JavaRock-Thrash の実装

小 池 恵 介[†] 榎 戸 健 二[†] 船 田 悟 史[††]
三 好 健 文[††] 中 條 拓 伯[†]

JavaRock は Java 言語のソースプログラムから VHDL コードを出力する高位合成ツールである．本研究では，JavaRock をもとにして，Java から Verilog HDL を出力する高位合成ツール「JavaRock-Thrash」を提案する．本ポスター発表では JavaRock-Thrash の設計方針を示すとともに，コード変換規約について説明し，生成されるモジュールの入出力ポートなどの例を示す．

An Implementation of JavaRock-Thrash

Keisuke Koike,[†] Kenji Enokido,[†] Satoshi Hunada,[††]
Takefumi Miyoshi[††] and Hironori Nakajo [†]

JavaRock is a high-level synthesis tool generating a VHDL code from a Java source file. In this study, we propose JavaRock-Thrash generating a Verilog HDL code from Java based on JavaRock. We describe our design policy of JavaRock-Thrash and explain code conversion rules as well as examples of I/O ports of the generated module.

1. はじめに

近年，半導体技術の進歩によって LSI が高集積化し，回路設計が複雑化してきている．現在 LSI は VHDL や Verilog HDL を用いた Register Transfer Level(RTL) の設計が主流であるが，回路規模の増加に伴い，より効率的な設計手法が求められている．そこで，RTL よりも抽象度の高い設計が可能な高位合成技術が注目されている．高位合成ツールは商用，研究用を含め多数存在するが[1)2)]，本研究では JavaRock について取り上げ，その拡張を行う．JavaRock は，イーツリーズ・ジャパンの三好健文氏が作成されたもので，Java を入力として VHDL を出力する高位合成ツールである．本研究では，JavaRock をもとにして，Java から Verilog HDL を出力する高位合成ツール JavaRock-Thrash を提案する．

2. JavaRock-Thrash の設計方針

JavaRock-Thrash は，ソフトウェアとして実行可能な Java ソースコードから Verilog HDL を生成するという設計方針を持つ．即ち，ハードウェア化のための特別な型の導入や構文拡張を行わず，一般的な開発環境と実行環境を用いて開発が行えるようにする．この設計方針は，JavaRock から受け継いだものである．

しかし，JavaRock-Thrash は JavaRock と全く同等の回路を出力することを目的とするのではない．JavaRock-Thrash は JavaRock よりも高速に動作する回路を作成することを目的とし，Data Flow Graph（DFG）を用いた最適化や浮動小数点演算のサポートなど，現段階で JavaRock において未実装であることもサポートする予定である．したがって，JavaRock-Thrash と JavaRock の出力する回路は，インタフェースや処理速度が異なる場合がある．

3. Java から Verilog HDL へのコード変換例

本章では，Java から Verilog HDL へのコード変換例として，クラス，メソッド，変数宣言の変換例について示す．

3.1 クラス

Java のクラスは， Verilog HDL のモジュールに変換される．モジュールは，同期式順序回路であり，クロック（clk）とリセット信号（reset）を持つ．図 1 にクラスから作成されるモジュールの例を示す．

3.2 メソッド

JavaRock Verilog では，各メソッド毎に処理の開始

† 東京農工大学
Tokyo University of Agriculture and Technology
†† 株式会社イーツリーズ・ジャパン
e-trees.Japan.Inc

