論 文

VHDL 환경 설계 및 구현

正會員 金 忠 錫* 正會員 表 昌 祐** 正會員 元 裕 憲*

Design and Implementation of VHDL Environment

Choung Seok Kim* Chang Woo Pyo** Yoo Hun Won* *Regular Members*

要 約

IEEE에서 표준화된 하드웨어 설계언어인 VHDL은 하드웨어 설계분야에서 그 사용이 점차 확산되고 있다. 본 연구에서 개발된 VHDL 환경은 VHDL 지원환경(Support Environment)과 VHDL 사용환경(Using Environment)으로 구성되었다. VHDL 지원환경은 분석기, 상위수준합성을 위한 CDFG (Ccontrol/Data Flow Graph) 생성기, CDFG를 입력으로하는 합성기, CDFG로부터 VHDL을 생성하는 VHDL생성기로 구축되었다. 이러한 지원 환경을 사용자가 보다 편리하게 사용할 수 있게 VHDL 사용환경을 개발하였다. VHDL사용환경은 VHDL 지원환경의 각 도구들을 그래피컬 사용자 인터페이스를 통하여 사용할 수 있게 하였고, 설계된 하드웨어의 구조로부터 VHDL 프로그램을 자동생성한다.

ABSTRACT

VHDL, which is the IEEE standard HDL, has gradually become popular in the area of hardware design. The VHDL Environment developed in this study consists of VHDL Support Environment and VHDL Using Environment. The VHDL Support Environment is composed of Analyzer, CDFG Generator for synthesis, Synthesizer, and VHDL Generator converting CDFG to VHDL. The VHDL Using Environment provides users with more convenient access to the VHDL Support Environment. The VHDL Using Environment allows accessing the tools in the VHDL Support Environment through Graphical User Interface. VHDL program can be automaticaly generated from schematics in the VHDL Using Environment.

I. 서 론

전자 시스템 하드웨어 설계란 기능을 서술하는 행위 기술로부터 논리 수준의 기술, 검증을 위한 시뮬레이션, 최종적으로는 레이아웃으로 변환시켜주는 일련의 과정을 의미한다. 반도체 칩의 고밀도화에 따른 복잡도(complexity)의 폭발적 증가에 효과적으로 대처하려면 이러한 일련의 변환 과정이 자동화되어야 한다. 이에 따라 많은 설계 자동화 도구들이 출현되었고, 이를 뒷받침하는 여러 종류의 하드웨어 기술

*釜山女子大學校 電子計算學科
**弘益大學校 컴퓨터工學科
論文番號 : 92-124(接受1992. 7. 18)

언어(HDL : Hardware Description Language)가 개발되었다[1,3,6]. HDL이 매우 다양하게 설계되어 구현되어 왔기 때문에 여러 설계 자동화 도구들끼리의 접속문제와 설계자 사이의 통신 문제가 심각하게 제기되었다. 이러한 문제를 제거하기 위해 설계하고자 하는 시스템의 동작특성과 상호간의 연결관계 및 설계의 문서화를 효과적으로 표현할 수 있는 표준 HDL의 개발이 미 국방부의 주도로 시작되었다. 표준 HDL을 제공함으로써 하드웨어 설계사양과 각 도구들사이의 일관성과 호환성을 이룰 수 있다. 표준 HDL인 VHDL(Very high speed integrated circuit HDL)의 개발은 1983년 8월에 시작하여 86년 10월 VHDL version 7.2 Release 2에 이르러 처음으로 하드웨어 시뮬레이션이 가능하게 되었다. IEEE에서는 표준 HDL로 VHDL을 선정한후 VASG(VHDL Analysis and Standardization Group) 주관으로 일부 특성을 보강하여 1987년 VHDL1076을 발표하였다[1]. IEEE가 표준 하드웨어 설계언어로 VHDL을 택한 이유는 다음과 같다.

- 이 언어로 전자 시스템의 기능을 완벽하게 기술할 수 있다.
- 하드웨어 시뮬레이션을 할 수 있도록 타이밍을 기술할 수 있다.
- 행위 기술(behavioral description)로부터 하드웨어 부품 연결 형태인 구조적기술(structural description)이 가능하여 계층적 설계가 가능하다.
- 기술된 하드웨어가 동작하는 환경을 거의 완벽하게 정의할 수 있다.
- 언어가 어느 특정한 반도체 기술에 종속되어 있지 않아 새로운 방법과 기술이 개발되더라도 이에 적응할 수 있다.

최근 외국 조사에 의하면 하드웨어 설계자의 85%가 VHDL이 표준 하드웨어 설계언어로서 사용될 것이라고 응답하였으나 응답한 설계자 중 현재 VHDL을 사용하고 있는 설계자는 16%에 불과하다[14]. 즉 앞서 기술된 많은 장점에도 불구하고 설계자 사이에서는 아직도 VHDL이 많이 이용되지 않고 있다. 그 이유로는

첫째 VHDL지원 환경의 조성이 아직 완벽하게 이루어지지 않고 있다.

둘째 이제까지 CAD 도구들만 사용하던 설계자에게는 프로그래밍 언어 형태인 VHDL이 생소하고 복잡하다는 것이다[2,3,14].

본 연구는 위와 같은 문제점을 해결하기 위한 연구로서 전자통신연구소와 협력하여 4년간 VHDL 환경을 구축하기 위한 도구들을 개발하였다[25,26,27,28,29]. VHDL 환경에 대한 국외에서의 연구를 보면 매우 활발하게 도구들이 개발되고 있으며 초기의 시뮬레이터 중심 개발에서 최근에는 합성기 개발에도 많은 진전을 보이고 있다[14]. 상용화되어 판매되고 있는 도구들을 보면 Synopsys, MCC, IBM, Vantage, CLSI, MINC, Viewlogic, Siemans, Intergraph 등 많은 회사에서 시뮬레이터, 합성기, VHDL자동생성 도구들이 판매되고 있다[14]. 이러한 VHDL 환경 도구들이 국내에 고가로 반입되고 있는 실정이며 반입된 도구들은 사용에만 치중되고 있다. 즉 이러한 VHDL환경의 도구들을 국내에서도 개발하여 VHDL 지원환경을 구성할 필요성에 의해 본 연구는 시작되었다.

본 연구에서 개발된 VHDL 환경[그림1]은 VHDL 사용환경과 지원환경으로 구성되었다.

VHDL 사용환경은 본 연구에서 개발된 VHDL 지원환경과 연계하여 개발되었으며 그래픽사용자 인터페이스, 라이브러리 시스템, 변환기로 구성되었다. 그래픽사용자 인터페이스는 지원환경의 각 도구들에 대해 인터페이스를 제공하며, 기존의 CAD 설계 시스템의 형태로 하드웨어를 설계할 수 있는 기능을 가지고 있다. 라이브러리 시스템은 하드웨어 설계시 필요로 되는 콤포넌트들을 제공하며 사용자가 설계한 하드웨어를 새로운 라이브러리로 저장할 수 있다. 변환기는 그래픽 인터페이스를 통하여 생성된 하드웨어 구조로부터 VHDL 프로그램을 자동생성해내는 부분으로 기존의 CAD 설계 시스템에 익숙한 설계자가 복잡한 VHDL 구문구조를 모르고서도 VHDL 프로그램을 생성하여 지원환경의 각 도구들을 사용할 수 있다.

