論文94-31A-2-16

게이트 분할을 고려한 Lookup Table 방식의 기술 매핑 알고리듬

(A Technology Mapping Algorithm for Lookup Table-based FPGAs Using the Gate Decomposition)

李 載 興*, 鄭 正 和*

(Jae Heung Lee and Jong Wha Chong)

要 約

본 논문에서는 lookup table 방식 필드 프로그래머블 게이트 어레이의 논리합성에서 칩면적과 경로 지연시간의 최소화를 위한 top-down 방식의 새로운 기술 매핑 알고리듬을 제안한다.

다단 논리합성시 문자 수의 최소화 대신에 기본 논리블럭 수의 최소화와 기본 논리블럭 레벨의 최소화를 목적함수로하는 회로 분할 방법을 채택함으로써 보다 정확한 칩 면적과 지연시간을 고려하였다. 특히, 게이트 분할을 고려한 큐브 packing 알고리듬을 제안하여 공통함수를 이용한 회로 분할 알고리듬에 적용함으로써, 기존의 게이트 분할과 큐브 packing 을 별도로 수행하는 알고리듬에 비하여 면적과 지연 시간에 있어서 우수한 결과를 얻을 수 있었다.

Abstract

This paper proposes a new top-down technology mapping algorithm for minimizing the chip area and the path delay time of lookup table-based field programmable gate array (FPGA).

First, we present the decomposition and factoring algorithm using common subexpression which minimizes the number of basic logic blocks and levels instead of the number of literals. Secondly, we propose a cube packing algorithm considering the decomposition of gates which exceed m-input lookup table. Previous approaches [5-8] perform the cube packing and the gate decomposition independently, and it causes to increase the number of basic logic blocks. Lastly, the efficiency

I. 서 론

최근 IC칩의 집적도가 종래의 설계 전문가의 수작업으로 설계 가능하던 한계치를 초월하여 단일 칩내에 수 백만개의 트랜지스터가 집적되는 VLSI 단계에 이르면서 설계자동화에 대한 요구가 크게 증가하고 있다.[1-3]

종전의 설계 전문가의 경험이나 능력에 전적으로 의존하는 수작업 설계 방법으로는 설계 면적에 대한 최적화 목표는 어느 정도 달성할수 있으나 설계 시간의 증가, 오류 발생 요인 증가 등의 단점을 가지고 있다.

* 正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1993年 8月 31日

따라서 디지탈 시스템 설계에 대한 정확성의 보장과 설계 시간을 단축 하기위해 CAD(Computer Aided Design) 기술을 VLSI 설계에 활용하는 설계 자동화 시스템 개발에대한 연구가 매우 활발히 진행되고 있다. CAD 시스템의 장기적인 목표는 동작기술로부터 설계자에 의한 사양을 만족하는 최적에 근사한 해를 구하는 실리콘으로의 자동합성에 있다. 이러한 CAD의 능력은 ASIC(Application Specific Integrated Circuit) 시장의 요구가 빠른속도로 증가함에 따라 중요성이 증대되고 있다.

특히, 최근에는 기존의 스탠다드 셀 또는 게이트 어레이 방식에 비하여 설계 시간과 설계비용을 대폭 감소시키는 FPGA(Field Programmable Gate Array)의 개발로 인하여 ASIC 시장에 커다란 영향을 미치기 시작하였고, FPGA 에 대한 집중적인 연구가 시작되었다. FPGA는 PAL(Programmable Array Logic)의 설계자에 의한 프로그램 가능성과 게이트 어레이의 다양성을 결합한 새로운 디바이스로써, 설계시간과 설계비용의 감소 뿐 아니라, testability의 극대화로 시스템 설계에서 빠르게 prototype 용도로 사용하는 요구가 증대되고 있다.

일반적으로, 프로그래머블 디바이스를 분류해 보면 크게 다음과 같이 분류한다.

1) Programmable Logic Device(PLD)
   전형적으로 PLA(Programmale Logic Array)의 구조를 갖고 있으며 A.M.D.와 Altera 에서 제공하는 아키텍쳐가 보편적으로 사용되고 있다.
2) 필드 프로그래머블 게이트 어레이(FPGA)
   전형적으로 게이트 어레이의 구조와 비슷하며, 다단 논리회로를 실현하는데 사용되고, 크게 Xilinx 와 Actel 에서 제공하는 아키텍쳐 로 구분한다.

동작기술언어로 부터 얻어진 논리회로 표현식을 PLD 상에 실현하는 툴로는 우수한 ESPRESSO-II 가 있으며 PLA 구조는 다양하지 않으므로 보편적으로 사용하는데 어려움이 없다. 그러나 FPGA 상에 구현할수 있는 툴은 극히 제한되어 있다. 왜냐하면, 각 FPGA 마다 기본 논리 블럭이 다를 뿐 아니라 배선 resource 가 다르기 때문에 일반적인 게이트 어레이의 경우 보다 본질적으로 훨씬 어려운 합성 문제가 되고 있다.

기본적인 FPGA 의 공통적인 구조는 동일한 기본 논리 블럭의 반복된 어레이로 구성되어 있으며, 기본 논리 블럭은 설계자에 의하여 다양한 논리 소자로 프로그램하여 사용할 수 있도록 되어 있으며, 기본 논리 블럭 사이에 있는 배선 resource 를 충돌이 일어나지 않도록 원하는 프로그램을 통하여 회로의 배선을 할 수 있도록 되어 있다. FPGA 는 크게 Xilinx 의 Lookup Table 방식과 Actel 의 Multiplexor 에 기초를 둔 아키텍쳐로 나누는데 Xilinx 는 Lookup Table 을 LCA(Logic Cell Array) 내의 static RAM에 저장하여 각 기본 논리 블럭의 논리를 결정하고, 배선 경로를 결정하는 방법을 채택하고 있으며, Actel 은 각 기본 논리 블럭과 배선 경로를 antifuse 방법을 통하여 결정하는 방법을 채택하고 있다. 특히, Lookup Table 방식의 FPGA 에서는 제한된 기본 논리 블럭 수와, 제한된 배선 resource 를 갖기 때문에 칩의 면적 이용률을 높히기 위하여는 최소의 기본 논리 블럭으로 주어진 함수의 실현이 필요하다. 각 기본 논리 블럭은 m 개의 입력 변수를 갖는 모든 함수를 실현할 수 있는데(m 2), XC2000 families 인 경우 m=4 이고, XC3000 families 인 경우 m=5 이다. m 입력 Lookup Table 은 2^m 비트의 디지탈 메모리가 필요하고 이들 Lookup Table 의 수는 2^{2^m} 개가 된다. 각 기본 논리블럭의 지연시간은 함수에 무관하게 일정한 값을 가지므로, 주어진 논리회로 식으로 부터 m 개의 입력을 갖는 최소의 기본 논리 블럭으로 기술 매핑하는 것이 칩 면적 이용률에 미치는 영향은 매우 크다.

