韓國 OA學會 論文誌
第4卷 第1號, 1999. 3.

99-4-1-1-2

ILP기법을 이용한 연산자할당 및 바인딩에 관한 연구

신 인 수*, 이 근 만**

A Study on Operator Allocation and Binding by ILP

In-soo, Shin*, Keun-man, Yi**

요 약

본 논문에서는 상위수준합성에서의 연산자 할당 및 바인딩을 위한 방법을 다루었다. 스케쥴링 후 각 제어스텝에 지정된 연산을 연산자에 할당하기 위한 새로운 방법을 제시하였다. 특히 최적의 할당 및 바인딩 결과를 얻기위해 ILP 기법을 이용하였다.

HAL시스템에서 사용된 Differential Equation DFG 모델을 실험대상으로 하여 본 연구의 효용성을 입증하였다.

Abstract

In this paper, we deals with operator allocation and binding in high-level synthesis. We proposed a new methode to allocation and bind for scheduled operations on each control step after scheduling. Especially, we take account ILP formulation in order to get optimal allocation and binding result.

Experiment was done on the HAL DFG model of Differential Equation, to show it's effectiveness.

* 제천기능대학 정보통신기술학과 전임강사
** 청주대학교 첨단공학부 교수
논문접수 : 99. 2. 27. 심사완료 : 99. 3. 29.

I. 서 론

동작기술(Behavioral Description)에서 구조적 기술(Structure Description)로의 변환하는 과정을 합성(Synthesis)이라한다. 합성과정은 동작기술과 생성된 하드웨어의 면적이나 지연시간등 여러 가지 제약조건을 입력으로하여, 제약조건들을 만족하면서 주어진 동작을 수행하는 하드웨어 구조를 찾아내는 것이다. 합성과정(Synthesis Process)은 하위(Physical)영역의 합성기술, 중위영역(Logical Domain)의 합성기술, 상위영역(High-level Domain)의 합성으로 세분화된다[3].

하위영역(Physical Domain)의 합성기술에서는 공간분할, 배치, 경로등이 다루어지고, 중위영역(Logical Domain)의 합성기술에서는 RT수준(Register-Transfer Level)의 HDL의 설계 및 컴파일링, FSM(Finite State Machine)의 설계와 이에 따른 논리합성 등이 다루어진다.

상위영역(High-level Domain)의 합성기술에서는 상위수준 하드웨어 언어의 설계 및 변환(Transformation), 설계공간의 탐색, 설계유형의 선택 및 기능 분할, 제어구간(Control step) 할당(Allocation) 및 하드웨어 할당등이 다루어진다. 각 합성기술의 표현방식은 하위영역과 중위영역보다는 상위영역의 표현방식이 포괄적(Global)이다. 따라서 반도체 설계에서 하위영역의 자동합성기술이 최적의 상태로 운용되기 위해서는 중위영역의 자동합성 기술이 요구된다. 또한 중위영역의 자동합성 기술이 최적의 상태로 운용되기 위해서는 상위영역의 자동합성 기술이 요구된다. 따라서 동일한 반도체 설계라 할지라도 상위영역에서 합성을 하는 것이 하위영역에서 합성을 하는 것 보다 이론적이고 체계적이다.

특히, 이러한 합성 과정중 상위영역에서의 합성과정인 상위수준합성(High Level Synthesis)은 실리콘 컴파일러에의한 VLSI 자동생성 과정의 하나로써, 설계하고자하는 하드웨어의 동작을 알고리즘 수준으로 기술한 HDL(Hardware Description Language)을 받아들여 RT (Register Transfer)수준의 데이터 패스 구조와 그 데이터 패스를 제어하기 위한 콘트롤러로 구성된 구조기술로 변환하는 과정을 의미한다.

알고리즘 수준의 동작기술은 HDL 분석과정을 거쳐 상위 수준에 편리한 중간형태로 변환되어 합성과정의 입력으로 사용된다.

RT수준의 데이터패스는 레지스터(Register), 연산자(Operator) 그리고 레지스터와 연산자간의 연결구조(Interconnection : MUX, BUS,..)로 구성되어있다. 주어진 하나의 동작 기술(Behavioral Description)로부터 그 기능을 수행하는 구조는 여러 가지가 존재한다.

상위수준합성(High Level Synthesis)과정에서는 설계 공간내의 구조 중에서 주어진 제약조건인 클럭싸이클, 칩 면적등을 만족하면서 특정비용을 최소화하는 구조를 찾아내는 것을 목표로 한다.

