[4-6]

통신용 신호처리 칩 컴파일러 개발

김 종 현 , 장 호 랑 , 송 준 규 , 이 승 호 , 이 광 엽 , 이 문 기
연 세 대 학 교 전 자 공 학 과

A Development of the Chip Compiler for Communication Signal Processing

Jong Hyun Kim, Ho Rang Jang, Jun Geu Song, Seung Ho Lee, Kwang Yeob Lee, Moon Key Lee
Yonsei University , Dept. of Electronic Engineering

ABSTRACT

This paper concerns the development of a CAD tool supporting the design of a custom VLSI chip directly by a communication system designer in accordance with the needs of his system.

This tool is a communications chip compiler and is named YSSC. The objective is to produce the physical layout from a design described in a high level description language. The target architecture is a bit serial architecture with which real-time operation is possible in a broad frequency range from modem to PCM communication.

This paper describes the design of the YSSC's data structure and the target architecture.

1. 서 론

통신 시스템의 설계는 기존의 아날로그 방식에서 디지탈 방식으로의 변화가 급속도로 이루어지고 있으며 하나의 시스템을 구현하기 위하여 수 많은 반도체 칩을 이용하고 있다. 또한 통신 시스템에서는 다량의 데이터에 대한 고속의 처리 능력과 시스템의 소형화에 대한 요구를 고려할 때 통신 시스템을 구성하기 위한 VLSI 설계기술의 응용은 필수적이라고 할 수 있다.[1]

한편 일반적인 범용의 반도체 소자를 이용하여 시스템을 구현할 경우에 비하여 특정 용도의 반도체 칩을 사용하여 시스템을 구현할 경우 시스템의 소형화, 고성능화의 이득을 얻을 수 있으며 신뢰성을 높일 수 있다.

그러나 특정 용도의 칩을 설계, 제작하기 위해서는 수 많은 노력과 시간을 필요로 하며, 그 과정에서 오류를 범할 수 있으므로 시험 제작된 칩에 대해 여러번의 검증 과정을 거쳐야 한다. 또한 근래와 같이 칩 위에 보다 많은 회로 소자를 집적할 수 있는 기술이 발전되어 회로의 복잡도가 기하급수적으로 증가하는 현실에서는 통신용 반도체 칩을 자동으로 생성해낼 수 있는 도구의 필요성이 그만큼 크다고 할 수 있다.

그러한 필요성에 의하여 컴퓨터를 이용한 자동화에 대한 연구가 진행되고 있으며, 자동화의 범위도 종래의 단순히 설계자의 각 단계의 작업을 대치하는 것으로 부터 설계자의 개념적인 요구를 받아 레이아웃의 생성까지를 자동화하는 것으로 변화되고 있다.[1]

본 논문에서는 통신용으로 사용할 수 있는 반도체 칩을 생성하기 위한 칩 컴파일러 (YSSC) 개발의 일환으로 컴파일러에서 필요로하는 자료구조의 일부를 수립하는 연구를 수행하였다. 그 결과 언어의 컴파일 과정을 거친 후 Layout generator 의 입력으로 사용되기 위한 자료구조를 수립하였다.

2. 실리콘 컴파일러의 구성

YSSC 의 구성은 설계자가 시스템에 대해서 High level language 입력으로 표현하였을 때 Physical layout 을 자동 생성하도록 Compile 및 simulation, floorplan 등의 독립된 기능과 자료 구조로 되어 있다.그림 1 은 YSSC 의 구성도를 보여준다. 입력 언어

[illegible] 언어이며 기능 블록 단위로 표현한다. Language Compiler 는 사용자가 제공한 Description File 을 Compile 하여 중간 단계의 정보 형태로 변환한다. 변환된 정보는 Simulation 이나 Layout Generation 을 위해 사용된다. Layout Generator 는 중간 형태의 정보와 Cell library 를 이용하여 주어진 Design Rule 에 위배되지 않도록 Layout 을 자동 합성하며 Simulator 는 설계된 시스템의 동작 검증을 위하여 사용된다.

3. 중간 자료 구조의 설계

3.1 Targer Architecture

디지탈 신호 처리를 위하여 고려할 수 있는 하드웨어 구조로는 Bit serial, Multi processor, Systolic array 등이 있으며 YSSC 에서는 Bit serial 구조로 칩을 구성한다.

