論文 92-29A-2-12

# 논리 회로의 기술 매핑 시스템 설계

## (Design of a Technology Mapping System for Logic Circuits)

金 泰 善,** 黃 善 泳*

(Tae Sun Kim and Sun Young Hwang)

**要 約**

본 논문에서는 기술 독립적인 Boolean equation에 라이브러리 게이트를 매핑하는 효율적인 기술 매핑 방법을 제안한다. 제안된 시스템은 그래프 커버링에 의해 매핑을 수행하며 최적의 면적 커버를 얻기 위하여 새로운 비용 함수와 부분 면적 최적화 방법을 사용한다. 실험 결과는 제안된 시스템의 성능이 우수함을 보여준다.

**Abstract**

This paper presents an efficient method of mapping Boolean equations to a set of library gates. The proposed system performs technology mapping by graph covering. To select optimal area cover, a new cost function and local area optimization are proposed. Experimental results show that the proposed algorithm produces effective mapping using given library.

## I. 서 론

작은 면적의 칩 안에 VLSI 시스템을 구현하기 위해서는 주어진 제한 조건을 만족시키는 상위 수준 합성을 거친 후, 회로의 기능을 유지하면서 회로의 면적을 줄이고 실제 테크놀러지로 구현 가능한 게이트로 회로를 구성하는 논리 설계 과정이 필요하다. 이를 위해 회로의 redundancy를 제거하고 공통된 기능을 가진 부분을 찾아내어 하나의 부분으로 대치시키는 기법이 연구되어 왔다.[2-3] 특히 기술 매핑(technology mapping)의 문제는 이의 넓은 응용과 다른 문제로의 응용이 가능하며 VLSI 기술의 발달과 함께 더욱 큰 관심의 대상이 되었다. 테크놀러지의 발달과 함께 새롭게 부각되고 있는, 기존의 테크놀러지 하에서 최적화된 회로를 다른 새로운 테크놀러지로 재구현하는 기술 전환 (technology translation)의 문제도 기술 매핑의 한 응용 분야이다. 기술 전환은 두가지 방법으로 가능하다. 그 한 방법은 기존의 테크놀러지에 의해 최적화된 회로를 구성하는 각각의 게이트를 주어진 테크놀러지의 라이브러리에 매핑시키는 방법으로 이는 구현이 간단한 대신 기존 테크놀러지의 틀을 벗어나지 못하므로 진정한 의미에서 기술 의존 최적화되었다고 보기 힘들다. 다른 방법은 주어진 회로가 테크놀러지와 관계없다고 가정하여 주어진 테크놀러지에서 제공되는 게이트들로 최적화시키는 방법으로, 이 방법은 최적화된 결과를 얻을 수 있는 반면 시간이 오래 걸린다는 단점을 가진다.

기술 매핑은 기술 독립적 논리 최소화 과정을 거친 후의 조합 논리 회로의 테크놀러지에 따른 라이브러리에서 제공하는 게이트들로 재구성하는 과정으로 주어진 테크놀러지 하에서 최적화된 회로를 만드

*正會員, **準會員, 西江大學校 電子工學科
(Dept. of Elec. Eng., Sogang Univ.)

接受日字 : 1991年 7月 4日
(※ 이 논문은 1990년도, 1991년도 교육부 학술 연구 조성비에 의하여 연구되었음.)

는 작업은 그 복잡함과 어려움 때문에 설계 자동화 하려는 노력이 계속되어져 왔다.[5-7,10,13,14] SOCRATES,[5][9] LSS[6]와 같은 초기 기술 의존 최적화 시스템들은 규칙 기반(rule-based) 시스템들로서 회로의 일부분의 성능을 개선하기 위하여 각 룰들의 일련의 반복된 적용으로 회로의 성능을 향상시킨다. 하나의 룰이 인식할 수 있는 성능 향상 정도는 단지 룰에 정의된 transformation에 의한 국소 면적에 국한된 성능 향상 정도 뿐이며 각 룰들의 조합에 의한 전체 회로 성능 향상은 meta-rule의 조정에 따라 큰 영향을 받는다.[5] Meta-rule의 사용으로 직접 하나하나의 룰을 적용할 때 발생할 수 있는 국소 최적화는 벗어날 수 있으나 전체 회로를 고려한 최적화를 얻을 수 없으며, 또한 룰 자체가 라이브러리에서 제공되는 게이트 종류에 의해 영향을 받게 되므로 테크놀러지 변화에 따라 룰을 재조정해야 하는 문제가 있다.[4]

이에 대한 대안으로 그래프 커버링에 의한 기술 매핑 방식은 라이브러리를 주어진 테크놀러지에 독립적인 형태로 구성하고 DAG(directed acyclic graph)로 나타난 대상 논리 회로를 라이브러리의 게이트 종류에 무관하게 기술 매핑할 수 있는 여러 부분 회로로 분할 (partitioning)하여 각각의 부분 회로를 최적화시킨다. 회로의 면적은 분할하는 방법에 따라 더욱 최적화될 수 있다. 초기의 DAGON[10][11] 시스템은 DAG로 표현되는 대상 논리 회로를 회로의 fanout 노드들과 primary output 출력 노드들을 root 노드로 하는 부분 회로들로 분할하여 각각을 최적화시키는 방법을 이용하였고, mis-Ⅱ technology mapper[7][15]시스템은 primary output을 root로 하는 모든 transitive fanin을 하나의 cone으로 하여 각각의 cone을 최적화시키는 방법을 이용하였다. 이들 시스템들은 DAG covering 문제를 tree covering 문제로 바꾸어 해결하였고 최근에는 wire 면적도 고려하여 면적을 최적화시켜 주는 시스템도 발표되었다.[14] 그러나 DAGON 시스템은 fanout 노드를 boundary로 설정하여 줌으로써 면적의 최적화에 한계를 보였고, fanout 노드에 boundary를 두지 않고 logic cone을 최적화시킨 mis-Ⅱtechnology mapper는 fanout 노드에서의 면적 손실을 감소시켰으나 어느 logic cone 부터 매핑시키는가에 따라 면적의 손실이 발생할 수 있게 되었다. Lily 시스템은 linear ordering에 의해 매핑될 cone의 순서를 설정해 주는 방법을 제안하였으나 그 결과는 비용 함수에 의해 크게 좌우된다.[14]

본 논문에서는 그래프 커버링에 의해 논리 최적화된 회로를 생성하는 기술 매핑 시스템인 SiLOS-Ⅱ에 대하여 기술하고 그 실효성을 보인다. 본 시스템은 fanout 노드를 boundary로 설정하지 않음으로써 fanout 노드에서 발생할 수 있는 면적 손실을 줄이는 한편 cell 수 최적화와 부분 면적 최적화를 수행함으로써 cone ordering에 무관하게 회로 면적 최적화를 이루었다. 또한 부하량 (loading value)을 고려하여 시간 제약하에서의 회로 면적 최적화를 구현하였다.

