特輯 技術解說

디지틀 설계 툴

李 哲 東
韓國電子通信研究所 應用技術開發部

Ⅰ. 서 론

전자산업이 발전함에 따라 회로의 규모는 갈수록 증가하고 있다. 하나의 회로를 표현하기 위해서 수백장의 회로도(schematic)가 필요한 경우도 생겨나고 있는데, 이러한 회로의 전반적인 동작을 이해하는 것은 매우 힘든 일이다. 그리고 회로의 규모가 크기 때문에 그의 동작을 브레드보드(breadboard) 상에서 시험하는 것은 것은 거의 불가능하며, 시제품 제작에 많은 비용이 소요되고 있다.

증가하는 회로의 복잡도(complexity)로 인해, 게이트(gate) 수준에서 회로를 그리는 것에서 출발하는 전통적인 설계 방식으로는 대규모 디지틀 회로를 설계하는 것이 매우 힘들게 되었다. 설계가 성공할지라도 막대한 양의 설계 정보를 유지보수하는 것은 더욱 힘든 일이다. 그래서 회로의 동작을 게이트 보다 높은 수준에서 표현한 후, 합성(synthesis)의 과정을 통해 원하는 기능을 실현하는 하드웨어 구조(hardware structure)를 발생하는 새로운 형태의 설계 방식이 개발되었다. 합성을 이용하면 게이트 수준 이상에서 설계의 개념을 표현할 수 있게 되며 또한 그 수준에서 시뮬레이션(simulation)하는 것이 가능하다. 그래서 설계 시간을 크게 단축시킬 수 있을뿐 아니라, 합성 툴이 가지는 성능에 따라서 설계자가 미처 생각하지 못했던 형태의 하드웨어 구조를 만들어 낼 수도 있다. 즉, 설계 영역(design space)을 확장하는 계기가 될 수 있다.

현재 합성은 활발히 연구되고 있으며 어느정도 실용화가 되고 있다. 본 글에서는 디지틀 회로 설계의 새로운 방식인 합성의 의미와 그를 이용한 설계 과정을 간략히 살펴보기로 한다.

Ⅱ. 하드웨어 표현 언어

1. 디지틀 시스템의 계층구조

일반적으로 하드웨어의 표현 영역은 행위(behavior), 구조 (structure) 그리고 물질(physical design)의 3가지로 분류되고 있다. 이들의 의미는 다음과 같이 쓰여질 수 있다.

- 행위 : 어떤 시스템이나 그의 성분이 주위 환경에 작용하는 성질을 말하는 것으로, 입력과 출력간의 대응관계를 말함.
- 구조 : 시스템을 구성하기 위해 결합되어 있는 회로 성분들의 연결관계를 의미. 이 표현은 다음의 물질적 표현과 일치되는 관계를 가짐.
- 물질 : 시스템 내부의 물리적인 구성을 표현.

디지틀 시스템 설계의 계층구조는 컴퓨터 시스템을 예로 들어서 그림 1과 같이 나타낼 수 있다[1]. 그림 1에서 최상위 수준인 PMS는 컴퓨터가 프로세서(processor), 메모리(memory), 스위치(switch)가 결합된 집합임을 의미하는 것으로, 전체적인 시스템의 구조나 정보의 흐름을 나타내고 있어서 시스템 수준이라고 말할 수 있다. 명령어(instruction) 집합 수준에서는 각각의 프로세서에 의해 수행되는 내용에 초점이 맞추어져 있는데, 일반적인 디지틀 회로의 경우에서 이것은 알고리듬(algorithm) 수준에 해당한다. 알고리듬 수준 다음은 레지스터 전송(register-transfer) 수준이다. 디지틀 시스템은 기억 소자들과 기능 블록이 결합된 집합으로 볼 수 있다. 그래서 레지스터 전송 수준에서 회로 행위는 기억 소자들 사이의 연속적인 데이타 전송으로 기술된다. 레지스터 전송 수준과 알고리듬 수준은 내부 구조가 표현된 정도의 차이에 의해 구별된다. 예를들어 알고리듬 수준의 표

현에서는 사용된 변수가 설계의 내부 레지스터와 대응할 필요가 없으며, 그리고 지정문(assignment statement)도 실제적인 데이타 전송과 관련될 필요가 없다.

