STARC RTL設計スタイルガイドを「こう使おう」 Verilog-HDL版

名古屋市工業研究所

小川清

概要

- 背景
- 1. 振る舞い言語の標準化
- 2. ＳＴＡＲＣ RTL設計スタイルガイド
- 3. 教育のための準備
- 4. 業務利用のための準備
- 5. 逸脱の手続きと事例
- まとめ　と　付録

事例1

- 2.10.6. 算術演算の注意点
  - ⑤ 大規模乗算器は、RTL記述で推定せず、乗算器の中身を論理式で記述する: 推奨1
    - 算術演算は1行に2つ以上記述しない方がよいでしょう。
- 3.2.5. 算術演算の機能ライブラリはデータ・パス設計の性能を向上する
  - ④ 乗算器、複合演算がある場合は、専用ツールを用いる方法がある: 参考

事例4

- 1.7. FPGA① 1 つのFPGA 内でゲーティッドクロックを使用しな複数のクロックを1つのFPGA で実現したい場合は、「1.5.2 ①非同期間インターフェースに反転ラッチを挿入する」で紹介されているゲーティッドクロック間に反転ラッチを挿入する方法もあります。
- 3.1.2. 再利用を考慮した記述スタイルを行なう
  - ③ 単純なクロック系統にする推奨3
- 3.3.1. DFT 向きのクロック、リセットを考える
- ① クロックは、LSI 外部入力端子から直接制御可能にする: 必須
- ② 2 系統のクロックを切り替えて使用している場合、テスト時はどちらか片方のクロックに固定する: 必須

事例5

- 電源をONしてもFPGAが正常に動かない
- 周辺でバイスへのリセット回路を使って制御している
- コンフィギュレーション完了ピンを見ると正常に終わっている
- 電源起動時のマイコンからのFPGA設定シーケンス，設定値を確認したが問題なし
- 動かない状態で再度マイコンからFPGA設定を行うと正しい動きになる
  - 電源起動時のみになる

背景

- MISRA-Cでソフトウェアの空間的な部分集合化は成功してきたが時間の指針が弱い。
- HDL言語の標準化をIEEEで決めた。標準に適合する書き方だけでは適切な動作をしない記述が可能である。
- 標準に適合する書き方だけでは，言語処理系，物理的な回路によって，同じ振る舞いをしない。
- 可読性，保守性の高い設計の振る舞い記述を実現したい。
- ASIC設計とFPGA設計の違いを認識したい。

名古屋市工業研究所FPGA研修

- Verilog -HDL利用して6日間（9時半から4時半）
- Xilinxのチップで，ISEを利用
- 講師：
  - 渡部謹二
  - 小川清（RTL設計スタイルガイド担当）
  - 松浦商事　松浦光洋（CPU利用，試作指導、試作デモ）

岐阜大学公開講座

- 経緯
  - 平成20年　経済産業省組込中核人材プロジェクト
- 内容
  - C言語による音声処理
  - VHDLによる画像処理
    - Xilinx FPGA
- 今年度は無料で実施（文部科学省の事業）

SESSAME

組込みソフトウェア管理者・技術者育成研究会　http://www.sessame.jp/

SESSAME Working Group 3

MISRA-C研究会

http://www.sessame.jp/workinggroup/WorkingGroup3/index.htm

MISRA-C研究会の成果物

MISRA-C研究会の様子

これまでのMISRA-C研究会の成果を纏めた、MISRA-Cの解説書（ISBN:4542503348 ￥3,570 税込）を2004年5月13日に日本規格協会から発刊しました。詳しい情報はhttp://www.sessame.jp/workinggroup/WorkingGroup3/MISRA-C_GuigeBook.htmをご覧ください。 MISRA-C研究会では、本ガイドブックをもとに引き続きMISRA-Cの実用的な運用について検討していきます。

