論文 92-29A-9-10

디지틀 신호처리용 실리콘 컴파일러를 위한 사용자 툴 개발

(The Development of the User Interface Tool for DSP Silicon Compiler)

李文基* 張虎郎* 金鍾鉉* 李承浩* 李光燁*

(Moon Key Lee, Ho Rang Jang, Jong Hyun Kim, Seung Ho Lee, and Kwang Youb Lee)

要 約

본 논문에서는 시스템 설계자가 디지틀 신호 처리용 시스템을 기술한 시스템 기술 언어를 입력으로 하여 셀 라이브러리에서 제공하는 셀들의 연결관계로 변환하는 시스템 기술 언어 번역기를 개발하였다. 시스템 설계자는 시스템을 기술한 다음 시스템의 동작을 실제 레이아웃을 행하기 전에 검증할 필요가 있다. 본 논문에서는 이를 위하여 이벤트 드리븐 방식을 채택하고 C++ 언어에서 제공하는 다중 처리 라이브러리를 이용하여 시스템 검증기를 개발하였다. 시스템의 동작이 시스템 검증기에 의하여 검증되면 실제 레이아웃을 행하게 된다. 본 논문에서는 실제 레이아웃에 이용되는 셀들을 셀라이브러리에서 제공하는 레이아웃들로 자동생성하는 모듈발생기를 개발하였으며 이 모듈 발생기의 결과는 위치설정기의 입력으로 이용되어 각 셀의 위치를 설정한다. 위치 설정기는 Min-Cut 알고리듬과 시뮬레이티드 어닐링 알고리듬을 이용하여 C++ 언어로 개발하였다. 위치 설정이 이루어지면 다른 툴들이 수행되어 집적회로의 레이아웃 화일이 생성된다.

Abstract

The DSP silicon compiler consists of language compiler, module generator, placement tool, router, layout generation tools, and simulator. In this paper, The language compiler, the module generator, placement tool, and simulator were developed and provided for the system designer. The language compiler translates the designer's system description language into the intermediate form file. The intermediate form file expresses the interconnections and specifications of the cells in the cell library. The simulator was developed and provided for the behavioral verification of the DSP system. For its implementation, the event-driven technique and the C++ task library was used. The module generator was developed for the layout of the verified DSP system, and generates the functional block to be used in the DSP chip. And then the placement tool determines the appropriate positions of the cells in the DSP chip. In this paper, the placement tool was implemented by Min-Cut and Simulated Annealing algorithm. The placement process can be controlled by the several conditions input by the system designer.

*正會員, 延世大學校 電子工學科
(Dept. of Elec. Eng., Younsei Univ.)

接受日字 : 1991年 12月 26日
(※이 논문은 상공부 공업기반 기술과제에 의한 연구 결과임.)

I. 서 론

반도체 관련 기술 발달로 집적회로의 복잡성이 증가함에 따라 집적 회로의 설계시간도 기하급수적으로 증

가하여 칩의 유용시간(Life Time) 보다 더 증가하는 문제점이 발생하였다. 최근에는 이런 문제점을 해결하기 위하여 실리콘 컴파일 방식을 취하는 추세에 있다.

실리콘 컴파일러는 설계의 한 레벨에서 다른 레벨로, 또는 같은 레벨에서 서로 다른 표현으로 변환시켜 주는 번역기라고 할 수 있다. 여기서의 레벨이란 제품의 명세와 같은 추상적이거나 칩의 마스크와 같이 구체적일 수도 있다. 표현은 시스템을 정의하는 방법으로 표현하는 레벨에 따라 동작적(Behavioral), 기능적(Functional), 구조적(Structural), 물리적(Physical)인 것 일 수 있다. 일반적으로 실리콘 컴파일러는 높은 레벨에서 레이아웃을 위한 자료를 자동적으로 생성하는 것을 말한다.

실리콘 컴파일러는 비교적 실리콘의 면적과 직접회로의 성능에는 아직 미흡하지만 설계시간의 단축이라는 면에서는 탁월한 효과를 내고있다.

본 논문에서는 비교적 실리콘 컴파일 기법에 적합한 디지틀 신호처리 분야에 이용되는 실리콘 컴파일러의 사용자 툴들을 개발하였다.

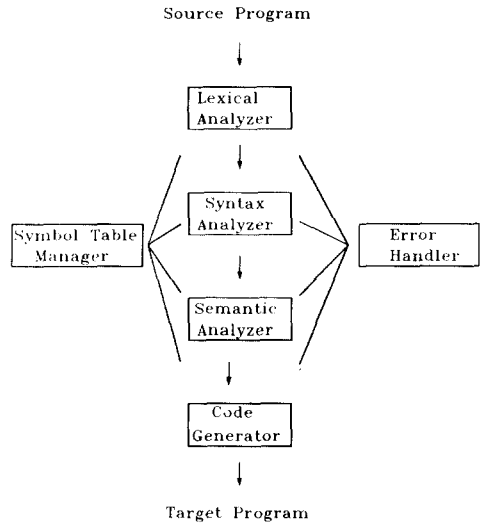


그림 1. 시스템 기술 언어 번역기
Fig. 1. Language compiler.

