Verilog-2001 파서의 설계와 구현

김 영 수† · 김 태 석† · 김 상 필† · 조 한 진††

요 약

본 논문에서는 IEEE Verilog 1364-2001 표준을 지원하는 Verilog-2001파서를 개발하였다. 개발된 파서의 어휘 분석 및 구문 분석기는 Verilog-2001을 지원할 수 있도록 개발되었으며 Verilog-2001 테스트 슈트(test suite)를 개발하여 검증하였다. XML 중간형식을 설계하여 사용함으로써 Verilog시맨틱 조작의 응용에 적합하게 설계되었으며 기존의 구조 수준의 파서의 단점을 극복하기 위하여 문맥 정보의 처리가 가능하도록 개발되었다.

The Design and Implementation of Verilog-2001 Parser

Youngsoo Kim† · Taesuk Kim† · Sangpil Kim† · Hanjin Cho††

ABSTRACT

The Verilog parser libary for IEEE Verilog 1364-2001 Standard is developed in the paper. The lexer and scanner are developed and tested to handle "Verilog-2001" which is the first major update to the Verilog language since its inception in 1984. Also the newly developed XML intermediate format for Verilog-2001 is presented. By using the XML intermediate, it allows the portable and scalable development of various kinds of verilog applications that are mainly focused on semantic manipulation.

키워드 : Verilog, 파서, XML

1. 서 론

Verilog HDL은 Gateway사에 의해서 1984년 소개되었으며 Cadence사로 합병된 이후에 1995년 발표된 IEEE 1364-1995에 의해 공식 Verilog 표준이 되었다.

Verilog를 이용한 하드웨어 설계 흐름은 보편화되어 있으며 이에 따른 시뮬레이션, 합성 및 검증 등의 도구들도 Verilog 기반으로 개발되고 있다. 이러한 도구들을 개발하기 위해서 Verilog 구문 및 문맥의 분석을 포함한 일련의 절차적 접속을 위한 API(Application Programming Interface)들이 정의되어야 할 뿐만 아니라 도구간의 접속을 위한 중간 형식도 표준화되어야 한다.

기존의 Verilog 파서 및 표준 인터페이스 라이브러리들은 툴 벤더를 중심으로 개발되어 있다.

U. C. Berkeley의 VIS(Verification Interacting with Synthesis)의 일부인 vl2mv는 Verilog 프론트엔드로서 BLIF-MV 형식으로 변환하는 툴이다. BLIF-MV는 HSIS의 입력 형식으로 기본적으로 모듈 선언, 생성 및 심볼릭 래치(symbolic latch) 형태의 기술로 구성되어 있다. vl2mv는 구조적인 정보를 얻어 내는 데는 사용될 수 있으나 이를 Verilog의 시맨틱을 얻거나 조작하는 등의 레지스터 전송 수준의 조작은 힘든 단점을 가지고 있다[1].

Cadence사의 Verilog Procedural Interface는 PLI(Programming Language Interface) 표준에 근거한 것이며 C 접속 API 들을 제공한다. Verilog 모듈에 대한 내부 데이터 구조에 접근할 수 있으며 정보를 추출해 낼 수 있다. PLI는 access와 utility 함수로 구성되어 있으며 사용자의 어플리케이션에서 호출하여 사용할 수 있다. PLI 루틴들은 시뮬레이터와 접속하여 모듈의 구조정보와 이벤트 오더링(event ordering) 및 Verilog 데이터의 읽기와 쓰기 등을 지원한다[2]. 그러나 시뮬레이터 접속을 위한 표준으로 개발되었기 때문에[3, 4] 로직 시뮬레이터나 타이밍 분석기 등의 CAD 툴 이외에 시멘틱 정보들을 얻거나 조작해야 하는 코드 룰 체커나 코드 커버리지 체커의 개발에는 적합하지 않다.

본 논문에서는 Verilog 1364-2001(이하 Verilog-2001)을 지원하는 Verilog-2001 파서를 개발하였다. Verilog 중간형식을 XML(Extensible Markup Language)로 정의함으로써 XML 자체가 가진 확장성과 이식성을 최대한 활용하였으며 Verilog의 시멘틱을 효과적으로 조작할 수 있는 절차 접속 API와 함께 XML의 XSL(Extensible Stylesheet Language)

† 정 회 원 : 한국전자통신연구원 시스템 IC 설계팀
†† 정 회 원 : 한국전자통신연구원 시스템 IC 설계팀 팀장
논문접수 : 2003년 3월 26일, 심사완료 : 2003년 8월 7일

유틸리티를 이용한 접속 방법을 개발하였다. 또한 이를 이용한 Verilog 코드 룰 체커의 개발 사례를 소개하였다.

