논문 2005-42SD-11-4

상위 수준 설계 도면의 자동 생성

(Automatic generation of higher level design diagrams)

이 은 철*, 김 교 선**

(Eunchoul Lee and Kyosun Kim)

요 약

회로도면 자동생성 분야는 지난 수십 년간 HDL기반 설계과정에서 사용되어 왔다. 그러나 회로 도면은 더욱 복잡해져서 레지스터 및 시스템 레벨에서 자동 생성된 회로도면을 보고 신호의 흐름을 파악하기 어렵다. 이와 같이 복잡해진 회로도면의 가독성을 향상시키기 위해 본 논문에서는 4가지 기법, 즉 i) 심볼이나 터미널들과 같이 반복되는 회로 패턴을 벡터 형태로 치환, ii) 피드백 루프 절단 알고리즘 개선, iii) 번들 네트 생성시 발생하는 다단 연결을 간결 화할 수 있는 압축 탭, iv) 연결도에 따라 블록열을 구분하고 정렬하는 알고리즘을 제안한다. 제안된 회로도면 생성 기법의 효용성을 확인하기 위해 도면 자동 생성 프로그램을 개발하고, 계층적으로 설계된 미디어 프로세서의 다양한 모듈의 도면을 생성시켰다. 실험한 결과 도면 면적을 비롯하여 배선 수, 길이 등을 90%까지 감소시키고 가독성을 높이는 효과를 보였으며 블록의 분산 및 빈 공간 발생을 억제하는 효과를 보였다.

Abstract

The automatic generation of circuit diagrams has been practically used in the HDL based design for decades. Nevertheless, the diagrams became too complicated for the designers to identify the signal flows in the RTL and system level designs. In this paper, we propose four techniques to enhance the readability of the complicated diagrams. They include i) the transformation of repetitive instances and terminals into vector forms, ii) an improved loop breaking algorithm, iii) a flat tap which simplifies the two level bus ripping structure that is required for the connection of a bundle net to multiple buses, and iv) the identification of block strings, and alignment of the corresponding blocks. Towards validating the proposed techniques, the diagrams of an industrial strength design are generated. The complexity of the diagrams has been reduced by up to 90% in terms of the number of wires, the aggregate wire length, and the area.

Keywords: Diagram Generation, Vector Instance, Bundle Net, Flat Taps, Loop Breaking.

I. 서 론

반도체 미세화의 발달로 칩 상에 탑재되는 회로 규모는 대형화되는 반면 선 폭과 간격이 줄어들어 신호선간 잡음 및 전원 전압강하 문제가 동작 불량 및 성능 저하를 일으키고 있다. 또한, 설계 기간을 단축하기 위해서는 문제를 찾아내는 것뿐만 아니라 문제를 사전에 방지하는 설계 방식을 요구하고 있다. 이는 레이아웃 또는 게이트 설계 단계뿐만 아니라 레지스터 또는 시스템 레벨의 계획 단계로부터 적용될 수 있어야 한다. 계획 단계에서 설계자는 넓은 설계 공간을 탐색하면서 다양한 사양을 쉽고 신속하게 평가해 보는 것이 중요하다. 특히, 상황의 빠른 이해와 창조적인 직관력을 끌어내기 위해서는 설계 대상 시스템을 도면 형태로 표현하는 것이 효과적이다. 따라서 설계자가 Verilog 또는 VHDL등의 하드웨어 기술 언어나 그 밖의 방법으로 작성된 시스템 기술을 입력하면 이것의 블록 도면이 자동으로 생성되고 이를 통해 시스템의 전체를 파악하거나 계층 구조를 따라 가면서 세부를 살필 수 있어야 한다. 도면 자동 생성 분야는 80년대 말 90년대 초에 많은 연구가 있어 게이트 설계 단계에서는 실용화되어 있으나[1,2,3,4,5,6,7,8]

* 학생회원, ** 정회원, 인천대학교 전자공학과
(Department of Electronic Engineering University of Incheon)

※ 본 연구는 산업자원부, 한국산업기술평가원 지정 인천대학교 멀티미디어연구센터의 지원에 의한 것임.
접수일자: 2005년6월23일, 수정완료일: 2005년11월7일

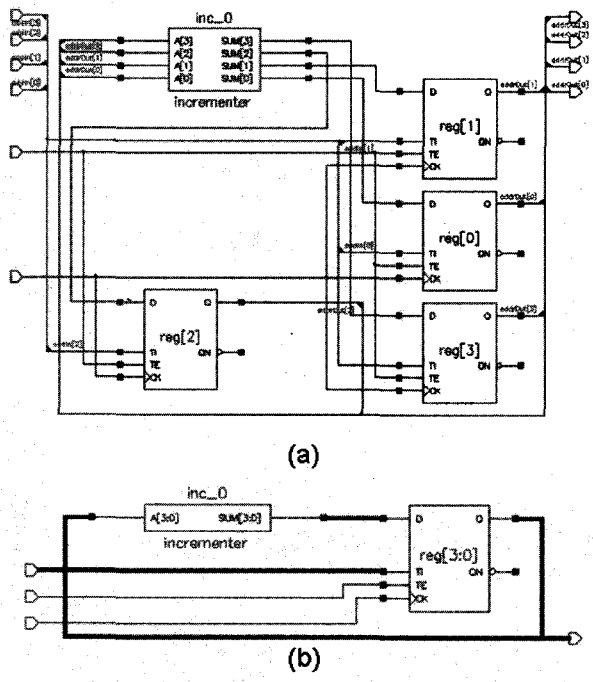


그림 1. 계수기 회로 도면
Fig. 1. Diagrams of a counter.

아날로그 회로[9]나 레지스터 또는 시스템 레벨에서는 연구가 매우 초보적이다[10]. 본 논문은 HDL기반 설계과정에서 필요한 도면 자동 생성 소프트웨어에 관한 것이며, 게이트 레벨에서 확장하여 레지스터 및 시스템 레벨의 도면을 자동 생성하는 기법을 연구한 것이다.