비트 직렬 구조의 경우 칩 면적의 상당 부분을 차지하는 배선 면적을 크게 감소시킬 수 있으며, 각 연산자를 pipeline 으로 연결하여 처리 속도를 향상시키기에 적합하다. 이는 통신 분야와 같이 연속적인 신호의 처리가 요구되는 경우에 적합하며 약 10KHz - 100KHz 정도의 처리율을 기대할 수 있다. 만약 보다 높은 처리 속도가 필요한 경우에는 다른 구조를 이용하여 칩을 구현해야 하지만 Modem, 계측 장비, Digital Audio 등과 같이 수요가 넓은 분야에서는 비트 직렬 구조의 처리 속도를 통한 구현으로 필요한 처리율을 만족시킬 수 있다.[3]

3.2 회로의 입력 형태

YSSC 에서의 회로 입력은 언어의 형태로 주어지며 회로의 기술은 이미 정의되어 제공되는 기능 블럭들의 구조적 연결을 선언하는 형태로 이루어진다. 언어 컴파일 단계에서는 Text file 에 대하여 Lexical analysis, Syntatic Analysis, Code generation 을 거쳐 각 기능 블럭에 대한 정보를 포함하는 중간 단계의 file 을 생성한다.[2][4]

시스템을 구성적 측면에서 기술할 경우 (Structural Description) 개념 설계 과정의 복잡성을 증가시키는 단점을 갖는 반면, 컴파일 과정의 간략화와 시스템 기술에 대한 에매모호함의 배제를 통하여 불필요한 낭비를 방지한다는 장점을 가진다. 한편 회로의 기술은 계층적으로 보다 상위의 블럭이 하위의 블럭을 호출하는 형태로 이루어지며 가장 하위의 블럭은 컴파일러가 제공하는 Primitive 들로 이루어진다. 따라서 호출에 대한 반복적인 탐색을 통하여 평면화된 일차원적인 Primitive 들의 배열을 얻게 되며 이의 가상적인 예를 그림 2 에 나타내었다. 한편 평면화된 각각의 Primitive 들은 신호 및 제어 입출력 Node 의 이름을 정규화하여 가진다. 평면화된 Primitive 의 연결은 각 Primitive 를 Vertex 로 하고, 각 node 를 Arc 로 하는 graph 로 간주할 수 있다.

3.3 자료 구조

언어의 컴파일을 거쳐 생성되는 자료는 다음과 같은 요구를 만족시켜야 한다.

- 비 계층적인 기능 블럭의 선언
- 정규화된 Node 번호의 사용
- 각 블럭에서 추출된 파라미터의 수용
- 임의 갯수의 입출력 신호의 제공
- 배치 배선 정보의 수용 능력

사용자의 기술 파일은 다수의 기정의 및 사용자 정의 기능 블럭이 계층적으로 호출되는 형태를 취하므로 각각의 블럭이 동등한 단계로 확장되어 Layout 을 반복적인 작업을 통하여 생성하도록 한다. 또한 변화되는 Technology 와 기술 파일의 독립성을 보장하기 위해서는 이에 대한 정보를 포함하지 않도록 한다. 배치와 배선에 대한 정보는 설계자의 요구가 있을 경우에만 포함되도록 하여 칩 면적 효율을 높이도록 한다.

결과적으로 컴파일 이후 Primitive 들에 대하여 생성되는 자료는 다음과 같은 간략한 정보만을 가지도록 제한된다. (그림 3 참고)

- Primitive Name (Type)
- Parameters
- Input Signals
- Output Signals
- Input Controls
- Output Controls

PLA 혹은 ROM Module 을 포함하는 경우에는 위와 같은 기본 Primitive 에서의 정보외에 추가적인 정보를 필요로한다.

- User Defined Delay
- Boolean Expression or Truth Table
  (혹은 ROM Code)

  Input/Output Type
  (Buffered Serial Or Parallel)

위와 같은 정보를 컴파일러 내부에서 보관, 처리하기위해 그림 3 과 같은 자료 구조를 고려할 수 있다. 그림 3 에서 각 header 는 linked list 구

조로 각 입출력 line 을 관리하기위해 사용되는 pointer 를 나타낸다. array 를 사용할 경우 자료 관리는 상대적으로 간단하게 되지만 임의의 입력과 출력을 가질 수 있는 PLA 혹은 ROM 과 같은 블럭을 선언하기가 어렵게 된다.