2. Verilog-2001 표준 분석

현재의 IEEE Verilog 1364-1995 표준 이후에 2001년에 IEEE Verilog 1364-2001 표준이 제정되었다[5-7]. 개발된 Verilog-2001 파서는 IEEE Verilog 1364-2001 표준에 근거하여 개발되었으며 IEEE Verilog 1364-1995에 대비하여 추가된 사항은 크게 deep-submicron 대비한 IP 모델링을 위한 기능과 시뮬레이션 및 합성 툴 벤더의 요구사항을 반영한 것이다. 또한 IEEE Verilog 1364-1995 표준에 있던 모호함과 에러도 수정되었다. 각각의 세부적인 사항은 다음과 같다.

2.1 모델링 개선 사항

Verilog 모델링의 개선을 위한 21개의 추가된 항목 중에서 대부분은 합성 가능한 RTL 모델의 기술에 용이성과 정확성을 향상시켰고, 나머지 항목들은 모델의 확장성과 재사용성을 향상시켰다.

2.1.1 설계 관리를 위한 컨피규레이션(configurations)

각 Verilog 모듈의 정확한 버전과 소스 코드 로케이션을 Verilog의 일부분으로 상세화 하는 것을 허용한다. 이식성을 위해서 virtual model library는 컨피규레이션 블록 내에서 사용되고, virtual library를 physical location과 연결해주는 라이브러리 맵 파일을 분리하였다. 모듈 정의의 바깥 부분에 서술되며 컨피규레이션 블록은 모든 혹은 특정한 모듈 인스턴스의 소스 코드 로케이션을 기술한다.

2.1.2 스케일러블한 모델링을 위한 generate 루프

모듈 및 프리미티브들의 인스턴스를 다중 생성하기 위하여 사용되며 이외에 variables, nets, tasks, functions, continuous assignments, initial procedures, always procedure의 다중 생성을 위하여 사용되도록 추가되었다.

2.1.3 상수 함수(constant function)

상수 함수의 정의는 다른 Verilog 함수와 동일하다. 컴파일 또는 elaboration 시에 값이 결정되는 구성 요소를 사용하는 것으로 제한된다.

2.1.4 인덱스 벡터의 부분 선택(indexed vector part select)

변수를 사용하여 특정 바이트를 선택하기 위해 사용된다. base expression은 width expression, offset direction으로 구성되며 시뮬레이션(simulation run-time) 동안에 변할 수 있다. width expression은 반드시 상수이어야 한다. offset direction은 base expression으로부터 width expression을 더할 것인지 뺄 것인지를 나타낸다.

2.1.5 다중 배열의 부분 선택

다중 배열을 지원한다. variable과 net 데이터 타입에 대한 배열을 지원한다. 또한 배열의 경우 array word에 대한 bit selects와 part select를 직접 접근할 수 있도록 하였다.

이외에 signed arithmetic extensions, arithmetic shift operator, signed radix, power operator, re-entrant tasks and recursive function, combinational logic sensitivity 토큰, comma-separated sensitivity list 등이 확장되었다.

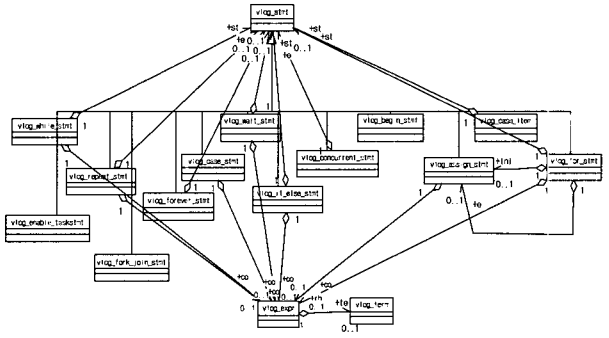
2.2 ASIC/FPGA를 위한 정확도 개선 사항

Verilog는 2~5㎛ 공정을 위한 설계 언어로 만들어졌기 때문에 공정기술에 따른 변화에 따라 deep-sub micron 공정을 대비한 정확도 개선을 위한 항목들이 on-detect pulse error propagation, negative pulse detection 및 새로운 timing constraint check 관련 항목들이 새로이 추가되었다.