Ⅱ. 본　론

1. 시스템 기술 언어 번역기

시스템 기술 언어 번역기는 설계자가 기술한 시스템 기술언어를 입력으로 받아들여 각 셀들의 연결관계로 변환하는 역활을 한다.

설계자가 기술한 시스템 기술 언어를 실제로 수행하기 위해서는 여러 단계를 거친다. 사용자가 기술한 시스템 기술 언어 화일은 우선 전처리(Preprocess) 단계를 거치게 되며 이 단계에서 화일의 포함이나 그 밖에 컴파일 시에 필요한 기본적인 일들을 수행한다. 본 논문에서는 C 언어의 전처리 툴을 이용함으로 C 언어에서 제공하는 모든 전처리 명령어를 제공한다. 전처리가 끝난 화일은 정의된 언어의 문법으로만 이루어진 화일이 되며 이어서 비로서 컴파일이 행하여 진다. 컴파일 단계를 거친 이후에는 특정 컴퓨터를 위한 명령들로 이루어진 화일이 된다. 이 화일은 어셈블러를 통하여 특정 컴퓨터의 기계어로써 변환이 되고 이 데이타들을 메모리에 저장한 다음 각 명령어에 해당하는 일들을 컴퓨터가 수행을 하게 된다.

시스템 기술 언어 번역기는 그림 1에서와 같이 여러 단계로 구성되며 일반적으로 몇 개의 단계가 하나의 프로그램으로 된다. 시스템 기술언어 화일은 그림 1과 같은 여러 단계를 거치면서 그것의 의미가 해석되어 특정 컴퓨터에 적합한 언어로써 번역이 된다. 어휘 분석기(Lexical analyzer)는 시스템 기술 언어의 단어가 주어진 규칙에 맞게 쓰여졌는 지를 검사한 다음 그것의 종류(수, 이름, 예약어,…)를 해석하여 그 결과를 문법 분석기(Syntax analyzer)에게 출력한다. 문법 분석기는 어휘 분석기로부터 받은 결과들을 해석하여 시스템 기술 언어가 정의한 문법에 맞는 지를 조사한 다음 의미 분석기(Semantic analyzer)에게 결과를 출력한다. 문법 분석기는 문법적 오류에 대해서 그것에 합당한 에러 메세지를 출력하게 된다. 의미 분석기는 시스템 기술 언어의 의미가 무엇인지를 해석하여 그것이 적절한 표현인지를 조사한다. 이런 일련의 단계가 완료된 다음 시스템 기술 언어 번역기는 어셈블리 언어를 발생한다. 각 단계에 필요한 기본 정보는 심볼 테이블로써 관리를 하며 이것은 각 단계를 거치면서 그 내용이 처리된다. 본 논문에서는 어휘 분석기와 문법 분석기를 UNIX 상에서 lex와 yacc 유틸리티를 이용하여 각각 완성하였다.

본 논문의 시스템 기술 언어는 직접회로의 계층적 구조를 지원하며 또한 직접회로 상호간의 연결을 위하여 시스템 기술 언어에서 각 입출력 패드의 위치를 설계자가 설정할 수 있도록 하였다.

시스템 기술 언어 번역기가 분석한 어셈블리 언어는 메모리를 적게 쓰는 스택 구조를 이용하는 가상적인 컴퓨터의 언어이다. 가상적 컴퓨터는 컴퓨터의 동작을 하나의 부 프로그램으로 표현한 것이며 실제의 컴퓨터

처럼 동작하여 집적회로의 설계시에 필요한 중간 형식 화일을 만들어낸다.

이와 같이 언어번역기를 여러 단계로 구분한 것은 차후 기능 첨가나 문법 첨가에 있어서 큰 어려움을 없애기 위해서이다. 문법의 변화는 yacc 의 입력 화일의 형태을 바꾸고 변화된 부분에 해당하는 루틴들을 첨부함으로써 해결할 수 있다. 나머지 기존의 루틴들은 변하지 않고 이용될 수 있다.

본 논문에서 제안한 시스템 기술 언어 번역기는 일반 CAD 툴에서와는 달리 중간에 어셈블러 언어를 채택하였다. 이는 시스템 기술 언어를 분석하면서 모든 정보를 메모리에 저장하여 이를 중간형식화일을 생성하는 것에 비하여 메모리 사용면이나 여러개의 설계 단위를 분석할 때에 비교적 적은 메모리와 빠른 속도로 분석할 수 있는 장점이 있다.

2. 시스템 검증기

시스템 검증은 시간 변화에 따르는 시스템의 동작을 예측하기 위하여 수행하는 과정이다.