수 준	영 역		
	Behavior	Structure	Physical
PMS (System)	Communicating Processes	Processors Memories Switches	Cabinets Cables
Instruction Set (Algorithm)	Input-Output	Memory, Ports Processors	Board Floorplan
Register-Transfer	Register Transfers	ALUs, Regs Muxes, Bus	ICs Macro Cells
Logic	Logic Equations	Gates Flip flops	Standard Cell Layout
Circuit	Network Equations	Transistors, Connections	Transistor Layout

그림 1. 컴퓨터 시스템 설계의 계층구조

논리(logic) 수준에서 시스템은 게이트와 플립-플럽(flip-flop)등에 의해 나타나며, 그의 행위는 논리 방정식에 의해 표현된다. 그리고 회로(circuit) 수준에서 시스템은 트랜지스터의 결합체로 표현되며, 그림 1에는 나타나 있지 않지만 소자(device) 수준에서는 트랜지스터의 내부 구조와 그의 특성으로 표현된다.

2. VHDL

컴퓨터 시스템과 같이 대규모 디지틀 회로를 설계할 때 게이트 수준에서부터 시작하는 방법은 설계 시간, 유지 보수등에서 한계에 부닥친다. 그래서 설계의 개념을 보다 높은 수준에서 표현하여 시뮬레이션 할 수 있게 함으로서, 상세한 설계에 들어가지 않고도 회로의 동작을 판단 가능하게 하는 수단이 필요하게 되었다.

이를 위해 미국방성에서는 VHSIC(very high speed integrated circuit) 프로젝트를 수행하기 위해 VHDL (VHSIC hardware description language)이라는 하드웨어 기술 언어를 '81년에 발표하였다. 그 목적은 복잡한 회로를 기술하는 수단 및 VHSIC 프로젝트의 참여자들에게 표준적인 회로 표현 방법을 제공하는 것이었다. 이 언어는 '87년에 IEEE에 의해 표준 언어로 채택 되었으며, 그동안 많은 수정과 개선이 가해졌다. 초기에 있어서 VHDL 표현은 시뮬레이션만 가능하였으나, 최근에 있어서 VHDL은 설계 합성 분야에 많이 이용되고 있다. 그리고 VHDL은 시뮬레이션을 지원하므로, 앞으로 시험성(testability) 향상을 위해서도 이용될 것이다.

VHDL을 이용한 하드웨어 기술은 그의 입출력 인터페이스(interface) 및 내부 표현으로 구성된다. 예를 들어, 지연 시간이 6.0nsec이고, 두개의 입력 i1, i2 및 출력 o1을 가지는 AND 게이트(and2)를 행위 수준에서 VHDL로 모델링(modeling)한 결과는 그림 2와 같다.[9]

```
ENTITY and 2 IS
   PORT (i1, i2:IN BIT, o1:OUT BIT);
END and 2;
ARCHITECTURE behavior OF and 2 IS
   CONSTANT prop-delay:TIME:=6.0 NS;
BEGIN
   01<=i1 AND i2 AFTER prop-delay;
END behavior;
```

그림 2. AND 게이트의 VHDL 모델링

그림 2의 기술에서 대문자로 된 단어들은 VHDL의 예약어(keyword)인데, 그 중에서 ENTITY는 회로도의 심볼에 해당한다. 그리고 ARCHITECTURE는 회로도에 해당하는 것으로서, 회로의 행위나 구조를 표현하기 위해 사용된다. 나머지 기술들은 사용자가 선택할 수 있는 것들이다.

Ⅲ. 합 성

1. 합성의 장점 및 적용 예

하드웨어 표현이 여러 수준에서 이루어질 수 있는 것처럼 합성도 설계의 각 단계에서 이루어질 수 있다. 일반적으로 상위 수준 합성은 디지틀 회로의 알고리듬 수준 행위 표현으로부터 그 행위를 실현하는 레지스터 전송 수준의 하드웨어 구조를 만드는 것을 의미한다. 상위 수준 합성 툴은 입력 사양(specification)으로부터 레지스터, 기능 유니트(unit), 멀티플렉서(multiplexer), 버스(bus) 등의 회로망(network)인 데

이타 패스(path)에 대한 사양과 제어(control)부분에 대한 사양을 발생한다.

