論文93-30A-2-10

상위단계 설계 검증을 위한 논리/타이밍 추출 시스템의 설계

(Design of A Logic/Timing Extraction System for Higher-level Design Verification)

李 容 在*, 文 寅 鎬**, 黃 善 泳**

(Yong Jae Lee, In Ho Moon and Sun Young Hwang)

要 約

본 논문은 트랜지스터 단계 회로 기술 언어로 부터의 기술 독립적인 논리 및 타이밍 추출 시스템에 대하여 설명한다. 추출하고자 하는 게이트 및 기능 블록의 구조 기술을 테크노리지 화일의 형태로 제공함으로써 회로 구현 기술에 독립적인 회로 추출이 가능하도록 하였다. 구현된 시스템은 cell-based 설계에서 추출하고자 하는 셀만을 기술함으로써 효과적으로 사용될 수 있다. 게이트의 연결 지연을 포함한 선형 RC 모델을 이용하여 타이밍 추출을 하도록 하였다. 실험 결과 논리 회로에 있어서 계산된 지연 시간은 SPICE와 비교하여 10% 내의 오차를 가진다. 논리 추출을 통한 상위 단계 기술을 통하여 설계 시간의 감소를 얻을 수 있다.

Abstract

This paper describes the design of a technology-independent logic, function, and timing extraction system from SPICE-like network descriptions. Technology-independent extraction mechanism is provided in the form of technology files containing the rules for constructing logic gates and functional blocks. The designed system can be more effectively used in cell-based design by describing the cells to be extracted. Timing extraction is performed by using a linear RC gate delay model which takes interconnection delay into account. Experimental results show that estimated delay is within 10 percents for logic gate circuits when compared with SPICE. Through higher-level design descriptions obtained by extraction, design cycles can be considerably reduced.

I. 서 론

최근 반도체 기술의 발달로 단일 칩 내에 집적할 수 있는 회로 소자의 수가 크게 증가함에 따라 VLSI 회로의 설계와 검증이 어려워지게 되었으며, 회로 설계에 있어서의 비용과 시간을 줄이기 위해서는 정확하고도 빠른 설계 오류의 검증이 필수적이다. 설계된 레이아웃의 검증은 크게 구조 검증(structural verification)과 행위 검증(behavioral verification)으로 나눌 수 있다. 회로의 행위 검증은 function 검증과 타이밍 검증으로 구분될 수 있으며, 회로 시뮬레이터나 타이밍 검증 시스템을 이용하여 회로의

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

接受日字: 1992年 1月 7日

function과 회로가 주어진 시간 안에 동작하는지를 검증할 수 있다. 구조 검증은 레이아웃의 위상 구조의 검증이 주된 목적으로 DRC(Design Rule Checking) 시스템[3]과 netlist comparison 시스템[2,6,13,15,18]이 포함될 수 있다.

논리 설계 과정에서 정확히 고려될 수 없는 연결선에 의한 지연과 같은 정보는 레이아웃 합성이 완료된 후에야 알 수가 있다. 즉 설계된 회로의 실제 동작은 레이아웃으로 부터 추출되어지며 추출된 정보로 부터 회로의 동작을 모델링하여 논리 설계된 회로와 동작을 비교한다. 따라서 ACE[7], EXCL[10], Magic's circuit extractor[16], DEC's hierarchical circuit extractor[17]와 같은 많은 회로 추출기가 개발되었다. 회로 추출기는 설계된 레이아웃으로 부터 트랜지스터 및 회로 소자와 이들의 연결 관계를 나타내는 노드를 추출하며 회로의 시간 특성을 얻기 위하여 트랜지스터의 크기와 연결선에 분포하는 기생 저항 및 기생 커패시턴스의 값을 추출한다. 회로 추출 결과는 트랜지스터 단계 회로 기술 언어로 기술되어지며 이는 회로 시뮬레이터인 SPICE[11]의 입력으로 이용될 수 있으나 트랜지스터 단계 회로 기술은 VLSI 회로의 경우 많은 자료의 양으로 인하여 회로 시뮬레이션에 많은 시간이 걸릴 뿐 아니라 회로를 이해하기 어려운 단점이 있다.

이와 같은 단점을 보완하기 위하여 많은 연구가 진행되어 왔으며, 시뮬레이션 시간을 줄이기 위하여 LOGEX[4]나 BLEX[9]와 같은 트랜지스터 단계로 부터의 논리 추출 시스템이 개발되었다. LOGEX는 트랜지스터 단계 회로 기술로 부터 알고리드믹한 방법으로 논리를 추출 하지만 반면에 다양한 회로 구현 기술에 적용될 수 없는 단점을 가진다. BLEX는 임의의 논리 블럭을 레이아웃으로 부터 추출된 셀 정보로 부터 추출하지만 논리 블럭의 행위는 추출할 수 없다는 단점이 있다. 본 논문에서의 논리 추출 메카니즘은 추출하고자 하는 논리 소자의 구조를 사용자가 구성 법칙의 형태로 제공하게 함으로써 회로 구현 기술에 관계없이 논리 추출을 가능하게 하였다. 즉 추출하고자 하는 회로의 구조에 관계 없이 추출하고자 하는 기본 cell의 정보만으로 회로 소자의 연결 관계를 추출할 수 있으므로 CMOS 및 dynamic CMOS, domino CMOS등과 같이 다양한 구조를 가지는 회로에 대하여 논리를 추출할 수 있으며 특히 cell-based design시에 효과적으로 회로를 추출하도록 하였다. 또한 논리 소자의 동작을 추출하기 위하여 function extraction 알고리듬을 사용자가 주는 게이트 구성 법칙에 적용함으로써 레이아웃으로 부터 추출한 회로를 논리 단계에서 VHDL로 기술할 수 있도록 하였다. 따라서 전체적인 추출 과정을 거친 후의 레이아웃은 VHDL의 구조 기술및 동작 기술을 자동적으로 생성하여 논리 단계의 시뮬레이션을 함으로써 시뮬레이션 시간을 크게 줄일 수 있고 논리 회로 설계자에게 생성된 레이아웃의 동작의 back-annotation이 가능하도록 하였다.

