論文 92-29A-4-13

검증 테스팅을 위한 새로운 설계방법

(A New Design Method for Verification Testability)

李 永 浩*, 鄭 正 和*

(Young Ho Lee and Jong Wha Chong)

要 約

본 논문에서는 검증 테스팅이 용이한 조합회로 설계를 위한 새로운 휴리스틱 방법을 제안한다. 검증 테스팅을 위한 조합회로의 설계는 크게 입력변수 축소, 입력 변수 분할, 출력 변수 분할 및 논리 최소화로 구성되어 있다. 이 중에서 입력 변수 분할과 출력 변수 분할을 위한 새로운 휴리스틱 알고리듬을 개발한 후 조합회로의 설계에 적용하여 제안된 방법의 유용성을 입증한다.

Abstract

In this paper, a new heuristic algorithm for designing combinational circuits suitable for verification testing is presented. The design method consists of argument reduction, input partitioning, output partitioning, and logic minimization. A new heuristic algorithm for input partitioning and output partitioning is developed and applied to designing combinational circuits to demonstrate its effectiveness.

I. 서 론

최근 논리 합성이 회로의 테스팅에 막대한 영향을 미친다는 사실이 지적되었으며[5,7] 이 사실을 토대로 테스팅 문제를 논리합성 과정 또는 논리합성 전에 해결하려는 경향이 있다[3-5] 이런 추세에 따른 설계 방법은 크게 순서회로 설계 방법과 조합회로 설계 방법으로 구분된다. Devadas는 FSM(finite state machine)으로 모델링 되는 회로에 적용할 수 있는 테스트 용이한 회로 설계방법[5]을 개발했으며 Krasniewski는 검증 테스팅 (verification testing)[10]이 용이한 조합회로 설계방법[7-9]을 개발했다.

Kransniewski에 의하여 개발된 방법은 입력 변수 축소(argument reduction), 입력 변수 분할 (input partitioning), 출력 변수 분할(output partitioning) 및 논리 최소화(logic minimization)로 구성되어 있다. 이 방법은 테스팅을 무시하고 논리 설계를 행한 후 회로를 변형하여 테스팅의 용이성을 증가시키던 기존의 논리합성 방법보다 테스팅 면에서 우수한 회로를 생성한다[7-9]

그러나 입력 변수 분할 과정에서 dependency space에 부합되는 모든 maximal input subset을 생성한 후, 이 maximal input subset으로 부터 dependency space에 부합하는 모든 minimum cardinality input partition을 구한다[7,9] 이 과정에서 많은 계산 시간이 소요되므로 논리함수의 입출력 수가 많은 대규모 조합회로의 설계에 이용할 수 없다[1] 또한, 함께 논리 최소화할 출력을 결정하는 출력 변수 분할 과정에서 입력변수 축소 후 생성된 정보를 효율적으로 이용하지 못함으로써 논리 최소화의 결과가 나빠진다.

본 논문에서는 입력변수 분할 과정에서의 계산 시간을 단축하기 위하여 휴리스틱에 근거한 새로운 입력 변수 분할 알고리듬을 개발하고, 입력변수 축소의

*正會員, 漢陽大學校 電子工學科
(Dept. of Elec. Eng., Hanyang Univ.)
接受日字 : 1991年 12月 14日

결과를 효율적으로 이용하기 위한 출력변수 분할 알고리듬을 개발한다.

이를 위하여 Ⅱ장에서는 전반적 구성, Ⅲ장에서는 입력변수 분할 알고리듬, Ⅳ장에서는 출력 변수 분할 알고리듬과 논리 최소화, Ⅴ장에서는 개발된 알고리듬을 조합회로의 설계에 적용한 실험결과를 기술하고 Ⅵ장에서 결론을 맺는다.

Ⅱ. 전반적 구성

본 논문에서 제안하는 새로운 설계 방법의 전체구성은 OVERALL_ALGORITHM()에서 알 수 있듯이 Krasniewski 방법의 전체 구성과 같다. 그러나, 입력 변수 분할을 위한 새로운 휴리스틱 알고리듬을 개발하여 대규모 조합회로의 설계에 이용할 수 있도록 하고, 출력 변수분할 단계에서 입력변수 축소의 결과를 효율적으로 이용하여 회로의 면적을 줄인다.

```
OVERALL_ALGORITHM( ) {
1. ARGUMENT_REDUCTION( )
2. INPUT_PARTITIONING( )
3. OUTPUT_PARTITIONING( )
4. LOGIC_MINIMIZATION( )
}
```

먼저 입력변수 축소 단계에서 기존의 입력변수 축소 알고리듬[6]을 이용한다. 이 단계에서는 n 개의 입력 변수와 m개의 출력 변수로 정의되는 논리함수의 각 출력 y_j를 구현하는데 필요한 입력 변수들의 집합들 중에서 원소의 수가 최소인 집합 즉, minimum cardinality subset을 구한다.[2] 최소의 입력 변수로 각 출력 y_j를 구현하는 방법이 다양하므로 입력 변수 축소 후 생성되는 입력 변수들의 집합은 여러개 존재할 수 있다.[3] 출력 y_j에 대하여 입력변수 축소 후 생성된 minimum cardinality subset을 dependency subset이라고 하고 DS^j_k로 나타낸다. DS^j_k의 집합을 dependency family라고 하고 DS^j로 나타낸다. DS^j의 집합을 dependency space라고 하고 DS로 나타낸다.