VHDL 지원환경의 구성은 VHDL 원시프로그램의 구문과 의미분석을 수행한후 AST(Abstract Syntax Tree) 형태의 중간코드를 생성하는 분석기, 합성기의 입력으로 사용되는 CDFG(Control/Data Flow Graph)를 생성하는 VHDL2CDFG 생성기, 합성기, CDFG를 그래픽 인터페이스를 통하여 편집할 수 있는 편집기, CDFG로부터 VHDL을 생성하는 CDFG2VHDL생성기로 구성되었다. 본 지원환경에서 시뮬레이터는 아직 개발되지 않았으며, 현재의 분석기 출력은 구문구조 형태의 중간코드를 생성하고 있으며

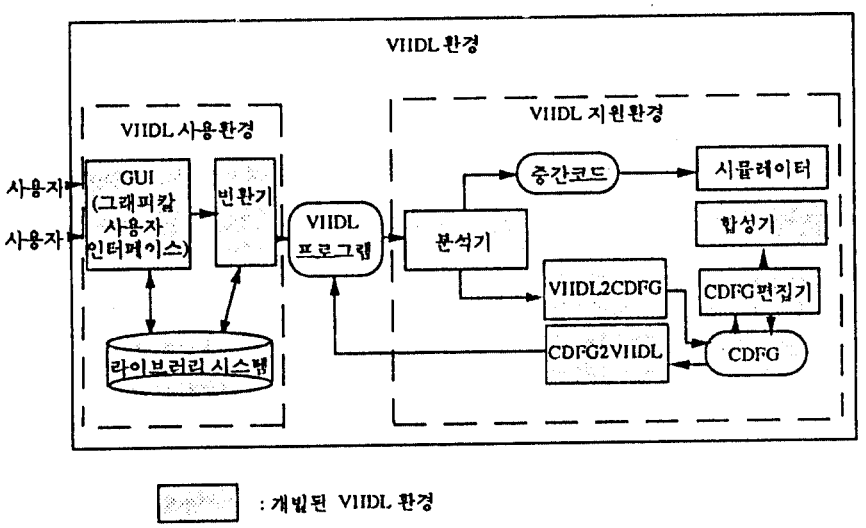


그림 1. 개발된 VHDL 환경

시뮬레이터 개발시 시뮬레이터의 입력에 적합한 중간코드를 분석기로부터 생성하여 본 시스템과 연결할 예정이다.

본 논문에서는 VHDL환경의 모든 도구들에 대한 개발내용은 지면상 수록하지 못하였고 지원환경의 각 도구들에 대해서는 도구들의 구조와 실행예를 들어 간략하게 기술하고, 개발내용과 결과에 대한 자세한 내용은 논문지나 보고서를 통하여 이미 발표 되었으므로 참고문헌으로 제공하였다[19-29].

II장에서는 현재 개발중인 VHDL 사용환경 기술하고 III장에서는 VHDL 지원환경에 관해 기술하고 IV장에서 결론을 맺는다.

Ⅱ. VHDL 사용환경

본 연구에서 개발된 사용환경은 세 부분으로 구성되어 있으며 본 장의 각 절로 기술한다. 사용자가 아이콘을 사용하여 지원환경의 도구들을 사용하고 하드웨어를 설계할 수 있게 하는 GUI(Graphical User Interface), GUI에서 하드웨어 설계시 사용할 수 있는 셀(이미 설계된 부품)들을 저장하고 있는 라이브러리 시스템, 설계된 하드웨어를 VHDL 프로그램으로 변환해 주는 부분으로 구성되었다. 사용환경의 초기형태가 UNIX의 Motif 환경에서 구현되었다.

1. 그래피컬 사용자 인터페이스(Graphical User Interface)

GUI(Graphical User Interface) 부분은 설계자가 직접 접근하여 사용하는 부분으로서 지원환경도구의 사용과 하드웨어 설계를 지원한다. 사용자는 라이브러리에 저장된 셀들을 이용하여 하드웨어를 구조적(structural)으로 설계할 수 있고 VHDL의 논리연산자(and,or,nand,nor,xor,not)를 이용하여 자료흐름적(dataflow)으로 설계할 수 있다[1,2,5]. 또한 텍스트 편집기를 이용하여 구조적으로 기술할 수 없는 하드웨어 행위(behavioral)기술이나 타이밍 등을 추가할 수 있다[1,2,5].

UNIX의 X를 이용하여 개발된 GUI는 하드웨어를 설계할 수 있는 도면(drawing) 윈도우와 라이브러리에서 제공되는 있는 셀의 심볼(셀의 도형적 표현)을 나타내는 심볼 윈도우 그리고 여러 기능의 메뉴들로 구성된다. 도면 윈도우는 설계하려는 하드웨어의 구조를 보여주며 하드웨어의 규모에 따라 윈도우의 크기를 사용중에 변환할 수 있다. 심볼 윈도우는 사용이 요구되는 라이브러리 셀의 심볼들을 보여주는데 사용된다. 설계 절차는 우선 마우스를 이용하여 필요한 셀의 심볼을 선택하여 도면 윈도우에 위치 시킨 다음 입출력 포트들을 선언하고 각 심볼을 연결하여 하드웨어를 설계한다. 또한 설계자는 설계된 하드웨어에서 콤포넌트로 사용된 심볼의 하드웨어구조를

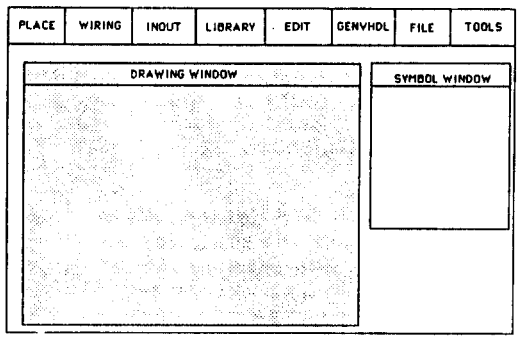


그림 2. 사용환경의 GUI 초기화면

뷰(view) 윈도우를 통하여 참조할 수 있다.

설계시 필요한 모든 기능은 9개의 주메뉴와 19개의 서브메뉴에서 선택하여 구동시킨다. 주메뉴는 화면 상단에 FILE, LIBRARY, INOUT, WIRING, PLACE, EDIT, GENVHDL, TOOLS 메뉴들로 구성되어 있으며 [그림2]와 같다. 각 메뉴는 서브메뉴를 가지고 있다.

메뉴와 서브메뉴들의 기능, 그리고 그 기능과 연관된 화일은 다음과 같다.

①FILE 메뉴

NEW : 새로운 하드웨어 설계 화일의 목록을 보여준다.

OPEN : 저장된 하드웨어 설계 화일의 목록을 보여준다.

SAVE : 설계된 하드웨어를 디스크에 화일로 저장한다.

PRINT : 도면 윈도우의 내용을 프린트 한다.

EXIT : 사용환경을 빠져나간다.

SAVE 서브메뉴를 선택하여 설계된 하드웨어를 저장하면 다음과 같은 두개의 화일이 자동생성된다.

〈filename〉.scs : 도면(drawing) 윈도우의 하드웨어 이미지(image)를 저장하는 화일.

〈filename〉.dst : 설계된 하드웨어의 구조와 사용된 셀, 논리연산자들의 연결관계를 저장하는 화일로서 중간코드 생성시 사용된다. 화일은 하나의 MNAME절과 여러개의 NAME절들로 구성된다. MNAME절에는 설계된 화일의 이름과 사용된 셀, 연산자들의 갯수, 그리고 입출력 포트들을 나타낸다. NAME절은 설계된 하드웨어에서 사용된 셀과 연산자들을 나타낸다. 셀 또는 연산자의 이름과 타입, 포트수, 그리고 연결관계를 포트의 순서에 따라 나타낸다.(예1참조)

②LIBRAY 메뉴

CELL LIST : 라이브러리가 제공하는 셀들의 목록을 보여주며 선택된 셀의 심볼을 심볼윈도우에 나타낸다.

OPERATOR : VHDL이 제공하는 논리연산자들의 목록을 보여주며 선택된 연산자에 대한 심볼을 심볼윈도우에 나타낸다.

CELL SCS : 이 메뉴를 사용하면 심볼윈도우에 있는 심볼의 하드웨어구조를 뷰(view) 윈도우를 생성시켜 나타낸다.

③INOUT 메뉴

INPUT : 입력포트명을 지정한다.

OUTPUT : 출력포트명을 지정한다.

INOUT : 입출력포트명을 지정한다.

④PLACE 메뉴

PLACE : 심볼윈도우의 심볼을 도면 윈도우내의 마우스에 의해 지정된 부분에 위치시킨다.

ROTATE : 심볼윈도우의 심볼을 오른쪽으로 90도씩 회전시킨다.

⑤WIRING 메뉴

WIRING : 마우스의 버튼을 이용하여 각 심볼들을 연결한다. 마우스의 3개의 버튼으로 윈도우내에서의 좌표를 이용하여 연결하며 연결하고자 하는 포인트의 ±5 범위내에서 마우스에 의해 지정되면 그 포인트에 자동연결된다.

⑥GENVHDL 메뉴

GENVHDL: 설계된 하드웨어에 해당되는 VHDL

화일(〈filename〉. vhdl)을 생성한다. EDIT 메뉴의 TEXT 서브메뉴에 의해 확인 및 수정할 수 있다.