레이아웃에서 추출된 트랜지스터 및 RC 파라미터로 부터 회로의 시간 특성을 얻기 위하여 linear RC 모델링을 이용한 많은 타이밍 분석 툴이 개발되었다.[5,8,12,14] 최근 칩의 크기의 증가와 트랜지스터의 크기의 감소로 인하여 회로 소자의 RC 보다 연결선에 존재하는 RC 파라미터가 더 중요하게 되었다. 따라서 이를 고려하지 않은 타이밍 분석 툴은 많은 오류를 가지게 된며 게이트 지연 시간 계산에 있어서 연결선의 지연을 고려한 게이트이 지연 시간 계산이 불가피하게 되었다. 본 논문에서는 논리 추출 시에 연결선의 지연을 고려한 타이밍 추출을 하도록 하였다. 연결선에 분포하는 기생 저항 및 기생 커패시턴스는 distributed RC 모델의 사용시 가장 정확한 예측이 가능하나 이의 해석에는 많은 시간이 걸리므로 이를 테이블의 형태로 단순화하여 계산한다.

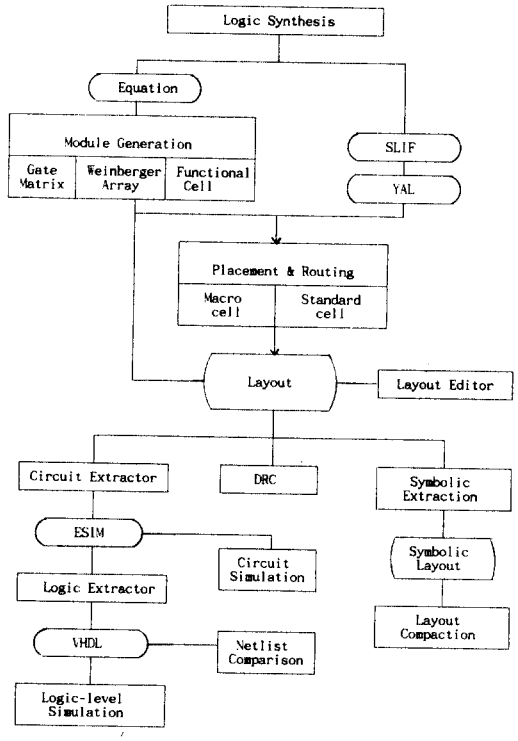


그림 1. 서강 레이아웃 디자인 및 검증 시스템.
Fig. 1. Sogang layout design and verification system.

그림 1 에 SOLID (SOgang Layout verIfication and Design) 시스템의 전체적인 구성을 보였다. 시스템의 전체적인 구성은 레이아웃의 설계 규칙을 검사하는 DRC 시스템과 레이아웃의 검증을 위한 회로 및 논리/타이밍 추출 시스템 및 레이아웃으로 부터 추출한 회로를 설계된 회로와 비교하기 위한 회로 비교 시스템으로 구성되며, technology migration을 위한 symbolic extractor와 layout compaction 시스템으로 구성된다.

Ⅱ. 논리 추출

논리 추출 과정에서는 트랜지스터와 이들의 연결도로 부터 해당 논리를 추출한다. 논리 추출은 추출하고자 하는 회로의 구조를 technology 화일에 기술하고 이를 읽어들여 해당 논리 회로를 추출하므로 회로 구현 기술에 관계 없이 다양한 구조의 논리 회로에 대하여 논리 추출을 행할 수 있어 본 논리 추출기는 nMOS, CMOS, dynamic CMOS, domino CMOS등의 구현 논리에 관계 없이 논리 추출이 가능하다. 또한 논리 소자의 동작에 관한 정보는 사용자가 주는 게이트 구성 법칙에 function extraction 알고리듬을 적용하여 얻도록 하였다. 사용자가 게이트의 구성 법칙을 주는 것은 현재의 회로 설계 방식이 cell library 에 바탕을 두어 이루어지기 때문에 사용자에게 부담을 주지 않는다.

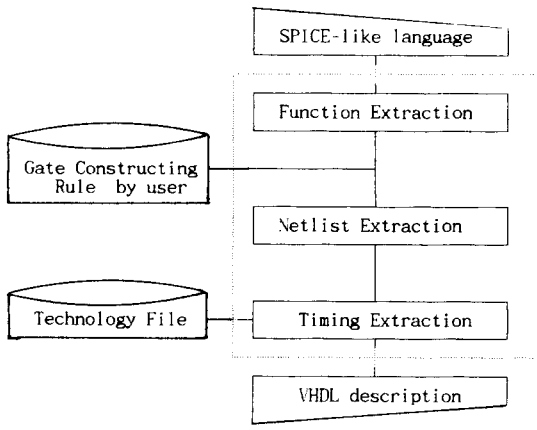


그림 2. 논리/타이밍 추출기.
Fig. 2. Logic/timing extractor.

논리 추출 과정은 세 단계로 나눌 수 있다. 첫 단계는 사용자가 기술한 게이트 구성 법칙을 컴파일하고 입력을 읽어 들여 내부 자료를 구성하는 과정이며, 두 번째 단계는 사용자가 기술한 게이트 구성법칙으로 부터 AND, OR 트리를 구성하고 이로 부터 게이트의 행위를 추출하는 function extraction 과정이다. 마지막 단계는 입력으로 부터 게이트 및 게이트의 연결 관계를 추출하는 netlist extraction 을 수행한다. 즉 VHDL의 behavior part는 function extraction 알고리듬을 적용하여 얻을 수 있고 structural part는 netlist extraction 알고리듬을 적용하여 전체적인 회로를 추출한다. 그림 2에 논리 및 타이밍 추출기의 구성을 보였다. 추출된 결과는 VHDL로 출력되어져 VHDL 시뮬레이터를 이용하여 회로의 동작을 확인할 수 있으며 logic synthesis 툴로 레이아웃 정보를 back-annotate 할 수 있다. 또한 설계된 회로와 레이아웃으로 부터 추출한 회로를 비교할 수 있다.