회로 도면 생성 기법이 게이트 레벨에서는 이미 성숙되었음에도 불구하고 레지스터 또는 시스템 레벨에서 새로운 연구를 해야 할 동기를 부여하기 위해 그림 1(a)와 같은 회로 도면을 검토해 보자. 4개의 플립플롭 reg[0], reg[1], …, reg[3]이 4비트 레지스터를 형성하고 증산기가 이 레지스터의 값을 증가시키는 계수기를 구성한다. 4비트임에도 불구하고 터미널10개, 심볼 5개, 배선용 선분 52개가 사용되어 시각적으로 회로를 신속히 이해하는데 도움을 주고 있지 못하다. 즉, 정보의 세부 사항을 유실하지 않고 오히려 가독성을 높이면서도 같은 공간에 더 큰 회로를 함축적으로 표현하도록 하는 기법이 필요하다. 그림 1(b)는 4개의 플립플롭을 하나의 벡터 심볼로 묶어 레지스터의 개념으로 추상화하고 낱개의 비트로 표현된 터미널이나 배선들도 버스 형태로 묶어줌으로써 같은 회로를 훨씬 간결하고 함축적으로 표현하고 있다. 배선에 사용된 선분의 총수도 13개로서 75%가 감소했음을 알 수 있다. 따라서 회로 도면의 생성문제는 주어진 연결 정보에 따라 회로 소자들을 단순히 평면상에 배치하고 연결하는 문제에서 벗어나 좀 더 간결하고 함축적인 표현을 개발하고 이를 기반으로 주어진 정보를 재구성하는 문제로 확장되어야 한다.

본 논문은 다음과 같이 구성한다. 먼저, Ⅱ장에서 일반적인 회로도면 자동생성 과정을 소개하고 레지스터 및 시스템 레벨회로를 자동 생성하는데 있어 이슈가 될 수 있는 회로의 축약, 피드백 절단 알고리즘, 압축 탭을 이용한 연결 간소화 및 블록 정렬에 대해 Ⅲ장에서 소개한 후 그에 대한 알고리즘을 Ⅳ장에서 설명한다. 이후 제안된 기법의 효용성을 검증하기 위해 개발한 프로그램으로 실험한 실험 결과를 Ⅴ장에서 설명하고 Ⅵ장에서 결론을 맺을 것이다.

Ⅱ. 회로 도면의 자동생성

HDL 네트리스트로부터 회로 도면을 자동으로 생성시키는 절차를 그림 2에 나타내었다. 먼저 심볼들의 상대적 위치가 위상정렬로 결정된다. 일반적으로 신호의 흐름을 좌측에서 우측으로 정의하기 때문에 입력의 최근접 심볼이 좌단에, 출력의 최 근접 심볼이 우단에 위치하게 된다. 입력에서부터 출력까지 경로가 형성되는데 이 경로에 따라 심볼들의 순서를 정하는 것이 수평정렬이다. 신호의 흐름을 좌에서 우로 정하였다 할지라도 이에 반하는 신호도 존재하게 되는데, 이것을 피드백 루프 (Feedback Loop)라 한다. 이러한 루프가 회로 안에 존재하면 회로는 트리 구조에서 벗어나기 때문에 위상정렬 시 심볼간 상대적인 위치를 결정할 수 없다. 따라서 위상정렬 시에는 먼저 피드백 루프를 절단해야만 한다. 모든 심볼들의 수평적 상대위치가 결정되면

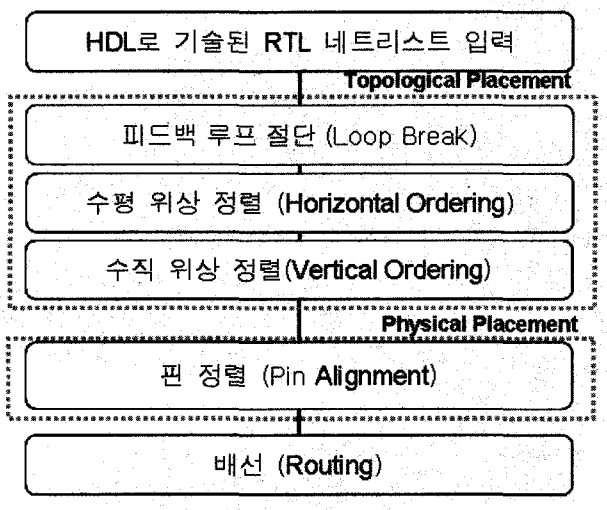


그림 2. 회로도면 자동생성의 흐름도
Fig. 2. A general flow automatic diagram generation.

입력으로부터 출력까지 경로가 다수 형성된다. 배선 복잡도를 줄이기 위해 이 경로들 간의 교차가 최소화 되도록 심볼들 간의 상대적 수직위치를 결정하는 수직정렬도 필요하다[1]. 위상정렬 후에는 가능한 한 배선이 꺾여서 연결되지 않도록 심볼의 핀들을 실제 좌표 상에서 수평 정렬시켜야 하며[6] 최종적으로 배선이 수행된다.

회로도면 자동생성 분야는 알고리즘[1,3,6,8], 휴리스틱[2], 혼성[4,10] 및 규칙기반 시스템[5,7]으로 그 연구 개발에 다양한 시도가 이루어 졌다. 또한 피드백 루프 절단 과정에서 절단될 네트의 선택에 따라 나타나는 차이에 대해 여러 가지 논의[5]가 있었지만 검출 속도의 중요성을 우선시하여 이러한 차이는 무시되었다. 최근에는 배치와 배선의 과정을 동시에 수행하며 심볼들 간의 배선 교차수를 최소화하여 도면의 복잡도를 감소시키는 기법[11] 및 부울 표현식을 기초로 트랜지스터 개수를 최소화하여 회로로 변환하는 기법[12] 등 변화하는 설계방법론에 따라 발생하는 새로운 이슈들에 대해 간헐적이지만 지속적인 연구가 이루어져 왔다.

HDL 기술에서 하드웨어를 컴파일 할 때 추상도가 더 높은 표현을 사용하여 RTL 도면의 복잡도를 줄이고자 하는 것은 최근 도면 자동생성에 있어 가장 큰 이슈다[10]. 이 접근은 대부분의 상용 HDL 기반 설계 소프트웨어[13][14]에서 사용하고 있는 기법으로 컴파일 과정시 반복되는 형태가 낱개로 풀어지는 것을 억제하여 도면의 복잡도를 감소시키는 기법일 뿐 동일한 패턴의 벡터화는 아니다. 본 논문에서는 단순한 억제가 아닌 더 적극적인 방법으로 반복되는 구조를 벡터화 하였으며 그 결과 생성된 회로도면은 더욱 간결하고 함축적이며 원래의 정보는 전혀 유실하지 않는다.

오늘날 IC 설계에서는 여러 가지 단위 공정들 간 고품질의 디자인 데이터교환이 요구되며 이들의 호환성 및 상호 이용성이 필요하다[15]. 이를 위해 본 논문에서는 EDA 업계 표준으로 제안된 공용 데이터베이스인 OpenAccess[16]상에 기능 모듈을 개발하였다.