최근, 발표된 모든 논문[5,10] 들은 기술 독립적인 단계인 다단 논리합성과 다단 논리 최적화 단계를 이미 거친 후, 모든 게이트의 입력이 m 이하로 구성된 이미 최적화된 회로로 가정하고 테크놀로지 매핑을 수행하고있다. Chotle[7,8] 의 경우 이미 최적화된 회로로 부터 DAGON[11] 에서와 마찬가지로 leaf 노드로 부터 동적 프로그래밍 기법을 이용한 매핑을 수행하고 있다. 그러나, 주어진 회로의 구조상에서 매핑 과정을 수행하기 때문에 결과적으로 구조상의 한계점을 극복하지 못하였다. 한편, mis-pga 의 경우 이러한 문제점을 부분적인 collapsing 과 decomposition 을 통하여 해결하고자 하였다.

따라서, 본 논문에서는 Lookup Table 방식 FPGA의 논리합성에서 칩 면적과 경로 지연시간의 최소화를 위한 top-down 방식의 새로운 FPGA 기술 매핑 알고리듬을 제안한다. 특히, 다단 논리합성 시 literal 수의 최소화 대신에 기본 논리블럭 수의 최소화와 기본 논리블럭의 레벨을 최소화시킴으로써 면적과 지연시간의 최소화를 얻도록 하였고, 또한 게이트 분할을 고려한 큐브 packing 알고리듬을 제안함으로써 기존의 게이트 분할과 큐브 packing 을 별도로 수행하는 알고리듬에 비하여 면적과 지연시간에

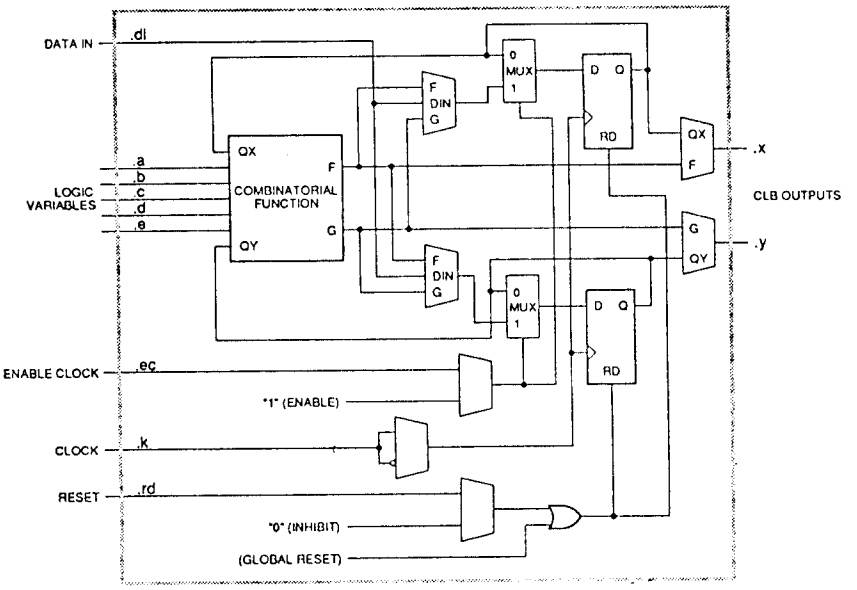

그림 1. CLB(Configurable Logic Block)의 구조
Fig. 1. The architecture of a CLB(Configurable Logic Block).

있어서 우수한 결과를 얻을 수 있었다.

Ⅱ. Lookup Table 방식의 FPGA 구조

Lookup Table 방식 FPGA의 전체 구조는 크게 입. 출력 블럭, 논리블럭인Configurable Logic Blocks (CLBs), 배선 영역으로 구성되어 있는데 이들은 수 십만개의 configuration 메모리에 의해서 제어된다. 입. 출력 블럭은 논리어레이와 핀 사이의 인터페이스를 제공해주고, Configurable Logic Block(CLB)는 사용자가 정하는 논리를 실현하며, 배선부분은 블럭들 사이의 신호선 연결 정보를 포함하고 있다. [13]

각 CLB는 그림 1. 에서 보는 바와 같이 조합논리 부분, 기억 소자 부분, 내부 배선과 제어 부분을 포함하고 있으며, XC3000 families 디바이스의 경우 조합논리 부분은 속도가 빠른 32 비트의 Table Lookup 메모리를 가지고 있어, 이들 메모리를 제어하여 설계자가 원하는 논리 함수를 실현할 수 있도록 하였다. 각 CLB 는 5 개의 일반적 목적의 입력 A, B, C, D, E를 포함하고 있고, 인접된 블럭과 함께 연결되는 클럭 입력(K)을 포함하고 있다.또한 각 CLB 는 2 개의 출력 X 와 Y 를 가지고있다.

XC3000 families 디바이스의 경우, 각 CLB 에서 조합 논리 부분이 실현가능한 함수는 다음 3가지의 경우로 나눌 수 있다.

선택 1) 5 개의 변수로 구성된 모든 논리함수
선택 2) 4 개의 변수로 구성된 모든 논리함수 2개
선택 3) 약간의 6 개 변수로 구성된 논리함수

5 개의 변수로 구성된 하나의 논리함수를 구현할때 (선택 1), 조합논리 부분의 출력 F 와 G 는 서로 동일한 논리를 가진다. 4 개의 변수로 구성된 2개의 논리함수를 구현할때(선택 2), 출력 F 와 G 는 서로 독립적인 논리로 사용되며, F 와 G의 변수의 합은 5 개를 초과할 수 없다. 또한, 입력 A, B, C, D, E 와 내부 기억소자 입력 Q 중에서 각각 4 개의 변수를 선택하여 전체 6 개의 변수로 구성된 한정된 논리함수를 구현할 수 있다(선택 3).

