論文 89-26-4-18

VLSI 테스트 이론을 이용한 Global Redundancy 조사

(Global Redundancy Check by VLSI Test Theory)

李 省 奉*, 鄭 正 和*

(Seong Bong Lee and Jong Wha Chong)

要 約

본 논문에서는 게이트레벨회로 최적화를 위한, 논리적 redundancy를 제거하는 새로운 방법을 제안한다.

본 방법은 회로내의 모든 신호선에 대한 redundancy 조사를 피하여 일부의 신호선-fanout branch 신호선에 한정하여 조사를 행한다. 또 조사한 신호선이 nonredundant 할 경우에는, 그 신호선에 대한 조사 과정에 생성된 정보만을 이용하여, 다른 nonredundant한 신호선을 유도하는 효율적인 procedure을 사용한다. 그리고, 한 신호선에 대한 redundancy 재조사를 피하기 위해, 신호선의 조사순서를 결정하는 휴리스틱한 방법을 제안한다.

본 방법은 기존의 테스팅이론을 응용한 휴리스틱한 방법으로, 각 신호선에 대한 redundancy 재조사를 행하지 않기 때문에 기존의 방법에 비해 실행시간이 매우 빠르다.

Abstract

In this paper, a new method is proposed to remove the logical redundancy for the gate-level circuit optimization.

In this method, only the fanout-branch signals in the circuits, not all the signals, are examined for redundancy. When a signal is determined to be nonredundant, other nonredundant signals are found out by the efficient procedure, using only the informations which are generated in the course of the redundancy-check. In order to avoid the re-examination of a signal for redundancy, a heuristic method is proposed to determine the redundancy-checking order of signals.

The proposed method is heuristic, based on the VLSI test theory. It is much faster than other methods, since it does not reexamine a signal for redundancy.

I. 서 론

최근 VLSI 집적도의 증가에 따른 설계시간의 과다한 증가를 줄이기 위해 설계자동화에 관한 연구가 활발히 진행되고 있다. 그러나 하드웨어 컴파일러와 같은 설계자동화 시스템에 의해 설계된 회로에는, 불필요한 신호선이나 게이트들(이하 redundancy)이 많이 존재한다. 이러한 redundancy의 존재는 최종 칩면적의 증대를 초래할 뿐만 아니라, 설계된 회로의 테스팅에 많은 어려움을 주게 된다.[1] 따라서, 이를 효율적으로 제거하는 설계최적화 시스템에 대한 연구가 크게 요구되고 있다.

*正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1989年 2月 3日

Multi-level 회로에 대한 최적화방법으로는, 'local transformation'[2,3]이 제안되었으나, 이 방법으로 제거할 수 있는 redundancy는 크게 제안되어 있고, 특히 회로의 don't-care 조건를 고려할 수 없다는 문제점이 있다.[4] 최근 이를 고려한 최적화방법[1]이 제안되었으나, 이 방법은 회로내의 한 신호선에 대한 redundancy 여부를 조사하는 문제에 국한되어 있다.

본 논문에서는 게이트레벨 회로내에서 redundant 한 신호선을 찾아내고, 이를 제거하는 새로운 방법을 제안한다. 본 방법은 기존의 VLSI 테스트이론을 이용한 휴리스틱한 방법으로서, 각 신호선에 대한 redundancy 재 조사를 행하지 않기 때문에 기존의 방법에 비해, 실행시간이 매우 빠르다.

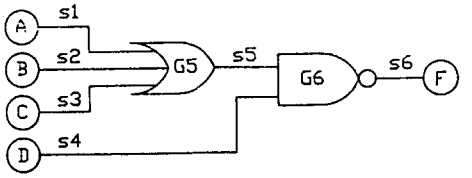
Ⅱ. Redundancy와 테스팅

1. Redundancy의 정의

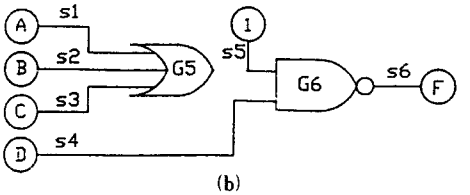
본 논문에서 한 신호선이 redundant 하다는 것을 다음과 같이 정의한다.