MISRA-C研究会の活動についてのお問い合わせはSESSAME事務局までご連絡ください。

MISRA-Cと RTL設計スタイルガイドの違い

- MISRA-C:
  - プログラミング言語,
  - 空間領域の部分集合
  - 逸脱の手続きを含めて文書化
  - 複数の道具／方法を使うことを推奨
- RTL設計スタイルガイド:
  - HDL,
  - タイミングの部分集合
  - 逸脱の手続きについて不明確
  - 道具の選択の指針が不明確

ASICとFPGAの違い

- FPGA(flexible programmable gate array)は入出力、RAMなどの機能ブロックが事前に決まっている。
  - 構成要素として、LUT(look up table)、FF(flip flop)などの配置
  - クロック配線
  - 場合によってはCPU, DSP, ネットワークなどがハードウェアまたはソフトウェアで導入
- FPGAではFFの配置を前提にした論理記述かどうか
- FPGAで4入力LUTと6入力LUTの製品がある

1. 振る舞い言語の標準化

- 設計自動化(Design automation)/振る舞い言語(Behavioural languages)
  - IEEEで設計自動化として標準化、順次IECで振る舞い言語国際規格化
  - VHDL, Verilog-HDL
- RTL設計スタイルガイド
  - ㈱半導体理工学研究センター(STARC:Semiconductor Technology Academic Research Center)
- FPGAベンダによるガイド
  - Xilinx合成シミュレーション設計ガイド
  - Altera ハンドブック第1巻推奨HDLコーディング構文など

HDL(hardware Description Language)

- 設計自動化(Design automation)/HDLハードウェア記述言語: IEEE
  - VHDL:VHSIC Hardware Description Language
    - VHSIC: Very High Speed Integrated Circuits
  - Verilog-HDL
- 振る舞い言語(Behavioural languages):IEC
  - IEEEが決めたものを順次国際規格に

IEC/IEEE Behavioural languages

- IEC/IEEE 61691-1-1, Part 1: VHDL language reference manual
- IEC/IEEE 61691-2, Part 2: VHDL multilogic system for model interoperability
- IEC/IEEE 61691-3-2, Part 3-2: Mathematical operation in VHDL
- IEC/IEEE 61691-3-3, Part 3-3: Synthesis in VHDL)
- IEC/IEEE 61691-4, Part 4: Verilog® hardware description language
- IEC/IEEE 61691-6, Part 6: VHDL Analog and Mixed-Signal Extensions
- IEC/IEEE 61961-7, Part7: System C Language Reference Manual
- IEC 62530/IEEE Std 1800-2005: Standard for SystemVerilog - Unified Hardware Design, Specification, and Verification Language

IEEE DASC (The Design Automation Standards Committee)

- http://www.dasc.org/new_working-groups.html
- VHDL Working Groups
- P1076 Standard VHDL Language Reference Manual (VASG)
- P1076.1 Standard VHDL Analog and Mixed-Signal Extensions (VHDL-AMS)
- P1076.1.1 Standard VHDL Analog and Mixed-Signal Extensions - Packages for Multiple Energy Domain Support (StdPkgs)
- P1076.4 Standard VITAL ASIC (Application Specific Integrated Circuit) Modeling Specification (VITAL)

- P1800 SystemVerilog: Unified Hardware Design, Specification and Verification Language
- P1850 Standard for PSL: Property Specification Language

(cont.)

- P1481 Standard for Integrated Circuit (IC) Open Library Architecture (OLA)
- P1647 Standard for the Functional Verification Language 'e' (eWG)
- P1666 Standard System C Language Reference Manual (system c)
- P1685 SPIRIT XML Standard for IP Description
- P1699 Rosetta System Level Design Language Standard
- P1735 Recommended Practice for Encryption and [Use Rights] Management of Electronic Design Intellectual Property (IP)
- P1778 ESTEREL v7 Language Standardization
- P1801 Standard for the Design & Verification of Low Power ICs

