論文 87-24-2-23

논리회로 상호간의 연결도 검증

(Verification of Logic Gate Interconnection)

鄭 子 春** 慶 宗 旻*

(Ja Choon Jung and Chong Min Kyung)

要 約

이 논문은 논리설계 단계에서의 논리회로와 layout에서부터 추출된 논리회로를 직접비교함으로써 layout 상에서의 연결상태를 검증하는 방법에 관해서 기술한다. 두개의 논리회로를 weighted multi-place 그래프로 나타내었고 다음에 논리회로가 가지는 topology를 그래프 isomorphism에 근거하여 비교하였으며 연결상의 차이점이 발견되면 error message를 내보낸다. Candidate selection과 equal weight partition의 두 단계로 이루어진 효율적인 그래프 isomorphism test algorithm은 전체적인 칩의 연결상태를 검증하는데 O(n log n)의 시간이 필요하다.

Abstract

This paper describes a method for verifying whether a given geometrical layout correctly reflects the original logic level description. The logic description extracted from layout data was directly compared with the original logic diagram generated at logic level design stage where the logic diagram is represented as a weighted multi-place graph. The comparison is based on graph isomorphism and error messages (error categories and locations) are invoked if any difference is found between the two logic descriptions. An efficient partitioning algorithm which consists of two steps, candidate selection and equal weight partitioning procedure, enables the entire verification process to occur in O(n log n) time.

I. 서 론

최근의 VLSI 설계 동향을 보면, 칩 설계시간을 줄이기 위해서는 표준셀 및 gate array 방식등이 사용되고 있으나 칩 면적을 최소화 한다는 관점에서는 manual editor에 의한 설계 시스템이 중요한 부분을 차지하고 있다. 그러나, 이 방법은 설계과정에서 설계자의 실수에 의한 오류를 피할 수 없게 되며, 이러한 오류가 제작된 칩을 테스트하는 과정에서 발견된다면, 소요되는 시간이나 경비면에서 매우 비경제적이므로 오류가 없는 layout을 보장하기 위해서 여러가지의 검증 program이 개발되어 사용되고 있다.[1-7]

오류가 발생한 곳을 쉽게 찾기 위해서, 논리설계단계에서의 논리회로와 layout 데이타로 부터 추출된 논리회로 상호간의 연결도를 직접 비교하는 것은 검증하는데 걸리는 시간과 완벽한 검증이라는 관점에서 매우 효과적임이 입증되었다.[3,4] 그림 1은 논리회로 설계자료를 보고 사람이 layout editor를 써서 작성한 설계도면이 원하는 대로 그려졌는지를 검증하기 위한 과정을 보인 것이다.

그림 1에서 보는바와 같이 논리설계단계에서 작성된 논리회로에 대한 정보는 logic simulator 입력 file에서 얻을 수 있으며, layout 데이타로 부터 논리추출은 (1) layout 데이타로 부터 transistor 회로추출,

*正會員, 韓國科學技術院 電氣 및 電子工學科
(Dept. of Elec. Eng., KAIST)
**正會員, 韓國電子通信研究所
(Electronics and Telecommumnications Research Institute)

接受日字 : 1986年 3月 27日

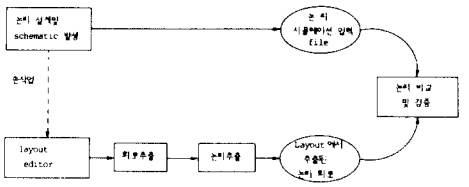


그림 1. 논리 검증 유통도
Fig. 1. Logic verification flowchart.

(2) 추출된 트랜지스터 회로의 조합에서 논리 gate 추출의 두 과정을 거친다. (1)의 과정은 이미 연구된 바 있으며,[7] 본 논문에서는 (2)의 과정과 두개의 논리회로를 비교하므로써 layout상에서 연결도를 검증하는 방법에 대해서만 기술한다.

Layout 데이타와 논리설계단계에서 논리회로를 직접 비교함으로써 layout 상에서 연결상태를 검증할 수 있는 많은 방법들이 제안되고 있다.[1-6] 초기의 방법[1]은 layout상에 logic simulator의 단자나 소자의 대응되는 이름을 미리 정해주고 각각 단자에 해당되는 소자가 올바르게 연결되었는가를 검사하였다. 이렇게 하면 논리검증하는 방법은 매우 쉬워지나, layout 데이타상에 각각의 이름을 붙여주는 것은 설계자에게 부담이 될 뿐아니라 또다른 오류의 근원이 될 수 있다.