〔정의 1〕 어떤 신호선을 0(or 1)의 상수로 변경할 경우, 주출력(primary output) 신호선의 논리값이 care조건의 모든 입력에 대해 변경이전과 동일한 값을 갖는다면, 그 신호선은 0-redundant (or 1-redundant) 하다고 한다.

〔정의 2〕 한 신호선이 0-redundant 또는 1-redundant하면, 그 신호선은 redundant하다고 하고, 그렇지 않으면 nonredundant하다고 한다. 또 한 신호선이 0(or 1)-redundant하지 않으면, non-0(or 1)-redundant하다고 한다.



(a)



(b)

Don't-Care
조건 : (A B)

그림 1. Redundant 한 신호선
Fig. 1. A redundant signal.

예를들어, 그림 1(a)와 같은 회로와 don't-care 조건이 주어졌을 때, 그림 1(b)에서 처럼 신호선 s5를 상수 1로 대치하여도, 회로의 출력은 care 조건의 입력에 대해 전혀 변화가 없다. 이때 신호선 s5는 1-redundant 간단히 redundant하다고 한다.

기존의 설계자동화 시스템에 의해 설계된 회로내에는 redundant 신호선이나 게이트들이 많이 존재한다. 어떤 게이트가 redundant하다는 것은 그 게이트의 출력신호선이 redundant하다는 것과 동일하다(그림 1(b)) 따라서, 회로내의 불필요한 게이트나 신호선을 찾는 것은 어떤 신호선이 redundant한 가를 조사하는 것과 같다. 이러한 redundant한 신호선을 제거하면, 회로의 최종 칩면적을 줄이는것 외에도, 그 회로의 테스팅을 쉽게하는 장점이 있다.

2. Redundancy 조사와 테스팅

Redundancy 조사 알고리듬의 설명을 위해, 다음과 같은 용어를 정의한다.

〔정의 3〕 한 게이트 G의 입력신호선 s에 의해 이 게이트의 출력값이 결정되면, 이 입력신호선 s는 다른 입력신호선을 'block'한다고 한다.

〔정의 4〕 두 게이트 또는 신호선 사이에 경로(path)가 존재하고, 이 경로에 속하는 모든 신호선이 다른 신호선에 의해 block되지 않으면, 이 경로를 'non-blocked 경로'라 한다.

예로서, AND 게이트의 한 신호선이 0이면, 이 신호선에 의해 게이트의 출력값이 0으로 결정된다. 이때 이 신호선은 다른 입력신호선을 block한다고 한다. 기존의 알고리듬[1]은 이 non-blocked 경로를 이용한 것으로, 이를 간단히 기술하면 그림 2와 같다.

예로서, 그림 1의 신호선 s5가 1-redundant함을 조

단계 1 : 조사할 신호선이 1(0)-redundant하지 않다고 가정하여, 이 신호선을 0(1)로 만드는 주입력(primary input)값을 구한다.

단계 2 : 조사하는 신호선에서부터 주출력(primary output)까지의 non-blocked 경로를 생성하는 주입력값을 구한다.

단계 3 : 이 주입력값이 care 조건에 속하면, 이 신호선은 non-1(0)-redundant하다고 판정하고, 만약 don't-care 조건에 속하면 단계 1, 2를 반복하여 새로운 주입력값을 구한다. 만약 이러한 주입력값이 존재하지 않을 경우에는 이 신호선은 1(0)-redundant하다고 판정한다.

그림 2 한 신호선에 대한 redundancy 조사방법
Fig. 2. Redundancy checking method for a signal.

사하면, "단계 1에서 신호선 s5를 0으로 만드는 주입력 A=B=C=0를 구할 수 있다. 단계 2에서 D=1. 만약 D=0이면 신호선 s4가 신호선 s5를 block하게 되어, non-block 경로를 생성할 수 없다. 단계 3에서 주입력값(A=B=C=0, D=1)이 don't-care 조건에 포함되므로, 단계 1과 2를 반복하여 다른 주입력값을 구하게 된다. 그러나, 다른 주입력값이 존재하지 않기 때문에 신호선 s5는 1-redundant하다고 판정한다.