상위 수준 합성이 이루어지면 레지스터 전송 수준의 합성이나 논리 수준 합성이 수행되어 사용자가 원하는 설계 기술(design technology)을 이용하는 회로를 생성하게 된다.

회로의 행위가 주어졌을 때, 그를 실현하는 구조는 상당히 많이 존재할 수 있다. 그래서 합성 툴은 설계의 각 단계에서 클럭 사이클(clock cycle), 면적, 전력 소모량등과 같이 사용사가 설정한 제약 조건을 가장 잘 만족하는 하드웨어 구조를 찾아내 준다. 이러한 합성 툴을 설계에 이용함으로써 얻을 수 있는 장점은 다음과 같이 요약될 수 있다.

- 설계 시간 단축 : 설계의 과정들이 점차 자동화됨에 따라 짧은 시간내에 설계를 이룩할 수 있다. 따라서 설계 비용도 감소된다.
- 설계 오류 감소 : 성능이 검증된 합성 툴을 이용하여 설계를 수행하므로 오류가 발생할 여지가 없게 된다.
- 설계 영역 확장 : 합성 툴은 주어진 입력으로부터 그의 기능을 실현하는 하드웨어 구조를 여러개 만들어 주므로, 설계자는 그들에 대하여 비용, 속도, 전력 소모량등의 요소들을 절충하여 구조를 선택하면 된다. 따라서 설계 영역이 넓어지게 되는 큰 장점을 가진다. 만약 설계가 사람의 손으로 이루어지는 경우일지라도, 합성 툴에 의해 만들어진 결과는 여러가지 제약 조건에 대한 출발점을 설계자에게 제안하는 효과를 지니고 있다.
- 설계 과정 기록 : 좋은 합성 툴은 설계의 여러 기로에서 선택의 이유와 그의 효과에 대한 기록을 가지고 있으므로, 설계 과정의 문서화가 자연히 이루어진다.
- IC 제작 기술 보편화 : 반도체 설계의 전문가가 아니라 하더라도 합성 툴을 이용하게 되면 원하는 기능을 가지는 칩(chip)을 가질 수 있게 되므로, IC 제작 기술 보편화가 이루어질 수 있다.

상위 수준 합성의 기본은 '60년대부터 있었는데, 예를들어, ALERT[2]의 경우 APL로 기술된 레지스터 전송 수준의 행위 표현으로부터 논리 수준의 회로를 합성하였다. '70년대에는 설계의 자동화에 대한 많은 노력이 있었지만 주로 레이아웃(layout) 수준에서 이루어졌고, 합성에 대한 연구는 대학에서 학문적으로만 이루어졌다. '70년대 초반에 Carnegie-Mellon에 만들어진 Expl은 설계 영역을 탐사하는 능력을 가졌으며, ISPL이라는 하드웨어 기술언어를 이용하여 표현된 레지스터 전송 수준의 회로 행위를 입력으로 받아들여서 레지스터 전송 수준의 하드웨어 구조를 출력으로 생성하였다.

현재 상위 수준 합성 툴은 논리 합성 툴이나 레이아웃 툴과 결합되어 실제 제작 가능한 결과를 만들어 내고 있다. 우수한 결과는 디지틀 신호 처리와 같은 특별한 분야에서 나타나고 있다. 예를들어 Cathedral[3]과 같은 툴은 복잡한 신호 처리 알고리듬을 구현하는 칩을 다수 제작하였으며, FACE[4]도 실제 설계를 위해 사용되고 있다고 보고되고 있다.

일반적 용도를 가지는 합성 툴인 Yorktown Silicon Compiler를 이용하여 100여개의 명령어를 가지는 801 마이크로프로세서를 합성한 결과가 발표되어 있다.[5] 이 마이크로프로세서는 IBM 3090에서 약 4 CPU 시간만에 완전히 합성되었는데, 레지스터 전송 수준에서 평가한 결과 사람이 수행한 것과 성능은 동일하였고 약 26% 많은 트랜지스터를 사용하였다고 한다. 이때 트랜지스터 수준으로의 합성은 셀 생성기(generator)에 의해 수행되었다.