1. 논리 추출기 입력 및 게이트 구성 법칙

1) 논리 추출기 입력 및 파싱

논리 추출기의 입력은 트랜지스터의 타입, 트랜지스터 채널의 폭과 길이 그리고 각 노드에 분포하는 저항과 커패시턴스를 입력으로 한다. 그림 3에 논리 추출기 입력의 예를 보였다. 트랜지스터의 타입은 PMOS 의 경우 "P" 로 시작하고 NMOS 의 경우 "N" 으로 시작하도록 정의 하였다. 다음에 오는 identifier들은 각기 트랜지스터의 gate, source, drain 터미날을 나타내며, source와 drain 터미날의 구분을 두지 않아 레이아웃으로 부터의 회로 추출 결과가 바로 논리 추출기의 입력이 되도록 하였다. 트랜지스터의 크기를 기술하기 위하여 CW(Channel Width)와 CL(Channel Length)를 주게 된다. 트랜지스터의 연결 관계 및 크기를 모두 기술하고 나면, 게이트 지연 시간 계산을 위하여 각 노드에 분포하는 기생 저항 및 커패시턴스를 기술한다.

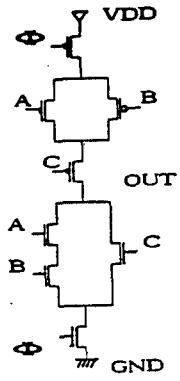
```
BEGIN_COMP
P1 IN1 VDD OUT CW=2 CL=4;
P2 IN2 VDD OUT CW=2 CL=4;
N1 IN1 OUT ND1 CW=2 CL=4;
N2 IN2 ND1 GND CW=2 CL=4;
BEGIN_PARA
OUT 20 0.6;
END
```

그림 3. 논리 추출기 입력 구성
Fig. 3. Input of logic extractor.

논리 추출기 파서는 그림 3와 같은 트랜지스터 단계 기술 입력 언어를 받아들여 트랜지스터를 element로 이들의 연결 관계를 나타내는 node로 구성되는 전체적인 netlist를 구성한다.

2)게이트 구성 법칙

기술 독립적인 논리 추출을 하기 위해서는 추출하고자 하는 게이트의 구조을 알아야 하며 이는 사용자가 기술하거나 레이아웃으로 부터 회로 추출을 통하여 자동적으로 생성할 수 있다. 논리 추출기는 추출하고자 하는 게이트의 게이트 구성 법칙을 읽어들여 게이트 구성 테이블을 구성한다. 논리 추출 과정은 게이트 구성 법칙으로 부터 구성된 게이트 구성 테이블에 function extraction 알고리듬을 적용하여 회로의 행위 부분을 추출하는 과정과 전체적인 논리회로의 연결 관계를 알아내기 위하여 게이트 구성 테이블과 논리 추출하고자하는 회로의 pattern matching에 의해 회로의 구조 부분을 추출하는 과정으로 나뉜다. 그림 4(a)에 Clocked CMOS 회로의 구조를 나타내었으며 이의 게이트 구성 법칙을 그림 4(b)에 나타내었다. 게이트 구성 법칙의 head 부분에서는 추출하고자 하는 회로의 입/출력과 양 방향성 port를 기술하게 되며 논리 추출에서의 입/출력 관계는 이를 바탕으로 하여 추출 된다. 게이트 구성 법칙의 body 부분에서는 트랜지스터의 연결 관계를 기술하며 이로 부터 추출하고자 하는 게이트의 구조를 알 수 있다.



(a)

```
.clk_and_or
input A B C
output OUT
P1 Ø VDD ND1;
P2 A ND1 ND2;
P3 B ND1 ND2;
P4 C ND2 OUT;
N1 Ø GND ND3;
N2 A ND3 ND4;
N3 B ND4 OUT;
N4 C ND3 OUT;
```

(b)

그림 4. (a) Clocked CMOS 게이트 구조
(b) 게이트 (a) 에 대한 법칙 기술

Fig. 4. (a) Gate structure of clocked CMOS.
(b) Rule description of gate (a).

2. Function Extractor

1) AND-OR 트리의 구성

논리 추출을 하고자 하는 회로의 행위 부분 기술을 얻기 위하여 논리 추출 알고리듬을 전체 회로에 적용하는 것보다는 전체 회로를 구성하는 게이트 구성 법칙에 대하여 논리 추출 알고리듬을 적용하여 행위 추출을 하는 것이 효율적이다. 게이트의 행위를 추출하기 위해서 게이트 구성 테이블로 부터 트랜지스터의 직, 병렬 관계를 나타내는 AND-OR 트리를 구성하게 된다. AND-OR 트리를 구성하기 위해서는 그림 5에서와 같이 직렬로 연결된 트랜지스터를 AND 연산을 나타내는 "*"로 병렬로 연결된 트랜지스터를 OR 연산을 나타내는 "+" 변환하는 네 가지 법칙의 예를 보였다.

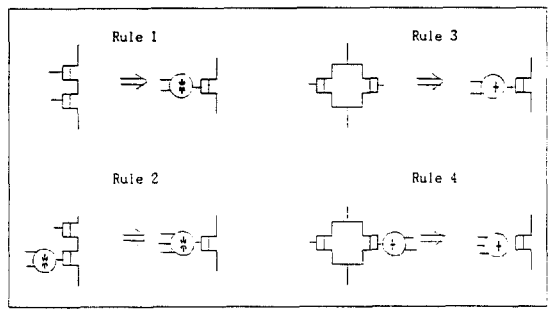


그림 5. AND-OR 트리 구성을 위한 법칙.

Fig. 5. Rule for constructing AND-OR tree.