최근에 와서[3,6] 그래프 isomorphism을 이용하여 이 문제를 해결하고자 하는 방법이 제시되었다. 예를들면, E. Barke[6]의 algorithm은 비교적 크기가 작은 bipolar 설계 시스템에 잘 동작하나 규모가 큰 VLSI 회로에서는 구체적인 언급이 없다. T. Watanabe 등[3]의 방법은 4가지의 다른 분할 방법과, subgraph isomorphism algorithm을 사용하였다. 이렇게 하면 논리검증을 하고 오류가 발생한 곳을 정확히 지정할 수 있으나, 분할방법에서 사용된 shortest path algorithm은 $O(n;$ 여기서 n은 기본적인 소자 수)의 시간이 소요된다. 이 논문에서 제안되고 구현된 그래프 isomorphism algorithm은 거리분할과 무게분할의 두 과정으로 구성되어 있는데, 전체적인 검증시간이 $O(n;$ n은 소자 수)의 시간이 필요하다. 2장에서는 트랜지스다 회로도로부터 논리게이트를 추출하는 방법에 대하여 간단히 설명하고 3장에서는 논리비교에 의한 검증을 설명했으며 끝으로 간단한 예제에 대하여 program을 run시킨 결과를 보였다.

Ⅱ. 논리추출 과정

Layout 데이타로 부터 얻어진 transistor level 회로도로 부터 논리회로를 추출하는 것은 논리 simulation을 한다든지 논리비교에 의하여 연결도를 검증하기 위해서 필요하다. 회로추출 과정[7]과 마찬가지로 이 과정도 process 기술과 관계가 있으며, 본고에서는 NMOS와 CMOS로 된 디지탈 회로에 대해서만 논리를 추출하는 방법을 기술한다. 논리 추출을 용이하게 하기위해서 전체의 transistor 회로도에서 기본적인 논리기능을 형성하는 transistor들의 그룹을 찾아내는 회로분할 과정과 각 그룹내의 transistor들의 연결상태를 살펴서 기본적인 게이트를 찾아내는 두개의 절차로 구성되어 있다. 이 과정의 결과는 게이트 level logic simulator인 KAISLOG[10]의 입력으로 작용된다.

1. 회로분할 과정

Transistor들의 그룹을 찾기위해서는 MOS transistor로 이루어진 디지탈 회로가 어떻게 논리를 형성하는지를 알아야 한다. 이 방법은 여러가지가 있으나 본고에서는 transistor의 역할에 따라 load부분과 driver 부분으로 이루어진 디지탈회로에 대해서만 논리를 추출하는 방법을 기술한다. 그림 2 (a)에는 depletion NMOS 회로의 load블럭과 driver 블럭을 보였다. CMOS 논리회로는 그림 2 (b)와 같이 p-channel MOSFET로 이루어진 load블럭과 n-channel MOSFET 로 이루어진 driver블럭으로 구성된다. NMOS나 CMOS 회로에서 logic을 형성할 수 있는 또 다른 형태인 pass transistor에 의한 것은 그림 2 (c)에 보이고 있다.

2. 기본적인 Gate로 분리

전 단계에서 형성된 transistor의 그룹은 NOR gate, NAND gate 또는 임의의 AOI (AND-OR-INVERTER) 블럭이다. 각각의 분리된 transistor 그룹내의 transistor들은 직렬 혹은 병렬로 연결되어 있는데 그 연결상태를 tree 구조로 나타내었다. 임의의 AOI 블럭의 tree 구조 표현을 그림 3에 나타내었다.

Tree에서 각각의 vertex는 '*' (또는 '+')로 표시되었는데, 이것은 직렬(또는 병렬)로 연결된 요소들을 나타낸다. 각각의 leaf는 각 gate의 입력단자로 표시되었고, root는 출력단자를 표시한다. 원하는 논리를 얻기위해서는 tree의 변형이필요한데 여기서는 depth first traversal 방법을 써서 descendent node들을 다 방문한 다음에 다음 node를 찾는다. 예를들면, load tree의 경우 '+'로 표시된 node는 그 descendent node들을 입력으로 하는 OR gate를 나타내고, '*'로 표시된 node는 AND gate를 나타낸다. 출력은 그 subtree의 직렬 또는 병렬의 연결상태에 따라 NAND 또는 NOR gate가 된다. 그림 3에서 추출된 논리회로는 그림 4와 같다.

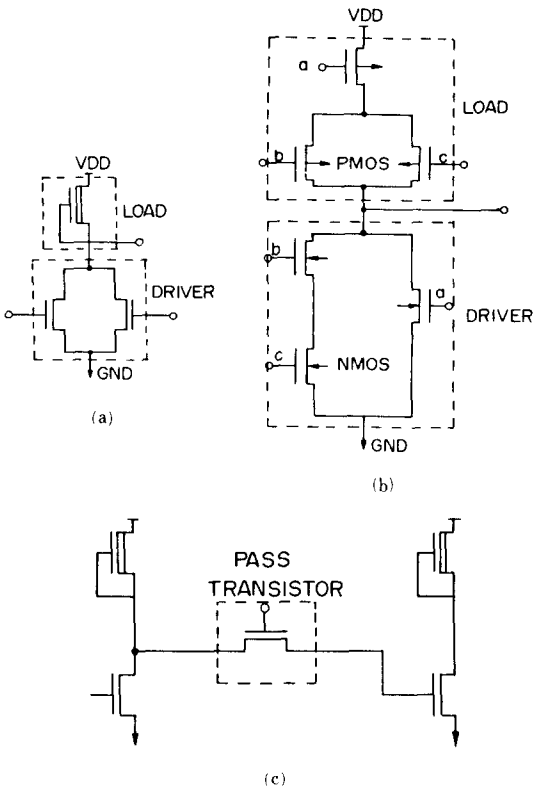


그림 2. (a) Depletion NMOS 회로;(b) CMOS회로; (c) Pass transistor에서의 여러가지 논리 블럭.

