論文93-30A-12-14

멀티플렉서 구조의 FPGA를 위한 BDD를 이용한 논리 합성 알고리듬

(Logic Synthesis Algorithm for Multiplexer-based FPGA's Using BDD)

姜 圭 顯*, 李 載 興*, 鄭 正 和*

(Kyu Hyeon Kang, Jae Heung Lee and Jong Wha chong)

要 約

본 논문에서는 멀티플렉서 구조를 갖는 FPGA에 대한 테크놀로지 매핑 알고리듬을 제안한다. 본 논문의 알고리듬은 세가지 단계로 구성된다. FPGA상에 실현하고자하는 대상 논리함수와 FPGA의 단위 논리모듈을 BDD(Binary Decision Diagram)로 모델링한 후, 단위 논리모듈의 BDD로 대상 논리함수의 BDD를 커버링하며, 마지막으로 cell merging과정을 통하여 대상 논리함수의 실현시 사용되는 단위 논리모듈의 수를 감소시키는 방법을 제안한다. BDD를 구성할때, 대상 논리함수의 입력 변수를 오더링하는데 있어서 binate selection을 이용하여 BDD가 대칭적이며 또한 작은 레벨을 갖도록 하였다. 이렇게 함으로써 실현된 회로의 면적과 delay time을 감소시킬 수 있다.

기존의 연구방법은 대상 논리함수를 단위 모듈에 매핑할때 하나의 2입력 또는 3입력 부(副) 함수를 실현하는데 하나의 단위 논리모듈을 사용하지만 본 논문에서 제안된 알고리듬은 각각의 2입력 혹은 3입력 부 함수의 쌍을 하나의 단위 논리모듈에 할당함으로써 매핑의 결과를 상당히 개선할 수 있다. 기존의 연구결과와 본 논문에서 제안된 알고리듬의 매핑 결과를 비교함으로써 본 논문의 효율성을 검증한다.

Abstract

In this paper we propose a new technology mapping algorithm for multiplexer-based FPGA's. The algorithm consists of three phases: First, it converts the logic functions and the basic logic module into BDD's. Second, it covers the logic function with the basic logic modules. Lastly, it reduces the number of basic logic modules used to implement the logic function after going through cell merging procedure. The binate selection is employed to determine the order of input variables of the logic function to constructs the balanced BDD with low level. That enables us to constructs the circuit that has small size and delay time.

Technology mapping algorithm of previous work used one basic logic module to implement a two-input or three-input function in logic functions. The algorithm proposed here merges almost all pairs of two-input and three-input functions that occupy one basic logic module, and improves the mapping results. We show the effectiveness of the algorithm by comparing the results of our experiments with those of previous systems.

I. 서 론

IC (Integrated Circuit)의 집적도가 VLSI의 단계에 이르면서 설계자가 경험등에 의존하여 수작업으

*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 2月 16日

로 설계하는 일이 어려워지게 되고 따라서 디지탈 시스템 설계의 정확성과 시간단축을 위하여 CAD (Computer Aided Design)기술을 VLSI설계에 도입하는 설계 자동화 시스템에 대한 요구가 증가하고 있다. 특히, ASIC(Application Specific Intergrated Circuit)의 활용이 활발해지면서 기존의 Standard Cell 또는 Gate Array 방식보다 설계시간과 설계비용을 상당히 감소시키는 FPGA(Field Programmable Gate Array)의 개발로 이전의 ASIC 시장에 커다란 영향을 미치게 되었고, 이에따라 FPGA에 대한 연구가 활발하게 진행되는 추세에 있다.

FPGA는 PAL(Programmable Array Logic)의 특성인 사용자가 programming하기에 용이한 점과 Gate Array의 특성인 다양성을 결합한 디바이스로서 다단(multi-level)논리함수의 실현이 용이하며, 크게 Xilinx사의 lookup-table 방식과 Actel사의 Multiplexer구조를 기본으로 하는 방식으로 분류할 수 있다. Xilinx 방식은 기본적으로 I/O block, 각각의 logic을 실현하는 CLB(Configurable Logic Block)와 Interconnection area로 구성되며 내부의 static RAM을 통하여 CLB와 배선을 결정한다. Actel 방식은 단위 논리 모듈과 배선을 anti-fuse technology를 이용하여 논리함수를 실현 한다. 그림 1은 Actel 방식의 FPGA의 단위 논리모듈을 나타낸 것인데 3개의 멀티플렉서와 하나의 OR 게이트로 이루어져 있다.