AND-OR 트리의 구성은 PMOS와 NMOS 트랜지스터에 대하여 개별적으로 적용되며 PMOS 트랜지스터에 대하여 series/parallel reduction 에 의하여 생성된 트리를 p-part 트리 구조라 하고 NMOS 트랜지스터에 적용하여 생성된 트리를 n-part 트리 구조라고 한다. 그림 6에 그림 4(a)게이트의 p-part 트리 구조를 구성하기 위하여 적용되어지는 법칙과 그 과정을 보였다. 그림 4(a)의 게이트에 대하여 구성된 p-part, n-part의 AND-OR 트리 구조를 그림 7에 나타내었다.

2) AND-OR 트리로 부터의 행위 추출

AND-OR 트리로 부터 논리 동작을 추출하기 위하여 먼저 해당 게이트의 p-part AND-OR 트리와 n-part AND-OR 트리의 dual 관계가 성립하는 지를 조사하게 된다. P-part AND-OR 트리의 AND 노드를 OR 노드로 OR 노드를 AND 노드로 바꾸었을 때 p-part AND-OR 트리 구조와 n-part AND-OR 트리 구조가 같을 경우에 이 회로의 n-part 와 p-part 트리 구조는 dual 하다고 한다. 이와같이 n-

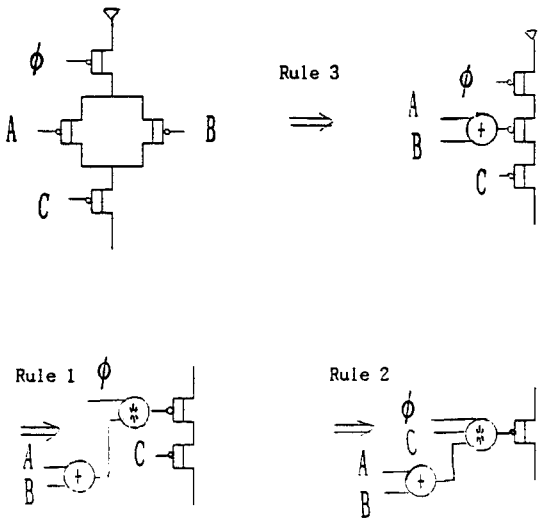


그림 6. AND-OR 트리 구성을 위한 법칙 적용 과정
Fig. 6. Rule application process for constructing AND-OR tree.

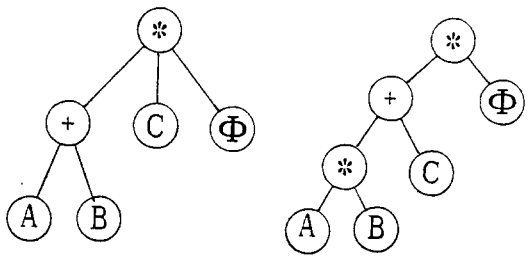


그림 7. (a) 그림 4(a)에 대한 p-part AND-OR 트리
(b) 그림 4(a)에 대한 n-part AND-OR 트리
Fig. 7. (a) P-part AND-OR tree for Fig. 4(a).
(b) N-part AND-OR tree for Fig. 4(b).

part와 p-part 트리 구조가 dual 할 경우의 논리 추출은 트리의 직, 병렬 관계로 부터 쉽게 알아낼 수가 있다. 반면에 dynamic 게이트와 같은 경우에는 clock 입력으로 인하여 p-part 와 n-part 트리 구조의 dual 관계가성립하지 않는다. 이와 같이 n-part 와 p-part 트리 구조의 dual 관계가 성립하지 않는 경우에는 트리 구조의 직, 병렬 관계로 부터 논리 추출할 수가 없다 . 따라서 이와 같은 경우에는 clock 을 입력으로 하는 트랜지스터를 제외하면 p-part 트리 구조와 n-part 트리 구조의 dual 관계가 성립하므로 각각의 트리 구조로 부터 clock 입력에 해당하는 트랜지스터를 제외하고 해당 논리를 추출할 수 있도록 하는 clock separation 과정을 먼저 적용한 후 논리 추출 알고리듬을 적용한다. 그림 8에 그림 4의 p-part, n-part AND-OR 트리 구조에 대하여 clock separation 알고리듬을 적용하는 과정을 보였다. Clock separation 과정을 적용하여 트리 구조를 변경한 후 root 노드를 clock 입력에 해당하지 않는 노드로 한 레벨을 낮추어 이를 새로운 root 노드라 하면 p-part와 n-part AND-OR 트리 구조는 새로운 root 노드에 대하여 dual 관계에 있음을 알 수 있다.

3. Netlist Extractor

논리회로의 연결 관계를 추출하기 위한 전체적인 알고리듬 기술을 그림 9에 보였다. 논리 회로의 연결 관계 추출을 위한 알고리듬은 크게 세 단계로 구분할 수 있다. 첫 단계는 읽어 들인 입력 회로를 단위 게이트로 grouping 하는 회로 분할 과정이다. 두 번째 단계는 사용자로 부터 기술된 게이트의 구조와 추출하고자 하는 회로의 구조를 비교하는 그래프 패턴 매

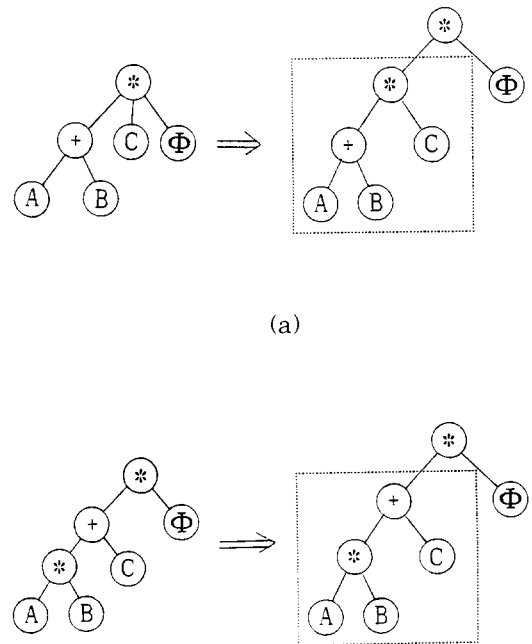


그림 8. (a) P-part 트리에 대한 clock separation
(b) N-part 트리에 대한 clock separation
Fig. 8. (a) Clock separation for p-part tree.
(b) Clock separation for n-part tree.

