Vol.2018-CE-147 No.14
2018/12/2

手軽でポータブルなVHDL演習環境のための フロントエンドツールGGFrontの開発

藤枝 直輝[1,a)]

概要：本稿ではGGFrontという，VHDLを用いたディジタル回路設計演習を手軽かつポータブルに実施できる環境の提供のために開発しているツールについて述べる．GGFrontはC#で記述され，VHDLシミュレータGHDLと波形ビューアGTKWaveに対するフロントエンドのGUIを提供する．GHDLには初学者にとって扱いづらい点がいくつかあるため，GGFrontではこれらをユーザから隠蔽し，ユーザのハードウェア記述をすぐにシミュレートできるように設計されている．本稿では本ツールの設計と実装，動作例について述べるとともに，本ツールの試作版を講義で利用した結果について報告する．

Development of front-end tool GGFront for convenient and portable VHDL practice environment

Naoki Fujieda[1,a)]

Abstract: This report describes GGFront, a tool developed by the author to present a convenient and portable practice environment of digital circuit design using VHDL. GGFront is written in C# and it presents a front-end GUI to the GHDL simulator and the GTKWave waveform viewer. Since GHDL has some difficulties for beginners to deal with, GGFront is designed to hide such difficulties from users and let users simulate their own hardware description quickly. The design, implementation, and working examples of the tool are described in this report. Also, the result of the use of a prototype version of the tool in a lecture is reported.

1. はじめに

ハードウェア記述言語（HDL）の習得は，ディジタル回路の本格的な設計を学ぶために重要である．基本的な設計原理を学ぶには回路図を用いれば十分であるが，ある程度以上の規模の回路を扱い，その動作を確認したい場合は，HDLを用いた抽象的な記述を用いた方が便利である[1]．また，企業などの現場での設計はHDLを用いることが主流であることから，技術者としてHDLを習得していることにはメリットがある．

しかしながら，HDLの演習を手軽に行える環境を整備することはそれほど容易ではない．特に，演習を講義科目の授業外学習として行えるようにすることは，挑戦的な課題である．演習室でHDLのシミュレーション環境を整える場合，部屋や機材の確保が問題となるし，部屋を利用できる時間帯も限られがちである．学生各自のPCに何らかのツールをインストールさせる場合，大量のダウンロードやディスク容量を必要としたり，利用にノウハウが必要だったりと，学生にとっての負担が大きい．

著者は，こうした問題を解決し，HDLの一種であるVHDLを用いたディジタル回路設計演習を手軽かつポータブルに実施できる環境を提供するため，**GGFront** [*1]というフロントエンドツールを開発している．この環境では，学生が各自で利用することを念頭に，ダウンロード量や必要なディスク容量を数十MB程度に小さくし，かつパッケージを展開したらすぐに利用を開始できることを目標とした．そのため，フリーソフトウェアのVHDLシ

[1] 豊橋技術科学大学
Toyohashi University of Technology
[a)] fujieda@ee.tut.ac.jp

[*1] GHDL/GTKWave GUI Frontendの意．

ミュレータである GHDL [2]，および波形ビューアである GTKWave [3] を配布用パッケージに同梱し，GGFront はこれらに対する GUI フロントエンドを提供するツールとして開発している．なお，学生の多くは Windows に習熟していることから，本環境や GGFront も Windows 用として提供している．

本稿では関連研究として既存の HDL シミュレータやディジタル回路教育プラットフォームについて述べたあと，本ツールの設計と実装，動作例について述べる．さらに，本ツールの試作版を講義で利用した結果について報告する．