Ⅲ. 이슈 및 접근 방법

1. 회로의 축약

회로는 규모가 증가함에 따라 다수의 모듈로 분할되어 설계되며 이들 모듈 간의 연결은 한 단계 상위의 구조적 설계로 표현하는 계층구조가 형성된다. 이와 같은 계층적 설계는 회로의 상세를 숨기고 거시적으로 표현하는 추상화의 주체적 역할을 하지만 도면 표현에는 충분하지 않다. 그림 1(a)는 증산기를 부 회로로 추상화 하였음에도 불구하고 레지스터를 구성하는 플립플롭이 분산되어 있고 연결이 거미줄처럼 얽혀있어 신호의 흐름이나 기능을 쉽게 파악하기 어렵다. 이를 간결하고 함축적으로 표현하여 가독성을 높이려면 그림 1(b)와 같이 반복되는 회로 패턴을 묶어 벡터형태로 치환하는 방법이 효과적이다. 이것은 입출력 터미널, 네트, 셀이나 블록에 적용된다. 증산기의 외부 입출력 터미널들과 셀의 입출력 터미널들이 버스 형태로 축약된 것과 플립플롭 4개가 벡터 심볼 형태로 묶여 레지스터 하나로 축약된 것에 주목하라. 이러한 축약 기법에는 먼저 회로의 패턴 인식이 필요한데 그림 1과 같이 벡터형태로 표현된 네트나 심볼의 이름으로부터 축약될 대상을 추출할 수 있으나 그것만으로는 실제 벡터 형태로 치환할 수 있는 경우가 제한적이어서 연결 패턴을 인식하는 기술을 보완적으로 적용해야 한다.

2. 피드백 루프 절단 알고리즘

신호의 흐름에 따라 심볼들을 좌에서 우로 정렬시키기 전에 먼저 피드백 루프를 절단하여 회로를 트리 구조로 바꾸어 주어야 한다. 이를 위해 깊이 우선 탐색에 근거한 Tarjan의 알고리즘[17]이 흔히 사용되어 왔으나 회로의 탐색 순서에 따라 결과에 차이가 있을 수 있다는 점이 지적되었다. 피드백 루프 절단 시 문제를 파악하기 위해 그림 3(a)와 같이 그래프 형태로 간략화 된 회로를 검토해 보자.

심볼 1, 2, 3, 4, 5가 네트 a, b, c, d, e, f, g, h로 연결되어 있으며 루프 c, d, e, h가 존재한다. 심볼 1에서 네트 b가 먼저 탐색되어 네트 c, d, e를 거친 후 네트 h에 이르게 된다. 이 네트 h를 통해 이미 방문했던 심볼 2에 도달하면서 루프가 검출되며 이 네트 h를 절단하면 피드백은 제거된다. 그러나 그림 3(b)와 같이 네트 f가 먼저 탐색되면 루프는 네트 e위치에서 검출되며 이 네트 e가 절단된다. 이때 출력 g에 연결된 심볼 5가 좌측

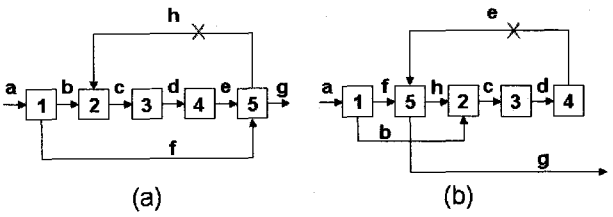


그림 3. 피드백 루프의 선택과 위상정렬
Fig. 3. Horizontal ordering affected by the ordering of the input data in the feedback loop breaking.

으로 치우치게 되어 배선이 복잡해지고 신호 흐름 파악에 어렵게 된다. 다행히도 게이트 레벨의 도면에서는 피드백 루프가 적어 이러한 차이는 무시하고 검출 속도를 향상시키는 것이 더 중요시 되었다[5]. 그러나 시스템 레벨의 회로에는 기능 블록들이 상호 데이터를 주고받는 구조를 취하고 있어 많은 피드백 루프들이 존재할 뿐만 아니라 블록의 크기가 상대적으로 크고 입출력이 많아 루프를 절단하는 위치에 따라 도면의 전체 모양이 크게 달라질 수 있다. 따라서 루프 상에서 절단될 네트를 선택할 때 우연성에 의존하지 말고 좀 더 정교한 노력을 들여야 한다.

3. 연결 간소화

3.1 절에서 제안한 바와 같이 심볼 및 외부 입출력 터미널을 벡터 형태로 축약하면 연결 구조에 변화가 발생한다. 이러한 변화가 배선에 어떤 문제를 일으키는지 파악하기 위해 그림 4(a)를 검토해 보자.

심볼 U의 터미널 X[5], X[4], X[3], X[2]는 버스 A[15:0] 로부터 한 비트씩 탭을 내어 A[7], A[6], A[5], A[4]에 각각 연결되어 있으며 터미널 X[1]과 X[0]는 버스 B[7:0]에서 탭을 낸 B[3] 및 신호선 C와 각각 연결되어 있다. 이때 이 터미널들을 버스 형태인 X[5:0]로 축약하면 그림 4(b)와 같이 버스 A[15:0] 및 B[7:0], 그리고 신호선 C로부터 각각 탭을 내어 A[7:4], B[3] 및 C를 각각 분기하고 이들을 번들 네트 A[7:4],B[3],C로 묶어준 후 터미널 X[5:0]에 연결해야 한다. 그러나 이러한 다단 연결은 회로 도면을 복잡하게 하며 만약 터미널 Y가 연결되어야 한다면 이 다단 연결 패턴을 우회하여야 하고 이를 자동화 하기란 쉽지 않다.

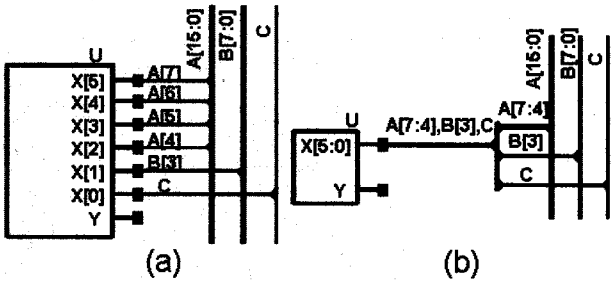


그림 4. 탭을 사용한 번들네트의 연결
Fig. 4. Vectorization induced bundle net connection.