⑦ EDIT 메뉴

TEXT : 〈filename〉. vhdl 화일의 목록이 나타나고 선택된 화일을 편집기 윈도우에 나타낸다. 프로그램을 확인하고 수정할 수 있으며 행위적인 기술이나 타이밍등을 추가할 수 있다.

SYMBOL : 설계된 하드웨어를 라이브러리에 저장할 경우에 사용된다. 저장될 셀의 도형적 표현인 심볼을 그리기 위해 비트-맵(bitmap) 편집기를 호출한다. 작성된 심볼은 〈filename〉. sym 이름의 화일로 저장된다.

⑧ TOOLS 메뉴

ANALYZER : 분석기를 실행시킨다. 입력화일이 요구된다.

VHDL2CDFG : CDFG를 생성한다.

CDFG2VHDL : VHDL을 생성한다.

SYN EDIT : 합성기의 편집기를 호출하여 합성기를 사용한다.

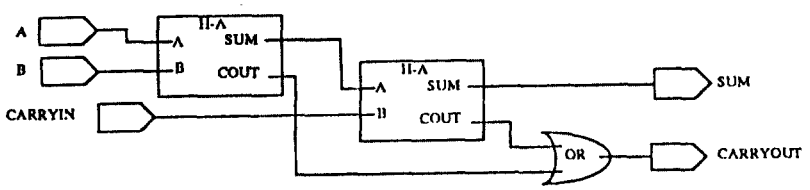
〈예1〉 GUI를 이용하여 설계된 전가산기의 예

전가산기를 구성하기 위해 GUI의 라이브러리 메뉴의 CELL 서브메뉴를 이용하여 2개의 반가산기와 1개의 OR 게이트를 선택하여 도면 윈도우에 위치시킨다. INOUT 메뉴를 이용하여 전가산기의 입출력을 정의하고 WIRING 메뉴로 각 셀들과 입출력 포트들을 연결한다. GUI를 이용하여 설계된 하드웨어는 SAVE 서브메뉴로 저장되는데 〈filename〉. scs 화일은 설계된 하드웨어의 구조를 나타내는 화일이며 〈filename〉. dst 화일은 설계된 하드웨어에 대한 텍스트 화일로서 설계된 하드웨어의 구조와 연결관계를 나타낸다. 전가산기에 대한 두 가지 형태의 화일은 다음과 같다.

FULLADDER. DST 화일

```
MNAME : FULLADDER  //설계된 하드웨어의 이름//
NUM OF CELL : 3  //설계된 하드웨어에서 사용한 라이브러리 셀의 수//
NUM OF OPERATOR : 0  //설계된 하드웨어에서 사용한 연산자의 수//
INPUT : A, B, CARRYIN  //설계된 하드웨어에서 사용한 입력 포트//
OUTPUT : SUM, CARRYOUT  //설계된 하드웨어에서 출력 포트//
INOUT :  //설계된 하드웨어에서 사용한 입출력포트//
NAME : HALFADDER. 1  //사용된 셀 또는 연산자의 이름//
TYPE : CELL  //타입(CELL OR OPERATOR)//
PORTS : 4  //포트 수//
PORT1 : EXT,A  //HALFADDER.1 셀의 연결관계에 대한 정보 형식 [포트번호 : 외부 또는 내부, 포트이름, 셀의 이름(INT일 경우)]//
PORT2 : EXT,B
PORT3 : INT,A,HALFADDER.2
PORT4 : INT,A,OR.1
NAME : HALFADDER.2
TYPE : CELL
PORTS : 4
PORT1 : INT,SUM,HALFADDER.1
```

FULLADDER. SCS화일



PORT2 : EXT,CARRYIN
PORT3 : EXT,SUM
PORT4 : INT,B,OR.1
NAME : OR.1
TYPE : CELL
PORTS : 3
PORT1 : INT,COUT,HALFADDER.2
PORT2 : INT,COUT,HALFADDER.1
PORT3 : EXT,CARRYOUT

2. 라이브러리 시스템

라이브러리는 GUI에서 하드웨어 설계시 필요로 되는 셀들을 저장하고 있으며 LIBRARY 메뉴의 CELL 서브메뉴로써 접근한다. 본 시스템에서 제공되고 있는 라이브러리는 우선 ETRI 표준셀[15]을 기본으로 구성되어 있다. 본 연구에서 구축된 지원환경에서는 라이브러리에 저장된 셀은 게이트 수준부터 ALU 같은 서브시스템 수준까지 제공되고 있다. 또한 사용자가 새로 설계한 하드웨어를 라이브러리의 한 셀로서 저장할 수 있다.

새로운 셀을 라이브러리에 저장할 경우 그 셀에 대한 두 개의 화일이 저장되어야 한다. 첫째 저장될 셀의 도형적 표현인 심볼의 모양과 그 심볼에서의 각 포트들에 대한 좌표에 관한 정보를 가진 화일(〈filename〉.sym)을 저장해야 한다. 심볼의 모양은 EDIT 메뉴의 SYMBOL 서브메뉴를 이용하여 작성하며 포트들의 좌표에 관한 정보는 적성된 심볼 모양으로부터 자동적으로 생성된다. 둘째 저장될 셀에 대한 VHDL 프로그램 화일(〈filename〉.vhdl)이 저장되어야 한다. 즉 사용자가 설계한 셀을 라이브러에 저장하기 위해서는 위와 같은 두개의 화일을 라이브러리에 저장해야 하며 두 화일의 화일이름(〈filename〉)은 같아야 한다. 현재 저장된 라이브러리의 셀에 대한 VHDL 프로그램은 모두 행위기술(behavioral desciption)이며 사용자는 라이브러리의 셀을 이용하여 원하는 하드웨어를 설계한다.

3. 변환기(Translator)

변환 부분은 GUI에서 설계된 하드웨어의 〈filename〉.dst 화일로부터 중간코드를 생성하는 부분과 중간코드로부터 VHDL 프로그램을 생성하는 부분으로 구성된다[그림3]. 중간코드를 사용함으로서 시스템을 전반부와 후반부로 나누어 설계할 수 있으며, 다른 CAD 시스템이 출력으로 이 중간코드를 사용할 경우 본 시스템과 접속가능하며, 중간코드를 다른 하드웨어기술언어 생성에 적용시킬 수 있다. 이 변환(translation) 부분은 GUI의 GENVHDL 메뉴에 의해 수행된다.

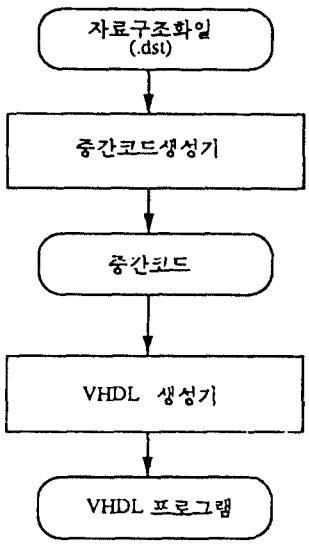


그림 3. VHDL생성을 위한 전체적인 구성도

1)중간코드와 중간코드생성

여기에서 사용한 중간코드는 하드웨어의 구조에 적합한 형태로 어떠한 구조의 하드웨어 형태도 다 나타낼 수 있으며 간략한 문법을 가지고 있다.

1)중간코드의 구문

본 시스템에서 사용하는 중간코드의 구문은 다음과 같다.

〈intermediate-code〉 : : = EXTERNAL〈external-clause〉
END_EXTERNAL
INTERNAL〈internal-clause〉
END_INTERNAL
〈external-clause〉 : : = '('EXTERNAL〈exinternal-clause〉')'
〈exinternal-clause〉 : : = 〈exmain-clause〉〈exsub-clause〉
〈exmain-clause〉 : : = '('MODmain ID 〈io-list〉')'
〈io-list〉 : : = '('ID {',' ID}' : '〈port-type〉{';'〈io-list〉}')'
〈exsub-clause〉 : : = '('MODsub ID')'{〈exsub-clause〉}|null
〈internal-clause〉 : : = '('INTERNAL ID','ID

〈node-list〉〈function-list〉')'
〈node-list〉::='('NODE〈innode-list〉')'
〈innode-list〉::='('INT 〈node-type〉 ID')'{〈innode-list〉}
〈function-list〉::='('FUNCTION〈infunction-list〉')'
〈infunction-list〉::='('〈cell-list〉';'〈mappint-list〉')'
{〈infunction-list〉}
〈cell-list〉::=INT 〈mapping-list-type〉 ID
{','〈mapping-list〉}
〈node-type〉::=〈cell-type〉|〈mapping-list-type〉
〈cell-type〉::=MOD|OPS
〈mapping-list-type〉::=〈port-type〉|SIG
〈port-type〉::=IN|OUT|INOUT

ID는 영문 알파벳으로 구성된 스트링 형태의 이름이고 INT는 정수를 의미한다.