칭 과정이다.

```
/*
* G = (V, A) : Circuit graph
* T = {(V1, A1), --, (Vi, Ai)} : Rule table
* Circuit_partition() partitions G into subcircuit s
*/

procedure        Global_netlist_extract(G)

begin
 Circuit_partition(G);
  for each subcircuit s in G do
   for each rule structure t in T do
    if structure of s and t match
     begin
      Create logic element;
      Set input/output port;
      Calculate gate delay;
      Report the logic description;
     end;
end;
```

그림 9. Netlist extraction 알고리듬
Fig. 9. Netlisrt extraction algorithm.

세 번째 단계는 회로가 매치되었을 때 새로운 논리 게이트를 생성하고 게이트의 입/출력을 게이트 구조 법칙의 입/출력 노드의 정의에 따라 setting 하는 과정을 말한다. 또한 이 과정에서 생성된 논리 게이트의 시간 정보를 얻기 위하여 게이트 지연 모델에 따라 지연 시간을 계산한다.

1) 회로 분할

트랜지스터 단계 회로 기술로 부터 논리 소자와 이들간의 입/출력 관계를 추출하기 위해서는 먼저 회로 분할 과정을 거치게 된다. 회로 분할 과정은 enhancement 트랜지스터를 driver 트랜지스터와 pass 트랜지스터로 나누는 일이다. Driver 트랜지스터들은 multiple 입력 single 출력의 combimational 논리 블록으로 grouping 되며, pass 트랜지스터는 pass 트랜지스터 논리 블록으로 grouping 된다. 따라서 전체 회로는 출력 노드와 VDD 혹은 GND 노드사이의 트랜지스터로 구성되는 subcircuit으로 분할 된다. 출력 노드는 PMOS 와 NMOS 의 드레인이나 소스 터미날이 동시에 연결된 노드를 말하며, 그림 10에 출력 노드 정보를 이용한 회로 분할의 예를 보였다.

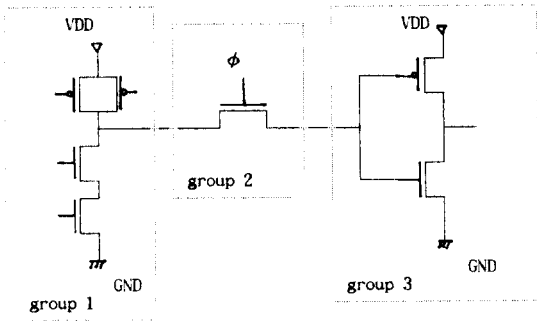


그림 10. 회로 분할의 예.
Fig. 10. Example of circuit partitioning.

2) 패턴 매칭 (Pattern Matching)

논리 소자의 연결 정보의 추출을 위한 pattern matching은 추출하고자 하는 입력 회로의 subcircuit 구조와 게이트 구성 법칙으로 부터 만들어진 구조를 비교하는 과정을 말한다. 그림 11에 회로 매치를 위한 게이트 구성 법칙으로 부터 구성된 테이블의 구조를 보였다. 게이트 구조 테이블은 추출하고자 하는 게이트의 이름을 rule_name으로 가지고 있으며 게이트의 입/출력 노드에 관한 정보를 가지고 있다. 또한 게이트의 구조를 나타내는 element와 node의 자료 구조를 가지고 있어 이를 매치시킨다. 논리 게이트를 추출하기 위해서는 해당 게이트 구성 테이블의 모든 element와 node의 매치를 나타내는 all_el_match와 all_nd_match가 set되어야 한다. Pattern matching 알고리듬은 element 와 node 매치 알고리듬으로 구성되어 있으며 두 회로의 트랜지스터의 타입과 이들의 연결 관계를 나타내는 노드의 구조를 비교하여 모든 element와 node가 매치될 경우에만 새로운 논리 소자를 생성하고 이의 입/출력 관계를 추출하도록 하였다.

char	*Rule_name	Elem	*elem
Node	**Input	Node	*node
Node	**Output	tint	all_el_match
Node	**Biput	int	all_nd_match

그림 11. 게이트 구성 법칙으로 부터 구성된 테이블
Fig. 11. Table constructed from gateconstructing rule.

회로 match를 위하여 게이트 구성 법칙으로 부터 구성된 element와 node의 자료 구조는 매치된 정보를 유지하기 위하여 match, matched-elem,

matched-node의 field를 가지며 매치가 이루어져 새로운 논리 소자를 생성한 후 또는 매치가 되지 않았을 경우에 다음 매치를 위하여 clear한다. Pattern matching을 위한 알고리듬은 recursive한 element와 node 매치 알고 리듬을 적용한다.

3) 논리 게이트 생성 및 입, 출력 노드 추출

두 회로의 구조가 일치하는 경우 새로운 논리 element를 생성하고 이를 hash-table에 유지하게 된다. 또한 논리 element의 입/출력 관계의 추출은 사용자가 정의한 입/출력 정보에 따라 추출된다.

