SLDS機構を用いたSystemC-Verilog HDLトランスレータの開発

佐藤 賢文† 三井 浩康†
東京電機大学理工学部‡

1. はじめに

近年、組込みシステムは適用分野の拡大、機能の多様化により、ソフトウェア規模が拡大し、搭載されるLSIは大規模化・複雑化している。この様な現状で、組込みシステムの新しい設計手法が提唱され、その一つにSystemC言語(以下、SystemC)を用いたトップダウン設計がある。

しかし、SystemCを用いて記述したハードウェア(以下、HW)は、シミュレータ上での動作シミュレーションは可能であるが、対象とするデバイスへの書き込みができないという問題がある。この問題を解決するために、企業はSystemCソースコードからHW記述言語のソースコードを生成する動作合成ツールを利用する。動作合成ツールは非常に高価であり、教育現場での利用は難しい。このため学生が実機を用いてSystemCによるシステム設計を学ぶのが困難な現状がある。

2. 研究目的

本研究では、学生が実機を用いてSystemCによる開発、設計を学習する際に必要な機能を提供するSystemC-Verilog HDLトランスレータの開発を目的とする。

3. 本研究で使用する技術

3.1 SystemC言語[1]

システム・レベルの設計や検証に扱う言語をシステム・レベル設計言語という。その一つがSystemCであり、SystemCの普及推進・標準化団体であるOpen SysmteC Initiative(OSCI)によって、無償配布されている。

SystemCはC++のクラスライブラリの一つとして定義されているため、文法はC++に依存する。

SystemCはシステムの設計段階に応じて、異なる抽象度のモデルでシステムを記述することができる。本稿ではRTLモデルの設計を対象としている。RTLモデルは設計するシステムのHWモジュールに対して、モジュール内で記述されたアルゴリズム、モジュール同士のインターフェース間の通信など、全ての動作をシステムクロック同期として記述するモデルである。

Development of SystemC-Verilog HDL Traslator Using SLDS Mechanism

†Masafumi Sato †Hiroyasu Mitsui
‡ School of Science and Engineering,Tokyo Denki University

3.2 トランスレータ[2]

トランスレータは言語処理系の一つである。入力と出力が同抽象度であることが特徴である。

本研究ではSystemC(RTLモデル記述)を入力として、等価な回路動作記述を行っているVerilog HDL(RTL記述)を出力とするトランスレータを作成する。

トランスレータは機能ごとに分割して、フロントエンド、ミドルエンド、バックエンドに分けられる。

(1) フロントエンドでは入力されたソースコードの言語使用に応じて字句解析、構文解析を行う。
(2) ミドルエンドでは構文解析の結果を元に中間コードを生成と最適化を行う。
(3) バックエンドは出力コードを生成する。

3.3 Simple Logic Design System(SLDS)

SLDSは、Design Methodology Lab提供のディジタル回路設計システムである。

SLDSはブロックダイアグラムと呼ばれるグラフィカルな回路入力と、Logic Description Language(以下、LDF)と呼ばれる独自のHW記述言語を入力として、Verilog HDLを出力する機構を持つ。LDFを元にVerilog HDLを出力する機構はトランスレータとして見なせる。本研究ではSLDSのミドルエンド、バックエンドを利用する。

4. 研究内容

4.1 SystemC-Verilog HDLトランスレータ

図1に開発するトランスレータの内部構成図を示す。

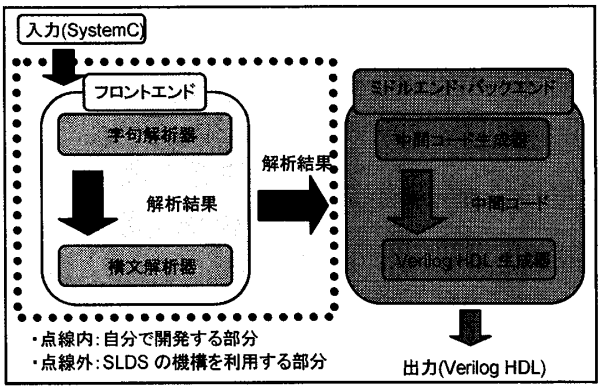

図1 開発するトランスレータの内部構成図

SystemCのRTLモデル記述を入力とし、Verilog HDLによるRTL記述を出力とするトランスレータを開発する。入力するSystemCソースコードにはヘッダファイルとインプリメンテーションファイルの二種を用いる。

ヘッダファイルにはモジュールのポートや、モジュール間通信のためのチャネル、モジュールのプロセス化のための宣言が行われている。インプリメンテーションファイルではモジュールの機能について記述されている。SystemC 記述の半加算器と、トランスレータの出力である Verilog HDL 記述の半加算器を図 2 に示す。