결국, 주어진 논리함수를 LCA(Logic Cell Array) 칩에 구현하기 위해서는 앞에서 이미 언급한 3 가지의 선택함수에 맞도록 조합 논리함수를 할당하는데 최소의 CLB 가 소요되도록 하는것이 필요하다. 이 할당 방법이 FPGA 에서의 테크놀로지 매핑이다.

Ⅲ. 준 비

본 장에서는 본 논문의 알고리듬을 기술하는데 필요한 기본적인 정의와 개념을 기술한다

1. 기본적인 정의

변수는 부울계의 하나의 좌표를 나타내는 하나의

기호(즉, a) 이며 문자는 하나의 변수 또는 그 변수의 보수이다.(즉, a 또는 $\bar{a}$) 하나의 큐브는 $x \in C$ 이 $x \notin C$ 을 의미하는 그러한 문자들의 집합 C를 나타내는데 예를 들어 { a, b, c }는 하나의 큐브이지만 { a, $\bar{a}$ }는 큐브가 아니다.

큐브들의 집합 f 를 표현식이라 하는데 예를 들어 { {a}, {b, c}} 은 2개의 큐브 {a} 와 {b, c}로 구성되어 있는 표현식이며 한 표현식은 그 큐브 들의 분리형으로 나타내짐을 알 수 있다. 부울 표현식 f는 필수적인 큐브들의 집합으로써 다른 큐브에 적당히 포함되는 큐브가 없는 것을 의미한다. 예를 들어 ab + c = { {a,b}, {c} }은 필수적인 큐브들의 집합이며 $\bar{a}b$ + b 는 {b}⊆{a, b} 이므로 불필요한 큐브를 포함하고 있으며 표현식 f 에서 큐브의 수는 |f|로 나타낸다. 어떤 표현식 f 의 support 는 다음과 같이 정의한다 supp(f) = { 변수 v | 어떤 큐브 C 에 대해서 v 또는 v∈C∈f } 예를 들어 supp($\bar{a}$b + c) = {a, b, c} 이다.

2. kernel 과 co-kernel [1]

효율적인 공통 부분 표현식을 찾기 위한 수단으로 논리 표현식의 kernel이 소개 되었는데 kernel은 대수 표현과 factored 형과의 중개 역할을 하며 표현식 f 의 kernel은 다음 2 가지 조건을 만족한다. [1]

1) 표현식 f 의 kernel k 는 f 와 한 cube c에 대한 몫으로 표현 하는데 즉, k = f/c 이다.

2) kernel k 는 " cube free " 이다.

여기서 cube free 는 k = dg (여기서 d는 불필요한 cube 이고 g 는 하나의 표현식) 으로 쓰여질 수 없는 것을 말한다. 다시말해서 표현식을 정확하게 나누는 큐브가 없다면 그 표현식은 cube free 하다고 한다. 표현식 f 의 primary divisor는 D(f) = {f/c| c 는 하나의 큐브} 의 집합이다.

표현식 f 의 kernel 은 K(f) = { g | g∈D(f) 이고 g 는 cube free }인 집합이다. 예를 들어 f = abc + abde 로 부터 f 에 대한 큐브 a 의 몫 즉, f/a = bc + bde 는 cube free 하다고 할 수 없다. 왜냐하면 f/a = b(c + de) 로 표현 가능하기 때문이다. 단일 큐브는 cube free 하지 않기 때문에 kernel 은 최소한 2 개 이상의 큐브로 구성해야 한다. 또한, uni-versal 큐브인 1 도 큐브이고 f/1 = f 이므로 f 가 cube free 하다면 f 는 그 자신의 kernel 중의 하나이다. 각 kernel과 관련된 큐브를 co-kernel 이라 부르고 간단히 kernel 를 얻기위해 사용된 큐브의 divisor 라고도 한다. 예를 들어 x = adf + aef + bdf + bef + cdf + cef + g = (a + b + c)(d + e)f + g에서 a + b + c 는 co-kernel df 와 ef 와 관련된 kernel 이고 비슷하게 d + e 는 co-kernel af, bf, cf 와 관련된 kernel 이다.

3. 부울회로

부울 회로는 다단 논리 함수를 표현하기 위한 실제 회로 이지만 실현기술과는 무관한 구조이다. 그 회로는 다음 조건을 만족하는 각 노드 i 에 대해서 비순환 방향성 그래프이다. 변수 y_i 와 논리 함수의 표현식 f_i 에 대해서 함수 f_i 가 함축적으로 변수 y_i에 의존한다면 즉, $y_i \in supp(f_j)$ 이면 노드 i 로 부터 노드 j 로 에지를 연결한다.

그림 2.는 hrd84 벤치마크에 대한 부울회로로써 다단 구조를 가지고있다. 그림 2.는 0, 1, 2, 3, ···, 7로 나타내는 8 개의 primary 입력과 4 개의 primary 출력 po0, po1, po2, po3 을 가지고 있는데, 이들은 각각 source 노드, 또는 sink 노드이며, 나머지 중간 노드는 다단 논리 회로에서 중간 변수에 해당된다.

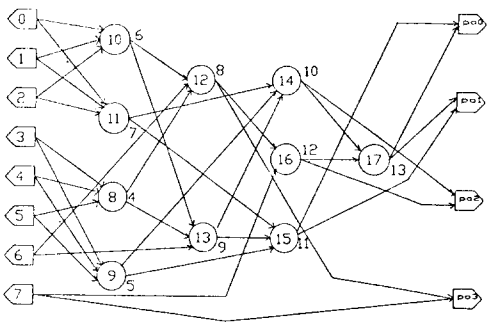

그림 2. hrd84에 대한 다단 부울 회로
Fig. 2. Boolean network of hrd84 circuit.

다단 논리 회로에서 한 함수와 부울 회로에서의 한 노드 사이에는 1 대 1 대응 관계가 있다. 따라서 부울 회로에서 각 함수와 노드는 혼용해서 사용되어도 무방하다.

Ⅳ. Lookup Table 방식 FPGA 에 대한 테크놀로지 매핑 알고리듬

함수 F = { f_0, f_1, ···, f_{n-1}} 의 부울회로에서 모든 노드 f_i 가 $supp(f_i) \le m$(XC2000 families 인 경우 m=4, XC3000 families 인 경우 m=5) 이면 함수 fi 는 feasible 함수라고 하며, 모든 노드가 feasible 함수이면 부울회로 F 는 feasible 함수라고 한다.