시스템 검증을 행하는 방법은 우선 회로의 각 요소들을 모델링한 다음 그 요소들 간의 시간 지연을 계산하여 이루어 진다. 회로의 요소를 어떻게 모델링하는가에 따라서 검증기는 여러가지로 나누어진다. 아나로그 신호를 해석하기 위하여 각 회로 요소들을 트랜지스터의 레벨로 모델링하는 것을 회로 레벨 검증기라고 한다. 이것은 많은 계산 시간을 요구하므로 큰 시스템의 검증에는 이용되기가 어렵다. 그래서 여러개의 물리적 회로 요소들을 하나로 엮어 하나의 회로 요소로 모델링을 한다. 그 모델링 정도에 따라 로직 레벨(Logic Level), 기능 및 동작 레벨(Functional, Behavioral Level)의 검증기로 구분된다.

본 논문에서는 설계자가 디지틀 신호처리용 실리콘 컴파일러에 기술한 시스템의 동작이 의도한 바와 같이 동작하는가를 검증하기 위하여 셀의 동작 기술 방법에 따라 로직 레벨에서 동작 레벨까지를 지원할 수 있도록 시스템 검증기를 개발하였다. 본 논문의 시스템 검증기는 시스템의 동작적 측면을 검증하는 데 중점을 두고 있다. 실제적 동작의 검증은 완전한 레이아웃이 된 상태에서 회로를 축출하여 기타 다른 검증기(예: SPICE)로 확인하도록 하였다.

본 논문에서의 시스템 검증기는 셀의 모델링 레벨에 따라 아나로그 입력(실수형 자료) 또는 디지틀 입력(0, 1, X)을 입력으로 받고 출력하며 각 회로 요소들은 셀 라이브러리에서 제공하는 셀들의 동작으로 모델링 된다. 회로 요소들은 또한 독립된 프로세서(독립된 하나의 프로그램)로 모델링되어 가상적으로 병렬 수행된다. 이렇게 각 셀들을 독립적인 프로세서로 모델링해 둠으로써 차후 셀의 증가가 있을 시에는 셀에 대한 모델링을 첨가하여 그것만을 재컴파일하여 두면 되므로 쉽게 셀의 첨가가 이루어 지도록 하였다. 시스템 검증기에서는 회로 수준의 검증기에서 제공하는 전력 소비, 정확한 시간 지연 등은 검증하기 어렵다. 그러나 각 셀들은 이미 회로 레벨에서 검증이 되어 있고 이 결과로 각 셀의 동작을 모델링하므로 시스템의 기능을 검증하는 데 충분하다.

본 논문에서는 사용자의 시스템 기술언어를 시스템 기술 언어 번역기가 처리하여 각 기능셀 간의 연결관계를 표현한 중간형식화일을 생성한다. 시스템 검증기는 이 중간 형식 화일을 입력으로 하여 각 셀들의 연결관계 및 출력해야할 신호 네트에 관한 정보를 얻어 시스템 검증을 수행한다. 입력 신호에 관한 값은 따로 입력화일을 읽어 수행하며 입력 화일의 형식은 기본적으로 시간과 각 입력 노드에 관한 값으로 주어진다.

본 논문의 시스템 검증기는 중간 형식 화일을 그에 상응하는 C++ 언어 프로그램으로 변환하여 각 셀들의 기능을 기술한 프로그램과 함께 컴파일하여 UNIX 상에서 실행하도록 설계되었다. 그림 2는 시스템 검증을 위한 흐름과 각 흐름에 필요한 자료를 나타낸 것이다. 시스템 검증기의 기본 알고리듬은 값이 변한 노드를 시간의 순서에 따라 연결하여 그에 해당하는 프로세스를 행하는 이벤트 드리븐(event-driven) 방식을 채택하였으며 이를 객체지향 언어인 C++ 언어에서 제공하는 다중 프로그래밍(Multi-Programing : Task Library) 기법을 이용하여 완성하였다.

3. 모듈 발생기

종래의 집적 회로 설계 방식에서는 필요한 기능을

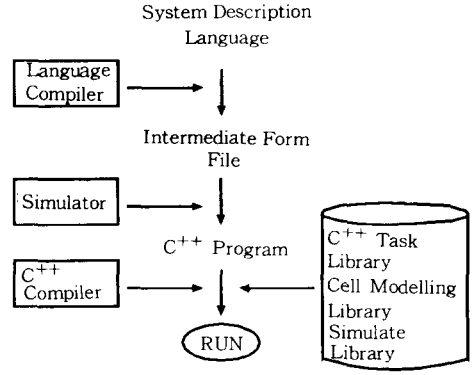


그림 2. 시스템 검증기의 수행 절차
Fig. 2. Simulation.

하는 모듈을 만들기 위해서 직접 그 셀을 처음부터 설계하여 이용하거나 이미 만들어 놓은 기본셀들을 가지고 설계자가 직접 레이아웃하여 이용하였다. 하지만 이 방법은 다른 복잡도를 가지며 같은 기능을 하는 모듈을 설계할 때도 설계자가 처음부터 다시 설계해야하는 단점이 있다.

