論文95-32A-6-11

# 클럭주기 최소화를 위한 효율적인 연결구조 할당 알고리듬

# (An Efficient Interconnect Allocation Algorithm for Clock Period Minimization)

金 泳 盧 *, 李 海 東 **, 黃 善 泳 **

(Kim Young No, Hae Dong Lee, Sun Young Hwang)

요 약

본 논문은 성능 구동 연결구조 할당 알고리듬의 설계에 대해 기술한다. 수행시간이 긴 하드웨어 모듈을 경유하는 통신경로들에 대해서 부하를 분배하므로써 클럭주기는 최소화될 수 있다. 제안된 알고리듬은 임계경로를 형성할 가능성이 있는 통신경로들에 대해서만 부하분배를 고려하여 빠른 시간내에 최소의 부하에 의한 지연시간을 갖는 연결구조를 생성한다. 밴치마크 프로그램에 대한 타 시스템과의 실험결과 비교를 통하여 제안된 알고리듬이 보다 최소화된 클럭주기를 갖도록 연결구조를 생성함을 보인다.

**Abstract**

This paper presents the design of a performance-driven interconnect allocation algorithm. The algorithm is based on the idea that the clock period can be minimized by balancing the load for each of the communication paths following specific hardware modules. By performing load balancing for only the communication lines on critical paths, the proposed algorithm generates interconnection structures with minimum delays. This approach also shows run time efficiency. Experimental results confirm the effectiveness of the algorithm by constructing the interconnection structures such that the clock period can be minimized for several benchmark circuits available from the literature.

## I. 서 론

VLSI 설계기술의 발달로 칩 집적도가 증가하고 설계시간의 감소가 요구됨에 따라 칩 설계에 있어서 자동화의 범위가 확대되고, 기존의 bottom-up 설계 방식보다는 실리콘 컴파일러를 사용하는 top-down 설계 방식이 대두되고 있다[1][2]. 칩 설계자의 설계요구에 대한 합성에 있어서 그 abstraction level이 높을수록 설계시 고려되어야 할 사항에 대한 설계자의 부담이 줄어들게 되고, 더 넓은 설계공간이 탐색될 수 있으며, 설계자의 다양한 설계결정이 반영될 수 있는 잇점이 있다. 따라서, 실리콘 컴파일러 상에서의 최상위 레벨인 상위수준 합성(high-level synthesis)에 관한 연구가 활발히 진행되어 왔으며 최근에는 상용화되고 있는 단계이다. 상위수준 합성은 알고리듬 수준의 하드웨어 동작기술(behavioral description)을 입력으로 받아들여 넓은 설계공간을 탐색하면서 레지스터, 메모리, 연산모듈, 그리고 그들 간의 연결구조로 구성된 레지스터 전송수준(register-transfer level)의 데이타 패스를 합성하는 과정이다[1][2][3][4]

* 正會員, LG 半導體 技術硏究所
(LG Semicon, Technology R&D Lab.)
** 正會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)
接受日字: 1994年11月25日, 수정완료일: 1995年6月3日

상위수준 합성은 스케쥴링과 모듈할당의 두 과정으로 나뉜다. 스케쥴링은 설계하고자 하는 동작기술에서 사용된 연산을 하드웨어의 면적, 지연시간, 파워소모 등과 같은 설계 제약조건을 만족하는 범위에서 최적으로 제어구간(control step)에 할당하는 과정으로서 생성된 하드웨어의 동작속도와 면적 등에 큰 영향을 미친다. 모듈할당 과정은 연산기 할당, 기억소자 할당, 연결구조 할당의 부 과정으로 구성된다. 연산기 할당은 스케쥴링에 의해 각 제어구간에 할당된 연산을 연산기에 할당하는 과정이고, 기억소자 할당은 제어구간의 경계에 존재하는 동작기술상의 변수를 타겟 아키텍춰에 따라 레지스터 혹은 메모리 요소로 할당하는 과정이다. 이 두 과정에서 할당된 연산기와 기억소자 간의 통신경로(communication path)들은 연결구조 할당과정에서 버스와 mux를 이용한 연결구조로 할당된다.

스케쥴링이 끝나면 클럭주기와 동작수행에 필요한 제어구간의 수가 결정되므로, 모듈할당 과정에서는 필요한 하드웨어 수의 최소화를 목적으로 한다. 특히, 연결구조에 있어서는 그 면적이 전체 칩 면적의 상당부분을 차지하므로 대부분의 시스템들이 연결구조를 최소화하는 방향으로 모듈할당을 수행한다[5][6][7][8]. HAL[5]은 clique partitioning 알고리듬을 사용하여 레지스터 할당을 수행하는데 있어서 호환 그래프(compatibility graph) 상의 각 에지에 연결구조의 비용을 목적함수로 고려하므로써 전체적으로 연결구조의 면적을 최소화한다. FACET[6]는 연결구조의 할당에 있어서 연산기 할당과 레지스터 할당의 결과로부터 추출된 각 통신경로를 노드로, 통신경로들 간에 제어구간이 겹치지 않는 경우에 에지로 연결된 호환 그래프를 구성하고 clique partitioning 알고리듬을 이용하여 버스를 합성한다. 분할된 그래프에서 clique의 수는 버스의 수를 나타내고, 각 clique를 구성하는 통신경로들은 한 버스를 공유하게 된다. 이처럼 대부분의 상위수준 합성 시스템들은 연결구조의 할당에 있어서 그 면적요소만이 목적함수(objective function)에 반영되므로써 연결구조 할당의 결과로 인한 부가적인 지연시간에 대해서는 고려되지 않은 합성결과를 보였다. 그러나 실시간 처리 회로 등 많은 응용분야에서 시스템의 클럭주기는 전체 성능을 결정하는 매우 중요한 요소이므로 연결구조 구성에 따른 부가적인 지연시간은 가능한한 최소화되어야한다.

클럭주기는 스케쥴링에 의해 결정된 각 제어구간에 존재하는 기억소자와 연산기의 최대 수행시간(execution delay)과 모듈할당의 결과로부터 생성되는 연결구조의 지연시간에 의해 결정된다[3][9] 연산기나 기억소자의 경우 설계되어 사용되는 라이브러리에 의해 그 지연시간이 결정되나, 연결구조의 경우 그 할당 결과가 클럭주기 결정에 영향을 미친다. 따라서, 연결구조의 할당은 그 면적 뿐아니라 지연시간에 관한 고려를 기반으로 이루어져야한다[8][9]. Jiang[9]은 데이타 전송에 있어서 데이타 근원 요소(기억소자 혹은 연산기)의 부하와 버스의 부하가 전체 지연시간에 영향을 미치는 점에 착안, 처음으로 성능을 고려한 연결구조 할당을 시도하였다. 즉, 연결구조를 할당함에 있어서 클럭주기의 최소화를 위해 데이타 근원 요소의 부하와 데이타 전송자인 버스의 부하가 균등히 분배되도록 버스를 할당하므로써 데이타 전송시 각 요소들의 부하에 기인한 부가적인 지연시간이 최소화되도록 하였다. 연결구조 할당의 방법으로서는 최적의 해를 산출하기 위한 ILP(Integer Linear Programming) 방법과 효율적으로 해를 구하기 위한 휴리스틱 알고리듬을 각각 제시하였다.

ILP에 의한 방법에서는 최소화할 부하와 사용될 하드웨어 수에 관한 제약조건들을 각각 목적함수와 제약수식(constraint expression)으로 하는 ILP 문제로 정의하였다. ILP 방법에 있어서 문제의 크기가 증가함에 따라 수행시간이 기하급수적으로 증가하는 단점을 극복하기 위한 휴리스틱 알고리듬에서는 통신경로들과 사용될 버스들의 집합을 두 개의 노드 집합으로 구성하고, 서로 다른 집합의 노드들 간에 존재하는 에지의 목적함수를 증가되는 부하량으로 정의한 이중분할 그래프(bipartite graph)를 구성하고, 이중분할 매칭(bipartite matching) 알고리듬을 적용하여 연결구조 할당을 수행하였다.

그러나, 실제적으로 클럭주기는 임계경로(critical path)에 의해서 결정되므로, 데이타패스 상의 전체 하드웨어 요소들에 대해서 균등히 분배된 부하는 클럭주기의 최소화에 기여하지 못할 수가 있다. 본 논문에서는 연산기와 기억소자에 대한 할당이 수행된 후, 사용된 하드웨어의 수행 지연시간을 참조하여 임계경로를 형성할 가능성이 있는 통신경로들에 대해서만 부하분배를 고려하므로써 클럭주기가 보다 최소화되도록 하고, 부하분배의 대상이 되는 통신경로들을 제한하므로써 빠른 수행시간 내에 연결구조 할당을 수행하는 새로운 알고리듬을 제안한다. 2 장에서는 본 논문에서 다루는 문제에 대한 정의와 문제해결을 위한 접근방법에 대해 기술하고, 3장에서는 연결구조 할당 알고리듬을 기술한다. 제안된 알고리듬의 성능비교를 위해 4 장에서는 기존의 연구에서 사용한 네가지 벤치마크에 대해 수행한 실험결과를 기존방법의 실험결과와 비교하고, 마지막 장에서는 본 논문에 대한 결론을 제시한다.