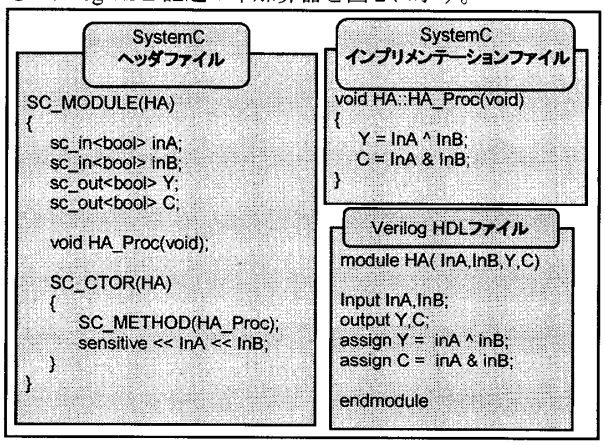

図 2 SystemC と Verilog HDL による半加算器の記述

ミドルエンド部分とバックエンド部分は、 SLDS の機構を利用し、実際にはフロントエンドと、ミドルエンド部分とのインターフェースの設計、開発を行う。

SLDS をミドルエンド、バックエンド部分に利用する理由は、商用に開発されたツールの機構を利用することで、良質な Verilog HDL ソースコードを出力できることにある。 実装する機能は以下に示す回路の SystemC 記述を Verilog HDL 記述に変換可能とする。

- 組み合わせ回路
- 順序回路
- 単一モジュール構成システム
- 複数プロセスシステム
- 階層構成システム

SystemC の初心者向け教本[2]に載っている内容を検討して、上記の回路、システム記述を扱えれば、学生が用いるツールとして十分な機能を持っていると考えた。

4.2 フロントエンドの開発

字句解析器の開発には flex、構文解析器の開発には bison を用いた。flex、bison は Linux 系システムに標準で搭載されているコンパイラ・コンパイラである。

字句解析器は入力であるソースコードを単語毎に区切り、構文解析、意味解析に必要なトークン番号、トークン種別を割り振り構文解析器に渡す役割を持っている。

構文解析器は字句解析器の解析結果を受けて、ソースコードが構文規則に則って記述されているかチェックを行い、プログラム全体の構成を構文木の形で表す。

字句解析器実装において、SystemC 記述と C 言語記述の間で、同じ記述であるが持つ意味が違う単語が発生した。この問題を解決するため、flex のスタート状態を利用した。スタート状態を利用すると、一つの字句解析器で複数の字句解析器の機能を持つことができる。この機能を利用して、SystemC 記述と C 言語記述の両方を一つの字句解析器で扱うことができた。

構文解析器の実装において SystemC 独自の記述方式と C 言語記述の両方を解析できる必要があったため、C 言語記述の構文規則は ANSI 規格を参考に作成を行い、SystemC 記述の構文規則は独自に生成した。

開発したフロントエンド内の構成を、主要モジュールを用いて表した概形を図 3 に示す。

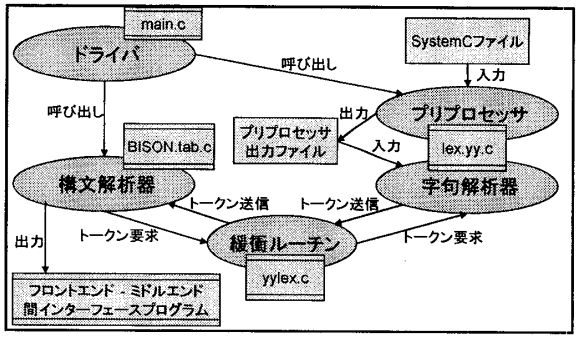

図 3 フロントエンド内主要モジュール構成図

lex.yy.c は flex によってコンパイルされたプログラムであり、字句解析器として動作する。また、スタート状態を利用してプリプロセッサの機能の一部としても動作する。Bison.tab.c は bison によってコンパイルされたプログラムであり、構文解析器として動作する。main.c はドライバとして動作する。yylex.c は字句解析器、構文解析器間でトークンに対する処理を行うプログラムである。

4.3 評価

4.1 で述べたトランスレータの入力対象とする SystemC 記述の各回路、各システムに対して字句解析、構文解析が可能か検証を行った。

検証には bison のアクション(構文規則にマッチした際に動作する C 言語命令)を用いた。各回路、システムの SystemC ソースコードを入力し、端末上に構文解析の進捗を表示させ、正常に動作していることを確認した。

5. まとめと今後の予定

SLDS 機構を用いた SystemC – Verilog HDL トランスレータフロントエンドを開発した。今後、SLDS ミドルエンドとのインターフェース部の開発を行い、意図した Verilog HDL が生成されるか検証を行う。

参考文献

（1） 並木秀明、 後簡哲也、 片岡忠士：“SystemC による System デザイン入門”、 技術評論社、 Vol2005、 pp12-21、pp78-338 2005
（2） 宮本 衛市： “はじめてのコンパイラ 原理と実践”、 Vol2007、pp25-94、 2007