그림 3 이 각 Primitive 의 관점에서의 node 의 표현인데 반하여 그림 4 는 각 node 의 관점에서의 연결 관계를 나타내고 있다. 그림 4 의 Extended Node Name 은 node 의 확장된 이름이며 각 node 가 연결된 Primitive 들의 I/O port 를 저장한다. 따라서 Layout generation 과정에서 각 블럭을 생성하기 위해서는 그림 3 과 유사한 형태의 자료 구조가 유리하며 routing 시에는 그림 4 와 같은 자료구조가 유리하다.

PLA 의 expression tree 의 구조는 그림 5 와 같은 형태로 이루어지며 각 tree node 에서 inorder 형식의 탐색을 이용하여 수식의 실행이나 표현을 이룰 수 있다.

언어 컴파일러부와 Layout generator 부의 Interface 를 위해서는 그림 3 수준의 정보를 이용하며 PLA 나 ROM 이 포함된 경우, 혹은 배치 정보를 포함할 경우등에는 필요한 형태의 정보를 추가한다. 여기서 정보의 저장 및 공유를 위하여 Text file 의 형태를 이용한다. 이는 차후의 확장이나 다른 Computer 와의 공유를 쉽게하기 위해서이며, debugging 의 편의를 제공한다.

3.4 File 의 구성

언어 컴파일 후 평면화 된 각 Primitive 들의 나열로 이루어진 정보는 중간 단계의 Text file 로 저장되어 Layout generation 과정으로 넘겨진다.

중간 file 은 3.3 절에서 언급된 바와 같이 각 primitive 의 외형적인 연결과 parameter 들로 이루어진다. 하나의 primitive 를 표현하기 위한 형태는 다음과 같다.

< primitive 종류 > < parameter lists > < input nodes > < output nodes > <input controls > < output controls >

< PLA 선언 > < input 수 > < output 수 > < input control 수 > < output control 수 > < input nodes > < output nodes > < input controls > < output controls > < delay > < I/O type > < 선언의 종류 > { < minterms of one output > $ }

위에서 하나의 output 을 표현하기 위하여 그림 5 와는 별도로 sum of product 의 형태로 이루어진 다수의 field 를 포함한다. 이 경우 각 input 위치에 대하여 0 (compliment), 1 혹은 2 (don't care) 의 값을 이용하여 하나의 minterm을 표현하며 output 들은 '$' 기호로 서로 구분한다.

4. 결 론

본 논문에서는 통신용 칩 컴파일러인 YSSC 개발을 수행하였다.

특히 Target architecture 로 모뎀에서부터 PCM 통신에 이르는 주파수대역에서 충분히 동작이 가능한 비트 직렬 구조를 선택하였고, 이 구조를 효율적으로 지원하는 자료구조를 설계하였다.

본 자료 구조는 칩 컴파일러에서 언어 컴파일부와 Layout 생성부의 정보 교환을 위하여 사용될 수 있으며, Text file 로 기록되어 사용자가 해독 가능하다는 특징을 가진다.