설계과정을 자동화함으로써 칩의 설계 시간을 줄일 수 있어 급변하는 반도체 시장에 대처하게 할 수 있고 칩 개발에 소요되는 비용중 대부분이 개발에 사용된다는 점을 감안할 때, 비용을 효과적으로 줄일 수 있다.

상위수준합성(High Level Synthesis) 시스템을 사용하여 설계가 가능한 구조를 넓게 탐색함으로서 최적의 하드웨어 구조를 찾을 수 있다.

이러한 자동합성 과정중에서 가장 중요한 분야로 여겨지는, 합성수순은 DFG(Data Flow Graph)의 생성 및 변환, 데이터패스(Data-path) 및 제어부 합성으로 이루어 질 수 있으며, 여기서 데이터패스합성은 다시 스케쥴링(Scheduling)과 하드웨어 할당(Allocation) 및 바인딩(Binding)과정으로 세분화 된다.

본 연구는 상위수준합성 과정중 중요한 데이터패스 합성에 대하여 다루었다. 특히, 연산자 할당 및 바인딩 문제를 중점적으로 다루었으며, 스케쥴링 결과를 기준으로 연산자, 레지스터(Register)의 수가 제한된 조건을 만족할 수 있는 최적의 결과를 얻기위한 ILP 표현법을 제안하였다.

2장에서는 연산자 할당 및 바인딩에 대한 관련이론과 ILP표현법을 기술하였으며, 3장에서는 실험모델에 대한 실험 결과를 나타내었으며 결론은 4장에서 다루었다.

Ⅱ.할당 및 바인딩

스케쥴링의 결과로부터 각 연산이 정확하게 수행하기 위해서는 연산 및 레지스터등에 대한 자원 할당이 이루어져야 한다. 자원 할당은 설계를 위해 요구되는 연산자, 레지스터, 멀티플렉서, 버스등의 수와 각 유형의 모듈의 수를 결정하는 작업이다. 바인딩은 변수를 레지스터에, 연산을 모듈에 맵핑하는 작업으로서, 할당과 바인딩은 상호의존하며, 스케쥴링, 연결비용, 지연시간의 결과에 달려있다.

본 논문에서는 다양한 자원 중에서 연산자와 레지스터에 대한 할당 및 바인딩에 대하여 ILP표현법에 의하여 설명한다.

1. 기능연산자 할당 및 바인딩

DFG상의 연산을 설계시 요구되는 기능연산자(Function Unit)를 할당하는 과정은 스케쥴링의 결과로부터 스케쥴된 연산의 유형(type)과 수를 결정하여, 기능연산자(Function Unit)에 바인딩하는 과정으로서 기능연산자의 수는 스케쥴링 결과로부터 연산 유형별(곱셈기, ALU)로 분류하여 소요 연산자를 결정한다. 결정된 연산자의 수를 제약조건으로하여 연산을 기능연산자에 할당한다. 연산을 기능연산자에 할당하려면 다음과 같이 할당 및 바인딩 과정을 거친다.

1.1 스케줄링 범위

임의의 제어스텝내에 사용된 모든 연산(Oi)이 주어진 제약조건을 만족하도록 스케쥴링이 수행되었을 때, 각 연산은 임의의 제어스텝에 스케쥴 되어야 한다. 즉, DFG상의 모든 연산은 스케쥴된 제어스텝내에 존재해야 연산자에 할당된다. 스케쥴된 제어스텝 내에 존재하는 연산은 지정된 제어스텝 내에서만 연산을 수행하기 시작해야 한다.

임의의 제어스텝에 스케쥴된 연산(Oi)이 해당 제어스텝에서 연산을 수행하기 시작하면 그 연산은 참("1")의 값을 갖고 그렇지 않으면 거짓("0")의 값을 갖는다.

이와 같이 스케쥴된 연산에 대한 제어스텝 내에서의 연산의 수행 여부는 다음과 같이 0과 1의 값을 갖는 다음과 같은 식으로 표현된다.

$$X_{il} = 1 \ , \ (i = 1,2,..n_{ops} \ ; \ l = 1,2,..L) \ \cdots \ (1)$$

아래의 그림 1과 4개의 제어스텝 내에서 모든 연산이 선후관계를 유지하며, 스케쥴링이 끝났을 때의 결과 DFG이다. 〈표 1〉과 [그림 2]는 [그림 1]의 DFG상의 모든 연산에 대한 스케쥴링 결과 및 소요자원 그리고 할당 및 바인딩 결과이다.