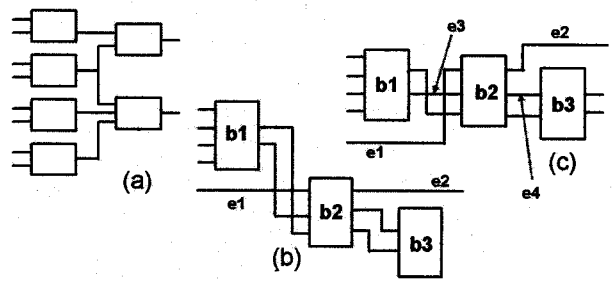


그림 5. 블록 정렬의 필요성
Fig. 5. A needs of block ordering.

4. 블록의 정렬

게이트 수준의 회로는 심볼의 크기가 비교적 작고 핀 수가 적어 도면이 그림 5(a)와 같이 트리 형태로 생성될 때 가장 이상적이다. 따라서 핀 수가 적은 네트 및 심볼들을 기준으로 심볼을 정렬시키는 것이 일반적이다. 그러나 시스템 수준에서는 블록 b1, b2, b3와 같이 심볼이 크고 핀 수가 많기 때문에 이와 같이 하면 그림 5(b)에서와 같이 빈 공간이 많이 생기고 불균형적인 구조가 될 경우가 많다. 일반적으로 그림 5(c)와 같이 블록들을 중심으로 정렬시키는 방법이 선호된다. 특히 블록들과 작은 게이트들이 섞여 있을 때 블록들을 우선적으로 정렬하고 그 주위로 게이트를 배치하면 자연스럽게 된다.

Ⅳ. 알고리즘

1. 수평 위상 정렬

피드백 루프는 (1) 외부 입력과 가장 가까운 심볼의 입력 및 (2) 외부 출력과 가장 가까운 심볼의 출력과 연결된 네트에서 절단하는 것이 가장 이상적이다. 그림 3(a)는 조건 (1)과 (2)를 만족하는 네트 h를 절단 했지만, 그림 3 (b)는 조건 (2)를 만족하지 않는다. 두 조건을 만족시키기 위해서는 깊이 우선 탐색에 의해 루프가 발견되면 이를 구성하는 네트들 중 외부출력과 가장 가까운 심볼을 찾아 그것의 출력 네트를 절단해야 한다. 그러나 깊이 우선 탐색은 임의의 심볼에서 외부 출력까지의 거리를 알 수 없다. 반면에 너비 우선 탐색은 외부 입출력으로부터 각 심볼까지 최단 거리를 알 수 있다. 따라서 너비 우선 탐색을 통해 입출력으로부터 각 심볼의 거리를 측정한 후, 깊이 우선 탐색에서 찾은 루프에서 출력에 가까운 심볼의 출력을 절단하도록 할 수 있다.

피드백 루프 절단이 포함된 수평 위상 정렬 알고리즘을 그림 6에 나타내었다. S는 네트리스트에 있는 모든 심볼들의 집합이고 A는 수평 위상 정렬된 심볼들의 배열이며 B는 루프를 이루는 넷들의 배열이다. 먼저 너비 우선 탐색으로 외부 출력에서 입력까지 네트에 역순번

```
   S = { g | symbols in the netlist }
   A[] = symbol array for backtracing
   B[] = net array for backtracking
   HorizontalOrder(S)
   {
1:   BackwardBFS(S);
2:   s.enter ← s.leave ← 0 ∀s ∈ S;
3:   aIndex ← bIndex ← 0;
4:   foreach s ∈ S
5:     if(!s.enter) HorizontalOrderRec(0, s);
6:   return (A);
   }
   HorizontalOrderRec(n, s)
   {
1:   if(s.enter && !s.leave) {
2:     for(i ← bIndex - 1; i ≥ 0; i++)
3:       if(n.level < B[i].level) n ← B[i];
4:     n.break ← 1;
5:     return (backtrack);
     }
6:   s.enter ← 1;
7:   B[bIndex++] ← n;
8:   foreach n ∈ s.children {
9:     c ← n.symbol;
10:    if(HorizontalOrderRec(n, c) = backtrack) {
11:      if(n.break) continue;
12:      s.enter ← 0; bIndex--;
13:      return (backtrack)
       }
     }
14:  A[aIndex++] ← s;
15:  s.leave ← 1;
16:  return (normal);
   }
```

그림 6. 수평 위상 정렬 알고리즘
Fig. 6. Horizontal ordering algorithm.

을 부여한다 (1번 줄). 2-3번 줄에서 모든 심볼의 방문표시와 귀환표시, 그리고 배열 A와 B의 첨자로 사용할 aIndex와 bIndex 변수를 초기화 한다. 4, 5번 줄에서 각 심볼에 대해 방문표시가 아직 안되어 있으면 HorizontalOrderRec()을 호출한다. HorizontalOrderRec()에서는 먼저 루프가 탐색 경로 상에 형성되었는지 확인하여 (1-5번 줄) 그렇지 않으면 입력 심볼 s에 방문표시를 하고 (6번 줄) 루프가 발생했을 때 필요한 후진에 대비하여 배열B에 입력 네트 n을 저장한다 (7번 줄). 8-13번 줄에서 심볼 s의 모든 자손에 대해 그 자손에 진입하는 네트 및 그 자손을 입력으로 HorizontalOrderRec()를 수행한다. 모든 자손에 대한 탐색이 끝나면 경로상의 심볼들이 배열 A에 역순으로 채워지면서 후 추적이 진행된다 (14-16번 줄). 한편 1-5번 줄에서 s에 방문표시만 있고 귀환표시가 없으면 루프가 검출되며 루프 안에서 가장 큰 역 순번의 네트를 절단한다. 5번 줄에서 후진을 시작하여 11-13 번 줄에서 절단된 네트에 도달할 때까지 계속한다.

그림 7(a)는 상기 알고리즘이 각 심볼을 이동하는 경로이다. 심볼 1에서 심볼 5를 먼저 방문한 후 심볼 4, 2, 3을 거쳐 다시 심볼 4를 방문할 때 루프가 검출되며 루프 안에서 가장 큰 역순 번호를 가진 네트 f에 연결된 심볼 4 까지 후진한다. 이후 후 추적을 수행하여 배열 A가 채워진다. 배열 A가 결정되면 각 심볼들의 수평 순서를 결정하는데 우선 모든 수평 순서를 0으로 초기화 시키고 배열 A의 마지막 원소부터 역순으로 자기 부모 심볼의 수평순서보다 한 단계 더 늦은 순서를 부여한다. 그림 7(b)의 마지막 원소인 심볼 1의 부모는 없기 때문에 순서가 0이고 심볼 2의 부모는 심볼 1이기 때문에 심볼 2의 순서는 심볼 1의 순서보다 한 단계 늦은 1을 갖게 된다. 한편 심볼 4의 부모는 심볼 3과 7이 있는데 그 중 심볼 7의 순서가 늦기 때문에 심볼 4의 순서는 심볼 7의 3보다 한 단계 늦은 4가 된다.