Fig. 2. Various logic blocks in (a) Depletion NMOS; (b) CMOS and (c) Pass transistor.

이상에서 설명한 논리블럭으로 분해단계와 각 논리블럭을 그림 5에 보인것과 같은 기본 gate들로 분해하는 단계를 모두 거치면 주어진 회로에 대한 gate level 연결정보가 얻어진다. 예를들어, NAND/NOR gate 가 서로 물린 형태로 되어 있으면 NAND/NOR latch로 추출된다. 이와같이 추출된 소자들은 논리 시뮬레이터[10] 의 입력을 나타내기에 충분하다.

Ⅲ. 논리비교에 의한 검증

이 단계에서는 논리설계단계에서 만들어진 gate-level logic simulator의 입력file과 layout에서 추출된 논리회로를 그래프로 표시하여 두개의 논리회로를 비교하는 문제를 두개의 그래프에 대한 isomorphism 을 검사하는 문제로 변환하였다.

1. Multi-place 그래프

복잡한 논리회로를 표시하는데는 multi-place 그래프

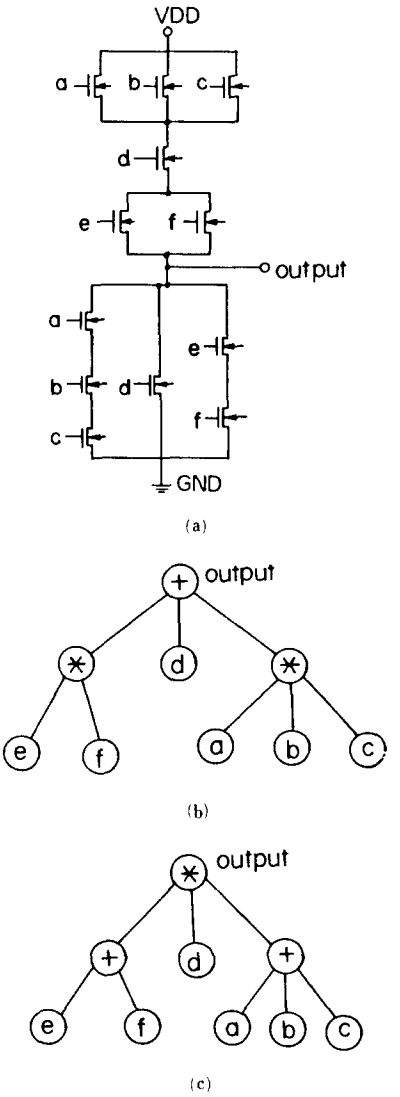


그림 3. AOI 논리 블럭의 (a) 회로도, (b) driver tree, (c) 부하 tree

Fig. 3. (a) Circuit representation of a AOI logic block. (b) driver tree and (c) load tree.

를 쓰면 편리하다. Multi-place 그래프는 참고문헌[6, 9]에 잘 정의되었다.

Multi-place 그래프는 node와 spider로 구성되어 있는데, node는 각각의 특정지워질 수 있는 요소를 나타내는데, 논리회로상에서 각각의 gate를 표시하며, spider는 각 gate간의 연결선을 나타낸다. Multi-place 그래프의 각 node의 입·출력단자에 어떤 특정한 값으로 weight를 주어서 표시한 그래프를 weighted device 그래프라고 한다. 그리고 spider의 weight는 그 spider에 가입된 각 node의 weight의 곱으로 표시된다.

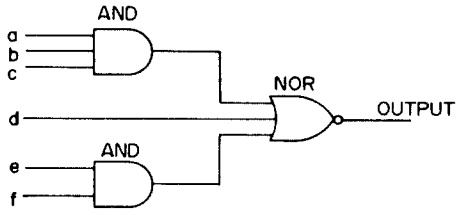


그림 4. 그림 3 의 tree 모델에서 주출한 논리 회로도
Fig. 4. logic circuit extracted from the tree models of Fig. 3.

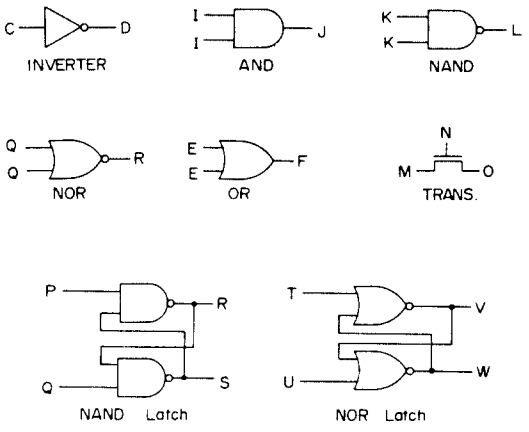


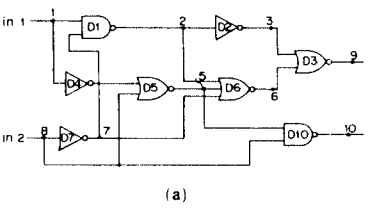
그림 5. 기본논리 소자들과 각 핀의 알파벳 무게치
Fig. 5. Basic logic elements and their pin weights in alphabet.