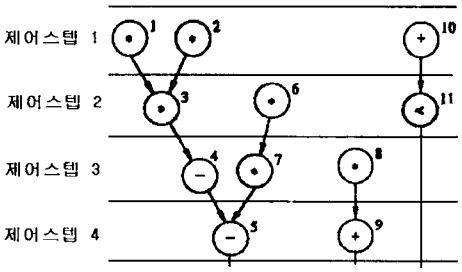

그림 1. DFG 모델
Fig. 1. DFG Model

표 1. 스케쥴링 결과 및 소요자원
Table 1. The scheduling result and resources

제어스텝	연산	소요자원
1	O1,O2,O10	* : 2개 ALU: 2개
2	O3,O6,O11	
3	O4,O7,O8	
4	O5,O9	

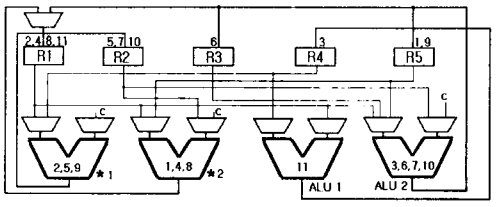

그림 2. 할당 및 바인딩 결과
Fig. 2. The result allocation and binding

그림 1의 DFG상의 모든 연산(O1, O2, O3, O4, O5, O6,O7, O8,O9,O10,O11)에 대한 스케쥴링 결과를 관계식(1)에 적용하면,

X11=1, X21=1, X32=1, X43=1, X54=1, X62=1, X73=1, X83=1, X94=1, X101=1, X112=1

의 값을 가지며 다음과 같은 의미를 내포한다.

연산O1은 제어스텝 1에 지정.
연산O2은 제어스텝 1에 지정.
연산O3은 제어스텝 2에 지정.
연산O4는 제어스텝 3에 지정.
연산O5는 제어스텝 4에 지정.
연산O6은 제어스텝 2에 지정.
연산O7은 제어스텝 3에 지정.
연산O8은 제어스텝 3에 지정.
연산O9는 제어스텝 4에 지정.
연산O10은 제어스텝 1에 지정.
연산O11은 제어스텝 2에 지정.

1.2. 연산유형 및 연산자 수 결정

표 1과 그림 1의 스케쥴링 결과로부터 알 수 있듯이 연산유형을 곱셈 연산과 ALU연산으로 분류하였을 경우, 연산에 사용될 연산유형 및 연산의 수는 2개의 곱셈기(*)와 2개의 ALU가 소요됨을 알 수 있다.

즉, 연산 유형(opertion type)을 곱셈기, ALU(+,/,()로 분류하였을 경우, 분류된 유형의 연산이 DFG상의 연산이 수행되기 위한 최소의 연산 개수를 결정한다. 그러므로 사용될 기능연산자(fumction unit) 유형 및 개수는 각각 곱셈기 2개, ALU 2개가 할당되어야 한다.

1.3. 연산자에 연산 할당

연산유형과 연산자의 수가 결정되면, 연산유형에 따라 결정된 기능연산자에 어떤 연산을 사용할 것인지를 결정해야하는데 연산자 할당은 다음과 같이 표현된다. 즉, 연산 유형이 t인 연산은 스케쥴링 결과로부터 산출된 m개의 연산자(rt)중 하나의 연산자에 할당되어야 한다.

$$\sum_{r_t=1}^{m} H_{ir_t}=1,\ (i=1,2,\ldots n_{ops}) \quad \cdots\cdots\cdots\cdots (2)$$

여기서 rt는 연산유형이 t인 연산자(Operator)이고, i는 연산유형이 t인 연산(operation)을 의미한다.

예를 들면 (그림 1)과 〈표 1〉에서 연산유형이 곱셈연산인 경우, 2개의 곱셈기(Multuplier; M1,M2)가 소요되며, 연산 O1, O2, O3, O6, O7, O8는 2개의 곱셈 연산자(M1, M2)중 하나의 연산자에 할당되어야 한다.

마찬가지로, ALU 유형의 2개의 연산자에는 연산 O4, O5, O9, O10, O11가 할당될 수 있으며, 덧셈 유형의 연산들은 2개의 ALU중 하나의 연산자에는 할당되어야 한다.