Ⅲ. 게이트 단계 지연 모델

칩의 크기가 커지고 트랜지스터의 크기가 작아짐에 따라 게이트 자체의 지연 시간보다 연결선에 의한 지연 시간이 전체 회로의 지연 시간에 있어서 많은 부분을 차지하게 되었다. 따라서 게이트 단계 지연 시간을 계산하는데 있어서 이러한 연결선에 의한 지연 시간을 고려한 지연 모델이 필요하게 되었다. 회로에서 연결선이 긴 경우 연결선의 기생 저항 및 기생 커패시턴스가 트랜지스터의 등가 저항과 트랜지스터의 입력 커패시턴스 보다 크기 때문에 이 경우에는 연결선에 의한 지연이 논리회로의 지연에 있어서 더 중요하게 된다. 연결선 전체에 분포하는 저항 및 커패시턴스는 distributed RC 회로 모델로 해석되며, 이의 해석에는 많은 시간이 걸리므로 Distributed RC 회로 모델은 나 T모델과 같은 lumped model로 근사하여 지연 시간을 계산한다. 본 논문에서의 타이밍 추출은 계단파 입력에 대한 출력 변화를 테이블의 형태로 구성하고 이로 부터 게이트의 지연 시간을 계산하므로 빠른 타이밍 계산을 할 수 있다. [1]

1 연결선의 지연을 고려한 게이트 지연 모델

연결선의 지연을 고려한 게이트의 지연 시간은 다음과 같은 세 가지 지연요소로 분리될 수 있다.

(1) Intrinsic delay:게이트 자체의 고유 지연 시간

$T_{intrinsic}=R_{tr}C_g$

R_{tr}:Transistor on resistance

C_g:Gate capacitance

(2) Transition delay:게이트 출력단에 걸린 로딩에 의한 지연 시간

(3) Interconnection delay:연결선에 존재하는 파라미터에 의한 지연 시간

표1에 lumped RC 회로와 distributed RC회로에 계단 입력이 주어졌을 때의 출력단 변화에 필요한 RC 시상수를 나타내었으며 게이트의 지연 시간 계산을 표1을 참조하여 게이크의 해당 지연 시간을 계산한다.

표 1. 분산 회로와 집중 회로의 계단파 응답

Table 1. Step response of distributed and lumped RC circuits.

Output range	Time elapsed Distributed RC	Time elapsed Lumped RC
0 to 50%	0.4	0.7 RC
0 to 90%	RC 1.0	2.3 RC
10 to 90%	RC 0.9 RC	2.2 RC

2 Timing Report in VHDL

논리 추출 과정에서 같은 타입을 가지는 게이트라 할 지라도 게이트의 fanout 로딩 커패시턴스가 다르고 연결선에 존재하는 지연 때문에 다른 시간 정보를 가지게 된다. 즉 한 게이트의 입/출력에 관한 정보나 논리 동작에 관한 정보는 같지만 시간 정보는 다르게 된다. 이와같이 instance-specific한 정보는 VHDL에서 제공하는 "generic statement"를 이용하여 해결할 수 있다. 즉 같은 타입의 같은 동작을 하는 게이트가 다른 시간 정보를 가지고 동작할 때 이는 "generic statement"를 이용하여 추출한 시간 정보를 리포트 할 수 있다. 그림 12에 AND 게이트에 있어서의 rise, fall 시간과 load 커패에서 보듯이 두개의 AND 게이트 E1의 rise 지연은 2ns 인데 E2의 rise 지연은 3ns로 다름을 알 수 있다.

```
ARCHITECTURE testarch OF test IS
COMPONENT AND2
GENERIC(rise, fall: TIME; load: INTEGER);
PORT(a,b:IN t_wlogic;
c:OUT t_wlogic);
END COMPONENT;
variable ina, inb, out1, inc, ind, out2;
BEGIN
E1: AND2 GENERIC MAP(2 ns, 3 ns, 3)
PORT MAP(ina, inb, out1);
E2: AND2 GENERIC MAP(3 ns, 4 ns, 5)
PORT MAP(inc, ind, out2);
END test_arch;
```

그림 12. "generic statement"의 사용 예

Fig. 12. Use of "generic statement".

3. Subcircuit Path Analysis

일반적으로 게이트의 지연 시간을 모델링하는 것은 게이트의 지연 시간이 입력 패턴이나 입력 기울기에 따라 변하기 때문에 어렵다. 게이트의 rise지연 시간을 계산하기 위하여 파워 라인으로 부터 출력단까지의 path는 입력 패턴에 따라 구동되는 트랜지스터에 의해 구성되므로 입력 패턴에 따라 다른 게이트 지연 시간을 가질 수 있다. 따라서 subcircuit path analysis 알고리듬을 적용하여 여러 path중에서 min/max 지연 시간을 계산할 수 있다. 그림 13에 subcircuit path analysis 알고리듬을 적용하여 VDD 노드로 부터 게이트의 출력 노드까지의 path를 찾아서 구성된 path list를 보였으며 구동되는 트랜지스터의 min/max on-resistance를 path list에서 찾아서 계산 한다.

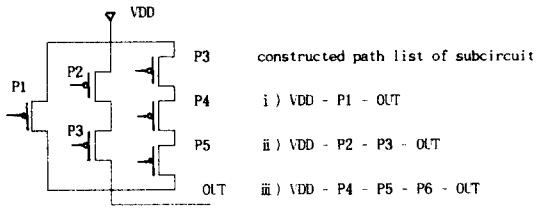


그림 13. Subcircuit에서의 path list의 구성
Fig. 13. Constructing path list in subcircuit.

Ⅳ. 실험 결과

본 논문에서의 논리/타이밍 추출 시스템은 SUN-4 workstation상에서 C 언어로 구현 및 실험 되었다. 논리 추출 시스템의 검증을 위하여 트랜지스터 단계 언어로 기술된 여러 회로로 부터 추출하고자 하는 법칙을 주어 해당 논리를 추출하도록 하였다. 표 2에 논리 추출하고자 하는 회로와 이의 논리 추출을 하는데 걸리는 수행 시간을 나타내었다.

표　2. 논리 추출기 수행 시간
Table 2. Run time of logic extractor.