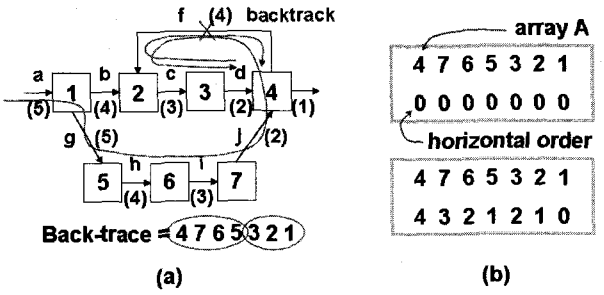


그림 7. 수평정렬 알고리즘 수행
Fig. 7. A horizontal ordering example.

2. 압축 탭

압축 탭을 이용하여 그림 4를 개선하면 그림 8과 같이 된다. 그림 4(b)와 동일한 연결을 가지고 있지만 간결하면서도 의미가 전혀 손상되지 않는다. 버스 이름 A[15:0]는 어간 A와 비트 구간 [15:0]로 구성되어 있다. 번들 네트 A[7:4],B[3],C는 세 개의 어간(base) A, B, C를 가지고 있으며, 어간이 하나뿐인 세 개의 단일 어간 네트 (single base net) A[7:4] 및 B[3] 그리고 C로 나누어진다. 이들은 버스 A[15:0] 및 B[7:0] 그리고 신호선 C로부터 탭을 통해 분기되고 연결되어 연결 정보를 시각적으로 이해할 수 있을 뿐만 아니라, 네트 이름으로부터 연결 정보를 정확히 파악할 수 있다. 터미널 Y도 우회하지 않고 연결 할 수 있다는 장점도 있다.

본 논문에서 제시하고자 하는 압축 탭 및 이를 이용한 번들 네트의 연결 기법은 다음의 세 가지 규칙으로 구성된다. 먼저, 기존의 탭 표현 (그림 4(b))은 분기된 비트가 주축 네트에 포함된다. 그러나 압축 탭은 이 포함 조건을 제거함으로써 번들 네트와 단일 어간 네트 연결을 그림 8에서와 같이 1단계로 압축할 수 있다. 둘

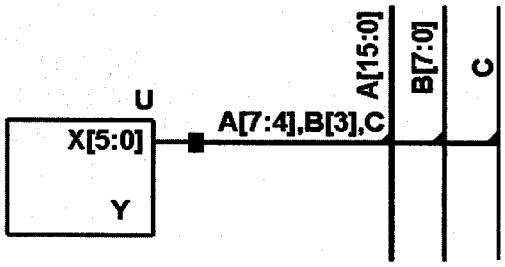


그림 8. 압축 탭을 이용한 번들 네트의 연결
Fig. 8. A bundle net connection using flat tpas.

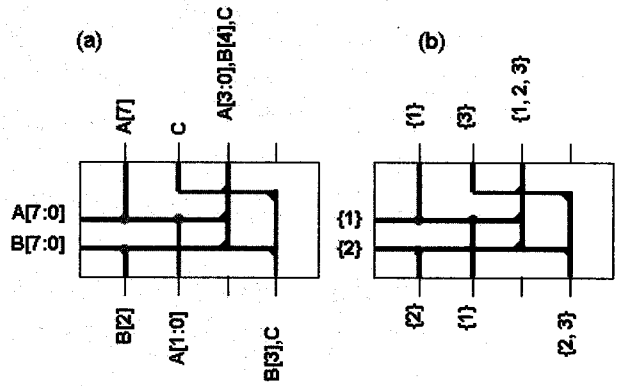


그림 9. 압축 탭 연결이 존재하는 채널배선 문제
Fig. 9. Channel routing problem with flat taps.

째, 압축 탭의 주축 네트는 반드시 최대 단일 어간 네트이어야 한다. 이 최대 단일 어간 네트의 비트 구간은 같은 어간을 가지는 모든 네트의 비트들이 모두 포함될 수 있는 연속구간이며 필요하면 새롭게 생성시킬 수도 있다. 예를 들어, 만약 A를 어간으로 하는 두 개의 단일 어간 네트 A[3:0] 및 A[12:11]이 존재한다면 최대 단일 어간 네트 A[12:0]이 생성된다. 셀 터미널에 연결된 대부분의 번들 네트가 단거리 내에서 하나 또는 다수의 최대 단일 어간 네트에 즉시 연결되고 주로 이 최대 단일 어간 네트들이 원거리를 연결하기 때문에 도면상에 나타난 최대 단일 어간 네트의 분포를 통해 연결 상황을 파악할 수 있다. 다수의 어간을 포함하고 있어 이름이 복잡한 번들 네트 다수가 도면 전체에 분포되면 연결 상황 파악이 어렵게 되는데 이것을 방지하고자 함이다. 마지막으로 다수의 최대 단일 어간 네트로부터 각각 탭을 내어 하나의 번들 네트에 연결할 때 이 다수의 압축 탭들이 번들 네트를 구성하는 직선 하나에 정렬된다. 기존의 탭은 하나의 분기선이 하나의 주축에서만 분기될 수 있었는데 압축 탭은 여러 개의 주축에서 하나의 분기선으로 분기될 수 있다. 셀 터미널에서 분기의 주축인 최대 단일 어간 네트 다수와 연결할 때 단 하나의 선만 필요하기 때문에 다른 터미널의 연결을 방해하지 않는다.

3. 벡터 심볼 및 터미널

회로를 간결하고 함축성 있게 축약하여 가독성을 향상시키려면 반복되는 회로 패턴을 묶어 벡터 형태로 치환하는 방법이 효과적이다. 이를 위해 ① 이름에 동일 어간을 가지고 비트 구간이 연속적인 심볼 혹은 터미널들, ② 출력 터미널이 동일 버스 네트에 연결되고 마스터가 같은 심볼들, 그리고 ③ 버퍼와 같이 입력과 출력이 하나씩인 심볼이면서 출력이 동일 버스네트에 연결된 경우는 해당 심볼 혹은 터미널들을 벡터 형태로 묶어서 치환할 수 있다. 이러한 축약 기법은 그림 2에서 피드백 루프 절단 전에 입력된 네트리스트에 대해 적용된다. 한편, 치환된 벡터 심볼 혹은 터미널을 연결할 때 각 핀에 연결되는 네트가 다중 어간의 번들 네트로 묶이면 분기선보다 주축 네트가 많아지는 기형적 배선이 발생한다. 만약 이 네트들이 지역 네트이면 버스 형태로 변환시켜 어간 수 및 주축 네트 수를 줄일 수 있다.