많은 모듈들은 구조적인 규칙성을 가지고 있으며 이들은 기본 셀들이나 여러셀들에 의해 상대적 위치에 따른 어떤 형태(가로 혹은 세로)로 만들어질 수 있다. 이런 형태로는 8비트, 16비트와 같이 상이한 복잡도를 갖는 덧셈기, 곱셈기 등이 있다. 특성 기능을 하는 모듈을 실제적으로 레이아웃 해 두는 것이 아니라 그 레이아웃 요소들을 상대적인 위치 관계로 표현 해둠으로써 종래의 설계방식의 문제점을 해결할 수 있다. 이 상대적 위치는 모듈을 정의하는 매개변수에 의하여 실제적 위치로 변환되어 레이아웃이 생성된다. 이렇게 함으로써 같은 상대적 위치 관계를 갖는 모듈들은 그 복잡도에 따라 새로 설계할 필요가 없다. 예를 들면 4비트의 곱셈기를 설계하거나 10비트의 곱셈기를 설계할 때 종래의 방법은 각각 따로 설계를 해야하며 각 모듈의 특성이나 검증 또한 따로 해야한다. 그러므로 중복되는 일을 여러번 해야하는 단점이 생긴다. 하지만 곱셈기를 한 비트 곱셈기 셀들의 상대적 위치 관계로 기술해 두면 그 비트 수에 따라서 한 비트 곱셈기의 각 셀을 실제 위치에 위치시키므로써 레이아웃을 할 수 있다. 이 방법은 각 비트 수에 따라 새로 설계하는 번거러움을 줄일 뿐 아니라 모듈의 성능도 간단히 계산되어 질 수 있다. 이러한 일을 하나의 프로그램으로 처리할 수 있으며 이런 프로그램을 모듈발생기라고 한다.

본 논문에서는 UNIX 시스템에서 제공하는 다중 처리 기법을 이용하여 C++ 언어로 모듈발생기를 개발하였다.

모듈 발생기는 시스템 기술 언어 번역기가 발생한 중간 형식 화일을 읽어 집적 회로를 만드는 데에 필요한 각 셀들의 레이아웃을 발생한다. 이 셀들은 셀 이름, 셀의 처리 비트 수, 그리고 지연 시간 등을 정의하는 매개변수로 결정된다. 이 매개 변수는 셀을 만드는데 이용되는 중요한 정보이며, 이에 따라 셀의 크기와 입출력 단자의 위치가 바뀌게 된다.

모듈 발생기는 중간 형식 화일로부터 각 셀에 관한 정보를 받아들여 그것에 해당하는 셀의 레이아웃을 발생하는 독립된 프로그램을 동작 시킨다. 한번 발생시킨 셀은 다시 발생시키지 않도록 하기 위하여 이미 발생한 셀의 이름과 매개변수를 Tree 구조로 저장하고, 다음 셀의 정보가 입력 되었을 때 그것이 이미 발생한 셀인 지를 조사하여 이미 생성된 셀이면 다시 발생시키지 않는다. 즉, 모듈발생기는 각 셀들을 발생시키는 프로그램을 하위 프로세서로 관리하여 이것의 결과를 파이프로 받아들여 처리한다. 그림 3은 모듈 발생기와 그 하위의 각 셀들을 생성하는 프로세서 간의 관계를 도식화한 것이다. 이렇게 각 셀의 생성을 위한 프로그램을 하위 프로세서로 관리함으로써 새로운 셀의 첨가나 갱신 시에 여러가지 편리한 점을 제공한다. 각 셀을 생성하는 것을 각각의 독립된 프로그램으로 처리함으로써 셀의 첨가나 갱신이 이루어진다 하더라도 모듈 생성기의 모든 부분을 재 컴파일할 필요가 없다. 새로운 셀이나 갱신된 셀을 발생하는 프로그램만을 재작성하여 이용하면 된다. 모듈 발생기는 각 셀을 발생하는 프로그램을 내부에서 하나의 하위 프로세서로 실행시켜서 셀의 물리적 레이아웃을 얻을 수 있다. 하위 프로세서를 실행시킨 후 그 결과는 UNIX 시스템이 제공하는 자료 파이프를 열어서 입력으로 받아들여 이용한다.

다음은 모듈 발생기의 기본 알고리듬이다.

```
while (input_file. eof() )
{
      read_a_cell(input_file, cell)
      if (! is_in_tree(cell, tree))E
      {
          child_pocess = run_child_process(cell)
          result = get_cell(child_process)
          insert_tree(result, tree)
      }
      else
       result = get_from_tree(tree, cell)
      output(output
_file, result)
}
```

모듈 발생기의 기본 알고리듬

에러 복구는 오류가 발견되는 셀을 무시하는 방법으로 수행한다. 이것은 오류가 발견될 경우 세미 콜론을 만날 때까지 입력 문자들을 건너뛰는 것으로 간단히 행하여 진다.