(2)중간코드의 의미

중간코드는 자료흐름을 정의하며 설계한 하드웨어의 외부 인터페이스 부분을 나타내는 EXTERNAL 부분과 내부 기능을 나타내는 INTERNAL 부분으로 구성된다.

①External 부분

설계된 하드웨어의 입출력 포트와 사용된 셀들에 관한 정보를 나타낸다.

```
EXTERNAL
(EXTERNAL
  (MODmain 모듈이름
   (입출력 정보) //설계된 하드웨어에서 사용된 입
   )              출력 포트들에 관한 정보를 나타
                  낸다.//
  (MODsub 셀이름) //설계된 하드웨어에서 사용된
   ..............  셀들에 관한정보를 나타낸
                  다.//
  (MODsub 셀이름)
)
END_EXTERNAL
```

모 듈 이 름 : 사용자가 정의한 화일이름
입출력정보 : 설계된 하드웨어의 입출력을 나타낸다. GUI의 INOUT 메뉴에서 지정된 포트 이름과 타입을 나타낸다. 반드시 1개 이상의 포트가 있어야 한다.
셀 이 름 : 사용된 셀의 이름

②INTERNAL 부분

설계된 하드웨어의 내부 기능부분에 관한 정보들을 나타내게 된다.

```
INTERNAL
(INTERNAL 모듈이름
 (NODE          //설계된 하드웨어에서 사용된 셀,
  (노드정보...)  연산자, 임시변수, 입출력포트들
 )               을 고유의 노드번호로 선언한다.
                 형식 : (노드번호 노드타입 노드
                 명) //
 (FUNCTION      //설계된 하드웨어의 연결관계를 나
  (기능정보...)  타낸다. 형식 : (타겟노드 : 매핑리
 )               스트,...,매핑리스트) //
)
END_INTERNAL
```

모듈이름 : 하드웨어의 화일이름과 설계된 형태에 따라 정해진다.
예)STRUCTURAL.FULLADDER, MIXED.FULLADDER, DATAFLOW.FULLADER
노드정보 : 노드는 셀, 연산자, 입출력변수, 그리고 각 포트들을 연결하는 임시변수를 나타낸다. 노드는 고유의 노드번호, 노드타입, 그리고 이름으로 구성된다. 노드의 타입에는 MOD(셀), OPS(연산자), IN, OUT, INOUT(입출력변수), SIG(임시변수) 타입이 있다.
기능정보 : 설계된 하드웨어의 기능을 나타내며 타겟노드와 매핑리스트로 구성된다. 타겟노드는 사용된 셀이나 연산자를 나타내며 형태는 노드정보 형태와 같다. 단 노드타입이 MOD,OPS만을 가진다. 매핑리스트는 셀이나 연산자의 각 포트에 연결될 리스트로서 형태는 노드정보형태와 같다. 단 노드타입이 IN, OUT, INOUT, SIG만을 가진다. 매핑리스트의 순서에 따라 타겟노드의 포트들과 연결된다. 타겟노드의 포트순서는 라이브러리에 이미 결정되어 있다.

(3)중간코드 생성

EXTERNAL의 MODmain절을 생성하기 위해 .dst 화일로부터 설계된 하드웨어의 이름과 입출력포트에 관한 정보로부터 MODmain절을 생성하고 사용

된 셀들에 관한 정보로부터 MODsub절을 생성한다. 같은 셀이 여러번 사용된 경우에도 MODsub 절에는 한번만 나타낸다.

INTERNAL의 노드정보 부분을 생성하기위해 .dst화일로부터 입출력 포트들과 사용된 셀, 연산자들에 관한 정보를 이용하여 노드로 선언한다. 같은 셀이나 연산자를 반복하여 사용된 경우도 서로 다른 노드번호를 가진 노드로 선언되어야 한다. 사용된 셀들간의 연결은 임시변수를 사용한다. 이러한 임시변수는 .dst 화일의 NAME 절의 포트 타입(EXT 또는 INT)이 INT일 경우 생성되며 이름은 TEMP1, TEMP2,...,TEMPn의 형태로 생성한다. 하드웨어의 연결 구조를 나타내는 기능정보를 생성하기 위해 셀이나 연산자 타입으로 선언된 노드를 타겟노드로 하여 대응되는 매핑리스트를 구하여 생성한다.

〈예2〉전가산기에 대한 중간코드

EXTERNAL 부분에서는 설계된 하드웨어의 입출력과 사용된 셀들에 관한 정보를 나타낸다. 한 셀이 여러번 사용된 경우에도 MODsub 절에는 한번만 나타난다. INTERNAL 부분은 노드부분과 기능부분으로 구성된다. 노드부분에서는 설계된 하드웨어에서 사용된 셀, 연산자, 그리고 입출력 포트, 임시변수들이 고유의 노드번호를 가지고 선언된다. 기능부분은 하드웨어의 기능을 나타내는 부분으로서 사용된 셀이나 연산자가 타겟노드가 되어 대응되는 매핑되는 리스트와 함께 나타난다.

```
EXTERNAL
(EXTERNAL
  (MODmain FULLADDER
   (A,B,CARRYIN : in ; SUM,CARRYOUT : out)
  )
  (MODsub HALFADDER)
  (MODsub OR)
)
END_EXTERNAL
INTERNAL
(INTERNAL STRUCTURAL.FULLADDER
 (NODE
  (1 SIG TEMP1)
  (2 SIG TEMP2)
  (3 SIG TEMP3)
  (4 MOD HALFADDER)
  (5 MOD HALFADDER)
  (6 MOD OR)
  (7 IN A)
  (8 IN B)
  (9 IN CARRYIN)
  (10 OUT SUM)
  (11 OUT CARRYOUT)
 )
 (FUNCTION
  (4 MOD HALFADDER : 7 IN A, 8 IN B, 1 SIG
   TEMP1, 2 SIG TEMP2)
  (5 MOD HALFADDER : 1 SIG TEMP1, 9 IN
   CARRYIN, 10 SIG SUM, 3 SIG TEMP3)
  (6 MOD OR : 3 SIG TEMP3, 2 SIG TEMP2,
   11 OUT CARRYOUT)
 )
)
END_INTERNAL
```

2)VHDL 생성

앞서 기술된 중간코드부터 VHDL 프로그램을 생성하는 과정은 두 단계로 나누어 진행된다. 중간코드로부터 여러가지 테이블들이 생성되고 이러한 테이블로부터 VHDL 프로그램이 만들어 진다.[그림4]

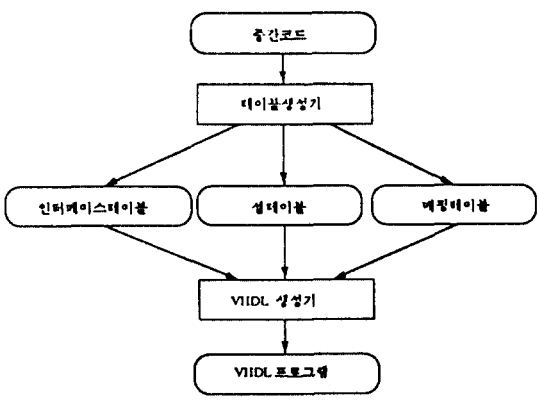


그림 4. 중간코드로부터 VHDL 생성과정

(1)테이블생성기

중간코드로부터 다음 3개의 테이블이 생성된다.

①인터페이스 테이블

설계된 하드웨어의 외부 인터페이스에 해당되는 정보를 저장한다. VHDL의 외부기술부분(ENTITY)에 해당되는 부분이다. 중간코드의 EXTERNAL 부분을 이용하여 테이블을 생성한다. 테이블은 다음과 같은 3개의 속성(attribute)을 가진 구조체로 정의된다.