(Sun 4)

Circuit	# trs	# cells	# gates	elapsed time
ex1	22	4	4	0.12(s)
shift reg.	32	2	16	0.14(s)
4-bit adder	272	7	82	1.5(s)
4-bit ALU	348	6	91	3.2(s)
32-bit ALU	2784	6	728	21.5(s)

논리 및 타이밍 추출 시스템인 TILEX(Technology Independent Logic/Timing Extraction System)의 지연 시간은 회로 시뮬레이터인 SPICE와 비교하여 일반적으로 10% 내의 오차를 보였다. 시뮬레이션 시간은 논리 단계로의 추출을 통하여 논리 단계 회로 시뮬레이션의 수행으로 SPICE와 비교하여 10^2-10^3배 빠르며 구현된 논리 및 타이밍 추출 시스템은 VLSI회로의 검증에 적합하다. 그림 14에 12개의 게이트로 구성된 full adder 셀에 대한 논리 회로를 나타내었으며 이 회로에 대한 rise, fall지연 시간을 SPICE와 비교하여 표 3에 나타내었다. 표 3에서 나타나는 오차는 RC모델링의 경우 비선형 소자인 트랜지스터를 선형 소자로 근사한데에 기인하며, 트랜지스터가 on/off되었을 때의 등가 저항이 입력 기울기에 따라 변하는데에도 기인한다. Full adder 셀에 대한 SPICE 시뮬레이션 시간은 90sec가 걸렸으며 반면에 논리 단계로의 추출을 통하여 논리 단계 시뮬레이션에 걸리는 시간은 논리 추출 시간 0.12sec와 논리 단계 시뮬레이션 시간을 합한 시간이 된다.

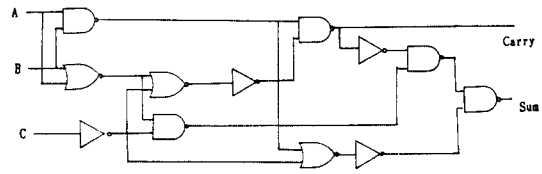


그림 14. Full adder 셀 logic diagram
Fig. 14. Full adder cell logic diagram.

표　3. SPICE와의 지연 시간 비교
Table 3. Comparison of delay with SPICE.

(ns)

	TILEX	SPICE	Td	%Td
Sum Rise	14.66	13.85	+0.81	+5.84
Sum Fall	18.72	16.71	+1.41	+8.44
Carry Rise	9.16	9.22	-0.06	-0.65
Carry Fall	8.16	7.68	+0.48	+6.25

그림 15에 레이아웃 검증을 위한 전체적인 환경을 보였다. Upper-left 윈도우는 회로 추출기의 입력이 되는 4-bit ALU 레이아웃을 나타낸다. 이로 부터 회로 및 논리/타이밍 추출을 하여 생성된 VHDL 기술은 upper-right 윈도우에서와 같이 논리 설계 시스템에 back-annotate되어 레이아웃 정보를 논리 설계자에게 레이아웃으로 부터 추출된 회로의 동작을 주며, 또한 lower-right 윈도우에서와 같이 VHDL

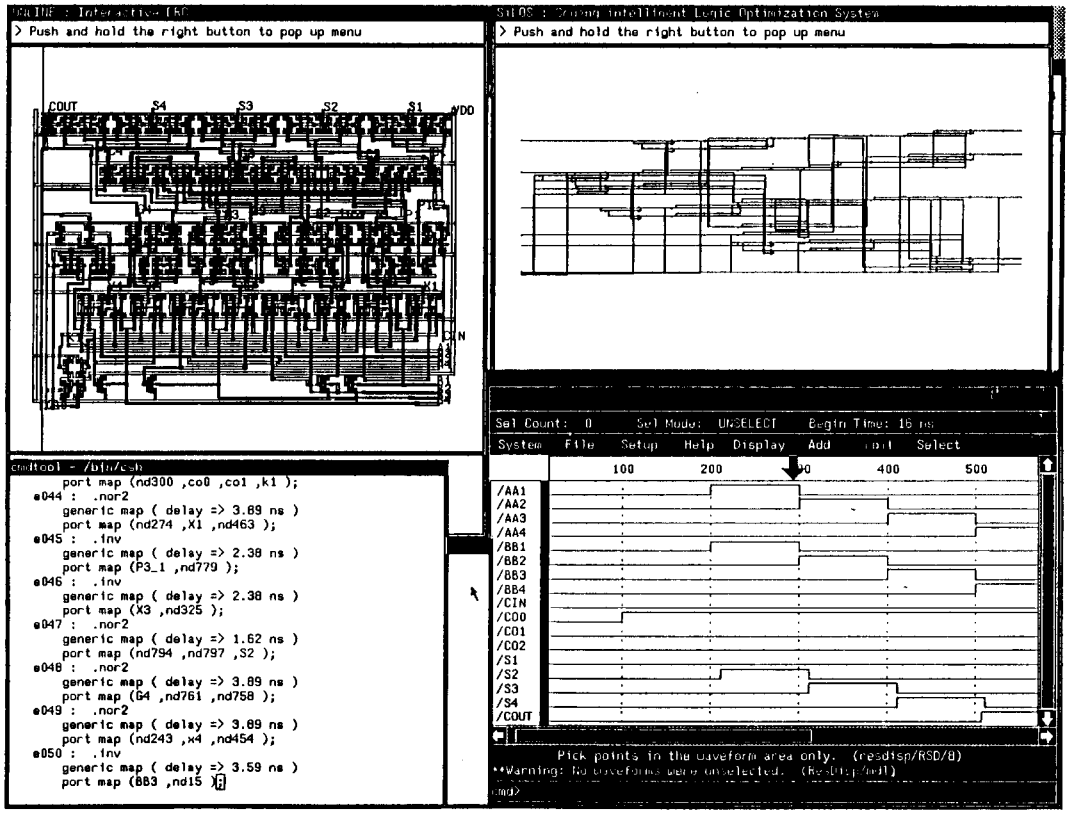


그림 15. TILEX 검증 환경
Fig. 15. TILEX layout verification environment.

시뮬레이터를 이용하여 논리 단계에서 회로를 시뮤레이션 함으로써 회로의 시뮬레이션 시간을 크게 줄일 수 있다.