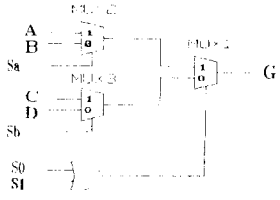


그림 1. Multiplexer-based FPGA의 단위 논리 모듈

Fig. 1. A basic logic module of the multiplexer-based FPGA.

다단논리회로의 설계는 기본적으로 기술독립적인 논리최적화단계와 셀 라이브러리등의 테크놀로지를 고려한 테크놀로지 매핑(technology mapping)으로 구성된다. 테크놀로지 매핑은 논리최적화 단계를 거친 조합논리회로를 실제의 테크놀로지에서 제공되는 셀로 재구성하는 과정으로, 요구되는 회로의 면적이나 동작속도등을 만족하는 회로를 실현하는 것이라 할 수 있는데 임의의 대상 논리회로를 FPGA로 실현할 때 어떠한 방법으로 FPGA의 단위 논리모듈을 매핑하는가에 따라서 사용되는 단위 논리모듈의 갯수와 회로의 레벨(level) 즉 실현된 회로의 크기와 딜레이 타임이 서로 다르게 된다. 따라서 효율적인 테크놀로지 매핑 알고리듬을 적용하면 회로 제작비용의 감소와 회로의 동작 속도의 개선이 가능하다.

테크놀로지 매핑 과정은 크게 라이브러리를 참조하는 매핑과 라이브러리와 무관한 매핑으로 분류할 수 있다. 전자의 경우는 MisII [4] 나 Ceres [5] 에서 처럼 대상 논리함수와 셀패턴간의 패턴매칭 문제이고 후자의 경우는 Mis-pga [6] 에서처럼 라이브러리대신 특정한 테크놀로지를 고려한 매핑문제이다.

본 논문에서는 multiplexer-based FPGA에 대한 새로운 테크놀로지 매핑알고리듬을 제안하는데 있어서 다음과 같은 접근방식을 취한다. 대상 논리함수와 FPGA의 단위 논리모듈을 모델링하는데 있어서 BDD(Binary Decision Diagram) [7] 을 도입하였으며 대상 논리함수의 BDD구성시 binate selection [8] 으로 입력변수를 오더링(ordering)한다.

기존의 Ceres [5], MisII [4] 같은 라이브러리에 의존하는 매핑 알고리듬에서는 매핑시간의 증가가 문제점으로 지적되어 왔는데, 라이브러리와 무관한 매핑방식을 택함으로써 FPGA의 특성을 최대한 활용하며 매핑과정에 융통성을 부여한다.

특히 기존의 Mis-pga [6], Proserpine [3] 같은 라이브러리와 무관한 매핑 알고리듬이 입력 변수의 ordering과 대상 논리함수의 커버링에 중점을 둠으로써 단위 논리모듈을 효율적으로 사용하지 못하였지만, 본 논문에서는 가능한 모든 2입력 또는 3입력 함수의 쌍을 cell merging과정을 통하여 적절히 하나의 모듈에 합함으로써 기존의 방법에 비하여 회로 면적의 감소와 동작속도의 개선에 있어서 효율적이다.

Ⅱ. Binary Decision Diagram

1. BDD (Binary Decision Diagram)의 기본 성질

그림 2의 예에서 볼 수 있듯이 BDD는 기본적으로 각 변수가 노드가 되고 그 노드값이 0 혹은 1인 경우에 따라 하위 노드가 결정되는 이진 트리의 성질을 갖는다. 그리고 논리 함수의 입력변수들을 어떻게 ordering하는가에 따라 동일 함수에 대한 각 노드값들이 달라지는 성질을 가진다. 즉 N개의 입력변수를 가지는 함수의 경우에 가능한 ordering의 경우의 수는 N! 이된다.

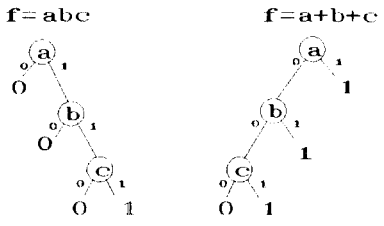


그림 2. BDD의 예
Fig. 2. Examples of BDD.

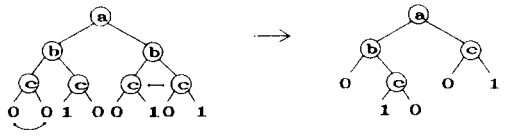


그림 3. BDD의 노드 삭제의 예
Fig. 3. An example of reducing nodes in BDD.