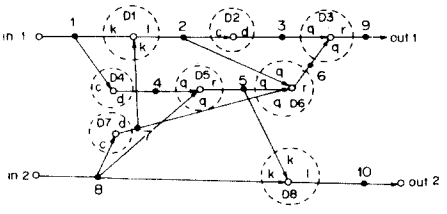
Device그래프에 weight를 주기 위해서 각각의 gate에 그림 5 와 같이 특정한 문자를 할당하였다. AND, OR, NAND 그리고 NOR gate의 입력단자들이 똑같은 문자로 weight된 것은 이 pin들이 서로 바꾸어져도 무관하기 때문이다. 그림 6 은 임의의 논리회로를 weighted device 그래프로 표시한 예를 보이고 있다.

예를들어, 그림 6 (c)에서 spider 5 의 weight가 "kqr"인데 이것으로 부터 논리회로상에서 5번 신호선은 NOR gate와 NAND gate의 입력단자와 연결되고 또 다른 NOR gate의 출력단자가 연결되었다는 것을 알 수 있고, d6의 weight는 "qqqr"인데 이것은 d6 device가 3-input NOR gate 라는 것을 나타낸다.

2. Graph isomorphism 검사

지금까지 모든 그래프에 대해서 polynomial 시간내에 해결할 수 있는 algoritm은 알려지지 않았으며, 그렇다고 그래프 isomorphism을 검사하는 문제가 NP-complete하다고 증명된바도 없다.[11] 그래프 isomorphism을 검사하는 과정은 크게 분할과정과 mapping과정으로 나누어진다. 분할과정은 같은 성질로 특정지어지는 부분으로 그래프의 요소(일반적인 그래프에서 edge나 vertex)를 분해하는 과정이고, mapping 과정은 같은 성질의 그룹에서 그래프의 요소들을 대응시키는 과정이다. 그런데 큰 그래프의 경우에는 mapping 과정에서 많은 시간이 소모되기 때문에 분할과정에서 mapping을 피할 수 있도록 하는 algorithm이 필요하다. T. Watanabe $O(n^3)$[3]와 N. Kubo $O(n^3)$[11] 등은 일반적인 digraph에서 vertex 사이의 shortest path를 구하여 그래프를 분할했는데 이 경우 shortest path를 찾는데 $O(n^3)$의 시간이 필요하다.





Spider \ 게이트	D1	D2	D3	D4	D5	D6	D7	D8	Spider 무게
1	k			c					ck
2	l	c				q			clq
3		d	q						dq
4				d	q				dq
5					r	q		k	kqr
6			q			r			qr
7	k					q	d		dkq
8					q		c	k	ckq
9			r						r
10								l	l
게이트 무게	kkl	c d	qqr	cd	qqr	qqqr	cd	kkl	

(c)

그림 6. (a) 임의의 논리회로, (b) 그것의 weighted device graph, (c) weighted incidence행렬
Fig. 6. (a) A sample logic circuit, (b) its weighted device graph, (c) the relevant weighted incidence matrix.

이 논문에서의 분할과정은 논리회로가 가지는 성질을 이용하여 다음과 같이 거리분할과 무게분할의 두단계로 구성하였다. 논리회로는 입력 단자에서 부터 출력단으로 향하는 신호 방향을 가지고 있기 때문에 논리회로를 표시한 weighted device graph는 digraph로 나타낼 수 있다.

첫째, 거리분할 단계에서는 특정한 입력단자에서 부터 거리에 따라서 spider를 분리시킨다. (이때 기본 gate 한개를 통과할 때 마다 거리가 1씩 증가된다) 둘째, 무게분할 단계에서는 각 spider에 할당된 특정한 문자들로 이루어진 weight에 의해 spider들을 분리시킨다. 이 두가지 분할 방법에 의하면 종래의 그래프 isomorphism을 검사하는 algorithm에서 볼 수 있었던 mapping에 의한 시간소모를 피할 수 있다. 다음에는 거리분할과 무게분할 방법에 대해서 구체적으로 설명한다.

1) 거리분할 절차

특정한 입력단자와의 거리에 따라 spider를 분리시켰다. 그림 7은 두개의 입력단자 in 1과 in 2로 부터 각 spider의 거리값을 산출하는 과정을 보여주고 있다.