CELL NAME : 설계된 하드웨어의 이름

PORT NAME LIST : 같은 타입을 가지는 포트들의 리스트

PORT TYPE : 포트들의 타입

②셀 테이블

설계된 하드웨어에서 사용된 라이브러리 셀들에 관한 정보를 저장한다.

EXTERNAL 부분의 MODsub 절로부터 셀이름을 구해 라이브러리로부터 그 셀의 포트에 관한 정보를 얻는다. 테이블의 구조는 인터페이스 테이블과 같고 3개의 속성중 CELL NAME은 설계된 하드웨어에서 사용된 셀이나 연산자의 이름이며 나머지 두 속성은 인터페이스 테이블과 같다.

③매핑 테이블

사용된 셀이나 연산자에 연결된 포트들을 저장하기 위해 사용된다. INTERNAL 부분의 NODE 부분과 FUNCTION 부분을 이용한다. 테이블은 다음과 같은 3개의 속성을 가진다.

LABEL NAME : 셀인 경우 자동적으로 표기(L1, L2,..,Ln)되고 연산자일 경우 표기되지 않는다.

CELL NAME : 셀 또는 연산자의 이름.

MAPPING LIST : 셀 또는 연산자의 각 포트에 매핑되는 리스트를 나타낸다.

〈예3〉전가산기의 중간코드로부터 생성된 3개의 테이블

INTERFACE TABLE

CELL NAME	PORT NAME LIST	PORT TYPE
FULLADDER	A B CARRYIN	IN
FULLADDER	SUM CARRYOUT	OUT

CELL TABLE

CELL NAME	PORT NAME LIST	PORT TYPE
HALFADDER	X Y	IN
HALFADDER	SUM CARRY	OUT
OR	A B	IN
OR	C	OUT

MAPPING TABLE

LABLE NAME	CELL NAME	MAPPING LIST
L0	HALFADDER	A B TEMP1 TEMP2
L1	HALFADDER	TEMP1 CARRYIN SUM TEMP3
L2	OR	TEMP3 TEMP2 CARRYOUT

(2)VHDL 생성기

앞서 생성된 3개의 테이블과 라이브러리를 이용하여 VHDL 프로그램을 생성한다.

[생성과정]

①셀 테이블에 저장된 셀들에 대한 VHDL 기술을 라이브러리로부터 구해 출력화일에 넣는다.

②인터페이스 테이블을 이용하여 VHDL 문법에 맞도록 ENTITY 선언 부분을 생성한다.

③VHDL의 ARCHITECTURE 부분을 생성한다.

ⓐ변수선언을 한다 : 매핑테이블의 매핑리스트로부터 TEMP로 시작되는 리스트에 대해 SIGNAL TYPE으로 선언한다.

ⓑCOMPONENT를 선언한다 : 셀 테이블의 셀등에 대해 라이브러리를 이용하여 COMPONENT를 실체화(instantiation)한다.

ⓒ연결관계를 기술한다 : 매핑테이블을 이용하여 레이블필드의 레이블명을 기록하고 해당되는 셀을 매핑리스트와 함께 기록한다. 만일 레이블이 존재하지 않는 경우 연산자로 취급하여 자료흐름형태(CARRYOUT 〈= TEMP3 OR TEMP2)의 문장으로 작성한다.

VHDL생성은 설계된 하드웨어에서 사용된 모든 셀에 대한 VHDL 기술을 출력에 포함시켰다. 한 출력화일에 다수개의 ENTITY가 나타날 수 있으며 이러한 형태의 입력을 분석기에서 받아들일 수 있도록 분석기를 설계하였다.

〈예 4〉VHDL 생성기에 의해 생성된 전가산기에 대한 VHDL 프로그램 예

```
entity ORGATE is
  port(A, B : in bit ; C : out bit) ;
end ORGATE ;

architecture BEHAVIORAL of ORGATE is
begin
  C <= A or B after 3 ns ;

end BEHAVIORAL ;

entity HALFADDER is
  port (X,Y : in bit ; SUM, CARRY : out bit ;
end HALFADDER

architecture BEHAVIORAL of HALF-ADDER is
begin
 process
  SUM <= X xor Y after 5 ns ;
  CARRY <= X and Y after 5 ns ;
  wait on X,Y ;
 end process ;
end BEHAVIORAL ;

entity FULLADDER is
 prot(A,B,CARRY-IN : in bit ;
    · AB,CARRY-OUT : out bit ;
end FULLADDER ;

architecture STRUCTURAL of FULLADDER is
 signal TEMP1, TEMP2, TEMP3 : bit ;
 component HALFADDER
 port(X,Y : in bit ;
    SUM,CARRY : out bit) ;
 end component ;
 component ORGATE
  port(IN1, IN2 : in bit ; OUT1 : out bit) ;
 end component ;
begin
 l0 : HALFADDER port map(A,B,TEMP1,
    TEMP2) ;
 l1 : HALFADDER port map(TEMP1,
    CARRY-IN, AB,TEMP3) ;
 l2 : ORGATE port map(TEMP2, TEMP3,
    CARRY-OUT) ;
end STRUCTURAL ;
```

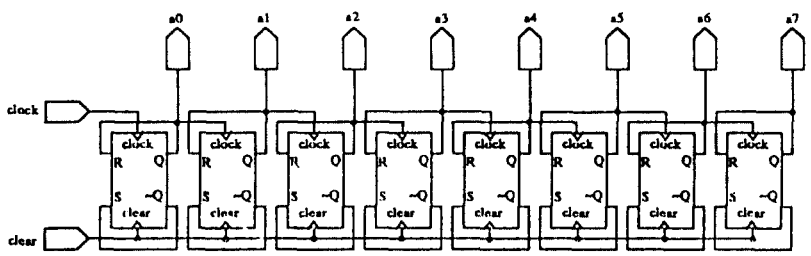
4. 설계 예

설계 예를 보이기 위해 8비트 리플 카운터를 설계하여 보았다. 설계된 카운터는 MS flip-flop을 이용하여 구조적으로 설계되었다.

[중간자료구조]

```
EXTERNAL
(EXTERNAL
 (MODmain COUNTER
  (CLEAR,CLOCK :IN ; A0, A1, A2, A3, A4, A5,
A6, A7 : OUT)
 )
 (MODsub MS-FF)
)
END_EXTERNAL
INTERNAL
(INTERNAL STRUCTURAL.COUNTER
 (NODE
 (1 IN CLEAR)(2 IN CLOCK)(3 OUT A0)(4
```

[GUI에서 설계된 하드웨어]



```
OUT A1)(5 OUT A2)(6 OUT A3)(7 OUT A4)(8
OUT A5)(9 OUT A6)(10 OUT A7)(11 SIG
TEMP0)(12 SIG TEMP1)(13 SIG TEMP2)(14
SIG TEMP3)(15 SIG TEMP4)(16 SIG
TEMP5)(17 SIG TEMP6)(18 SIG TEMP7)(19
MOD MS-FF)(20 MOD MS-FF)(21 MOD
MS-FF)(22 MOD MS-FF)(23 MOD MS-FF)(24
MOD MS-FF)(25 MOD MS-FF)(26 MOD
MS-FF)
)
(FUNCTION
(19 MOD MS-FF : 1 SIG CLEAR, 3 SIG A0, 11 SIG
 TEMP0, 2 SIG CLOCK, 3 SIG A0, 11 SIG TEMP0)
(20 MOD MS-FF : 1 SIG CLEAR, 4 SIG A1, 12 SIG
 TEMP1, 3 SIG A0, 4 SIG A1, 12 SIG TEMP1)
(21 MOD MS-FF : 1 SIG CLEAR, 5 SIG A2, 13 SIG
 TEMP2, 4 SIG A1, 5 SIG A2, 13 SIG TEMP2)
(22 MOD MS-FF : 1 SIG CLEAR, 6 SIG A3, 14 SIG
 TEMP3, 5 SIG A2, 6 SIG A3, 14 SIG TEMP3)
(23 MOD MS-FF : 1 SIG CLEAR, 7 SIG A3, 15 SIG
 TEMP4, 6 SIG A3, 7 SIG A4, 15 SIG TEMP4)
(24 MOD MS-FF : 1 SIG CLEAR, 8 SIG A5, 16 SIG
 TEMP5, 7 SIG A4, 8 SIG A5, 16 SIG TEMP5)
(25 MOD MS-FF : 1 SIG CLEAR, 9 SIG A6, 17 SIG
 TEMP6, 8 SIG A5, 9 SIG A6, 17 SIG TEMP6)
(26 MOD MS-FF : 1 SIG CLEAR, 10 SIG A7, 18 SIG
 TEMP7, 9 SIG A6, 10 SIG A7, 18 SIG TEMP7)
 )
)
END_INTERNAL
```

