社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ARC－176 (2)
2008－EMB－ 7 (2)
2008／1／15

Verilog HDL スタイルガイドの利用方法

長谷川誠(大同工大), 渡会勝彦(富士通 VLSI),
米永裕司(中央製作所), 渡部謹二, 小川清(名古屋市工業研究所)

抄録:Verilog HDL の教育を Verilog HDL スタイルガイドを用いて行った。スタイルガイドの構造と内容を確認し、どの水準で何を教えるとよいかを検討した。

Trials on training engineers for Verilog HDL using the HDL style guide

HASEGAWA Makoto(Daido IT), WATARAI Katsuhiko(Fujitsu VLSI),
YONENGA Yuji(Chuo seisakusho), WATABE Kinji, OGAWA Kiyoshi(NMIRI)

Abstract: Engineers are trained Verilog HDL using the HDL style guide on Verilog HDL. We make an order of the rules on the phase of training and restructuring the rules.

1. はじめに

名古屋市工業研究所では, 組込みソフト技術者の育成のため, 自動車業界向けにTOPPERS/ JSP, /OSEK を使い MISRA-C を適用したプログラミングの教育を行ってきた[1][2][9][10]。近年, 開発の期間短縮, 試作内容が膨大になるなど, 開発の過程で FPGA を使うことが多くなってきた。そこで FPGA の開発を, ソフトウェアエンジニアが行う際に C 言語に類似した Verilog HDL(以下 Verilog と略)を用いることがあるため、FPGA を用いた Verilog 研修を企画した。対象は地元の中小企業であるが、大学、大企業の C 言語を習得している技術者とも幅広く意見交換を行ってきた。また, 名古屋市工業研究所では, 大学から卒業研究のために研修生を受け入れており, 企業向け研修で利用する教材の検討を担当している。

2. FPGA 研修

回路設計をしたことのないエンジニアが FPGA のプログラムを担当する場合に, 回路の知識が必要になる。基本的には, デジタルフリップフロップ回路の挙動が理解できる人と理解できない人を申し出てもらい, 後者の方にはデジタルフリップフロップの理解をまずお願いした。

初年度は, C 言語がわかるソフトウェア技術者が FPGA のプログラムを開発する場合に, 厳密で検証しやすいコードを書くことができるようするにはなにをしたらいいかを模索した。2006 年 11 月の産業技術連絡会議組込みシステム研究会において研修の現状を報告したところ[8], 熊本大学末吉敏則教授より, STARC が作成した RTL スタイルガイド[3]を利用するように, ご指導を受けた。

2.1 Style Guide 勉強会

第1回の FPGA 研修受講者を中心に, RTL スタイルガイドの勉強会を 2007 年 3 月から始めた。RTL 設計スタイルガイド Verilog-HDL 編をもとに, 各ルールの説明を輪講形式で行った。月 1 回, 1 章ずつ, 6 月までの 4 ヶ月に実施した。第 2 版を前提として行ったが, 第 1 版, 英語版[3]を持っている参加者があったため, 版による違いを確認したり, 日本語で読んでわからない内容を, 英語の表現から理解するようにし、規則の日英対比表を作成した。また, 規則の対応関係の本文中の記載を利用し, 対比表に関連規則を記載した。検討の成果を 2007 年の研修で紹介し, 再分類, 体系化を行うことにより, より実用的なコーディングに利用できるように検討している。

2.2 FPGA の課題学習

研修では, 講義のほかに, 各自で課題を用意し, それを実装し, 最終日に報告をしてもらうようにしている。

特定の開発案件を抱えていない方には, 例えば, 電卓を FPGA 上で構築することを例示している。簡単なゲート回路や, フリップフロップを Verilog で記述し, まずは Verilog に慣れていく。次に, 四則演算のゲート回路の知識について学習する。最後に, 乗数や対数などの複雑な機能を盛り込んでいくようにしている。また, 研修での実習手順に沿って, FPGA による Verilog 開発の学習の問題点を ML に報告し、整理するようにしている。

2.3 研修の実習手順

2.3.1 ボードの準備

FPGA 上でのソフトウェアの構築に用いた対象となるボードは Xilinx の FGPA を搭載した AVNET DESIGN SERVICES の ADS-XLX-SP3E-EVL100 を使用した。ボードの他に, USB ケーブルを用意した。PC との USB ケーブルを容易接続においては, ドライバの誤認識があり, 回復手順について, 初年度は手間取った部分があった。