그런데, 이 예에서 주입력값(A=B=C=0, D=1)에 대한 non-blocked 경로(s5-G5-s6-F)에 속한 모든 신호선은 조사신호선 s5를 상수1로 대치한 경우와 그렇지 않은 경우에 대해 서로 반대값을 갖는다(대치한 경우 : s5=1, s6=0, 대치이전 : s5=0, s6=1).

〔정의 5〕 그림 1의 신호선 s5처럼, 그림 2의 단계 2에서 구한 주입력값에 대해, 조사신호선을 상수로 대치하였을때는 1의 값을, 그렇지 않은 경우에는 0 값을 갖는 경우, 이 신호선은 DB값을 갖는다고 하고, s6처럼 그 반대인 경우 DV의 값을 갖는다고 한다(D알고리듬[5]에서 DB=D, DV=D).

정의 5에 의하면, non-blocked 경로에 속하는 모든 신호선은 DB 또는 DV를 갖는다는 것을 알 수 있다. 즉 redundancy 조사에서의 non-blocked 경로를 구하는 것은 테스팅에서의 'sensitized 경로'를 구하는 것과 같다. 따라서, 기존의 방법[1]은 테스트패턴생성 알고리듬[5]과 매우 유사하다. 차이점은 테스팅에서는 don't-care 조건을 고려하지 않는다는 것이다. 만약 don't-care 조건이 없을 경우, 어떤 신호선이 1(0)-redundant하다는 것은, 테스팅에서 그 신호선의 stuck-at-1(0) fault에 대한 테스트패턴이 존재하지 않는다는 것과 동일하다. 따라서 기존의 테스트패턴생성 알고리듬을 이용하여 테스트패턴의 존재여부로서, redundancy를 조사할 수 있다. 한편, don't-care 조건이 존재할 경우에도 그림 3(a)의 형태로 입력회로를 변경하면, don't-care 조건이 없는 경우와 마찬가지로 기존의 테스트생성 알고리듬으로 redundancy를 조사할 수 있다. 이렇게 회로를 변경하면, don't-care 조건의 주입력에 대해서는 신호선 s1이 1이 되어, 신호선 s2를 block하기 때문에, 새로운 주출력 F까지의 non-blocked 경로가 생성될 수 없기 때문이다. 그림 3(b)는 그림 1(a)의 회로와 don't-care 조건을 그림 3(a) 형태로 변경한 예를 보인 것이다.

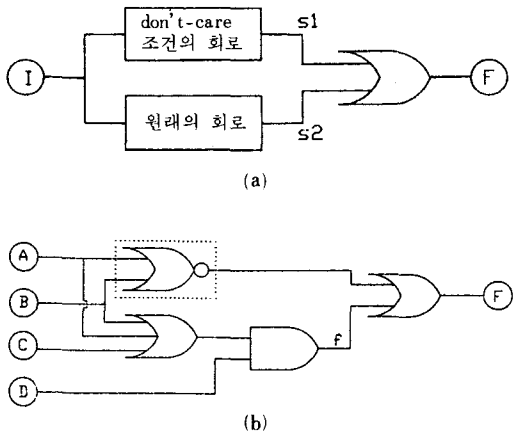


그림 3. Don't-care 조건을 고려한 회로
Fig. 3. Circuits considering don't-care conditions.

3. 기존 방법과 주요 문제점

기존의 방법[1]은 한 신호선에 대한 redundancy를 조사하는 방법이다. 이를 이용하여, 전체 회로내의 모든 redundancy를 제거하는 방법은 다음과 같은 procedure로 나타낼 수 있다.

```
회로최적화( )
{
  모든 신호선 unmark;
    DO {
      unmarked 신호선 선택; /*과정1*/
      선택된 신호선에 대한 redundancy 조사; /*과정2*/
      IF (redundant ?) {
        신호선제거;
        모든 신호선 unmark; /*과정3*/
      } ELSE
        신호선 mark; /*과정4*/
  } UNTIL (모든 신호선 marked);
```

그림 4. 일반적인 redundancy 조사과정
Fig. 4. General procedure of the redundancy checking.