Lookup Table 방식의 FPGA 에 대한 테크놀로지 매핑의 목적 함수는 CLB 수의 최소화와 CLB 레벨 수의 최소화이므로 주어진 부울회로로 부터 최소의 노드 수를 가지며 모든 노드는 m 이하의 변수를 갖는 부울회로로 분할하는 문제로 귀착된다. 즉, 최소의 노드 수를 갖는 feasible 부울회로를 얻는 문제이다. 예로써 m=5 일때 함수 f = abc+degh+ij 를 f = x+y+z, x = degh, y = abc, z = ij 로 실현할 경우 4개의 CLB 가 소요되고, f = x+z+abc, x = degh, z = ij 로 실현 할 경우 3 개의 CLB 가 소요되고, f = u+degh, u = abc+ij 로 실현 할 경우 2개의 CLB가 소요되므로 decomposition 경우에 따라 상당한 면적의 차이를 보일 수있다. 따라서, 본 장에서는 주어진 부울회로로 부터 설계자가 원하는 feasible 부울회로로 분할하는 방법 즉, 테크놀로지 매핑 알고리듬을 기술한다.

1. 본 논문의 테크놀로지 매핑 알고리듬의 전반적인 구성

본 논문에서 제안하는 Lookup Table을 이용한 테크놀로지 매핑 알고리듬의 전체적인 구성도는 다음 그림 3.과 같다.

```
Techmap(F={f1, f2, ·····, fn}) {
    CSE = CSGEN(F); /* Common Subexpressions Generation */
    CST = Construct_cost_tab(F, CSE); /* using the bin-packing */
    best_cell = Best_selection(CSE,CST);
    while(SV_COST(best_cell) > 0) {
        Substitution(F, best_cell); /* Algebraic division */
        Update_tab(CSE, CST, best_cell); /* Update CST&CSE table */
        best_cell = Best_selection(CSE,CST);
    }
    for(each function fi ∈ F)
        Bin_packing(fi); /* Top-down method */
    Func_Merge(F);
}
```

그림 3. 본 논문의 테크놀로지 매핑 알고리듬의 전체 구성도

Fig. 3. Overall flow of technology mapping algorithm.

위 알고리듬은 크게 3단계로 구분할수 있는데 첫번째 단계로는 공통 함수를 이용한 방법[1,3]이다. 먼저 주어진 함수 표현식 F 에 대해서 앞에서 이미 정의한 kernel 들을 구하고 그 kernel 들의 공통 부분을 이용하여 가능한 후보 공통 부분 표현식을 모두 얻어내는 작업을 CSGEN(F)에서 수행한다. kernel 를 구하는 과정은 전체 면적을 최소화 시키는데 매우 중요한 과정이다. 이 과정에서는 부울회로에서 함수를 공통으로 사용하여 전체 면적을 감소시키기 위하여 2개 이상의 함수에 공통으로 존재하는 부분 표현식을 찾는다. 얻어진 모든 공통 부분표현식 CSE 에 대해서 큐브 packing 알고리듬을 적용하여 함수를 실현하는데 소요되는 전체 CLB 수와 전체 CLB 레벨 수에 따라 cost 테이블을 작성하고, 이들 공통 부분 표현식으로 부터 cost를 가장 감소 시키는 즉, CLB 수와 CLB 레벨 수를 가장 감소시키는 공통 부분 표현식을 Best_Selection() 함수에서 선택하여, Substitution() 함수에서 새로운 중간변수로 대치하게 되는데 이때 대수 나눗셈 알고리듬[1]을 도입하여 이용한다. 이와 같은 작업이 반복적으로 수행하여 주어진 함수의 분할 과정을 마치게 된다. 이 과정은 가능한 공통 부분 표현식을 추출함으로써 큐브 상호 연관관계의 정도를 줄이는 작업을 수행한다. 두번째 단계로는 support 수가 m 을 초과하는 각 함수들에 대해서 큐브 packing 알고리듬을 적용하여 각 함수가 feasible 함수가 되도록 Bin_packing() 에서 수행한다. 이때 각 함수에서 support 수가 m을 초과하는 큐브에 대해서는 게이트 분할을 동시에 고려한다. 세번째 단계로 $supp(f_i) < m$ 를 만족하는 2개의 함수를 하나의 CLB 에서 실현할 수 있다는 선택 2. 의 조건에 따라 m 보다 작은 입력 변수로 구성되는 함수 쌍을 찾아, 그 함수쌍의 전체 입력 변수가 m 보다 작을 경우, 그 함수쌍은 병합시켜 하나의 CLB 에 실현시키는 작업을 Func_Merge() 에서 수행한다.

1) 공통 부분 표현식 생성

본 논문에서 새로운 중간노드를 생성하기 위한 후보자 즉, 공통 부분표현식을 생성하기 위한 알고리듬은 그림 4.와 같다.

```
CSGEN(F) {    /* Common Subexpression Generation */
    for(each function f_i∈F )
        (K, Fac) = KF_gen(fi); /* Kernel and Factor generation */
    CK = Common_kern_gen(K); /* Kernel intersection */
    Compl_common_kern_gen(CK); /* Complement of common kernel 취가 */
    CC = Common_cube_gen(F); /* 공통 cube 생성 */
    return(CK ∪ CC);
}
```

그림 4. 공통부분 표현식 생성의 전체구성도

Fig. 4. An algorithm for the generation of common subexpressions.

먼저 주어진 부울회로의 각 노드에 대한 kernel 과 factor(co-kernel 이라고도 함)를 KF_gen() 에서 생성한다. Common_kern_gen() 은 서로 독립적인 kernel들 사이에 교집합을 구함으로써 공통 kernel, CK를 생성하도록 하였고, 또한 공통 kernel의 보수

도 Compl_common_kern_gen()에서 구하여 공통 kernel에 포함시켰다. 그 다음 Common_cube_gen ()에서 공통 큐브를 생성하였는데, 이것은 다출력 함수에서 주어진 함수의 큐브들에 대한 교집합으로 얻어진다. 공통 kernel, CK 와 공통 큐브 CC 합집합이 얻고자 하는 공통부분 표현식이 된다.