3. Verilog-2001 파서의 구조

3.1 Verilog-2001 파서의 클래스 다이어그램

Verilog-2001에 대한 표준을 만족하는 Verilog-2001 파서를 개발하기 위하여 분석 및 설계한 IEEE Verilog 1364-2001 표준의 일부인 스테이트먼트(statement) 클래스 다이어그램이 (그림 1)에 표시되어 있다. UML(Unified Modeling Language)을 사용하여 분석하였으며 분석 도구로서 Rational Rose를 사용하였다[8].



(그림 1) Verilog-2001 파서의 UML 다이어그램

UML 모델링 기법 중에서 정적인 모델링인 클래스 다이어그램을 추출하는 것은 중간 형식 파일을 구성할 때 요소와 속성을 정의할 때뿐만 아니라 Verilog-2001 파서의 클래스를 설계할 때 중요한 개발 항목이 된다.

Verilog-2001 파서의 전체구조는 (그림 2)와 같다. Verilog-2001로 기술된 소스 파일을 받아서 어휘 분석과 구문 분석을 한 후에 중간형식으로 저장한다. 중간형식은 XML로 기술되며 이에 대한 처리는 XSL을 이용하여 할 수도

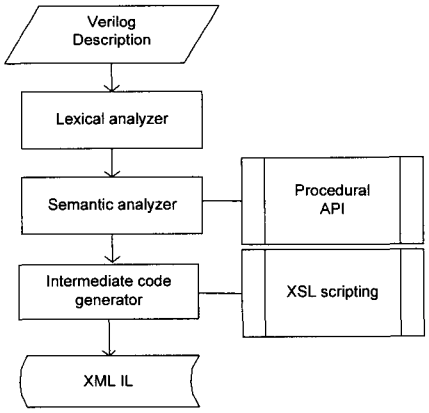
있으며 미리 정의된 접속 API를 이용하여 할 수도 있다.

3.2 어휘 분석기 및 구문 분석기

Verilog 1364-2001 표준에 추가된 항목들을 지원하는 문법을 기술하여 ANTLR(ANother Tool for Language Recognition)가 제공하는 프레임워크와 Visual C++ .Net 7.0 도구를 이용하여 어휘 및 구문분석기를 개발하였다[9, 10].

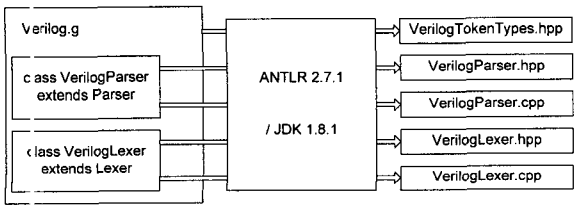
ANTLR은 C++ 또는 Java로 표현된 액션을 포함하는 문법적 표현으로부터 인식기 및 트랜스레이터에 대한 프레임워크를 제공하는 툴이며 Verilog-2001 문법을 정의하여 이에 따른 어휘분석기 및 구문분석기를 생성하여 사용하였다. Verilog-2001 문법을 grammar file(.g)에 정의하여 이에 따른 어휘분석기 및 구문분석기를 생성하고, 리스트와 셋 등의 STL(Standard Template Library)에서 제공하는 컨테이너, 반복자와 알고리즘을 사용하여 저장된 정보에 대한 삽입, 검색 및 삭제 연산을 수행하는 기본 액션을 문법 파일 내에 정의하였다.

Verilog-2001 문법은 VerilogLexer와 VerilogParser에 규칙을 정의한다. 규칙은 LL(3)의 문법으로 구성되고, 각 규칙에는 해당 규칙이 입력되었을 경우에 수행해야 할 액션을 기술하고, 이 규칙은 생성되는 두 개의 클래스에서 멤버 메소드(member method)로 구현된다.



(그림 2) Verilog-2001 파서의 전체 구조

(그림 3)은 ANTLR을 이용하여 문법 파일의 VerilogLexer와 VerilogParser 클래스(class)로부터 생성되는 헤더 파일과 소스 파일의 종류를 나타낸다.



(그림 3) Verilog 파서의 어휘분석기/구문분석기 개발

3.3 Verilog 중간 형식

표준화된 중간형식을 사용하게 되면 개발할 CAD 도구의 전반부와 후반부를 각각의 경우를 따로 개발할 필요 없이 중간형식을 대상으로 개발할 수 있다.

HDL(Hardware Description Language)의 중간형식을 표준화하려는 노력들은 이미 오래 전부터 있어왔다. 대표인 것은 AIRE/CE(Advanced Intermediate Representation with Extensibility/Common Environment)로서 IIR(Internal Intermediate Representation)과 FIR(File Intermediate Representation)로 나눌 수 있으며 IIR 초안이 나와 있으나 VHDL과 VHDL-AMS 형식만이 지원되며 플랫폼에 따른 이식성이 떨어지는 단점을 가지고 있다[11, 12].