그림 3의 (a)의 b, c 노드처럼 BDD에서 임의의 노드의 좌 우 sub-tree가 leaf 노드까지 같으면 그림 3의 (b)와 같이 그 노드는 삭제될 수 있다. [7]

2. BDD에 대한 노드의 교환성

테크놀로지 매핑에서 이용할 수 있는 BDD의 다른 특성은 Boolean 함수식의 교환법칙을 적용할 수 있다는 것이다. 그림 4의 (a)는 함수 a'(x+y) + ab에 대한 BDD이고 노드 1과 노드 2로 이루어지는 부(副) 함수는 x + y 인데 이 함수는 y + x와 같다. 따라서 (a)의 BDD가 나타내는 함수식을 a'(y+x) + ab로 바꿀 수 있는데 이 것은 그림 4의 (b)의 BDD

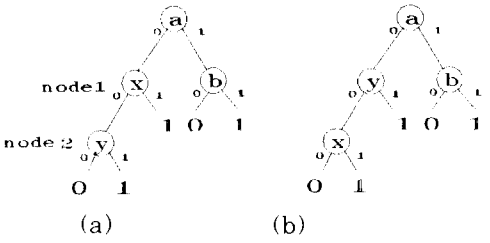


그림 4. BDD의 노드 교환의 예
Fig. 4. A example of changing nodes in BDD.

에 대한 식과 같으므로 그림 4의 (a)와 (b)의 두 BDD는 동일한 함수를 표현하고 있다. BDD내에서 sub-tree가 나타내는 함수식이 교환법칙을 만족하면 그 sub-tree들의 노드들은 서로 교환이 가능하고 이러한 성질은 BDD간의 매칭에 이용될 수 있다.

Ⅲ. HMAP 테크놀로지 매핑 알고리듬

본 논문에서 제안한 multiplexer-based FPGA에 대한 테크놀로지 매핑 알고리듬은 다음의 3 단계로 구성된다.

Step 1 Multiplexer-based FPGA의 단위 논리모듈과 대상 논리회로를 BDD로 모델링
Step 2 대상 논리회로를 단위 논리모듈로 커버링
Step 3 대상 논리함수안의 2입력 함수나 3입력 함수로 이루어지는 모듈의 쌍을cell merging 알고리듬을 이용하여 하나의 단위 논리모듈에 합한다.

1. 단위 논리모듈의 BDD표현

Multiplexer-based FPGA의 단위 논리모듈의 각 multiplexer의 selection 입력은 BDD의 각 노드와 매칭되므로 FPGA의 MUX 구조와 BDD의 특성을 고려하여 그림 5와 같이 각 MUX들의 selection 입

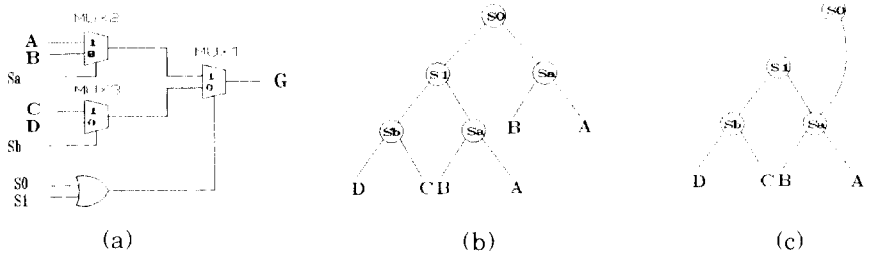


그림 5. 단위논리모듈에 대한 BDD의 실현
Fig. 5. The basic logic module and its BDD.

력을 BDD의 노드로 한다.

그림 5는 FPGA의 단위 논리모듈과 그것의 BDD를 나타낸 것인다. 단위 논리모듈의 OR 게이트는 그림 5 (b)에서처럼 불필요한 duplication을 발생시켜서 대상 논리함수를 단위 논리모듈로 covering하는데 문제를 복잡하게 만들며 그림 5의 (c)에서와 같이 cycle을 형성하여 tree pattern matching을 불가능하게 만든다. 따라서 그림 5에서 나타내고있는 단위 논리모듈내의 OR 게이트의 한 입력인 S0를 stuck-at 0로 고정하여 그림 6에 있는 간략화된 BDD를 구함으로써 매핑의 효율성을 높일 수 있도록 하였다.