2. 関連研究

2.1 HDL シミュレータ

商用の HDL シミュレータとして，特に FPGA（Field Programmable Gate Array）を用いた実機での動作確認を念頭に置いている場合は，各 FPGA ベンダーが提供している CAD ツールに同梱されたシミュレータがしばしば用いられる．Xilinx 社の Vivado には Vivado Simulator [4]，Intel 社の Quartus Prime には Mentor 社の ModelSim - Intel FPGA Starter Edition [5] が同梱されており，それぞれ無償で利用できる．FPGA を利用する場合はこれらを用いることが妥当であるが，CAD ツールのインストールには数 GB のダウンロードと数十 GB のディスク容量が必要である．また，FPGA 製品自体の終息に伴い古いツールのサポートが打ち切られ，新しい OS で動作させることが困難になる場合もある．例えば，Xilinx 社の旧世代のツールである ISE は Windows 10 での動作をサポートしておらず，実際に多数の不具合が生じている．Xilinx 社は Windows 10 での ISE の利用に際し，仮想マシン上の Linux 環境で動作させる方法を案内している [6]．

国産の HDL シミュレータとしては Veritak [7] がある．対応する言語は Verilog HDL であるが，VHDL からのコード変換もサポートしているエディションがある．高速なシミュレーションとデバッグのための豊富な機能が特徴であるが，有料のツールであり，授業で学生各自の PC に導入させる用途には適さない．

フリーソフトウェアの HDL シミュレータとしては，Verilog HDL 向けには Icarus Verilog [8]，VHDL 向けには GHDL [2] が知られている．これらはいずれも波形ファイルを出力する機能をもっており，得られた波形ファイルを GTKWave [3] に与えることにより，シミュレーション結果をグラフィカルに確認できる．Icarus Verilog は VHDL からのソース変換機能をもつものの，実験的機能として位置づけられており，正しい変換ができない場合がある．

本稿で提案する GGFront は，内部で GHDL と GTKWave を呼び出す GUI フロントエンドツールである．GHDL と GTKWave の組み合わせに対するインタフェースを提供する他のツールとして，ModelSim 風のコマンド（vcom や vsim など）を提供するラッパである model-ghdl [9] がある．ただし，これは ModelSim と組み合わせることを意図した開発環境で使うことを想定したツールである．

2.2 既存プラットフォーム

ディジタル回路の包括的な実習プラットフォームとして，ジェノヴァ大学の Deeds [10] がある．Deeds は回路図ベースのシミュレーション環境である Deeds-DcS を含んでおり，回路図によるディジタル回路設計とその結果の確認を容易に行える．また，設計した回路を VHDL へとエクスポートする機能も備えており，FPGA 評価ボード上での動作確認も行える．しかし，VHDL を直接シミュレートすることは想定されていない．

ディジタル回路の設計を含む，マイコンや計算機システムを広く学ぶための教材には，徳山高専の TeC [11] や東工大の MieruPC [12] などが挙げられる．これらは FPGA を搭載し，通常は CPU や入出力回路などを搭載したマイコンを FPGA に書き込むようになっている．そのためこれらは，通常時は組込みソフトウェアや OS の学習に利用でき，FPGA 自体を書き換えればディジタル回路の設計演習にも使用できる，という特徴をもっている．また仙台高専のカメレオン AVR および Donkey [13] も，マイコンと CPLD（Complex Programmable Logic Device）を混載することによって，同様の性質を達成している．こうした教材は演習室内での利用を想定したものだが，遠隔操作のシステムを別途開発することで，演習室外でも利用可能にすることが試みられている [13]．こうした教材は，複数の講義・実験科目で横断的に利用することにより高い教育的効果を発揮できると考えられるが，単一の講義科目の授業外学習として用いるにはコストが大きすぎる．

3. 設計

3.1 シミュレーションの流れ

図 1 に GGFront を用いた場合の VHDL シミュレーションのフローを示す．これらは以下の手順からなる．なお，白い矢印は GUI 画面上でのユーザとのインタラクション，薄いグレーの矢印は外部プログラムの実行，濃いグレーの矢印はファイルの流れを表す．

(1) ユーザは VHDL ソースファイルとシミュレーション対象のエンティティ名を指定し，シミュレーションを指示する．
(2) GGFront は指定されたソースファイルを作業ディレクトリにコピーする．
(3) GGFront は GHDL を所定のオプションをつけて実行する．
(4) GHDL はソースファイルを作業ディレクトリから読み出し，コンパイルし，シミュレーションを行う．シミュレーション結果は波形ファイル（vcd 形式）とし