위와 같은 내부적 중간 형식 이외에도 파일을 기반으로 한 시도들도 있었으며 대표적인 것은 XML을 기반으로 한 중간형식들이다. 현재 XML은 네트워크를 통한 데이터 교환에 효율적인 형식으로 표준화되었으며 특히 어플리케이션과 어플리케이션간의 데이터 교환의 표준으로 자리잡고 있다.

계층적인 하드웨어 모듈의 연결을 기술하기 위한 형식으로 개발된 MoML(Modeling Markup Language)은 XML 기반의 마크업 언어로써 웹과의 통합 편이성을 극대화하고 그래픽을 위한 주석 등의 추가들을 하였으나 HDL의 중간형식에 사용하기에는 적당하지 않은 단점을 가지고 있다[13].

이외에 EdaXML이 XML 기반의 형식으로 정의되어 회로의 스키매틱을 기술하기 위하여 사용되는 EDIF 형식을 대체하기 위해서 제안되었으나 주로 보드 수준의 부품들의 연결정보 혹은 구조 정보를 기술하기 위한 형식으로 사용가능

```
always @ (negedge rst)
begin
  rwb = 1 ;
  pc = 0 ;
  acc = 0 ;
  state = 'READ_INST' ;
end
```

(a)

```
<? xml version = "1.0" encoding = "EUC-KR" standalone = "yes" ?>
< vlog_description >
 < vlog_module >
  < vlog_concurrent_stmt type = "nededge" sen = "rst" >
   < vlog_begin_end_stmt >
     < vlog_assign_stmt lhs = "rwb" rhs = "1" />
     < vlog_assign_stmt lhs = "pc" rhs = "0" />
     < vlog_assign_stmt lhs = "acc" rhs = "0" />
     < vlog_assign_stmt lhs = "state" rhs = "READ_INST" />
   </ vlog_begin_end_stmt >
  </ vlog_concurrent_stmt >
 </ vlog_module >
</ vlog_description >
```

(b)

(그림 4) XML 형식의 Verilog 중간형식

하며 HDL의 시멘틱들을 정의하여 사용할 수 없다[14-17].

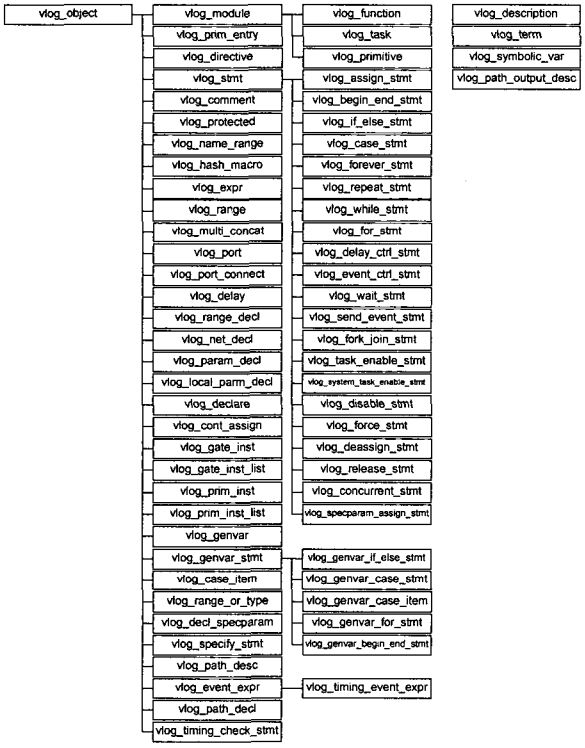
본 논문에서는 Verilog HDL을 기반으로 하는 CAD 도구의 중간형식을 XML로 정의함으로써 XML 자체가 가진 확장성과 이식성을 최대한 활용하여 기존의 중간형식에 비해 더욱 효율적으로 CAD 도구의 개발이 가능함을 보여주고 있다. Verilog HDL의 XML DTD을 정의하고 이를 이용한 Verilog 중간 형식을 설계하였다.

본 논문에서 중간형식을 정의하기 위해 사용한 XML은 SGML(Standard Generalized Markup Language) 기반의 마크업 언어로서 개방적 구조 및 확장성을 가지며 기본적으로 계층적인 트리 형식이기 때문에 대부분의 데이터 구조를 모델링 하는데 편리하다.