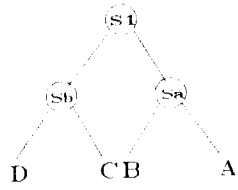


그림 6. 간략화된 단위 논리모듈의 BDD
Fig. 6. A reduced BDD of the basic logic module.

2. 대상 논리함수의 BDD표현

Multiplexer-based FPGA를 위한 테크놀로지 매핑의 목적함수는 최소의 단위 논리모듈을 사용하여 주어진 회로를 실현하는 것이다. 따라서 본 논문에서는 회로 면적의 최소화를 위하여, 대상 논리함수를 BDD로 모델링하는데 있어서 가능한한 BDD가 대칭적이고 작은 depth를 갖도록 하기 위하여 binate selection을 응용하여 대상 논리함수의 입력변수들을 ordering하고 Shannon expansion [8] 을 이용하여 BDD로 실현한다.

실현된 회로의 레벨은 BDD의 depth에 좌우되며 회로의 delay time은 회로의 레벨에 비례한다.

$$Delay = (l - 1) * D_p$$

단, l 은 회로의 레벨이고 D_p는 레벨당 delay time이다.

대상 논리함수의 BDD가 한쪽 노드로 치우치지 않고 대칭적으로 생성되면 단위 논리모듈을 부분적으로 사용하는 경우를 줄이고 모듈의 전체를 사용할 수 있다. 또한 그렇게 함으로써 BDD가 작은 depth를 갖게 되어서 대상 논리함수를 매핑하는데 있어서 사용되는 단위 논리모듈의 수를 줄이며 회로의 레벨을 작게 하여 실현된 회로의 delay time을 감소시킨다.

논리함수의 cube form에서 0, 1을 포함하는 변수를 binate 변수라하고 0 또는 1만을 포함하는 변수를 unate 변수라 정의한다. BDD를 구성하는 알고리듬 MAKE_BDD는 논리함수의 cube form에서 각 입력변수를 binate 변수와 unate 변수로 분리한후 most binate한 순서로 ordering하고 입력변수의 ordering결과에 따라 BDD를 구성한다.

예제 1)

f = abc + ac' d + ab + de 〈cube form of f〉
= c' (ad+ab+de) + c(ab+ab+de) 111--
= c' (a' (de)+a(b+d))+c(a '(de)+a(b)) 1-01-
= c' (a '(de)+a(b' (1+d)+b(1+d))) 11---
+ c(a '(de)+a(b)) ---11

예제 1에 MAKE_BDD를 적용한 경우에 f의 cube form에서 각 변수의 1과 0의 갯수를 검색하여 변수들을 binate 변수와 unate 변수로 분리한 후 f의 입력변수 5개중에서 most binate한 변수순으로 ordering하면 order는 c, a, b, d, e 의 순으로 되며 그러한 순으로 Shannon expansion을 반복적으로 수행하면 그림 7의 BDD를 구할 수 있다.

```
MAKE_BDD

  term = no_of_cubes
  pi = no_of_inputs
  Begin
    for(i = 1…pi) {
      for(j = 1…term) {
        count_1_and_0(j)
        check_binate_or_unate(i)
      }
      if(input = binate)
        binate [] 〈- input
      else
        unate [] 〈- input
    }
    for(i = 1…pi) {
      order [] 〈-sort_with_binate(binate [] )
      order [] 〈-sort_with_unate(unate [] )
    }
    cofactoring(order [] )

  End
```

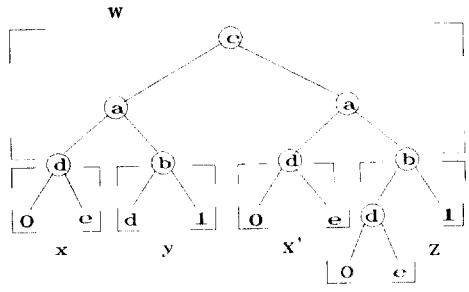


그림 7. 예제 1의 BDD
Fig. 7. A BDD of example 1.

3. Covering 과 Cell Merging

단위 논리모듈과 대상 논리함수로부터 BDD를 구한후 대상 논리함수의 BDD를 root 노드로부터 leaf 노드쪽으로 단위 논리모듈의 BDD로 covering한다. 그림 7의 BDD에서 박스로 둘러싸인 각 sub-tree들은 하나의 단위 논리모듈에 대응되는 것을 나타낸다. Covering 과정이 수행된 후 사용된 단위 논리모듈의 수를 줄이기위해 하나의 단위 모듈에 merging이 가능한 cell들의 쌍을 검색하여 cell merging 과정을 수행한다.