1.4. 연산자 할당 및 바인딩

위의 관계식(2-1),(2-2)를 요약하면, 다음과 같이 식을 정리할 수 있으며,

$$\sum_{r_t=1}^{m} H_{ir_t}=1,\ (i=1,2,\ldots n_{ops}) \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

$$\sum_{i\in T(O_i)=r_t} H_{ir_t} X_{il} \le 1,\ (i=1,2,\ldots n_{ops}\ ;\ r_t=1,2,..m)$$

$$H_{ir}\in\{0,1\},(i=1,2,\ldots N_{ops,};\ r=1,2,..m)$$

$$X_{il}\in\{0,1\},(i=1,2,\ldots N_{ops,};\ l=1,2,..L) \cdots (3)$$

여기서 rt는 연산유형이 t인 연산자, Hi는 연산유형이 t인 연산, l은 제어스텝(l=1,2,..L)를 나타낸다.

식(3)은 임의의 제어스텝에서, 연산자에 할당된 연산중 단지 하나의 연산만 수행됨을 의미한다.

1.5. ILP 표현법

위의 관계식을 적용하여 연산자 할당을 수행하였을 때의 정수계획법(Integer Linear Programming)는 다음과 같이 표현된다.

표 2. 연산자 할당 및 바인딩을 위한 ILP
Table 2. The ILP formulations for operation allocation and binding

```
SUBJECT TO
M11+M12=1;M21+M22=1;
M31+M32=1;M61+M62=1
M71+M72=1;M81+M82=1;
A41+A42=1;A51+A52=1
A91+A92=1;A101+A102=1;
A111+A112=1;
M11+M21<=1;M31+M61<=1;
M71+M81<=1;M12+M22<=1
M32+M62<=1;M72+M82<=1;
A41<=1;A51+A91<=1;
A101<=1;A11<=1
A42<=1;A52+A92<=1;
A102<=1;A112<=1
END
```

ILP에 의한 실험결과는 다음과 같다.

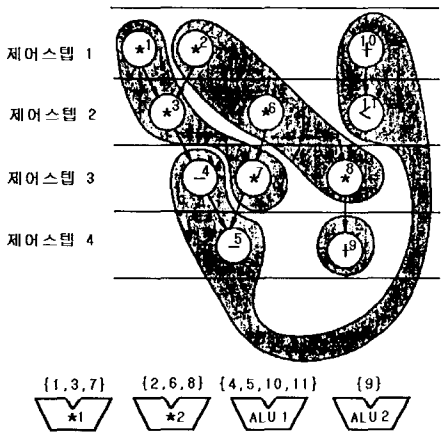

그림 3. 할당 및 바인딩 결과
Ffig. 3. The result of allocation and binding

표 3. 바인딩 결과
Table 3. The result of binding

연산자(자원)	바인딩 결과
M1	O1, O3, O7
M2	O2, O6, O8
A1	O4, O5, O10, O11
A2	O9

4개의 제어스텝에 스케쥴된 11개의 연산은 2개의 곱셈기와 2개의 ALU를 사용할 경우 각 연산이 연산자에 할당될 경우의 결과이다. 즉, 사용된 연산자가 곱셈기1(M1)과 곱셈기2(M2), ALU1(A1), ALU2(A2)라고 할 때, 각 연산이 결정된 연산자에 할당되는 것을 나타낸다.

2. 레지스터 할당 및 바인딩

레지스터 할당 과정은 레지스터의 수를 결정하고 연산의 출력인 변수를 레지스터에 할당하는 과정으로써 변수의 특성과 제약조건에 따라 레지스터 할당에 영향을 미친다. 변수의 일반적인 특징은 하나 이상의 출력이 되는 연산을 가지며 생존기간(Life Time)을 갖는다. Life time은 변수의 입력과 출력 연산사이의 제어스텝 간격이며 변수가 레지스터에 할당되는 제어스텝이다.

2.1. 레지스터의 수 결정

스케쥴링 결과로부터 각 연산의 입력과 출력 사이에 존재하는 변수를 기준으로 각 변수에 대한 Life Time을 조사하여 가장 많은 변수가 존재하는 제어스텝내의 변수를 기준으로 레지스터의 수를 결정한다.

아래의 그림은 각 연산과 관련된 변수와 Life Time을 나타낸 그림이다.