이 procedure에서 mark된 신호선은 non-redundant하다는 것을 나타낸다. 한 신호선에 대한 redundancy 조사는 기존의 방법(그림 2), 또는 기존의 테스트패턴생성 알고리듬[5,8]을 사용할 수 있다. 그런데, 과정 3은 redundant한 신호선을 제거하는 경우, 이미 조사한 신호선을 재조사하기 위해 사용된다. 이 과정이 필요한 이유는 한 신호선이 redundant 하다고 판정되면, 이 신호선의 제거에 따라 이미 nonredundant하다고 판정된 신호선이 redundant하게 되는 경우가 발생하기 때문이다. 그림 5는 이러한 경우의 예를 보인 것이다. 신호선 s1이 이미 nonredundant하

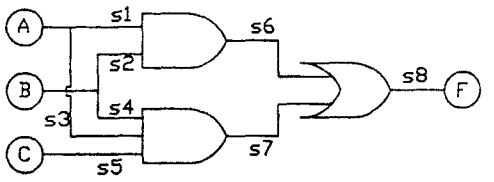


그림 5. Redundant 한 신호선의 영향
Fig. 5. Effect of a redundant signal.

다고 판정되었다고 하자(입력패턴 : A=B=1, C=0). 그러나 신호선 s5가 1-redundant하여 제거되면, 이 신호선은 0-redundant하게 된다.

그런데, 한 신호선의 redundancy 여부를 조사하는 과정 2는 회로의 모든 경로(또는 care 조건의 모든 경우)를 조사해야 하기 때문에 많은 계산시간이 필요하다. 따라서, 전체 최적화 실행시간을 감소시키기 위해서는 redundancy 여부를 조사하는 신호선의 수를 최소로 하는 것이 매우 중요하다. 기존의 방법[1]은 이를 위해 별도의 fault 시뮬레이션[6]의 사용을 제안하고 있다. 그러나, 이 fault 시뮬레이션 또한 많은 계산시간이 소요되며[7], 특히 redundant 한 신호선이 발생하여, 이를 제거하는 경우에는, 전체 회로를 재시뮬레이션해야 하는 문제점이 있다.

Ⅲ. 본 논문의 redundancy 조사방법

본 논문에서 제안하는 redundancy 조사방법은 조사할 신호선의 수를 줄이기 위해 다음의 사항을 고려하였다.

1) 회로내의 모든 신호선에 대한 조사를 피하여, 일부의 신호선에 한정하여 redundant 여부를 조사한다(그림 4의 과정 1을 변경).
2) 조사한 신호선이 nonredundant하다고 판정되는 경우에는, 조사과정에서 생성된 중간결과를 이용하여, fault 시뮬레이션 없이 다른 nonredundant한 신호선들을 유도한다(그림4의 과정 4를 변경).
3) 한 신호선에 대한 redundant 여부의 재조사를 피한다.(그림 4의 과정 3을 생략)

본 방법은 한 신호선에 대해 최대 1회의 redundancy 여부를 조사하기 때문에 기존의 방법[1]에 비해 최적화시간이 매우 짧다. 그러나, 기존의 방법[1]과 마찬가지로 회로내의 모든 redundancy를 제거할 수 없다.

1. Fanout branch 신호선

회로내의 모든 redundancy를 제거하기 위해서, 반드시 모든 신호선에 대해 redundant 여부를 조사해야 하는 것은 아니다. 주입력신호선과 fanout branch 신호선들만을 조사하는 것으로 충분하다. 여기서 fanout branch 신호선이란 그림 6의 신호선 s2, s3처럼 한 게이트의 fanout이 다수인 경우, 그 게이트의 출력신호선 각각을 fanout branch 신호선이라고 한다.[7] 또, 신호선 s1을 fanout stem 신호선이라고 한다.

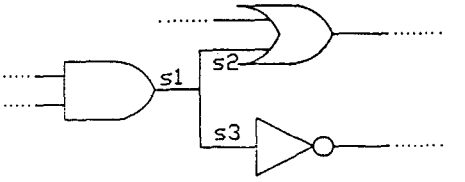


그림 6. Fanout branch 신호선
Fig. 6. Fanout branch signals.