어떤 특정한 입력단자에서 부터 시작하여 한 spider를 지날 때마다 candidate distance를 1만큼 증가시켰고, 만약, 선택된 spider가 이미 다른 값으로 할당되었다면 둘중에 큰값을 할당하도록 하였다. 입력단자 in 1에 대해서 거리값에 따라 각 spider를 그룹으로 나눈 결과는 표1과 같다. 입력단자 in2에 대해서도 똑같은 방법을 적용할 수 있다. 두 그래프G와 G'에서의 각 path가 서로 isomorphic한 path인가를 결정하는 것은 특정한 입력단에서 부터 연결된 path에 대해 거리분할에 의해서 분리된 각 그룹에 속하는 spider의 무게가 같을 때 그 path를 isomorphic한 path로 간주한다. 주어진 두개의 그래프 G와 G'의 모든 path쌍이 서로 isomorphic한 관계에 있으면 G와G'는 서로 isomorphic하고, 그렇지 않으면 isomorphic한 path에 의하여 cover 되지 않는 요소들의 gate 형태나 연결상태를 검사함으로써 잘못된 부분을 찾아낼 수가 있다.

2) 무게분할과 matching

Isomorphic path 상에서 거리값에 의해서 분리된 그룹내의 spider가 두개 이상이면 이들을 character weight에 의해서 다시 분리시킨다. 그림 8은 비교될 두개의 논리회로에 대해서 이 과정의 진행과정을 보이고 있다. 입력단자 in 1에 대해서 거리분할에 의해 분리된 각 그룹내의 spider에 대해 무게분할을 적용하면 표 2와 같다.

이 표에 의하면 group 2를 제외한 모든 그룹에는 spider가 한개만 있고 그것의 character weight가 같

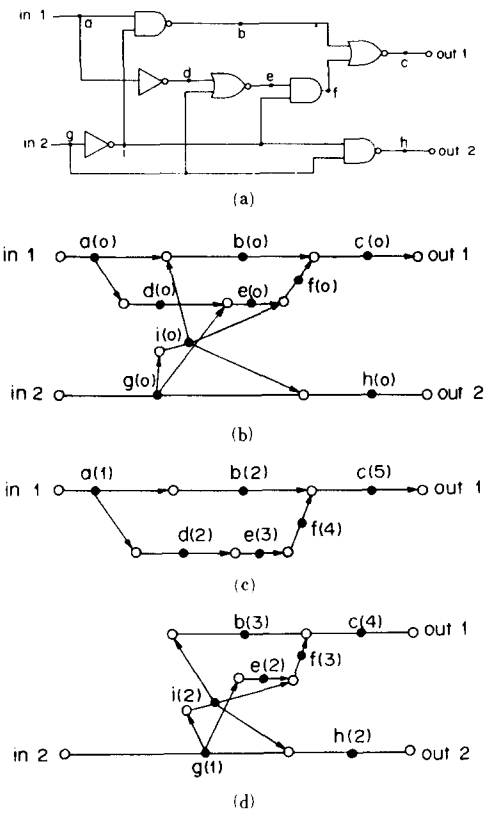


그림 7. 거리분할 과정을 보이는 그림
(a) 2입력, 2출력 논리회로의 각 spider 이름,
(b) 각각 0으로 초기 reset된 각 spider의 거리값,
(c) 입력 1과 (d) 입력 2에 대한 각각의 거리값 산출결과

Fig. 7. Illustration of distance grouping operation.
(a) Each spider's name(weight) in a 2-input, 2-output logic circuit.
(b) Initial distance value of every spider set to zero.
(c) Evaluation of distance value for in 1 and (d) in 2.

표 1. 1번 입력단으로 부터의 거리분할에 따른 각 그룹과 소속된 spider

Table 1. Each group and the associated spiders according to the distance grouping for in 1.

in 1으로 부터의 거리값에 따른 그룹	각 그룹에 속한 spider
Group 1	a
Group 2	b, d
Group 3	e
Group 4	f
Group 5	c

표 2. 기준논리회로와 추출된 논리회로상의 각 대응되는 spider의 무게치의 비교

Table 2. Comparison of character weights of the corresponding spider in the reference and retrieved logic circuit.

	Reference	Retrieved
각 distance 그 룹	Character weight (Spider)	Character weight (Spider)
Group 1	ck (1)	ck (1)
Group 2	lq (2)	dq (2)
	dq (3)	lq (5)
Group 3	er (4)	er (3)
Group 4	fq (5)	fq (4)
Group 5	r (6)	r (8)

표 3. 기준 논리회로와 추출된 논리회로상의 각 노드와 spider 간의 대응관계

Table 3. Correspondence between nodes and spiders in the reference and retrieved logic circuit.

대응된 Spider		대응된 Node	
Reference	Retrieved	Reference	Retrieved
1	1	d 1	d 1
2	5	d 2	d 4
3	2	d 3	d 1
4	3	d 4	d 2
4	4	d 5	d 3
6	8		

표 4. in 1 단자로 부터 out 1 단자간의 경로상에서의 거리분할에 의한 각 대응 spider 무게치의 비교

Table 4. Comparison of the corresponding spider's weight according to the distance grouping from in 1 to out1.