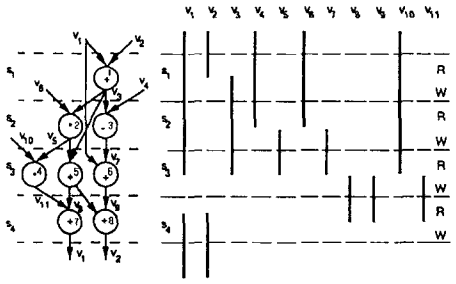

그림 4. 스케쥴된 연산에 대한 변수의 생존시간
Fig. 4. The variable lifetime for operation scheduled

여기서 제어스텝 1과 2 그리고 제어스텝 2와 3 사이에서 Life time에 대한 각 변수가 가장 많이 사용되므로 요구되는 레지스터 수는 5개 이다.

2.2. 변수의 생존시간

스케쥴링 결과로부터 각 연산의 입력과 출력 사이에 존재하는 변수의 생존시간을 산출해야한다. 모든 변수

는 임의의 제어스텝내에 존재한다.

모든 변수의 생존여부는 전체 제어스텝중임의의 제어스텝에 존재할 때, 레지스터에 할당될 수 있다. 변수의 생존시간에 대한 관계는 다음과 같은 식으로 표현된다.

$X_{is} = 1$, $(i = 1,2,..n_{var}$; $s = 1,2,..L)$…(4)

여기서 s는 제어스텝을 의미한다.

2.3 레지스터 할당 관계

스케줄된 연산과 관련된 변수들 그리고 결정된 레지스터의 수를 고려하여 레지스터에 할당될 수 있는 변수에는 다음과 같은 관계를 갖는다.

$$\sum_{m=1}^{r} V_{im} = 1, \quad \forall i \in \{1,2,3,..n_{var}\} \quad \cdots (5)$$

여기서, m은 레지스터를 의미한다.

즉, 사용할 레지스터의 수가 결정되면 DFG상의 모든 변수는 결정된 레지스터에 적어도 하나가 할당되어야 한다.

예를들면, 5개의 레지스터를 변수 Vi가 사용할 수 있는 레지스터의 범위 관계는 다음과 같이 표현된다.

Vi1+Vi2+Vi3+Vi4+Vi5=1, ∀i∈{1,2,3,..11}

V1에 대한 표현식은 v11+v12+v13+v14+v15=1이다. 즉, 변수 V1은 적어도 결정된 5개의 레지스터 중 하나의 레지스터에 할당되어야 한다.

그림 4의 각 변수(V1,V2,...V11)에 대하여 위 식을 적용하여 표현하면 다음과 같이 표현된다.

표 4. 변수할당을 위한 ILP표현법
Table 4. The ILP formulation for variables allocation.

V11+V12+V13+V14+V15=1
V21+V22+V23+V24+V25=1
V31+V32+V33+V34+V35=1
V41+V42+V43+V44+V45=1
V51+V52+V53+V54+V55=1
V61+V62+V63+V64+V65=1
V71+V72+V73+V74+V75=1
V81+V82+V83+V84+V85=1
V91+V92+V93+V94+V95=1
V101+V102+V103+V104+V105=1
V111+V112+V113+V114+V115=1

2.4. 레지스터 할당 및 바인딩

위의 관계식(4)와 (5)를 요약하면, 다음과 같이 수식을 정리할 수 있다.

$$\sum_{m=1} V_{im} X_{is} M \le 1, \forall i \in \{1,2,3,..n_{var}\} ; (s = 1,2,...L)$$

…… (6)

여기서, m은 레지스터수

위의 식(6)은 할당할 레지스터의 수가 결정되고, 변수를 레지스터에 할당하기 위하여 하나의 레지스터에 할당된 여러 개의 변수중 적어도 하나만 변수를 사용할 수 있다.

그림 4에서 11개의 변수(V1~V11)는 다섯 개의 레지스터(R1,R2,R3,R4,R5)에 할당되었다. 변수의 생존시간과 레지스터의 개수 등 제약조건을 만족하는 레지스터 할당 결과는 다음과 같다.

R1={V4, V5, V8}
R2={V1, V9},
R3={V6,V7}
R4={V10}
R5={V2, V3, V11}

Ⅲ. 실험결과

본 연구에서 제안한 방법의 효용성을 검증하기 위하여 HAL 시스템에서 사용한 Differential Equation DFG 모델을 실험 대상으로 할당 및 바인딩을 수행하였고, 할당 및 바인딩 문제를 해결하기 위한 모든 정수계획법(ILP formulation)은 SUN UltraSPARC 하에서 Lindo패키지 프로그램을 이용하여 해를 구하였다.