각 셀을 발생하는 프로세서는 셀의 매개 변수를 받아서 실제적 레이아웃을 생성한다. 이것은 CIF 형태의 화일을 발생하게 되므로 CIF 화일의 각 레이아웃 요소를 발생하는 루틴들을 라이브러리로 만들어 놓아서 각 셀을 발생하는 프로그램을 작성 후 같이 링크해야 한다.

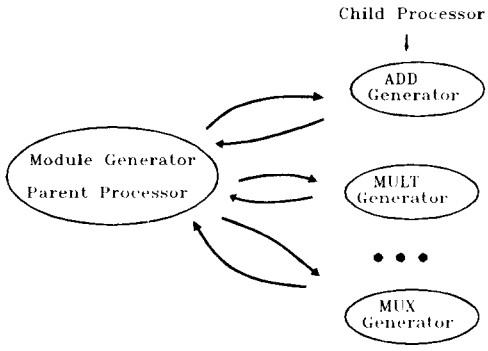


그림 3. 모듈 발생기
Fig. 3. Module generator.

각 셀의 생성은 각각의 독립된 프로그램으로써 관리되고 셀을 생성하는 방법은 셀라이브러리에 있는 레이아웃들을 조합함으로써 이루어 진다. 셀의 생성을 위한 프로그램은 셀 생성에 필요한 CIF 형식의 화일을 읽어 그것을 호출하는 것으로 이루어져 있다. 예를 들면 8비트 시리얼 곱셈기를 생성한다고 하면 2비트에 해당하는 기본 레이아웃을 4번 호출하여 조합하면 된다. 곱셈기에 해당하는 기본 알고리듬은 아래와 같다.

```
main(int argc, char **argv)
{
 각 매개 변수 검증 및 초기화;
 for(i=0 ; i < 셀의 비트 수; i++
 {
 draw(x_position, y_position, "기본셀의 이름");
 update(x_position, y_position);
 }
cout << 생성한 셀의 자료;
}
```

4. 위치 설정기

자동 레이아웃에는 필수적인 기능으로 셀의 위치를 결정하는 일과 각 셀 간의 배선을 하는 일이 있다. 위치 설정기는 설계하려고하는 집적 회로 내에 각 셀들의 위치를 결정한다. 좋은 위치 설정은 자동 레이아웃의 아주 중요한 요소이며 배선을 보다 쉽게한다. 위치 설정은 직접적으로 회로의 연결을 위한 도선의 최소 길이를 결정하며, 도선에 의한 시간지연이 신호 응답의 중요한 요소가 될 수 있으므로 위치 설정은 종종 설계된 회로의 성능을 결정하기도 한다.

본 논문에서는 초기 위치를 설정하는 단계에서는 채널의 수를 결정하고 Min-Cut 알고리듬을 변형한 방법을 이용하여 각 채널의 상, 하부에 셀들의 초기 위치를 결정하였다. 초기 위치가 결정된 이후 각 채널에서의 셀들의 위치설정은 시뮬레이티드 어닐링 방법(Simulated Anealing Method)을 적용한 알고리듬을 이용하였다.

본 논문에서 개발한 위치설정기는 기본적으로 아래와 같은 알고리듬으로 동작을 한다.

```
placement()
{
 Initialize();
 Read_Input();
 Place_Pads();
 number_of_channels = Determine_Number_of_Channels();
 Channels = Make_Channels_By_Min_Cut();
 for(i = Channels→first(); i ; i = i → next)
   Place_Cells_By_Simulated_Annealing();
 Ouput_Placement();
}
```

위치 선정기의 기본 알고리듬

우선 자료들을 초기화(Initialize())하고 중간 형식 화일로부터 회로를 레이아웃하는 데 이용되는 각 셀의 명세를 읽어들여 필요한 자료 구조로 변환한다. (Read_Input()) 본 논문에서는 입력 자료를 위해서는 리스트 구조(Linked List)의 자료구조를 이용하였으며 위치선정기의 결과를 위해서는 채널의 상, 하부를 각각 하나의 리스트로 관리하는 리스트의 리스트 구조(List of List)를 이용하였다.