	Spider의 무게치 (괄호안은 spider의 번호)	
거리그룹	기준회로	추출된 회로
Group 1	ckq (8)	ckk (6)
Group 2	dekk (7)	dekq (7)
Group 3	lq (2)	er (3)
	fq (5)	–
Group 4	r (6)	fq (4)
Group 5	–	r (8)

으므로 쉽게 matching이 되며, group 2의 경우 character weight에 의해서 다시 분리시키면 (lq) 와 (dq) 로 분리되므로 matching 시킬 수 있다. 일단 spider가 matching 되면 그것에 연결된 node도 matching 시킬 수가 있다. 입력단자 in 2 경로상에 있는 요소들 (spiders 또는 nodes) 의 대응 요소들은 표 3 과 같다. 역시 그림 8 에 대하여 입력단자 in 2 에 대한 거리분할 방법을 적용한 결과는 표 4 (out 1 로 향하는 path) 와 표 5 (out 2 로 향하는 path) 와 같다. 이 경우에는 출력단자 out 1 과 out 2 로 향하는 두개의 경로가 존재하는데 어느 경우에도 모든 path쌍이 isomorphic하지 않음을 알 수 있다. 이 경우에는 우선 matching 가능성이

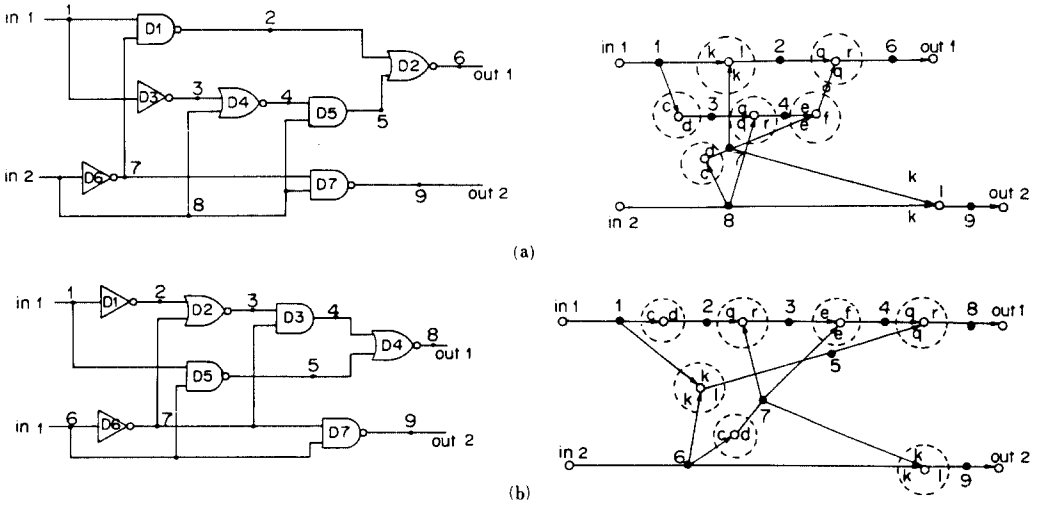


그림 8. (a) 한 논리회로A에 대한 회로도와 weighted device 그래프
(b) 다른 논리회로B에 대한것

Fig. 8. (a) Circuit representation and the weighted device graph model for a logic circuit "A".
(b) for another logic circuit"B".

표 5. 1번 입력단자로 부터 2번 출력단자간의 경로에 대한 표4와 같은 표현

Table 5. Similar expression as Table4 for paths from in 1 to out 2.

거리그룹	spider의 무게치(spider의 번호)	
	기준회로	추출된 회로
Group 1	ckq (8)	ckk (6)
Group 2	dekk (7)	dekq (7)
Group 3	ℓ (9)	l (9)

가장 큰 spider나 node를 찾아서 matching 시키고 나머지는 error로 처리하게 된다.

그림 9에는 두개의 논리회로를 비교할 때 어떤 node가 다른 node와 구분될 필요가 있는 경우와 없는 경우를 보이고 있다. 즉, 그림 9 (a)에서 두개의 입력 in1과 in 2는 서로다른 입출력 특성을 가지고 있는 반면 그림 9 (b)의 경우에서 처럼 서로 구분할 수가 없으면 어떤쪽을 matching 시키든지 나중에 일관성만 보장된다면 topology 상에는 아무런 변화가 없다.

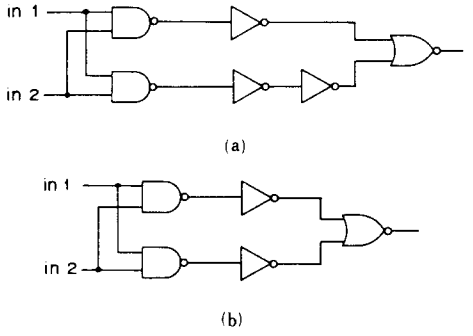


그림 9. (a) 두개의 입력이 구분되는 경우와 (b) 구분되지 않는 경우

Fig. 9. (a) Distinguishable inputs, (b) Indistinguishable inputs.