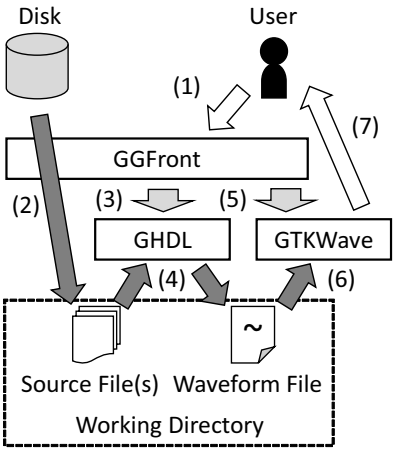

図 1　GGFront を用いた VHDL シミュレーションのフロー．
Fig. 1　VHDL simulation flow using GGFront.

て出力される．

(5) ユーザはシミュレーションが期待通り終了したことを確認し，波形の確認を指示する．

(6) GGFront は GTKWave を実行する．

(7) GTKWave は波形ファイルを読み出し，ユーザに対してグラフィカルに表示する．

また，図 1 には示していないが，波形ファイルは後でも確認ができるよう，VHDL ソースファイルと同じディレクトリにも保存される．

3.2　GHDL の扱いづらい点とその解決

本節では，GGFront が GHDL に与えるオプションの選択理由と，必要な前処理・後処理について述べる．GHDL を利用するにあたっては，適切なオプションをつけて GHDL を複数回実行する必要があり，煩雑である．また，そもそも GHDL は CUI のツールのため，Windows を利用する学生にとっては操作が難しい．GGFront の最大の目的は，図 1 の手順（3）においてこうした煩雑な手順をユーザから隠蔽し，ユーザのハードウェア記述をすぐにシミュレートできるようにすることである．

コマンドの扱い

GHDL では，まず一度ソースの解析（`-a` オプション）を行ってからコンパイル・実行（`-r` オプション）を行う必要がある．当然ながら，途中でエラーが発生した場合はその内容をユーザに伝えて，図 1 に示したシミュレーションのフローを中止しなければならない．またソースの解析においては，ソースファイルの指定順によっては，あるエンティティが存在しているにもかかわらず，それが見つからない場合がある．これは警告として報告させることもできる（`--warn-default-binding` オプション）．これが偽の警告であるか，本当にどのソースファイルにも存在しないエンティティを呼び出そうとしているかは，もう一度ソースの解析を実行することで確かめられる．

以上より，GHDL でシミュレーションを行う場合，出力を確認しながら 2 回もしくは 3 回繰り返して GHDL を実行しなければならない．GGFront ではこれらの操作を自動的に行い，途中でエラーが発生すればシミュレーションのフローを中止し，警告が発生すればフローを継続するかどうかユーザに確認する．

VHDL におけるシミュレーション終了の扱い

広く利用されている VHDL-93 の言語仕様では，シミュレーションを終了するための標準的な手法がない．広く使われている方法として以下の 3 つが挙げられる．

(1) テストベンチでの信号の遷移を停止させる方法

(2) 常に失敗する assert 文でシミュレーションを失敗させる方法

(3) report 文でシミュレーションを失敗させる方法

GGFront では (2) の方法をサポートする．つまり，assert 文によってシミュレーションが失敗した場合は，それをシミュレーションの正常終了とみなす．また，assert 文の記述がない場合にもシミュレーションが終了するように，シミュレーション時間の上限を設けている．上限は 1 ミリ秒に設定している．GHDL に与えるべきオプションは，`--stop-time=1ms` である．

出力される波形ファイルの末尾の扱い

GHDL に`--vcd=[`ファイル名`]` オプションを与えることで出力できる波形ファイルの末尾には，シミュレーションの終了時刻の情報が付与されていない．このため，このファイルをそのまま GTKWave に与えると，最後にいずれかの信号が遷移してからシミュレーションが終了するまでの間の波形が表示されない．シミュレーションの終了の方法として上述の（1）の方法を用いる場合には，信号の遷移が停止したあとの余分な部分を除去できるので，この扱いは妥当である．しかし，（2）または（3）の方法を用いる場合には都合が悪い．