그리고 CADD/DSL이라는 툴은 MC 68000을 Siemens 7561 상에서 수시간 만에 합성하였다.[6] 이 때 회로의 행위는 상당히 상세하게 표현되었으며, MC 68000을 완전히 기술하는데 소요된 시간은 0.5 MY (man-year) 정도였다. 생성된 회로는 원래의 MC 68000과 상당히 유사하다고 한다.

2. 논리 수준 합성의 과정

논리 수준 합성을 통한 설계 과정을 예로 들어 전통적인 디지틀 회로 설계와 합성의 차이를 살펴보기로 한다. 논리 합성시에 시스템 설계자는 설계 개념을 저수준의 회로연결 정보(netlist) 표현에서부터 회로의 행위를 표현하는 상위 수준의 표현까지 선택하여 사용할 수 있다. 이러한 입력들에 대해 논리 합성 툴은 설계자가 지정한 사양에 따라서 최적화된 회로 연결 정보를 발생한다.

그림 3에서는 일반적 논리 합성 툴의 흐름도를 보여 주고 있다.[7] 그 첫단계는 설계하고자 하는 회로의 아키텍처를 설정하면서 설계 과정을 다루기 좋도록 적당한 크기의 부분 회로로 분할하는 것이다. 즉, 설계 사양이 지시하는 연결도, 데이타 흐름, 그리고 계층구조가 만들어지며, 결국, 합성하기 위한 특정한 기능들이 설정된다. 이러한 기능들을 표현하기 위해 사용되는 언어 수준은 합성의 최종 결과에 상당한 영향

을 미친다. 단순히 게이트들간의 연결관계를 나타내는 표현보다는 회로의 행위를 나타내는 표현이 합성 툴의 관점에서 더욱 요구된다. 그러나 브레드보드나 상세한 회로도에 친숙한 시스템 설계자들은 회로 행위를 표현하는 것에 대하여 부자연스러움을 가질 수 있는데, 이런 경우일지라도 그들 회로의 면적이나 시간 지연(delay)을 감소할 목적으로 논리 합성 툴이 이용될 수 있다.

일단 상위 수준의 언어를 이용하여 회로를 기술하기로 결정하게 되면 설계자가 가지는 선택의 폭은 크게 증가한다. 하드웨어 기술 언어중에서 대표적인 것은 VHDL과 Verilog가 있으며, Boolean 방정식, 진리표(truth table), 입출력 파형, 또는 FSM(finite state machine) 표현등이 이용될 수 있다. 모듈(module)의 특성과 표현의 난이도를 바탕으로 사용될 표현 형식이 결정된다. 논리 합성 툴이 표현 형식의 혼재를 허용하므로 회로내의 각 모듈은 가장 적절한 표현 형식으로 기술된다.