2) Cost 함수

부분 표현식 C_i 에 대한 Cost 함수는 그 부분 표현식을 새로운 중간 변수로 대치하였을때, 면적과 지연시간의 향상 정도로 면적은 소요되는 CLB의 수에 지연시간은 CLB의 레벨에 비례하며 다음과 같이 정의한다.

$$COST(c_i) = \alpha * SVAREA(c_i) + \beta * SVDELAY(c_i)$$

여기서, α 와 β 는 면적과 지연시간의 가중치로 설계자에 의해서 주어진다

소요되는 CLB 의 수와 레벨은 각 함수에 대해서 큐브 packing 을 수행하여 모든 함수를 feasible 함수로 만들 경우, 필요한 CLB 의 전체수와 가장 긴 경로의 CLB 의 레벨로 계산된다.

2. 게이트 분할을 고려한 큐브 packing 알고리듬

이 방법은 Bin Packing 방법[7,8] 이라고도 불리우며, 주어진 부울회로에 대해서 2 단 분할 방법으로 쉽게 적용할수 있는 방법이다. 특히 큐브들 간에 상호 연관관계의 정도가 적을 경우 좋은 결과를 나타낸다. 본 논문에서는 이 방법을 공통함수를 이용한 방법에서 가장 좋은 후보 공통함수를 선택을 위한 cost 의 계산에 이용하고 최종 feasible 함수를 만들때 이용하고 있다. 즉, 공통함수를 이용한 방법은 큐브 상호간에 연관관계의 정도를 줄이는 과정으로 최적의 공통 부분표현식을 선택하여 대치할때 감소하는 CLB 수와 레벨을 계산하는 용도로만 가상적으로 큐브 packing 알고리듬을 이용하고, 더 이상의 공통 부분 표현식이 존재하지 않을때 실제적으로 큐브 packing 알고리듬을 적용하고 있다. 그런데, 기존의 큐브 packing 알고리듬에서는 모든 게이트의 fanin 수가 m 보다 작도록 게이트 분할을 수행한 다음, support 의 수가 m 을 초과하지 않도록 최소의 bin 에 큐브를 packing 하는 방법을 채택하고 있다. 그러나 그림 5:에서 보는 바와 같이 게이트 분할의 결과에 따라 packing 된 결과는 다르다. 따라서, 본 논문에서는 게이트 분할을 동시에 고려한 큐브 packing 알고리듬과 collapse 를 고려한 OR 게이트 분할을 제안한다

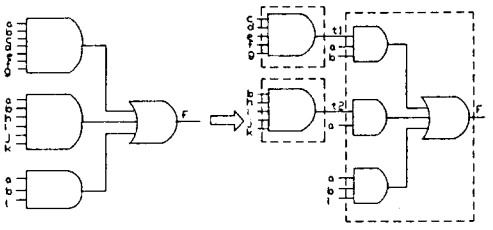

그림 5. 게이트 분할을 고려한 큐브 packing

Fig. 5. A cube packing considering the gate decomposition.

본 논문에서 게이트 g_i 를 같은 형태의 dg_i 로 분할할때, fanin 이 m 인 게이트로 반복적인 분할을 수행하는데 그림 6.에서 보는 바와 같이 분할되는 각 게이트의 입력값에 따라 수 많은 permutation 이 생기지만 분할되는 게이트(dg_i)들 간의 구조는 일정하게 그림 7.과 같은 알고리듬으로 수행한다.

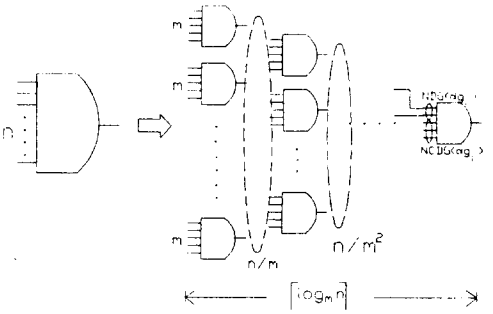

그림 6. 게이트 분할 예

Fig. 6. An example of gate decomposition.

```
Gate_decomp(gate_in, supp_num, dgate_num) {
    while(gate_in, LUT_num) {
        gate_num = gate_in/LUT_num;
        gate_in = gate_num * (LUT_num-1);
    }
    supp_num = gate_in;
    dgate_num = gate_num;
}
```

그림 7. 게이트 분할 알고리듬

Fig. 7. The algorithm for gate decomposition.

게이트 분할 알고리듬 Gate_decomp() 는 m을 초과하는 큐브에 대해서 fanin 수가 m인 게이트로 분할하여, 루트 게이트의 fanin 수(NFI(c_i))와 분할 게이트 입력 수(NDG(c_i)) 를 구하게 되는데, 이 단계

에서는 가상적인 게이트 분할을 할 뿐이고 NFI(c_j)와 NDG(c_j)를 이용하여 다음 절에서 설명할 [조건1] 과 큐브 packing 알고리듬을 거쳐 루트 게이트의 입력 변수가 결정된 후에 실제적인 게이트 분할이 그림 7의 알고리듬과 같이 수행된다. 그림 6.은 게이트 분할 과정을 나타낸 것으로써 최종 루트 게이트의 fanin 수와 분할 게이트 입력 수를 나타내고 있으며, 게이트의 레벨은 실제 CLB 의 레벨에 비례하여 경로 지연시간 계산에 이용된다.

[정의 1] 게이트 g_i 를 fanin 이 m 인 게이트들의 집합인 dg_i로 분할할때 dg_i의 루트 게이트의 fanin 수를 NFI(dg_i)라 하고, 이들 fanin 중에서 dg_i 의 어떤 게이트를 통하여 루트 게이트로 입력 되는 수를 NDG(dg_i), 그렇지 않은 경우를 NCDG (dg_i) 라고 정의하며 다음의 식을 만족한다.

NFI(dg_i) = NDG(dg_i) + NCDG(dg_i)

[정의 2] 그림 7.의 게이트 분할 알고리듬을 적용할 경우 경로 지연시간에 비례하는 게이트 레벨은 l = [$\log_m n$] 이다.

큐브 packing 문제는 주어진 각 함수 f = {c_1, c_2, c_3, ……, c_n} (여기서, c_i 는 큐브를 나타내고 함수 f 는 각 큐브들의 논리합) 에 대해서 B = {b_1, b_2, b_3, …., b_p} (여기서, bin b_i = {c'_j | 큐브 $c'_j \in$ f 이고, |supp(b_i)| ≤ m}) 인 permutation 에서 p가 최소가 되도록 B 를 구하는 문제이다.