## Ⅱ.문제의 정의 및 접근방법

### 1. 클럭주기의 결정

레지스터 전송수준의 데이타패스에서 클럭주기는 임계경로 상의 레지스터간 데이타 전송에 필요한 시간에 의해 결정된다. 그림 1은 버스 기반의 연결구조를 가진 간단한 레지스터 전송수준의 데이타패스를 나타낸다.

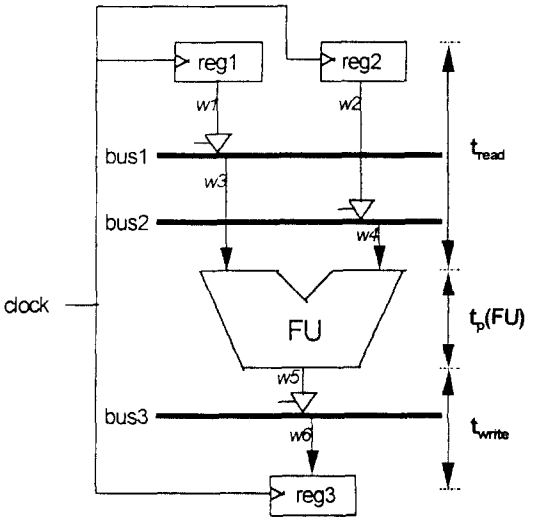



그림 1. 버스를 기반으로 한 레지스터 전송수준의 데이타패스

Fig. 1. An RT level datapath with bus-based interconnections.

그림 1에서 보듯이 데이타가 근원 레지스터(source register)로부터 목적 레지스터(destination register)까지 전달되는데 걸리는 시간은 근원 레지스터의 지연시간, 연결구조와 연산기의 지연시간 그리고 목적 레지스터의 준비시간(setup time)에 의해 결정된다. 따라서 $t_p(\cdot)$를 전달지연시간(propagation delay), $t_{setup}(\cdot)$ 를 setup time 으로 표기할때, 그림 1의 데이타패스에서 read phase에서의 연결구조에 의한 지연시간 $T_{read}$ 와 연산의 결과가 레지스터에 전달될 때까지의 시간 $T_{write}$ 는 각각 아래의 식 (1)(2)와 같이 표현되고, 클럭주기 $T_{clock}$은 식 (3)과 같이 표현된다.

$$T_{read} = \max(t_p(reg1) + t_p(w1) + t_p(bus1) + t_p(w3),\ t_p(reg2) + t_p(w2) + t_p(bus2) + t_p(w4)) \quad (1)$$

$$T_{write} = t_p(w5) + t_{p(bus3)} + t_p(w6) + t_{setup}(reg3) \quad (2)$$

$$T_{clock} = t_{read} + t_p(FU) + t_{write} \quad (3)$$

위의 식 (1)에서 $t_p$(reg)와 $t_p$(FU)는 연산기 할당과 레지스터 할당이 수행된 후 설계된 컴포넌트 라이브러리에 의해 그 값이 정해진다[11]. 연결구조 요소인 버스와 wire의 지연시간은 wire segment가 갖는 RC 값과 구동할 컴포넌트의 캐패시턴스 성분에 의해 결정된다[3][12]. 그러나 wire segment의 RC 값은 그 길이에 의존적이고 보다 구체적으로 placement/routing이 수행된 후에야 측정이 가능하므로 여기서는 구동할 fanout 부하량에 의한 지연시간만을 고려하도록 한다. Read phase 에서의 연결구조에 의한 지연시간은 레지스터가 구동해야할 버스 입력단의 버퍼(tri-state buffer)의 부하량과, 레지스터로부터 연산기로 데이타를 전달하는 버스가 구동해야할 부하량에 의해 결정되고, write phase 에서는 연산기의 출력포트가 구동해야할 버퍼의 부하량과 연산기로부터 레지스터로 데이타를 저장하는데 사용되는 버스 출력단의 부하량에 의해 정해진다.

### 2. 접근방법

Jiang[9] 은 데이타의 근원이 되는 레지스터와 연산기 그리고 버스의 부하량에 따른 지연시간의 증가량을 살펴보기 위해 Gensil 시스템을 사용하여 1.2㎛ CMOS 공정의 타겟 라이브러리를 사용한 실험을 수행하고 그 결과를 제시하였다. 부하에 따른 지연시간의 증가는 선형적인 특성을 나타내었으며, 컴포넌트당 그 부하량에 따른 지연시간의 증가량는 타겟 테크놀로지 라이브러리에 상당히 의존적이다. 따라서 본 논문에서는 동일한 조건하의 성능비교를 위해 각 컴포넌트당 부하에 따른 지연시간의 증가량에 대해서는 Jiang의 데이타를 사용하였다. 본 논문에서는 각 레지스터와 연산기의 출력포트에 연결된 버퍼(TSB)의 수에 대한 지연시간의 증가분을 각각 $\Delta_{tr}$와 $\Delta_{tf}$로 표기하고, 버스가 구동해야할 컴포넌트 즉, 레지스터, 연산기, mux 입력의 수에 대한 지연시간의 그 증가분은 각각 $\Delta_{br}, \Delta_{bf}$, $\Delta_{bm}$로 표기한다. 통신경로는 데이타의 근원지와 목적지로 구성되어 read phase의 경우는 레지스터로부터 연산기로 데이타를 읽어내는 경로가 되고, write phase에 있어서는 연산기의 연산결과가 레지스터로 적재되는 경로가 된다. Read phase와 write phase 에서의 통신경로들의 수를 각각 $N^R$, $N^W$라 하면, read, write phase의 통신경로들의 집합은 $C^R = \{C_i^R \mid 1 \le i \le N^R\}$, $C^W = \{c_j^W \mid 1 \le j \le N^W\}$와 같이 표기될 수 있다. Read phase의 통신경로 $c_i^R$의 레지스터가 구동하는 TSB의 수를 $N_i^{rt}$, 레지스터로 부터 연산기로 데이타를 전달하는 버스가 구동하는 레지스터

의 수와 연산기의 수를 각각 $N_i^{br}, N_i^{bf}$로 표기하고, write phase의 통신경로 $c_j^w$ 의 연산기가 구동하는 TSB의 수를 $N_j^{tt}$, 연산의 결과를 레지스터로 전달하는 버스가 구동하는 레지스터의 수와 연산기의 수를 각각 $N_j^{bt}$, $N_j^{bf}$로 표기하면, 각 phase에서의 부하에 의한 지연시간은 아래의 식 (4)(5)와 같이 정의될 수 있다.

$$t_{read}(C_i^R) = \Delta^{rt} \cdot N_i^{rt} + \Delta_{br} \cdot N_i^{bf} \quad (4)$$

$$t_{write}(C_j^w) = \Delta_{ft} \cdot N_j^{ft} + \Delta_{br} \cdot N_j^{br} + \Delta_{bf} \cdot N_j^{bf} \quad (5)$$

레지스터나 연산기를 구동함에 따른 버스의 부하가 비슷한 경우 $\Delta_{br}$와 $\Delta_{bf}$는 $\Delta_{bl}$로 표기하고, 각 phase에서의 버스의 부하를 $N_i^{bl}$과 $N_j^{bl}$로 표기하면 위의 식 (4)(5)는 아래의 식 (4′)(5′)와 같이 표현될 수 있다.

$$t_{read}(c_i^R) = \Delta_{rt} \cdot N_i^{rt} + \Delta_{bl} \cdot N_i^{bl} \quad (4')$$

$$t_{write}(c_j^w) = \Delta_{ft} \cdot N_j^{ft} + \Delta_{bl} \cdot N_j^{bl} \quad (5')$$

이상에서 살펴본 바와 같이 연결구조 할당에 따른 클럭주기의 증가는 데이타 근원요소(레지스터와 연산기)와 데이타 전송자인 버스가 구동하는 부하량에 비례하므로, 최소의 클럭주기를 얻기 위해서는 연결구조 할당과정에서 가능한한 각 컴포넌트가 구동해야할 부하를 최소화해야하며, 이는 곧 $N_i^{rt}$, $N_i^{bl}$, $N_j^{ft}$, $N_j^{bl}$ 등을 최소화하므로써 가능해진다.

