Active-HDL GUI 基本シミュレーション

Rev. 5.0

作成日：2007/4/2

最終改定日：2016/05/09

1　はじめに

ここでは、Active-HDL によるシミュレーションの基本フローを GUI で実行する方法について解説します。

サンプルデザイン V_Bjack.zip (Verilog) / bjack.zip (VHDL) を任意のディレクトリに展開してください。
Verilog：C:¥My_Designs¥V_Bjack
VHDL： C:¥My_Designs¥bjack

2　Active-HDL の起動

デスクトップ上のショートカット  をダブルクリックして Active-HDL を起動します。

- License Configuration ウィンドウ（図 2-1）

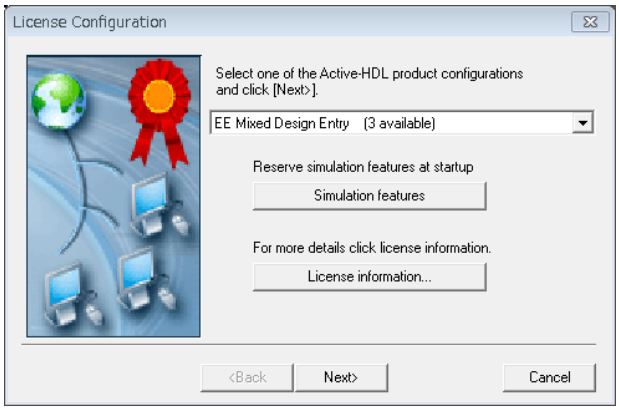


図 2-1

License Configuration ウィンドウは Active-HDL をフローティングライセンスで起動する際に表示され、使用するエディションを選択したり、使用可能なライセンス数を確認することができます。ノードロックライセンスでは表示されません。

- Getting Started ウィンドウ（図 2-2）

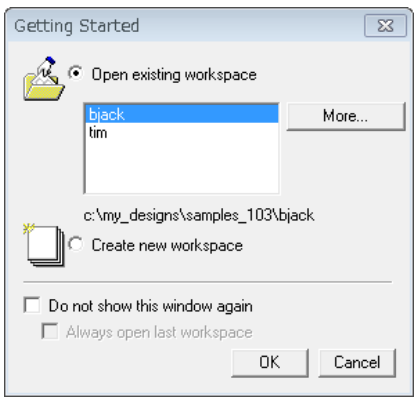


図 2-2

- Open existing workspace ：過去に作成したワークスペースが表示されています。選択して起動します。
- Create new workspace ：新しくワークスペースを作成します。
- Do not show this window again ：次回の起動から現在の設定に基づいて動作し、Getting Started ウィンドウは表示されなくなります。
- Always open last workspace ：Active-HDL を起動した際に、常に最後に使用したワークスペースを表示する場合にチェックします。

ここでは Cancel ボタンをクリックします。

3 プロジェクトの作成

Active-HDL を GUI 実行する場合、初めにワークスペース（.aws）と呼ばれるプロジェクトを作成します。更にワークスペース内にデザイン（.adf）を作成します。