본 논문의 위치 설정기는 자료 구조로 변환된 셀들의 정보를 가지고 채널의 수를 결정한다. 이때에는 총 네트의 갯수와 각 셀의 크기를 입력으로 하여 채널의 폭과 상, 하부의 높이를 계산한다. 이 계산 결과로 설계하려고 하는 집적회로의 가로, 세로의 비율을 원하는 비율에 맞도록 셀들을 배치한다. 가로, 세로의 비율은 임의로 설계자가 설정할 수 있도록 프로그램화 하였다. 채널의 수가 결정된 이후에는 Min-Cut 알고리듬에 의하여 각 채널의 구성 셀들을 결정한다. 본 논문에서는 네트의 연결에 관한 상태값 뿐 아니라 각 채널의 길이의 차에 관한 상태값을 갖는 상태평가함수를 이용하여 채널의 모양이 규칙적인 배열을 갖도록 하였다.

$$Cot(S) = Wn\sum n(i, j) + W1\sum \{Length(i) - Length(j)\}$$

S : 상태, $i \neq j \in Row$
Length(j) : j Row의 길이
N(i, j) : i Row와 j Row 간의 연결 네트의 수
Wn, W1 : 각각 네트, 길이의 차에 관한 가중치

결정함수=셀의 집합 간의 연결 네트수+각 채널의 길이의 차의 합

이렇게 결정된 각 채널에 대하여 시뮬레이티드 어닐링 방법을 적용하여 각 셀의 위치를 결정하게 된다.

시뮬레이티드 어닐링에 쓰이는 상태값은 각 네트 길이의 합, 가능한 채널의 최소 폭과 채널의 상단과 하단 길이의 차이, 채널의 상단과 하단의 높이 등이다. 이 각각의 값들은 최대값 1로하고 최소값은 0으로 정규화 시켜서 이용하였다. 또한 가중치를 설계자가 임의로 설정할 수 있도록 하였다.

현재의 상태 값은 아래와 같은 식으로 표현 할 수 있다.

$$Cost = \sum^{N} Cost(i)*Weight(i)$$

단, i는 각 평가를 나태내는 인수, N은 평가함수의 수, Cost는 평가함수, Weight는 가중치이다.

마지막으로 위치가 결정된 셀들의 결과를 중간형식 화일 형태로 변환하여 출력하게 된다.

Ⅲ. 결과 및 고찰

그림 4는 임펄스 응답 특성을 가진 콘벌루션을 수행하는 기본 구조이다. N 차수의 콘볼루션을 수행하기 위하여, N 의 곱셈과 N 번의 가산을 행해야 한다. 본 논문에서는 이를 비트 시리얼 방식으로 시스템을 설계하여 시뮬레이션 하였다. 그림 5는 이것의 스케메틱과 시스템 검증기를 통한 출력 신호들의 그림이다. 이 시스템의 입력 자료는 14비트이고 필터의 계수는 8비트로 정수화 한 후 MSB의 비트를 사인 비트로하여 2의 보수 형태로나타낸 것이다.

콘볼루션 코어는 8비트 계수의 곱셈을 위해 4개의 비트-시리얼 곱셈 블럭으로 구성된 하나의 곱셈기와, 하나의 비트 시리얼 가산기 및 3개의 MUX와 두개의 단어 지연부를 사용하였다.

모듈 발생기는 시스템 기술 언어 번역기가 출력한

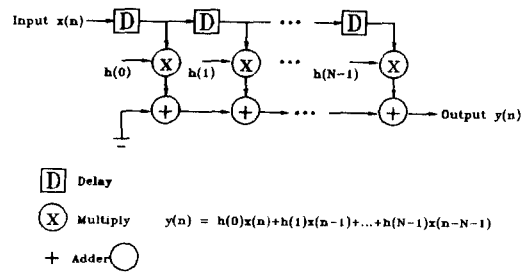
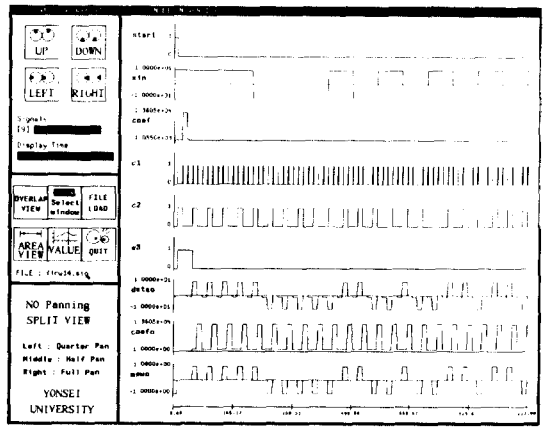


그림 4. 콘볼루션 연산구조
Fig. 4. Convolution system.



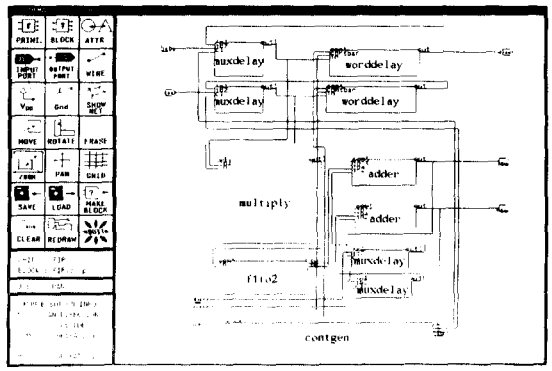


그림 5. 콘볼루션의 스케메틱과 시스템 검증기의 결과
Fig. 5. Schematic and simulation result.