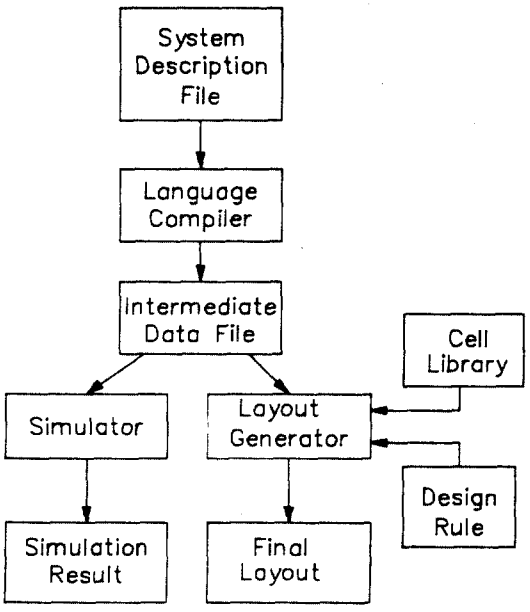
REFERENCE

1 D. D. Gajski, Silicon Compilation, Addison Wesley 1988

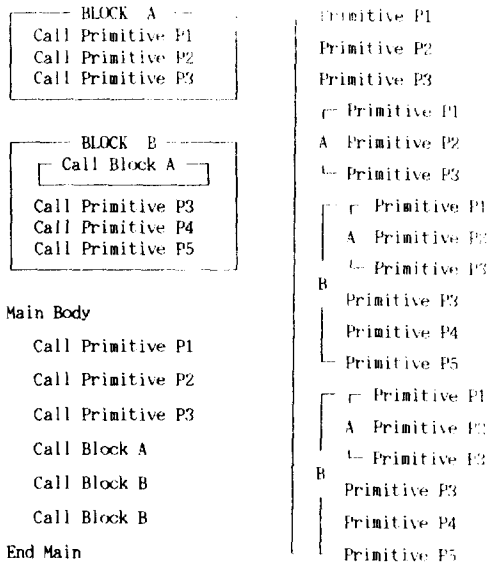
2 P. DENYER, D. RENSHOW, VLSI Signal processing : A Bit Serial Approach, Addison Wesley 1985

3 R. F. Lyon, A Bit Serial Architecture Methodology for Signal Processing, VLSI 81, Academic Press 1981

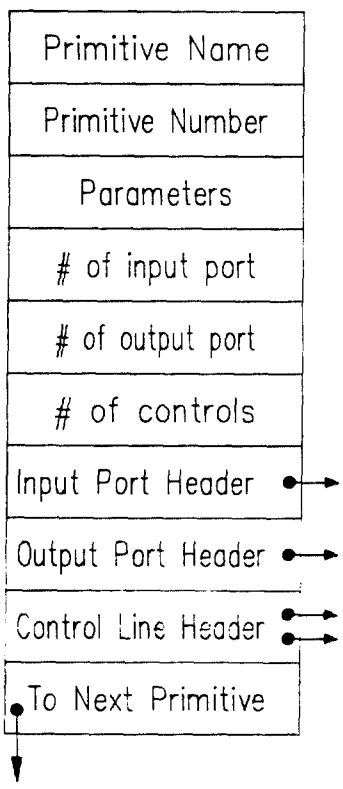
4 N. Bergman, A Case Study of The F.I.R.S.T. Silicon Compiler, 3rd CALTECH Conference on VLSI, 1983



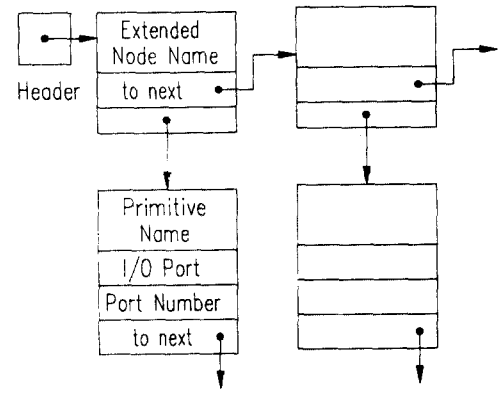
(Fig 1) YSSC 의 구성



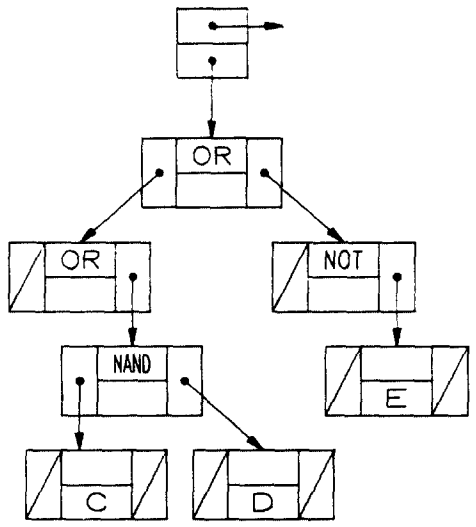
(Fig 2) Primitive 확장 예



(Fig 3) Primitive 의 자료구조



(Fig 4) Node 의 자료구조



(Fig 5) Expression Tree