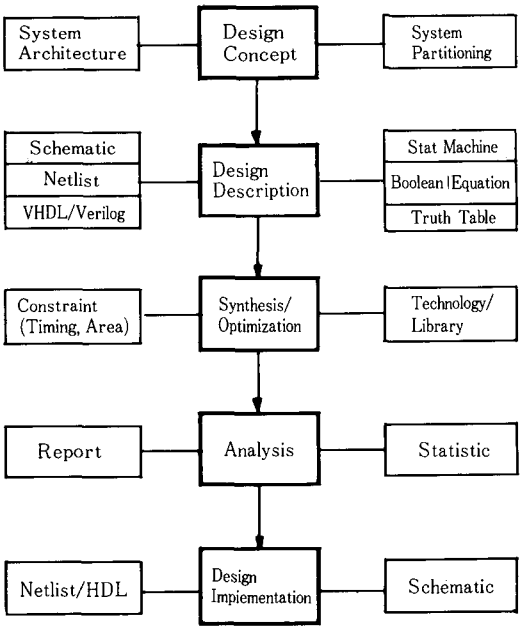

그림 3. 논리 합성의 흐름도

그림 3의 합성 및 최적화 과정에서 설계가 게이트 수준으로 변환되는데, 이 때 설계자는 제약 조건을 설정할 수 있다. 예를들어 입력이 4개 이하인 게이트들만을 이용하여 합성을 하도록 제약할 수 있다. 또한 면적, 시간 지연, 핀의 부하(load), 시험성등의 최적화보다 높은 우선 순위를 가지는 제약 조건들을 설정할 수도 있다. 라이브러리(library)는 설계 사양을 실제 게이트로 대응하는데 있어서 필요한 정보를 제공한다. 그리고 이 과정에서는 설계의 계층 구조 해제, 추상적인 회로 모델을 대상 라이브러리의 성분으로 대응, 내부의 타이밍 분석 기능을 이용하여 임계 경로(critical path) 관찰, 그리고 최적화를 위한 비용 함수(cost function)의 평가와 같은 일을 수행한다. 이중에서 타이밍 해석과 비용 함수 최적화는 논리 합성기가 수행해야 하는 일 중에서 가장 중요하다. 이 과정을 수행한 결과는 시스템의 사양에 맞게 제작된 회로 연결 정보이다.

합성 툴은 제작된 회로 연결 정보를 분석하는 다양한 기능을 제공한다. 예를들어 합성 툴은 회로 연결 정보로부터 회로도를 생성해 줄 뿐 아니라 내부 노우드(node)나 면적/시간 지연값등을 표나 정리된 리스트로 만들어 준다. 생성된 회로도는 설계자에 의해 EWS(engineering workstation) 상에서 편집될 수도 있다. 그래서 최종적으로 생성된 회로 연결 정보는 칩 제작을 위해 ASIC 설계자에게 넘겨지게 된다.

현재 논리 수준 합성을 위해 개발된 상용의 툴들 중에서 대표적인 것은 Synopsys의 Design Compiler, Racal-Redac의 Silcsyn, Mentor Graphics의 Design Consultant, Viewlogic의 VHDL Designer, ISS의 Instant Logic 등이다. 그리고 ASIC 제조 환경(foundry)을 가진 LSI Logic과 VLSI Technology는 LES, ASIC Synthesizer라는 합성 툴을 가지고 있다. 그림 4에서는 이러한 툴의들 입출력 및 사용 언어를 보여준다.

3. 합성 툴의 운용

합성 툴의 목표는 면적과 전력 소모량을 가능한 줄이면서 원하는 기능과 성능을 가지는 회로를 구현하는 것이다. 그리고 합성 툴은 하드웨어의 표현 수준을 가능한 높이고 가장 진보된 반도체 설계 기술을 채택하여야 한다. 그런데 설계 개념의 표현 수준이 점차 높아지는 이러한 추세는 소자 기술의 발전과 잘 부합하여야 한다.

합성 툴에 의해 만들어진 결과는 총 게이트 숫자와 임계 경로의 시간 지연을 줄이므로 우수한 것으로 보일 수 있다. 그러나 이런 결과는 레이아웃이 실제로 제작되었을때 사라질 수 있다. 합성 툴은 보통 회로

회 사(제 품 명)	입 력	출 력	사 용 언 어
Mentor Graphics (Design Consultant)	FSM, HDL, netlist, schematic	netlist, schematic	VHDL
Seattle Silicon (Finesse)	FSM	layout, netlist	Finesse language
Racal-Redac (SilcSyn)	FSM, HDL, netlist	netlist, schematic test vector	VHDL, SilcSyn Design Description Language
Synopsys (Design Compiler)	FSM, HDL, netlist	netlist, schematic	Verilog, VHDL
Viewlogic (VHDL Designer)	FSM, HDL, netlist, schematic	netlist, schematic	VHDL
VLSI Technology (ASIC Synthesizer)	FSM, HDL, netlist	layout, netlist, test vector, schematic	ABEL, Verilog, VHDL

그림 4. 논리 합성 툴

의 면적을 단순히 게이트 숫자만으로 계산하고 있으므로 게이트 사이의 배선을 위해 사용되어지는 면적이 무시되며, 배선을 수행함으로써 발생하는 정전용량(capacitance)이 미치는 부하도 정확히 고려하지 않고 있기 때문이다. 그래서 합성 툴에 의해 만들어진 회로는 ASIC 제작자들에 의해 수정되어야 할 필요가 있다. 이러한 번거로움을 막기 위해서는 합성 툴의 성능을 더욱 개선하거나 합성 툴과 레이아웃 툴을 밀접하게 결합하여 설계를 하여야 한다.

