VLSI회로의 그래프 탐색 알고리즘에 관한 연구

김 현 호*, 장 중 식**, 이 천 희**
옥천대학*, 청주대학교**
yicheon@chongju.ac.kr

A Study on the Graph-Search Algorithm for VLSI Circuits

Kim, Hyeon Ho* Jang, Joong Sik** Yi, Cheon Hee**
Okchon College*, ChongJu University**

Abstract

최근 VLSI 디자인의 비용과 복잡성은 디자인 과정에서 필수조건이다. 소자 모델링과 수치적 방법은 spice와 같은 회로 시뮬레이터를 사용하여 얻을 수 있으며 simulated annealing과 같은 기법의 기술적인 장점은 많은 부분에서 응용된다. 이러한 기법들은 다량의 메모리 제조와 소규모 연구의 프로젝트까지 거의 모든 칩 디자인에 사용된다. 따라서 본 논문에서는 VLSI회로의 패턴 매칭에 관한 역트랙킹(backtracking) 깊이-우선 탐색을 할 수 있는 그래프 탐색 매칭 알고리즘을 제안하였다.

I. 서 론

최근 VLSI 디자인의 비용과 복잡성은 디자인 과정에서 필수조건인데 소자 모델링과 수치적 방법은 spice와 같은 회로 시뮬레이터를 사용하여 얻을 수 있다. 특히, 디자이너는 다품종 소량생산의 디자인이나 복잡한 디자인에 있어서 디자인 비용을 감소시키고 빠른 제작 싸이클 시간을 얻기 위해 많은 노력을 들인다 자동 배치와 배선, 논리 사양으로부터 게이트 합성, 그리고 테스트 패턴 생성 등은 디자이너의 노력으로 디자인의 비용과 시간을 줄일 수 있다.

따라서 본 논문에서는 VLSI회로의 패턴 매칭[1]에 관한 역트랙킹(backtracking) 깊이-우선 탐색을 할 수 있는 그래프 탐색 매칭 알고리즘을 제안하였다.

II. 부-그래프의 패턴 객체

패턴과 작용의 선언은 패턴 객체를 나열하기 위해 사용되는데 패턴 객체의 실행은 아래와 같이 동작한다. 즉, 패턴 선언의 "start" 네트 또는 소자를 매치하고 다음에 네트리스트[2]로부터 네트 또는 소자를 찾고 그 다음에 네트리스트에서 패턴의 나머지에 매치된 것을 찾는다. 그리고 매치된 후 선언에 일치하는 이름의 네트리스트에서 매칭 네트와 소자를 결합한 다음 선언의 작용(action) 코드가 수행된다. 이러한 과정에서 그림 1과 그림 2는 전체 진행 단계를 묘사하는데 그림 1은 패턴 객체 선언을 나타내고, 그림 2는 패턴을 수행하는 응용부분의 외부 제어 흐름 루프를 나타낸다. 이러한 패턴 객체와 루프를 나타내는 패턴과 작용(action)의 특징은 4가지로 언급되는데 첫 번째는 패턴 명칭, 두 번째는 변수 선언, 세 번째는 위상 묘사, 네 번째는 작용(action)을 묘사한다.

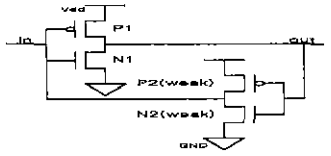
* 패턴 명칭은 그림 1의 패턴을 수행하는 그림 2의 루프처럼 패턴을 수행하기 위해 사용된다.

* 변수선언 부분은 패턴의 네트와 소자의 명칭을 할당한다. 명칭은 패턴의 상호 연결을 묘사하는 위상 부분에서 사용된다 그림 1에서 패턴은 네트 변수 "Vdd" 와 "GND"가 네트리스트에서 Vdd 와 GND 네트를 매치 할 수 있다는 것을 나타내기 위해 "literal"을 사용한다.

* 위상(topology) 묘사는 선언부에서 이름의 항목으로 쓰여진 netlist 이다. 네트리스트는 터미널과 네트 연결의 각각의 소자 이름을 가진 목록이다.

