特輯 技術解說

Top-Down Design Synthesis

姜 昌 錄
멘토그래픽스 코리아, 지사장

지금까지 설계자들은 전통적인 bottom-up, 게이트레벨 회로에 기초를 둔 설계방식에 주로 의존해 왔다. 그러나 10만 게이트가 넘는 ASIC등과 같이 설계복잡도가 증가하고, 제품 개발기간의 급속한 단축등의 개발환경 변화는 설계자들에게 기존 설계방식을 포기하도록 강요하고 있다. Top-down 설계는 이와 같은 90년대의 설계요구에 부응하기 위한 방식으로 현재 많이 채택 되어지고 있다.

Top-down 설계는 공통적으로 VHDL을 기본으로 하여 진행될 것이다. 설계스펙은 VHDL을 이용하여 주로 정의하고 시뮬레이션을 통해 검증한다. 이를 통해 회로의 동작이 검증되면 합성 툴을 이용하여 게이트레벨의 회로가 완성된다. 이 방식은 과거 bottom-up 설계과정 중 가장 지루하고 많은 에러를 내포하는 단계를 자동화하므로 단시간에 최적화된 회로를 설계할 수 있다는 장점이 있다. 게이트레벨 설계방식으로 하루에 보통 50여개 정도의 게이트를 설계할 수 있는 설계자인 경우 top-down 방식의 설계를 이용할 경우 200여개 정도의 게이트를 설계할 수 있을 것이다. 이는 주로 자동화된 합성 툴의 덕택이며 또한 C나 Pascal이 소프트웨어 엔지니어에게 준 영향과 비슷하게 하드웨어 엔지니어에게도 영향을 줄 것이다.

Ⅰ. AutoLogic VHDL－멘토의 하이레벨 합성 툴

AutoLogic VHDL은 멘토그래픽스가 제공하는 top-down VHDL 합성 툴이다. 이는 설계자가 게이트레벨보다는 회로의 기능에 중점을 둔 설계를 가능하게 해준다. 또한 technology independent한 설계방식을 지원하여 설계자가 아키텍쳐 설계시 구체적인 technology와 무관하게 설계하며 최종 실현단계에서 ASIC, FPGA, PLD, IC등 구체적인 technology를 지정할 수 있다.

Top-down 설계의 다른 특징은 설계 스펙 변화시 용이하게 대처할 수 있다는 점이다. 게이트레벨의 설계과정시 시스템 스펙의 변화는 전체 개발시간에 큰 영향을 주는데 반해 VHDL 합성등을 이용한 top-down 설계방식은 신속하게 스펙의 변화를 수용하여 검증할 수 있다. VHDL과 같은 하드웨어 설계언어에 기초를 둔 설계과정은 기본적으로 프로젝트를 문서화하므로 기록등을 위해서도 유용하다.

Ⅱ. Top-Down 설계방식

Top-down 설계의 시작은 VHDL과 같은 하드웨어 기술언어를 이용하여 설계하고자 하는 시스템의 아키텍쳐를 생성하는 일이다.

이 단계에서는 설계의 기능 및 스펙에 중점을 둔다. 그림 1은 VHDL을 이용한 top-down 설계의 흐름도를 보여주고 있다. 일단 생성된 VHDL은 컴파일을 한다.

멘토그랙픽스는 그림 2와 같이 한번의 컴파일로 시뮬레이션과 합성을 동시에 할수 있는 환경을 제공하므로 게이트 레벨의 합성을 위해 VHDL을 다시 코딩하는 번거로움을 줄여 준다. 컴파일이 끝난 후 VHDL 스펙의 기능을 검증하기 위해 시뮬레이션을 한다.

만약 시뮬레이션시 에러가 발견되면 VHDL을 다시 코딩한 후 반복한다. 이러한 VHDL 코드생성-컴파일-시뮬레이션의 순환과정은 초기 아키텍쳐 레벨의 VHDL 모델의 검증을 위한 기본적인 단계이다. VHDL 모델의 기능이 시뮬레이션을 통해 검증된 후 AutoLogic VHDL을

