IEEE Verilog 1364-2001 표준 인터페이스 라이브러리의 개발

김영수, 김상필, 조한진
한국전자통신연구원 시스템 IC 설계팀
전화 : 042-860-1645 / e-mail: youngsoo@etri.re.kr

A Procedural Interface Library for IEEE Verilog 1364-2001

Youngsoo Kim, Sangpil Kim, Hanjin Cho
System IC Design Team, Electronics and Telecommunications Research Institute

Abstract

A procedural interface libary for IEEE Verilog 1364-2001 is developed. The lexer and scanner are developed 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 in the paper. By using the XML intermediate, it allows the portable and scalable development of various kinds of applications. The XML DTD(Document Type Definition) of Verilog is defined and the corresponding XML intermediate format is developed. The paper describes example application of code rule checker which is built using the language interface library.

I. 서론

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 중간형식도 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 입력형식으로 기본적으로 module declaration, instantiation 및 symbolic latch 형태의 description으로 구성되어 있다. vl2mv는 구조적인 정보를 얻어내는 데는 사용될 수 있으나 이를 Verilog의 시맨틱을 얻거나 조작하는 등의 RT 레벨의 조작은 힘든 단점을 가지고 있다.[1]

Cadence사의 Verilog Procedural Interface는 PLI(Programming Language Interface) 표준에 근거한 것이며 C 접속 API 들을 제공한다. Verilog module에 대한 내부 데이터구조에 접근할 수 있으며 정보를 추출해 낼 수 있다. PLI는 access와 utility function으로 구성되어 있으며 사용자의 application에서 호출하여 사용할 수 있다. PLI 루틴들은 시뮬레이터와 접속하여 module의 구조정보와 event ordering 및 verilog data의 읽기와 쓰기 등을 지원한다.[2] 그러나 시뮬레이터 접속을 위한 표준으로 개발되었기 때문에 로직 시뮬레이터나 타이밍 분석기 등의 CAD 툴 이외에 시맨틱 정보들을 얻거나 조작해야 하는 코드 룰 체커나 코드 커버리지 체커의 개발에는 적합하지 않다.

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

유틸리티를 이용한 접속방법을 개발하였다. 또한 이를 이용한 Verilog 코드 룰 체커의 사례를 소개하였다.

II. Verilog-2001 표준 분석

현재의 IEEE Verilog 1364-1995 표준 이후에 2001년에 IEEE Verilog 1364-2001 표준이 제정되었다.[3]

개발된 Verilog 접속 라이브러리는 IEEE Verilog 1364-2001 표준에 근거하여 개발되었으며 IEEE Verilog 1364-1995에 대비하여 추가된 사항은 다음과 같다.

- deep-submicron 대비한 IP 모델링을 위한 기능 추가
- 시뮬레이션 및 합성 툴 벤더의 요구사항을 반영
- IEEE Verilog 1364-1995 표준에 있던 모호함과 에러를 수정

각각의 세부적인 사항은 다음과 같다.

2.1 모델링 개선 사항

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

- Design 관리를 위한 Verilog configurations 문 - 각 Verilog module의 정확한 version과 source location을 Verilog language의 일부분으로 상세화 하는 것을 허용한다. 이식성을 위해서 virtual model library는 configuration block 내에서 사용되고, virtual library를 physical location과 연결해주는 library map files를 분리하였다. module 정의의 바깥 부분에 서술되며 configuration block은 모든, 또는 특정한 module instance의 source code location을 기술한다.

- scalable 모델링을 위한 generate loop - module 및 primitive들의 multiple instances 생성하기 위하여 사용되며 이외에 variables, nets, tasks, functions, continuous assignments, initial procedures, always procedure의 multiple occurrence 생성을 위하여 사용되도록 추가되었다.

- constant function - constant function의 정의는 다른 Verilog function과 동일하다. compile 또는 elaboration time에 값이 결정되는 constructs를 사용하는 것으로 제한된다.

- indexed vector part select - variable을 사용하여 특정 byte를 선택하기 위해 사용된다. base expression은 width expression, offset direction으로 구성되며 simulation run-time 동안에 변할 수 있다. width expression은 반드시 상수이어야 한다. offset direction

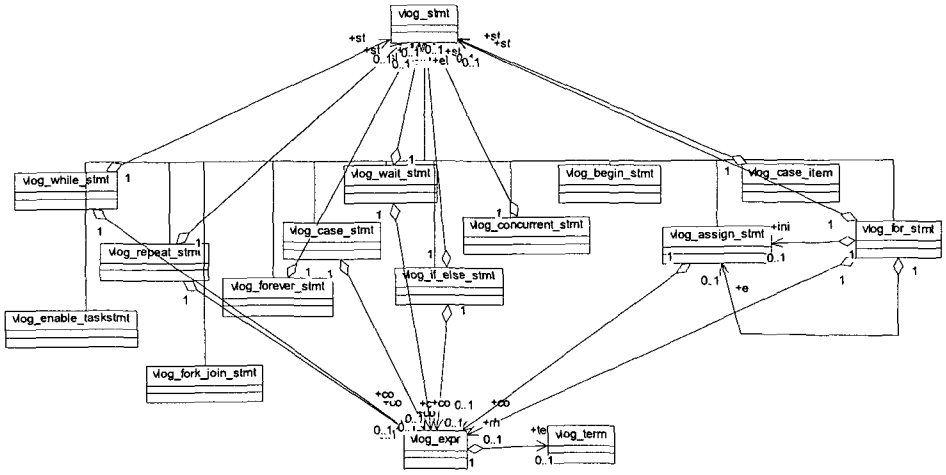

그림 1. Verilog 1364-2001의 UML 모델

은 base expression으로부터 width expression을 더할 것인지 뺄 것인지를 나타낸다.