입력변수 분할 단계에서는 입력변수 축소 단계에서 만들어진 DS를 바탕으로 동일한 테스트 신호선에 연결될 조합회로의 입력들을 결정한다. 이 단계에서 입력 변수들이 p개의 그룹 $-X_1, X_2, X_3, \cdots, X_p-$으로 분할되는데 p의 크기가 곧 테스트 신호선의 수를 의미하므로 p를 최소화할 필요성이 있다. 입력변수 분할 후 최소의 p를 갖는 입력 변수들의 분할이 여러개 존재할 수 있다.[7-9] 이러한 경우 검증 테스팅의 용이성 측면에서는 모든 입력변수 분할이 동등하지만 면적의 최소화 가능성을 동시에 고려하면 동등하지 않다.[7-9] 그러므로, 최소의 p를 갖는 입력 변수 분할 중 논리 최소화에 가장 바람직한 영향을 미치는 입력변수 분할을 선택할 필요성이 존재한다.

이를 위해 기존의 방법[7-9]은 DS에 부합되는 모든 maximal input subest을 생성한 후, 이 maximal input subset들 중에서 DS에 부합하는 모든 minimum cardinality input partition을 구하고, 논리 최소화에 바람직한 영향을 미치는 입력변수 분할을 휴리스틱을 이용하여 선택한다. 이로 인하여 많은 계산 시간이 요구된다. 입력변수 분할 문제는 최소의 테스트 신호선에 회로의 입력 신호선을 연결하는 문제이다. 그러므로, 본 논문에서는 입력 변수 분할 문제를 bin packing문제[11]로 모델링 한다. bin packing 문제에서는 bin의 수와 bin packing 순서가 중요하므로 3장에서 설명하는 휴리스틱 방법을 이용하여 이를 효과적으로 결정한다. 이러한 휴리스틱 방법을 적용함으로써 막대한 계산시간을 줄인다. 출력변수 분할 단계에서는 입력 변수 분할 단계에서 결정된 dependency subset DS^j_k를 이용하여 최소 면적의 논리 회로를 합성할 방법을 모색한다. 논리 최소화 단계에서는 입력 변수 분할 단계와 출력 변수 분할 단계에서 결정된 정보를 이용하여 논리 최소화를 행한다. 이 단계에서 사용되는 논리 최소화 알고리듬은 기존의 알고리듬[1]을 이용한다.

위와 같이 본 논문에서 제안하는 설계 방법은 크게 4단계로 구성되는데 입력 변수 축소 알고리듬[6]과 논리 최소화 알고리듬[1]은 기존의 알고리듬을 이용하므로 설명을 생략하고 입력 변수 분할 알고리듬과 출력 변수 분할 알고리듬에 대하여 상세히 기술한다.

Ⅲ. 입력 변수 분할 알고리듬

10개의 입력과 7개의 출력을 가지는 논리함수[7]에 대해 입력 변수 축소를 행하면 그림1[7]과 같은 결과를 얻는다. 입력 변수 분할 단계에서는 이러한 DS로 부터 테스트 신호선의 수와 회로의 면적이 최소화 되도록 입력 변수들을 p개의 그룹으로 분할한다. 검증 테스팅을 수행하기 위해서는 각 출력 y_i를 구현할 때 이용되는 dependency subset DS^j_k의 어떤 두 원소도 동일한 테스트 신호선에 연결 되어서는 안되는데,[10] 이것은 정의1[9]과 정리1[7]로 요약된다.

정의 1[9] : 입력 변수의 전체 집합이 X일 때, X의

DS

DS^1:
- $DS^1_1=\{x_1, x_3, x_4, x_7, x_8, x_{10}\}$
- $DS^1_2=\{x_3, x_4, x_7, x_8, x_9, x_{10}\}$

DS^2:
- $DS^2_1=\{x_1, x_3, x_4, x_7, x_8\}$
- $DS^2_2=\{x_1, x_4, x_7, x_8, x_{10}\}$
- $DS^2_3=\{x_1, x_4, x_8, x_9, x_{10}\}$

DS^3:
- $DS^3_1=\{x_2, x_3, x_6, x_7, x_8, x_9\}$
- $DS^3_2=\{x_2, x_3, x_7, x_8, x_9, x_{10}\}$
- $DS^3_3=\{x_2, x_4, x_5, x_7, x_8, x_9\}$
- $DS^3_4=\{x_2, x_4, x_6, x_7, x_8, x_9\}$
- $DS^3_5=\{x_2, x_4, x_7, x_8, x_9, x_{10}\}$

DS^4:
- $DS^4_1=\{x_1, x_2, x_4, x_5, x_7, x_{10}\}$

DS^5:
- $DS^5_1=\{x_1, x_2, x_3, x_5, x_7, x_{10}\}$
- $DS^5_2=\{x_1, x_2, x_3, x_7, x_9, x_{10}\}$
- $DS^5_3=\{x_1, x_3, x_4, x_5, x_7, x_{10}\}$
- $DS^5_4=\{x_1, x_3, x_5, x_6, x_7x_{10}\}$
- $DS^5_5=\{x_2, x_3, x_6, x_7, x_8, x_{10}\}$
- $DS^5_6=\{x_3, x_4, x_5, x_7, x_8, x_{10}\}$
- $DS^5_7=\{x_3, x_4, x_6, x_7, x_8, x_{10}\}$
- $DS^5_8=\{x_3, x_5, x_6, x_7, x_8, x_{10}\}$
- $DS^5_9=\{x_3, x_6, x_7, x_8, x_9, x_{10}\}$

DS^6:
- $DS^6_1=\{x_1, x_3, x_6, x_9, x_{10}\}$
- $DS^6_2=\{x_1, x_4, x_6, x_9, x_{10}\}$
- $DS^6_3=\{x_3, x_5, x_6, x_9, x_{10}\}$
- $DS^6_4=\{x_3, x_6, x_7, x_9, x_{10}\}$