Definition 1. ***Cell*** 대상 논리함수의 BDD에서, 하나의 단위 논리모듈에 대응되는 각각의 sub-tree들을 cell이라 정의한다.

Definition 2. ***Leaf cell*** 임의의 cell의 모든 leaf 노드가 BDD의 leaf 노드와 대응되면 그러한 cell을 leaf cell이라 정의한다.

N개의 입력변수를 갖는 함수의 BDD는 최대 2^{N-1} 개의 leaf 노드를 포함하므로 입력변수의 증가에 대하여 leaf 노드는 지수적으로 증가한다. 이러한 leaf 노드의 수를 많이 줄일수록 논리함수를 실현하는데 사용되는 단위 논리모듈의 수를 감소시킬 수 있다.

Definition 3. ***Mergeable leaf cell*** leaf cell 중에서 depth d가 $d < 2$인 cell을 mergeable leaf cell이라 정의한다.

매핑과정에있어서 상당수의 Leaf cell들이 mergeable leaf cell이다. Mergeable leaf cell들이 각각 하나의 단위 논리모듈에 할당될 경우 단위 논리모듈의 3개의 multiplexer중에서 하나를 사용하지 못하기 때문에 단위 논리모듈의 효율적 이용이 불가능해지고 따라서 논리함수를 실현하는데 필요한 단위 논리모듈의 수를 크게 증가시킨다. 그림 7에서 cell x와 x'와 y는 mergeable leaf cell이고 단위 논리모듈내의 한개의 multiplexer를 사용하지 않는다. 이러한 mergeable leaf cell들을 merging할 수 있다면 cost를 상당히 감소시킬 수 있다. 대부분의 경우에 이러한 cell들을 merging할 수 있으며 따라서 회로의 크기를 감소시킬 수 있다.

Theorem 1. 논리함수의 BDD에서 임의의 두 sub-tree의 모든 leaf 노드가 BDD의 leaf 노드와 대응되고 두 sub-tree가 서로 동일(identical)하면 그 sub-tree들에 대응하는 두 함수값은 서로 동일하다.

증명 : 두 함수 $F(x_1, x_2, \cdots, x_n)$과 $G(y_1, y_2, \cdots, y_n)$에 대하여 반복적으로 Shannon expansion을 수행하여 BDD를 구성하면 각각의 함수값은 다음과 같다.

$F = x_1, x_2, \cdots, x_n F_{x1, x2, \cdots, xn} + x_1, x_2', \cdots, x_n F_{x1', x2', \cdots, xn'}$

$G = y_1, y_2, \cdots, y_n G_{y1, y2, \cdots, yn} + y_1, y_2', \cdots, y_n G_{y1', y2', \cdots, yn'}$

위의 두 식에서 각각의 xi와 yi $(1 \le i \le n)$는 BDD의 노드와 대응되며 $F_{x1, x2, \cdots, xn}$ 과 $F_{x1', x2', \cdots, xn'}$ 그리고 $G_{y1, y2, \cdots, yn}$ 과 $G_{y1', y2', \cdots, yn'}$ 는 BDD의 leaf 노드에 대응된다. 가정에서 F와 G의 BDD가 leaf 노드까지 동일하면 $x_i = y_i$, $F_{x1, x2, \cdots, xn} = G_{y1, y2, \cdots, yn}$ 그리고 $F_{x1', x2', \cdots, xn'} = G_{y1', y2', \cdots, yn'}$ 를 만족한다. 따라서 두 함수 F와 G는 동일하다.

Lemma 1. 동일한(identical) leaf cell들은 theorem 1에 의해서 동일한 출력함수에 존재하는 경우와 서로 다른 함수에 존재하는 경우에 모두 하나의 단위 논리모듈에 매핑할 수 있다.

그림 8는 그림 7의 BDD를 단위 논리모듈로 매핑한 결과이며 그림 7의 박스로 둘러싸인 각각의 sub-tree들이 하나의 단위 논리모듈에 실현된 것을 볼 수 있다. x로 표시된 단위 논리모듈은 동일한 세개의 cell들이 하나의 단위 논리모듈로 merging된 것이다.