Links to web pages of inactive Working Groups

- P1076.2 IEEE Standard VHDL Mathematical Packages (math)
- P1076.3 Standard VHDL Synthesis Packages (vhdlsynth)
- P1164 Standard Multivalue Logic System for VHDL Model Interoperability (Std_logic_1164) (vhdl-std-logic)
- P1364.1 Standard for Verilog Register Transfer Level Synthesis (VLOG-Synth)
- P1497 Standard for Standard Delay Format (SDF) for the Electronic Design Process
- P1499 Standard Interface for Hardware Description Models of Electronic Components
- P1577 Object Oriented VHDL (oovhdl)
- P1603 Standard for an Advanced Library Format (ALF) Describing Integrated Circuit (IC) Technology, Cells, and Blocks
- P1604 Library IEEE
- P1076.6 Standard for VHDL Register Transfer Level (RTL) Synthesis

IEC TC93 Design automation

- Chairman : Mr Osamu Karatsu (JP)
- Liaisons : Internal IEC Liaison : SC 3D, SC 23J, SC 65E, TC 91
- Liaison ISO : ISO/TC 44/SC 12, ISO/TC 184
- Working Group :
  - WG 1 - Electronical data harmonization (Approaches, methodologies and technologies)
  - WG 2 - Component, circuit and system description languages
  - WG 3 - Product Data Exchange (PDX) characteristics and methodology
  - WG 5 - Test, validation, conformance and qualification technologies
  - WG 6 - Library of Reusable Parts for Electrotechnical Products
  - WG 7 - Testing of Electrotechnical Products
- Joint Working Group : JWG 11 - Product description standard for printed board, printed board assembly, and testing in XML schema

2. STARC RTL設計スタイルガイド

- 論理回路の振る舞いを記述する手引き
  - 規則に沿った書き方をすると、間違が少ない
- VHDL版とVerilog-HDL版がある
  - それぞれ日本語版と英語版がある
- 約600個の規則を3つ（5つ）に分類
  - 必須。守らないときには、理由を文書化するとよい
  - 推奨。推奨には1，2，3の区分あり
  - 参考。自社の規則を作る際に検討

	VHDL	Verilog-HLD
日本語	初版	第2版（追加、改定あり）
英語	初版	初版(pdfは第2版)

スタイルガイドの状況

- ASIC向けのガイドが基本。
- Verilog-HDLでは,第に版でFPGA向けの節を追加。
- Verilog-HDLの第二版は本では日本語だけ。
- CSVなど古典的ソフトでの記述がある。
- 定量的な記述は当時の制約の影響を受けている。
- FPGAの場合の詳細が未整備。
  - Xilinx, Alteraのガイドを参考に対応付けをするとよい。

スタイルガイドの章構成

- 1章　基本設計制約
- 命名規則,設計スタイル,クロック,(非)同期設計,階層設計
- 設計を開始時に考慮する設計制約
- 2章　RTL記述テクニック
- 組合せ回路,順序回路の記述スタイ
- always 文,function 文,if 文,case 文
- 3章　RTL設計手法
- 分割設計,機能ライブラリ,設計資産のパラメータ化、テスト容易化設計、低消費電力設計、設計データの管理
- 4章　検証のテクニック
- テストベンチのパラメータ化、タスクの使用、検証の進め方

3. 教育のための準備

- ルールの体系の確認
  - 番号の振りなおし
- 教育の順番を確認
  - <1>:実習前に
  - <2>:実習中に
  - <3>:進んでいる人が自習
  - <4>:教育の作業の範囲外

次頁以降の検討成果の汎例

- <1>,<2>,<3>,<4>が教育順序。
- (A.x), (B.x), (C.x), (D.x)は新しい章節番号
- 下線の文：Verilog版2.0での追加項目
- 青文字の項目：RTL設計スタイルガイドで省略している注記
- ＜小川清による補足＞
- 赤字：初学者のうちから習慣付けするとよい項目

7章　基本設計制約 (A)