GGFront ではシミュレーションが assert 文によって終了した場合，後処理として出力された波形ファイルの末尾に assert 文が実行された時刻を追記する．これにより，シミュレーションが終了するまでの波形を正しく表示できる．

ライブラリの扱い

VHDL では，std_logic_unsigned などの非標準のライブラリが広く使われている．これらのライブラリには=などの演算子の定義に問題があり，こうした演算子を用いた場合，文法上厳密に解釈すると演算子の多重定義によるエラーを生じる．GHDL ではデフォルトでは IEEE の標準のライブラリだけを使用し，非標準ライブラリによるこうしたエラーを無視しないように設定されている．

GGFront では非標準のライブラリを使用することを想定し，GHDL にライブラリの扱いを変更するオプションを与えている．具体的には，非標準のライブラリを使用する `--ieee=synopsys` オプションと，上述のエラーを無視す

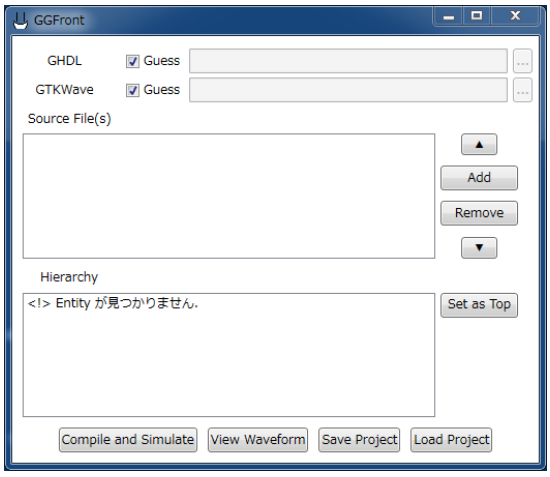

図 2　GGFront のウィンドウの外観.
Fig. 2　Appearance of GGFront window.

る`-fexplicit` オプションの 2 つである．加えて，シミュレーション開始直後の不定値により発生するライブラリの警告を抑制するため，`--ieee-asserts=disable` オプションを与えている．

コメントの扱い

GHDL では，デフォルトではコメントに日本語などのマルチバイト文字が含まれていた場合には文法エラーとして解釈する．これを回避する`-C` オプションも用意されているが，GGFront ではソースファイルを作業ディレクトリにコピーする時に，前処理としてコメントを除去することで対処している．

本節のまとめ

以上より，GHDL を実行するにあたり必要な処理をまとめると以下の通りである．まず，前処理としてソースファイルを作業ディレクトリにコピーする際，コメントを除去する．次に，GHDL を

`ghdl -a [コンパイルオプション] [ソースファイル名]`

と実行してソースの解析を行う．解析に問題がなければ，

`ghdl -r [コンパイルオプション] [エンティティ名] [シミュレーションオプション]`

と実行してシミュレーションを行う．シミュレーションが正常終了した場合（すなわち assert 文によって「失敗」した場合），後処理として波形ファイルの末尾にシミュレーションの終了時刻の情報を追記する．ここで，コンパイルオプションとは`-fexplicit --ieee=synopsys --warn-default-binding` であり，シミュレーションオプションとは`--vcd=wave.vcd --ieee-asserts=disable --stop-time=1ms` である．

4.　実装

図 2 に，2018 年 9 月末時点での GGFront の最新版である v0.3.0 のウィンドウの外観を示す．GGFront は C#を使用した，WPF（Windows Presentation Foundation）による GUI プログラムとして記述されており，その開発には Visual Studio 2017 を使用している．全ての機能を単一のウィンドウにまとめることにより，利用法を簡潔化している．ウィンドウは上から順に 4 つの部分に大別される．まず GHDL や GTKWave のバイナリの所在を指定する部分，次にソースファイルを管理するリスト，エンティティの階層構造を管理するリストが続き，最後に操作を行うための一連のボタンが配置されている．なお，ウィンドウのサイズは可変であり，ウィンドウの縦幅を変更した場合，リストの縦幅がそれに応じて変更される．

表 1　GGFront v0.3.0 の主要なソースコードの一覧.
Table 1　List of main source codes of GGFront v0.3.0.