함수 f가 게이트 분할 후에 각 루트 게이트들이 같은 bin b_i 에 packing 되기 위한 조건을 얻기위하여 다음과 같이 정의한다.

[정의 3] 함수 f = {c_1, c_2, c_3, ……, c_n}를 bin b_i 에 packing 할때 다음과 같이 정의한다.

CV1 = {c_j | 큐브 c_j 는 |supp(c_j)| ≤ m 이고, $c_j \in b_i$}

CV2 = {c_j | 큐브 c_j 는 |supp(c_j)|〉 m 이고, $c_j \in b_i$}

CV3 = {c_j | 큐브 c_j 는 |supp(c_j)|〉 m 이고, |supp (CV1)∩supp(c_j)| 〈NCDG(c_j)이고, $c_j \in b_i$}

[정의 3] 에서 packing 된 bin b_i 의 부분 집합인 큐브들 중에서 CV1은 m 을 초과하지 않는 큐브들의 집합이고, 그렇지 않는 경우의 집합을 CV2에 나타내고 있다. CV3 는 CV2 의 큐브 c_j 가 CV1 과의 공통 support 의 수가 NCDG(c_j) 보다 작은 큐브들의 집합을 나타내고 있다. 어떤 bin 에 포함되어 있는 큐브들이 게이트 분할 후에 하나의 bin에 packing 되기 위한 충분 조건은 다음과 같다.

[조건 1]

$$|supp(CV1)| + \sum_{c_i \in CV2} NDG(c_i) + \sum_{c_j \in CV3} (NCDG(c_j) - |supp(CV1) \cap supp(c_j)|) \leq m$$

여기서, NCDG(c_j) 〈|supp(CV1)∩supp(c_j)| 일때,

NCDG(c_j)-|supp(CV1)∩supp(c_j)| = 0

[조건 1] 이 의미하는 것은 bin 에 포함되어 있는 모든 큐브들이 게이트 분할 후에 각 루트 게이트에서 support 의 합이 m을 초과하지 않도록 한 조건이다. 즉, packing 된 어떤 bin 의 각 큐브들의 루트 게이트가 하나의 CLB 에 실현되기 위한 조건으로 그림 8. 에서 나타내고 있다. 그림 8. 에서 나타낸 바와 같이 큐브 c_1, c_2, c_4 는 같은 bin 에 할당이 가능하지만, 큐브 c_1, c_2, c_3는 불가능 하다. 한편, 필요 충분 조건이 되기 위한 조건은 CV3 에 포함되어 있는 큐브 들이 서로 공통이고, CV1 에는 포함되지 않는 support 수 만큼 보상해주면 된다.

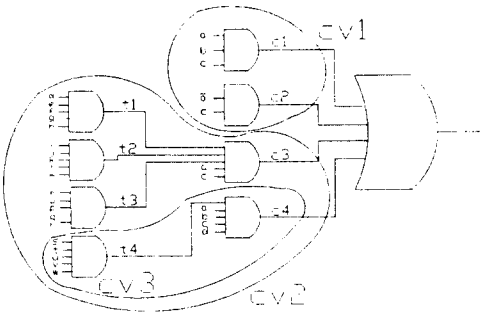

그림 8. 큐브 packing 을 위한 조건
Fig. 8. The condition for a cube packing.

위의 조건을 만족 할 경우 각 큐브는 같은 bin 에 할당하여 각 큐브의 루트 게이트의 fanin 값을 고려한 게이트 분할을 수행하여 하나의 CLB에 할당한다. 그림 9.는 본 논문에서 제안하는 bin packing 알고리듬을 나타내고 있다.

```
Bin_packing(f1) {
    if(NS(f1) <= LUT_num)
        return;
    else {
        for(each cube ci ∈ f1) {
            Gate_decomp(NS(ci), &sn, &dn);
            Supp_list <-- sn;
            Dgate_list <-- dn;
        }
        Cube_pack(f1, Supp_list, Dgate_list);
        OR_gate_decomp(f1);
    }
    return;
}
```

그림 9. 본 논문의 bin packing 알고리듬
Fig. 9. The overall algorithm for bin packing.

함수 f_i 의 각 큐브 c_j 에 대해서 Gate_decomp() 를 통하여 가상적인 게이트 분할을 수행하여 루트 게이트의 fanin 수(NFI(c_j))와 분할 게이트 입력 수 (NDG(c_j)) 를 구하게 되는데 그림 9.에서는 각각 sn 과 dn 으로 나타내고 있으며, NS(c_i)는 큐브 c_i 의 support 수를 나타내고 있다. 실제적인 게이트 분할은 NFI(c_j)와 NDG(c_j)를 이용하여 [조건1] 과 큐브 packing 알고리듬의 결과에 따라 각 큐브들의 루트 게이트의 입력 변수가 정해진 후에 수행된다. 게이트 분할을 fanin 수가 m 이 되도록 수행하는 Gate_decomp 알고리듬을 그림 7. 에서 나타내고 있으며 이 과정에서 fanin 수가 m 인 분할 게이트는 필수적으로 하나의 CLB 에서 실현되므로 필요한 CLB 의 갯수와 경로 지연시간에 비례하는 CLB 의 레벨을 동시에 얻는데 이용된다. 특히, 게이트 분할 과정에서는 게이트의 fanin 수가 m 을 초과하지 않도록 게이트 분할 만을 수행하며, 각 게이트에 가해질 입력 값은 아직 결정하지 않고, 큐브 packing 과정이 끝난 다음에 조건에 맞도록 루트 게이트의 입력 값이 정해지게 된다. 각 입력 값의 결정에 따라 bin 의 수가 증가할 수있기 때문이다. 그림 10은 분할된 루트 게이트들에 대하여 큐브 packing 과정을 수행하는 알고리듬을 나타내는데 조건 1 에 따라 수행한다.

```
Cube_pack( fi ) {
    cb_ordered_list = Cube_pack_order(fi, cb_list );
    while( cb_list is not empty ) {
        best_id = Best_cb_sel( fi, cb_ordered_list, bin_cb_list );
        if(best_cube is not exist ) {
            bin_cb_list <--mew bin_cb_list;
            cb_ordered_list = Cube_pack_order( fi, cb_list );
        }
        else {
            insert best_id into bin_cb_list;
            remove best_id from cb_list;
        }
    }
    Update fi according to bin_cb_lists;
    return;
}
```

그림 10. 큐브 packing 알고리듬

Fig. 10. The cube packing algorithm.