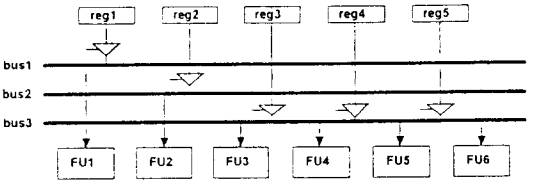
각 컴포넌트(레지스터, 연산기, 버스 등)의 부하량을 최소화하기 위해서는 연결구조 할당과정에서 각 컴포넌트 별로 부하량을 균등히 분배함으로써, 레지스터와 연산기 그리고 버스의 fanout 부하량이 최소가 되도록 해야한다[9]. 그림 2는 조합논리 블럭인 연산기의 수행시간이 모두 같은 경우의 데이타패스에서, read phase에서의 버스의 부하분배가 클럭주기에 미치는 영향을 보여주는 예이다. 그림 2(a)는 통신경로들의 예로 데이타의 근원지와 목적지를 가지고 있으며, ( ) 안은 그 경로가 사용되는 제어구간들을 나타낸다. 그림 2(b)는 그림 2(a)의 통신경로들에 대해서 각 제어구간별로 수행되는 통신경로들을 보인다. 그림 2(c)는 버스의 부하분배를 고려하지 않고 버스를 할당한 경우로서 $N_1^{bl}$와 $N_2^{bl}$는 1이고, $N_3^{bl}$, $N_4^{bl}$, $N_5^{bl}$, $N_6^{bl}$은 모두 4이므로 $\max_i(N_i^{bl})$ 은 4가 되고, 그림 2(d)는 각 버스의 부하가 균등히 분배되도록 할당한 경우로서 모든 경로에 대해서 버스의 부하가 2이므로 $\max_i(N_i^{bl})$이 2가 된다. 따라서 그림 2(d)와 같이 부하분배를 고려하는 경우에 버스가 구동해야할 부하가 감소하므로 클럭주기가 감소됨을 알 수 있다.

$C_1^R$ : reg1 →FU1(cstep1 cstep3)
$C_2^R$ : reg2→FU2(cstep1 cstep2 cstep4)
$C_3^R$ : reg4→FU6(cstep1)
$C_4^R$ : reg5→FU5(cstep2)
$C_5^R$ : reg3→FU3(cstep2)
$C_6^R$ : reg3 →FU6(cstep4)

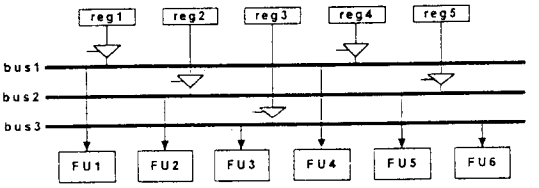
(a)

cstep1 : reg1→FU1, reg2→FU2, reg4 →FU4
cstep2 : reg2→FU2, reg3→FU3,
cstep3 : reg1→FU1, reg5→FU5,
cstep4 : reg2 →FU2, reg3→FU6,

(b)




(c)




(d)

그림 2. 부하분배를 고려한 버스 할당
(a) 통신경로 (b) 각 제어구간에 존재하는 통신경로 (c) 버스의 부하분배를 고려하지 않은 할당 (d) 각 버스의 부하가 균등히 배분된 할당

Fig. 2. Bus biniding considering load balance. (a) Communication paths (b) Communication paths for each control step (c) Binding without considering load balance (d) Binding obtained by considering load balance.

그러나, 실제적으로 클럭주기와 관계있는 부하는 임계경로 상에 존재하는 컴포넌트들의 부하이므로, 전체 컴

포넌트들에 대해 분배된 부하는 클럭주기와는 무관할 수 있거나, 오히려 클럭주기를 증가시킬 수도 있다. 연산기의 수행시간이 같지 않은 경우의 클럭주기의 결정에 관해 살펴보기 위해, 그림 2에서 FU1과 FU2가 40ns의 수행 지연시간을 갖는 곱셈기이고 나머지는 10ns의 덧셈기로 가정할때, 임계경로는 FU1, FU2의 수행시간이 dominant하여 이들을 경유하는 통신경로 $C_1^R$, $C_2^R$중에서 결정되므로 $C_1^R$, $C_2^R$에 무관한 bus3의 부하는 클럭주기와는 무관하게 된다. 따라서 이 경우에 부하분배 대상 버스를 $C_1^R$, $C_2^R$들이 할당된 bus1과 bus2로 제한하면 그 할당 결과는 그림 2(c)와 같이 되고, 이 때 $\max(N_i^{bl})$은 4이지만 실제 클럭주기에 관계되는 버스의 부하는 $\max(N_1^{bl}, N_2^{bl})$으로 그 값은 1이 되므로, 그림 2(d)와 같이 전체 버스에 대해서 부하를 분배함으로 인해 $\max(N_1^{bl}, N_2^{bl})$가 2가 되는 경우보다 최소화된 클럭주기를 얻을 수 있음을 알 수 있다. 결국 성능을 최대화하기 위한 연결구조 할당은 임계경로를 형성하는 read phase와 write phase의 통신경로 $C_i^R$, $C_j^W$에 대해서 $N_i^{rt}$, $N_j^{ft}$, $N_i^{bl}$, $N_j^{bl}$등의 값을 최소화 하는 문제로 귀착된다.

$C^R$과 $C^W$의 통신경로들 중 임계경로를 형성할 가능성이 있는 경로들의 부분집합을 각각 $C_c^R$, $C_c^W$, 임계경로 형성 가능성이 없는 경로들의 부분집합을 $C_{nc}^R$, $C_{nc}^W$라 하고,(이들의 분할 방법은 다음 절에서 기술한다.) 통신경로 $c_i^R$, $c_j^W$에 대해서 src(·)를 그 경로의 데이타 근원지, dest(·)를 데이타 목적지라 하면, 클럭주기의 결정과 관계되는 부하량 $N_{\max}^{rt}$, $N_{\max}^{ft}$, $N_{\max}^{bl}$는 아래의 식 (6)(7)(8)과 같이 정의된다.

$$N_{\max}^{rt} = \max(N_i^{rt}) \quad where\ c_i^R \in C_c^R \tag{6}$$

$$N_{\max}^{ft} = \max(N_j^{ft}) \quad where\ c_j^W \in C_c^W \tag{7}$$

$$N_{\max}^{bl} = \max(N_i^{bl}) + \max(N_j^{bl}) \tag{8}$$
$$where\ c_i^R \in C_c^R,\ c_j^W \in C_c^W,\ dest(c_i^R) = src(c_j^W)$$

따라서 제안된 연결구조 할당 알고리듬에서는 집합 $C_c^R$와 $C_c^W$내의 통신경로들에 대해서만 부하분배를 수행하고, 집합 $C_{nc}^R$, $C_{nc}^W$에 대해서는 면적비용만을 최소화한다.

## Ⅲ. 연결구조 할당 알고리듬

본 연결구조 할당 알고리듬은 연산기 할당과 레지스터 할당이 완료된 후 추출되는 통신경로들을 입력으로 하여 수행된다. 한 연산의 수행에 있어서, read phase에서는 레지스터로부터 연산기의 각 입력포트로 데이타를 전달하는 2개의 통신경로가 존재하고 write phase에서는 연산기의 출력포트로부터 레지스터로 연산의 결과를 적재하는 최소한 1개의 통신경로가 존재하게 된다. 대부분의 경우 read phase의 통신경로의 수가 write phase의 경우보다 많고 부하분배의 대상이 되는 컴포넌트도 read phase의 경우가 더 많으므로, 제안된 알고리듬은 read phase와 write pahse의 순으로 통신경로의 할당을 수행한다. 그림 3은 연결구조 할당 알고리듬의 개관을 보여준다. 본 알고리듬은 전처리 및 초기 할당, read phase 바인딩, write phase 바인딩으로 구성되며 각 과정은 다음 절에서 자세히 기술한다.

### 1. 전처리 및 초기 할당

#### 1) 통신경로 수의 최소화

교환법칙이 가능한 연산기에 있어서, 연산기의 각 입력포트에 데이타를 전달하는 통신경로들 중 서로 다른 제어구간에서 동일한 레지스터로부터 양 포트에 모두 데이타가 전달되는 통신경로들은 교환법칙을 적용, 연산기의 한 포트로 병합하므로써 통신경로의 수를 줄일 수 있다. 이는 2-color 그래프 채색 알고리듬을 사용하여 최적의 결과를 얻을 수 있으며 해당 연산기에 대해서는 mux 입력 갯수가 최소화된다[13], [14]. 또한 이 과정의 결과는 각 레지스터가 데이타를 전달하는 목적지의 수를 줄이게 되므로, 이후의 부하분배를 고려한 버스의 할당에 있어서 레지스터가 구동해야할 TSB의 수, $N_i^{rt}$를 감소시킨다.