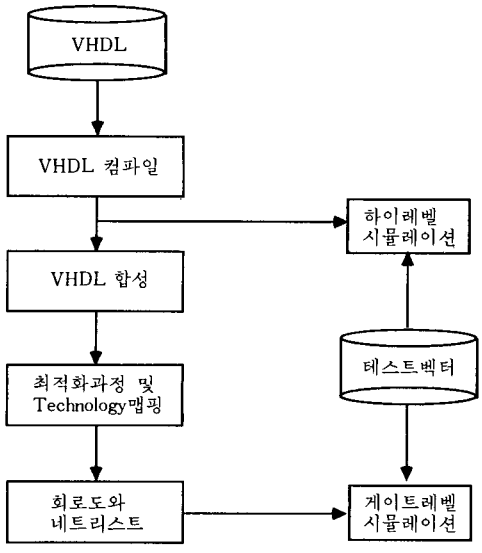


그림 1. Top-down 설계흐름도

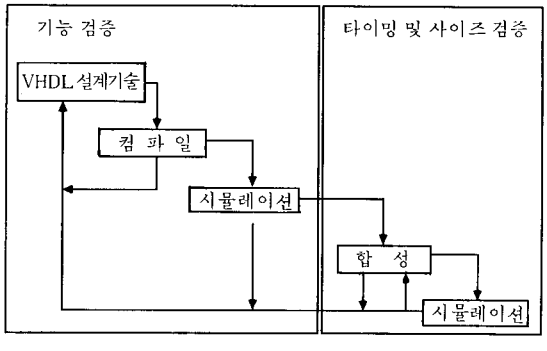


그림 2. AutoLogic VHDL을 이용한 top-down 설계

Ⅲ. AutoLogic VHDL 합성과정

AutoLogic VHDL을 이용한 합성과정은 다음과 같다.

1. 하이레벨 최적화 과정

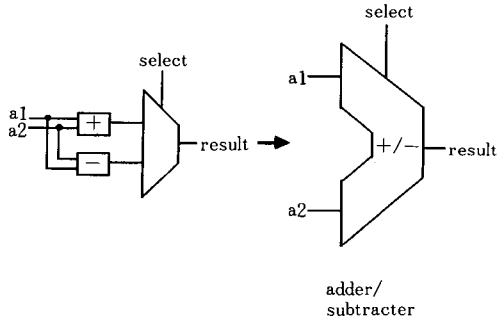
VHDL과 같은 하드웨어 기술언어를 합성할 때의 잇점 중의 하나는 합성시 효율적인 회로를 생성하기 위해 하이레벨의 최적화 과정을 할 수 있다는 점이다. 이는 게이트레벨의 논리합성과는 달리 전체 시스템 아키텍쳐를 최적화시키므로 결과적으로 최종 회로의 게이트 수를 크게 줄일 수 있다. AutoLogic VHDL은 여러종류의 하이레벨 최적화 알고리즘을 이용하여 효율적인 technology independent 설계를 생성시킨다. 다음은 최적화 과정시 사용되는 여러 기법 중 일부를 설명한 것이다.

1) Resourse sharing/Allocation

한 동작 단위에 대해 여러 연산이 수행될 때 각 연산별로 별도의 단위(unit)가 할당되지 않고 전체 동작 단위에 대해 하나만 할당된다.

예를 들면,

```
IF Selection='1';
   THEN result<=a1+a2;
   ELSE result<=a1-a2;
END IF;
```



이용하여 논리합성을 한다.

일단 논리합성 후 여러 최적화 알고리즘을 이용하여 목적하는 technology에 맞게 technology 맵핑을 한다. 논리 최적화는 주로 게이트 수, 시간, 임계경로등을 고려하게 된다. 이런 과정을 걸쳐 생성된 최종 논리 회로는 게이트레벨의 시뮬레이션을 다시 수행하여 시스템의 기능 및 지연등이 적절하게 설계되었는지를 검증한다.

2) Bit-Pruning

시그날로 정의된 정수의 범위에 따라 정수연산의 폭을 줄일 수 있다.

예를들어

```
Signal a,b:Integer Range -64 to 63;
Signal c  :Integer Range -8 to 7;
c<=a+b;
```

신호 c의 범위는 −8에서 7이므로 4bit만이 필요하다. 즉, 합성된 덧셈 연산기는 a,b를 처리하기 위해 7bit가 필요한 것이 아니고 4bit adder만이 필요하다.

3) Comparator sharing

같은 피연산자(operand)를 가지는 비교연산동작을 하나의 다기능 연산자에 복합 부여할 수 있다.

4) Function transformation

하나의 연산동작을 기능적으로 동일하며 보다 효율이 높은 다른 연산동작으로 변환시킨다. 예로써 2를 곱하는 동작을 shift-left 동작으로 변환하는 동작을 들 수 있다.

5) Constant folding

입력변수로써 상수를 취하는 연산자들은 그 연산동작을 수행 후 회수값(return value)으로 치환된다.

6) Constant propagation

하나의 변수에 상수값이 인가(assign)되면 그 인가동작에 연관된 변수 사용시 인가된 상수값이 사용된다.

7) Expression sharing

동일한 수식이 반복 수행된 경우 그 수행을 하지 않는다.