이는 다음의 정리에 의해 증명할 수 있다.

〔정리 1〕 다출력회로에 있어서, 모든 주입력신호선과 fanout branch 신호선이 nonredundant하면, 회로내의 모든 신호선이 nonredundant하다.

이 정리는 기존의 테스팅문제에 대한 정리[7]를 본 redundancy 조사에 맞게 변경한 것이다. 실제로 redundant할 가능성이 가장 큰 신호선은 fanout branch 신호선이다.[4] 본 논문에서는 주입력신호선(fanout stem 신호선인 경우)과, fanout branch 신호선이면서 주출력신호선인 신호선에 대한 redundancy는 조사하지 않는다. 이는 대부분의 실제 회로에서 이러한 신호선이 redundant한 경우가 거의 없기 때문이다. 이 방법은 초기에 조사할 신호선의 수를 감소시킨다.

2. Nonredundant 신호선의 유도

만약 조사한 신호선이 nonredundant하다고 판정된 경우, 이 조사과정에서 생성된 정보를 이용하여, 다른 nonredundant한 신호선들을 찾아낼 수 있다. Nonrednundant한 신호선을 조사한 경우에는 이 신호선이 nonredundant하다는 것을 입증하는 주입력값(테스팅에서의 테스트패턴)이 존재한다. 이 주입력값을 이용하여 fault시뮬레이션이나 TESTDETECT[8] 등을 실행하면, 이 주입력값에 의해 nonredundant함을 입증할 수 있는 신호선들을 유도해 낼 수 있다. 그러나, 이 방법은 많은 계산시간과 메모리을 소요한다는 문제점이 있다.

본 논문에서의 nonredundant 신호선의 유도과정은 redundancy 조사과정에서 생성된 정보를 직접 이용하기 때문에 매우 빠른 시간에 nonredundant한 신호선들을 유도해 낼 수 있다. 본 논문에서 제안하는 nonredundant한 신호선을 유도하는 procedure는 다음과 같다.

```
Nonredundant_신호선유도( )
{
  FOR (non-blocked 경로에 속하는 게이트 G에 대해) {
    /*주출력신호선부터 조사신호선까지 역으로 조사*/
    IF (게이트 G의 입력신호선중에서 DV/DB값을 갖는
      신호선이 2이상)/*단경로가 아닐때*/
      RETURN;
    신호선 S=게이트 G의 입력신호선중 DV/DB값을 갖는 신호선;
    IF (게이트 G가 AND 또는 NAND) {
      IF (신호선 S의 값==DV)
        게이트 G의 모든 입력신호선은 non-0-redundant;
      ELSE/* 신호선 S의 값==DB*/
        신호선 S는 non-1-redundant;
    } ELSE {/*게이트 G는 OR, NOR*/
      IF (신호선 S의 값==DB)
        게이트 G의 모든 입력신호선은 non-1-redundant;
      ELSE/*신호선 S의 값==DV*/
        신호선 S는 non-0-redundant;
    }
  }
}
```

그림 7. Nonredundant 한 신호선의 유도
Fig. 7. Derivation of nonredundant signals.

이 procedure에서 게이트 G가 NOT이면 NAND/NOR에서 입력신호선이 하나인 것으로 하여 처리한다. 이 방법은 다음 정리를 이용한 것이다.

〔정리 2〕 Redundancy 조사에서 구한 주입력값에 대해, non-blocked 경로가 단경로(single path)일 때, 이 경로에 속하는 모든 신호선들은, 그 신호선의 값이 DV이면 non-0-redundant하고, DB이면 non-1-redundant하다.