#### 2) 이중 분할

앞 절에서 기술한 바와 같이 본 알고리듬은 임계경로를 형성할 가능성이 있는 통신경로들에 대해서만 부하분배를 수행한다. 따라서 버스 할당에 앞서서 각 phase의 통신경로들의 집합 $C^R$, $C^W$에 대해서 이중분할을 수행하여 각각 $C_c^R$와 $C_{nc}^R$, $C_c^W$와 $C_{nc}^W$으로 된 부분집합을 구성한다. 기억소자들이 동일한 지연시간을 갖는 레지스터일 경우, 통신경로들을 분할하는 기준은 그 통신경로가 경유하는 연산기의 수행 지연시간이 된다. 각각 40ns와 10ns의 지연시간을 갖는 곱셈기와 덧셈기가 사용되는 데이타패스의 경우, 덧셈기를 경유하는 경로에 많은 부하가 존재하더라도 임계경로를 형성할 가능성은 매우 적고, 곱셈기를 경유하는 경로중에서 임계경로가 형성되게 된다.

제안된 알고리듬에서는 가장 지연시간이 긴 연산기 $fu_{\max}$를 선정하고, 각 연산기에 대해 그 수행 지연시간

과 그 연산기를 경유하는 경로가 최악의 경우 갖게되는 부하에 의한 지연시간의 합이 $t_p(fu_{\max})$보다 큰 값인지의 여부에 따라 그 연산기를 경유하는 경로가 임계경로를 형성할 가능성이 있는지를 판별하는 연산기의 특성함수 $\Psi(fu_i)$를 정의한다. 정의된 함수값에 따라 임계경로를 형성할 가능성이 있는 연산기는 $FU_c$집합에 포함시키고 그렇지 않은 경우의 연산기는 $FU_c$집합에 포함시킨다. 이를 바탕으로 $FU_c$집합내의 연산기를 경유하는 통신경로들은 그 phase에 따라 $C_c^R$, $C_c^W$에, $FU_{nc}$내의 연산기를 경유하는 통신경로들은 $C_{nc}^R$, $C_{nc}^W$에 포함시킨다.

```
Interconnection Binding ()
{
 /* Preprocessing and initial binding */
 Minimize the number of connections using 2-color graph coloring :
 Classify C_c^R, C_c^W into C_c^R, C_nc^R and C_c^W, C_nc^W respectively:
 Group the communication paths in C_c^R, C_c^W with the same data
 sources:
 Sort the groups in decreasing order of the number of destinations:
 Perform initial bus binding :

 /* read phase binding */
  /* minimize load of register and bus in read phase */
 for each group in C_c^R do
   for each bus in the paths belonging to the group do
     Calculate objective function:
   Bind the paths in the group to the bus with maximum objective
   function:
 end for:
 Perform postprocessing for minimun load balancing:
 /* minimize the number of mux inputs */
 for each path C_nc^R, in C_nc^R do
     Bind C_nc^R, to a bus or buses such that min. # of mux inputs be
     required:
 end for:

 /* write phae binding */
   /* minimize load of FU and bus in write phase */
 for each path c_c^W, in C_c^W do
     Bind c_c^W, to the bus with minimum load of FU:
 end for:
 for each path c_nc^W, in C_nc^W do
     Bind c_nc^W, to the bus with minimun mux inputs:
 end for:
}
```

그림 3. 연결구조 할당 알고리듬의 개관
Fig. 3. An overview of the interconnection bindiding algorithm.

임의의 연산기를 경유하는 경로가 최악의 경우 갖게되는 최대 부하량은 그 연산기에 데이타를 전달하는 read phase의 경로들과 그 연산기의 결과를 적재하는 write phase의 경로들에 의해 가능한 최대의 부하량들의 합으로서 결정된다. Read phase의 경로들에 의해 가능한 최대의 부하량은 그 연산기에 데이타를 전달하는 레지스터중 데이타를 전달하는 목적지의 수가 가장 많은 레지스터가 그 목적지의 수 만큼의 서로 다른 버스에 바인딩되는 경우에 그 레지스터가 갖는 부하량으로, 이 경우 연산기에 데이타를 전달하기위해서는 해당 레지스터가 그 출력단에 연결된 버스의 수만큼의 버퍼를 구동해야한다. 연산기의 연산결과를 적재하는 write phase의 경로들에 의해 가능한 최대 부하량은 그 연산기가 데이타를 적재하는 모든 경로들이 서로 다른 버스로 바인딩되는 경우에 연산기가 갖는 fanout 부하량으로, 이 경우에 연산의 결과가 적재되기 위해서는 해당 연산기가 데이타를 적재하는 목적지(레지스터) 수 만큼의 버스를 구동해야한다. 따라서 연산기의 특성함수는 임의의 연산기를 경유하는 경로들이 최악의 경우 갖게 되는 부하에 의한 지연시간과 그 연산기의 수행지연시간의 합이 $t_p(fu_{\max})$를 초과할 수 있는지를 판별할 수 있도록 정의된다. 이 때 그 함수값이 양수인 경우 그 연산기는 임계경로를 형성할 가능성이 없는 특성을 갖게 되고, 그렇지 않은 경우에는 임계경로를 형성할 가능성이 있게되므로 그 연산기를 경유하는 경로들에 대해서는 부하분배의 고려가 필요하다. 레지스터나 연산기에 대해서 Num_dest( · )를 그들이 데이타를 전달하는 목적지의 수로 정의하고, src_set( · )를 레지스터나 연산기의 데이타 근원지의 집합으로 정의하면, 연산기의 특성함수 $\Psi(fu_k)$는 아래의 식 (9)와 같이 정의된다.

$$\Psi(fu_k) = t_p(fu_{.ax}) - t_p(fu_k) - \alpha \cdot (Nm\_dest(fu_k) + \max_{s \in src\_set(fu_k)} (Nm\_dest(s))) \quad (9)$$

where $\alpha$ is a empirical tuning parameter.

3) 통신경로의 grouping 및 정렬

$C_c^R$와 $C_c^W$에 있어서 한 데이타의 근원지로부터 여러개의 목적지로 데이타가 전달되는 통신경로들에 대해서는 여러가지 버스 할당 방법이 가능하다. 그림 4는 서로 다른 제어구간에서 한 레지스터로부터 여러개의 연산자로 데이타가 전달되는 통신경로들에 대해 가능한 버스 할당의 예이다. 그림 4 (a)의 세 통신경로에 대해서 그림 4 (b)는 세 통신경로가 하나의 버스로 할당된 경우이고, 그림 4 (c)는 각각 서로 다른 버스에 할당된 경우로 $N_{\max}^{rl}$와 $N_{\max}^{bl}$는 각각 1,3 과 3,1이 된다. 그림 4 (d)는 곱셈기인 ful을 경유하는 경로의 버스의 부하와 레지스터 reg1의 부하를 최소화한 할당으로 $N_{\max}^{rl}$는 2이고 $N_{\max}^{bl}$는 1이 된다. 이는 같은 근원지를 갖는 통신경로들에 대해서 $FU_c$의 연산기를 경유하는 경로에 존재하는 버스의 부하를 우선적으로 최소화한 후, $N_i^{rl}$나 $N_j^{fl}$를 최소화하므로써 얻어진 결과이다. 따라서 부하분배를 고려한 버스할당에 있어서 같은 데이타 근원지를 갖는 통신경로들은 함께 처리할 필요가

있고, 또한 목적지의 수가 많을수록 임계경로를 형성할 가능성이 높으므로 부하분배의 과정에서 우선적으로 처리되기 위해서, $C_r^R$와 $C_r^W$의 통신경로들은 같은 근원지를 갖는 경로들끼리 인접하도록 grouping하고, 그 목적지의 수가 많은 순으로 정렬한다.

1: reg1 * fu1 (*)
2: reg1 * fu2 (+)
3: reg1 * fu3 (+)

(a)

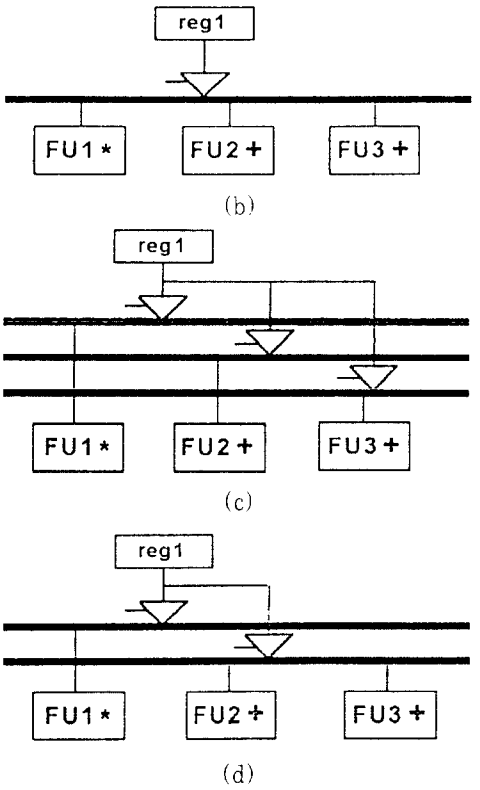



그림 4. 여러가지 버스 바인딩 방법
(a) 통신경로 (b) 한 개의 버스로 바인딩 경우 (c) 세 개의 버스로 바인딩한 경우 (d) 두 개의 버스로 바인딩한 경우

Fig. 4. Various bus bindings.
(a) Communication paths. (b) Binding using one bus. (c) Binding with three buses. (d) Binding with two buses.