이러한 이유로 합성 툴을 운용하는 데는 세밀한 관심이 필요하다. 논리 합성 툴을 이용하는 일반적인 지침이나 제약은 다음과 같다.[7,8]

- 합성은 모든 설계에 대해 적용될 수 있는 것은 아니다.
  설계 개념이 현실적이어야 하며, 합성 툴은 시스템 설계자를 완전히 대체하는 것이라기 보다는 설계에 도움을 주는 것으로 인식되어야 한다. 그리고 사용자 지침서를 완전히 파악하여 합성 툴을 운용하지 않으면 좋은 결과를 얻기 힘든다.
- 합성의 결과는 설계 표현에 상당히 의존적이다.
  설계 표현이 너무 상세하면 합성의 범위가 좁아지기 때문에 좋지 못하고 반면에 너무 추상적인 것도 좋지 않다.
- 회로의 분할과 구조는 결과에 큰 영향을 미친다.
  합성하는 회로의 한 집합이 너무 작거나 크면 결과가 좋지 않을 수가 있다. 보통 300~3,000개 정도의 논리 게이트에 대해 좋은 결과가 발생한다. 그리고 데이타 패스나 산술(arithmetic) 논리와 같이 회로의 구조가 규칙적이거나 반복되는 형태를 갖는 경우에는 합성의 결과가 좋지 않고, 제어 논리나 FSM의 경우는 우수한 결과가 발생할 수 있다.
- 면적과 시간 지연은 독립적인 것이 아니다.
  면적의 최적화는 게이트 숫자를 줄이는 것으로 얻어지므로, 때때로 시간 지연의 최적화를 얻을 수 있게 한다.
- 레이아웃의 결과는 합성 툴이 예상하는 것과 다를 수 있다.
  임계 경로를 가지는 회로를 설계하는 경우에는 ASIC 제작 환경과 밀접히 관계하면서, 배선이 차지하는 면적 및 배선의 정전용량이 미치는 부하 효과등을 고려하여 합성 툴을 운용하여야 한다.
- 설계의 추적성이 유지된다.
  합성의 최적화 과정에서 회로도가 변경됨으로 인해 특정한 노우드나 신호선의 이름이 사라질 수 있다. 그러나 동작을 검증하기 위해 회로도를 선호하는 설계자들을 위해서, 상위 수준 표현은 이를 위한 추적성을 제공한다.
- 불필요한 투자는 하지 말자.
  대부분의 합성 툴은 매우 고가이므로 정확히 판단하여 불필요한 옵션(option)을 구입하지 않도록 한다.
- 상위 수준 합성은 부자연스러운 설계 방식이다.
  합성을 통한 설계 방식이 필요하다고 인식하였더라도, 시스템 설계자들이 합성 툴을 운용하는

것을 배우기 위해서는 다소의 시간을 투자하여야 한다. 그리고 합성의 결과는 그를 운용하는 경험에 비례할 것이므로 단번에 좋은 결과를 얻지는 못할 것이다. 그러나 설계의 규모가 커질수록 새로운 논리 설계 기법을 배우는데는 많은 투자가 필요하지만, 합성 툴을 운용하는 것을 배우는데 필요한 시간은 완만히 증가할 것이다.

Ⅳ. 설계 검증 및 설계 구현

설계 검증은 합성된 회로가 사양에 맞는 행위를 하는가를 판정하는 일이다. 이러한 판정이 사람에 의해 행하여질 때는, 주어진 입력들에 대하여 초기 사양과 최종 결과가 같은 결과를 갖는 가를 각각 따로 시뮬레이션하여야 한다. 그러나 VHDL이나 Verilog와 같은 하드웨어 기술 언어는 회로의 행위와 구조를 함께 표현하므로 즉, 입력인 행위와 출력인 구조를 동시에 표현하고 있으므로, 합성을 통해 설계를 하는 경우에는 검증의 과정이 간단해진다. 더구나 합성 툴은 항상 같은 알고리듬 또는 절차에 의해 회로를 만드므로 성능이 확인된 합성 툴인 경우에는 설계 검증이 필요없을 수도 있다.