중간 형식 화일을 읽어 필요한 각 셀들의 레이아웃을 만들고 물리적 특성들을 출력한다. 그림 6은 4비트 곱셈기의 레이아웃이다. 1.2μm CMOS 설계규칙을 이용하여 전체 면적이 232μm×373μm이며 6클럭 이후에

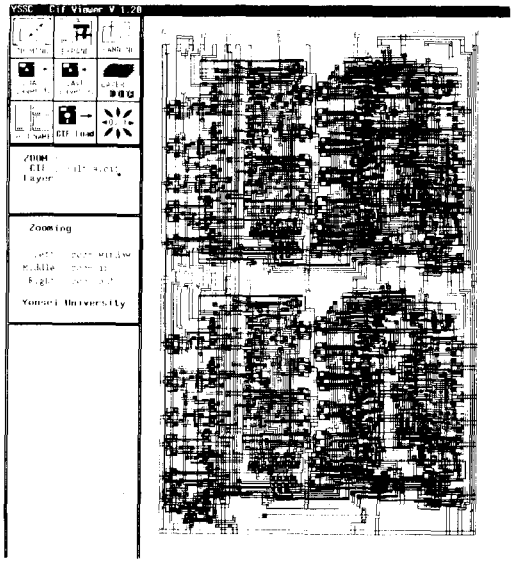


그림 6. 4 비트 비트시리얼 곱셈기
Fig. 6. 4 bit multiplier.(bit serial)

출력이 나오게 된다.

모듈 발생기의 출력으로부터 각 셀의 물리적 크기와 네트에 관한 정보를 얻어 위치 설정기는 집적회로 내에 각 셀들의 위치를 설정한다. 그림 7은 앞의 콘벌루션 코어를 이용하여 만든 2차 IIR 필터에서의 각 셀들의 위치를 위치 설정기를 통하여 설정한 예이다. 이 예는 상태 평가 함수 중 집적 회로의 크기에 중점을 두어 생성한 것이다.

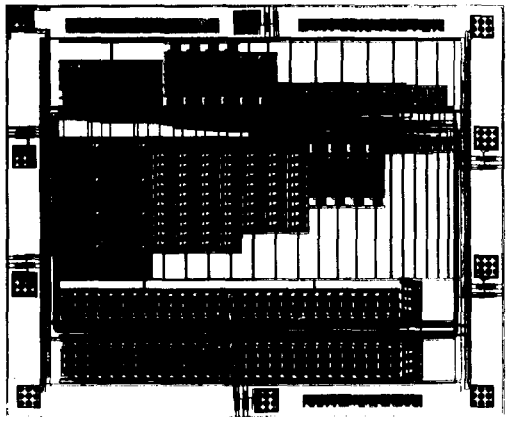


그림 7. IIR 디지틀 필터의 레이아웃
Fig. 7. IIR digital filter.

본 예의 IIR 필터는 14비트의 시스템 단어 길이를 가지며 데이타 샘플링 률은 44.1㎑이다. 시스템 클럭은 45.158㎒이다. 면적은 4.85㎟이고 약 13,000여개의 트랜지스터가 이용되었다.

Ⅳ. 결 론

최근의 공정기술의 발달과 반도체 시장의 급변으로 집적회로의 설계시간이 제품의 중요한 요소가 되었다. 실리콘 컴파일러의 개발은 좋은 해결책을 제시하였다. 본 논문에서는 실리콘 컴파일 기법에 알맞는 디지틀 신호 처리에 이용되는 실리콘 컴파일러의 사용자 툴들을 개발하였다. 시스템 설계자가 상위 레벨에서 설계한 시스템을 기술하는 방법을 위하여 본 논문에서는 시스템 기술 언어를 정의하였고 이를 셀 라이브러리에서 제공하는 셀의 인접 관계로 번역하는 시스템 기술 언어 번역기, 시스템의 성능 및 기능을 검증하기 위한 시스템 검증기, 필요한 각 셀의 실제 레이아웃을 하는 모듈 발생기, 각 셀들의 위치를 설정하는 위치 설정기를 개발하였다. 이들 각 툴들은 시스템 설계자에 의하여 각각 독립적으로 수행될 수 있다.

집적회로 설계시에 필요한 각 셀을 발생하는 모듈 발생기는 각 셀들을 발생하는 프로그램을 관리하는 프로그램으로 개발하였다. 각 셀을 발생하는 각각의 프로그램은 모듈 발생기에 의하여 호출되어 병렬 수행되는 자프로세서이다. 즉, 모듈 발생기는 하나의 모 프로세서로서 각 셀을 발생하는 자프로세서를 발생시키는 역할을 한다.