File Name	#Lines	Description
App.xaml	12	Properties of Application
MainWindow.xaml	122	Properties of Window
App.xaml.cs	30	Main Function
MainWindow.xaml.cs	416	Interaction Logic
Util.cs	609	Utility and other classes
(Total)	1,189	1,055 lines in C#

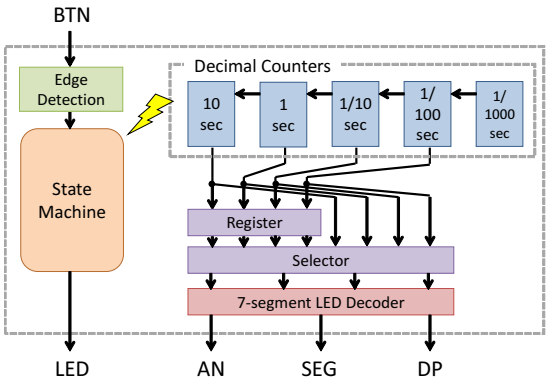

図 3　ストップウォッチ回路のブロック図.
Fig. 3　Block diagram of a stopwatch circuit.

表 1 に，GGFront v0.3.0 の主要なソースコードの概要および行数を示す．C#ソースコードにおいて，ウィンドウ中の各種コントロールとの相互作用は MainWindow.xaml.cs に，その他のユーティリティ関数をもつクラスや処理結果を保存するクラスなどは Util.cs に記述されている．メイン関数を含んだ C#ソースコードの合計は 1,055 行である．

5.　動作例

本節では，ストップウォッチ回路を例に GGFront がどのように動作するかを説明する．ストップウォッチ回路は 8 個の回路記述のソースファイルと 1 個のテストベンチのソースファイルから構成される．回路記述はトップレベルの記述のほか，構成要素であるステートマシン，エッジ検出回路，10 進カウンタ（4 桁，1 桁），スプリット時間を保持するレジスタ，表示すべき時間を選択するセレクタ，そ

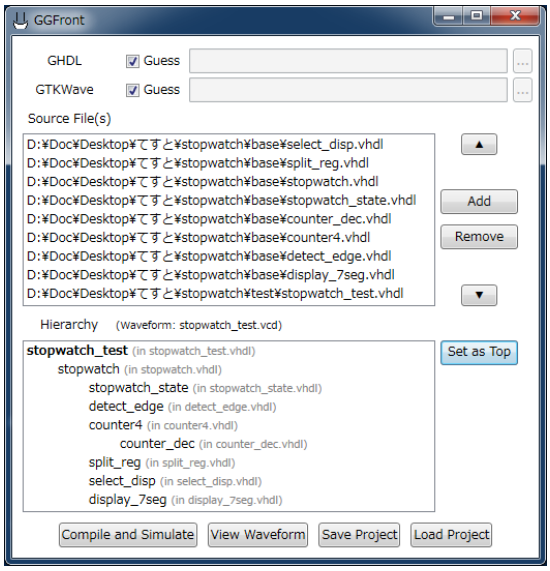

図 4 GGFront にストップウォッチ回路の記述とテストベンチを与えた場合の様子.

Fig. 4 A behavior of GGFront when circuit description and testbench of a stopwatch circuit are given.

して 7 セグメント LED デコーダからなる．これらの構成要素の接続関係を**図 3** に示す．

GGFront を最初に実行する場合，まずは GHDL と GTKWave のバイナリの所在を指定する．GGFront は，通常 GHDL と GTKWave のバイナリを同梱したパッケージで配布されるので，パッケージのディレクトリ構造を変更していなければ，自動的にパッケージ内のバイナリが検出される．そうでない場合は適当なディレクトリを自分で指定する必要がある．