(그림 4)(a) Verilog 파일의 일부이며 (그림 4)(b)는 이를 변환한 XML 중간 형식의 일부이다. 클래스로 모델링 한 개개의 클래스와 속성들이 엘리먼트와 속성으로 변형되었다.

3.4 Verilog-2001 파서의 세부 구현 및 기본 API 개발

Verilog 파서의 세부 클래스 구현은 (그림 5)와 같다. Verilog 표준의 구성 요소를 각각의 객체로 구별하여 클래스를 정의하였다.



(그림 5) 전체 클래스 구현

vlog_object 클래스는 Verilog의 각 요소에 대한 베이스 클래스로서 파싱될 Verilog 파일에 대한 정보를 포함하고, 다른 클래스에 대한 부모 클래스로서 역할을 하게 된다.

vlog_object 클래스는 vlog_module 클래스를 프랜드 클래스로 가지며 vlog_module 클래스는 네트의 선언 및 이름들과 해당하는 값들을 포함하는 모듈 내부에 선언된 값들의 리스트를 가지고 있는 기본적인 클래스이다.

또한 vlog_module 클래스는 모듈 내부의 여러 정보들을 포함하고, vlog_task, vlog_function, vlog_primitive 클래스 등을 프랜드 클래스로 가진다.

vlog_primitive 클래스는 조합회로인 경우와 순차회로인 각각의 경우에 각각의 선언된 리스트 등의 정보를 가지고 있다. vlog_function과 vlog_task는 각각 Verilog 함수와 태스크에 대하여 각각의 이름과 파라미터 및 선언된 스테이트먼트(statement)등을 가진다.

기본 API는 공통적으로 객체 생성, 삽입 및 검색 함수를 가지고 있으며 자세한 구현 사항은 다음에 상술하였다.

3.4.1 객체 생성

각 클래스에는 생성자를 이용한 객체 생성 함수를 포함하고 있다. 이러한 생성 함수는 룰의 액션 부분에 포함되고, (그림 6)과 같은 형태를 가진다.

```
// VlogClass.cpp
vlog_port *
vlogCreatePort (short t, vlog_name_range *name,
list < vlog_name_range *> *exprs)
{
    vlog_port *port = new vlog_port() ;

    port → type (t) ;
    port → name = name ;
    port → nameList = exprs ;
    return(port) ;
}
```

(그림 6) vlogCreatePort 함수의 예

이러한 생성 함수는 (그림 7)에 예시되어 있는 것과 같이 문법 파일에 선언되어 있는 규칙에 대한 액션 부분에 포함되어서, 해당되는 객체를 생성하는데 사용된다.

```
// Verilog.g

port
returns [ vlog_port *ret_port = NIL (vlog_port) ]
{
list < vlog_name_range *> *in_port_expression =
NIL(list<vlog_name_range *>) ;
vlog_name_range *in_name_of_port = NIL (vlog_name_range) ;
}
 :
(in_port_expression = port_expression)?
{
ret_port = vlogCreatePort (MODULE_PORT, (vlog_name_range *)
NULL, in_port_expression) ;
}
| DOT in_name_of_port = name_of_port LPAREN (in_port_
expression = port_expression)? RPAREN
{
ret_port = vlogCreatePort (NAMED_PORT, in_name_of_port,
```

```
in_port_expression) ;
}
;
```

(그림 7) Verilog.g 문법 파일의 예

3.4 2 객체 삽입 및 검색

위에서 만들어진 객체들은 각 클래스에 정의된 멤버 메소드를 이용하여 리스트나 셋 등의 컨테이너에 저장된다.

```
// Verilog.g

list_of_ports
return; [list < vlog_port * > *ret_list_of_ports = NIL (list < vlog_port *>)]
{
vlog_port *in_port1 = NIL (vlog_port) ;
vlog_port *in_port2 = NIL (vlog_port) ;
}
:
LPAREN
in_port:1 = port
{
ret_list:_of_ports = new list < vlog_port *> ;
ret_list:_of_ports → push_back (in_port1) ;
}
(COMMA in_port2 = port
{
ret_list:_of_ports->push_back (in_port2) ;
}
)*
RPAREN
;
```