그림 5는 elliptic wave filter의 동작기술에 대해 제어구간 18로 스케쥴링되고, 모듈할당이 수행된 후, 전처리 과정의 결과로 얻어진 $C_r^R$의 통신경로들의 예로서 동일한 근원지(레지스터)에 대해 grouping 되어 있고, 목적지의 수가 큰 순으로 정렬되어 있음을 볼 수 있다.

| 통신 경로# | 통신경로 | 초기바인딩 된 버스 | 경로가 존재하는 제어구간 |
|---|---|---|---|
| 1 | reg6 → fu2.1 | bus3 | 1 2 12 |
| 2 | reg6 → fu1.1 | bus6 | 3 6 |
| 3 | reg6 → fu2.2 | bus2 | 7 17 18 |
| 4 | reg6 → fu1.2 | bus5 | 8 16 |
| 5 | reg6 → fu3.2 | bus1 | 9 13 |
| 6 | reg4 → fu2.2 | bus2 | 2 12 13 |
| 7 | reg4 → fu3.2 | bus1 | 4 5 8 |
| 8 | reg4 → fu2.1 | bus3 | 7 11 |
| 9 | reg4 → fu1.2 | bus5 | 15 |
| 10 | reg5 → fu2.1 | bus3 | 13 |
| 11 | reg5 → fu1.1 | bus6 | 14 |
| 12 | reg5 → fu3.2 | bus1 | 15 |
| 13 | reg5 → fu1.2 | bus5 | 17 18 |
| 14 | reg3 → fu1.2 | bus5 | 2 3 6 9 |
| 15 | reg3 → fu1.1 | bus6 | 7 8 10 11 |
| 16 | reg3 → fu3.2 | bus1 | 14 |

그림 5. $C_r^R$의 통신경로들의 예

Fig. 5. An example of communication paths in $C_r^R$.

4) 버스할당과 초기 바인딩

연결구조 구성에 필요한 최소의 버스의 수는 가장 많은 통신경로가 존재하는 제어구간 $cstep_{max}$에서 그 통신경로의 수로써 결정된다. 따라서 $cstep_{max}$에 존재하는 통신경로들은 버스의 수를 구함과 동시에 할당이 이루어지게 되고, 할당된 각 버스는 출력단에 한 개의 연산기가 연결되게 된다. Read phase의 통신경로들에 대해서 같은 연산기를 목적지로 가지는 버스로 바인딩을 수행할 경우 부가적인 mux 입력을 갖지않게 되므로, 버스의 할당과 $cstep_{max}$에 존재하는 통신경로들의 바인딩이 수행되고난 후 모든 통신경로들에 대해 같은 목적지를 갖는 버스로써 초기 바인딩을 수행한다. 다음 스텝인 read phase 바인딩 과정에서는 이를 토대로 부하분배를 고려하여 재바인딩을 수행한다.

그림 5에는 초기 바인딩된 버스와 각 통신경로가 존재하는 제어구간들이 나타나있다. 모듈할당 결과 40nsec의 수행 지연시간을 갖는 2단 파이프라인 곱셈기 1개($fu_3$)와 10nsec의 덧셈기 2개($fu_1$, $fu_2$)가 사용되었다면, 전술한 연산기 특성함수 $\Psi(fu_k)$에 의해 $FU_c = \{fu_3\}$이고 $FU_{nc} = \{fu_1, fu_2\}$이 된다. 클럭주기의 최소화를 위해 $FU_c$의 입력단에는 가능한한 mux가 구성되지 않도록 해야한다. 따라서 그림 5의 경로들 중 $FU_c$ 내의 연산기인 $fu_3$를 경유하는 경로들은 같은 연

산기가 출력단에 연결된 버스로 바인딩된 초기바인딩 결과를 유지해야한다.

2. Read phase 바인딩

Read phase 바인딩의 목표는 모든 통신경로 $c_i^R$에 대해서 $N_i^{rt}$와 $N_i^{bl}$를 최소화하는 것이다.

따라서 우선적으로 $C_c^R$내의 같은 근원지를 가지는 통신경로 group에 대해서 초기 바인딩 결과를 바탕으로 $N_{\max}^{rt}$ 와 $N_{\max}^{bl}$이 최소가 되도록 버스의 재바인딩을 수행하고, $C_{nc}^R$에 대해서는 면적비용 (mux 입력의 수)을 최소화한다. 각 통신경로의 초기 바인딩된 버스는 $cstep_{\max}$에 존재하는 통신경로들에 의해 결정된 버스들에 대해서 같은 연산기로 데이타를 전송하는 버스로서 정해졌으므로, 재바인딩시 다른 버스로 바인딩되면 연산기 입력단에 mux를 필요로 하게된다. 클럭주기를 최소화하기위해서는 가능한한 $FU_c$에 속하는 연산기들의 입력단에는 mux가 필요하지 않도록 해야한다. 따라서 $FU_c$에 속하는 연산기를 경유하는 통신경로들은 재바인딩의 대상에서 제외한다. 또한 $cstep_{\max}$에 존재하는 통신경로들은 초기에 바인딩된 버스이외에 다른 버스로의 바인딩은 불가능하므로 역시 재바인딩의 대상에서 제외한다.

그림 4에서 보인바와 같이 최소의 $N_i^{rt}$를 위해서는 $FU_{nc}$에 속하는 연산기를 경유하는 경로들에 대해서는 가능한한 적은 수의 버스로 병합되도록 재바인딩을 수행해야한다. 그림 5에서 레지스터 reg6을 데이타의 근원지로 하는 통신경로들의 group( $c_1^R$, $c_{c,2}^R$, $c_{c,3}^R$, $c_{c,4}^R$, $c_{c,5}^R$)을 살펴보면, reg6이 데이타를 전달하는 버스의 수는 reg6의 목적지의 수 Num_Dest(reg6) = 5 와 같으므로 초기 바인딩 결과의 $N_i^{rt}$은 5가 된다. 곱셈기인 $fu_3$을 경유하는 $c_{c,5}^R$는 그 입력단에 mux가 필요하지 않도록 초기 할당 결과가 유지되어야 하므로 이를 제외한 나머지 4개의 통신경로들의 초기할당 버스들에 대해서, 4개의 통신경로가 한 버스로 바인딩 될때 감소되는 $N_i^{rt}$의 값을 목적함수로 하여 그 함수값이 최대가 되는 버스를 선정, 이 버스로서 group내 경로들의 재바인딩을 수행한다. $c_1^R$의 초기할당 버스인 bus3으로 $c_{c,2}^R$, $c_{c,3}^R$, $c_{c,4}^R$를 재바인딩 하는 경우를 살펴보면, $c_{c,2}^R$의 제어구간(cstep3, cstep6)과 $c_{c,4}^R$의 제어구간(cstep8, cstep16)에 대해 초기할당 결과 bus3이 사용된 제어구간(cstep2, cstep7, cstep11, cstep12, cstep13)이 겹치지 않으므로 $c_{c,2}^R$와 $c_{c,4}^R$는 bus3으로 바인딩될 확률이 1이라고 할 수 있고, $c_{c,3}^R$에 대해서는 cstep7에서 bus3이 $c_{c,8}^R$에 초기 바인딩되어 있으므로 바인딩될 확률은 1이 될 수 없으나, $c_{c,8}^R$이 $FU_c$의 연산자를 경유하거나 $cstep_{\max}$에 존재하는 통신경로가 아니기 때문에 $c_{c,8}^R$도 다른 버스로 재바인딩 될 수 있으므로 그 확률은 0.5가 된다. 이 확률들을 모두 더한 값을 목적함수로 할 때, 목적함수의 값은 2.5가 되며 이는 감소될 수 있는 $N_i^{rt}$값으로 볼 수 있다. 이와 같은 방식으로 $c_{c,2}^R$, $c_{c,3}^R$, $c_{c,4}^R$의 초기할당 버스에 대해서도 목적함수를 구하고 그들 중 최대값을 갖는 버스로 group내 경로들의 재바인딩을 수행한다. 이 때 그 확률이 1인 경로에 대해서만 재바인딩하고 확률이 0.5인 경로는 $C_c^R$ 전체에 대해 위와 같은 과정을 수행한 후, 0.5의 재바인딩 확률을 갖는 모든 경로들에 대해 $N_{\max}^{rt}$의 감소가능 여부에 따라 재바인딩을 수행한다.