각 fi 의 packing 하고자 하는 큐브 리스트들에 대해서 큐브 packing 과정을 수행하는데, 함수 Cube_pack_order()는 packing 하고자 하는 큐브들에 대한 다른 큐브와의 공통 support 의 정도에 따라 각 큐브에 가중치가 주어지고 그 가중치에 따라 내림 차순으로 나열한 다음 그 순서에 따라 각 큐브의 packing 과정을 수행하고자 한다. 큐브 packing 과정은 support의 수가 m 보다 작도록 가장 적합한 큐브를 선택하여 bin에 할당하는 작업을 반복적으로 수행하는데, 적합한 큐브가 존재하지 않을 때에는 새로운 bin 에 packing 하게 된다. 다음 그림 11. 큐브들에 대한 가중치를 계산하여 내림차순으로 나열하는 함수 Cube_pack_order() 알고리듬을 나타내고 있다.

```
Cube_pack_order(fi, cb_list) {
    for(each input ii ∈ fi) {
        w_num = 0;
        for(each cube cj ∈ cb_list)
            if(ii-th variable is exist in cj)
                w_num++;
        weight_list[ii] = w_num;
    }
    for(each cube ci ∈ cb_list) {
        cb_weight_list[ci] = sum of each weight corresponding to
                             NCD(ci) variables in ci;
    }
    cb_ordered_list <-- ordered cb_weight_list by decreasing;
    return(cb_ordered_list);
}
```

그림 11. 큐브의 공통 support 에 따른 큐브의 가중치 알고리듬

Fig. 11. The cube ordering corresponding to the weights of cubes.

위의 알고리듬은 각 큐브에 대해서 입력값이 나타난 정도에 따라 각 입력 변수의 가중치가 정해지고, 각 큐브 c_i 의 가중치는 루트 게이트의 NCD(c_i) 수 만큼 ci 에 나타난 입력 변수의 가중치 합으로 나타냈다. 그 이유는 루트 게이트의 NCD(c_i) 수 만큼 만이 최종 큐브 packing 에 영향을 주기 때문이다. 다음 그림 12.는 큐브 packing 을 위한 가장 적합한 큐브를 선택하는 알고리듬을 나타내고 있다.

```
Best_cb_sel(fi, cb_ordered_list, bin_cb_list) {
    if(bin_cb_list is empty)
        best_id = cb_ordered_list[0];
    else {
        for(each cube ci ∈ cb_ordered_list)
            if(ci is a candidate cube)
                cb_candidate_list <-- ci;
        for(each cube cj ∈ cb_candidate_list)
            if(next candidate cube is exist corresponding to cj) {
                best_id = cj;
                return(best_id);
            }
        best_id = cb_candidate_list[0];
    }
    return(best_id);
```

그림 12. 가장 좋은 큐브의 선택 알고리듬

Fig. 12. The selection algorithm of the best cube.

위 알고리듬은 각 큐브의 가중치 순서에 따라 가장 적합한 큐브를 선택하는 과정을 나타내고 있다. 앞에서 이미 기술한 [조건 1] 을 만족하는 모든 큐브를 후보 큐브리스트(cb_candidate_list)에 저장하고 이들 각 후보 큐브들 중에서 bin 에 할당하더라도 다음에 할당할 큐브가 존재하는지를 조사하여 다음에 할당할 큐브가 존재하는 큐브를 가장 좋은 큐브로 선택하는, 즉 look ahead 방법으로 조사하는 방법을 채택하고 있다.

한편, 각 큐브를 최소의 bin 에 할당하는 큐브 packing 과정이 끝나면 이들 bin 들의 논리합으로 함수 f가 표현된다. 따라서, bin 의 수가 m 을 초과할 경우 각 큐브의 AND 게이트 분할과 같은 방법으로 그 bin 들의 OR 게이트 분할을 수행한다. 이때, packing 된 bin을 collapse 시켜서 루트 게이트가 m을 초과하지 않는 bin 이 존재할때, 그 bin 을 collapse 시킨채로 OR 게이트 분할을 수행하여 bin packing 과정을 마치게 된다.

3. 함수의 병합

지금까지 공통함수를 이용한 decomposition 방법과 bin packing 을 이용한 decomposition 방법에 대해서 기술하였다. 그러나 이미 FPGA 구조에서 언급한 바와 같이 XC3000 시리즈의 경우 하나의 CLB 에서 실현 할수있는 함수는 다음과 같다.

선택 1. feasible 함수 (m = 5)

선택 2. 2 개의 feasible 함수 f 와 g

여기서, $|supp(f)| < m$, $|supp(g)| < m$ 이고,

$|supp(f) \cup supp(g)| \leq m$

앞에서 기술한 decomposition 방법은 선택 1. 만을 고려하여 수행하였기 때문에 선택 2.를 고려하기 위하여 다음을 정의한다. 한편, 선택 3.는 어떤 특정한 함수만이 실현가능 하기 때문에 고려하지 않았다.

[정의3] 선택 2.를 만족하는 함수 f와 g가 존재할때 함수 f 와 g 는 mergeable 함수하고 정의한다.

Decomposition 과정을 거쳐 주어진 부울회로 F가 feasible 함수 일때 , 그 feasible한 부울회로로 부터 mergeable 함수 쌍을 찾아 2-ouput CLB 로 할당하여 전체 CLB의 숫자를 줄인다.

예를 들어, 주입력 9 개, 주출력 4 개를 가지고 있는 함수 $f_1 = ab + cd$, $f_2 = f_1h + t$, $f_3 = etf_4$, $f_4 = fghi$, $t = abc$ 에 대해서, $f_1 = ab + cd$, $f_2 = (ab + cd)h + abc$, $f_3 = abcef_4$, $f_4 = fghi$ 로 수행할 경우, mergeable 함수 쌍은 없고, 4 개의 feasible 노드가 있으므로 4 개의 CLB가 필요하다. 그러나 위의 함수를 그대로 적용할 경우 $\{f_1, t\}$ 와 $\{f_2, f_3\}$가 mergeable 함수쌍으로 얻어지므로 3 개의 CLB 로 실현할 수 있게 된다