DS^7:
- $DS^7_1=\{x_1, x_2, x_3, x_4, x_5, x_7\}$
- $DS^7_2=\{x_1, x_2, x_3, x_5, x_6, x_7\}$
- $DS^7_3=\{x_1, x_2, x_3, x_5, x_7, x_8\}$
- $DS^7_4=\{x_1, x_2, x_3, x_6, x_7, x_{10}\}$
- $DS^7_5=\{x_1, x_2, x_3, x_7, x_7, x_{10}\}$
- $DS^7_6=\{x_1, x_2, x_4, x_5, x_7, x_8\}$
- $DS^7_7=\{x_1, x_2, x_4, x_5, x_7, x_{10}\}$

그림 1. [7]의 10입력, 7출력 논리함수의 입력변수 축소 후 생성된 DS

Fig. 1. Result of argument reduction for a logic function with 10 inputs and 7 outputs of[7].

부분집합 X_i로 구성되는 집합 $IP=\{X_i|X_i\subset X,$ 서로 다른 i, j에 대하여 $X_i\cap X_j=\phi$이고 $\cup X_i=X\}$를 논리함수 F의 입력변수 분할 (input partition)이라고 한다. IP의 모든 원소 X_i에 대하여 X_i의 원소를 1개 이하 갖는 DS'_k가 각 DS'에 1개 이상 존재하면 "IP는 DS와 부합한다"라고 말한다.

정리 1[7]: 입력 변수 분할 IP가 논리함수 F의 DS와 부합한다면, IP에 의해 결정되는 검증 테스팅 계획으로 테스트할 수 있는 조합회로가 반드시 존재한다.

예를 들면, 상기 논리함수의 입력 변수의 전체집합 $X=\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}\}$이다. 입력 변수 분할 $IP'=\{\{x_1, x_9\}, \{x_2, x_3\}, \{x_4, x_6\}, \{x_5, x_8\}, \{x_7\}, \{x_{10}\}\}$는 그림1의 DS와 부합하므로 테스트 신호선에 조합회로의 입력을 그림2와 같이 연결하면 검증 테스팅이 가능하고, 이러한 검증 테스팅 계획으로 테스트할 수 있는 회로를 구현할 수 있다. 그러나, $IP''=\{\{x_1\}, \{x_2, x_3\}, \{x_5, x_8\}, \{x_6, x_7\}, \{x_{10}\}\}$는 그림1의 DS와 부합하지 않으므로 만약 입력 변수를 이와 같이 분할하면 검증 테스팅이 불가능하다.

그림1의 DS를 보면 각 출력 y_i를 구현하기 위하여 반드시 필요한 입력들이 존재하는데 이런 입력들은 반드시 서로 다른 테스트 신호선에 연결되어야 한다. 이와같이 서로 다른 테스트 신호선에 연결되

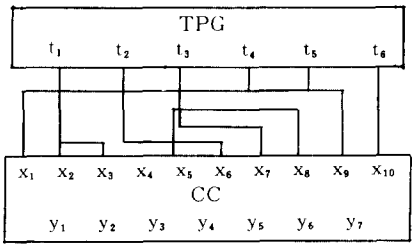

그림 2. 그림1의 DS에 대한 검증 테스팅 구성도

Fig. 2. Verification testing scheme for DS in fig.1.

어야 할 입력을 효과적으로 표현하기 위하여 정의 2와 정의 3의 개념을 도입한다.

정의 2[8]: DS'의 모든 DS'_k에 속하는 입력 변수들의 집합을 필수 집합이라고 하고 E_i로 표시한다.

정의 3[9]: 입력 부합 그래프(ICG:input consistency graph)는 n개의 입력 변수를 노드로 하는 무방향 완전 그래프로부터 E_i의 모든 원소들의 쌍으로 구성되는 간선(edge)을 제거하여 얻은 그래프이다.

E
- $E_1=\{x_3, x_4, x_7, x_8, x_{10}\}$
- $E_2=\{x_1, x_4, x_8\}$
- $E_3=\{x_2, x_7, x_8, x_9\}$
- $E_4=\{x_1, x_2, x_4, x_5, x_7, x_{10}\}$
- $E_5=\{x_3, x_7, x_{10}\}$
- $E_6=\{x_6, x_9, x_{10}\}$
- $E_7=\{x_1, x_2, x_7\}$

그림 3. 필수 집합

Fig. 3. Essential set.

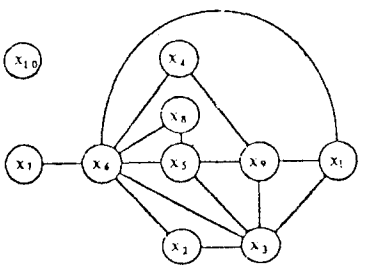

그림 4. 입력 부합 그래프

Fig. 4. Input consistency graph.

그림3은 그림1의 DS로부터 얻어진 필수 집합이고 그림4는 그림3으로 부터 얻어진 입력 부합 그래프이다. 이 그래프의 두 노드 사이에 간선(edge)이 존재

하지 않으면 이 두 노드로 표현되는 두 입력 신호선은 동일한 테스트 신호선에 연결될 수 없으므로 입력 변수 분할 단계에서 서로 다른 그룹으로 분할 되어야 한다.