(그림 8) 객체 삽입의 예

위와 같이 만들어진 포트 리스트에서 특정한 포트에 대한 검색은 다음과 같이 이루어진다. 포트의 이름이 "co"라고 가정한다면 포트에 대한 클래스인 vlog_port를 리스트 형태로 저장하고 있는 port_list에서 co에 대한 정보를 가져온다. 이 경우에 먼저 리스트에 대한 포인터 역할을 하는 반복자를 선언하고, 리스트의 처음부터 끝까지 vlog_port 클래스의 name 영역에 있는 값이 "co"인 것을 찾는다. 예제 코드는 (그림 9)에 표시되어 있다.

```
// …
vlog_port *port
list <vlog_port *> :: iterator portgen ;
for ( portgen = port_list → begin() ; portgen != port_list → end() ;
++portgen)
{
port = *porgen;
if (port → name == "co")
// …
}
```

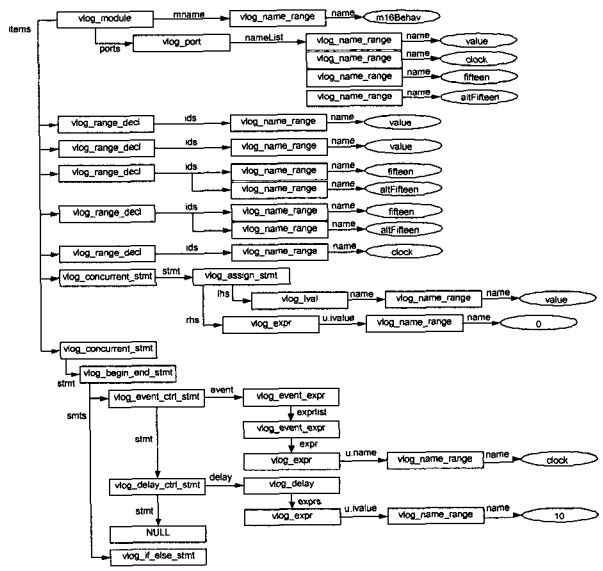
(그림 9) 포트 검색의 예

본 논문에서 개발한 Verilog 파서의 전체 동작을 설명하기 위하여 (그림 10)에 표시된 m16Behav 모듈의 예제 코드를 대상으로 하여 어떻게 동작되는지를 (그림 11)에 표시하였다.

```
// alternate behavioral definition of m16Behav
module m16Behav (value, clock, fifteen, altFifteen) ;
output [3 : 0] value ;
reg [3 : 0] value ;
output fifteen, altFifteen ;
reg fifteen, altFifteen ;

input clock ;

initial value = 0 ;

always
        begin
        @(negedge clock) #10 value = value + 1 ;
                if (value == 15)
                        begin
                        altFifteen = 1 ;
                        fifteen = 1 ;
                        end
                else
                        begin
                        altFifteen = 0 ;
                        fifteen = 0 ;
                        end
        end
endmodule
```

(그림 10) Verilog 예제 코드

(그림 11)에서 보는 바와 같이 Verilog 코드를 어휘 분석기와 구문 분석기가 입력을 받아 해당하는 액션이 호출된다. Verilog 문법의 액션에는 각각의 Verilog 클래스의 기본 함수들로 작성이 되어 있으며 이들의 함수를 통하여 각각의 클래스의 멤버들이 (그림 11)에 표시된 순서대로 채워지게 된다.



(그림 11) Verilog 파서의 동작 흐름

3.5 XSL을 이용한 Verilog-2001 파서 적용

정의한 XML 중간형식을 이용한 Verilog-2001 파서를 이용하여 실제의 CAD 툴 개발에 적용하였다. 적용이 가능한 분야는 Verilog 파서가 필요한 전 분야로서 XML의 유틸리티인 XPath와 XSL을 적용하면 Verilog 소스 파일을 처리하는 루틴을 기존의 하드 코딩에 의한 방법에 비해 손쉽게 개발할 수 있다. 뿐만 아니라 중간 형식이 XML로 구성되어 있어 계속해서 개발되고 있는 XML의 유틸리티들을 적용이 가능하여 CAD 툴 개발에 필요한 시간을 단축시킬 수 있는 장점을 가지게 된다.

본 논문에서 개발한 Verilog-2001 파서를 이용하여 Verilog 코드 룰 체커의 개발에 적용하였다. Verilog 코드 룰 체커는 Verilog로 기술되어 있는 설계사양을 검사하여 사양이 Verilog 합성의 표준에 부합되는가를 판별하는 역할을 하는 툴로서 코드 커버리지 체커와 함께 Verilog 시맨틱을 조작해야 하는 대표적인 CAD 툴의 한 종류이다.