## Ⅱ. 시스템 개관

기술 매핑을 수행하는 전체적인 시스템의 구성을 그림1에 보였다. SiLOS-Ⅱ는 본 연구실에서 개발한 다단 논리 최적화 시스템인 SMiLE[19]을 거쳐 기술 독립 최적화된 equation을 입력으로 취한다. Equation을 파싱하여 얻어진 netlist는 2-input NAND게이트와 인버터로 구성된 DAG로 만들어지며 preprocessing 과정에서 reconvergence를 검사한다. DAG 형태의 회로는 한 primary output을 root 노드로 하는 모든 transitive fanin으로 구성된 logic cone 들의 집합으로 인식되며 각 logic cone들은 라이브러리의 게이트를 나타내는 여러 이진 트리의 문자열과 비교하여 라이브러리 게이트로 매칭 가능한 모든 패턴을 찾아 낸다. 동작 모드에 따라 면적이나 cell 수 만을 최적화시키거나 여러 가능한 매칭된 패턴들의 조합으로 부터 사용자에 의해 주어진 동작 제약 시간을 넘지 않는 범위 안에서 최적의 커버를 구한다. 마지막으로 대상 형태가 있을 경우에 한하여 부분 면적 최적화를 수행하여 결과를 VHDL structural 기술[12]로 출력시킨다.

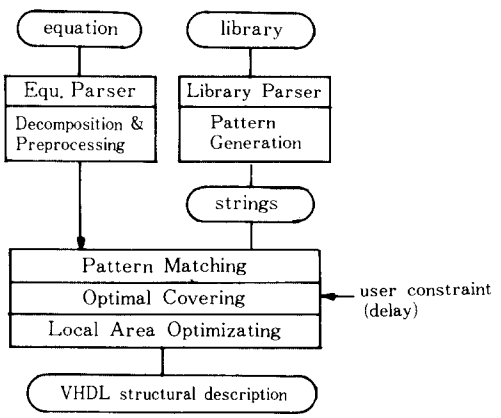



그림 1. 시스템의 구성도
Fig. 1. System flow diagram.

## Ⅲ. 그래프 커버링에 의한 기술 매핑

### 1. Preprocessing

회로 내의 한 fanout 노드에서 분리된 두 path가 회로 내의 다른 노드에서 합쳐질 때 reconvergence로 정의된다. Fanout 노드를 커버링의 boundary로 설정하지 않게 됨에 따라 reconvergence는 그래프 커버링에 의한 면적 최적화 과정에서 패턴 매칭과 면적 계산시 예외적인 처리를 필요로 한다. Reconvergence가 발생했을 경우 그림 2(a)와 같이 fork 노드에서의 면적은 양쪽 path를 따라 join 노드로 전달된다. 만약 이를 전처리 단계에서 처리해 주지 않을 경우 join 노드에서 가지는 최소 면적을 계산할 때 fork 노드의 면적이 중복되어 계산되므로 실제의 최소 면적을 계산할 수 없다. 또한 회로 내에 redundant한 부분이 있을 경우 패턴 매칭 과정에서 그림 2(b)와 같이 잘못된 패턴 매칭이 일어날 수 있다. 이러한 문제점들을 방지하기 위하여 fork, join 쌍들의 테이블을 유지하고 각각의 노드에는 표시를 한다.

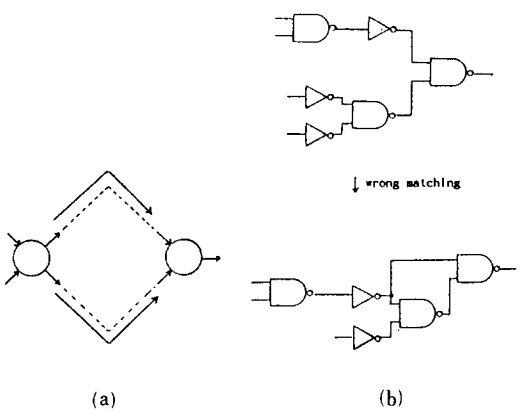



**그림 2.** Reconvergence로 인해 발생되는 문제점들
(a) 면적이 중복되어 계산
(b) 잘못된 패턴 매칭

**Fig. 2.** The problems incurred by reconvergence.
(a) duplicated area estimation,
(b) wrong pattern matching.

### 2. Decomposition

#### 1) Base function

대상 회로는 2-input NAND 게이트와 인버터들로 재구성되며 이 과정에서 생겨난 redundant한 인버터들은 제거되어 진다. 회로를 구성하는 제한된 형태의 게이트들은 base function으로 정의되며,[4] base function은 패턴 매칭 과정에서 더 이상 쪼개어지지 않는 기본 구성 요소이다. 회로를 구성하는 base function으로서 인버터와 2-input NAND 게이트만을 제공하는 이유는 먼저 게이트 타입을 통일시킴으로써 여러가지 종류의 게이트를 제공할 때 발생할 수 있는 패턴 매칭의 어려움을 피할 수 있고, 주어진 회로를 이진 트리로 구성하여 사용되는 테크놀러지에 독립적인 형태로 바꾸어 줌으로써 제공되는 라이브러리에 관계없이 언제나 최적에 가까운 결과를 얻을 수 있다는 장점을 갖기 때문이다. DAGON 시스템의 경우, 회로를 구성하는 base function으로서 NAND만이 아닌 AND나 OR 타입도 가능하며 입력 갯수도 2개로 제한하지 않는다. Base function 으로 3개나 4개의 입력을 가진 게이트를 허용함에 따라 라이브러리의 게이트를 표현할 수 있는 패턴의 가짓수 자체는 2-input 게이트들로 표현할 경우에 비해 절반 이하로 줄어들어 4-input NAND 게이트의 경우 2-input NAND 게이트와 인버터로 표현 가능한 형태가 다섯가지임에 반해 4-input이 지원될 경우 한가지 패턴만으로 표현 가능하다. Compound 게이트가 지원될 경우, 3개나 4개의 입력을 가진 게이트를 base function으로 정하여 decomposition한 회로보다 2-input 게이트로 decomposition한 회로가 커버링에 유리하다. 그림3의 (a)에 라이브러리에서 지원되는 compound 게이트 OAI21을 보이고 base function으로 인버터와 2-input 게이트를 지원할 경우에는 OAI21 게이트가 매칭 가능하지만 DAGON 시스템에서는 매칭될 수 없는 대상 회로를 보였다.

주어진 대상 회로에 라이브러리 게이트를 매칭시키는 과정에서 NAND 게이트는 DAGON 시스템의 base function이기 때문에 2-input 게이트로 쪼개지지 않아 매칭이 이루어지지 않으며 커버의 질을 떨어뜨리는 결과를 가져온다. 일반적으로 2-input NAND 게이트와 인버터만으로 대상 패턴을 정의하여 그래프 커버링을 행할 때, 보다 우수한 회로 결과를 얻을 수 있는 것으로 알려져 있다.[4]