이 정리는 테스팅에서의 'single-sensitized 경로'에 대한 정리를 redundancy 조사에 적용한 것이다.[5] 예를 들어 그림 8에서 신호선 s1가 0-redundant한지를 조사할 경우, 이 신호선이 nonredundant함을 나타내는 주입력값(A=C=1, B=0)이 존재한다. 이때, 신호선 s1로 부터, 주출력 F까지의 단경로(s1—G3—s8—G5—s10—F)에 속하는 모든 신호선은 모두 DV값을 가지므로, 모두 non-0-redundant하다. 이와 더불어, 신호선 s5, s6 역시 non-0-redundant하다. 따라서, 한 신호선에 대한 redundancy 조사로 부터 많은 nonredundant한 신호선을 유도할 수 있음을 알 수 있다. 이에 따라 redundant 여부를 조사할 신호선을 감소시킨다는 것을 알 수 있다.

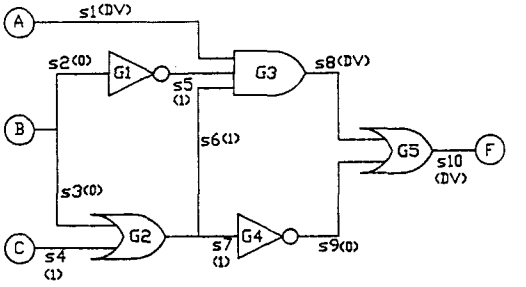


그림 8. 그림 7의 예제회로
Fig. 8. An example circuits of fig. 7.

3. Redundancy 조사순서 결정

전체 회로내에서 각 신호선에 대한 redundancy 조사순서를 최적으로 결정하는 문제는 매우 어렵다. 특히 한 신호선에 대한 재조사를 피하기 위해서는, redundant한 신호선을 먼저 조사하는 것이 매우 중요하다. 그러나, 어떠한 신호선이 redundant한지는 실제로 조사하기 이전에는 알 수 없기 때문에 휴리스틱하게 정할 수 밖에 없다.

본 논문에서의 redundancy 조사순서는 다음과 같이 휴리스틱하게 결정한다. 먼저 각 신호선에 대한 주입력과 주출력으로부터의 거리를 계산한다. 이 거리를 이용하여 fanout branch 신호선들의 조사순서를 정한다. 그 순서는 주입력으로부터 가장 가까운 신호선을 먼저 선택한다. 만약 같을 경우에는 주출력으로 부터 가장 먼 신호선부터 먼저 선택한다. 이렇게 순서를 정하는 것은 이러한 신호선들이 redundant할 가능성이 크기 때문이다. 그리고, 이러한 신호선이 nonredundant할 경우에는 많은 nonredundant 한 신호선들을 유도할 수 있어, 조사할 신호선의 수를 감소시키기 때문이다. 또한 주출력으로부터의 거리가 먼 신호선이 redundant한 경우에는 이 신호선의 제거됨에 따라, 전체 회로의 지연시간을 줄일 수 있다는 장점도 있다.

Ⅳ. 실험 및 고찰

본 논문에서 제안하는 방법의 유효성을 입증하기 위해 여러 예제회로에 대하여 프로그램 실행을 행하였다. 본 방법은 IBM PS/2 MODEL80(OS: XENIX)상에서 C언어로 실현하였다. 한 신호선에 대한 redundancy 조사(그림 4의 과정 2)는 기존의 테스트 패턴생성 알고리듬[8]을 변형하여 사용하였다. 이 알

고리듬은 기존의 방법[1]과는 달리 조사하는 신호선이 redundant하면, 반드시 이를 찾아내는 장점이 있다.

표 1. 실험결과
Table 1. Experiment results.

회 로	게이트 (신호선)	경우 1		경우 2	
		제거신호선	실행시간	제거신호선	실행시간
ADR	95 (158)	1	36.2	1	7.6
ALU4	107 (204)	16	56.4	16	7.0
CNT8	129 (221)	5	23.9	5	7.4
CCT	201 (305)	26	723.3	22	146.5
CNT16	249 (649)	52	2198.9	52	52.3

실행시간 sec.

표 1은 실험결과를 보인 것이다. 표 1에서 경우 2는 본 논문에서 제안한 방법을 사용한 경우이며, 경우 1은 사용하지 않은 경우로서 기존의 방법[1]과 유사하다.