8) Expression migration

반복구문(loop)안에 그 반복 구문 수행과 관계없는 수식이 들어있는 경우 그 수식을 반복구문 밖으로 이동시킨다. 예를 들어 복수개의 IF 혹은 CASE 구문에 동일한 수식이 사용된 경우 이 수식을 IF나 CASE 구문 밖으로 이동시킨다.

9) Dead code removal

IF 혹은 CASE 구문에 의해 조건부로 수행될 구문중 논리상 수행이 불가능한 부분을 제거한다.

2. 마이크로 아키텍쳐 선택

AutoLogic VHDL은 하나의 VHDL model로부터 복수개의 아키텍쳐를 합성해 낼 수 있다. 속도/면적의 최적화 곡선상의 여러가지 제한조건(constant)들을 지정함으로써 사용자는 tool로 하여금 여러가지 제한조건을 만족하는 복수개의 회로를 구현하도록 할 수 있다. 이것은 그림 3과 같다.

속도/면적에 관한 제한조건들을 이용하여 회로합성 tool의 모듈 발생기는 전체 회로에 필요한 여러 종류의 논리연산용 블록들을 생성해 낸다. 생성된 모듈들의 평행성(parallelism)은 제한조건들에 따라 결정된다. 제한조건들은 전체 설계 중의 일부분에 각각 적용시킬 수도 있고 전체 설계에 공통적으로 적용시킬 수도 있다.

이와 같은 기능을 이용함으로써 설계자가 여러가지 구현된 아키텍쳐 중 최적의 것을 선택할 수 있을 뿐 아니라

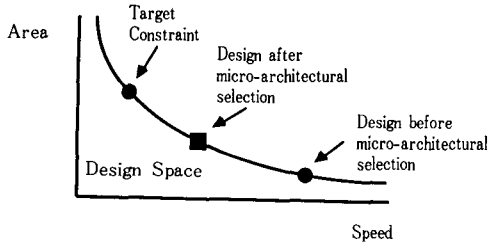


그림 3. 속도/면적 최적화 곡선

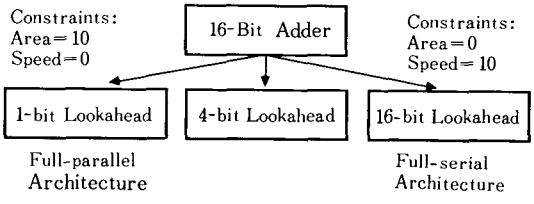


그림 4. 속도/면적 최적화 곡선에 따른 모듈 생성 예

최적화(optimization) tool이 회로 전체에 대한 제한조건들을 손쉽게 만족시킬 수 있도록 도와줄 수 있다.

VHDL 코드를 직접 ASIC의 게이트들로 이루어진 회로로 변환시키고 최적화를 수행시키는 것보다 일단 구현가능한 아키텍쳐들을 생성하여 그들중 가장 최적화 가능성이 높은 것을 선택하여 게이트 단계로 구현시킴으로써 최적화 작업을 쉽고 신속하게 수행시킬 수 있으며 보다 효율이 높은 최종회로를 생성시킬 수 있다. 아키텍쳐 설계단계에서의 제한조건에 따른 최적화 과정을 거치지 않을 경우 최적화 tool이 최정 게이트레벨 구현과정에서 모든 최적화 과정을 수행하여야 하기 때문에 지나치게 많은 시간이 걸릴 수 있으며 어떤 경우에는 제한조건을 만족시키지 못할 수가 있다.

3. Finite State Machine Synthesis

사용자의 지시에 따라 AutoLogic VHDL은 FSM에 관한 여러가지 상태부여(encoding/state assignment) 방식을 사용할 수 있다.

- ○ 순차적 방식:각 상태에서 최소한의 비트 수를 사용하여 순차적인 코드를 부여하는 방식
- ○ 그레이코드 방식:각 상태의 순차적 전이시 하나의 비트만이 변하도록 각 상태에 코드를 부여하는 방식

○ One hot 방식:전 상태에 해당하는 길이를 가지는 2진수 벡터에 상태를 할당하는 방식. 각 상태에 해당하는 부분의 비트만 1이 되고 나머지 비트는 0이 된다.