これ以降はシミュレートしたい回路ごとに行われる手順である．回路記述およびテストベンチの全てのソースファイルを，Add ボタンを押すと表示されるダイアログから選択するか，エクスプローラ等から上部のリストにドロップすることによって，上部のリストに追加する．このとき，自動的にファイル内の entity 文，component 文の記述が解析され，エンティティの階層関係が下部のリストに表示される．

図 4 に，ファイルのリストにストップウォッチ回路の全てのソースファイルを追加したときの GGFront の様子を示す．下部のリストから，最上位にテストベンチ（stopwatch_test），その下にストップウォッチ回路のトップレベル記述（stopwatch），更にその下に各種の構成要素が並んでいることが確認できる．また，4 桁出力の 10 進カウンタ（counter4）は 1 桁の 10 進カウンタ（counter_dec）を複数使って構成されているので，counter4 の下位に counter_dec があることも確認できる．

所望の階層構造が得られていることが確認できたら，下部のリストからテストベンチを選択して Set as Top ボタンを押すと，これをシミュレーションの対象にできる．下部のリスト中では対象のエンティティは太字で表示される．

その後，Compile and Simulation ボタンを押すと図 1 のフローの（4）までが行われる．正しく解析・シミュレーションが行えた場合には，「シミュレーションは ● ns 後に停止しました.」というメッセージが表示される．もしエラーが発生した場合は，その旨のメッセージが表示されるとともに，エラー内容を記したテキストファイルが開かれる．最後に，View Waveform ボタンを押すと図 1 のフローの（5）以降が行われ，GTKWave が起動する．

表 2 レポート課題の一覧.

Table 2 List of assignments.

#	Topic	Sub.	Avg.
1	Simple combinational circuit	57	4.1
2	Combinational circuit (Determination of holidays)	40	4.4
3	3-bit multipliers	42	3.6
4	JK flip-flop	44	4.1
5	Sequential circuit (Simple state machine)	42	4.0
6(A)	Stopwatch circuit (Modification of counters)	24	3.0
6(B)	Logic gates with delay	31	3.5
7	Simple multi-cycle processor	36	3.8

6. 講義での利用

本節では，GGFront の試作版（v0.1.0 ～ v0.2.1）を学生に提供し，実際に講義科目の授業外学習のために用いた結果について報告する．対象科目は，著者が非常勤講師として勤務している愛知工業大学電気学科電子情報工学専攻の，3 年次前期科目である「ディジタル回路 2」である．受講者は 2 年次後期の必修科目で主に組合せ回路について回路図を用いて学修しており，本科目は順序回路を含むより実践的なディジタル回路の設計・検証を学ぶための科目として開講されている．

GGFront の試作版では v0.3.0 と異なり，エンティティの階層関係は表示されず，シミュレーション対象のエンティティや波形ファイルの保存先は直接指定する必要があった．GHDL と GTKWave のバイナリの自動検出は行わず，また，当初は 3.2 節で述べた非標準ライブラリの扱いに起因するエラーを取り扱っていないなどの不具合があった．

授業外学習として，2 週に 1 回のペースで，VHDL を記述してその動作を確認するレポート課題を課した．計 7 回の課題のうち，5 回以上のレポートを提出することを単位取得の条件とした．**表 2** に，各回のレポート課題で扱ったトピックと，各回のレポート提出者の人数（Sub.），提出されたレポートの 5 段階評価での平均点（Avg.）を示す．出題の都合により，第 6 回の課題は（A）（B）の 2 種類を用意し，そのいずれかに解答（両方解答してもよい）するも

のとした．

課題では，第 3 回までで組合せ回路，第 4〜5 回でフリップフロップや簡単な順序回路について学習させたあと，第 6 回（A）と第 7 回では応用として，少し規模の大きい回路の VHDL 記述（300 行程度）の一部を修正して，その機能を追加することに取り組ませた．最終的に 5 回以上のレポートを提出した学生は 43 人であった．表 2 より，応用的な課題では提出人数・平均点ともにやや低い傾向が見られた．