3.1 ワークスペースの作成

3.1.1 New Workspace ウィザードの起動

メニューから File | New | Workspace を選択し（図 3-1）、New Workspace ウィザードを表示します。

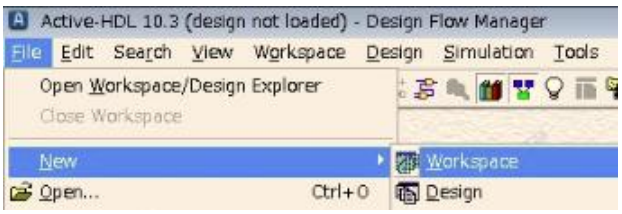


図 3-1

3.1.2 New Workspace ウィザードの設定

ワークスペースの名前と作成ディレクトリを指定します。

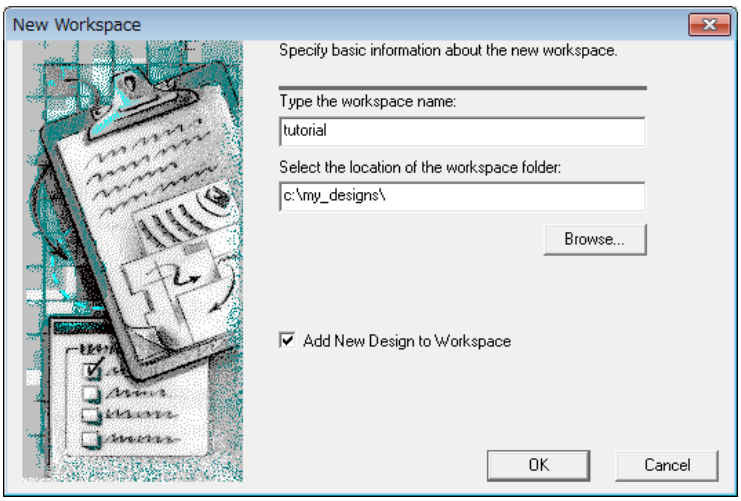


図 3-2

- Type the workspace name ：ワークスペースの名称を入力します。ここでは tutorial と入力します。
- Select the location of the workspace folder ：ワークスペースを作成するディレクトリを指定します。
  Type the workspace name で指定したディレクトリ内に、ワークスペースが作成されます。
  ここではデフォルトのまま C:¥My_Designs とします。
- Add New Design to Workspace チェックボックス ：作成したワークスペースに新規のデザインを追加します。
  ここではデフォルトのままチェックした状態とします。

図 3-2 のようになっていることを確認し、OK ボタンをクリックします。

3.2 デザインの作成

3.2.1 New Design ウィザード

New Design ウィザードではデザインの作成方法を選択します。ここでは、Create an Empty Design を選択し、次へボタンをクリックします。

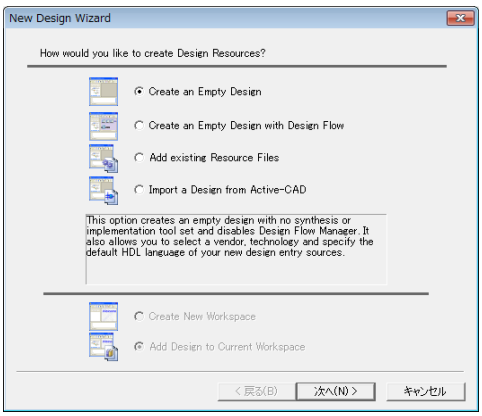


図 3-3

- Create an Empty Design ：ソースファイルを追加しないで、空のデザインを作成します。
- Create an Empty Design with Design Flow ：空のデザイン作成時に、Active-HDL から起動する論理合成ツールや配置配線ツールを指定します。
- Add existing Resource Files ：デザイン作成と同時にデザインに追加する既存の HDL ファイルを指定します。
- Import a Design from Active-CAD ： Active-CAD で作成したデザインを Active-HDL に読み込みます。
  （Active-CAD はアルデックのスケマティック・デザイン作成およびシミュレーション・ツールで、現在は販売しておりません。）

3.2.2 Property Page

使用する設計言語や FPGA ベンダを指定します。

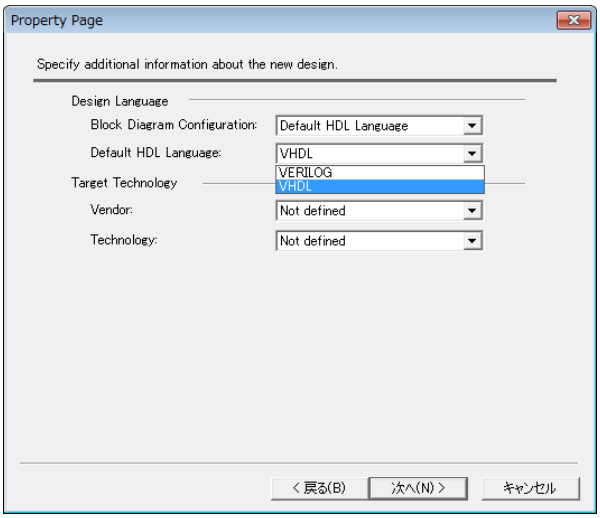


図 3-4

- Block Diagram Configuration ：ブロックダイアグラムエディタを使用する際の言語を指定します。
  ここでは Default HDL Language のままにします。
- Default HDL Language ：使用する言語を指定します。
- TagetTechnology
  - Vendor ：使用する FPGA ベンダを指定します。ここでは Not defined のままにします。
  - Technology ：使用する FPGA のテクノロジを指定します。ここでは Not defined のままにします。