2.3.2 開発環境の準備

Xilinx の開発ツールである ISE 9.1i 及び Model Sim XE III 6.2c の 2 つのソフトを Windows 搭載 PC にインストールした。研修では無償の評価版を利用したが, ライセンス登録は必要であり, 研修生について個々に行った。

ソフトウェアの Update のダウロードにあたっては, 研修室のゲートウェイの設定から大規模なものがダウンロードできなかったことがあった。

2.3.3 研修資料と ML での議論

研修資料は研修用のサーバから FTP でダウンロードできるようにしており, 質疑応答はメーリングリストを運営して対応するとともに, 記録に残るようにした。また, シミュレーションの結果などは再現しにくいことがあるため, 画面を alt+screen で取得して, 画像ソフトでファイルにして報告してもらうようにお願いした。ただし, メールに添付するとファイルが大きい場合には, 白黒で判定できるものは白黒にすることにした。

Verilog で FPGA のプログラムを記述する際に課題となるのは次の事項があった。

・四則演算のゲート回路の知識の理解が十分でないと, 特に割り算におけるゼロ割などの処理をどのように実現するとよいか。

・Verilog の単純な回路を構築する場合でも, 回路の基礎知識がないと C 言語のような動作をしない理由が分からない。

・開発環境ソフトウェアの機能が多く, 作業手順に前後があると, 動作しない。

・コンパイルエラー, 予期せぬエラーが発生すると, WEB で検索して対応するが, WEB でも検索して出てこないと対応が困難。

・Verilog で書いても, タイミング(遅延)がどうなるかわからない。

・テストベンチのシミュレーションではうまくいっていても, 実装してから不具合が出ることがある。

・Verilog で書いたファイルを論理合成すると, できているはずの回路ができていないことがある。

これらは, Style Guide で指摘している事項を含むため, Style Guide の指摘事項を実習中のプログラムとの対応関係を作成することにした。

3 コーディングスタイルガイドの利用

3.1 スタイルガイドの概要

基本設計制約, RTL 記述技法, RTL 設計手法,

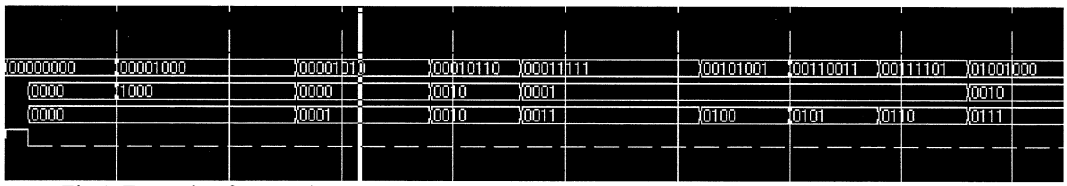

Fig 1. Example of report data

検証技法からなる。ただし, 基本設計制約には, 命名規則のような静的な事項と, 同期設計,初期リセット, クロック, 非同期設計のような動的な要素と, 階層設計, FPGA のような高度なものまで含んでいる。まず, 学習の進度に応じて参照するガイドを例示するために<番号>を付与した。

List 1. Classification on training

<1>初学者に, 初日に説明する。
<2>実習内容に含めて対応する。
<3>進んでいる人のみ対応する。
<4>研修外の課題として読んでもらう。

また, 基本設計制約を3つに分類して, 章構成

2.3.4 実習内容

Xilinx の開発ツールである ISE 9.1i で Verilog を記述し, コンパイル, テストベンチファイルの作成, 論理合成し, Model Sim XE III 6.2c でそのテストベンチファイルで確認し、FPGA ボードにダウンロードし動作が同じであるか確認した。必要に応じて、デジタルオシロスコープを用い、信号の確認を行った。

開発事例を準備していない方には, 電卓の機能である和, 差, 積, 商のそれぞれの回路を作り, 各モジュールで動作確認をした後 1 つにまとめるようにした。

2.4 Verilog 記述上の課題

に構造を持たせた。() 書きは元の章構成であり，区別をつけるために，ABCDEF を新たな章の記号とした。第1章は List 2 の通り。

List 2. Restructuring and evaluation of Section 1
A　基本設計制約
A.1 命名規則(静的規則)(1.1)<1>
A.2 動的規則<2>
 A.2.1 同期設計(1.2)
 A.2.2 初期リセット(1.3)
 A.2.3 クロック(1.4)
 A.2.4 非同期設計(1.5)
A.3 より複雑な事象<4>
 A.3.1 階層設計(1.6)
 A.3.2 FPGA(1.7)」