· multidimensional arrays - 다중 array를 지원한다. variable과 net data type에 대한 array를 지원한다.

· bit and part selects within arrays - array word에 대한 bit selects와 part select를 직접 접근할 수 있도록 하였다.

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

2.2 ASIC/FPGA를 위한 정확도 개선 사항

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

III. Verilog 인터페이스 라이브러리

3.1 Verilog 인터페이스 라이브러리 클래스 다이어그램

Verilog-2001에 대한 표준을 만족하는 Verilog 접속 라이브러리를 개발하기 위하여 분석한 IEEE Verilog 1364-2001의 클래스 다이어그램의 일부가 그림 1에 표시되어 있다. 일부의 클래스 다이어그램만을 표시하였으며 분석 방법으로는 UML(Unified Modeling Language)을 사용하였다.[4] UML 모델링 기법 중 동적인 모델링과 정적인 모델링에서 정적인 모델링인 클래스 다이어그램을 추출하는 것은 중간 형식 파일을 구성할 때 요소와 속성을 정의할 때와 인터페이스 라이브러리의 클래스를 설계할 때 중요한 개발 항목이 된다.

3.2 Verilog 인터페이스 라이브러리 전체 구조

Verilog 접속 라이브러리의 전체구조는 그림 2와 같다. Verilog-2001 기술된 소스 파일을 받아서 lexical analysis와 semantic analysis를 한 후에 중간형식으로 저장한다. 중간형식은 XML로 기술되며 이에 대한 처리는 XSL을 이용하여 할 수도 있으며 미리 정의된 절차 접속 API를 이용하여 할 수도 있다

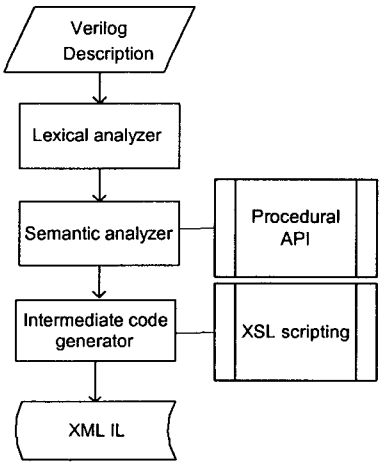

그림 2. Verilog 접속 라이브러리 전체 구조

3.3 어휘 및 구문분석기

Verilog 1364-2001 표준에 추가된 항목들을 지원하는 문법을 기술하여 ANTLR(ANother Tool for Language Recognizer)가 제공하는 컴파일러 프레임워크를 이용하여 어휘 및 구문분석기를 개발하였다.[5]

ANTLR은 C++ 또는 JAVA로 표현된 action을 포함하는 문법적 표현으로부터 recognizer 및 translator에 대한 프레임워크를 제공하는 랭귀지 툴이며 Verilog-2001 문법을 정의하여 이에 따른 lexer 및 scanner를 생성하여 사용하였다.

3.4 Verilog 중간 형식

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

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

```
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 3. XML 형식의 Verilog HDL 중간형식

3.5. XSL을 이용한 접속 API 개발

정의한 XML 중간형식을 이용하여 실제의 Verilog 접속 라이브러리를 이용한 CAD 툴 개발에 적용하였다. 적용이 가능한 분야는 Verilog 파서가 필요한 분야로서 XML의 유틸리티인 XPath와 XSL을 적용하면 Verilog 소스 파일을 처리하는 루틴을 기존의 하드 코딩에 의한 방법에 비해 손쉽게 개발할 수 있다.

XSL은 정규식에 기반한 XPath 표현식들을 사용하여 XML로 기술된 Verilog 파일에 대해서 특정 엘리먼트들과 이들의 패턴을 검색하고 처리할 수 있다.[6][7]

```
<?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 4. XSL를 이용한 접속 API 개발

그림4의 a)는 Verilog의 중간형식으로서 Verilog 소스 파일이며 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.

V. 결론

본 논문에서는 Verilog-2001을 지원하는 Verilog 인터페이스 라이브러리를 개발하였다. Verilog 중간형식을 XML로 정의하고 개발사례로 코드 룰 체커의 사례를 보였다. XSL을 이용한 처리는 파싱을 포함한 대부분의 CAD 툴에서 유용하게 사용될 수 있으며 특히 시맨틱 처리가 필요한 코드 커버리지 툴 등에서 활용이 가능하다.

참고문헌

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

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

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

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

[7] 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.