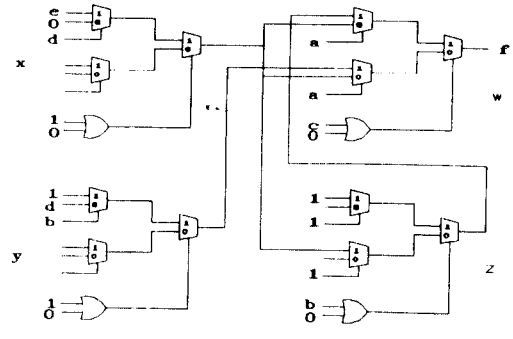


그림 8. 예제 1의 매핑 결과
Fig. 8. The result of example 1.

그림 8에서 cell x와 y는 하나의 multiplexer를 사용하지 않고 있음을 알 수 있다. 서로 다른 mergeable leaf cell들은 단위 논리모듈의 MUX 1 (그림 5)의 selection 입력을 고려하여 하나의 단위 논리모듈에 merging할 수 있다.

그림 9의 예제 2에서 f1, f2의 leaf cell들은 모두 mergeable leaf cell이고 k는 3개의 동일한 leaf cell을, l은 2개의 동일한 leaf cell을 나타낸다. cell k와 l은 lemma 1에 따라 각각 하나의 단위 논리모듈에 합할 수 있다. 하지만 cell k와 l을 합한 각각의 cell들은 아직도 mergeable leaf cell이다. 이러한 cell들의 쌍을 하나의 단위 논리모듈에 merging할 수 있는데 이 것은 2 입력 또는 3 입력 함수로 이루어지는 cell들의 쌍을 하나의 cell로 merging할 수 있음을 의미한다.

Lemma 2. Depth d가 d 〈 2인 두개의 서로다른 mergeable leaf cell들은 다음과 같은 경우에 하나의 단위 논리모듈에 merging할 수 있다

경우 Ⅰ : 두개의 mergeable leaf cell 모두가 하나이상의 동일한 leaf cell들을 갖지 않는 경우에 대상 논리함수에 대한 BDD에서 두 mergeable laef cell들을 구분할 수 있는 상위 노드가 존재하면, 그러한 노드의 변수를 merging되는 단위 논리모듈의 MUX 1의 selection 입력으로 함으로써 두개의 mergeable leaf cell들을 하나의 단위 논리모듈에 merging할 수 있다.

예제 2에서 cell k는 노드 e와 d의 좌측 child이다. 그리고 cell m은 e의 좌측 child이고 d의 우측 child이다. 따라서 노드 d는 두 cell k와 m을 구분할 수 있는 상위 노드이다.

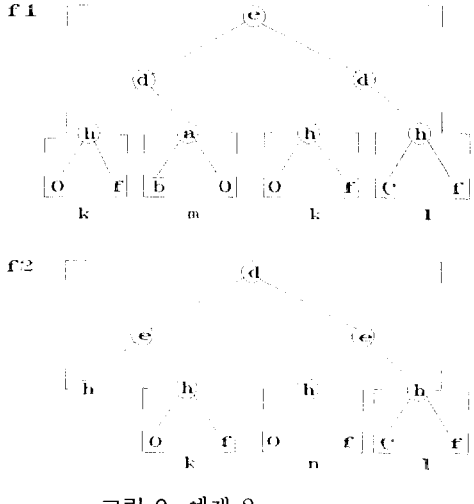


그림 9. 예제 2
Fig. 9. Example 2.

경우 Ⅱ : 두개의 mergeable leaf cell중 하나이상의 동일한 leaf cell들을 갖는 경우에 각각의 동일한 mergeable cell의 group이 공통으로 가지는 상위 노드들이 존재하고 그러한 노드들중에 두 group을 구분할 수 있는 상위 노드가 존재하면 그러한 노드를 단위 논리모듈의 MUX 1의 selection 입력으로함으로써 두 group을 하나의 단위 논리모듈에 merging할 수 있다.

예제 2에서 f1과 f2의 모든 cell k는 노드 d의 좌측 child이므로 cell k의 공통 상위노드는 d이다. 그리고 f1과 f2의 모든 cell l은 노드 d와 e의 우측 child이므로 d와 e가 l의 공통 상위노드이다. 따라서 각각의 cell group의 공통 노드중에서 두 group을 구분할 수 있는 상위노드는 d이다.