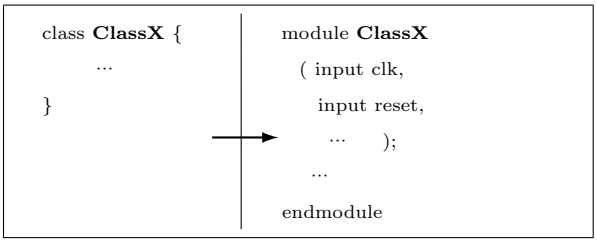

図 1 クラス → モジュール変換例
Fig. 1 An example of converting class into module

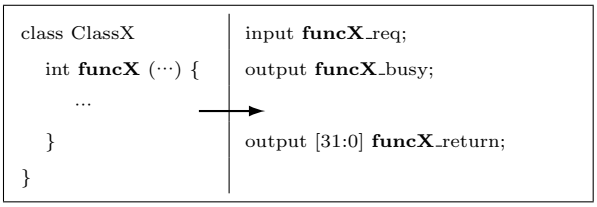

図 2 メソッド → ポート変換例
Fig. 2 An example of converting method into ports

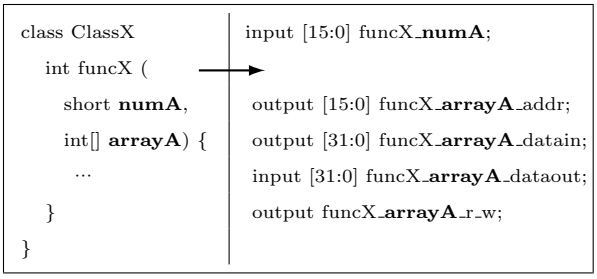

図 3 パラメータリスト → ポート変換例
Fig. 3 An example of converting parameter list into ports

を命令する入力ポート（メソッド名_req）と，メソッドが処理中かどうかを示す出力ポート（メソッド名_busy）が存在する．また，メソッドの戻り値がプリミティブ型の場合，戻り値の出力ポート（メソッド名_return）が存在する．図 2 にメソッドから作成されるポート例を示す．

3.3 パラメータリストの変数

パラメータリストに変数を宣言すると，それぞれの変数に対応した入出力ポートが作成される．変数がプリミティブ型の場合，入力ポート（メソッド名_変数名）が宣言され，メソッド開始時にこのポートの値がモジュール内部のレジスタに格納される．一方，変数が配列型の場合，モジュール外部にある Block RAM 制御用の 4 つの入出力信号（メソッド名_変数名_addr，メソッド名_変数名_datain，メソッド名_変数名_dataout，メソッド名_変数名_r_w）が宣言される．図 3 にパラメータリストから作成されるポート例を示す．

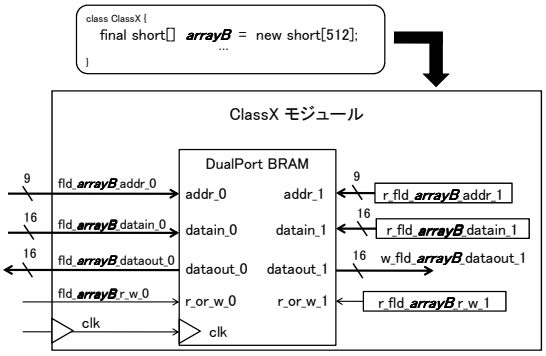

図 4 フィールドの配列 →BlockRAM 例
Fig. 4 An example of converting field declared array into BlockRAM

3.4 フィールドまたはメソッド内部に宣言したプリミティブ型変数

フィールドまたはメソッドにプリミティブ型の変数を宣言すると，それぞれの変数に対応したレジスタが，モジュール内部に作成される．

3.5 フィールドに宣言した配列型変数

フィールドに final 指定した配列変数を宣言すると，図 4 のようにモジュール内部にデュアルポート Block RAM が作成される．この Block RAM の片方のポートは，モジュールの入出力ポートにつながっており，モジュール外部からデータの入出力が可能となっている．もう片方のポートには，モジュール内部でデータを入出力するためのレジスタやワイヤが接続される．fld から始まる名前の信号は，モジュールの入出力ポートである．r_fld，w_fld から始まる名前の信号は，モジュール内部で宣言した Block RAM の入出力用レジスタ及びワイヤである

謝辞 本研究の一部は，独立行政法人日本学術振興会とフィンランドアカデミーとの二国間交流事業（共同研究）による支援を得た．

参 考 文 献

1) P. Bellows and B. Hutchings. Jhdl - an hdl for reconfigurable systems. In Proceedings of the IEEE Symposium on FPGAs for Custom Computing Machines, FCCM '98, pp.175-184, Washington, DC, USA, 1998. IEEE Computer Society.
2) Gokhale, M.; Stone, J.; Arnold, J.; Kalinowski, M.; , "Stream-oriented FPGA computing in the Streams-C high level language" Field-Programmable Custom Computing Machines, 2000 IEEE Symposium on , pp.49-56