図 3-4 のようになっていることを確認し、次へボタンをクリックします。

3.2.3 New Design Wizard

デザインの名前と作成ディレクトリを指定します。

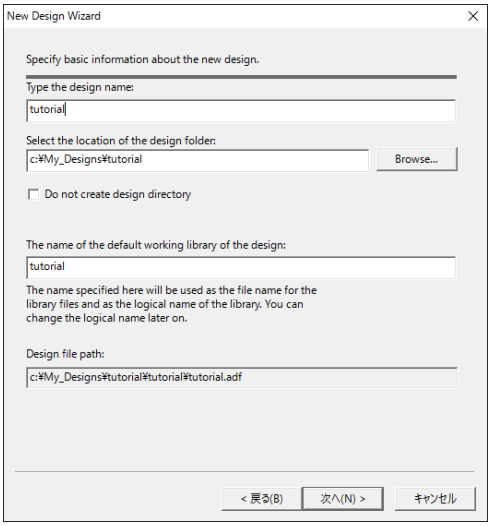


図 3-5

- Type the design name ：作成するデザイン名を入力します。ここでは tutorial と入力します。
- Select the location of the design folder ：デザインが作成されるディレクトリを指定します。
  デフォルトではワークスペース内に作成されます。ここでは C:¥My_Designs¥tutorial のままとします。
- Do not create design directory ：ワークスペースとデザインを同じディレクトリとする場合にチェックします。
- The name of the default working library of the design ：ワーキングライブラリ名を指定します。
  デフォルトでは、デザインと同名のワーキングライブラリが作成されます。
- Design file path ：デザインコンフィグレーションファイルのパスが表示されます。

図 3-5 のようになっていることを確認し、OK ボタンをクリックします。

デザイン名とディレクトリが表示されます。完了ボタンをクリックし、New Design Wizard を終了します（図 3-6）。

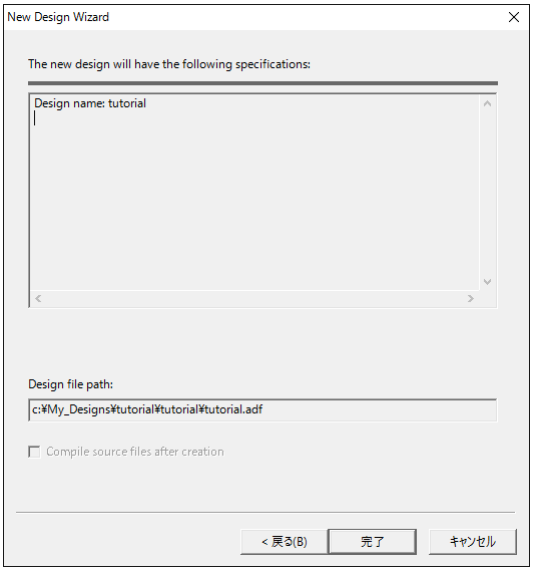


図 3-6

3.3 Active-HDL GUI

ワークスペースとデザインを作成すると Active-HDL は図 3-7 のようになります。

図 3-7

① メニュー：各種操作を用意しています。

② ツールバー：各種操作をアイコンとして用意しています。

③ Design Browser： Files タブ、Structure タブ、Resources タブで構成されています。
Files タブ： ワークスペース、デザイン、HDL ファイルや波形データを管理します。
Structure タブ： デザインの階層構造を表示します。
Resources タブ ：ログや波形ファイル等を拡張子別に分けて表示します。

④ ドキュメントウィンドウ ：波形ウィンドウや HDL テキストエディタを表示します。

⑤ Console：入出力ウィンドウで、実行コマンドや実行結果を表示します。

3.4　作成したプロジェクトの確認

ワークスペースとデザインの作成が終了すると Design Browser Files タブが下図 3-8 のようになります。

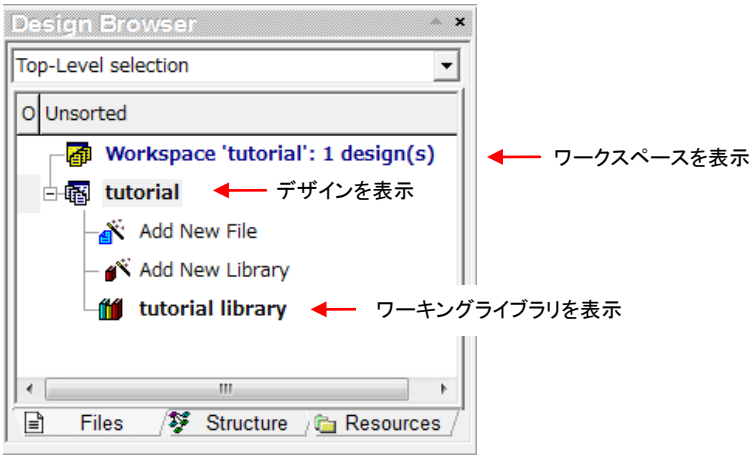


図 3-8

ここまでの作業で生成されたプロジェクトファイルを確認します。エクスプローラ等で C:¥My_Designs¥tutorial を表示します。

- tutorial.aws　:ワークスペース・コンフィギュレーション・ファイル
- tutorial　ディレクトリ　:作成したデザイン・ディレクトリ
- library.cfg　:ライブラリ・コンフィギュレーション・ファイル