V. 결론

본 논문에서는 상위 단계 설계 검증을 위한 논리/타이밍 추출기의 설계와 구현에 대하여 설명하였다. 논리 추출 과정은 추출하고자 하는 게이트의 구조를 사용자가 기술하고 이로부터 function extraction 알고리듬과 netlist extraction 알고리듬을 적용하여 논리 추출을 하기 때문에 회로 구현 기술에 관계 없이 다양한 구조를 가지는 회로에 대하여 논리 추출을 할 수 있다. 따라서 본 논리 추출기는 static CMOS, dynamic CMOS, domino CMOS 등 다양한 회로 구현 기술에 대하여 적용되어 질 수 있다.

본 논리/타이밍 추출 시스템에서는 최근 VLSI 회로에서 연결선의 지연이 많은 부분을 차지하게 됨에 따라 이러한 연결선의 지연을 고려하여 게이트의 지연 시간을 계산하도록 하였다. 연결선에 분포하는 파라미터는 집중 모델이 아닌 분산 모델을 이용하여 계산하게 되지만 이의 해것에는 많은 시간이 걸리므로 테이블을 이용한 간단한 RC 계산으로 해당 게이트의 연결 지연 시간까지 포함한 게이트의 전체 지연 시간을 계산하도록 하였다. 본 논문에서는 해당 게이트의 min/max 지연 시간을 계산하기 위하여 power 선으로 부터 출력 노드에 이르는 path를 찾아내는 알고리듬을 적용하여 구동되는 min/max 트랜지스터의 등가 저항을 계산하도록 하였다.

트랜지스터 단계 회로 기술로 부터 게이트의 동작과 시간 정보를 포함하는 논리 단계로의 추출을 통하여 회로의 검증을 논리 단계에서 함으로써 시뮬레이션 시간을 크게 줄이고 회로의 설계 주기를 줄이도록 하였다.

參考文獻

[1] H.B. Bakoglu, Circuits, Interconnections, and Pakaging for VLSI, Addison Wesley:Reading, Mass., 1990.

[2] E. Barke, "A Network Comparison Algorithm for Layout Verification of

integrated circuits," *IEEE Trans. CAD*, vol. CAD-3, April 1984, pp. 135-141.
[3] J. Berger and G. Mazare, "A Range Searching Algorithm Sub-sysytem Used to Perform Efficient VLSI Design Checks," in *Proc. ICCAD*, Nov. 1986, pp. 408-411.
[4] Boehner, "LOGEX:An Automatic Logic Extractor from Transistor Level for CMOS Technology," *in Proc.* of 25th DAC, June 1988, pp. 517-522.
[5] M. Brocco, P. Mccormick, "Macromodelling CMOS Circuits for Timing Simulation," *IEEE Trans. CAD*, vol. 7, no. 12, Dec. 1988, pp. 1237-1249.
[6] C. Ebeling and O. Zajicek, "Validating VLSI Circuit Layout by Wirelist Comparison," in *Proc. ICCAD*, Nov. 1983, pp. 172-173.
[7] A. Gupta, "ACE:A Circuit Extractor," *in Proc.* of 20th DAC, June 1983, pp. 721-725.
[8] M. A. Horowitz, "Timing Models for MOS Circuits," Ph. D Dissertation, Stanford Univ., Dec. 1983.
[9] F. Luellau, T. Hoepken, E. Barke, "A Technology Independent Block Extraction Algorithm," *in Proc.* of 21st DAC, June 1984, pp. 610-615.
[10] Steven P. McCormick, "EXCL: A Circuit Extractor for IC Designs," *in Proc.* of 21st DAC, June 1984, pp. 616-623.
[11] L. N. Nagel, "SPICE2:A Computer Program to Simulate Semiconductor Circuits," ERL Memo No. ERL-M520 Univ. of Cal., Berkeley, May 1975.
[12] J. K. Ousterhout, "A Switch-level Timing Verifier for Digital MOS VLSI," *IEEE Trans. CAD*, vol. CAD-4, no. 3, July 1985, pp. 336-349.
[13] M. M. Peter and D. S. Alexander, "A Logic-to-Logic Comparator for VLSI Layout Verification," *IEEE Trans. CAD*, vol. 7, no. 8, August 1988, pp. 897-907.
[14] T. Sakuria, "Approximation of Wiring Delay in MOSFET LSI," *IEEE Journal of Solid-State Circuits*, vol. Sc-18, Aug. 1983, pp. 418-426.
[15] R. L. Spickelmier and A. R. Newton, "Wombat: A New Netlist Comparison Algorith," *in Proc. ICCAD*, Nov. 1983, pp. 70-82.
[16] W. S. Scott and J. K. Ousterhout, "Magic:s Circuit Extractor," *in Proc.* of 22nd DAC, June 1985, pp. 286-292.
[17] G. M. Tarolli and W. J. Herman, "Hierarchical Circuit Extraction with Detailed Parasitic Capacitance," *in Proc.* of 20th DAC, June 1983, pp. 337-345.
[18] T. Watanabe, M. Endo, and N. Miyahara, "A New Automatic Logic Interconnection Verification System for VLSI design," *IEEE Trans. CAD*, vol. CAD-2, Jan. 1970, pp. 70-82.

著者紹介

黃 善 泳 (正會員)
1976年 2月 서울대학교 전자공학과 졸업(학사). 1978年 2月 한국과학원전기및전자공학과(공학 석사). 1986年 10月~미국 Stanford 대학 공학 박사학위 취득. 삼성 반도체 주식회사 연구원, Stanford 대학 CIS연구소 연구원 Fairchild Semiconductor 기술자문. 1989年 3月 ~ 현재 서강대학교 전자공하과 교수. 주관심 분야는 CAD시스템, Computer Architecture 및 Systems Design, VLSI 설계등임.

李 容 在 (準會員)
1967年 3月 19日生. 1990年 2月 서강대학교 전자공학과 졸업. 1992年 2月 서강대학교 전자공학 공학석사 취득. 현재 한국통신 전임연구원 주관심분야는 Design Verification, Simulation, CAD 시스템, Computer Archtecture 등임.

•

文 寅 鎬(正會員) 第 29 卷 A編 第 1 號 參照.