* 작용(action)은 변수 이름에 의해 패턴 변수에 부합되는 일반-목적 프로그래밍 언어 C 코드[3] 부분이다. 작용을 한번 실행시킨 후 "return"문이 포함하지 않는다면 탐색은 계속 될 것이다.

그림 1은 staticized 래치의 weak-궤환을 제거하기 위한 패턴 실행에 대한 특징을 나타낸다. 패턴은 두 인버터의 루프를 나타내는 4개의 트랜지스터 회로망으로 설명되는데 패턴에서 소자 N1은 시작 소자를 선언하고 그림 2에서 while 루프는 N1에 대한 매치로서 네트리스트의 소자에 의해서 제안된다. 작용은 일치하는 변수 동의 명칭에 의해서 매치된 소자의 특성을 어세스 할 수 있고 "N2.width"는 N2에 매치된 netlist에서 소자의 넓이로 적용한다.




```
pattern static_latch
start device N1
device N2, P1, P2
net in, out
literal net Vdd. GND

위상(topology)
((N1 nfet) ((gate in) (srcdrn out) (srcdrn GND)))
((P1 pfet) ((gate in) (srcdrn out) (srcdrn Vdd)))
((N2 nfet) ((gate out) (srcdrn in) (srcdrn GND)))
((P2 pfet) ((gate out) (srcdrn in) (srcdrn Vdd)))

작용(action){
if ((N1.width >= N2.width) && (P1.width >= P2.width) {
  delete_device (N2) ;
  delete_device (P2) ;
  num_static_latches++ ;
}
return ;
}end_action
```

그림 1. 정적 래치의 weak 궤환을 제거하기 위한 패턴

```
while (d= /*netlist에서 모든 소자(패턴명칭)*/)
  static_latch(d, Vdd, GND) ;
```

그림 2. 패턴 객체의 호출

III. 부-그래프의 동형(isomorphism)[4]

III-1. 역트랙킹(backtracking) 깊이-우선 탐색

네트리스트(netlist)에서 네트와 소자를 일치 시키기 위해서 패턴/부회로는 네트에 대해서 그리고 소자에 대해서 변수 리스트를 고려해야만 한다 역트랙킹 탐색은 네트리스트로부터 특정 후보의 경계변수 중 하나에서 시작한다 실행 후 남은 변수 중 하나는 패턴 객체 위상(*topology*)을 매칭 한다 그림3은 의사코드(pseudo-code)의 역트랙킹 깊이-우선 탐색에 대한 그래프 탐색 매칭 알고리즘을 나타낸 것으로 알고리즘의 2, 3번째 라인에서 매칭 되는 변수는 한번에 실행되고 매칭 되지 않는 변수를 탐색하기 위해 또는 이미 매칭 된 인접변수를 탐색하기 위해 선택한다. 이 알고리즘은 변수의 순서에 관계없이 이웃을 탐색하는 다중 매치 효과를 발견할 수 있다

```
1  while (모든 변수가 경계되지 않음) {
2    비경계 변수 u를 선택,
3    경계 변수 b로 패턴에 인접시킴.
4    while (모든 객체 o, b로 경계 객체에 netlist로 인접시
                                          킴) {
5      if (o가 항시 변수에 경계된다면) 다음 o로 skip.
6      while (모든 경계변수 a를 u에 인접 시킴){
7        if (a의 객체 경계가 u의 객체경계에 인접되지 않는
                                  다면) 다음 경계변수 o로 skip.
8        }
9        u에서 o로 포함시킴.
10       다음 u로 처리.
11     }
12     실패이면 return.
13   }
14   성공이면 return.
```

그림 3. 그래프 탐색 알고리즘

만일 패턴이 CMOS 인버터이고 이웃의 변수-매칭 순서를 선택한다면 그림 4는 CMOS 인버터 스케메틱에서 첨기되는

탐색순서를 보여준다. 매칭된 변수의 각각의 쌍과 전에 매칭된 이웃은 패턴의 인접 그래프의 호(arc)를 나타내는데 이러한 모든 쌍으로부터 호들은 인접 그래프 트리를 형성한다. 역에이지(back-edge) 트리가 아닌 인접 그래프 패턴에서 호는 매칭위상을 검증하기 위하여 탐색(그림 3의 6-8 라인의 루프에 의해서)하며 그림 4의 CMOS 인버터에 대한 탐색 순서도는 그림 5에서 보여준다.