上記ファイルは、通常ユーザ側で内容を把握・修正する必要はありません。

次回　Active-HDL　を起動した際に作成したワークスペース・コンフィギュレーション・ファイルを指定すると、前回終了した時点と同じ状態で起動します。下記の指定方法があります。

- Getting Started　ウィンドウ（図 2-2）でワークスペース名を指定します。
- メニューから File | Open を選択し、Open ウィンドウでワークスペース・コンフィギュレーション・ファイル .aws を指定します。
- ワークスペース・コンフィギュレーション・ファイルをダブルクリックして Active-HDL を起動します。

ワークスペース内の tutorial デザイン・ディレクトリには各種ファイルが生成されています。

- tutorial.adf　:デザイン・コンフィギュレーション・ファイル
- src ディレクトリ　:次の作業でインポートする HDL ファイルはこのディレクトリへ格納されます。
- log ディレクトリ　:操作履歴が console.log へ記録されます。

  注）console.log は Active-HDL を起動し、デザインを開くタイミングでクリアされます。
  ログをクリアしないで追記する場合には、Tools | Preferences | Environment | Console の Clear log file のチェックをはずしてください。

3.5　ソースコードのインポート

作成したデザイン tutorial へ、シミュレーション実行対象の HDL ファイルを登録します。
Design Browser Files タブの tutorial デザインを右クリックし、コンテキストメニューから Add Files to Design を選択します（図 3-9）。

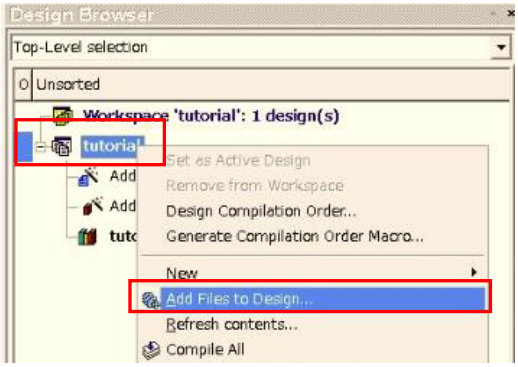


図 3-9

3.5.1　Verilog の場合

Add Files to Design ウィンドウにて V_Bjack ディレクトリへ移動し、ディレクトリ内の全ての Verilog ファイルを選択します（図 3-10）。Make local copy にチェックして、開くボタンをクリックします。

参考）Make local copy のチェックをはずすと、選択したファイルは上述の src ディレクトリへはコピーされず、元のディレクトリ内のファイルをリンクしてデザインに付加されます。