[생성된 VHDL]

```
entity RS_FF is
 port(r,s : in bit ; q,qbar : out bit) ;
end RS_FF ;

architecture rs_data of RS_FF is begin
 q <= r nor qbar after 3ns ;
 qbar <= s nor q after 3ns ;
end rs_data ;

entity MS_FF is
 port(Clr,R,S,C : in bit ; Q,Qbar : out bit) ;
end MS_FF ;

architecture ms_data of MS_FF is
 signal Clrbar,x0,y0,r0,s0,q0,qbar0,x1,y1,r1,s1,
    Cbar : bit ;
 component RS_FF
  port(r,s : in bit ; q,qbar : out bit) ;
 end component ;
begin
  Clrbar <= not Clr after 3ns ;
  x0 <= R and C after 3ns ;
  y0 <= S and C after 3ns ;
  r0 <= x0 or Clr after 3ns ;
  s0 <= y0 and Clrbar after 3ns ;
  rs0 : RS_FF port map(r0,s0,q0,qbar0) ;
  Cbar <= not C after 3ns ;
  x1 <= qbar0 and Cbar after 3ns ;
  y1 <= q0 and Cbar after 3ns ;
  r1 <= x1 or Clr after 3ns ;
  s1 <= y1 and Clarbar after 3ns ;
  rs1 <= RS_FF port map(r1,s1,Q,Qbar) ;
end ms_data ;
entity COUNTER is
 port(CLEAR,CLOCK : in but ; A0,A1,A2,A3,A4,
 A5,A6,A7 : out bit) ;
end COUNTER ;
architecture STRUCTURAL of COUNTER is
 signal TEMP0,TEMP1,TEMP2,TEMP4,
 TEMP5,TEMP6,TEMP7 : bit ;
 component MS_FF
  prot(Clr,R,S,C : in bit ; Q,Qbar : out bit) ;
 end component ;
begin
  l0 : MS_FF port map(CLEAR,A0,TMEP0,CLOCK,A0,
     TEMP0) ;
  l1 : MS_FF port map(CLEAR,A1,TMEP1,CLOCK,A0,
     A1,TEMP1) ;
  l2 : MS_FF port map(CLEAR,A2,TMEP2,CLOCK,A1,
     A2,TEMP2) ;
  l3 : MS_FF port map(CLEAR,A3,TMEP3,CLOCK,A2,
     A3,TEMP3) ;
  l4 : MS_FF port map(CLEAR,A4,TMEP4,CLOCK,A3,
     A4,TEMP4) ;
  l5 : MS_FF port map(CLEAR,A5,TMEP5,CLOCK,A4,
     A5,TEMP5) ;
```

```
l6 : MS_FF port map(CLEAR,A6,TMEP6,CLOCK,A5,
     A6,TEMP6) ;
l7 : MS_FF port map(CLEAR,A7,TMEP7,CLOCK,A6,
     A7,TEMP7) ;
end STRUCTURAL ;
```

Ⅲ. VHDL 지원환경

본 연구에서 구축된 지원환경[그림1]은 분석기. VHDL2CDFG 생성기, CDFG2VHDL생성기, 합성기 등으로 구성되어 있으며 시뮬레이터는 아직 개발되지 않았다. 지원환경의 개발은 ETRI(한국전자통신연구소)와 협력하여 본교에서 분석기와 VHDL2CDFG생성기, CDFG2VHDL생성기를 개발하였고 ETRI에서 개발된 CDFG 합성기를 연결하여 지원환경을 구성하였다. 본 장에서는 본 교에서 개발한 지원환경의 각 도구들을 기술하고 ETRI에서 개발된 합성기는 참고문헌으로 나타내었다[19]

1. 분석기

분석기는 사용환경에서 생성된 VHDL 프로그램을 입력으로 받아 구문검증과 의미검증을 수행한 후 AST형태의 중간코드를 생성하게 된다. 분석기의 구조를 [그림5]에 나타내었다[26,27]. 분석기는 VHDL의 전체사양(faullset)을 입력 받으며, 개발은 UNIX의 컴파일러 구성도구인 LEX와 YACC를 이용하여 설계하였다. VHDL의 BNF 형태를 YACC의 입력인 LALT(1)문법으로 변환하였고, 에러보정루틴, 심볼테이블생성루틴, 의미검사 루틴, AST 생성루틴이 포함되었다. VHDL 전체사양의 LALR(1)은 371개의 문법규칙과 1150개의 state로 구성되었다. 사용환경에서 구조적으로 설계된 하드웨어에 대해 자동 생성된 VHDL 프로그램은 설계된 하드웨어에서 사용된 모든 콤포넌트들을 ENTITY로서 프로그램에 나타낸다. 분석기는 이러한 다수개의 ENTITY를 포함하고 있는 프로그램을 처리할 수 있도록 개발하였다.

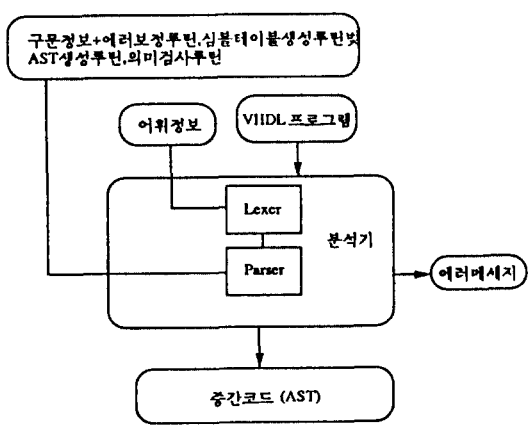


그림 5. 분석기의 구조

〈예〉분석기로부터 생성된 D flip-flop의 엔티티에 대한 중간코드 일부

```
/1 ; 2, lx_283, as_1 ; 340(1), as_1 ; 4, ls_294, as_1 ; 5, ,as
_1 ; 6, lx_282, as_1 ; 7, lx_340(1), as_1 ; 8, lx_343, as_1 ;
9, lx_314, as_5 ; 10, lx_356, as_5 ; 11, , as_5 ; 12, lx_357,
as_5 ; 13, lx_343, as_5 ; 14, , as_11 ; 15, lx_343, as_11 ;
16, , as_11 ; 17, , as_14 ; 18, lx_342, as_14 ; 19, lx_292, as
_14 ; 20, lx_340(4), as_14 ; 21, lx_340(5), as_16 ; 22, lx_
342, as_16 ; 23, lx_312, as_16 ; 24, lx_340(4), as_16 ; 25,
lx_340(2), as_17 ; 26, lx_341, as_17 ; 27, lx_340(3), as_
17 /
```

2. CDFG(Control/ Data Flow Graph) 정의

본 연구에서 정의된 CDFG는 상위 수준 합성시 필요로되는 자료구조의 특성을 가진 텍스트 형의 데이타 형식이다. CDFG는 VHDL의 엔티티(entity)에 대응하는 인터페이스기술 부분과 아키텍쳐(architecture)에 대응하는 함수 기술 부분으로 구성된다[23].

인터페이스 기술 부분은 해당 모듈이 외부 세계와 통신하는 수단을 나타내는 것으로서 입출력 단자들의 특성을 기술하게 된다. 만일 라이브러리가 존재한다면, 라이브러리의 이름과 지연 정보를 같이 나타낼 수 있다.

