989.

[4] Pierre G. Paulin and John P. Knight, "Scheduling and binding algorithm for high-level synthesis," 26th ACM/IEEE DAC, pp.1-6, 1989.

[5] W.F.J. Verhaegh, P.E.R. Lippens, E.H.L. Aarts, J.H.M. Karst, A. van der Werf, and J.L. van Meerbergen, "Efficiency improvements for force-directed scheduling," Proc. IEEE Int. Conf. on Computer-Aided Design, pp.286-291, 1992.

[6] Y-H Choi, "Synthesis of pipelined datapaths," computer-aided design, Vol.24, No.1, pp.36-40, Jan. 1992.

[7] Hong Shin Jun and Sun Young Hwang, "Automatic synthesis of pipeline structures with variable data initiation intervals," 31st ACM/IEEE DAC, pp.537-541, 1994.

[8] 이해동, 전홍신, 황선영, "VHDL 상위수준 합성 시스템의 설계", 한국정보과학회 논문지, Vol.20, No.6, 1993.6

[9] 박재환, 전홍신, 황선영, "파이프라인 시스템 합성을 위한 면적 제약조건 스케쥴링 알고리듬", 한국정보과학회 논문지, Vol.20, No.6, 1993.6

[10] Cheng-Tsung Hwang, Yu-Chin Hsu and Youn-Long Lin, "PLS: A scheduler for pipeline synthesis," IEEE trans. on CAD/ICAS, Vol.12, No.9, pp.1279-1286, Sept. 1993.

[11] Tsing-Fa Lee, Allen C-H Wu, Daniel D. Gajski and Youn-long Lin, "An effective methodology for functional pipelining," Proc. IEEE Int. Conf. on Computer-Aided Design, pp.230-233, 1992.

[12] Ki-Soo Hwang, Albert E. Casavant, Ching-Tang Chang and Manuel A. d'Abreu, "Scheduling and hardware sharing in pipelined data paths," Proc. IEEE Int. Conf. on Computer-Aided Design, pp. 24-27, 1989.

[13] Yuan-Long Jeang and Yu-chin Hsu, "Pipeline scheduling techniques in high-level synthesis," 30th ACM/IEEE DAC, pp.396-403, 1993.

[14] Cheng-Tsung Hwang, Yu-Chin Hsu and Youn-Long Lin, "Optimum and heuristic data path scheduling under resource constraints," 27th ACM/IEEE Design Automation Conference, pp.65-70, 1990.

유 동 진

1994년 서울산업대학교 전자계산학과 졸업(공학사)

1996년 홍익대학교 대학원 전자계산학과 졸업(이학석사)

1996년 2월~1996년 8월 현대전자 소프트웨어 연구소

1996년 8월~현재 현대정보기술

관심분야 : 설계자동화, 컴퓨터 구조

유 희 진

hjyoo@cs.hongik.ac.kr

1990년 원광대학교 전자계산공학과 졸업(공학사)

1992년 홍익대학교 대학원 전자계산학과 졸업(이학석사)

1995년 9월~현재 홍익대학교 대학원 박사과정 수료

1996년~현재 전주대학교 컴퓨터공학과 객원교수

관심분야 : 상위수준합성, Hardware/Software Co-Design

박 도 순

dspark@cs.hongik.ac.kr

1978년 서울대학교 전자공학과 졸업(공학사)

1980년 한국과학기술원 전산학과 졸업(이학석사)

1988년 고려대학교 대학원 수학과 졸업(이학박사)

1980년~1983년 국방과학연구소

1983년~현재 홍익대학교 컴퓨터공학과 부교수

관심분야 : 컴퓨터 구조, 설계자동화등