課題を解くための環境としては，GGFront により GHDL・GTKWave を用いる方法のほか，Xilinx 社の Vivado [4] を用いる方法（FPGA を用いた開発に興味がある学生を想定），GNU make により GHDL・GTKWave を呼び出す方法（mac OS を利用している学生を想定）の 2 通りの代替を案内した．学生の多くは GGFront を用いることを選択したが，これらの代替の方法を選択した学生もそれぞれ 1 割程度いた．

全学的に実施している授業アンケートを通じて学生の理解度・満足度をチェックしたところ，総合的に授業を「よかった」と回答した学生の割合は約 4 割であり，比較的好評なコメントが多くみられた．一方で，環境の導入にとまどいを感じたというコメントも残っており，また GGFront の不具合修正時のアナウンスが十分でなく，古いバージョンを使っていたために課題が解けなかったことへの不満のコメントもみられた．こうしたコメントのうちいくつかは，ツールの改良やノウハウの蓄積により解消されうるものと考えている．今後も本科目では継続的に GGFront を利用し，教育効果の長期的な確認・改善を図っていきたい．

7. おわりに

本稿では，VHDL を用いたディジタル回路設計演習の環境を手軽かつポータブルに提供するフロントエンドツール GGFront の設計・実装とその利用について報告した．本ツールは 2018 年 10 月より一般公開を開始しており，著者の個人 Web ページ `https://sites.google.com/site/nfproc/ggfront/` でダウンロード可能である．今後の課題として，継続的なツールの改良とツールを用いた教育効果の長期的な確認・改善が挙げられる．また，必要に応じて Verilog HDL への対応も検討したい．

謝辞 本ツールを用いた授業の実施にご理解をいただきました，愛知工業大学の小西たつ美先生，五島敬史郎先生をはじめとした電気学科電子情報工学専攻の先生方に感謝いたします．

参考文献

[1] Harris, S. L. and Harris, D. M.: ディジタル回路設計とコンピュータアーキテクチャ 第 2 版，翔泳社 (2017).
[2] Gingold, T.: GHDL, (online), available from ⟨`http://ghdl.free.fr/`⟩ (accessed 2018-08-29).
[3] Bybell, T.: GTKWave, (online), available from ⟨`http://gtkwave.sourceforge.net/`⟩ (accessed 2018-08-29).
[4] Xilinx Inc.: *Vivado Design Suite User Guide: Logic Simulation (ver2018.2)*, User Guide UG900 (2018).
[5] Intel Corp.: ModelSim – Intel FPGA Edition ソフトウェア，（オンライン），入手先 ⟨`https://www.intel.co.jp/content/www/jp/ja/software/programmable/quartus-prime/model-sim.html`⟩（参照 2018-08-29）.
[6] Xilinx Inc.: *ISE Spartan-6 VM for Windows 10*, User Guide UG1227 (2017).
[7] 菅原システムズ：Veritak, (online), available from ⟨`http://japanese.sugawara-systems.com/`⟩ (accessed 2018-08-29).
[8] Williams, S.: Icarus Verilog, (online), available from ⟨`http://iverilog.icarus.com/`⟩ (accessed 2018-08-29).
[9] Koch, M.: model-ghdl, (online), available from ⟨`https://github.com/m42uko/model-ghdl`⟩ (accessed 2018-08-29).
[10] Donzellini, G. and Ponta, D.: From Gates to FPGA: Learning Digital Design with Deeds, *Proc. 3rd Interdisciplinary Engineering Design Education Conference (IEDEC 2013)*, pp. 41–48 (online), DOI: 10.1109/IEDEC.2013.6526758 (2013).
[11] 重村哲至，守川和夫，力 規晃，新田貴之，原田耕治，山田健仁：教育用マイコンボードを用いた HDL 演習環境の実現，情報処理学会研究報告 2004-CE-078，pp. 43–48 (2005).
[12] 吉瀬謙二：シンプルな計算機システムの開発に向けた挑戦，情報処理学会論文誌，Vol. 54, No. 7, pp. 1902–1912 (2013).
[13] 千葉慎二，力武克彰，與那嶺尚弘：組込みシステムのための学習支援システムの開発と実践，情報処理学会論文誌教育とコンピュータ， Vol. 1, No. 3, pp. 30–37 (2015).