입력 변수들을 최소의 그룹으로 분할하는데 소요되는 계산 시간을 줄이기 위하여 논리함수를 구현했을 때 요구되는 테스트 신호선의 수를 예측할 필요가 있다. 그림1의 DS를 보면 7개의 출력 중에서 가장 많은 입력에 의존하는 출력이 6개의 입력에 의존하고 있다. 검증 테스팅[10]에서는 어떤 출력을 구현하는데 필요한 입력이 동일한 테스트 신호선에 연결되면 곤란하므로 적어도 6개의 테스트 신호선이 요구된다. 그러므로 정리 2가 유도된다. ∎

정리2 : 논리함수 F의 dependency space DS에서 가장 많은 원소를 가진 DS'_k의 원소수를 τ라고 하면, 논리함수 F의 동작을 검증 테스팅 하는데 적어도 τ개의 테스트 신호선이 요구된다. DS에 부합하는 입력 변수 분할 IP의 원소수는 τ개 이상이다.

정리2에 의해서 결정된 테스트 신호선의 수는 bin packing문제[11]에서의 bin의 수에 대응되므로 입력 변수 분할 문제는 bin packing 문제로 변환된다. 따라서, 각 bin에 들어갈 입력 변수를 결정하면 된다. 그러나, bin packing 문제는 NP-hard이므로 계산 시간을 줄이기 위해서 휴리스틱 방법을 이용해야 한다. 휴리스틱 방법에서는 bin packing 순서에 따라 그 결과가 달라지므로 bin packing 순서를 결정할 필요가 있다.

그래서, 본 논문의 알고리듬에서는 $|E_j|$, $|DS'_k|$ 및 $|DS'_j|$를 고려하여 어느 출력의 입력들을 먼저 bin에 넣을 것인지를 결정하고, 결정된 출력 y_j를 구현할때 이용할 입력 변수들의 집합 DS'_k를 선택한다. DS'_k가 선택되면 DS'_k의 원소들을 입력 부합 그래프의 노드의 degree를 이용하여 bin에 넣는다. 이를 위하여 정의4를 도입한다. 입력 부합 그래프의 노드 x의 d(x)가 크다는 것은 입력x가 다른 입력과 함께 동일한 테스트 신호선에 연결될 가능성이 많다는 것을 의미한다. 예를 들면, 그림4의 입력 부합 그래프에서 x_{10}은 다른 입력과 테스트 신호선을 공유할 수 없는데 $d(x_{10})=0$인 반면에 x_6은 여러개의 다른 입력과 테스트 신호선을 공유할 수 있는데 $d(x_6)=7$이다. 따라서, 입력 부합 그래프의 노드 x의 degree를 나타내는 d(x)는 bin packing 순서를 정할 때 사용된다.

정의 4 : d(x)는 입력부합 그래프의 노드 x의 degree이다.

정의5는 bin packing 과정에서 bin을 증가시킬 필요가 있는지를 결정하는데 이용된다. bin packing 과정에서 각 bin에 들어갈 변수들이 결정되는데 같은 bin에 들어간 입력 변수는 검증 테스팅이 행해질 때 동일한 테스트 신호선에 연결될 것이므로 DS'_k들 중에서 어떤 bin의 원소를 두개 이상 포함하는 DS'_k는 출력을 구현하는데 사용될 수 없다. 그래서, 이러한 DS'_k는 ×로 표시한다.

정의 5 : 입력 변수의 전체 집합인 X의 부분집합 X'에 대하여 X'의 원소를 1개 이하 갖는 DS'_k가 DS의 각 DS'_j에 존재하면 "X'는 DS에 부합한다"라고 말한다.

출력 y_j를 구현할때 사용할 수 있는 DS'_k를 표현하기 위하여 정의 6을 도입한다. 정의 6의 $CDS'_j(X')$는 Krasniewski의 정의[7,9]와 유사해 보이지만 두 정의 사이에는 현격한 차이가 있다. 본 논문의 정의는 입력 변수 분할을 구하는 과정에서 변화하는 DS에 대하여 $CDS'_j(X')$를 동적으로 정의하는 반면에 Krasniewski의 정의는 고정된 DS에 대하여 $CDS'_j(X')$를 정적으로 정의한다. 정의 6으로 부터 정리 3이 유도된다.

정의 6 : X'는 입력 변수의 전체 집합인 X의 부분 집합이라고 하자. $CDS'_j(X')$는 DS'_j의 원소들 중에서 X'의 원소를 1개 이하 포함하면서 ×로 표시되지 않은 DS'_k들의 집합이다. 여기서, ×로 표시된 DS'_k는 입력 변수 분할이 일부 진행된 상태에서 또는 입력 변수 분할이 완료된 후에 출력 y_i를 구현하는데 이용할 수 없는 입력 변수의 집합이다.

정리 3 : 입력 변수의 전체 집합인 X의 부분 집합 X'가 DS와 부합하면 모든 j에 대하여 $CDS'_j(X')$는 공집합이 아니다. 그 역도 성립한다.

Bin packing의 순서를 결정하기 위하여 정의 7과 정의 8을 도입한다. 이를 이용하면 그림1의 DS로부터 그림5의 부분집합 관계표를 구할 수 있다. 부분집합 관계표에는 각 출력 y_i를 구현하는데 어느 DS'_k를 이용하면 좋은지에 관한 정보가 표현되어 있다. 검증 테스팅을 효율적으로 수행하기 위해서는 각 출력이 동일한 입력에 의존하면 유리하다. 그러므로, 출력 y_i를 구현하기 위해서는 부분집합 관계표로 부터 $S(DS'_k)$의 값이 큰 DS'_k를 선택하면 유리하다.

정의 7 : 부분집합 관계표 (SRT:subset relation table)는 두 집합 E_j와 DS'_k에 공통인 원소의 수를 이용하여 부분 집합들간의 관계를 나타낸 표이다.