함수 기술에서는 새로이 구성하는 모듈의 세부 동작과 구조를 나타낸다. 함수기술 부분은 NODE, BEGIN, BLOCK, BRANCH, JOIN, RETURN, EVENT END절로 구서되며 절들은 순서없이 나타나며, 이들중 NODE, BLOCK, BRANCH, JOIN RETURN, EVENT 문장은 여러번 반복하여 나타날 수 있다. NODE 문장은 변수, 연산기호, 메모리 등의 노드들에 대한 정보를 표현하기 위한 부분이다. BEGIN 문장은 NODE 문장으로 정의한 노드들의 연산을 진행하기 위한 부분이다. BLOCK 문장은 노드들의 입출력 연결관계를 나타내는 부분으로서 BLOCK

문장의 자료는 데이터 패스(data path)합성 과정에 직접 이용된다. BRANCH 문장은 이벤트 읽기(event read)를 의미하는 문장으로서, 사용자가 가하는 인에이블(enable), 인터럽트(interrupt), 스타트(start) 및 각종 제어신호 또는 클럭신호 등을 읽어들이는데 이용한다. END 문장은 NODE 문장으로 정의한 노드들의 연산을 종료하기 위한 부분이다[23]. CDFG의 구문형태는 〈예6〉과〈예7〉에서 나타내었다.

전체 구조상으로 인터페이스 기술이 먼저 나타나고, 함수 기술이 나중에 나타난다.[그림6].

```
(인터페이스 기술)
(함수 기술
  (BEGIN절)
  (NODE절)
  (BLOCK절)
  (BRANCH절)
  (JOIN절)
  (EVENT절)
  (RETURN절)
  (END))
```

그림 6. CDFG의 전체 구성

3. VHDL2CDFG 생성기

본 연구에서 개발한 VHDL2CDFG 생성시스템은 3단계로 수행이 되도록 설계하였다[23,28]. 패스 1과 패스 2에서는 CDFG를 생성하기위한 테이블들이 생성되고 패스 3에서 이러한 테이블들을 이용하여 텍스트형태의 CDFG 화일을 생성한다. 생성기 구조는 [그림7]와 같다.

이러한 생성기는 테이블 핸들링 방법으로 설계되었으며 약 6500라인의 C프로그램으로 구성되었다.

〈예6〉VHDL로 기술된 전가산기를 VHDL2CDFG 생성기를 이용하여 CDFG로 변환한 예

```
--- VHDL 프로그램
ENTITY Full_Adder IS
 PORT(A, B, Carry_in : in BIT ;
   Sum, Carry_Out : out BIT) ;
end Full_Adder
ARCHITECTURE Behavior of Full_Adder IS
 signal S1, S2, S3 : Bit ;
```

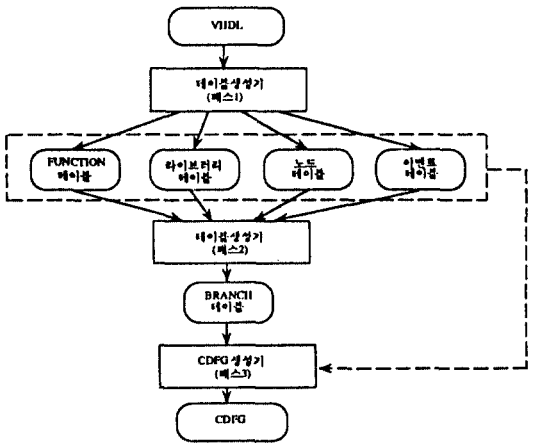


그림 7. VHDL2CDFG 생성기 구성도

```
begin
  S1 <= A xor B ;
  Sum <= S1 and Carry_in ;
  S2 <= A and B ;
  S3 <= S1 and Carry_in ;
  Carry_out <= S2 or S3 ;
end Behavior

--- 생성된 CDFG
(DEFUNC Full_Adder
(INPUT a 0 15, b 0 15, carry_in 015
 OUTPUT sum 0 15, carry_out 0 15))
(START Behavior)
(FUNCTION Full_Adder.Behavior
(NODE
(1 VAR a 0 15)(2 VAR b 0 16)(3 VAR carry_in 0 15)
(4 VAR sum 0 15)(5 VAR carry_out 0 15)
(6 VAR s1 0 15)(7 VAR s2 0 15)(8 VAR s3 0 15)
(9 OPS xor 0 15)(10 OPS xor 0 15)(11 OPS and 0 15)
(12 OPS and 0 15)(13 OPS or 0 15)
(BEGIN B1)
(BLOCK B1
(9_0 15 : 1_0 15, 2_0 15)
(6_0 15 : 9_0 15)
(10_0 15 : 6_0 15, 3_0 15)
(4_0 15 : 10_0 15)
(11_0 15 : 1_0 15, 2_0 15)
```

(7_0 15 ; 11_0 15)
(12_0 15 : 6_0 15, 3_0 15)
(8_0 15, 12_0 15)
(13_0 15 ; 7_0 15, 8_0 15)
(5_0 15 ; 13_0 15)
(NEXT END))
(END Full_Adder-Behavior))

4. CDFG2VHDL 생성기 개발

본 연구에서 구축된 지원환경[그림1]에서는 하드웨어 설계를 CDFG 편집기를 통해서도 설계할 수 있는 환경을 제공한다. 설계된 CDFG를 다시 VHDL로 변환기켜 줌으로서 설계된 하드웨어의 문서화(documentation)와 상용화된 VHDL 시뮬레이터에서 검증이 가능하다.

설계된 CDFG는 VHDL을 생성하기위해 테이블형태의 자료구조로 분해되며 다시 VHDL을 생성하기 위해 조합된다[24,29]. 이러한 CDFG2VHDL 생성기의 개발은 2단계로 구성되었는데 패스1에서는 CDFG 구문을 구분하여 각 테이블에 저장하였고 패스2에서는 이러한 테이블들을 이용하여 정의된 VHDL을 생성하게 된다. 시스템의 전체적인 구성은 [그림8]과 같다.

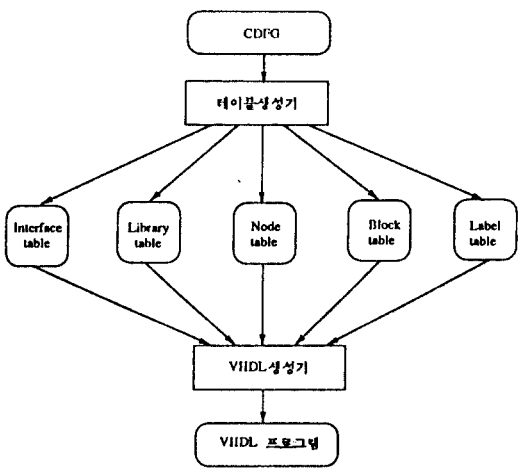


그림 8. CDFG2VHDL 생성기 구성도

〈예7〉CDFG로 기술된 멀티프렉서를 CDFG2VHDL 생성기를 이용하여 VHDL로 변환한 예

--- 입력된 CDFG

(DEFUNC mux 0 0
(INPUT i0 0 0, i1 0 0, i2 0 0, i3 0 0, a 0 0 , b 0 0
OUTPUT q 0 0)
(LIB STD.std_logic.ALL)
(FUNCTION mux.behave 0 0
(NODE
(1 VAR i0 0 0)(2 VAR i2 0 0)(4 VAR i3 0 0)
(5 VAR a 0 0)(6 VAR b 0 0)(7 VAR q 0 0)(8
VAR sel 0 1)(9 OPS=0 0)(10 OPS + 0 0)(11
OPS = 0 0)(12 OPS + 0 0)
(BEGIN B1
(EVENT B1
(1 i0 0 0 MODE CHANGING)
(2 i1 0 0 NODE CHANGING)
(3 i2 0 0 MODE CHANGING)
(4 i3 0 0 MODE CHANGING)
(5 a 0 0 MODE CHANGING)
(6 b O O MODE CHANGING)
(NEXT B2))
(BLOCK B2
(8 SEL 0 0 ; CONST 0)
(NEXT B3))
(BLOCK B3
(9 = 0 0 ; 5 a 0 0, CONST 1)
(NEXT B4))
(BLOCK B4
(COND 9 1,NEXT B5)
(COND 9 OTHERS, NEXT B6))
(BLOCK B5
(10 + 0 0 ; 8 sel 0 0 , CONST 1)
(8 sel 0 0 ; 10 + 0 0)
(BLOCK B6
(11=0 0 ; 6 b 0 0 ; CONST 1)
(NEXT B7))
(BRANCH B7
(COND 11 1,NEXT B8)
(COND 11 OTHERS, NEXT B9))
(BLOCK B8
(12 + 0 0 ; 8 sel 0 0,CONST 2)

```
 (8 sel 0 0 ; 12 + 0 0)
(BRANCH B9
 (COND 8 B00, NEXT B10)
 (COND 8 B01, NEXT B11)
 (COND 8 B11, NEXT B13)
(BLOCK B10
 (7 q 0 0 : 1 i0 0 0)
 (NEXT B14)
(BLOCK B11
 (7 q 0 0 : 2 i1 0 0)
 (NEXT B14))
(BLOCK B12
 (7 q 0 0 : 3 i2 0 0)
 (NEXT B14))
(BLOCK B13
 (7 q 0 0 : 4 i3 0 0)
 (NEXT B 14))
(END)
```