最初は命名規則のみを勉強し，テキストにあるような記述をしてもらう。

次に，クロックの影響のある回路になった時点で，1.2 を勉強してもらい，1.3 は自習とした。

第 2 章は，always 文，制御構造とそれ以外に分類した。第 3 章，第 4 章の構造はほとんどそのままである。

List 3. Restructuring and evaluation of section 2,3,4
B RTL 記述技法(2)
 B.1 Always 文<2>
  B.1.1 組み合わせ回路(2.1)
  B.1.2 組み合わせ回路の always 文記述(2.2)
  B.1.3 FF の推定(2.3)
  B.1.4 ラッチ記述(2.4)
  B.1.5 回路構造を意識した always 文記述(2.5)
 B.2 制御文<3>
  B.2.1 if 文記述(2.6)
  B.2.2 case 文記述(2.7)
  B.2.3 for 文記述(2.8)
  B.2.4 ステートマシン記述(2.9)
 B.3 演算子と代入文(2.10)<2>」
C　RTL 設計手法(3)
 C.1 機能ライブラリ<3>
  C.1.1 機能ライブラリの作成(3.1)
  C.1.2 機能ライブラリの使用(3.2)
 C.2 テスト容易化設計(DFT)(3.3)<4>
 C.3 低消費電力設計(3.4)<3>
 C.4 ソースコード，設計データの管理(3.5)<1>
D 検証技術(4)
 D.1 テストベンチ
  D.1.1 テストベンチ記述(4.1)<2>
  D.1.2 タスク記述(4.2)<3>
 D.2 検証の進め方(4.3)<3>
 D.3 ゲートアレイシミュレーション(4.4)<4>
 D.4 スタティックタイミング解析(4.5)<3>」

3.2 開発作業場の課題とスタイルガイド

開発上の課題とガイドとの関係を明確にすることにより，ガイドに従って書いていれば遭遇しない事象を避けるようにする。しかし，いきなりすべての規則を見ても，すべてを理解してからでないと書けないと思うと，内容の理解も進まないことがあった。そのため，初学者には，最低限のルールだけを教え，その後，進んでいくにしたがって，そこで関係するルールを追加していくという段階的な理解を進める方法をとった。

各ガイドの規則には，相互の関係性があり，一体的に理解する必要がある。そのため，本文中に関連性の記載のあるものを表にし，相互の規則の関係を検討した。

3.2.1 ビット幅

電卓を作る場合に，演算の下位ビットだけを取り出す処理を記述した。その際，

「B.3(2.10).3 右辺と左辺のビット幅を揃える」

を適用していないために処理が期待戸違っていたことがあった。この規則は，「<2>実習内容に含めて説明する」こととしていたが，実際に研修生が該当する記述をいつするかまで検討していなかったため，失敗例に出くわすことになった。この規則の細則である，

「B.3(2.3.10).3.3 代入する値のビット幅は，代入される信号のビット幅を超えない

B.3(2.3.10).3.4 代入する値のビット幅は，代入される信号のビット幅より小さくしない。

B.3(2.3.10).3.5 定数は('d, 'b, 'h, 'o)を明確に指定し，何進数で記述したかを意識する。」

を適用し，記述を見直した。

ビット幅より大きい場合には，ビット指定で揃えることを例示している。そこで，規則を次ぎのように補強することを考えた。

「B.3.3.3 代入する値のビット幅は，代入される信号のビット幅を超えないようにするために，assign 文でビット幅指定をする。」

ビット幅より小さい場合には，解説には上位ビットに 0 を挿入させるために連結{}を仕様すると書いている。そこで，規則を次のように記述を追加することとした。

「B.3.3.4 代入する値のビット幅は，代入される信号のビット幅より小さくしないために，0 を挿入する

部分を明記するため連結{}を用いる。」

対応方法まで明示したルールにすることと，スタイルガイドの解説を読めば，対応方法がわかるものの，対応方法が例示であり，規則にする抽象度になっていない場合がある。そのため，初学者に診察な規則記述とし，その規則にすると誤解を与えるかどうかを検討することとした。