3) 논리 검증 algorithm의 시간 해석

여기에서는 matching algorithm의 시간에 대해서 언급한다. VLSI설계에서는 그 취급하는 데이타량이 많기 때문에 소자의 수가 커짐에 따라 시간이 급격하게 증가하지 않도록 algorithm을 생각하는 것이 중요하다. 먼저 거리분할 절차에서는 어떤 특정한 입력단자에서 부터 연결된 요소들을 하나씩 지나가면서 수행되기 때문에 요소들의 수에 비례하는 시간이 소요된다. 그러므로 전체적인 spider수를 s, node수를 n이라고 하면 이 경우의 시간은 $O(s+n)$이 된다. 무게분할과 matching 절차에서는 서로 대응되는 요소를 찾아서 그 matching된 spider나 node들을 AVL tree를 사용하여 저장하였다. AVL tree는 어느 vertex에서도 양쪽 가지로의 깊이의 균형이 항상 유지되도록 하는 tree 구조로서 loading 시간과 searching시간이 모두 $O(\log n)$이 된다. 오류를 찾는 부분에서는 오류지역에 있는 요소들의 연결상태만을 검사하기 때문에 이 경우에는 $O(c \log n)$의 시간이 필요하므로 (c는 오류지역에 있는 요소의 수이다) 전체적인 소요시간은 $O(n \log n)$이 된다. 그림10에는 사용된 MV 10000 computer의 CPU 시간을 gate 숫자에 따라 측정한 결과를 보였다.

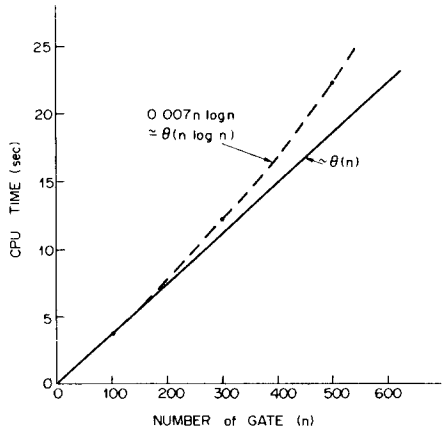


그림10. 논리게이트의 숫자에 따른 CPU시간

Fig. 10. CPU time vs. number of logic gates.

V. 실험 결과

이 장에서는 작은 example에 대해서 program을 run 시킨 결과를 보였다. 그림11(a)는 논리설계 단계에서의 논리회로를 나타내는 logic simulator input file이며, 그림11(b)는 layout data로 부터 추출된 회로를 나타내는 switch level netlist file이다. 그림11(c)는 분리추출기를 거친 gate level netlist 결과이고, 그림11(d)와 (e)가 각각 program 결과로서 서로 대응되는 signal net와 gate를 나타내고 추출된 논리회로를 나타내는 그림11(c)가 의 netlist 그림11(a)의 입력논리회로와 어떻게 다른가를 error message로 나타내었다.

VI. 결 론

이 논문에서는 두개의 논리회로간의 equivalence를 검증할 수 있는 program을 설명하였다. 이 program은 전체적인 칩의 연결상태를 검증하는데 사용될 수 있으며 오류에 대한 자세한 정보(오류의 종류와

```
*  Input data file which represent the logic
*  diagram of the logic level design stage.
$  LIST
$  TITLE
%%% RANDOM LOGIC CIRCUIC FOR TESTING %%%
///
///
///    Signal description
///
$  DELAY
$  END_DELAY
$  INPUT
       INPUT Ø
       INPUT1 1
       INPUT2 Ø
       INPUT3 1
$  END_INPUT
///
/// connectivity description
///
NOT  SIG2   INPUT1
NOT  SIG3   SIG2
AND  SIG4   SIG3   SIG8
NOR  SIG5   SIG4   SIG9
NOT  SIG6   SIG5
NOR  OUT1   SIG6   SIG7
NAND  SIG7   SIG5   SIG1Ø
NOT  SIG8   INPUT2
NAND  SIG9   SIG8   SIG1Ø
NOR  SIG1Ø   INPUT3 SIG8   SIG2
NOT  SIG11 INPUT4
NOR  SIG12   SIG1Ø SIG11
NAND  SIG13   INPUT4   SIG12
NAND  OUT2  SIG9   SIG12   SIG13;
///
///  Control command
///
$  MONITOR OUT1 OUT2
$  RUN_UNTIL/3300
$  END
```

(a)

```
(RESULTS OF CIREX)
VDD 3Ø
GND 4Ø
MOS1 ND  2 2 3Ø
MOS2 ND  3 3 3Ø
MOS3 ND  5 5 3Ø
MOS4 ND  6 6 3Ø
MOS5 ND  7 7 3Ø
MOS6 ND  18 18 3Ø
MOS7 ND  8 8 3Ø
MOS8 ND  9 9 3Ø
MOS9 ND  10 1Ø 3Ø
MOS10  ND  11 11 3Ø
MOS11  ND  12 12 3Ø
MOS12  ND  13 13 3Ø
MOS14  ND  19 19 3Ø
MOS15  NE  4Ø 12
MOS16  NE  4Ø 23
MOS17  NE  5 8 4
MOS18  NE  4Ø 34
MOS19  NE  4Ø 95
MOS20  NE  4Ø 56
MOS21  NE  45 57
MOS22  NE  4Ø 1Ø 45
MOS23  NE  46 6  18
MOS24  NE  40 7  46
MOS25  NE  4Ø 15 8
MOS26  NE  4Ø 8  47
MOS27  NE  47 1Ø 9
MOS28  NE  4Ø 16 1Ø
MOS29  NE  4Ø 8  1Ø
MOS30  NE  4Ø 2  1Ø
MOS31  NE  4Ø 17 11
MOS32  NE  4Ø 11 12
MOS33  NE  4Ø 10 12
MOS34  NE  48 17 13
MOS35  NE  4Ø 12 48
MOS36  NE  49 12 19
MOS37  NE  56 13 49
MOS38  NE  4Ø 9  56
END
```