--- 생성된 VHDL

```
USE STD.std-logic.All
ENTITY mux IS
 PORT (IQ,I1,I2,I3,A,B : IN BIT ;
       Q : OUT BIT) :
END mux;

ARCHITECTURE mux-behave OF mux IS
 SIGNAL SEL : INTEGER RANGE 0 TO 3 :
BEGIN
 B : PROCESS(A,B,I0,1I1,I2,I3,)
 BEGIN
  sel<=0 ;
  IF (A=1) THEN sel <=sel+1; END IF ;
  IF (B=1) THEN sel <=sel+2; END IF ;
  CASE sel IS
  WHEN 0 =>
    Q <= I0 ;
  WHEN 1 =>
    Q <= I1 ;
  WHEN 2 =>
    Q <= I2 ;
  WHEN 3 =>
    Q <= I3 ;
  END CASE ;
 END PROCESS ;
END mux_behave ;
```

Ⅳ. 결 론

본 논문에서는 본 교에서 개발한 VHDL환경에 관해 기술하였다. 전자통신연구소와 협력하여 개발된 도구들을 취합하여 지원환경을 구성하였으며 이러한 지원환경과 연계하여 사용환경을 설계 및 구현 하였다. 본 논문에서는 지원환경의 각 도구들에 대한 내용[26-29]은 간략히 기술하였고 주로 사용환경에 관해 기술하였다.

현재 외국에서 반입되고 있는 도구들은 부분 도구로서 전체적인 환경을 제공하고 있지 않은 실정이다. 본 연구에서 개발된 도구들과 외국 도구들과 의 비교는 분석기와 사용환경에서 대등한 성능을 보이고 있다[3,14,15,16]. 따라서 이러한 환경을 개발함으로써 고가의 외국도구들을 대체할 수 있는 기반을 마련하였다.

본 연구에서 구축된 VHDL환경의 각 도구들을 보강하여 새로운 버전으로 향상시키고, 지원환경에 시뮬레이터, 라이브러리 매니저, simplifier등을 추가하는 것이 향후 연구과제이고 본 연구와 연관되어 계속 연구될 것이다.

앞으로 하드웨어 설계 분야에서 VHDL의 사용이 더욱 강력히 요구될 것이며 이에 따라 VHDL의 모든 의미를 완벽하게 반영하는 지원환경과 다양하고 편리한 사용환경이 요구된다. 따라서 본 연구에서 구축된 VHDL 사용환경 및 지원환경은 VHDL의 국내환경조성과 하드웨어 설계분야에서 VHDL의 사용을 확산시키고 사용자 편의를 제공하는데 기반이 될 것이다.

참 고 문 헌

1. IEEE Standard VHDL Language Reference Manual, IEEE Inc
2. Douglas L. Perry "VHDL," 1991.
3. Choon B. Kim "Multiple Mixed-level HDL Generation from Schematic for ASIC Design" Fourth Annual IEEE International ASIC Conference and Exhibit, 1991.
4. Z. Navabi, S. Day "Tutorial on Use of VHDL

for Description of Digital Systems" ASIC Coference, 1991.
5. Roger Lipsett, "VHDL : Hardware Description and Design," Intermetrics Inc, 1989.
6. Daniel Fischer, Yossi Levhari, Gadi Singer "NETHDL : Abstraction of Schematics to High-Level HDL," IEEE DAC, 1990.
7. Blaauw, Saav, "Automatic Generation of Behavioral Moduls from Switch-level Description" Proc.DAC, 1988.
8. J.P.Schupp, J.Cockx, L.Claesen, H.De Man "SPI : An Open Interface Integrating Highly Interactive Electronic CAD Tools," IEEE DAC, 1990.
9. P.R.Santos, H.Sarmento, L.Vidigal "Ghost / Spook User Interface and Process Management in DACE framework," IEEE DAC, 1990.
10. Joseph W.Sullivan, Sherman W.Tyler "Intelligent User," ACM PRESS, 1991.
11. David R.Codelho "The VHDL Handbook," Kluwer Academic Publishers, 1989.
12. Naba Barkakati "X Window System Programming," SAMS, 1991.
13. Robert W.Scheifler, James Gettys "X Window System," Digital Press, 1990.
14. Dinesh Bettadaput "VHDL-A Management Overview" Fourth Annual IEEE International ASIC Conference and Exhibit, 1991.
15. VHDL Export tools, Intergraph Company
16. VHDL Toolset, university of Pittsburgh
17. 조중휘 "VHDL의 중간언어 정의 및 Reverse Analyser 구현," 최종연구보고서, 과학기술처, 1990.
18. 황선영, 이영희 "VHDL 설계환경 구축을 위한 Front-end의 설계" 한국정보과학회 논문지 18권 1호, 1991.
19. 전자통신연구소, "자동설계환경구축에관한연구 IV 상위수준합성기 개발," 최종연구보고서, 과학기술처, 1992.
20. 원유헌, 김충석, "VHDL Analyzer의 parser개발," 1989, 과학기술처 88 특정연구 결과 발표회 논문집.
21. 원유헌, 김충석, "VHDL Analyzer개발," 1990, 과학기술처 89 특정연구 결과발표회 논문집, 대한전자공학회, 대한 전기공학괴, 한국정보과학회, 한국통신학회.
22. 원유헌, 김충석, "VHDL지원환경을 위한 Frontend tool 개발에 관한 연구" 1988, 1989년도 반도체, 재료 및 부품 연구회, CAD 연구회 합동 학술발표회 논문집, 대한전자공학회, 대한전기공학회
23. 원유헌, 김충석, "VHDL 지원환경에서 CDFG생성," 한국정보과학회 춘계학술발표회 논문집, 1991.
24. 원유헌, 김충석, "CDFG로부터 VHDL Subset 생성," 한국정보과학회 춘계학술발표회 논문집, 1992.
25. 원유헌, 표창우, 김충석, 이준동 "X window를 이용한 Schematic Capture System으로 부터 VHDL 생성," 한국정보과학회 춘계학술발표회 논문집, 1992.
26. 원유헌, "VHDL Analyzer의 parser개발," 최종연구보고서, 과학기술처, 1989.
27. 원유헌, "VHDL Analyzer 개발," 최종연구보고서, 과학기술처, 1990.
28. 원유헌, "VHDL 중간코드로부터 CDFG생성," 최종연구보고서, 과학기술처, 1991.
29. 원유헌, "CDFG로부터 VHDL Subset 생성," 최종연구보고서, 과학기술처, 1992.

金 忠 錫(Choung Seok Kim) 正會員
1960년 12월 29일생
1986년 : 弘益大學校 電子計算學科 卒業(理學士)
1988년 : 弘益大學校 大學院 電子計算學科 卒業(理學碩士)
1990년 : 弘益大學校 大學院 電子計算學科 博士課程修了
1990년~現在 : 釜山女子大學校 電子計算學科 助敎授

元 裕 薰(Yoo Hun Won) 正會員
1948년 12월 10일생
1972년 : 成均館大學校 數學科 卒業(理學士)
1975년 : 韓國科學院 電子計算學科 卒業(碩士)
1985년 : 高麗大學校 大學院 卒業(理學博士)
1976년~現在 : 弘益大學校 컴퓨터工學科 敎授

表 昌 祐(Chang Woo Pyo) 正會員
1957년 10월 22일생
1980년 : 서울大學校 電子工學科 卒業(學士)
1982년 : 서울大學校 大學院 電子計算機工學科 卒業(碩士)
1989년 : University of Illinois at Urbana-Champaign, Ph.D. in Computer Science
1991년~現在 : 弘益大學校 컴퓨터工學科 助敎授