4. 압축 탭 연결을 위한 채널 배선 알고리즘

일반적인 채널 배선 문제는 핀들이 컬럼마다 상단과 하단에 각각 최대 한 개씩 고정되어 있고 좌단과 우단에 부동 핀들이 위치한다. 압축 탭을 사용하여 번들 네트를 다수의 주축 네트에 연결하는 경우 기존의 채널 배선 문제가 어떻게 수정되어야 하는지 확인하기 위해 그림 9(a)를 검토해 보자. 버스 A[7:0] 및 B[7:0], 그리고 신호 C로부터 비트들이 분기되어 채널 상하에 고정된 핀들에 연결되었다. 번들 네트 A[3:0],B[4],C 및 B[3],C에 연결하기 위해 압축 탭이 각각 3개 및 2개가 사용되었다.

4.2절에서 설명한 방법으로 압축 탭을 사용하면 배선에서는 네트의 비트구간을 따져 볼 필요 없이 어간이 같은 네트들만 연결해 주면 되며 그림 9(b)와 같이 네트의 어간을 중심으로 간략화 할 수 있다. 어간 A, B, C는 각각 번호 1, 2, 3으로 대응되며 하나의 핀에 네트가 다수 위치하고 있다. 이 문제는 얼핏 보면 다층배선 문제와 유사하지만 ① 배선 층이 단 하나 밖에 없으며 ② 각 컬럼 기준으로 같은 핀에서 나온 네트들은 겹칠 수 있으나 다른 핀에서 나온 네트들과는 수직 제약 조건이 발생한다는 점이 특징이다. 이 문제를 해결하기 위해 Yoeli의 강건 채널 배선 알고리즘[18]과 같은 기존 알고리즘을 수정하여 사용할 수 있다. Yoeli 알고리즘은 상단 및 하단부터 교대로 중앙으로 진행하면서 트랙을

하나씩 네트들에 할당 하는데 좌측 점 정렬 알고리즘을 근간으로 수평 제약 조건을 만족시키고 컬럼 밀도에 근거한 가중치 함수를 사용하여 수직 제약 조건 위반을 최소화 한다. 채널의 상단 쪽 트랙을 할당할 때 트랙에 할당 될 네트가 하단에서 들어오고 트랙 할당이 안 된 네트가 상단 핀에 남아 있을 때 수직 제약 조건 위반이 발생한다. 하단 쪽 트랙을 할당할 경우도 유사한 논리를 적용한다. 압축 탭을 사용하는 경우와 같이 핀에 다중 네트가 고정되어 있는 경우에 Yoeli 알고리즘을 적용하려면 이 위반 검사 부분을 수정하여 반대쪽 핀에 있는 모든 네트에 대해 트랙 할당이 되었는지 검사하면 된다.

5. 블록 정렬 알고리즘

그림 2에서 수직 위상 정렬이 끝나면 모든 네트의 연결을 핀의 쌍으로 이루어진 에지들로 표현하며 이들 중 정렬시킬 에지를 선별하게 된다.

먼저 서로 교차되는 에지가 없도록 일부 에지를 제거하는데 다른 에지를 많이 교차하는 에지를 우선적으로 제거한다. 교차하는 에지를 제거하면 평면 그래프 형태가 되며 여기서 다시 각 심볼마다 정렬시킬 에지를 최대 한 개씩 선택한다. 이 때 에지 중에서 연결선의 꺾임을 줄이거나 대칭성이 좋아지도록 하는 것을 우선적으로 선택하는데 입출력이 각각 단 한 개인 심볼 및 피드-쓰루가 먼저 정렬된다. 이러한 알고리즘 특성은 그림 5(a)와 같이 게이트 수준 회로일 경우 트리 형태를 발생시키지만 그림 5(b)와 같이 심볼에 핀이 많고 크기가 불균일한 블록 도면에서는 빈 공간이 많이 생기고 불균형적인 구조가 될 경우가 많다. 이것은 기존 알고리즘이 그림 5(b)에서 높은 연결도를 갖는 블록 b1, b2, b3을 무시하고 피드-쓰루 에지인 e1과 e2를 정렬 시킬 최선의 에지로 선택하기 때문이다.

블록 정렬을 위해 기존 알고리즘은 다음의 5가지 순서로 수정되어야 한다.

① 모든 심볼들 중 블록 심볼들을 구분하여 게이트 심볼은 블록열의 후보에서 제외시킬 수 있어야 한다.

② 블록 심볼이 존재한다면 연결도에 따라 정렬될 블록열을 찾아낸다 (그림 5(b)에서는 b1, b2, b3가 블록열을 이룬다).

③ ② 과정에서 블록열이 검출되었다면 이를 정렬시킬 때 기준이 될 핀 쌍인 에지(bE)를 결정 (그림 5(c)의 e3, e4)하고 검출되지 않았다면 그래프 평면화를 진행한다.

④ 그래프 평면화 시 bE가 존재한다면 정렬시킬 최선의 에지로 bE를 선택하고 없다면 일반적인 과정을 따른다.

⑤ 선택된 에지를 기준으로 정렬 한다 (선택된 에지가 bE라면 피드-쓰루 (그림 5(c)의 e1, e2)보다 먼저 정렬된다).

블록을 정렬하는 에지들에 우선권이 부여되었지만 기존의 게이트 정렬 체계를 그대로 유지하기 때문에 블록들이 먼저 정렬되고 그 주위로 게이트들이 트리 형태를 유지하며 자연스럽게 정렬된다.

V. 실험 및 고찰

제안된 레지스터 및 시스템 레벨 회로 도면 생성 기법들의 효용성을 확인하기 위해 도면 자동 생성 프로그램 oaTopo를 개발하고, 계층적으로 설계된 미디어 프로세서의 모듈들을 각 레벨에서 다양하게 선택한 후 도면을 생성시켜 표 1에 정리하였다. 첫째 열은 모듈 이름을 나타내며 둘째 열은 모듈의 종류를 나타낸다. 최상위 모듈인 processor부터 시스템 레벨 (SL), 레지스터 레벨 (RTL), 그리고 게이트 레벨 (GL) 회로를 각각 선택하였다. 다음 3 열은 제안된 기법이 적용되기 전 (old)과 후 (new)의 심볼 수, 그리고 감소 비율 (%)을 나타낸다. 이어 배선 수, 배선 총 길이, 그리고 도면의 면적을 같은 방법으로 나타내었다. GL은 최대 54% 감소에 그치고 있으나 SL 및 RTL에서는 도면 면적 기준 최대 90%까지 높은 감소율을 보였다. 같은 방법으로 상용 프로그램인 Synopsys사의 Design Compiler (DC)[19]와 oaTopo (oa)의 비교 결과를 표 2에 나타내었다. oaTopo는 DC와는 달리 핀 이름이 블록 내에 모두 포함되도록 크기를 결정하였기 때문에 블록의 크기가 상대적으로 크며 이를 연결하는 배선의 길이 및 면적 또한 증가할 수밖에 없으므로 공정한 비교를 위해 배선의 길이 및 면적의 비교는 생략하였다. 도면의 가독성 평가에 중요한 요소인 배선의 수는 oaTopo가 DC에 비해 더 적으며 대규모 회로에서는 61%의 감소율을 보이고 있다. oaTopo와 같은 적극적인 벡터화는 아니지만 DC의 경우도 블록에서 비트 구간이 연속적인 입출력 터미널들을 버스 형태로 표현하였기 때문에 표 2에서 DC의 선분의 수가 표 1의 old 보다 적은 경향을 보였다.