- A.1 静的規則
  - 1.1 命名規則　<1>

＜補足:プログラムを書く際に、名前は必ず使うため＞

- A.2 動的規則　<2>
  - 1.2 同期設計(A.2.1)
  - 1.3初期リセット(A2.2)
  - 1.4 クロック(A.2.3)
  - 1.5 非同期設計(A.2.4)
- A.3 より複雑な事象　<4>
  - 1.6 階層設計(A.3.1)
  - 1.7 FPGA(A.3.2)<Verilog-HDL版ver.2>

2章　RTL記述技法 (B)

- B.1 always文 <2>
  - 2.1組み合わせ回路(B.1.1)
  - 2.2組み合わせ回路のalways文記述(B.1.2)
  - 2.3 FFの推定(B.1.3)
  - 2.4 ラッチ記述(B.1.4)
  - 2.5 トライステート・バッファ(B.1.5)
  - 2.6 回路構造を意識したalways文記述(B.1.6)
- B.2 制御文 <2>
  - 2.7 if文記述(B.2.1)
  - 2.8 case文記述(B.2.2)
  - 2.9 for文記述(B.2.3)
- B.3 その他のRTL記述
  - 2.10 演算子と代入文の記述(B.3.1) <2>
  - 2.11 ステートマシン記述(B.3.2) <3>

3章 RTL設計手法(C)

- C.1 機能ライブラリ <3>
  - 3.1 機能ライブラリの作成(C.1.1)
  - 3.2 機能ライブラリの使用(C.1.2)
- 3.3 試験容易化設計(DFT)(C.2) <4>
- 3.4 低消費電力設計(C.3) <3>
- 3.5 ソースコード, 設計データの管理(C.4) <1>

＜補足:ソースコードを必ず扱うため＞

4章　検証技術(D)
(Verification Techniques)

- D.1 試験台(test bench)
  - 4.1 試験台記述(D.1.1) <2>
  - 4.2 手続記述(Task description)(D.1.2) <3>
- 4.3 検証の進め方(Verification process)(D.2) <3>
- 4.4 ゲート水準模擬試験(simulation)(D.3) <4>
- 4.5 静的(static)刻時(timing)解析(D.4) <3>

7章 基本設計制約
7.7. 命名規則

- 1.1.1. 基本命名規則を守る
- 1.1.2. 回路名や端子名は階層構造を意識した命名規則に従う
- 1.1.3. 信号名には意味のある名前を付ける
- 1.1.4. includeファイル、パラメータ、defineの命名規則(VHDL版と異なる)
- 1.1.5. クロック系を意識した命名をする

1.1.1. 基本命名規則を守る

- ① ファイル名は、”<モジュール名>.v” とする (Verilog) [推奨2] ＜解説：推奨なのは過去の道具、遺産があるため＞
- ② 使用文字は、英数字と’_’、最初の文字は英字のみ[必須]
- ③ Verilog HDL(IEEE1364), SystemVerilog(IEEE1800), VHDL(IEEE1076.X)の予約語は、使用しない [必　須] ＜解説：他の言語と連動して使う場合に困る＞
- ④ ”VDD”, ”VSS”, ”VCC”, ”GND”, ”VREF” で始まる名前は、使用しない（大小文字とも） [必須] ＜解説：回路シミュレータと連動して使う場合に困る＞
- ⑤ 英字の大文字/ 小文字で、名称を区別しない(Abc,abc など) [必須] ＜解説：可読性が良くないし、OSや道具によっては混同するため＞
- ⑥ 最上位のポート名・モジュール名は ’_’ を最後に使用及び連続使用しない [推奨1]
- ⑦ 負論理信号は極性がわかるように末尾に識別記号(”_X”, ”_N” など) を付ける [推奨2]<補足：_Xよりは_Nの方がよい。>
- ⑧ インスタンス名はモジュール名を基本とし、複数使用されるインスタンス名は”<モジュール名>_< 数値>” とする (Verilog) [推奨3]
- ⑨ 最上位のモジュール名・ポート名は16 文字以内で大文字/ 小文字を混在させない[推奨1]
- ⑩ 使用するASIC ライブラリと同じインスタンス名、セル名を使用しない[必須]