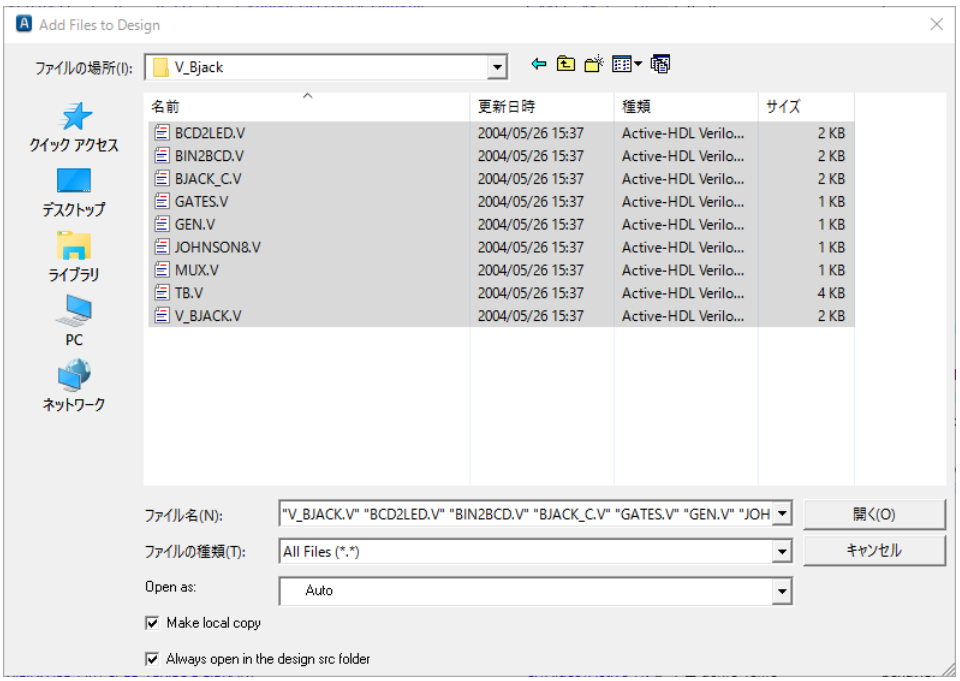


図 3-10

下図 3-11 のように Design Browser Files タブにファイルが登録されます。

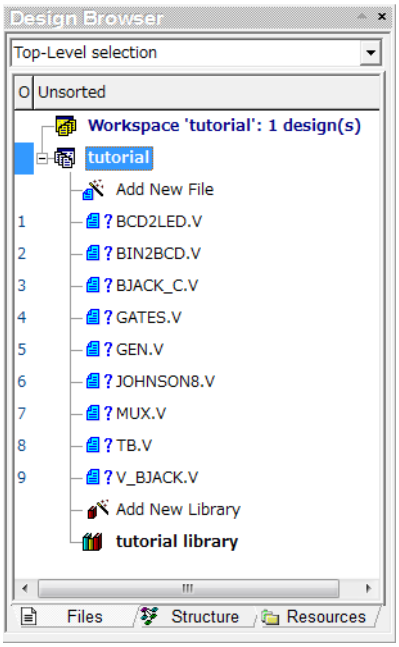


図 3-11

3.5.2 VHDL の場合

Add Files to Design ウィンドウにて bjack ディレクトリへ移動し、ディレクトリ内の全ての VHDL ファイルを選択します（図 3-12）。Make local copy にチェックして、開くボタンをクリックします。

参考）Make local copy のチェックをはずすと、選択したファイルは上述の src ディレクトリへはコピーされず、元のディレクトリ内のファイルをリンクしてデザインに付加されます。

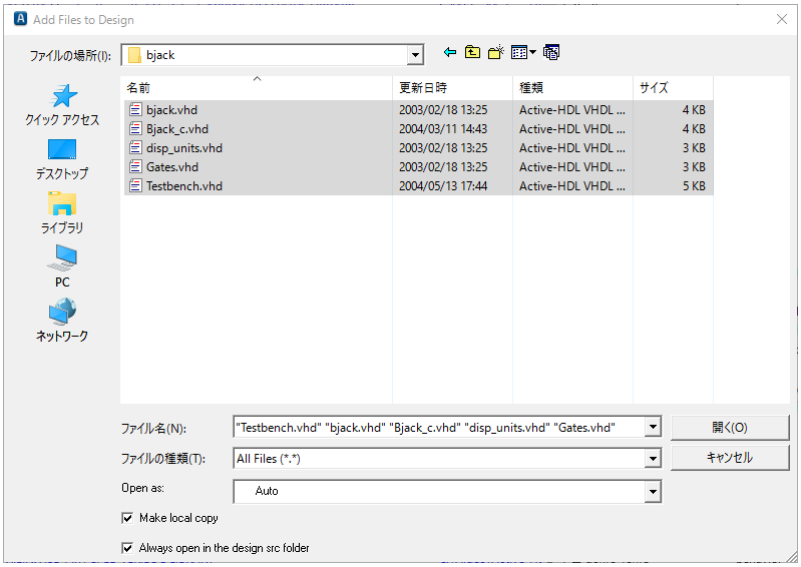


図 3-12

下図 3-13 のように Design Browser Files タブにファイルが登録されます。

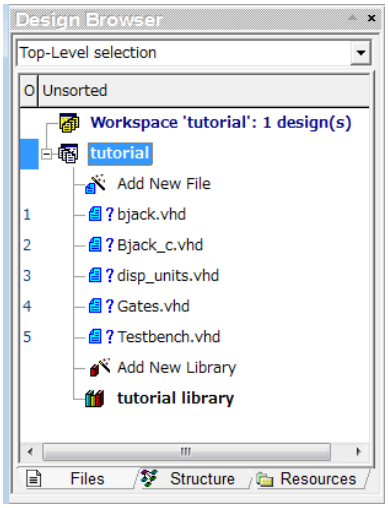


図 3-13

4　オプションの設定

カレントデザインに対するオプションの設定は Design Settings ウィンドウで行います。Design Settings ウィンドウはメニューから Design | Settings を選択して起動します。

参考）Active-HDL のツール設定は、Tools | Preferences にて行います。ここで変更した設定は、変更以降に作成されるデザインから反映され、カレントデザインには反映されません。