표 1. 미디어 프로세서의 15개 모듈 도면 발생 결과
Table 1. Diagrams of 15 modules in a media processor.

module name	type	#symbols			#segs			length			area		
		old	new	%	old	new	%	old	new	%	old	new	%
processor	SL	119	33	72	1,326	230	83	81,656	5,314	93	287,523	25,413	91
lps_branch	SL	3	3	0	279	47	83	3,450	715	79	15,147	2,320	85
dec2_su	SL	33	3	91	274	48	82	3,186	658	79	13,674	2,214	83
op_con_lock	SL	13	12	8	355	93	74	6,300	2,234	65	26,520	7,581	71
i8051	SL	5	5	0	524	135	74	8,079	2991	63	20,608	7920	62
dec_itc	SL	11	9	18	306	73	76	4,904	1,962	60	17,303	8,442	51
triggers_x	RTL	16	2	88	99	15	85	941	66	93	3,956	315	92
rom_control	RTL	20	3	85	185	22	88	2,142	222	90	9,976	968	90
sub12	RTL	22	11	50	193	66	66	2,803	626	78	12,963	3,619	72
sub5	RTL	6	4	33	69	33	52	539	262	51	2,303	1,334	42
new_opcode	GL	256	189	26	1,238	850	31	51,950	29,138	44	288,090	133,475	54
addsub_6_0	GL	10	7	30	83	50	40	714	485	32	4,018	2,095	48
interlock	GL	235	183	22	1,087	810	25	51,749	30,959	40	230,516	174,440	24
decoder	GL	505	505	0	2,078	2,024	3	202,293	200,183	1	519,288	511,420	2

표 2. 상용 툴과의 비교
Table 2. Comparison with a commercial tool.

module name	type	#symbols			#segs		
		DC	oa	%	DC	oa	%
processor	SL	119	33	72	602	230	61
i8051	SL	5	5	0	177	135	24
dec_itc	SL	11	9	18	107	73	31
op_con_lock	SL	13	12	7	136	93	31

그림 10은 미디어 프로세서의 최상위 모듈 (processor)에 대해 DC에서 생성한 회로도면 (그림 10(a))과 oaTopo에 의해 간결하고 함축적인 표현으로 가독성이 향상된 도면 (그림 10(b))을 나타내었다. 특히 반복되는 회로 패턴의 벡터화 기법은 시스템 레벨 회로에서 상대적으로 중요하지 않은 버퍼들의 연결을 간소화하여 블록간 신호의 흐름을 쉽게 파악하는데 중요한 역할을 수행함을 알 수 있다. 피드-쓰루 정렬로 인해 블록들이 계단식으로 분산되어있던 그림 10(b)의 회로도면을 블록 정렬 기법으로 가독성을 높인 도면을 그림 10(c)에 나타내었다.

모듈 dec_itc를 DC에서 자동 생성하여 그린 도면을 그림 11(a)에 oaTopo에서 생성한 것은 그림 11(b)에 나타내었다. oaTopo가 DC에 비해 훨씬 간결한 이유는 낱개의 비트로 표현된 버퍼를 벡터화하여기 때문이다. 이때 여러 주축 네트에서 분기하여 번들네트 하나와 연결해야하는 새로운 문제가 발생하는데 선 하나만을 사용함으로써 다른 핀의 연결을 방해하지 않는 압축탭을 사용함으로써 해결되었다. 압축탭을 확대하여 그림 11 (c)에 나타내었다.

그림 12(a)는 DC에서 생성한 모듈 i8051의 도면을 나타내며 그림 12(b)는 oaTopo가 생성한 도면을 나타낸다. 적절한 피드백 루프의 절단과 연결도가 높은 세 블록의 정렬로 인하여 신호의 흐름이 명확해졌다. 또한

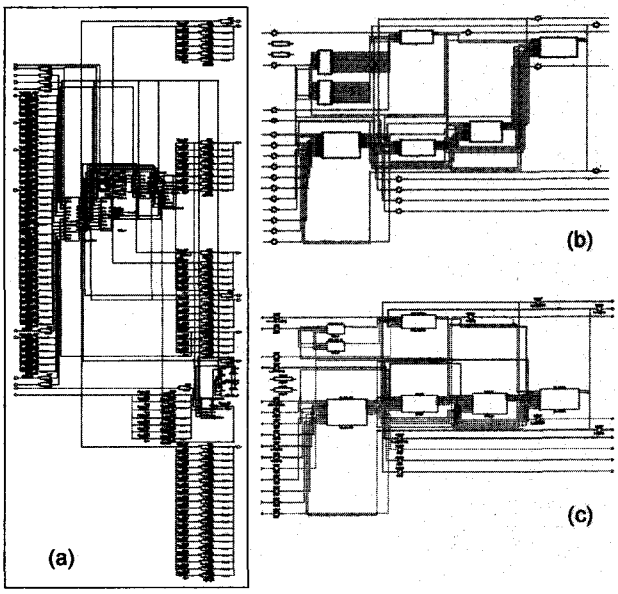


그림 10. 미디어 프로세서 최상위 모듈 (processor)
Fig. 10. Top module (processor) of the media processor.