XML 형식으로 생성된 Verilog 중간 형식을 처리하는데 사용된 XSL은 정규식에 기반한 XPath 표현식들을 사용하여 XML로 기술된 Verilog 파일에 대해서 특정 엘리먼트들과 이들의 패턴을 검색하고 처리할 수 있다[18].

```
<? xml version = "1.0" encoding = "EUC-KR" standalone = "yes" ?>
< vlog_description >
  < vlog_module name = "m" >
    < vlog_reg_name_range name = "a" />
    < vlog_reg_name_range name = "b" />
    < vlog_reg_name_range name = "c" />
    < vlog_always_stmt rel = "or" sen = "a" sen1 = "b" >
      < vlog_begin_end_stmt >
        < vlog_if_stmt cond = "a" />
        < vlog_assign_stmt lhs = "c" rhs = "a" />
      </ vlog_begin_end_stmt >
    </ vlog_always_stmt >
  </ vlog_module >
</ vlog_description >
```

(a)

```
< xsl : template match = "vlog_if_stmt/ [vlog_if_else_stmt]" >
  < xsl : value-of select = "." />
</ xsl : template >
```

(b)

(그림 12) XSL를 이용한 접속 API 적용

(그림 12)(a)는 Verilog의 중간형식으로서 Verilog 소스 파일이며 (그림 12)(b)는 XSL을 이용한 접속 API의 개발로서 else 문이 빠진 경우의 Verilog 코드를 검색하여 표시하는 기능을 한다. 위와 같은 XML를 이용한 Verilog 중간형식의 장점은 강력한 Verilog 파서를 개발하는데 적합하며 XML 파일을 생성하는 부분만을 개발하면 이에 따른 파싱 및 이의 처리는 공개된 XML 파서 및 XSL 프로세서 등을 사용하여 손쉽게 스크립팅할 수 있는 장점을 가지고 있다.

4. Verilog-2001 파서의 검증

본 논문에서 개발한 Verilog-2001 파서를 검증하기 위하여 Icarus Verilog 테스트 슈트를 이용하였다[19]. Icarus Verilog 테스트 슈트는 기본적으로 Verilog-1995에 기반한 테스트 항목으로 구성이 되어 있으며 Verilog-2001 항목을 검증하기 위하여 Verilog-2001 테스트를 추가로 작성하였다. 추가된 테스트는 셀프 테스트가 가능하도록 작성되었다.

〈표 1〉 Verilog 테스트 슈트 목록

구 분	키워드	테스트 항목
Verilog-1995	always	blocking assignment
		non-blocking assignment
		procedural continuous assignment
		procedural timing control statement
		conditional statement
		case statement
		loop statement
		wait statement
		disable statement
		event trigger
		seq block
		par block
		system task enable
	키워드	
	begin/end, deassign, defparam, disable, case/endcase/default, casex/endcase/default, casez/endcase/default, function/endfunction, module/endmodule, macromcdule/endmodule, task/endtask, event, for, force, forever, fork/join, if/else, initial, inout, input, output, parameter, posedge, release, repeat, wait, while	
Verilog-2001	inout, input, output	combined port and data type
	module	ANSI C style module declaration
	module, parameter	module port parameter list
	primitive /endprimitive	ANSI C style UDP declaration
	reg, integer, time	variable initial value at type declaration
	output	combined port/data type declaration and variable initialization
	task/endtask, automatic	ANSI C style task declaration and automatic task
	function /endfunction, automatic	ANSI C style function declaration and automatic function
	posedge, negedge	comma separated sensitivity lists
	always	variable vector part selects and multidimensional arrays
	real, realtime	arrays net and real data type
	inout, input, output	signed reg, net and port declaration
	attribute	attributes
	parameter	sized and typed parameter constant
	localparam	fixed local parameter

검증 항목을 선택하기 위하여 Verilog 키워드를 기준으로 선택적 트리내에서 키워드의 기본적인 사용을 확인하기 위한 테스트로 구성되어 있다. Verilog-2001 검증 항목은 Verilog-2001에서 추가, 변경된 문법에 대한 테스트를 모두 수행하였다.

보강된 Verilog-2001 테스트 슈트 및 기존의 Verilog-1995 수트의 구성은 <표 1>과 같다.

5. 결 론

본 논문에서는 Verilog-2001을 지원하는 Verilog-2001 파서를 개발하였다. Verilog-2001의 Verilog 중간형식을 XML로 정의하고 적용 사례로 코드 룰 체커의 사례를 보였다. XSL을 이용한 처리는 파싱을 포함한 대부분의 CAD 툴에서 유용하게 사용될 수 있으며 특히 시멘틱 처리가 필요한 코드 룰 체커 및 커버리지 툴 등에서 활용이 가능하다.