각 셀의 위치를 설정하는 위치 설정기는 Min-Cut 알고리듬과 시뮬레이티드 어닐링 방법을 이용하여 구현하였다. 셀의 초기 위치를 결정하기 위하여는 Min-Cut 알고리듬을 이용하였으며 각 채널의 셀을 위치시키는 데는 시뮬레이티드 어닐링 방법을 이용하였다. 본 논문에서는 시뮬레이티드 어닐링의 흐름을 제어하는 각 값들을 많은 예제의 수행 없이 이론적인 값으로 설정하였다. 차후에 이들 값들은 많은 예제를 수행하여 그결과를 분석하여 결정할 필요가 있다.

시스템 검증기는 이벤트 드리븐 방식을 채택하여 객체지향언어인 C++ 언어를 이용하여 완성하였다. 시스템 검증기는 시스템 기술 언어 번역기가 생성한 중간 형식 화일을 C++ 언어의 프로그램으로 변환하여 수행하는 방법을 취하였다. 본 논문의 시스템 검증기는 다중처리를 위한 AT&T C++의 라이브러리를 이용하여 병렬 처리 방식을 이용하여 개발하였다.

參 考 文 獻

[1] 이문기, 이광엽, 장호랑, 김종현, 송준규, 통신용 신호처리 칩 컴파일러 개발, 전기통신학술연구과제 보고서, 1991.

[2] 이문기, 장호랑, 김종현, HDTV용 ASIC 설계 자동합성기 개발에 관한 연구, 상공부 보고서, 1991.

[3] 김종현, 장호랑, 이승호, 이문기, "디지탈 신호처리용 실리콘 컴파일러를 위한 레이아웃 합성기 개발", 대한전자공학회 논문지, 제29권 제6호, pp. 54~61, 1992. 6.

[4] 이문기, 유향기기와 디지틀 신호처리용 ASIC 기술, 제 1 차 오디오 첨단기술 세미나, 한국유향기기연구조합, 1991. 3.

[5] Bryan T. Preas, Michael J. Lorensetti, *Physical Design Automation of VLSI Systems*, The Benjamin/Cummings Publishing Company, Inc 1988.

[6] Sun microsystems, *AT&T C++ Translator Library Manual*, AT&T 1989.

[7] Steven M. Rubin, *Computer Aids for VLSI Design*, Addison-Wesley Publishing Company, Inc 1987.

[8] Carl Sechen, *VLSI Placement and Gloabal Routing Using Silmulated Annealing*, Kluwer Academic Publishers, 1988.

[9] M. Morris Mano, *Computer System Architecture*, Pretice Hall, Inc.

[10] Danniel D. Gajski, *Silicon Compilation*, Addison-Wesley Publishing Company, Inc 1988.

[11] Aho, Alfred V., *Compilers, Principles, Techniques, and Tools*, Addison Wesley 1988.

[12] Axel T. Schreiner, H. George Friedman, Jr., *Instruction to Compiler Construction with UNIX*, Prentice-Hall, Inc.

[13] K. John Gough, *Syntax Analysis and Software Tools*, Addison Wesley 1988.

[14] Fathy F. Yassa, Jeffrey R. Jasica, Richard I. Hartley, Sharbel E. Noujaim, "A silicon compiler for digital signal processing : Methology, inplementation, and applications", *Proceedings of IEEE*, vol. 75, no. 9, Setempber 1987.

[15] Ayres, VLSI Silicon Compilation and the *Art of Automatic Macrochip Design*, Prentice Hall.

[16] P. Denyer, D. Renshow, *VLSI Signal Processing: A bit Serial Approach*, Addison Wesley 1985.

著 者 紹 介

李 文 基(正會員) 第28卷 B編 第7號 參照
현재 연세대학교 전자공학과 교수

●

張 虎 郎(正會員)
1966年 1月 15日生. 1990年 연세대학교 전자공학과 졸업. 1992年 연세대학교 전자공학과 대학원 석사학위 취득. 현재 삼성 종합기술원 연구소 연구원. 주관심분야는 CAD Tool 개발, High Level Synthesis, VHDL 등임.

●

金 鍾 鉉(正會員)
1967年 8月 26日生. 1990年 연세대학교 전자공학과 졸업. 1992年 연세대학교 전자공학과 대학원 석사학위취득. 현재 금성일렉트론 중앙연구소 연구원. 주관심분야는 CAD Tool 개발 및 VLSI 설계 등임.

李 承 浩(正會員)
1966年 1月 3日生. 1988年 연세대학교 전자공학과 졸업. 1990年 연세대학교 전자공학과 대학원 졸업. 현재 연세대학교 전자공학과 박사과정 재학중. 주관심분야는 VLSI 설계 및 Computer Architecture 등임.

●

李 光 燁(正會員)
1961年 9月 5日生. 1985年 서강대학교 전자공학과 졸업. 1987年 연세대학교 전자공학과 대학원 석사학위 취득. 현재 연세대학교 전자공학과 박사과정 재학중. 주관심분야는 VSLI 설계 및 Computer Architecture, CAD 시스템 등임.