1.1.2. 回路名や端子名は階層構造を意識した命名規則に従う

- ① モジュール名、インスタンス名は、2 文字以上、32 文字以下とする[必須] ＜解説：1文字の名前は一覧を作り、目的を記載する(逸脱文書）。それ以外の目的では使わない事、今後、1文字の名前を作らないことを徹底する＞
- ・16 文字以下を推奨する（推奨2）
- ・階層を含んだインスタンス名は128 文字以内とする（推奨3）
- ② 最上位(TOP) にある階層(ブロック名)の先頭には、階層識別文字を付加する[推奨3]
- ③ サブ階層の階層識別文字には、上位階層の階層識別文字を付加する[推奨3]
- ④ 各ブロック出力端子名の先頭文字は、”＜階層識別文字>”+”_” とする[推奨3]
- ⑤ ブロックの入力信号名・出力信号名は、内部の信号とは別の命名規則があるとよい [参考]
- ⑥ 出力端子名と接続されるネット名は同一とする[推奨2]
- ・上位階層のネット名と入力端子名を同一とする（推奨2）

1.1.3. 信号名には意味のある名前を付ける

- ① ブロック内部の信号名は、入出力信号名とは別の命名規則があるとよい [参考]
- ② 階層内部は、意味のある理解しやすい信号名を付ける [参考]
- ③ 信号名、ポート名、パラメータ名、define 名、ファンクション名は、2 文字以上、40 文字以下とする (Verilog) [必須]
  ＜解説：1文字の名前は一覧を作り、目的を記載する（逸脱文書）。それ以外の目的では使わない事、今後、1文字の名前を作らないことを徹底する＞
- ・24 文字以下を推奨する（推奨2）

1.1.4. include ファイル,パラメータ,define の命名規則(VHDL 版と異なる)

- ① include ファイルは、RTL 記述に対しては ”.h”, ”.vh”, ”.inc”、テストベンチに対しては ”.h”, ”.inc”, ”.ht”, ”.tsk”とする(Verilog HDL only) [推奨2]
- ② パラメータ名は、異なる命名規則があるとよい(Verilog) [推奨3]
- ③ 同一名称のパラメータを、異なるモジュールで使用しない(Verilog) [推奨3]
- ④ defineは、同一モジュール内で宣言されたもののみを使用する(Verilog HDL only) [推奨1]
- ⑤ 1 つの階層内だけで使用するパラメータは、階層識別文字を付加するとよい [参考]
- ⑥ 定数を出力ポートに直接出力しない [推奨1]
- ・入力ポートにも定数を入力しないほうがよい(参考)
- ⑦ 再利用を考えた回路は、必要なポートのビット幅をパラメータ化する [推奨3]
- ⑧ パラメータも<数値>’b,’h,’d,’oの指定を明確化する(Verilog HDL only) [推奨1]
- ⑨ 32ビット幅以上の場合は、ビット幅を指定する(Verilog HDL only) [必須]

1.1.5. クロック系を意識した命名をする

- ① レジスタの出力信号名にはクロック系やレジスタを意識した命名をする [推奨3]
- ② クロック信号名はCLK またはCK、リセット信号はRST_X またはRESET_X、イネーブル信号はEN を基本とし、末尾に種別を付加する[推奨3]
- ③ クロック系を意識した命名 [参考]
- ④ レジスタを意識した命名 [参考]

1.7. FPGA
1.7.1. ASICとFPGAの両方を使用する場合の注意点

- ① 1 つのFPGA 内でゲーティッドクロックを使用しない [推奨1]
- ② FPGA とASIC で異なる部分はCORE の記述から除外する [参考]
- ③ FPGA, ASIC で異なる記述は`ifdef で切り替える [参考]