정의 8 : $S(DS'_k)$는 부분집합 관계표에 있는 DS'_k 줄의 모든 값의 합이다.

입력변수 분할 알고리듬은 INPUT_PARTITIONING()이다. CANDIDATE_FAMILY()는 bin packi-

	E_1	E_2	E_3	E_4	E_5	E_6	E_7
DS_1	5	3	2	4	3	1	2
DS_2	5	2	3	3	3	2	1
DS'_1	4	3	2	3	2	0	2
DS'_2	4	3	2	4	2	1	2
DS'_3	3	3	2	3	1	2	1
DS'_1	3	1	4	2	2	2	2
DS'_2	4	1	4	3	3	2	2
DS'_3	3	2	4	4	1	1	2
DS'_4	3	2	4	3	1	2	2
DS'_5	4	2	4	4	2	2	2
DS'_1	3	2	2	6	2	1	3
DS'_1	3	1	2	4	3	1	3
DS'_2	3	1	3	5	3	2	3
DS'_3	4	2	1	4	3	1	2
DS'_4	3	1	1	3	3	2	2
DS'_5	4	1	3		3	2	2

	E_1	E_2	E_3	E_4	E_5	E_6	E_7
DS'_6	5	2	2	4	3	1	1
DS'_7	5	2	2	3	3	2	1
DS'_8	4	1	2	3	3	2	1
DS'_9	4	1	3	2	3	3	1
DS'_1	2	1	1	2	2	3	1
DS'_2	2	2	1	3	1	3	1
DS'_3	2	0	1	2	2	3	0
DS'_4	3	0	2	2	3	3	1
DS'_1	3	2	2	5	2	0	3
DS'_2	2	1	2	4	2	1	3
DS'_3	3	2	3	4	2	0	3
DS'_4	3	1	2	4	3	2	3
DS'_5	4	2	3	4	3	1	3
DS'_6	3	3	3	5	1	0	3
DS'_7	3	2	2	6	2	1	3

그림 5. 부분집합 관계표
Fig. 5. Subset relation table.

ng에 사용될 출력 y_j를 선택한다. 이 과정에서는 선택되지 않은 모든 출력들 중에서 가장 많은 원소를 갖는 E_j, 가장 많은 원소를 갖는 DS'_k 및 가장 적은 원소를 갖는 DS'를 동시에 고려한다. 이와 같은 방법으로 bin packing에 사용할 출력 y_j를 선택하는 이유는, $|DS'|$가 큰 출력은 다양한 방법으로 구현될 수 있기 때문에 일부 진행된 입력변수 분할에 의한 제약을 적게 받고, $|DS'|$가 작은 출력은 일부 진행된 입력 변수 분할에 의한 제약을 많이 받기 때문이다. 이러한 점을 고려하여 일부 진행된 입력변수 분할에 제약을 많이 받는 출력을 먼저 선택한다.

```
INPUT_PARTITIONING( ){
1  필수 집합을 구성한다.
2  입력 부합 그래프를 구성한다.
3  부분집합 관계표를 구성한다.
4  bin의 수 τ를 정한다.
5  τ개의 bin을 만든다.
6  WHILE 선택되지 않은 출력이 있는 동안
7    y_j = CANDIDATE_FAMILY( )
8    DS'_k = CANDIDATE_SUBSET(y_j)
9    DS'_k를 ○로 표시한다.
10   MERGE_SUBSET{τ, y_j, DS'_k}
11   p=τ
}

MERGE_SUBSET{τ, y_j, DS'_k} {
1  DS'_k의 원소가 들어 있는 bin을 +로 표시한다.
2  P=DS'_k-{x_i|x_i는 bin에 들어 있는 입력 변수}
3  WHILE 비어 있는 bin이 존재하고 P≠φ
4    P의 원소들 중에서 가장 작은 d(x)를 갖는 P의 원소 x를 선택한다.
5    x를 비어 있는 bin에 넣고, x를 넣은 bin을 +로 표시한다.
6    P=P-x
7  M=1
10 WHILE M≦|P|
11   M번째로 작은 d(x)를 가지는 P의 원소 x를 선택한다.
12   N=1
13   WHILE N≦τ
14     IF N번째로 큰 ∑d(y)를 가지는 bin B가 +로 표시되어
15       있지 않고, 모든 j에 대하여 CDS'(BUx)≠φ 이면
16         x를 B에 넣고 B를 +로 표시한다.
17         GOTO 19
18     N=N+1
19   M=M+1
20 IF P≠φ 이면
21   τ=τ+1로 하고, bin을 하나 증가시킨다.
22   GOTO 3
23 어떤 bin B에 대하여 |DS'_k∩B|≧2인 모든 DS'_k를 ×로 표시한다.
24 +로 표시된 모든 bin들의 +표시를 삭제한다.
}
```

CANDIDATE_SUBSET()은 CANDIDATE_FAMILY( )에 의해 선택된 출력 y_j의 구현에 사용될 DS'_k를 선택하는데, DS'의 원소이고 ×로 표시되지 않은 DS'_k 중에서 가장 큰 $S(DS'_k)$를 갖는 DS'_k 를 선택한다. 가장 큰 $S(DS'_k)$를 갖는 DS'_k를 선택하는 이유는, $S(DS'_k)$가 큰 DS'_k의 원소들을 출력 y_j를 구현하는데 사용하면 보다 많은 입력 변수가 여러 출력에 공통으로 사용되므로 결국 조합회로의 테스트 신호선의 수를 줄일 수 있기 때문이다. 선택된 DS'_k는 ○로 표시한다. ○로 표시된 DS'_k의 원소들은 출력 변수 분할과 논리 최소화 과정에서 출력 y_j를 구현하는데 이용된다. 그림6은 그림1의 DS에 INPUT_PARTITIONING( )을 적용했을 때 입력 변수들이 분할되는 과정을 나타내고 있으며, 그림7은 입력 변수 분할이 완료된 후의 DS을 나타낸다.