4.1　コンパイルの設定

言語規格や最適化のレベルを指定します。

4.1.1　Verilog コンパイルの設定

Category の Compilation | Verilog を選択した画面で行います（図 4-1）。

本チュートリアルでは Verilog コンパイルはデフォルト設定で実行します。

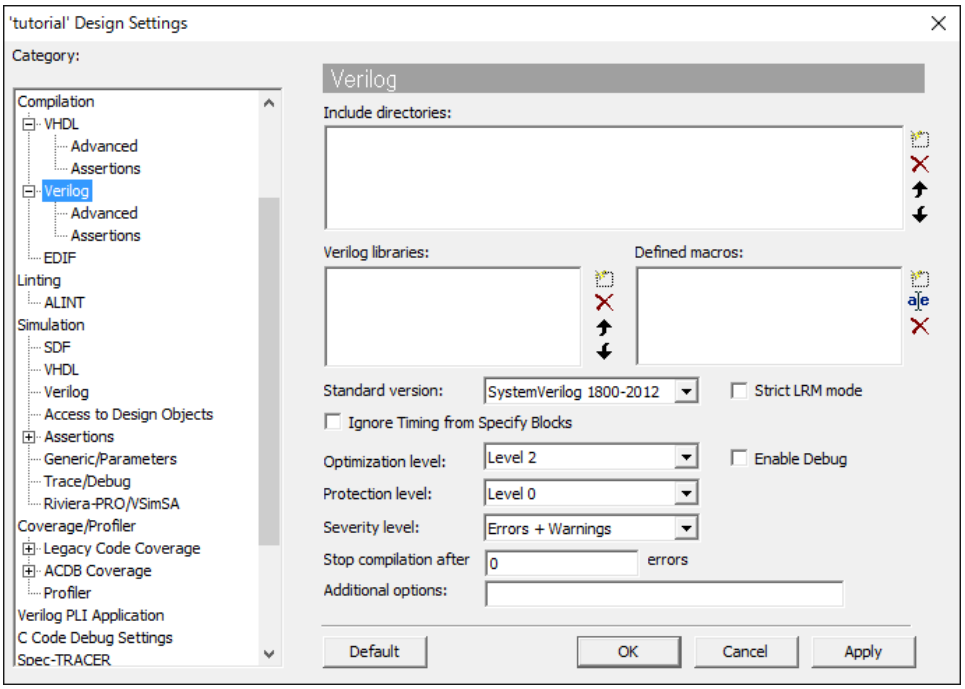


図 4-1

- Include directories　:`include コンパイラディレクティブで指定されたファイルが存在するディレクトリを指定します。デフォルトは<カレントデザイン>/src　ディレクトリです。
- Standard version　:Verilog 言語規格を指定します。Verilog 1364-1995、Verilog 1364-2001、Verilog 1364-2005、SystemVerilog 1800-200、SystemVerilog 1800-2009 または SystemVerilog 1800-2012 から選択します。
- Optimization level　:最適化のレベルを 0～3 で指定します。数字が大きいレベルほど最適化の度合いが強くなり、シミュレーション実行速度が速くなります。
- Enable Debug　:デバッグ情報を生成し、ソースコードトレース、ブレークポイント設定、カバレッジ収集、プロファイル収集やアサーション言語を含んだシミュレーションが実行できます。チェックした場合は、Optimization level の設定はできません。

上記のデザイン単位でのコンパイル設定とは別に、HDL ファイルごとにコンパイル時の設定をすることができます。Design Browser Files タブにて HDL ファイルを右クリックし、コンテキストメニューから Properties を選択し、Compile タブで必要な設定をします。