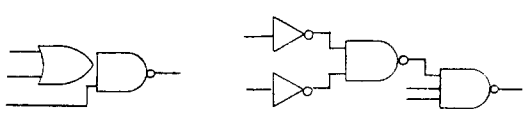



**그림 3.** Base function 선택에 따른 문제점
(a) 라이브러리에서 제공되는 OAI21 게이트
(b) 대상 회로

**Fig. 3.** Problem in base function selection.
(a) OAI21 gate supplied by library,
(b) target circuit.

2) Decomposition

Decomposition은 입력 회로를 base function으로 재구성하는 과정으로 대상 게이트는 입력 갯수가 3개 이상인 게이트이다. 문제는 어느 입력쌍을 택하여 decompose할 경우 비용을 감소시킬 수 있는가에 있으며 SiLOS-Ⅱ는 delay를 비용으로 택하였다. Silos-Ⅱ는 DFS(depth first search)에 의한 decomposition을 수행하므로 대상 게이트의 입력단에서의 decomposition은 모두 수행되어 있는 상태이다. 라이브러리에서 제공되는 게이트에 의한 게이트 지연 시간을 계산하기에 앞서 대상 노드의 입력을 구성하는 각각의 노드들에 라이브러리에서 제공되는 게이트들의 내부 지연 시간을 abstract delay로 주어 인버터는 $d_I$, 2-input NAND 게이트는 $d_{2N}$의 내부 지연 시간을 가진다고 가정하여 신호 대기 시간 (ready time)을 계산한다. 신호 대기 시간은 각 노드i에서 다음과 같이 계산되어 진다.

$$\text{신호 대기시간(i)} = \text{노드 내부 지연시간(i)} + \max_{j \in fanin(i)} [\text{신호 대기시간(j)}]$$

대상 NAND 노드에서 입력들의 신호 대기 시간이 가장 작은 두 입력을 택하여 NAND2, 인버터쌍의 입력으로 만든다. 이 과정을 반복하여 적용함으로써 abstract delay에 의한 대상 회로의 동작 시간을 감소시키는 방향으로 회로를 구성한다.

## Ⅳ. 패턴 매칭

대상 회로의 logic cone을 구성하는 게이트들은 라이브러리의 게이트들과 비교하여 매핑 가능한지의 여부를 검사한다. 트리로 나타난 부분 회로들의 각 내부 노드들에서 매핑 가능한 게이트를 검색하며 SiLOS-Ⅱ에서는 라이브러리의 게이트들의 패턴을 스트링으로 표현하여 패턴 매칭을 수행하였다.

### 1. 패턴 라이브러리

1) 라이브러리

시스템에서 지원되는 라이브러리의 형식은 게이트의 동작 기술이 간편하면서 사용자가 알아 보기 쉬운 Synopsys사의 라이브러리 형태를 따랐으며, 한 예로 그림4에 2-input NAND 게이트의 라이브러리 기술을 보였다[17] 대상 셀의 면적이 먼저 기술되고 입력단과 출력단은 나뉘어져 기술되며 입력 단자는 입력단을 구동시키기 위한 부하 용량이 정의된다. 출력은 그 게이트의 내부 지연 시간과 부하 저항이 rising과 falling시에 대해 각각 정의되어 있고 게이트의 function이 기술된다.

```
cell (ND2) {
  area:2;
  pin ("1A" "1B") {
    direction:input;
    capacitance:1.3;
  }
  pin (z) {
    direction:output;
    function:"(\"1A\"\"1B\")' " ;
    timing( ) {
      intrinsic_rise:0.39;
      intrinsic_fall:0.23;
      rise_resistance:0.0879;
      fall_resistance:0.0465;
      related_pin:"1A 1B";
    }
  }
}
```

**그림 4**. 2-input NAND 게이트의 라이브러리 기술
**Fig. 4**. Library format of a 2-input NAND gate.

2) 패턴 라이브러리

라이브러리의 한 게이트를 2-input NAND 게이트와 인버터로 표현할 경우, 가능한 패턴 수는 입력 갯수가 증가함에 따라 증가한다. 한 예로 아래 그림5의 3-input NAND 게이트의 표현 가능한 패턴은 2가지밖에 없으나 4-input NAND 게이트는 그림6과 같이 5가지이다. 이러한 패턴 트리들을 스트링으로 표현하기 위해 인버터와 2-input NAND 게이트를 하나의 문자로 나타내고 INORDER의 순서에 의해 트리 형태를 스트링으로 전환시킨다. 이를 위해 트리를 구성하는 노드들이 가질 수 있는 각각의 형태에 대하여 다음과 같이 문자를 부여한다.

1) Leaf 노드인 인버터 : 1
2) 중간 노드인 인버터 : 2
3) Leaf 노드인 NAND 게이트 : 3
4) 왼쪽 입력만이 non-primary 노드와 연결된 NAND 게이트 : 4
5) 오른쪽 입력만이 non-primary 노드와 연결된 NAND 게이트 : 5
6) 양쪽 입력 모두 non-primary 노드와 연결된 NAND 게이트 : 6

라이브러리 게이트의 패턴 트리를 구성하는 각 노드들은 위의 여섯가지 문자 중 하나로 표현 가능하므로 각각의 트리들은 내부 노드들을 INORDER로 방문하면서 각 노드를 표현하는 문자를 연결하여 만들어진 하나의 스트링으로 표현 가능하고 구분되어

질 수 있다. 위에서 보인 4-input NAND게이트는 이에 따라 다음 5가지 스트링의 패턴, 42423, 42523, 62323, 52523, 52423로 표현된다. 라이브러리를 구성하는 각각의 게이트들을 표현할 수 있는 패턴들은 모두 구하여 스트링의 형태로 패턴 라이브러리를 재구성한다.

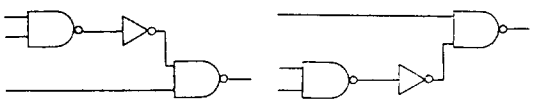


**그림 5.** 3-input NAND 게이트의 패턴
**Fig. 5.** Patterns of a 3-input NAND gate.

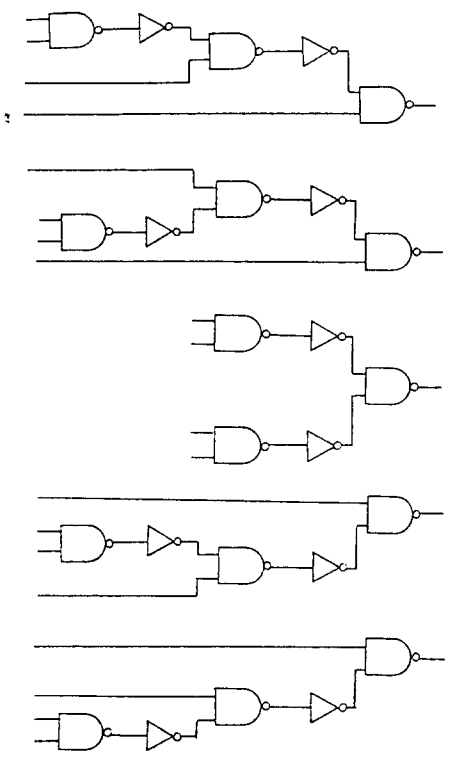


**그림 6.** 4-input NAND 게이트의 패턴들
**Fig. 6.** Patterns of a 4-input NAND gate.

2. 패턴 매칭

패턴 매칭은 라이브러리에서 제공되는 게이트들에 대하여 각각의 게이트가 가지는 스트링 패턴을 대상 회로의 각 노드에 매핑시켜 이루어진다. 한 노드에서 매핑 가능한 게이트는 여러가지일 수 있으며 매핑 가능한 게이트들 중 똑같은 패턴을 가지지만 비용 함수에 따른 시간 면적 비용 모두 뒤떨어지는 게이트는 제거된다. 만약 한 내부 노드를 루트 노드로 하는 트리가 하나의 라이브러리 게이트 패턴에 의해 leaf 노드들까지 모두 매핑된다면 그 노드에서의 패턴 매칭은 더 이상 일어나지 않으며 노드에서 유지되는 패턴은 유일해진다. 이것은 최적의 커버를 구하기 위한 탐색 범위를 줄이는 역할을 한다.