그림 5에서 인버터 탐색 순서도의 다중 while 루프의 예는 네트가 많은 팬아웃을 가질 때 실행하기 어렵다. 빠른 후보는 네트 변수 "O"를 선택하기 위해 네트리스트에 접지(gnd)를 만들어 주며 예제에서 n-Mos와 p-Mos사이를 구별할 수 없게 하기 위하여 접지에 연결된 모든 소자는 "P"에 대하여 적당한 후보를 만든다. "I"가 "P"에 인접한 곳에서 2개의 while 루프처럼 "P"에 대한 각각의 후보는 제거될 수 있다. 만일 "인버터-탐색"이 네트리스트에서 소자를 부른다면 내부에서 "I"로부터 while 루프는 O((Gnd의 소자 수)^2)시간으로 실행된다

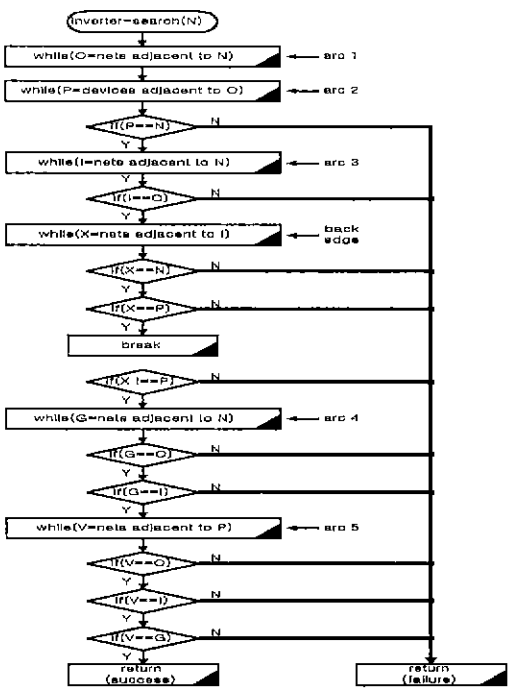


그림 5. 인버터 탐색 순서도

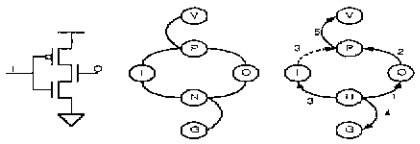


그림 4. CMOS 인버터의 탐색 순서

IV. 결 론

빌딩 블록을 구성하기 위한 툴은 네트리스트(netlist) 처리 단계를 열거하고 수행하기 위한 소프트웨어의 추상적 개념과 패턴 객체에 의해서 결정된다. 네트리스트를 전처리하고, 분석하고, 그리고 변환하기 위하여 패턴 객체를 이용한 시스템은 네트리스트를 수행하기 위해 사용된다.

따라서 본 논문에서 제한된 패턴 매칭 알고리즘과 탐색 순서도는 네트리스트 처리 툴의 개발에 좋은 선택권을 부여한다. 그래서 디자이너는 디자인 방법론을 추구하고 소자 네트리스트는 이러한 방법론에 의미가 있다. 그리고 이러한 알고리즘은 실용적인 CAD 해법을 빠르게 제공하기 위하여 지원될 수 있을 것이다.

참 고 문 헌

[1] J.Ousterhout, "Tcl and the Tk Toolkit",Addison-Wesley, 1993.

[2] G.Pelz, "An Interpreter for General Netlist Design Rule Checking", Proc. 29th Design Automation Conference, 305-310, 1992.

[3] A.Aho and J.Ullman, "Principles of Compiler Design", Addison-Wesley, 1985.

[4] Miles Ohlrich, Carl Ebeling, Eka Ginting and Lisa Sather, "Subgemini : Identifying Subcircuits using a fast Subgraph Isomorphism Algorithm", Proc 30th Design Automation Conference, 31-37, 1993.