예제 2에서 f1과 f2의 2개의 동일한 cell l들은 모두 노드 e와 d의 우측 child들이므로 공통의 상위노드는 e와 d가 되며, cell m은 다른 동일한 cell들을 갖지않고 노드 e의 좌측 child이며 노드 d의 우측 child이다. 따라서 cell l과 m을 구분할 수 있는 상위 노드는 e가 되며 입력 변수 e를 단위 논리모듈내부의 MUX 1의 selection 입력으로 하면 cell l과 m들을 하나의 모듈에 합할 수 있다. cell k와 n도 마찬가지로 입력변수 d를 MUX 1의 selection 입력으로 하여 합할 수 있다. 그림 10은 그림 9의 예제 2에 대한 결과이다.

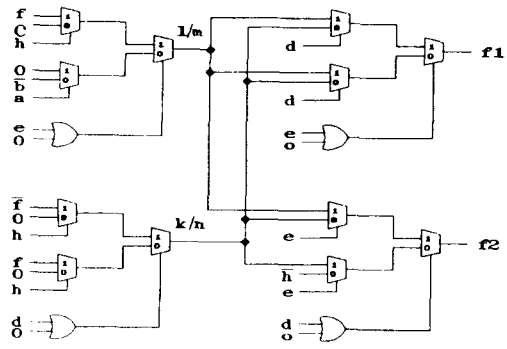


그림 10. 예제 2의 매핑결과
Fig. 10. The result of example 2.

그림 10의 예제 2에 대한 매핑결과에서 나타낸 바와같이 l과 m을 합한 단위 모듈을 l의 출력을 입력으로 사용하는 모듈과 m의 출력을 입력으로 사용하는 모듈에 모두 연결하면 위에서 언급한 selection 입력 e값에 따라 m 값이 필요한 모듈엔 m 값만이 그리고 l 값이 필요한 모듈엔 l 값만이 입력된다. Lemma 1

과 2를 만족하는 cell merging과정을 통하여 대상 논리함수를 매핑하는데 있어서 사용되는 단위 논리모듈의 수를 상당한 정도로 감소시킨다. 다음의 알고리듬 CELL MERGING은 앞에서 기술한 cell merging과정을 나타내고 있다.

```
CELL MERGING
  pi = no_of_inputs
  t_leaf = no_of_total_non-full_leafs
  Begin
    candidate [] <- leafs
    no_of_candidates <- t_leaf
    for(i = 1…t_leaf) {
      for(j = 1…t_leaf) {
        if(check_identity(i, j) = true) {
          delete_candidate(j)
          reduce_no_of_candidates(j)
          check_same_select_input(i, j)
        }
    }
    for(i = 1…no_of_candidates) {
      for(j = 1…no_of_candidates) {
      if(check_different_selectinput(i, j)
        = true) {combine_two_cells
          (i, j, selectinput)}
      }
    }
  End
```

Ⅳ. 실험결과

본 논문에서 제안한 알고리듬을 C 언어로 실현하여 대표적인 benchmark 회로에대하여 실험하였다. 표 1의 misII [4], mis-pga [6], proserpine [3], new mis-pga [9] 등은 관련 논문의 실험결과를 참조한 것인데, 각 회로에 대하여 사용되는 단위 논리모듈의 수가 mis-pga [6] 나 proserpine [3] 등에 비하여 20 - 50 % 정도로 감소된 결과를 나타낸다.

표 1. Benchmark에 대한 비교 결과

Name	misII[4]	mis-pga[6]	proserpine[3]	new mis-pga[9]	HMAP
f51m	52	48	63	39	37
5xp1	51	45	53	35	33
bw	81	65	67	54	53
rd84	62	61	70	36	36
misex1	22	20	25	17	15
rd73	32	31	-	25	23
adr4	-	-	-	-	31
cm163a	-	-	-	-	17

Ⅴ. 결 론

본 논문에서는 BDD와 multiplexer-based PGA의 특성과 유사한 점을 고려하여 단위 논리모듈의 BDD를 간략화하여 매핑의 효율성을 높였다. 그리고 실현하고자하는 논리함수의 BDD를 구성할때 binate selection을 이용하여 BDD의 노드가 한쪽으로 치우치지 않도록 대칭적으로 구성하는 방법을 제안하여 사용되는 단위 논리모듈의 수와 실현된 논리회로의 레벨을 최소화하였다. 기존의 연구들은 대상 함수나 단위 논리모듈의 입력변수에 대한 ordering과 graph pattern matching등에 치중하여 FPGA의 단위 논리모듈을 효율적으로 사용하는 것과 FPGA 회로의 레벨의 감소 즉 회로의 delay time의 감소에 대한 연구가 부족하였다. 본 논문에서는 실제상황에서 많이 발생할 수 있는 서로 다른 mergeable leaf cell들을 하나의 단위 논리모듈에 merging함으로써 회로의 실현에 필요한 단위 논리모듈의 수와 회로의 레벨을 최소화하였다. 결과적으로, 사용되는 논리 모듈의 수와 실제 회로의 level을 최소화함으로써 논리회로의 면적과 동작시간을 상당히 감소시키는 성과를 얻었다.