3. 인버터 주입

SiLOS에서는 재설계 방식으로 compounding, decomposition, inversion, conversion의 4가지 방법을 제안하였다.[18] 테크놀러지 매핑을 위해 그래프 커버링 방식을 이용함으로써 decomposition, conversion과 compounding 방법은 더 이상의 처리 과정 없이 구현이 가능하다. 즉 초기 회로를 2-input NAND 게이트와 인버터로 재구성해 주어 더 이상 쪼갤 수 없는 형태로 표현되므로 더 이상 decomposition과 conversion은 수행되지 않으며, 라이브러리 게이트를 문자열 형태로 바꾸어 줌으로써 compound 게이트의 패턴들도 역시 단순 게이트의 문자열 패턴들과 동일한 패턴 매칭 과정을 거치기 때문이다. Inversion 방법은 고려의 대상으로 남아 있으며 이를 해결하기 위하여 인버터 주입이 필요하다.

그림7에 인버터 주입을 위한 대상 형태를 보였다. 대상 회로의 일부가 그림7의 형태를 취하고 있을 경우, 라이브러리에서 NOR 게이트가 지원될 때 NOR 게이트의 패턴과 매칭시키는 과정에서 인버터 하나의 부족으로 인해 하나의 NOR 게이트로 매핑하지 못하고 NAND 게이트와 인버터들로 매핑하게 되는 결과를 초래한다. 이를 방지하기 위하여 패턴 매칭 과정에서 출력단이나 입력단에서 부족한 인버터를 주입하여 개선된 커버를 찾는다. 인버터 2개가 직렬 연결된 버퍼 타입은 기능상 회로의 동작에 영향을 미치지 않는다. 그러므로 인버터를 통하지 않고 직접 연결된 두 NAND 게이트는 그 사이에 버퍼가 연결되어 있는 것으로 간주하여 패턴 매칭을 수행한다. 이 버퍼는 그림7과 같은 경우에 한하여 두개의 인버터로 나누어 지며 보통의 경우에는 따로 라이브러리 게이트를 매핑시키지 않는다.

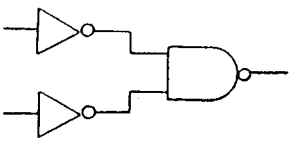


**그림 7.** 출력단에 인버터 주입이 필요한 형태
**Fig. 7.** Target form needing inverter injection at output.

## V. 커버링

기술 매핑은 매칭된 모든 패턴들을 가지고 대상 회로를 커버할 수 있는 조합을 구성함으로써 완료된다. 그림8에 루트 노드에 어느 게이트를 매핑시키는가에 따라 대상 트리의 커버가 바뀌어지는 예를 보였다.

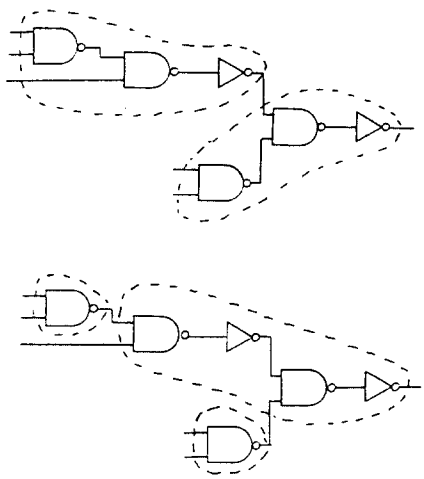


**그림 8.** 루트 노드에서의 게이트 매핑에 따른 여러 커버들

**Fig. 8.** Various covers feasible at root node.

각 노드마다 매칭된 게이트 패턴은 유일하지 않으며 그 노드에 어느 게이트를 매핑 시키는가에 의해 그 노드를 루트 노드로 하는 부분 트리 내의 노드들에 대한 매핑이 영향을 받게 되므로, 한 노드에서의 매핑 문제는 부분 트리의 다른 노드들에서의 매핑을 함께 고려해 주어야 한다. 전체 최적화를 이루기 위해서는 회로를 커버할 수 있는 모든 패턴의 조합들을 조사하여 시간 제약 조건에 맞는 조합들 중에서 최적의 비용을 가지는 커버를 구해야 한다. 이 문제는 게이트의 내부 지연 시간(gate intrinsic delay) 만을 고려할 경우 다이나믹 프로그래밍(dynamic programming) 기법으로 쉽게 해결할 수 있다[16] 그러나 기술 매핑 시스템에서는 특히 동작 시간에 대한 정확한 예측이 가능해야 하며, 게이트 내부 지연 시간만을 고려하여 회로의 동작 시간을 계산할 경우 출력단의 부하로 인해 시간 제약 조건을 만족시키지 못하는 커버를 선택할 수도 있다. 따라서 사용되는 delay model은 내부 지연 시간만이 아닌 부하량에 대한 고려가 요구된다.

### 1. Delay model

회로의 지연 시간은 다음 3가지로 구성된다.

1) 게이트의 내부 지연 시간

2) 게이트의 출력단 fanout load에 의한 지연 시간

3) 게이트와 게이트 사이에서의 wire delay

위의 지연 시간들 중 wire에 의한 delay는 실제 배치·배선이 이루어지기 이전까지 예측이 어렵기 때문에 delay model에서 제외하고 추후 post-layout 추출과 시뮬레이션을 통해 동작시간 제약 조건의 만족 여부 확인 과정을 거친다. 따라서 입력 신호가 한 게이트 g를 통과하기 위해 필요한 지연시간의 계산은 다음 식에 따른다.

$$\text{게이트 지연시간}(g) = \text{내부 지연시간}(g) + \left(\sum_{i \in fanout(g)} \text{입력용량}(i) * \text{부하저항}(g)\right)$$

### 2. 최소 면적을 위한 커버링

다단 논리 최적화 과정을 거친 회로는 extraction 과정에 의해 공통된 부분을 최대한 공유한 형태이며 최소의 literal 수를 갖도록 합성된 형태이다. 그러나 최소의 literal을 갖는 형태가 곧 최소의 면적을 갖는 회로로 기술 매핑됨을 의미하는 것은 아니며 대상 회로는 갖추어진 라이브러리에 따라 공통된 expression이 일부 중복 (duplication)된 형태에서 최소의 면적을 가질 수 있다. 회로 면적의 전체 최적화는 회로를 구성하는 fanout 노드들의 중복 가능한 형태를 모두 구하여 조합시킬 때 가능하므로 polynomial time 내에 전체 회로 면적 최적화는 불가능하다. 이러한 이유로 하나의 logic cone에서 최소 면적인 커버를 구할 경우 단지 면적이 최소인 커버를 구하기보다 fanout 노드에서의 면적을 포함한 면적이 최적인 커버를 구하는 것이 필요하다. 만약 fanout 노드를 전혀 고려하지 않는다면 회로는 중복부를 전혀 갖지않는 구조로 인식되어 개별적인 logic cone의 최적화는 이루어져도 전체적인 회로 면적 최적화는 이룰 수 없게 된다. 그림9는 대상 회로에서 fanout 노드를 가진 회로에서의 매핑시 fanout 노드를 고려했을 경우 면적 최적화된 회로와 그렇지 못한 회로의 차이를 보인다.