V. 실험 결과

본 논문에서 제안한 알고리듬은 SUN SPARC-IPC(15 mips) 에서 C 언어로 약 14,000 라인 정도로 코딩하여 벤치마크 회로에 대해서 프로그램 실험을 한 결과 다음 Table 1.와 같은 결과를 얻었다. Mis-pga 의 표준 script 에 대한 결과와 본 논문의 결과의 비교는 Table 1.에서 보는 바와 같이 전체적으로 CLB 수가 약 20 % 정도 감소하였고, CLB 의 레벨 수는 거의 비슷한 결과를 얻었으며, 실행시간 또한 우수한 결과를 보였다. 특히, mis-pga 는 내부 simplify 라는 커맨드를 script 에 포함시킨 결과이고 script의 기술에 따라 결과도 약간의 변동이 있을 수 있다. 본 논문의 결과는 다단 논리최소화(mis-pga에서 symplify)단계 뿐만 아니라 회로 다단화 과정의 기술 독립적 최적화 단계를 거치지 않은 결과인데도 CLB 수에서 우수한 결과를 보였으며, CLB 수와 지연시간에 비례하는 CLB 레벨 수를 동시에 고려한 결과를 나타내고 있다. 특히 큐브 packing 수행시에는 다단 packing 과정을 포함시키지 않아서 레벨 수의 증가를 초래하지 않도록 하였다. Boolean cover 형태를 입력으로 받고, 출력으로는 다른 툴과의 인터페이스를 위하여 EDIF(Electronic Design Interchange Format)와 Xilinx와의 인터페이스를 위하여 LCA(Logic Cell Array) 화일로 출력이 가능하도록 하였다.

표 1. 벤치마크 회로에 대한 실험결과

Table 1. The experimental results of benchmarks.

	mis-pga			본 논문		
예 제 회 로	CLB 수	level 수	cpu(sec)	CLB 수	level 수	cpu(sec)
ADR4	71	5	48.6	38	4	7.2
CANDY	8	3	1.6	7	3	0.6
F51M	60	5	49.7	33	4	3.8
HRDB4	7	5	7.1	6	5	0.9
RD53	3	2	29.1	3	2	0.8
RD73	155	5	2092.1	109	4	973.5
5XP1	42	4	48.6	27	4	2.8
ROOT	75	4	52.3	59	4	52.3
DUKE2	165	6	1306.2	183	6	1408
MLP4	163	5	749.0	148	5	1364
SQN	43	4	23.9	32	4	4.7
SQR6	21	3		21	2	1.9
RISC	27	3	19.9	28	3	1.1

Ⅵ. 결론

본 논문에서는 최소의 CLB 수와 최소의 CLB 레벨로 주어진 회로를 실현함으로써 칩 면적의 최소화를 얻을 수 있는 새로운 Lookup Table 방식의 FPGA 테크놀로지 매핑 알고리을 제안하였다.

공통 함수를 이용한 회로 decomposition 방법을 제안하였고, 게이트 분할을 고려한 Top-down 방식의 큐브 packing 제안함으로써 기존의 알고리듬에 비하여 좋은 결과를 얻을 수 있었다. 단 한번의 시도에서 얻어진 결과가 좋은 이유는 기존의 chotle 의 큐브 packing 알고리듬에서는 게이트 분할 과정과 큐브 packing 과정이 분리되어 적용 되었기 떠문이다.

얻어진 결과에 대하여 각 경로의 부분 collapse 와 재 매핑과정을 수행 할 경우 보다 좋은 결과를 얻을 것으로 기대되며, 또한 Sequential 논리를 포함한 매핑 방법에 대한 연구가 요구된다.

參 考 文 獻

[1] R.Rudell, R.K.Brayton, A.Sangiovanni-Vincentelli and A.Wang, "MIS: A Multi-level Logic Optimization System," *IEEE Transactions on Computer Aided Design CAD-6*, Nov. 1987.

[2] K.A.Bartlett, D.G.Bostick, G.D. Hachtel, R.M.Jacoby, M.R.Lightner, P.H.Moceyunas, C.R.Morrison, and D. Ravenscroft," BOLD : A multiple level logic optimization system," ICCAD 1987.

[3] R.K.Brayton and C.T.McMullen, "The decomposition and factorization of boolean expressions ," *Proc. Int. Symp.* on Circuits and systems, pp. 49-54, 1982.

[4] R.K.Brayton, G.D.Hachtel, C. McMullen, and A.Sangiovanni- Vincentelli, "Logic minimization algorithms for VLSI synthesis.", Boston : Kluwer Academic, 1984.

[5] Rajeev Murgai, Yoshihito Nischizaki, Narendra Shenoy, Robert k. Brayton and Alberto Sangiovanni-Vincentelli, "Logic synthesis for programmable gate arrays," *Proc.* 27th Design Automation Conference, June 1990, pp 620-625.

[6] Rajeev Murgai, Yoshihito Nischizaki, Narendra Shenoy, Robert k. Brayton and Alberto Sangiovanni-Vincentelli, "Improved logic Synthesis algorithm for table lookup architectures," *Proc. IEEE Int. Conf.* on Cad (ICCAD), November 1991, pp 564-567.

[7] Robert Francis, Johnathan Rose and Zvonko Vranesic, "Chortle-crf: fast technology mapping for lookup table-based FPGAs," *Proc.* 28th Design Automation Conference, June 1991, pp 227-233.

[8] Robert Francis, Johnathan Rose and Kevin Chung "Chotle: A Technology Mapping Program for Lookup Table-Based Field Programmable Gate Array," *Proc.* 27th Design Automation Conference, June 1990, pp 613-619.

[9] Nam-Sung Woo, "A heuristic method for FPGA technolgy mapping based on the edge visibility," *Proc.* 28th Design Automation Conference, June 1991, pp 248-251.

[10] Silvia Ercolani and Giovanni De Micheli, "Technology mapping for electrically programmable gate arrays," *Proc.* 28th Design Automation Conference, June 1991, pp 248-251.

[11] Keutzer K., "DAGON: Technology Binding and Local Optimization by DAG Matching," *Proc.* of 24th Design Automation Conference, June 1987, 341-347.

[12] Detjens E.,et.al, "Technology Mapping in MIS," *Proc.* of ICCAD-87, Nov. 1987, 116-119.

[13] The Programmable Gate Array Data Book Xilinx, San Jose, CA, 1991

著 者 紹 介

李 載 興(正會員) 第 28券 A編 10號 參照
현재 대전산업대학교 전자계산학과
조교수

鄭 正 和(正會員) 第 28券 A編 10號 參照
현재 한양대학교 전자공학과 교수