통신경로 $c_{c,i}^R$와 동일한 레지스터를 데이타의 근원지로 갖는 $C_c^R$내의 부분집합을 same_src( $c_{c,i}^R$)라 하고, 집합 same_src( $c_{c,i}^R$)내 각 경로의 초기할당 버스를 $bus_m$으로 표기하자. same_src( $c_{c,i}^R$)내의 한 경로가 버스 $bus_m$으로 재바인딩될 수 있는지의 여부를 나타내는 정규화된 확률 함수 Prob( $c_{c,i}^R$, $bus_m$)를 $c_{c,i}^R$의 제어구간이 $bus_m$과 겹치지 않을 경우 1의 함수값을 갖고, 제어구간이 겹치지만 그 제어구간들이 $cstep_{\max}$를 포함하지 않거나, 그 겹치는 제어구간에 $bus_m$으로 초기 바인딩된 경로가 $FU_c$내의 연산기를 경유하지 않는 경우 0.5의 함수값을 가지며, 그외의 경우에는 0의 함수값을 갖도록 정의한다. 이 때, same_src( $c_{c,i}^R$)의 모든 경로들이 그들 중의 한 초기할당 버스 $bus_m$으로 병합 할당되는 경우의 목적함수는 아래의 식 (10)과 같이 정의된다.

$$\Phi_c^R(same\_src(c_{c,i}^R), bus_m) = \sum_{c \in same\_src(c_{c,i}^R)} Prob(c, bus_m) \qquad (10)$$

그림 6은 그림 5의 $C_c^R$ 전체에 대해 재바인딩이 수행된 후 재바인딩 확률이 0.5인 경로들로 구성된 충돌 그래프(conflict graph)이다. 각 경로에 대해 좌측에 존재하는 노드는 초기할당된 버스를 나타내고, 우측의 노드는 $N_i^{rt}$가 감소될 수 있는 재바인딩 가능한 버스를 나타내며, 이 두 노드는 상호배제(mutual exclusive) 관계에 있으므로 충돌에지로 연결된다. 이때 좌측의 노드들은 각 경로에 대해 초기 바인딩된 버스노드이므로 좌측 노드들 간에는 충돌에지가 존재할 수 없고, 각 경

로에 대한 재바인딩 가능 버스를 나타내는 우측의 노드들간이나 서로 다른 경로에 대한 좌,우측의 노드들간에는 동일한 버스이고 그 경로의 제어구간이 겹치는 경우에 충돌에지가 존재한다. 후처리 과정은 이러한 충돌 그래프에서 각 경로당 존재하는 두 버스노드 중 한 버스노드를 선택해서 재바인딩하는 과정으로 이 과정이 수행되고나면 read phase 바인딩은 완료된다.

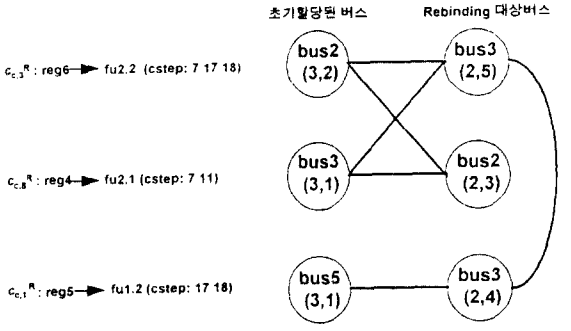



그림 6. 재바인딩 확률이 0.5인 경로들에 대해 구성된 충돌그래프의 예

Fig. 6. An example conflict graph for communication paths having rebinding probability of 0.5

그림 6에서 각 노드에 표기된 ( )안의 좌,우의 숫자는 주어진 경로가 그 버스로 바인딩되었을 경우 그 결과에 의한 $N_i^w$와 mux 입력의 수를 나타내는데, 그림에서 볼 수 있듯이 각 경로에 대한 좌,우의 두 노드는 $N_i^w$와 mux 입력의 수에 대해서 tradeoff의 관계가 있다. 본 과정은 $N_{\max}^w$의 최소화를 목적으로 하므로, 각 경로에 대해 바인딩할 한 버스노드를 선택함에 있어서 우측 노드의 선택이 좌측의 노드보다 $N_i^w$값이 작으므로 유리하게 된다. 그러나, 그림 5의 예와 같이 각 경로에 대해서 $N_i^w$ 측면에서 유리한 우측의 재바인딩 가능한 노드들 간에 에지가 존재하는 경우에는 한 경로에 대한 우측 버스노드로의 재바인딩으로 인해 그 노드와 충돌에지로 연결된 나머지 경로는 우측의 재바인딩 가능 버스노드를 선택할 수 없으므로 초기할당 버스가 그대로 유지되어야 한다. 따라서 일부의 경로에 대해서는 $N_i^w$가 감소하나 전체적으로 $N_{\max}^w$는 감소되지 않으므로, 이러한 경우는 그래프의 모든 경로에 대해 좌측의 노드를 선택하므로써 mux 입력의 수를 최소화하는 것이 면적비용 측면에서 유리하게 된다.

$C_{mc}^R$의 바인딩에 있어서는 $C_i^R$의 재바인딩의 영향을 받아서 초기할당된 버스를 상실한 통신경로들에 대해서만 수행된다. 이러한 경로들은 같은 연산자를 출력단에 가진 버스로의 바인딩은 불가능하므로 부가적인 mux 입력의 수가 최소화되도록 바인딩을 수행한다. 다만, 가 증가하지 않도록 출력단에 의 연산자를 가지는 버스들은 그 대상에서 제외한다. 통신경로 의 전 제어구간에 대해서 그 제어구간이 겹치지 않는 버스가 있는 경우 부가적인 mux 입력의 수는 1이 되므로 우선적으로 바인딩하고, 그러한 버스가 존재하지 않는 경우에는 최소의 버스들로 그 통신경로가 바인딩되도록 하며 이 경우에는 바인딩에 필요한 버스의 수만큼 mux 입력의 수가 필요하게 된다.

3. Write phase 바인딩

Read phase의 통신경로들에 대한 바인딩이 수행된 후 각 버스의 입력단과 출력단에는 각각 레지스터와 연산기가 연결되어 있으므로, write phase의 연결구조 할당에서는 read phase에서의 바인딩 결과를 토대로 임계경로의 지연시간이 최소화되도록 $C^W$의 각 경로들을 버스에 바인딩한다. $C_i^W$의 경로들은 $FU_i$내의 연산기의 연산결과를 레지스터에 저장하는 경로들이므로 연산기 출력포트의 부하와 바인딩할 버스의 부하가 최소가 되는 버스로 바인딩이 이루어져야 한다. 바인딩할 버스의 출력단에 연결된 컴포넌트가 임계경로 형성의 가능성이 있는 경우, 부가적인 레지스터의 연결로 의해 그 버스가 read phase에서 사용될때 부하가 증가하게 된다. 이로 인해 임계경로의 지연시간이 증가될 수 있으므로, 버스의 출력단에 연결된 컴포넌트의 수행시간도 고려되어야 한다. 따라서 $c_{c,j}^w$의 바인딩 버스를 결정하는 목적함수에 $c_{c,j}^w$의 데이타 근원지인 연산기의 부하량 $N_i^W$과 바인딩하고자 하는 버스의 fanout 부하량의 합 그리고 그 버스의 출력단에 연결된 컴포넌트들의 최대 지연시간을 반영한다. 이러한 목적함수를 사용하여 최소의 함수값을 갖는 버스로 바인딩하므로써 가능한한 임계경로의 지연시간이 증가않도록 바인딩을 수행할 수 있다. 주어진 통신경로의 바인딩 대상이 되는 m번째 버스 $bus_m$에 바인딩될 경우에 그 버스가 갖게되는 fanout 부하를 Load($bus_m$)라 하고, $bus_m$의 출력단에 연결된 컴포넌트 집합을 fanout($bus_m$)라 표기하면, $C_i^W$내의 통신경로 $c_{c,j}^w$을 $bus_m$에 할당하는 경우의 목적함수는 아래의 식 (11)과 같이 정의된다. 바인딩 과정에서 후보가 되는 모든 버스에 대하여 최소의 목적함수 값을 갖는 버스에 의해 바인딩을 수행한다.

$$\Phi_c^w((c_{c,j}^w), bus_m) = \beta_1 \cdot N_{fl,j} + \beta_2 \cdot Load(bus_m) \quad \beta_3 \cdot \max_{f \in fanout(bus_m)} (t_b(f)) \tag{11}$$

where $\beta 1$, $\beta 2$ and $\beta 3$ are empirical tuning parameters.