1) Logic cone에 대한 면적 최적화

대상 logic cone에서 게이트 매핑은 leaf 노드부터 루트 노드까지 올라가면서 실행한다. 대상 logic cone 내의 어느 한 내부 노드를 루트 노드로 하는 부분 트리에 대한 커버의 면적 계산 알고리듬은 그림10에 나타내었다.

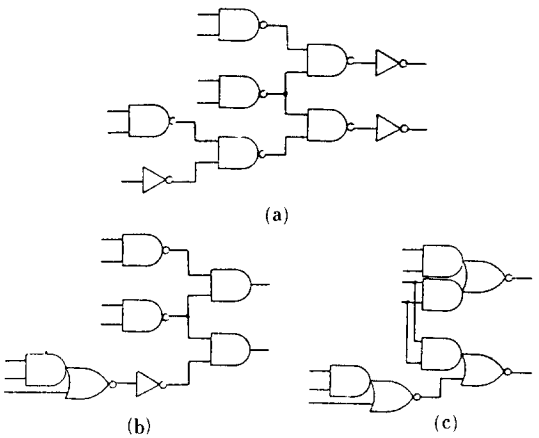



그림 9. Fanout 노드를 고려하지 않아 최적화에 실패한 예
(a) 대상 회로
(b) 면적 최적화에 실패한 매핑
(c) 면적 최적화된 매핑

Fig. 9. An example circuit where optimal result cannot be obtained due to neglecting fanout node.
(a) target circuit, (b) non-optimal mapping, (c) optimal mapping.

```
Compute_min_area(r, G)
/*
 *r:object node.
 *G:a set of library gates for the subtree rooted
 *   at node r.
 *M(r):gate mapped in node r.
 *A(g):area of matched gate g.
 *A_min(r):total area for the tree rooted at r
 *   stored at the local data structure for r.
 */
{
  if r is a leaf node then return;
  Compute_min_area (left_fanin_node);  /* depth first
                                          search */
  Compute_min_area (right_fanin_node);
  min_area=∞;
  for each g in G do
    H=a set of fanin nodes of g, except fanout nodes;
    if A(g)+∑_{h∈H} A_min(h) of M(h)<min_area then
      min_area=A(g)+∑_{h∈H} A_min(h);
      A_min(r)=min_area;
    M(r)=min_cost (M(r), M(g));
```

그림10. 최소 면적 커버를 위한 알고리듬
Fig. 10. Algorithm for finding the minimum area cover.

최소 면적을 갖는 커버를 구하기 위한 매핑 게이트의 선택은 leaf 노드부터 시작된다. 대상 노드가 leaf 노드일 경우 노드에서의 최소 면적은 매핑된 게이트만의 면적이고, 대상 노드가 내부 노드일 경우 매핑 가능한 여러가지 게이트들 각각에 대하여 그게이트를 매핑시켰을 경우 얻을 수 있는 커버들의 최소 면적을 구하고 그 중 최소의 비용을 가지는 커버를 선택한다. 각각의 커버에 대한 면적 계산은 그 내부 노드에서 매칭된 게이트가 가지는 자체 면적과 그 게이트를 매핑시켰을 경우 입력이 되는 노드에서 가지는 최소 면적의 커버가 가지는 면적을 더한다. 이를 위하여 입력 노드들은 이미 최소 면적을 가지는 커버 계산이 이루어져 있어야 하며 면적 계산은 leaf 노드부터 시작되므로 한 내부 노드가 면적 계산의 대상 노드가 되었을 때 leaf 노드에서 대상 노드 사이에 놓인 내부 노드들에 대한 최소 면적 계산은 모두 이루어진 상태이다. 입력 노드가 fanout 노드일 경우에는 그 면적을 커버의 면적으로 고려하지 않음으로써 초기 구조를 최대한 보장하며 fanout 노드를 포함하여 매칭된 게이트는 그 면적을 더해 주어 logic의 중복을 면적 계산에 반영하였다. 커버의 선택은 내부 비용 함수에 의해 이루어진다.

2) 비용 함수

면적 최적화를 위한 비용 함수는 5가지의 구성 인자로 이루어진다.

cost of area=F(area, cell, f(cell, eq_gate), g(area, eq_gate), real_area)

area:fanout 노드를 제외한 트리에서 매핑된 게이트의 총 면적.

cell:fanout 노드를 제외한 트리에서 매핑된 cell의 총 갯수.

f(cell, eq_gate):커버에 포함된 equivalent gate와 cell 수의 비.

g(area, eq_gate):커버의 면적과 equivalent gate의 비.

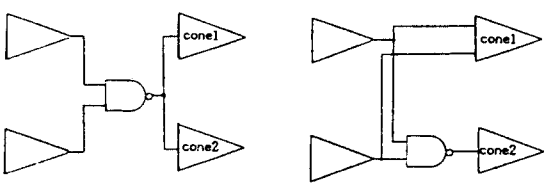
real_area:fanout 노드를 포함하는 logic cone에서의 총 면적.

최소 면적을 구하는 과정에서 고려된 equivalent gate는 NAND2 게이트의 면적을 1로 정의하며 입력 갯수가 2보다 커질 때마다 0.5씩 더하여 준다.[1] AND나 OR 게이트는 입력 갯수외에 인버터의 갯수 0.5를 더한다. 함수 f와 g는 literal 수를 면적으로 삼았을 경우의 수치에 대해 조정되어 있으며 f와 g의 값은 area에 따른 변화가 심하므로 라이브러리에 따라 그 의미를 상실할 수 있다. 이 의존도를 줄이기 위하여 라이브러리의 NAND 게이트의 면적과 literal의 비를 normalized constant로 구하여 f와 g에 곱해준다. 이 두 함수는 최종 커버에서 포함되는 cell의 수를 줄여 주는 역할을 한다. Real_area는 fanout 노

드를 포함한 면적으로 전체 최적화를 위하여 fanout 노드를 포함하는 커버가 선택될 확률을 높여 준다.

### 3. 부분 면적 최적화

Cost function에 의해 각 primary output의 transitive fanin에 대한 개별적인 면적 최적화를 수행하여 구한 커버는 다른 logic cone의 커버링 결과를 고려하지 않은 결과이므로 각 logic cone들의 개별적인 최적화 과정에서 그림11과 같이 하나의 2-input NAND 게이트가 매핑된 회로에 나타날 수 있다. Cone 1의 커버가 fanout 노드를 포함시킴에 따라 fanout 노드는 cone1, cone2에 중복되어 나타난다. Cone 2에 중복되어 나타난 fanout 노드의 면적은 최소 면적 계산시 cone2의 커버의 면적에 가산시키지 않았기 때문에 cone2의 커버에 가산시켜야 한다. 이에 따라 중복이 일어난 fanout 노드에서 cone2를 따른 커버링은 최소 면적의 커버가 아닐 수 있으므로 이러한 경우에는 cone2에 대한 부분 면적 최적화가 필요하며 대상 회로에서 부분 최적화 대상부를 결정하기 위해 대상 형태가 발생한 노드에서 root 노드까지의 path를 mark한다. 대상 path를 구성하는 노드들은 부분 최적화 대상이 된다. 대상 path에 대해서는 최소 면적 커버링을 재수행하며 이 과정에서 비용함수는 초기 커버링에서의 확률 요인을 최소한으로 주어 최소의 면적을 찾도록 조정된다. 때로는 fanout 노드의 중복이 일어나도 기존의 커버의 비용이 더 좋기 때문에 primary 노드에서의 커버가 변하지 않는 경우가 있으며 fanout 노드에서의 중복이 일어나지 않거나 primary 노드의 커버가 고정될때까지 반복하여 부분 최적화를 수행한다.