3章 RTL設計手法

3.5. ソースコード、設計データの管理

- 3.5.1. ディレクトリを目的ごとに作成する
- 3.5.2. ファイルのサフィックス名
- 3.5.3. ファイルヘッダに必要な情報を定義する
- 3.5.4. ファイルのバージョンを管理する
- 3.5.5. ファイルのバックアップは定期的に
- 3.5.6. コメントを多用する
- (3.5.7. バージョン管理にCVSを使用する)
- (3.5.8. CVSの基本操作)
- (3.5.9. CVSの高度な使い方)
- (3.5.10. CVSの履歴によって変更内容を確認する)
- 3.5.11. バグトラッキングシステムを構築する

3.5.1. ディレクトリを目的ごとに作成する

- ① RTL 記述、合成スクリプト、合成結果、Sim 結果は、異なるディレクトリに保管する [必須]
- ② RTL 記述のディレクトリにはテストパターンを置かない [推奨1]
- ③ データのバージョン管理はディレクトリをコピーして、いらないファイルを消去する [推奨2]
- ④ ファイル名は相対パスで参照する [推奨1]

3.5.2. ファイルのサフィックス名

- ① 各ファイルの参照は相対パスで指定（1 度master 階層まで戻る） [推奨1]
- ② RTL 記述は、”< モジュール名>” ＋ ”.v” (Verilog) [推奨1]
- ③ テストベンチは、”_tb.v”,”_test.v”, ”.vt” (Verilog) [推奨3]
- ④ ゲート記述は、”< モジュール名>” ＋ ”.v” あるいは、”.vnet” (Verilog) [推奨3]
- ⑤ include ファイルは、RTL 記述に対しては ".h",".vh",".inc"、テストベンチに対しては".h",".inc",".ht",".tsk" にする(Verilog HDL only) [推奨2]
- ⑥ Unix 上の実行ファイルは、”.run” [推奨3]
- ⑦ シミュレーション用(Verilog HDL)スクリプトファイルは、”.v_scr” [推奨3]
- ⑧ lint 用スクリプトファイルは、”.l_scr” [推奨3]
- ⑨ 合成スクリプトは、”.scr” [推奨3]
- ⑩ 合成、シミュレーション、レイアウトのlog はすべて、”.log” [推奨3]
- ⑪ lint のlog は、”.l_log” [推奨3]
- ⑫ 合成のレポートファイルは、”.rep” あるいは、”.tim” あるいは”.ara” [推奨3]
- ⑬ SDF ファイルは、”.sdf” [推奨1]
- ⑭ EDIF ファイルは、”.edif” または”.edf” [推奨1]
- ⑮ 合成データベースは、”.db” [推奨1]

3.5.3. ファイルヘッダに必要な情報を定義する

- ① ファイルヘッダに回路名、回路機能、作成者、作成日などを示す [推奨1]
- ② 再利用の場合には修正者や修正項目を示す [推奨1]
- ③ ファイルヘッダを共通化する [推奨1]
- ④ 必要があればCVS を使用する [推奨3] <補足:当時もRCSなどの別の版管理の道具はあった。現在はSubversionなどを使うことがある。CVS固有の記述は省略する。>

3.5.4. ファイルのバージョンを管理する

- ① 複数メンバーでの開発ではマスターデータと個人用データを分離する [推奨1]
- ② CVS でファイルのバージョンを管理することができる [参考]
- <補足:当時もRCSなどの別の版管理の道具はあった。現在はSubversionなどを使うことがある。CVS固有の記述は省略する。>

3.5.5. ファイルのバックアップは定期的に

- ① ファイルのバックアップを行なう [推奨2]
- ② 設計ディレクトリ全てを定期的にバックアップする [推奨2]
- <補足>版管理の道具を使ったり，ディスクごとバックアップを取ったり，バックアップを常時取っているネットワークディスクを使ったり，方法は様々。

3.5.6. コメントを多用する

① コメントの多用によりソースコードの可読性が向上する [推奨2]