앞으로의 연구과제는 대상 논리함수를 BDD로 실현할 때 multiple ouput 함수를 고려하여 입력변수를 ordering하는 문제와, 대상 논리함수를 더욱 효과적으로 모델링할 수 있는 매개체를 개발하여 효율적인 테크놀로지 매핑 알고리듬을 개발하는 것이다.

參 考 文 獻

[1] K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG matching", 24th DAC, pp.341-347,1987

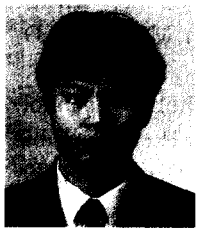
[2] E. Detjens, G. Gannot, R. Rudell, A. Sangiovanni-Vincentelli, A. Wang, "Technology Mapping in MIS",ICCAD, pp.116-119, 1987

[3] S. Ercolani, G. De Micheli, "Technology Mapping for Electrically Programmable Gate Arrays", 28th DAC,pp.234-239, 1991

[4] R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli,and A.R. Wang, "MIS: A Multiple-Level Logic Optimization System ", *IEEE Transactions on*

CAD, pp.1062-1081, 1987
[5] F.Mailhot and G. De Micheli, "Technology Mapping with Boolean Matching", European Design Automation Conference, pp.212-216, 1990
[6] R. Murgai, Y. Nishizaki, N. Shenoy, R.K. Brayton, and A. Sangiovanni-Vicentelli, "Logic Synthesis for Programmable Gate Arrays", 27th DAC, pp.620-625, 1990.
[7] S. B. Akers, "Binary Decision Diagrams", *IEEE transaction* on Computer, pp.509-516, 1978
[8] R.K. Brayton, G.D. Hachtel, C.T. McMullen, A.L.Sangiovanni-Vincentelli, "Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984
[9] R. Murgai, R.K. Brayton, A. Sangiovanni-Vicentelli, " An Improved Synthesis Algorithm for Multiplexer-based FPGA's", 29th DAC, pp.380-386, 1992
[10] S. D. Brown, R. J. Francis, J. Rose and Z. G. Vranesic, "Filed-Programmable Gate Arrays", Kluwer Academic Publishers, 1992
[11] R. J. Francis, J. Rose and Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-based FPGAs", Proc. 28th DAC, pp. 227-233, June, 1991
[12] M. R. Garey, D. S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", W. H. Freeman and Co., Newyork, 1979
[13] K. Bartlett, G. Hachel, "Library Specific Optimization of Multilevel Combinational Logic", Proc. of the IEEE International Conference on Computer Design, October, 1985
[14] E. Detjens et. al, "Technology Mapping in MIS", Proc. ICCAD, pp. 116-119, Nov. 1987
[15] R. E. Bryant, "Graph-Based Algorithm for Boolean Function Manipulation", *IEEE Trans.* on Computers, vol. C-35, no. 8 Aug. 1986
[16] R. A. Bergamaschi, "Automatic Synthesis and Technology Mapping of Combinational Logic", Proc. ICCAD, pp. 466-469, Nov. 1988
[17] A. El Gamal, et. al, "An Architecture for Electrically Configurable Gate Arrays", *IEEE JSSC* vol. 24, no. 2, pp. 394-398, April, 1989
[18] N. Calazans, Q. Zhang, R. Jacobi, B. Yernaux and Anne-Marie Trullemans, "Advanced Ordering and Manipulation Techniques for Binary Decision Diagram", EDAC, pp. 452-457, 1992

著 者 紹 介



姜 圭 顯(正會員)
1966年 8月 24日生. 1991年 한양대 전자공학과 졸업. 1993年 한양대 대학원 전자공학과 석사 학위 취득. 현재 한국통신 전임 연구원 재직. 주관심분야는 CAD시스템 및 FPGA 회로 설계 등임.

李 載 興(正會員) 第 28卷 A編 第 10號 參照
현재 대전공업대학교 전자계산학과 교수

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