(a) (b)

그림11. 부분 면적 최적화의 대상 형태
(a) 초기 회로
(b) fanout 노드에서의 중복이 있는 회로

Fig. 11. Target form of local area optimization.
(a) Initial circuit,
(b) circuit having duplicated fanout node.

### 4. 최단 동작 시간을 위한 커버링

대상 회로는 decomposition 과정에서 임계 경로의 길이를 가능한 한 작게 줄인 형태이며 최단 동작 시간을 가지는 커버를 찾기 위하여 신호대기 시간을 계산한다. 라이브러리를 구성하는 게이트와 delay model을 이용하여 보다 정확한 신호 대기 시간을 구하고 최단의 신호 대기 시간을 가지는 커버를 선택한다. 내부 지연 시간만을 고려하여 커버링을 수행할 경우 단순히 대상 노드에서 최단의 신호 동작 시간을 가지는 커버를 선택하면 되지만 부하량을 고려하여 최단의 신호 대기 시간을 가지는 커버를 구하기 위해서는 출력단의 부하가 결정되기 전까지 선택할 수 없다. 기술 매핑 과정에서 부하량을 고려해 줄 때의 문제점은 대상 노드를 입력으로 하는 게이트의 입력 커패시턴스에 따라 최소의 동작 시간을 가지는 커버가 결정되며 어느 게이트가 출력단에 매핑되는지는 primary 출력까지 기술 매핑이 수행되어야 결정되므로 fanout 노드에서 매핑될 게이트의 출력 갯수 예측이 불가능하다는 점이다. SiLOS-Ⅱ에서는 fanout 노드의 출력 갯수를 게이트의 출력 갯수로 하고 만약 노드의 출력 갯수가 fanout 제약에 위배될 경우 노드에 매핑된 게이트의 최대 허용 fanout 갯수를 노드의 출력 갯수로 하였다. Fanout 노드에서 중복이 일어날 경우에는 중복이 일어난 게이트의 입력 노드 출력 갯수를 하나씩 늘려서 계산하였다. 그림12에 최단 동작 시간의 커버를 구하기 위한 알고리듬을 나타내었다.

### 5. 시간 제약 조건하에서 최적화된 회로

최소의 면적을 갖는 커버나 최단 동작 시간을 갖는 커버를 구하는 것은 leaf 노드에서 루트 노드로 올라가면서 노드마다 하나의 커버, 즉 최소의 면적을 갖는 커버 혹은 최단 신호대기 시간을 갖는 커버만을 선택하여 이루어질 수 있었다. 그러나, 동작 시간의 제약 조건이 사용자로부터 주어질 경우 최소의 면적을 가지는 커버가 동작 시간의 제약에서 벗어날 수 있고 최단 동작 시간을 갖는 커버가 큰 면적을 가질 수 있으므로 동작 시간을 만족시키면서 최소의 면적을 갖는 커버를 구한다.

제약이 주어지지 않고 최단 동작 시간만을 구하는 경우, leaf 노드에서 루트 노드로 게이트의 매핑이 진행되어 나가는 과정에서 대상 노드에 매핑 가능한 여러 게이트들 중 하나를 선택하는 decision making이 일어난다. 제약이 주어지지 않기 때문에 구하는 커버에서 고려되어야 할 사항은 시간의 최소치이며,

```
Compute_min_ready_time(r, G)
/*
 *r:object node
 *G:a set of library gates for the subtree rooted
 *  at node r.
 R(g):intrinsic gate delay
 */
{
  if r is a leaf node then return;
  Compute_min_ready_time (left_fanin_node) ;
  Compute_min_ready_time (right_fanin_node) ;
  min_ready_time=∞;
  for each g in G do
    H=a set of fanin nodes of g;
    R_max=max_{h∈H}(C_g*R_load+R(h));
    if R(g)+R_max<min_ready_time then
      min_ready_time=R(g)+R_max;
  if r is a leaf node then return;
    R_min(r)=min_ready_time;
}
```

**그림12.** 최단 신호대기 커버를 위한 알고리듬
**Fig. 12.** Algorithm for finding the cover with minimum ready time.

면적의 고려없이 최단 동작 시간의 커버를 찾는 것만으로 충분하다. 그러나 시간 제약이 주어질 경우 시간 제약을 만족시키는 모든 커버들이 해로써 가능하며 최단 동작시간을 갖는 커버 역시 가능한 여러 해들 중 하나이다. 시간 제약을 만족시키는 여러 커버들은 모두 해로 선택 가능하므로 면적을 선택의 기준으로 삼는다. 이러한 이유로 각 노드에서 매핑 가능한 모든 커버들에 대한 정보가 신호 대기 시간 순으로 정렬되어 유지되고 최소 신호대기 시간을 갖는 커버를 노드의 대표 커버로 한다. 시간제약 하에서의 커버 선택은 다음 3가지 과정을 거친다.

1. Leaf 노드로 부터 루트 노드에 이르는 과정에서 매핑 가능한 모든 커버들이 각 노드에서 유지되며 루트 노드에는 대상 트리에서 가능한 모든 게이트 조합의 커버가 신호대기 시간 순으로 정렬되어 있다.

2. 이 루트 노드에서 유지되는 커버들 중 동작 시간제약을 만족시키지 못하는 커버들은 모두 제거되며, 루트 노드에서 조건을 만족시키지 못하는 커버들을 제거하는 기준으로 사용된 시간 제약은 내부 노드들의 커버들을 줄이기 위해 입력단으로 전달된다. 라이브러리 게이트 g가 한 노드 r에서 매핑 가능한 조건을 만족할 경우 그 게이트의 입력이 되는 노드 i에서의 시간 제약은 다음과 같이 계산되어 진다.

$TC_i = TC_r - R_g * C_g - delay_g$ ($TC_i$ : 입력단 노드의 시간 제약, $TC_r$ : 대상 노드의 시간 제약)

내부 노드에서 유지되고 있는 커버들 중 이 시간 제약을 만족시키지 못하는 커버는 제거된다. 이 과정을 루트 노드로 부터 leaf 노드까지 계속하면 남는 커버들은 모두 시간제약 조건을 만족시키게 된다.

3. 남은 커버들에 대하여 최소의 면적을 갖는 커버를 찾는 작업을 한번 더 반복하면 주어진 시간 제약을 만족하면서 최소의 면적을 갖는 커버를 구한다.