(b)

```
(* EXTRACTED LOGIC NET_LIST *)
D1 NOT  net2 net1
D2 NOT  net3 net2
D3 NOR  net5 net4 net9
D4 AND  net4 net8 net3
D5 NOT  net6 net5
D6 NAND net7 net5 net1Ø
D7 NAND net18 net6 net7
D8 NOT  net8 net15
D9 NAND net9 net1Ø net8
D10  NOR  net1Ø net16  net8 net2
D11  NOT  net11 net17
D12  NOR  net12 net11  net1Ø
D13  NAND net13  net17  net12
D14  NAND net19  net12 net13 net9
```

(c)

Net matched list

Reference	Retrieved
INPUT1	net1
SIG2	net2
SIG3	net3
SIG8	net8
SIG4	net4
SIG9	net9
SIG5	net5
SIG6	net6
SIG7	net7
OUT1	net18
SIG1Ø	net1Ø
INPUT2	net15
INPUT3	net16
INPUT4	net17
SIG11	net11
SIG12	net12
SIG13	net13
OUT2	net19

(d)

Device matched list

Reference	Retrieved
D1	D1
D2	D2
D3	D4
D4	D3
D5	D5
D6	D7
D7	D6
D8	D8
D9	D9
D1Ø	D1Ø
D11	D11
D12	D12
D13	D13
D14	D14

(e)

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - -
- - - - - - -   ERROR LISTS   - - - - - - - -
- - - - - - - - - - - - - - - - - - - - - - - - - - - -

ERROR : Reference SIG6 net, retrieved net6 are error matched.
ERROR : Reference SIG7 net, retrieved net7 are error matched.
ERROR : Reference OUT1net, retrieved net18 are error matched.
Reference D6 is NOR, but retrieved D7 is NAND.
    - - > correct NOR.
```

(f)

그림11. (a) Logic simulator의 입력
(b) Circuit extraction의 결과
(c) (b)로 부터 논리추출 결과
(d), (e) 비교에 의한 검증결과
(f) 출력된 오류표

Fig. 11. (a) Input file of the logic simulator.
(b) Result of circuit extraction.
(c) Result of logic extraction from (b).
(d) (e) Results of verification by comparison with error.
(f) Output error list.

위치)를 message로 내보내 준다.

이 program은 C 언어로 구현되었으며 schematic 발생에서 logic simulator의 입력 file을 생성하는 부분이 약 4,000줄 논리추출 부분이 약 1,000줄, 논리비교에 의한 검증 부분이 약 3,000줄, 그리고 논리회로를

그리는 부분이 약 700줄 정도되며 MV 10000computer 에서 run 되고 있다.

參 考 文 献

[1] A. Kishimoto et al., *An Interconnection Check Algorithm for Mask Pattern,* in Proc. ISCAS, pp. 669-672, 1979.

[2] K. Sato et al., *A Method for Connectivity Checking for MOS Circuit Considering Logical Equivalence,* in Proc. ISCAS, pp. 1285-1288, 1985.

[3] T. Watanabe et al., "A new automatic logic interconnection verificiation system for VLSI design," *IEEE Trans. on CAD,* vol. CAD-2, pp. 70-81, April 1983.

[4] Y. Wong, *Hierarchical Circuit Verification,* in Proc. 22nd DAC, pp.695-701, 1985.

[5] M. Takashima et al., *Program for Verifying Circuit Connectivity of MOS/LSI Mask Artwork,* in Proc. 19th DAC, pp. 544-550, 1982.

[6] E. Barke, "A new comparison algorithm for layout verifivation of integrated circuits," *IEEE Trans. on CAD,* vol. CAD-3, no. 2, pp. 135-141, April 1984.

[7] S.S. Kim, *Circuit Extraction from MOS/LSI Mask Layout,* M.S. thesis KAIST, Korea, 1986.

[8] R.M. Apte et al., *Logic Function Extraction for NMOS Circuits,* ICCC Proc. pp. 324-327, 1982.

[9] W.L. Engl et al., "Theory of multiplace graph," *IEEE Trans. on circuit syst.,* vol. CAS-22, pp. 2-8, 1975.

[10] T.W. Kweon, *Gate and Functional Level Logic Simulation with 8-state Logic Signal Model,* M.S. thesis KAIST, Korea, 1983.

[11] N. Kubo et al., *A Fast Algorithm for Testing Graph Isomorphism,* in Proc. ISCAS, pp. 641-644, 1979.

[12] Aho, Hopcroft and Ullman, *The Design and Analysis of Computer Algorithm,* Addison Wesley, 1976.