본 논문의 방법은, 기존의 방법과 마찬가지로 회로내의 모든 redundancy를 제거할 수 없다. 이는 첫째, 주입력과 주출력신호선에 대한 redundancy 조사를 행하지 않기 때문에 이들 신호선이 redundant한 경우에는 이를 찾지 못하는 경우가 있다. 둘째, fan-out branch 신호선에 부여한 조사순서가 최적임을 보장할 수 없고, 세째, 본 논문에서 제안한 방법에 의해 유도해 낸 nonredundant한 신호선이 다른 redundant한 신호선이 제거됨에 따라 redundant하게 되는 경우가 발생하지 않는다는 것을 보장할 수 없기 때문이다. 그러나, 대부분의 회로에서 기존의 방법과 같은 정도의 최적화결과를 얻을 수 있으며, 실행시간은 기존의 방법보다 매우 빠르다.

V. 결 론

본 논문에서는 게이트 레벨회로 최적화를 위한, 논리적 redundancy를 제거하는 새로운 방법을 제안하였다.

본 방법은 회로내의 fanout branch 신호선에 한정하여 redundancy를 조사하며, 또 조사한 신호선이 nonredundant할 경우, 이에 파생되는 nonredundant한 신호선들을 찾아내는 효율적인 방법을 제안했다. 그리고, 한 신호선에 대한 redundancy 재조사를 방지하기 위해, 조사할 신호선의 순서를 결정하는 휴리스틱한 방법을 제안했다.

본 방법은 기존의 테스팅이론을 응용한 휴리스틱한 방법으로, 각 신호선에 대해 redundancy 재조사를 행하지 않기 때문에, 다른 방법에 비해 최적화실행시간이 매우 짧음을 보였다.

앞으로의 연구과제로는 신호선의 redundancy 조사순서에 대한 보다 자세한 연구가 요구된다.

參 考 文 獻

[1] Daniel Brand, "Redundancy and don't cares in logic synthesis," *Trans. on Comp.*, vol. C-32, no. 10, pp. 947-952, Oct. 1983.

[2] J.A. Darringer, et al., "LSS: A system for production logic synthesis," *IBM J. Res. Develop.*, vol. 28, no. 5, pp. 537-544, 1984.

[3] W.W. Cohen, et al., "A rule-based expert system for optimizing combinational logic," *IEEE Design & Test of Comp.*, pp. 22-32, Aug. 1985.

[4] L. Trevillyan, et al., "Global flow analysis in automatic logic design," *IEEE Trans. on Comp.*, vol. C-35, pp. 77-81, Jan. 1986.

[5] J.P. Roth, et al., "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans. on Electronic Computers*, vol. EC-16, no. 5, pp. 71-83, Oct. 1967.

[6] D.B. Armstrong, "A deductive method for simulating faults in logic circuits," *IEEE Trans. on Comp.*, vol. C-21, no. 5, pp. 464-471, 1972.

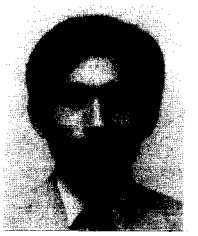
[7] J. Savir and J.P. Roth, "Testing for, and distinguishing between failures," 12th Int'l. Sym. on FTC., pp. 165-172, 1982.

[8] P. Goel, "An implicit enumeration algorithm to generate tests for combinational logic circuits," *IEEE Trans. on Comp.*, vol. C-32, pp. 215-222, March 1981. *

著 者 紹 介

李 省 奉(正會員)

1962年生. 1984年 한양대학교 전자공학과 졸업. 1986年 2月 한양대학교 대학원 전자공학과 졸업, 공학석사학위 취득. 1986年 3月～현재 한양대학교 대학원 전자공학과 박사과정 재학중. 주관심분야는 VLSI CAD 특히 논리설계 및 검증 등임.



鄭 正 和(正會員)

1950年生. 1975年 한양대학교 전자공학과 졸업. 1981年 3月 일본 와세다대학 박사학위 취득. 일본 NEC(주) 중앙연구소 연구원, KIET 위촉 연구원, University of California, Berkeley 교환교수. 1981年～현재 한양대학교 전자공학과 부교수. 주관심분야는 VLSI CAD 특히 Layout 및 HDL 등임.