## Ⅵ. 실험 결과

SiLOS-Ⅱ는 UNIX 환경하에서 C로 구현되었으며, 구현된 시스템의 성능을 평가하기 위해 표준 테스트 회로로 채택된 IWLS86 벤치마크 프로그램[8]에 TECHMAP 시스템에서 사용된 MCNC 라이브러리의 수정 라이브러리를 사용하여 기술 매핑을 수행하였다. 라이브러리에서 지원되는 게이트들은 literal 수를 게이트의 면적으로 정의하였고 라이브러리는 MCNC 라이브러리의 XOR, XNOR 게이트를 제외한 나머지 게이트들과 AOI22, OAI22 게이트로 구성된다. 이들을 이용하여 최소 면적 모드 (area mode)와 최소 cell 수 모드 (cell mode)에서 IWLS86 벤치마크에 대한 기술 매핑을 수행한 결과를 표1에 보였다.

Cell 수 최적화 모드는 면적 최적화 모드에 비해 평균 4.24% 면적의 증가를 가져온 반면 cell 수는 4.54% 감소하였다. 회로 f2와 con1과 같은 작은 회로에 대해서는 두 모드에서의 결과 차이가 없으나 bw와 같은 큰 회로에 대해서는 각각 10% 이상의 차이를 보였다. Cell 수 최적화 모드에서 면적과 cell 수 모두 좋은 결과를 얻은 vg2는 다른 회로들에 비해 많은 fanout 노드들을 가지고 있어 cell 수를 중점적으로 줄인 것이 유효하였다. 회로 분석 결과 fanout 노드가 많은 회로일수록 cell 수의 비용을 높게 주는 것이 면적을 줄일 수 있음을 알 수 있었다.

시스템간의 성능 비교를 위해 SiLOS-Ⅱ의 면적 모드에서 기술 매핑한 결과와 SKOL, DAGON 및 TECHMAP의 결과를 equivalent gate와 cell 수의 term으로 비교한 결과를 표2에 보이고, SKOL에서 기술 매핑 시스템의 성능 기준으로 제시한 literals / cell, literals/eq. gate, eq. gates/cell을 이용하여 SKOL과 비교한 결과를 표3에 보였다.

표 1. IWLS86 벤치마크에 대한 SiLOS-Ⅱ의 기술 매핑 결과
**Table 1.** Technology mapping results of SiLOS-Ⅱ on IWLS 86 benchmark.

| Circuit | 최소 면적 모드(area) | | | 최소 cell 수 모드(cell) | | | area : cell 비율(%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | #lits | #Eq. g | #Cells | #lits | #Eq. g | #Cells | △lits | △Eq. g | △Cells |
| misex1 | 75 | 50.5 | 32 | 85 | 61.5 | 33 | 13.3 | 21.8 | 3.1 |
| misex2 | 162 | 86 | 79 | 164 | 97 | 66 | 1.2 | 12.8 | −16.5 |
| f2 | 32 | 24 | 14 | 32 | 24 | 14 | 0 | 0 | 0 |
| vg2 | 135 | 77.5 | 62 | 134 | 81 | 60 | − 0.7 | 4.5 | − 3.2 |
| con1 | 32 | 24 | 15 | 32 | 24 | 15 | 0 | 0 | 0 |
| bw | 268 | 178 | 104 | 320 | 227 | 91 | 19.4 | 27.5 | −12.5 |
| rd53 | 66 | 49 | 24 | 68 | 52 | 24 | 3.0 | 6.1 | 0 |
| rd73 | 130 | 103 | 50 | 138 | 109 | 50 | 6.2 | 5.8 | 0 |
| f51m | 172 | 118 | 70 | 186 | 136 | 67 | 8.1 | 15.3 | 4.3 |
| 5xp1 | 192 | 130 | 77 | 197 | 152.5 | 66 | − 2.6 | 17.3 | −14.3 |
| z4 | 61 | 47.5 | 25 | 61 | 48.5 | 25 | 0 | 2.1 | 0 |
| sao2 | 198 | 142 | 73 | 204 | 154 | 28 | 3.0 | 8.5 | − 6.8 |

표 2. SiLOS-Ⅱ, SKOL, DAGON 및 TECHMAP의 성능 비교
**Table 2.** Performance comparision of SiLOS-Ⅱ, SKOL, DAGON and TECHMAP.

| Circuti | SiLOS-II | | SKOL | | DAGON | | TECHMAP | |
|---|---|---|---|---|---|---|---|---|
| | #Eq. g | #Cells | #Eq. g | #Cells | #Eq. g | #Cells | #Eq. g | #Cells |
| misex1 | 50.5 | 32 | 55 | 39 | 44.5 | 42 | 42 | 41 |
| misex2 | 86 | 79 | 101 | 80 | 83 | 90 | 81.5 | 85 |
| f2 | 24 | 14 | 16 | 14 | 20 | 16 | 18 | 18 |
| vg2 | 77.5 | 62 | 74 | 58 | 70.5 | 68 | 78 | 84 |
| con1 | 24 | 15 | 16 | 12 | 15.5 | 14 | 18 | 20 |
| bw | 178 | 104 | 143.5 | 93 | 140 | 138 | 117 | 112 |
| rd53 | 49 | 24 | 36.5 | 26 | 32.5 | 28 | 24.5 | 15 |
| rd73 | 103 | 50 | 99 | 64 | 77.5 | 64 | 59.5 | 52 |
| f51m | 118 | 70 | 107 | 72 | 99.5 | 86 | 71 | 65 |
| 5xp1 | 130 | 77 | 102.5 | 75 | 84.5 | 79 | 67.5 | 66 |
| z4 | 47.5 | 25 | 37 | 27 | 30.5 | 24 | 24 | 17 |
| sao2 | 142 | 73 | 146.5 | 98 | 112 | 86 | 122.5 | 115 |
| total | 1029.5 | 625 | 934 | 658 | 810 | 735 | 723.5 | 690 |
| △ % | − | − | − 9.28 | + 5.28 | − 21.32 | + 17.6 | − 29.72 | + 10.4 |

표2의 equivalent gate 수는 다단논리 최적화 시스템과 기술 매핑 시스템을 포함한 논리 설계 시스템의 성능을 나타낸다. SiLOS-Ⅱ와 타 시스템과의 비교 결과 cell수는 5−17% 감소된 반면 equivalent gate는 10−30% 증가된 수치를 보여주며 특히 TECHMAP 시스템과 30%에 가까운 차이를 보인다. 결과의 차이를 분석하면 기술 매핑의 결과는 회로의 초기 구조에 의존하므로 equivalent gate 수의 차이에 대한 원인으로 논리 설계의 front-end인 다단 논리 최적화 시스템의 성능과 TECHMAP에 구현된 기술 매핑을 위한 별도의 decomposition 과정에 의한 회로의 초기 구조 차이를 들 수 있다. 또한 SiLOS-Ⅱ, SKOL 및 DAGON의 라이브러리를 구성하는 게이트들의 면적이 literal 수임에 반해 TECHMAP의 라이브러리는 equivalent gate를 면적으로 주어 보다 효과적인 equivalent gate 수의 감소를 가져올 수 있

표 3. SiLOS-Ⅱ와 SKOL의 성능 비교
**Table 3**. Performance comparision of SiLOS-Ⅱ and SKOL.