本チュートリアルでは、HDL ファイルの個別設定は行いません。

4.1.2 VHDL コンパイルの設定

Category の Compilation | VHDL を選択した画面で行います（図 4-2）。

本チュートリアルでは VHDL コンパイルはデフォルト設定で実行します。

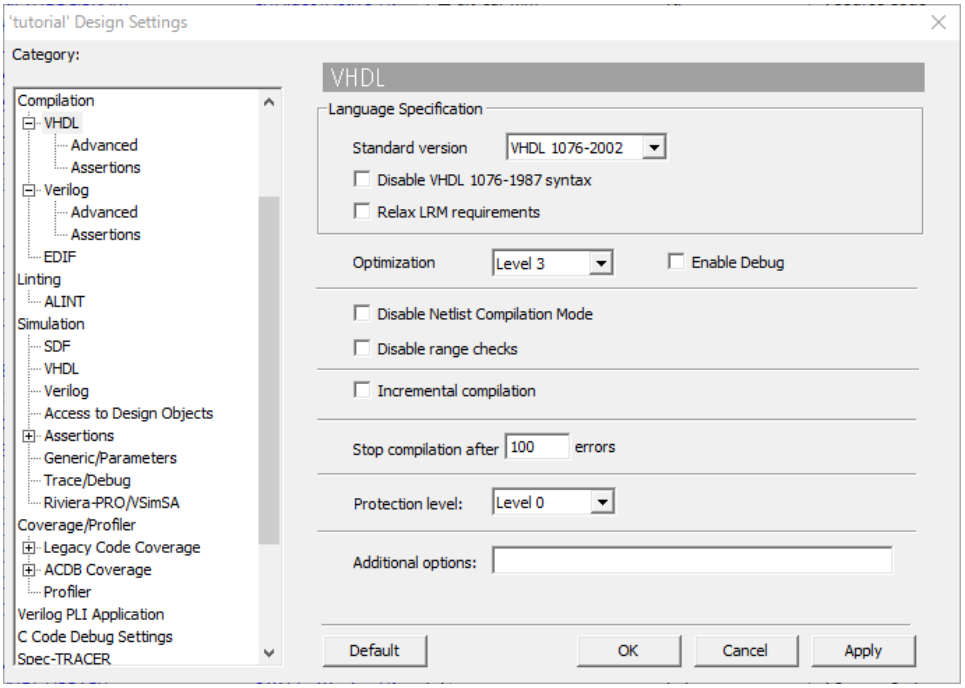


図 4-2

- Standard version ：VHDL 言語規格を指定します。VHDL 1076-1993、VHDL 1076-2002 または VHDL 1076-2008 から選択します。
- Optimization ：最適化のレベルを 0～3 で指定します。数字が大きいレベルほど最適化の度合いが強くなり、シミュレーション実行速度が速くなります。
- Enable Debug ：デバッグ情報を生成し、ソースコードトレース、ブレークポイント設定、カバレッジ収集、プロファイル収集やアサーション言語を含んだシミュレーションが実行できます。チェックした場合は、Optimization level の設定はできません。

上記のデザイン単位でのコンパイル設定とは別に、HDL ファイルごとにコンパイル時の設定をすることができます。Design Browser Files タブにて HDL ファイルを右クリックし、コンテキストメニューから Properties を選択し、Compile タブで必要な設定をします。本チュートリアルでは Testbench.vhd が 1993 準拠のため、Standard version を VHDL 1076-1993 に設定します（図 4-3）。

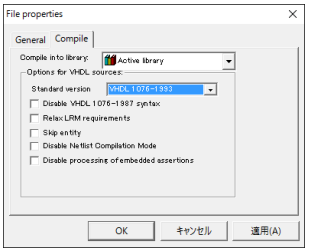


図 4-3

4.2　シミュレーションの設定

デバッグ機能に関する設定やレゾリューションを指定します（図 4-4）。

本チュートリアルでは、シミュレーションオプションはデフォルト設定で実行します。

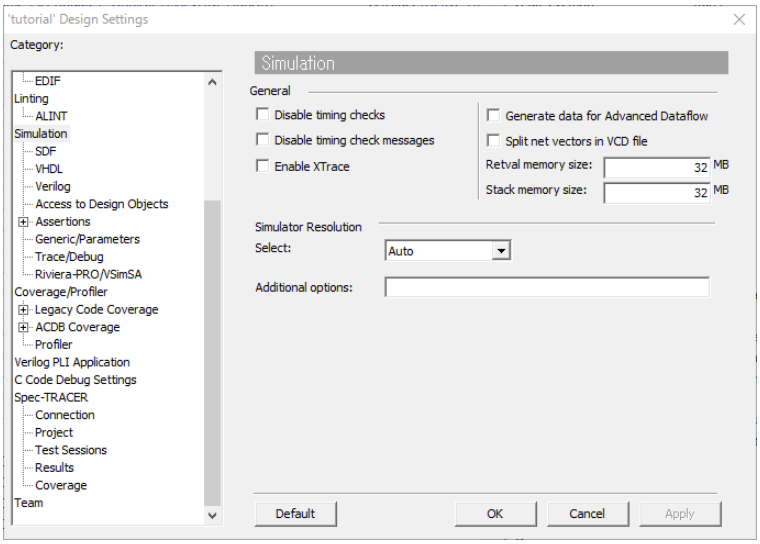


図 4-4

- Simulator Resolution　:Select 右横のプルダウンメニューでシミュレーションのレゾリューションを指定します。デフォルトは Auto で、Auto の場合 Timescale ディレクティブで指定された Verilog デザインの場合は、Timescale ディレクティブに含まれる最小のタイムユニットを使用します。