3.2.2 case 文

「B.2.3.5(2.8.5) case 文の選択式と項目

B.2.3.5.3(2.8.5.3) case 文の項目には変数(あるいは式:a+b)を記述しない」

Case 文に変数，式を記述して，コンパイル，シミュレーションがエラーなしで動作してしまうことがあった。しかし，処理結果が想定したものと異なるため，ソースコードを見てみると，case 文の項目に式がかかれていた。この規則は，当初「<3>進んでいる人のみ対応する。」という想定であった。しかし，C言語がわかるソフトウェア技術者は，コンパイラが通るのであれば，細かい制御を書きたくなるようである。

そこで，if 文，case 文，for 文記述の規則は，「<2>実習内容に含めて説明する」とし，同じ「B.2 制御文<3>」のステートマシン記述のみ，「<3>進んでいる人のみ対応する。」にすることにした。Case 文の事例は if 文に変更することによって解決した。Case 文の項目に変数または式を書くのは，プログラマの意図が何かによって，記述するとよい方法が違うため，一意に if 文に直すのがよいとは限らないため，規則記述の追加は行わないことにした。

このように，当初の想定と実際の演習で書くコードは，その技術者が C 言語をどの程度使いこなしているか，MISRA-C のような静的なプログラムを書くのに慣れているか，電子回路を頭に思い浮かべることができるか，規則を予習するかなどの行動の違いによっても失敗の経験が違うことが見受けられた。研修の間に，できるだけ多くの不具合を体験し，規則がどのような場合に対応しているものであるかを体験してもらうことは，研修の意義として重要であることを再確認した。

4 成果と今後の課題

2007 年の研修では，研修後 3 ヶ月ほどで，タブレットの入力を処理するシステムを試作して紹介に来た受講生があった。組込みシステムでは，研修受講後1年以内に試作を行った事例を開発事例集[5]で紹介しているが，研修開催から 2 年目で，初学者による試作の事例が 3 ヶ月と，これほど早くあったのは始めてである。

Style Guide には，動く形での具体例がないため，今回の作業は，具体例を作成する一部とすることができる。すべての Guide について動く形での具体例を作成することが今後の課題である。

将来実施する可能性のある VHDL を利用した研修または受講生による VHDL への質問，評価依頼に対応するため，VHDL のスタイルガイドとの比較を行っている。VHDL スタイルガイド[7]は，改訂が行われていないという点において，Verilog との違いがあるほかは，基本構造が同一で，言語による違う規則は，それぞれ Verilog only, VHDL only との記載があるため，比較は容易である。Verilog と VHDL の違いを理解することにより，言語とその処理系が担当する作業と，開発者が行う作業との境界線を明確にすることができつつある。

参考文献

[1] 組込み開発者におくる MISRA-C:2004—C 言語利用の高信頼化ガイド，SESSAME/MISRA-C 研究会，日本規格協会, 2006

[2] 組込み開発者におくる MISRA-C—組込みプログラミングの高信頼化ガイド，SESSAME/MISRA-C 研究会，日本規格協会, 2004

[3] Design Style Guide 2001 Verilog-HDL, STARC, hd-lab, 2001

[4] RTL 設計スタイルガイド，Verilog HDL 編 第 2 版，半導体理工学研究センター, 2006

[5] 組込みシステム開発事例集，産業技術連携推進会議情報電子部会組込み技術研究会，小川清，渡部謹二，斉藤直希ほか, 2006 年

[6] Verilog プログラムの課題，長谷川誠,渡部謹二，小川清，電気関係学会東海支部, 2007

[7] RTL 設計スタイルガイド，VHDL 編，半導体理工学研究センター, 2001

[8] Verilog 研修について，小川清,斉藤尚希,渡部謹二，産業技術推進連絡会議組込み技術研究会, 2006

[9] TOPPERS/JSP の M16C/M32C での移植，斉藤直希,渡部謹二,小川 清,大澤史郁,大槻直哉，情報処理学会研究報告「組込みシステム」No.2006-EMB-001

[10] 類似の CPU への組込み OS の移植のための訓練内容と構成管理表，斉藤直希,渡部謹二,小川清,大澤史郁,大槻直哉,片岡歩,服部博行，高速信号処理応用技術学会誌, 10 巻 1 号, 2007