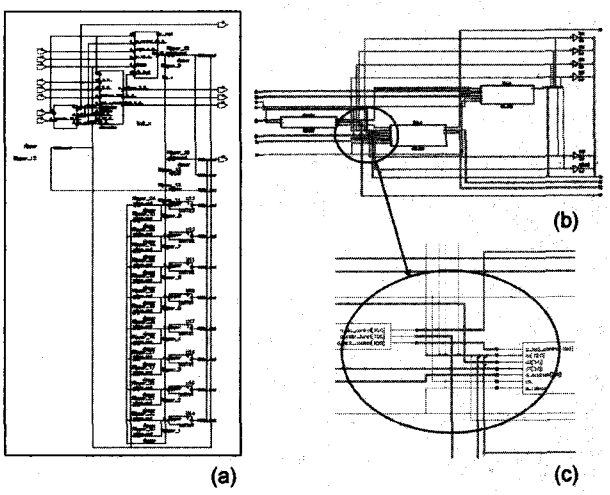


그림 11. 모듈 dec_itc 도면 상에 나타난 압축 탭
Fig. 11. Flat taps in module dec_itc.

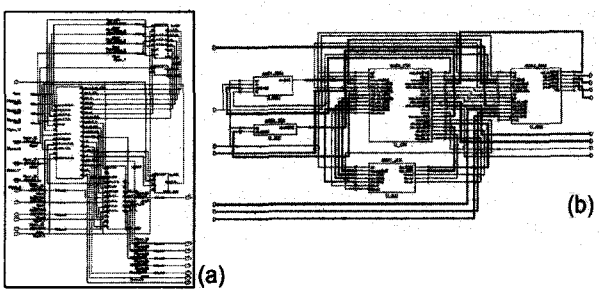


그림 12. 모듈 i8051
Fig. 12. module i8051.

연결도가 높은 세 개의 블록은 정렬되었지만 하단 두 개의 블록은 정렬되지 않았다. 블록 정렬은 연결도에 따라 정렬해야할 블록열을 검출하는 과정이 중요하다.

VI. 결 론

레지스터 및 시스템 레벨 도면 자동 생성을 위해 제안된 반복 패턴의 벡터 형 치환, 피드백 루프 절단 기법 개선, 압축 탭 연결 및 블록열을 검출하여 정렬하는 기법은 도면 면적을 비롯하여 배선 수, 길이 등을 90%까지 감소시키고 가독성을 높이는 효과와 블록의 분산 및 빈 공간의 발생을 억제하는 효과를 보였다. 또한 상용 프로그램인 Synopsys사의 Design Compiler와의 비교 결과에서도 배선 수에서 최대 61%의 감소율을 보였다.

참 고 문 헌

[1] M. May, A. Iwainsky and P. Mennecke, "Placement and Routing for Logic Schematics," Computer Aided Design, v. 15, no. 3, pp. 115-122, May 1983.

[2] Stephen T. Frezza and Steven P. Levitan, "SPAR: A Schematic Place and Route System," IEEE Transaction on CAD of IC&S, Vol. 12, No. 7, pp. 956-973, July 1993.

[3] M. May, "Computer-generated Multi-row Schematics," Computer Aided Design, v. 17, no. 1 pp. 25-29, January 1985.

[4] Tzi-cker Chiueh, "HERESY: A Hybrid Approach to Automatic Schematic Generation," Proceedings of the European Conference on Design Automation, pp. 419-423, February 1991.

[5] G.M. Swinkels and Lou Hafer, "Schematic Generation with an Expert System," IEEE Transaction on Computer-Aided Design, Vol. 9, No. 12, pp. 1289-1306, December 1990.

[6] Tsung D. Lee and L. P. McNamee, "Structure Optimization in Logic Schematic Generation," IEEE International Conference on Computer-Aided Design, pp. 330-333, November 1989.

[7] M.L. Ahlstrom, G.D. Hadden and G.R. Stroick, "HAL: A Heuristic Approach to Schematic Generation," ICCAD, pp. 83-86, November 1984.

[8] Mira A. Majewski, Fred N. Krull, Thomas E. Fuhrman, and Paul J. Ainslie, "AUTODRAFT: Automatic Synthesis of Circuit Schematics," ICCAD, pp. 435-438, November 1986.

[9] Lynne D. Green and Jonny Andersen, "Automated Generation of Analog Schematic Diagram," IEEE International Symposium on Circuits and Systems, Vol. 4, pp. 3197-3200, May 1990.

[10] Nam-Hoon Kim, Kyosun Kim, Kyu-Myung Choi, and Jeong-Taek Kong, "RightTopologizer: An Efficient Schematic Generator for Multi-level Optimization," IEEE International ASIC/SOC Conference, pp. 387-391, September 2000.

[11] T. Eschbach, W. Günther, and B. Becker, "Orthogonal Circuit Visualization Improved by Merging the Placement and Routing Phases", Proceedings of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (VLSID'05), pp. 433-438. January 2005.

[12] W. Chen and W. Shiue, "Circuit Schematic Generation and Optimization in VLSI Circuits", The 2004 IEEE Asia-Pacific Conference on Circuits and Systems, pp. 553-556, December 6-9, 2004.

[13] "Debussy Debug System," http://www.novas.com/Products/Debussy, Novas, 2005.

[14] "Synplify Pro," http://www.synplicity.com/products/synplifypro/index.html, Synplicity, 2005.

[15] T. Blanchard, R. Ferreri, and J. Wilmore, "The OpenAccess Coalition -- The Drive to an Open Industry Standard Information Model, API, and Reference Implementation for IC Design Data", Proceedings of the International Symposium on Quality Electronic Design, pp. 69-74, March 18-21, 2002.

[16] "Open Access", http://openeda.si2.org/

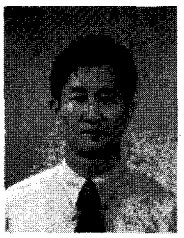
[17] R. E. Tarjan, "Enumeration of the elementary circuits of a graph," SIAM J. Comput., vol. 2, no. 3, pp. 211-216, Sept. 1973.

[18] U. Yoeli, "A Robust Channel Router," IEEE Transactions on Computer-Aided Design, Vol. 10, No. 2, pp. 212-219, February, 1991.

[19] "Synopsys", http://solvnet.synopsys.com

저 자 소 개

이 은 철(학생회원)
2005년 인천대학교 전자공학과 학사 졸업.
2005년~현재 인천대학교 전자공학학과 석사과정.
<주관심분야 : CAD Tools>



김 교 선(정회원)
1986년 연세대학교 전자공학과 학사 졸업.
1988년 연세대학교 전자공학과 석사 졸업.
1998년 Ph.D. Department of Electrical & Computer Engineering, University of Massachusetts, Amherst, U.S.A.
1988년~2003년 삼성전자 CAE Center 주임, 선임, 책임, 수석연구원.
현재 인천대학교 공과대학 전자공학과 조교수
<주관심분야 : 상위수준합성, Reconfigurable Computation, Fault-Tolerance, Embedded Systems, Low-Power Design, Nanoelectronic Architectures>