참 고 문 헌

[1] S. Cheng, R. Brayton, G. York, K. Yelick and A. Saldanha, "Compiling Verilog into timed finite state machines," Verilog HDL Conference, pp.32-39, 1995.

[2] C. Dawson, S. Pattanam and D. Roberts, "The Verilog Procedural Interface for the Verilog Hardware Description Language," Verilog HDL Conference, pp.17-23, 1996.

[3] J. Dimitrov, "Operational semantics for verilog," Proceedings of Eighth Asia-Pacific Software Engineering Conference, pp.161-168, 2001.

[4] B. Bailey and D. Gajski, "RTL semantics and methodology," Proceedings of the 14th International Symposium on System Synthesis, pp.69-74, 2001.

[5] The IEEE Verilog 1364-2001 Standard What's New, and Why Need It, Stuart Sutherland, 9th Annual International Conference and Exhibition, March, 2000.

[6] 1364-2001 IEEE Standard for Verilog Hardware Description Language, 2001.

[7] IEEE Std. 1364.1-2002 IEEE Standard for Verilog Register Transfer Level Synthesis, 2002.

[8] H. Eriksson and M. Penker, "UML Toolkit," John Wiley & Sons, Inc., 1998.

[9] ANTLR web site, http://www.antlr.org/.

[10] T. Parr, "Language Translation Using PCCTS & C++," Automata Publishing Company, 1997.

[11] M. Reshadi, A. Gharehbaghi and Z. Navabi, "AIRE/CE : a revision towards CAD tool integration," Proceedings of the 12th International Conference on Microelectronics, pp. 277-280, 2000.

[12] T. Karayiannis, J. Mades, T. Schneider, A. Windisch, and W. Ecker, "Using XML for representation and visualization of elaborated VHDL-AMS models," VHDL International Users Forum Fall Workshop, pp.83-87, 2000.

[13] E. A. Lee and S. Neuendorffer, "MoML A Modeling Markup Language in XML Version 0.4," Technical Memorandum ERL/UCB M 00/12, 2000.

[14] N. Kai and M. Luoming, "Design and implementation of the DTD-based XML parser," Proceedings of International Conference on Communication Technology, pp.1634-1637, 2003.

[15] J. Gi and C. Hirata, "XACDML-extensible ACD markup language," Proceedings of 36th Annual Simulation Symposium, pp.343-350, 2003.

[16] L. Lan, L. Xi, X. Yue and Z. Xuehai, "XM-ADL, an extensible markup architecture description language," Proceedings of 15th Annual IEEE International ASIC/SOC Conference, pp.63-67, 2002.

[17] EdaXML, http://www.e-tools.com/.

[18] F. Boumphrey et al., "Professional XML Applications," WROX, 1999.

[19] Icarus Verilog Test Suite, http://www.icarus.com/eda/verilog/.

[20] Matthew H. Austern, Generic Programming and the STL : Using and Extending the C++ Standard Template Library. Addison-Wesley, Reading, MA, 1999.

[21] Nicolai M. Josuttis, The C++ Standard Template Library : A Tutorial and Reference. Addison-Wesley, Boston, MA, 1999.

김 영 수

e-mail : youngsoo@etri.re.kr

1995년 인하대학교 전자공학과(학사)

1997년 인하대학교 대학원 전자공학과 (공학석사)

1997년~2001년 삼성전자 반도체 SYSTEM LSI 본부 근무

2001년~현재 한국전자통신연구원 시스템 IC 설계팀 근무

관심분야 : 설계자동화, 멀티미디어 설계

김 태 석

e-mail : kts81932@etri.re.kr

1999년 충남대학교 컴퓨터공학과(학사)

2001년 충남대학교 대학원 컴퓨터공학과 (공학석사)

2001년~현재 한국전자통신연구원 시스템 IC 설계팀 근무

관심분야 : 설계자동화, 멀티미디어 설계

김 상 필

e-mail : spkim@etri.re.kr

1988년 서강대학교 물리학과(학사)

1988년~현재 한국전자통신연구원 시스템 IC 설계팀 근무

관심분야 : 임베디드시스템, ECAD, 멀티미디어 시스템

조 한 진

e-mail : hjcho@etri.re.kr

1982년 한양대학교 전자공학과(학사)

1987년 New Jersey Institute of Technology 전기공학과(공학석사)

1992년 University of Florida 전기공학과(공학박사)

1992년~현재 한국전자통신연구원 시스템 IC 설계팀 팀장

관심분야 : SOC 설계 방법론, 무선통신, 멀티미디어 설계