MERGE_SUBSET( )은 CANDIDATE_SUBSET( )에서 선택된 DS'_k의 원소들의 bin packing 순서를 결정하는 알고리듬이다. 이 알고리듬은 작은 d(x)를 갖는 집합 P의 원소 x를 큰 $\sum d(y)$를 갖는 bin B에 넣는다. 여기서, y는 bin B에 들어있는 원소이고, d(y)는 입력 부합 그래프에서의 노드 y의 degree이다. 이와 같이 작은 d(x)를 갖는 P의 원소 x를 큰 $\sum d(y)$를 갖는 bin B에 넣는 이유는 $\sum d(y)$가 큰 bin B에는 다른 입력들과 동일한 테스트 신호선에 연결될 수 있는 입력 변수들이 많이 들어있기 때문이다. 따라서, 작은 d(x)를 갖는 집합P의 원소 x를 큰 $\sum d(y)$를 갖는 bin B에 넣고, 큰 d(x)를 갖는 집합 P의 원소 x를 작은 $\sum d(y)$를 갖는 bin B에 넣는다. 이렇게 하면 모든 j에 대하여 CDS'(B)가 공집합이 될 가능성이 감소하므로 bin의 수 즉, 테스트 신호선의 수가 증가하는 현상을 억제할 수 있다.

단계1:j=4 $DS^4_1=\{x_1, x_2, x_3, x_4, x_7, x_{10}\}$
IP=$\{\{x_1\},\{v_2\},\{x_4\},\{x_5\},\{x_7\},\{x_{10}\}\}$
단계2:j=1 $DS^1_1=\{x_1, x_3, x_4, x_7, x_8, x_{10}\}$ $\{x_2\}$,
IP=$\{\{x_1\},\{x_2, x_3\},\{x_4\},\{x_5, x_8\},\{x_7\},\{x_{10}\}\}$
단계3:j=7 $DS^7_7=\{x_1, x_2, x_4, x_5, x_6, x_{10}\}$ x_7,
IP=$\{\{x_1\},\{x_2, x_3\},\{x_4\},\{x_5, x_8\},\{x_7\},\{x_{10}\}\}$
단계4:j=3 $DS^3_5=\{x_2, x_4, x_7, x_8, x_9, x_{10}\}$
IP=$\{\{x_1, x_9\},\{x_2, x_3\},\{x_4\},\{x_5, x_8\},\{x_7\},\{x_{10}\}\}$
단계5:j=2 $DS^2_2=\{x_1, x_4, x_7, x_8, x_{10}\}$
IP=$\{\{x_1, x_9\},\{x_2, x_3\},\{x_4\},\{x_5, x_8\},\{x_7\},\{x_{10}\}\}$
단계6:j=6 $DS^6_4=\{x_3, x_6, x_7, x_9, x_{10}\}$
IP=$\{\{x_1, x_9\},\{x_2, x_3\},\{x_4, x_6\},\{x_5, x_8\},\{x_7\},\{x_{10}\}\}$
단계7:j=5 $DS^5_3=\{x_1, x_3, x_4, x_5, x_7, x_{10}\}$
IP=$\{\{x_1, x_9\},\{x_2, x_3\},\{x_4, x_6\},\{x_5, x_8\},\{x_7\},\{x_{10}\}\}$

그림 6. 입력 변수 분할의 예
Fig. 6. Example of input partition.

DS
DS¹:
○ $DS^1_1=\{x_1, x_3, x_4, x_7, x_8, x_{10}\}$
$DS^1_2=\{x_3, x_4, x_7, x_8, x_9, x_{10}\}$
DS²:
$DS^2_1=\{x_1, x_3, x_4, x_7, x_8\}$
○ $DS^2_2=\{x_1, x_4, x_7, x_8, x_{10}\}$
× $DS^2_3=\{x_1, x_4, x_8, x_9, x_{10}\}$
DS³:
× $DS^3_1=\{x_2, x_3, x_8, x_7, x_8, x_9\}$
× $DS^3_2=\{x_8, x_3, x_7, x_8, x_9, x_{10}\}$
× $DS^3_3=\{x_3, x_4, x_5, x_7, x_8, x_9\}$
× $DS^3_4=\{x_3, x_4, x_6, x_7, x_8, x_9\}$
○ $DS^3_5=\{x_2, x_4, x_7, x_8, x_9, x_{10}\}$
DS⁴:
○ $DS^4_1=\{x_1, x_2, x_4, x_5, x_7, x_{10}\}$
DS⁵:
× $DS^5_1=\{x_1, x_2, x_3, x_5, x_7, x_{10}\}$
× $DS^5_2=\{x_1, x_2, x_3, x_7, x_8, x_{10}\}$
○ $DS^5_3=\{x_1, x_3, x_4, x_5, x_7, x_{10}\}$
× $DS^1_4=\{x_1, x_3, x_5, x_6, x_7, x_{10}\}$
× $DS^1_5=\{x_3, x_3, x_4, x_7, x_8, x_{10}\}$
× $DS^1_6=\{x_3, x_4, x_6, x_7, x_8, x_{10}\}$
× $DS^1_7=\{x_3, x_6, x_6, x_7, x_8, x_{10}\}$
$DS^1_8=\{x_5, x_6, x_7, x_8, x_9, x_{10}\}$
× $DS^4_1=\{x_1, x_3, x_6, x_9, x_{10}\}$
× $DS^4_2=\{x_1, x_4, x_6, x_9, x_{10}\}$
$DS^4_3=\{x_3, x_6, x_6, x_9, x_{10}\}$
○ $DS^4_4=\{x_3, x_6, x_7, x_9, x_{10}\}$
DS⁷:
× $DS^7_1=\{x_1, x_2, x_3, x_4, x_5, x_7\}$
× $DS^7_2=\{x_1, x_2, x_3, x_5, x_6, x_7\}$
× $DS^7_3=\{x_1, x_2, x_3, x_6, x_7, x_8\}$
× $DS^7_4=\{x_1, x_2, x_3, x_6, x_7, x_{10}\}$
× $DS^7_5=\{x_1, x_2, x_3, x_7, x_8, x_{10}\}$
× $DS^7_6=\{x_1, x_2, x_6, x_6, x_7, x_8\}$
○ $DS^7_7=\{x_1, x_2, x_4, x_5, x_7, x_{10}\}$