합성의 결과는 반도체 설계자에 의해 칩으로 구현된다. 거의 모든 합성 툴들은 topdown 방식으로 설계를 하므로, 회로 행위나 구조가 추상적으로 표현된 수준들에서 최적화, 스케쥴링 그리고 할당등의 작업을 수행한다. 그래서 아키텍처 수준의 결정이 이루어지기 전까지는 하드웨어 모듈의 특성이나, 레이아웃, 배선 그리고 기생 효과(parasitic effect)등의 설계 구현 요소들이 제대로 고려되지 않을 수 있다. 합성 툴이 물리적 구현의 여러 측면과 관련을 맺지 않을 경우, 현실성이 없는 결과가 발생하기 때문에 레이아웃 설계 툴에 의해 만들어진 여러가지 정보들은 툴로 피이드백(feedback)되어 진다.

Ⅴ. 결 론

최근 논리 합성 툴은 설계자들의 생산성을 높여주는 주요한 요소가 되고 있다. 이러한 툴들이 상위 수준 표현을 받아들여서 최적화 과정을 수행한 후 발생하는 결과는 면적이나 성능면에서 점차 향상되고 있다. 그리고 VHDL과 같은 표준적인 하드웨어 기술언어를 사용하고 합성 기술이 상위 수준으로 발전함에 따라 앞으로 설계자들이 얻을 수 있는 집적도는 더욱 증가할 것이며, 설계 시간이 단축됨에 따라 칩의 종류는 더욱 다양해질 것이다. 이러한 추세로 비추어보아 디지틀 시스템 설계자들은 상위 수준 합성을 통한 설계 방법에 대해 지속적인 관심을 가져야 할 것이다.

參 考 文 獻

[1] M.C. McFarland, A.C. Parker, and R. Camposano, "The high-level synthesis of digital systems," *Proceeding of the IEEE*, vol. 78, no. 2, pp. 301-318, Feb. 1990.

[2] T.D. Friedman and S.C. Yang, "Methods used in an automatic logic design generator (ALERT)," *IEEE Trans. on Computer*, vol. C-18, pp. 593-614, 1969.

[3] S. Note, J. Van Meerbergen, F. Catthoor, and H. De Man, "Automated synthesis of a high speed Cordic algorithm with the Cathedral-III compilation systems," ISCAS '88, New York, pp. 581-584, Jun. 1988.

[4] A.E. Casavant, et. el.,"A Synthesis Environment for Designing DSP Systems," IEEE Design and Test, pp. 35-45, Apr. 1989.

[5] R. Camposano, "Design Process Model in the Yorktown Silicon Compiler," 25th Design Automation Conference, New York, pp. 489-494, Jun. 1988.

[6] R. Camposano and W. Rosenstiel, "'Synthesizing circuits from behavioral descriptions," *IEEE Trans. on CAD*, vol. 8, no, 2, pp. 171-180,, Feb. 1989.

[7] J.P. Paradise, "Logic Synthesis Tools Take the Tedium out of Logic Design," END, pp. 147-154, Sep. 2, 1991.

[8] B. Tuck, "Are Designers' Expectations for Synthesis Realistic?," Computer Design, pp. 110-129, Nov. 1, 1990.

[9] Z. Navabi and S. Day, "Tutorial on Use of VHDL for Description of Digital System," Proceeding of the 1991 ASIC Seminar and Exposition, pp. T12-1.1-T12-1.8, Sep. 1990.

筆 者 紹 介

李 哲 東

1952年 2月 12日生

1977年 2月 경북대학교 전자공학과

1989年 2月 한양대학교 산업대학원 전자공학과(석사)

1977年 2月～1984年 2月 한국전자기술연구소 설계개발실 선임연구원

1984年 2月～1985年 5月 한국전자기술연구소 설계자동화연구실 실장

1985年 5月～1989年 10月 한국전자통신연구소 자동설계기기연구실 실장

1989年 10月～1991年 3月 한국전자통신연구소 자동설계기술개발부 연구위원

1991年 3月～현재 한국전자통신연구소 응용개발부 연구위원

주관심분야 : VLSI설계, VLSI CAD 및 MCAD 등임.