임계경로를 형성할 가능성이 $C_{nc}^{w}$없는 의 경로들에 대해서는 그 경로의 근원지인 연산기가 $FU_{nc}$내의 연산기이므로 연산기의 fanout 부하 $N_f^{fl}$는 고려될 필요가 없고, mux 입력 수의 증가가 최소가 되도록 바인딩을 수행한다. 바인딩하고자 하는 버스에 $FU_c$내의 연산기가 연결되어 있는 경우 임계경로의 지연시간이 증가될 수 있으므로 $C_c^{w}$의 경우와 같이 가능한한 임계경로의 지연시간이 증가되지 않도록 바인딩하고자 하는 버스의 출력단에 연결된 컴포넌트들의 수행시간을 고려하여 바인딩해야한다. 따라서 $C_{nc}^{w}$의 목적함수에는 최소의 mux 입력의 수를 위해 주어진 $c_{nc,j}^{w}$와 바인딩하고자 하는 버스가 같은 레지스터를 목적지로 가지는 경우에 큰 가중치를 부여하는 인자와 해당 버스의 부하 그리고 그 버스의 fanout 컴포넌트들의 최대 지연시간 등이 반영되어야한다. Same_Dest( $c_{nc,j}^{w}$, $bus_m$)를 $c_{nc,j}^{w}$와 $bus_m$가 같은 레지스터를 목적지를 갖는 경우 1의 함수값을 갖고 그 외의 경우에 0의 함수값을 갖는 함수로 정의할때 $C_{nc}^{w}$바인딩의 목적함수는 아래의 식(12)와 같이 정의된다.

$$\Phi_{nc}^{w}(c_{nc,j}^{w}, bus_m) = \gamma 1 \cdot Same\_Dest(c_{nc,j}^{w}, bus_m) + \gamma 2 \cdot Load(bus_m) + \gamma \cdot \max_{f \in fanout(bus_m)} (t_p(f)) \quad (12)$$

## Ⅳ. 실험결과

본 절에서는 제안된 알고리듬의 성능을 검증하기 위해 기존의 연구에서 사용된 네 가지 벤치마크에 대한 실험결과를 제시한다. 제안된 알고리듬은 SODAS 시스템[15]의 연결구조 할당기로서 구현되었으며 표 1, 2, 3, 4는 제안된 알고리듬과 기존연구의 실험결과를 비교한 것이다. 기존연구의 실험결과는 참고문헌[9], [10]에서 인용하였다. 특히 부하분배를 고려한 연결구조 할당 알고리듬을 제시한 시스템인 참고문헌[9]와 동일한 조건에서 실험결과를 비교하기 위하여 기억소자로서 레지스터 파일을 사용한 타겟 아키텍춰로 합성하고[14], 각 컴포넌트당 지연시간의 증가량에 대해서도 참고문헌[9]에서 실험한 데이타를 사용하였다. 표에 제시된 SODAS 시스템의 CPU time은 59 MIPS의 성능을 가지는 Micro SPARC station에서 측정된 수행시간이다. 표 1과 표 2는 각각 5차 엘립틱 웨이브 필터를 제어구간 17과 제어구간 34에 대해 2단 파이프라인 곱셈기와 덧셈기를 사용하여 합성한 결과를 타시스템과 비교한것이다. 표의 bus 당 load는 클럭주기를 결정하는 임계경로상의 read phase와 write phase에서 사용되는 버스의 부하량을 나타낸다.

표 1. 5차 엘립틱 웨이브 필터에 대한 실험결과 (제어구간 17)

Table 1. Experimental result for the 5th-order elliptic wave filter with 17 control steps.

| | SODAS | Ref[9] ILP | Rer[9] Heuristic | FLORA |
|---|---|---|---|---|
| # buses | 6 | 6 | 6 | 6 |
| RF 당 load | 3 | 1 | 1 | 1 |
| bus 당 load (read/write) | 1/4 | 5/5 | 5/5 | 5/5 |
| FU 당 load | 1 | 2 | 2 | 2 |
| # TSBs | 20 | 9 | 11 | 9 |
| # mux inputs | 2 | 20 | 17 | 20 |
| # muxes | 1 | 8 | 7 | 8 |
| read time(nsec) | 3.4 | 3.8 | 3.8 | 3.8 |
| write time(nsec) | 3.5 | 4.6 | 4.6 | 4.6 |
| total time(nsec) | 6.9 | 8.4 | 8.4 | 8.4 |
| total time(%) | - | +21.7 | +21.7 | +21.7 |
| CPU time (sec) | 0.03 | 67 | 0.34 | 5.98 |

표 2. 5차 엘립틱 웨이브 필터에 대한 실험결과 (제어구간 34)

Table 2. Experimental result for the 5th-order elliptic wave filter with 34 control steps.

| | SODAS | Ref[9] ILP | Rer[9] Heuristic |
|---|---|---|---|
| # buses | 4 | 4 | 4 |
| RF 당 load | 2 | 2 | 2 |
| bus 당 load (read/write) | 2/3 | 4/4 | 4/4 |
| FU 당 load | 1 | 1 | 1 |
| # TSBs | 12 | 8 | 9 |
| # mux inputs | 5 | 3 | 8 |
| # muxes | 2 | 2 | 4 |
| read time(nsec) | 3.3 | 3.9 | 3.9 |
| write time(nsec) | 3.2 | 3.5 | 3.5 |
| total time(nsec) | 6.5 | 7.4 | 7.4 |
| total time(%) | - | +13.8 | +13.8 |
| CPU time (sec) | 0.05 | 17 | 0.85 |

표 1과 표 2에서 보듯이 제어구간 17으로 합성한 경우에는 제안된 연결구조 할당 알고리듬을 수행한

SODAS 시스템이 참고문헌[9]의 시스템과 FLORA[10] 시스템에 비해 부하에 의한 지연시간이 모두 21.7% 만큼 감소되었고, 면적측면에서는 SODAS 시스템이 타 시스템보다 많은 수의 버퍼가 사용된데 반해 mux 입력의 수는 현저히 감소된 결과를 나타냈다. 제어구간 34로 합성한 경우에는 참고문헌[9]의 시스템에 비하여 지연시간이 13.8% 만큼 감소되었고, 면적측면에서는 SODAS 시스템이 참고문헌[9]의 ILP 알고리듬보다 많은 수의 버퍼와 mux 입력이 사용됨을 보이고, 참고문헌[9]의 Heuristic 알고리듬에 비해 많은 수의 버퍼와 적은 수의 mux 입력이 사용되었다.

표 3. 16-point FIR 필터에 대한 실험결과
Table 3. Experimental result for the 16-point FIR filter.

| | SODAS | Ref[9] ILP | Rer[9] Heuristic | FLORA |
|---|---|---|---|---|
| # buses | 6 | - | 6 | 6 |
| RF 당 load | 2 | - | 1 | 3 |
| bus 당 load (read/write) | 1/4 | - | 4/4 | 5/5 |
| FU 당 load | 1 | - | 2 | 2 |
| # TSBs | 15 | - | 12 | 16 |
| # mux inputs | 4 | - | 18 | 10 |
| # muxes | 2 | - | 17 | 5 |
| read time(nsec) | 3.0 | - | 3.5 | 4.6 |
| write time(nsec) | 3.5 | - | 4.3 | 4.6 |
| total time(nsec) | 6.5 | - | 7.8 | 9.2 |
| total time(%) | - | - | +20.0 | +41.5 |
| CPU time (sec) | 0.04 | - | 0.14 | 3.4 |

표 3은 제어구간 8의 제약조건하에서 비파이프라인 곱셈기와 덧셈기를 사용하여 16 point FIR 필터를 합성한 결과로 참고문헌[9]의 ILP 방법은 해를 구하지 못했고, SODAS 시스템은 참고문헌[9]의 heuristic 알고리듬과 FLORA 시스템에 비하여 각각 20.0%와 41.5% 만큼 감소된 지연시간을 갖는 연결구조를 생성함을 보인다. 표 4는 제어구간 4의 제약조건으로 2차 미분방정식 산출기를 합성한 결과로 SODAS 시스템이 참고문헌[9]의 ILP 방법과 heuristic 알고리듬에 비해 각각 3.0% 와 12.1%만큼 지연시간이 감소된 결과를 보이고, FLORA와 HAL에 비해서는 모두 33.0% 만큼 감소된 지연시간을 갖는 연결구조를 생성함을 보인다. 면적측면에서는 표3의 FIR 필터나 표4의 미분방정식 산출기의 경우에 있어서 SODAS 시스템이 타 시스템과 비슷한 수의 버퍼를 사용하지만 mux 입력의 수에 있어서는 현저히 감소된 결과를 보임을 알 수 있다.