그림 7. 입력 변수 분할 후의 DS
Fig. 7. DS resulting from input partition.

Ⅳ. 출력변수 분할과 논리 최소화

입력 변수 분할에 의하여 결정되는 검증 테스팅 계획으로 테스트할 수 있는 회로를 설계하기 위한 가장 간단한 방법은 입력 변수 분할 과정에서 선택된 DS'_k의 원소로 각 출력을 독립적으로 논리 합성하는 것이다. 그러나, 기존의 논리 최소화 알고리듬에서 m개의 출력을 동시에 논리 최소화하면 최소 면적의 회로를 얻을 수 있었으므로 검증 테스팅을 고려한 논리 설계에서도 몇개의 출력을 동시에 논리 최소화하면 최소 면적의 회로를 구현할 수 있을 것으로 기대된다.

이것을 근거로 Krasniewski의 출력 변수 분할방법은 그림7의 DS에서 ×로 표시되지 않은 모든 DS'_k를 고려하여 출력 변수를 분할한다. 이러한 방법은 논리 최소화 과정에서 정확한 논리 최소화기를 이용하면 최소 면적을 갖는 조합회로의 생성을 보장한다. 그러나, 대규모 조합회로의 설계에는 [1]의 알고리듬을 바탕으로 하는 휴리스틱 논리 최소화기를 이용하므로 최소 면적을 갖는 회로의 생성이 보장되지 않는다. 따라서, 입력 변수 축소의 결과를 효율적으로 이용할 필요가 있다.

이를 위하여 본 논문에서는 함께 논리 최소화 할 출력들을 설정하는 간략한 출력 변수 분할 알고리듬을 구현했다. OUTPUT_PARTITIONING( )이 출력 변수 분할 알고리듬이다. 그림6의 입력 변수 분할과 그림7의 DS에 대한 출력 변수 분할의 결과 OP=$\{\{y_1, y_2\},\{y_3\},\{y_4, y_7\},\{y_5\},\{y_6\}\}$이다.

```
OUTPUT_PARTITIONING( ){
1   i=1
2   WHILE i≦m
3     j=i
4     WHILE j≦m
5       A=○로 표시된 DS'k
6       B=○로 표시된 DS'k
7       IF A⊂B이거나 A⊃B 이면
8         출력 yi와 출력 yj를 함께 논리합성한다.
}
```

논리합성 과정에서 면적만을 고려하는 기존의 이단 논리 최소화기[1]를 그대로 사용할 수 있다. 그러나, 입력 변수 분할에 의해 결정되는 검증 테스팅 계획으로 테스트할 수 있는 회로의 생성을 보장하기 위해서는 출력 변수 분할 과정에서 결정된 출력들만을 동시에 논리 합성해야 한다. 따라서, 함께 논리합성할 출력과 그 출력이 의존하는 입력만으로 구성되는 논리함수를 구성할 필요가 있다.

예를 들면, 출력변수 분할 과정에서 출력 y_1, y_2를 함께 논리합성 하기로 결정했고, 그림7의 DS에 의하면 y_1과 y_2를 구현하기 위해서는 DS^1_1과 DS^2_2의 원소인 $x_1, x_3, x_4, x_7, x_8, x_{10}$만 있으면 되므로[7]의 진리표에서 입력변수 x_2, x_5, x_6, x_9와 출력변수 y_3, y_4, y_5, y_6, y_7를 삭제하면 입력 변수 분할에서 결정된 검증테스팅 계획으로 테스트할 수 있는 출력 y_1, y_2를 기존의 이단 논리 최소화기[1]를 이용하여 논리 합성할 수 있다.

Ⅴ. 실험결과

본 논문에서 제안한 입력 변수 분할 알고리듬은

입력 변수 분할에 소요되는 계산 시간을 줄이는 휴리스틱 방법이므로 테스팅 용이성면에서 Krasniewski의 방법보다 열등한 회로를 생성할 수도 있다. 그러나, 본 논문의 입력 변수 분할 알고리듬과 [7, 9]의 입력 변수 분할 알고리듬을 [7-9]의 모든 회로에 적용하면 표1과 같은 결과를 얻는다. 입력 변수 분할 알고리듬에 대한 평가는 표1의 테스트 신호선의 수로 할 수 있는데 두 알고리듬의 결과가 검증 테스팅의 용이성 측면에서 동등하다. 입력 변수 분할에 소요되는 계산 시간도 표1에 나타냈다. 이시간은 Sun4/60에서 측정한 것이다. 입력 변수 분할 알고리듬을 적용한 예제 회로의 크기가 작기 때문에 정확한 평가는 어렵지만, Ⅲ장의 입력 변수 분할 알고리듬과 표1의 입력 변수 분할 시간으로 부터 입력 변수 분할에 소요되는 계산 시간이 단축됨을 알 수 있다. 따라서, 본 논문의 입력 변수 분할 알고리듬을 대규모 조합 회로의 설계에 이용하면 검증 테스팅이 용이한 회로를 단시간에 논리 합성할 수 있을 것으로 기대된다.