그림 5에서와 같이 DFG상의 모든 연산을 수행하기 위해서는 소요되는 자원은 2개의 곱셈 기능연산자와 2개의 ALU 기능연산자가 소요된다. 또한, 변수의 생성시간을 고려할 때 소요되는 레지스터 수는 5개가 소요된다.

이와 같이 자원에 대한 제한 조건을 만족하도록 기능연산자 할당 및 바인딩 된 결과는 그림 6과 같다.

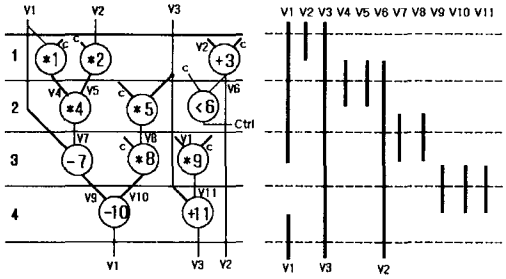

그림 5. HAL모델의 DFG 및 변수 생성기간
Fig. 5. The DFG and variables Lifetime of HAL model

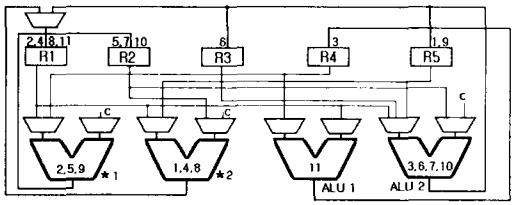

그림 6. HAL모델에 대한 실험 결과
Fig. 6. The experimental result of HAL model

표 5. Hal 모델에 대한 할당 및 바인딩 결과
Table 5. The result of allocation and binding for HAL model

■연산자 할당 및 바인딩 결과

<곱셈기(*)에 대한 결과>

M1∈{O_2, O_5, O_9}, M2∈{O_1, O_4, O_8}

<ALU(+,-,<)에 대한 결과>

A1∈{O_{11}}, A2∈{O_3, O_6, O_7, O_{10}}

■레지스터 할당 및 바인딩 결과

R1∈{V2,V4,V8,V11}, R2∈{V5,V7,V10},
R3∈{V6}, R4={V3}, R5∈{V1,V9}

Ⅳ. 결 론

본 논문에서는 상위수준합성에서의 중요한 할당문제를 다루었다. 특히, 스케쥴링 결과로부터 각 제어스텝에 지정된(assigned) 연산에 대한 연산자 할당 및 바인딩 그리고 레지스터 할당 및 바인딩 문제를 다루었다. 특히 할당 및 바인딩 문제를 해결하기위해 새로운 방법으로 ILP기법을 이용한 할당 및 바인딩 방법을 제시하였다. ILP기법을 이용한 할당 및 바인딩 방법은 표현 수식의 생성과 분석 그리고 재구성이 용이하였다. 표준 벤치마크 모델에 대하여 본 논문에서 제안한 할당 및 바인딩을 수행한 결과, 소요되는 기능연산자의 유형과 개수는 각각, 곱셈기 2개, ALU 2개, 레지스터 5개가 소요되며 DFG상의 연산 및 변수가 제한된 조건을 만족하면서 모두 연산자 및 레지스터에 할당 및 바인딩되어 DFG상의 모든 연산을 정상적으로 수행할 수 있어 다른 논문에서 제시한 결론과 일치함을 얻을 수 있었다. 본 논문에서는 단지 연산과 레지스터만을 고려한 할당 및 바인딩을 수행하였지만 상호연결선(MUX, BUS등)을 동시에 고려한 할당 및 바인딩 방법에 대한 연구가 지속되어야 한다.

참 고 문 헌

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

[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 Interna-

tional Conference on Computer Aided Design, pp. 20-23, 1989.

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

[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. pp. 263-270, 1986.

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

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

[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. pp. 464-475, 1991.

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

저 자 소 개

신 인 수

1991년 청주대학교 전자공학과 졸업(공학사)

1993년 청주대학교 대학원 전자공학과 졸업(공학석사)

1994년~현재 청주대학교 대학원 전자공학과 박사수료

1997년~현재 제천기능대학 정보통신 기술학과 전임강사

관심분야 : 디지털 시스템, High Level Synthesis, VLSI & CAD

이 근 만

1973년 한양대학교 전자공학과 졸업(공학사)

1980년 한양대학교 대학원 전자공학과 졸업(공학석사)

1992년 한양대학교 대학원 전자공학과 졸업(공학박사)

1982년~현재 청주대학교 첨단공학부 교수

관심분야 : 디지털 시스템, High Level Synthesis, VLSI & CAD