표 4. 2차 미분방정식 산출기에 대한 실험결과
Table 4. Experimental result for the 2nd order differential equation resolver.

| | SODAS | Ref[9] ILP | Rer[9] Heuristic | FLORA | HAL |
|---|---|---|---|---|---|
| # buses | 6 | | 6 | 6 | 6 |
| RF 당 load | 3 | 2 | 2 | 2 | 2 |
| bus 당 load | 1/3 | 3/3 | 4/4 | 5/5 | 5/5 |
| FU 당 load | 1 | 1 | 1 | 2 | 2 |
| # TSBs | 16 | 15 | 13 | 15 | 18 |
| # mux inputs | 2 | 5 | 8 | 6 | 13 |
| # muxes | 1 | 2 | 3 | 3 | 6 |
| read time(nsec) | 3.4 | 3.6 | 3.9 | 4.2 | 4.2 |
| write time(nsec) | 3.2 | 3.2 | 3.5 | 4.6 | 4.6 |
| total time(nsec) | 6.6 | 6.8 | 7.4 | 8.8 | 8.8 |
| total time(%) | - | +3.0 | +12.1 | +33.0 | +33.0 |
| CPU time (sec) | 0.03 | 63 | 0.05 | 1.25 | n/a |

실험결과, 제안된 연결구조 할당 알고리듬은 예상한 바와 같이 실험한 모든 경우에 대해서 부하에 의한 지연시간이 감소됨을 알 수 있었고, 연결구조의 면적 측면에 있어서는 전반적으로 타시스템에 비해 적은 mux 입력의 수를 보였으나, 버퍼의 수에 있어서는 증가된 특징을 보였다. 제안된 알고리듬이 mux 입력의 수에서 유리할 수 있었던 원인은 버스 바인딩시 가능한한 같은 목적지를 가지는 버스를 우선적으로 고려하는 인자가 목적함수에 반영되었기 때문으로 추정되며, 타 시스템에 비해 다소 많은 수의 버퍼가 사용된 결과를 보이는 이유는 제안된 알고리듬이 버스의 부하분배에 있어서 임계경로을 형성할 가능성이 있는 버스에 대해서는 최소의 부하를 분배하고 클럭주기 결정과 무관한 버스에 많은 부하를 분산시킴으로 인해 임계경로와 무관한 버스의 입력단에 연결되는 버퍼의 수가 증가하기 때문이다. 제안된 알고리듬이 타시스템에 비해 비교적 빠른 시간내에 감소된 클럭주기를 가지는 연결구조를 합성할 수 있었던 원인은 클럭주기 증가의 원인이 되는 부하량의 분배를 전체적으로 고려하지 않고, 임계경로를 형성할 가능성이 있는 경로들에 대해서만 최소의 부하를 갖도록 효과적으로 분배하는 접근방법을 사용했기 때문이다.

## V. 결 론

본 논문에서는 성능을 최대화하기 위한 연결구조 할당 알고리듬에 대해서 기술하였다. 클럭주기를 최소화하는 연결구조 할당에 있어서 모든 하드웨어 컴포넌트

의 부하를 고르게 분배하는 기존의 시스템에 비해, 본 논문에서 제안하는 알고리듬은 클럭주기가 임계경로에 의해서 결정되는 점에 착안, 임계경로를 형성할 가능성이 있는 통신경로들에 대해서만 부하분배를 고려하므로써 실제 클럭주기의 결정에 영향을 미치는 부하량을 보다 최소화되도록 고안되었다. 실험결과 제안된 알고리듬은 실험된 모든 경우에 대해서 보다 최소화된 클럭주기를 갖는 연결구조를 빠른 시간 내에 얻을 수 있었다. 이는 전체 경로들에 대해서 부하를 분배하지 않고, 임계경로를 형성할 가능성이 있는 경로들에 대해서만 최소의 부하를 갖도록 부하분배를 수행하고 임계경로를 형성할 가능성이 없는 경로들에 많은 부하를 분산시켰기 때문이며, 또한 부하분배의 대상경로를 제한하므로써 보다 적은 수행시간에 결과를 얻을 수 있었다.

본 논문에서 제안한 알고리듬은 연산기와 레지스터의 할당이 완료된 후 수행되어, 보다 최적의 결과를 얻기 위해서는 연산기와 레지스터 할당 과정에서 미리 임계경로를 형성할 가능성이 있는 수행시간이 긴 컴포넌트들에 대해서 부하가 최소가 될 수 있도록 할당을 수행하는 알고리듬의 개발이 필요하다. 또한 본 논문에서는 버스 자체의 지연시간은 고려하지 않고 부하를 구동하는데 따른 지연시간만을 고려하였으나 상위수준 합성의 결과가 보다 하위 레벨의 실제적인 설계에 정확하게 반영되기 위해서는 상위수준 단계에서 버스의 길이 등 연결구조의 면적에 대한 평가가 어느정도 정확히 이루어질 수 있는 연구가 필요하다.

## 감사의 글

본 연구는 서울대학교 반도체 공동 연구소의 교육부 반도체 분야 학술 연구 조성비(과제번호: ISRC 94-E-2030)에 의해 수행되었습니다.

## 참 고 문 헌

[1] D. Gajski, Silicon Compilation, Reading, MA: Addison Wesley, 1988.

[2] M. McFarland, A. Parker, and R. Composano, "Tutorial on high-level synthesis," in Proc. 25th Design Automation Conf., pp. 330-336, June 1988.

[3] D. Gajski, N. Dutt, A. Wu, S. Lin, High-level Synthesis, Kluwer Academic Publishers, 1992.

[4] De Micheli, Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994.

[5] P. Paulin, J. Knight, and E. Girzyc, "HAL: A multi-paradigm approach to automatic data path synthesis," in Proc.23nd Design Automation Conf., pp. 263-270, June 1986.

[6] C. Tseng and D. Siewiorek, "Automated synthesis of data paths in digital systems," IEEE Trans. Computer-Aided Design, vol. CAD-5, pp. 379-395, July 1986.

[7] F. Tsai and Y. Hsu, "Data path construction and refinement," in Proc. Int. Conf. Computer-Aided Design, pp. 308-311, Nov. 1990.

[8] L. Stok, "Interconnect optimization during data path allocation," in Proc. European Design Automation Conf., pp. 141-145, March 1990.

[9] Y. Jiang, T. Lee, T. Hwang, and Y. Lin, "Performance-driven interconnection optimization for microarchitecture synthesis," IEEE Trans. Computer-Aided Design, vol. 13, no. 2, pp. 137-149, Feb. 1994.

[10] T. Liu, and Y. Lin, "FLORA: A data path allocator based on branch-and-bound search," INTEGRATION, pp. 43-46, Mar, 1991.

[11] 임 완수, 박 재환, 황 선영, "상위수준 합성을 위한 컴포넌트 라이브러리의 설계," 한국 정보과학회 논문지, 30-A 권 12호, 1993년 12월, pp. 1867-1878.

[12] S. Su, V. Rao, and T. Trick, "A simple and accurate node reduction technique for interconnect modeling in circuit extraction," in Proc. Int. Conf. on Computer-Aided Design, pp. 270-273, Nov. 1986.

[13] 김 영노, 이 해동, 황 선영, "Graph Coloring에 기반을 둔 모듈할당 알고리듬," 한국 정보과학회 학술발표 논문집, 20권 2호, pp. 979-982, 1993년 10월.

[14] 이 해동, 김 영노, 황 선영, "다중포트 메모리를

지원하는 데이타패스 자동 합성 시스템의 설계," 대한 전자공학회 논문지, 31-A권 7호, pp. 117-124, 1994년 7월

[15] H. S. Jun, S. Y. Hwang, "Design of a pipelined datapath synthesis system for digital signal processing," IEEE Trans. VLSI Systems, vol. 2, no. 3, pp. 292-303, Sep. 1994.

## 저 자 소 개

**金 泳 鎭(準會員)**

1993년 2월 서강대학교 전자공학과 졸업. 1995년 2월 서강대학교 전자공학과 석사 취득. 1995년 3월~현재 LG기술연구소 재직중. 주관심분야는 high-level synthesis, Computer Architecture, DSP 설계 등임.

**黃 善 泳(正會員)**

1976년 2월 서울 대학교 전자공학과 졸업. 1978년 2월 한국과학원 전기 및 전자 공학과 공학 석사 취득. 1968년 10월 미국 Standford 대학 공학 박사 학위 취득. 1976년~1981년 삼성 반도체 주식회사 연구원. 1986년~1989년 Stanford 대학 Center for Integrated Systems 연구소 연구원, Fairchild Semiconductor Palo Alto Research Center 기술 자문. 1989년 3월~현재 서강대학교 전자공학과 부교수. 주관심분야는 CAD 시스템, Computer Architecture 및 Systems Design, VLSI 설계 등임.

**李 海 東(正會員)** 제 28권 A편 6호 참조