② 記述内の演算子などに、その目的や内容を示すコメントを付ける [推奨2]

③ 入出力ポート、宣言は1 行で記述し必ずコメントを付ける [推奨2]

④ コメントはできるだけ英語で記述する [推奨2]

⑤ 日本語のコメントはEDA ツールによっては読めないことがある [参考]

⑥ もっともトラブルの少ない日本語コードEUC を使用する [推奨1]

⑦ コメントは、// で始める [推奨3]

コメントに関する補足（小川）

- コメントを多用するとよくない場合
  - 言語記述と矛盾したり，意味が不明になることがある
  - 言語記述を改訂する際に，コメントも変更しないといけない。
  - コメントの自動生成部分以外は多用しない方がよい場合もある
- //コメントの方がよい理由
  - /* */の間に入るコード例えば /* など，/* を利用しているとMISRA-Cのように関連ルールを規定しないといけない
  - /* */だと，どこからどこまでがコメントかを理解するのにタブなどを利用する必要がある

3.5.11. バグトラッキングシステムを構築する

- ① 登録、アサイン、解決、検証の段階を明記する[参考]
- ② 状況をメールで送信するとともに表で表示する[参考]

〈補足：TRAC, Redmineなどの単独ソフトウェアを使う場合と統合設計環境に組込み可能な機能を使う場合がある。メール，表での表示はソフトウェアの機能にあるかを確認するとよい〉

4. 業務利用のための準備

- 優先順位のつけ直し
  - 個々人に優先順位をつける
  - 班構成の会合で上位10個を確認（班に一人は専門家）して班での合意を作成
- 過去に公開講座で数度実施（SWEST2008を含む）
  - 累計上位10を紹介
  - 関連項目の中で再掲載(KWIC:key word in context)

優先順位上位10

1.2.1	必須	② AND,OR などのプリミティブセルでRS ラッチ、FF を作成しない
1.2.1	必須	③ 組み合わせ回路のフィードバックは使用しない
1.3.1	必須	⑥ 同一リセットラインで非同期リセットと同期リセットを混在させない
1.4.3	推奨1	④ クロック信号は、FF のクロック入力端子以外（D 入力等）には供給しない
1.4.3	推奨3	⑤ クロック信号は、ブラックボックスや双方向端子、リセットラインに接続しない
1.5.1	必須	③ 非同期間転送後、1 段目のFF でフィードバックループをしない
2.1.2	必須	③ 引数はすべてfunction文の入力として定義する
2.1.3	必須	① 引数のビット幅とfunction文の入力宣言のビット幅を合わせる(Verilog HDL only)
2.1.3	必須	⑤ function文内では、グローバル信号に代入を行なわない(Verilog HDL only)
2.1.5		・(?)は最大でも10 回までにする

RTL1.2同期設計

	1.2.1. クロック同期設計を行なう
推奨1	① 可能であれば、単一クロック、単一エッジで設計する
必須	② AND,OR などのプリミティブセルでRS ラッチ、FF を作成しない
必須	③ 組み合わせ回路のフィードバックは使用しない

RTL1.3　初期化りセット

	1.3.1. 初期リセットは非同期リセットにする
参考	① 同期リセット記述をしても、リセットできない回路が合成されることがある
推奨3	② レジスタに対する初期リセットは非同期リセットの方が問題が少ない
	・レイアウト時、リセットツリー合成が実行しやすい
	・同期リセットではゲートレベル・シミュレーションで値が確定しないことがある
推奨1	③ 初期リセット以外では、非同期セット・リセット端子を使用しない
参考	④ 同期リセット回路を利用する場合は、同期リセット付きレジスタの階層を設ける
推奨3	⑤ 特定の論理合成ツールに依存する同期リセットディレクティブは使用しない
必須	⑥ 同一リセットラインで非同期リセットと同期リセットを混在させない
推奨1	⑦ 非同期セット、非同期リセット、両方付きのFF は使用しない