○ Random 방식:각 상태에 해당하는 코드를 무작위로 부여하는 방식

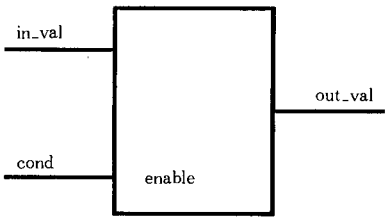
○ 변환 최적화 방식:상태가 무작위로 전이됨에 따라 변경되는 비트의 수가 최소가 되도록 코드를 부여하는 방식

4. 래치추정

회로합성 과정 중, VHDL 모델을 분석하여 변수와 시그날 변수가 실제 저장소자(physical storage device)를 필요로 하는 지를 결정하게 된다. 이와 같은 작업을 래치 추정이라 부른다. 예를 들면

```
Process(in_val, cond)
Begin
        If(cond='1') THEN
            out_val<=in_val;
        END IF;
END PROCESS;
```

위와 같은 구문에서 cond 신호가 '1'이면 in_val에 인가된 신호가 out_val로 출력된다. Cond 신호가 '1'이 아니면 out_val의 현재 값은 저장될 수 있어야 한다. 이 말은 입력이 in_val, 출력이 out_val이고 cond에 의해 동작되는 투명 래치(transparent latch)를 필요로 한다는 의미가 된다.



AutoLogic VHDL은 래치추정의 결과를 사용자에게 알려주며 따라서 VHDL 코드와 합성 출력된 저장 소자와의 관계를 명확히 알 수 있도록 해 준다.

5. AutoLogic VHDL로 부터의 Feedback

회로 합성 과정 중 어떻게 연산자가 합성되며 또는 복수의 연산기능이 하나의 복합연산자에 의해 공유되는 지에 대한 정보가 사용자에게 주어진다.

또한 저장 소자(storage element)가 사용되는 경우나 변수에 저장 소자가 추정 할당되는 지에 관한 정보도 사용자에게 주어진다.

○ 연산자:AutoLogic VHDL은 연산, 관계연산 및 시프트 연산자들을 사용자가 지정한 파라미터에 따라 자동 생성하는 기능 모듈 자동 생성 tool을 가지고 있다.

○ 타이밍 문제:마이크로 아키텍쳐 선택 단계 및 최종 게이트 구현 단계에서 타이밍 제한 조건을 설정하여 회로를 합성할 수 있다.

여기서 VHDL로 기술한 아키텍쳐는 다음과 같이 다섯개의 부분으로 나뉘어져 있다.

○ 데이타 입력을 이네이블하기 위한 래치

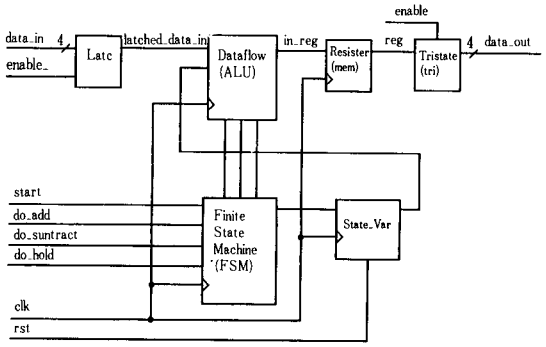
○ 4개의 상태를 가진 finite state machine

○ 데이타플로우의 동작을 기술하는 ALU

○ ALU에서 계산된 결과를 저장할 레지스터

○ 버스 시그날로 데이타를 내놓는 트라이스테이트

VHDL 합성으로 구성된 아키텍쳐의 블럭 다이아그램을 다음에 보여주고 있다.



Ⅳ. 결 론

설계가 점점 복잡해지고 시장에 제품을 내놓기까지 걸리는 시간이 중요해짐에 따라 90년대의 설계자들은 그들의 일을 쉽게 해줄 EDA 툴을 기대하고 있다. 논리회로 합성은 증가하는 복잡도와 시장에 제품을 내놓는 시간을 짧게 하는 것에 대응하여 극적으로 생산성을 향상시켜 줌으로써 미래 전자산업의 가장 중요한 부분이 될 것이다. 통합된 설계환경에 들어있는 VHDL 합성 툴은 설계자가

가능에 초점을 두고 서로 다른 제약하의 여러 아키텍쳐를 만들어 볼 수 있게 하여 준다.

상위 수준에서의 최적화는 하위 수준에서의 최적화에 의존하지 않으면서 더욱 더 효율적인 설계를 가능하게 한다. 또한 합성 툴의 테크날로지 맵핑 가능을 이용하여 설계한 회로는 ASIC, FPGA, PLD 또는 IC로도 구현하도록 할 수가 있다.

筆 者 紹 介

姜 昌 錄

1951年 7月 6日生

1974年 2月 한양대학교 공과대학 공업경영학과 졸업

1978年 3月～1980年 11月 금호실업

1980年 12月～1984年 9月 콘트롤 데이타 코리아(주)

1984年 9月～1987年 4月 서울 일렉트론(주) CAD/CAM 사업부

1987年 5月～현재 멘토그래픽스 코리아, 지사장

주관심 분야: CAE/CAD 분야