표1에서 볼 수 있는바와 같이 product term의 수와 literal의 수가 Krasniewski의 방법에 비해 감소되었다. 이것은 출력 변수 분할 과정에서 새로운 출력 변수 분할 알고리듬을 적용하고, 논리 최소화 과정에서 휴리스틱 알고리듬을 이용하였기 때문이다.

표 1. 실험결과
Table 1. Experimental results.

		기존의 방법	Kraniewski의 방법	본 논문의 방법
3입력	#product terms	5	5	5
4출력	#literals	17	16	17
회로 1	#test signals	4	3	3
[8]	#test patterns	8	8	8
	계산시간(초)	-	0.1	0.0
10출력	#product terms	23	34	33
7출력	#literals	133	143	136
회로 2	#test signals	10	6	6
[9]	#test patterns	165	64	64
	계산시간(초)	-	0.3	0.1
10출력	#product terms	26	39	38
10출력	#literals	146	168	161
회로 3	#test signals	10	6	6
[7]	#test patterns	165	64	64
10입력	계산시간(초)	-	0.5	0.2

Ⅵ. 결 론

본 논문에서는 검증 테스팅이 용이한 조합회로를 단시간에 설계하기 위한 새로운 휴리스틱 방법을 제안했다. 입력 변수 분할 과정에서 입력 변수분할 문제를 bin packing 문제로 모델링하고, bin packing 문제를 dependency subset들의 관계를 고려하여 입력 변수 분할에 소요되는 막대한 계산시간을 줄였다. 또한, 출력 변수 분할 과정에서 입력 변수 축소와 입력 변수 분할 과정에서 얻어진 정보를 효율적으로 이용하여 논리 회로의 면적을 줄였다.

본 논문에서 제안하는 설계 방법이 휴리스틱에 근거하여 입력 변수를 분할하므로 대규모 조합 회로의 설계에 이용될 수 있는 것으로 기대된다. 앞으로의 연구과제는 출력 변수 분할이 휴리스틱을 이용하는 이단 논리 최소화기에 미치는 영향을 효과적으로 고려할 방법을 찾는 것이다.

參 考 文 獻

[1] R.K. Brayton, G.D. Hactel, C.T. McMullen, and A.L. Sangiovanni-Vincentelli, Logic Minimization Algorithma for VLSI synthesis, Kluwer Academic Publishers, Boston, 1984.

[2] R.K Brayton, "Algorithms for mufti-level logic synthesis and optimization", in G.De Micheli, A. Sangiovanni-Vincentelli, P. Antognetti, Design systems for VLSI Circuits, Martinis Nijhoff Publ., Hingham 1987.

[3] K-T. Cheng, and V.D Agrawal, "Synthesis of testable finite state machine", *Proc. IEEE Int. Sump.* on Circuits and Systems, pp. 3114-3117, 1990.

[4] S. Debvadas, H-K. T.Ma, A.R. Newton, and A.L. Sangiovanni-Vincentelli, "Optimal logic synthesis and testablity: two faces of the same coin", *proc. International Test Conf.*, pp. 3-13, Sep. 1988.

[5] S. Devadas and K. Keutzer, "A unified approach to the synthesis of fully testable sequential machines", *Proc. 23rd Annual Hawaii international Conf.* on System Science, pp. 427-435, 1990.

[6] K. Jasinski, T. Luba, J. Kalinowski, "Paralel decomposition in logic synthesis", *Proc. 15th European Solid State Conf.*, pp. 113-116, 1989.

[7] A. Krasniewski, "Verification testing oriented logic synthesis", Technical Rport no 135, Warsaw Univ. of Technology, Institute of Telecommunications, Sept. 1989.

[8] A. Krasniewski, "Design for verification testability", *Proc, European Design Automation conf.*, pp.664-648, 1990.

[9] A.Krasniewski, "Logic Synthesis for Efficient Pseudoexhaustive Testability", *Proc. 28th Design automation Conf.*, pp. 66-72, 1991.

[10] E.J. McCluskey, "Verification Testing - Pseudoexhaustive Test Technique", *IEEE Trans.* on Computers, vol.. C-33, pp. 541-546, Jun. 1984.

[11] E. Horowitz, S. Sahni, Fundamentals of Computer Algorithms, Computer Science Press, *Inc.*, Maryland, pp. 572-574, 1978.

著 者 紹 介

李 永 浩 (正會員)

1965年 8月 7日生. 1989年 한양대학교 전자공학과 졸업. 1991年 한양대학교 전자공학과 대학원 석사학위 취득. 현재 한양대학교 전자공학과 박사과정 재학. 주관심분야는 VLSI 설계 및 CAD 특히, Logic Synthesis 및 Synthesis for Testability 등임.

鄭 正 和(正會員) 第28卷 A編 第10號 參照

1950年 3月 10日生. 1975年 한양대학교 전자공학과 졸업. 1981年 3月 일본 와세다대학 박사학위 취득. 일본 NEC(주) 중앙연구소 연구원, University of California, Berkeley 교환교수 역임. 현재 한양대학교 전자공학과 교수. 주관심분야는 VLSI CAD 특히, High-level Synthesis, Logic Synthesis, Layout Synthesis 등임.