| Circuit | SiLOS-II | | | SKOL | | |
|---|---|---|---|---|---|---|
| | Lit/Eq. g | Lit/Cell | Eq. g/Cell | Lit/Eq. g | Lit/Cell | Eq. g/Cell |
| misex1 | 1.49 | 2.34 | 1.58 | 1.40 | 1.97 | 1.41 |
| misex2 | 1.88 | 2.05 | 1.09 | 1.29 | 1.63 | 1.26 |
| f2 | 1.33 | 2.29 | 1.71 | 1.50 | 1.71 | 1.14 |
| vg2 | 1.74 | 2.18 | 1.25 | 1.24 | 1.59 | 1.28 |
| con1 | 1.33 | 2.13 | 1.60 | 1.19 | 1.58 | 1.33 |
| bw | 1.51 | 2.58 | 1.71 | 1.37 | 2.11 | 1.54 |
| rd53 | 1.35 | 2.75 | 2.04 | 1.40 | 1.96 | 1.40 |
| rd73 | 1.26 | 2.60 | 2.06 | 1.33 | 2.06 | 1.55 |
| f51m | 1.46 | 2.46 | 1.69 | 1.24 | 1.85 | 1.49 |
| 5xp1 | 1.48 | 2.49 | 1.69 | 1.29 | 1.76 | 1.37 |
| z4 | 1.28 | 2.44 | 1.90 | 1.40 | 1.93 | 1.37 |
| sao2 | 1.39 | 2.71 | 1.95 | 1.28 | 1.92 | 1.49 |
| avg. | 1.46 | 2.42 | 1.69 | 1.31 | 1.86 | 1.42 |

었다. 실제 기술 매핑 시스템만의 성능을 평가하는 척도인 literals/eq. gate, eq. gates/cell 및 literals/cell을 구하여 SKOL과 비교한 결과는 SiLOS-Ⅱ가 우수함을 보여준다.

## Ⅶ. 결 론

본 논문에서는 다이나믹 프로그래밍에 의한 기술 매핑 시스템의 설계에 대하여 설명하였다. 시스템은 대상 회로를 logic cone으로 분할하는 휴리스틱을 사용하였고 면적 최적화를 위해 cell 수와 fanout 노드를 고려하는 새로운 비용 함수를 제안하였다. 제안된 비용함수의 이용으로 전체 회로를 고려한 면적 최적화가 가능하였으며 logic cone으로 분할을 행할 경우 발생할 수 있는 면적 손실을 없애기 위하여 부분 최적화를 수행함으로써 logic cone의 ordering 문제를 해결하였다. 또한 회로의 동작 시간 계산시 부하량을 고려하여 줌으로써 보다 정확한 동작 시간의 예측이 가능하였다.

현재 시간제약이 만족되지 않을 경우 회로의 구조를 바꾸어 해결하는 timing optimization에 대한 연구와 fanout 제약 하에서의 회로 최적화에 대한 연구가 진행되고 있으며 제안된 비용 함수를 회로에 따라 dynamic하게 적용시키도록 개량하는 연구가 진행되고 있다. 앞으로 VLSI 시스템 설계 자동화 툴 환경하에 개발중인 Silicon Compiler에 integration을 위한 작업이 계속될 것이다.

## 參 考 文 獻

[1] R.A. Bergamaschi, "SKOL: A System for Logic Synthesis and Technology Mapping," *IEEE Trans. CAD of Int. Circ. Syst.*, vol. CAD-10, no. 11, Nov. pp. 1342-1355, 1991.

[2] R.K. Brayton, G.D. Hachtel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli. "Logic Minimization Algorithms for VLSI Synthesis," Kluwer Academic Pub., 1984.

[3] R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. CAD of Int Circ. Syst.*, vol. CAD-6, no. 6, Nov. pp. 1062-1081, 1987.

[4] R.K. Brayton, A.L. Sangiovanni-Vincentelli, and G.D. Hachtel, "Multilevel Logic Synthesis," *Proceedings of IEEE*, vol. 78, no. 2, Feb. pp. 264-300, 1990.

[5] K. Bartlett, W. Cohen, A. de Geus, and G.D. Hachtel "Synthesis and Optimization of Multilevel Logic under Timing Constraints," *IEEE Trans. CAD of Int. Circ. Syst.*, vol. CAD-5, no. 4, Oct. pp. 582-596, 1986.

[6] J.A. Darringer, D. Brand, J. Gerbi, W.H. Joyner Jr., and L. Trevillyan, "LSS: A System for Production Logic Synthesis," *IBM J. Research and Development*, vol. 28, no. 5, Sept. pp. 537-545, 1984.

[7] E. Detjens, G. Gannot R. Rudell, A.

Sangiovanni-Vincentelli, and A. Wang, "Technology Mapping in MIS," *in Proc. ICCAD,* Nov. pp. 116-119, 1987.

[8] A. de Geus, "Logic Synthesis and Optimization Benchmarks for the 1986 Design Automation Conference," *in Proc. 23rd Design Automation Conf.*, June p. 78, 1986.

[9] D. Gregory, K. Bartlett, A. de Geus, and G. Hachtel, "SOCRATES: A System for Automatically Synthesizing and Optimizing Combinational Logic," *in Proc. 23rd Design Automation Conf.*, June pp. 79-85, 1986.

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

[11] K. Keutzer and M. Vancura, "Timing Optimization in a Logic Synthesis System," in 'Logic and Architecture Synthesis for Silicon Compilers' G. Saucier and P.M. McLellan (ed.). North-Holland, 1989.

[12] R. Lipsett. "VHDL: Hardware Description and Design," Kluwer Academic Pub., 1989.

[13] C.R. Morrison, R.M. Jacoby, and G.D. Hachtel , "TECHAMP. Technology Mapping with Delay and Area Optimization," in 'Logic and Architecture Synthesis for Silicon Compilers' G. Saucier and P.M. McLellan (ed.), North-Holland, 1989.

[14] M. Pedram and N. Bhat, "Layout Driven Technology mapping " *in Proc. 28th Design Automation Conf.*, June pp. 99-105, 1991.

[15] K.J. Singh, A.R. Wang, R.K. Brayton, and A. Sangiovanni-Vincentelli, "Timing Optimization of Combinational Logic," *in Proc. ICCAD,* Nov. pp. 282-285, 1988.

[16] H.J. Touati, C.W. Moon, and R.K. Brayton, "Performance-Oriented Technology Mapping," *in Proc. of the Sixth MIT Conference,* MIT Press, pp. 79-97, 1990.

[17] "Library Compiler," *Reference Manual Version 1.2, Synopsys Inc.,* Nov. 1989.

[18] 이재형, 황선영, "성능 구동논리회로 자동 논리 설계 시스템," 대한 전자공학회 논문지, 28권 A편 1호, 1991년 1월, pp. 74-84.

[19] 임춘석, 김태선, 황선영, "SMiLE:다단 논리 최적화 시스템," 한국 정보과학회 가을 학술발표 논문집, 1990년 10월, 17권 2호, pp. 689-692.

## 著 者 紹 介

**金 泰 善**(準會員)
1968年 1月 18日生. 1990年 8月 서강대학교 전자공학과 졸업. 1990年 8月～현재 서강대학교 전자공학과 대학원 석사과정. 주관심분야는 CAD 시스템, Computer Architecture 및 VLSI설계 등임.

**黃 善 泳** (正會員) 第29卷 A編 第1號 參照
현재 서강대학교 전자공학과 교수