4.2.1　Verilog シミュレーションの設定

Category の Simulation | Verilog を選択した画面で行います（図 4-5）。

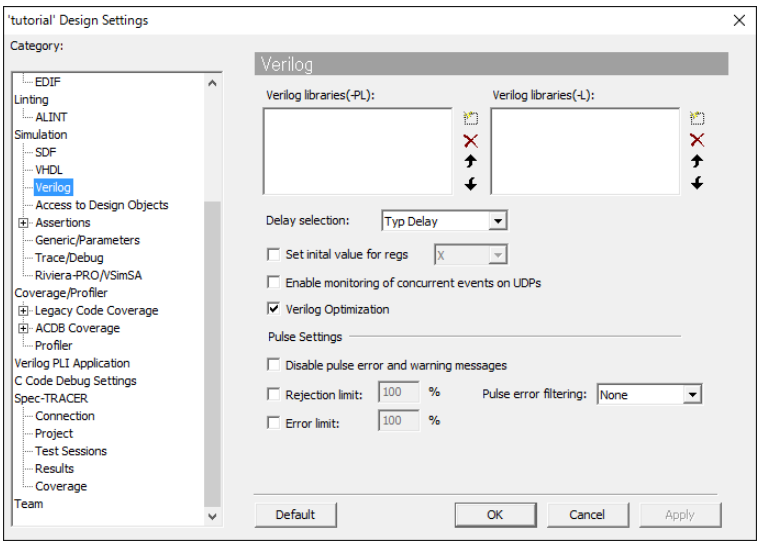


図 4-5

- Verilog libraries(-L)　:デザインが参照する FPGA ベンダライブラリ等の Verilog ライブラリを指定します。
- Verilog Optimization　:高速シミュレーション時にチェックします。デフォルトで有効となっています。

4.2.2 VHDL シミュレーションの設定

Category の Simulation | VHDL を選択した画面で行います（図 4-6）。

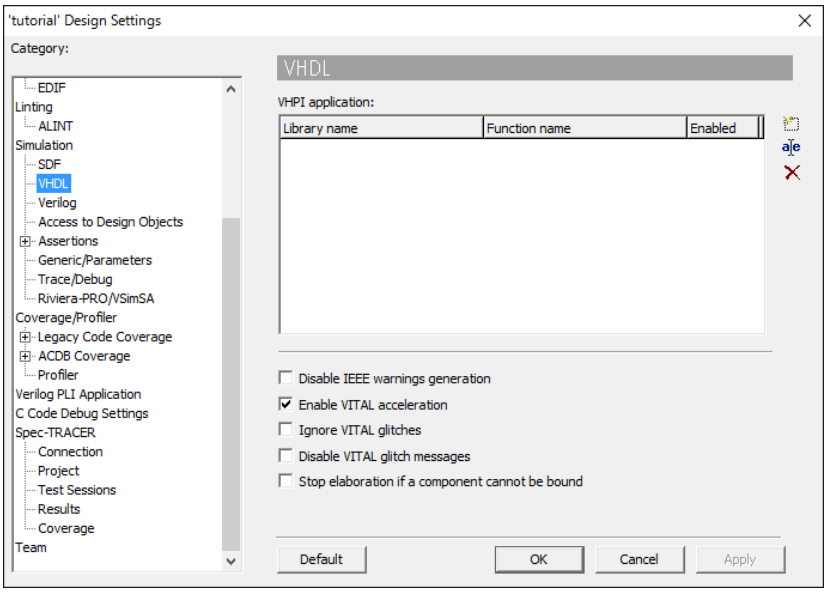


図 4-6

- VHPI application ：シミュレーションのイニシャライズ時にロードし実行させる VHPI ライブラリを指定します。
- Disable IEEE warnings generation ：IEEE ライブラリに関するワーニングのコンソール出力を無効にします。
- Stop elaboration if a component cannot be bound ：インスタンスしているコンポーネントが参照できない場合には、エラボレーションを停止します。

4.3　検証対象デザインへのアクセス設定

波形ウィンドウにて観測する信号のアクセス権を指定します。Category の Simulation | Access to Design Objects を選択した画面で行います（図 4-7）。本チュートリアルでは、全ての信号を高速表示波形データベース ASDB へ保存できるように、リードアクセスを有効にします。Limit read access to design top-level signals only のチェックをはずし、Enable Access 下の Read (+r)をチェックします。

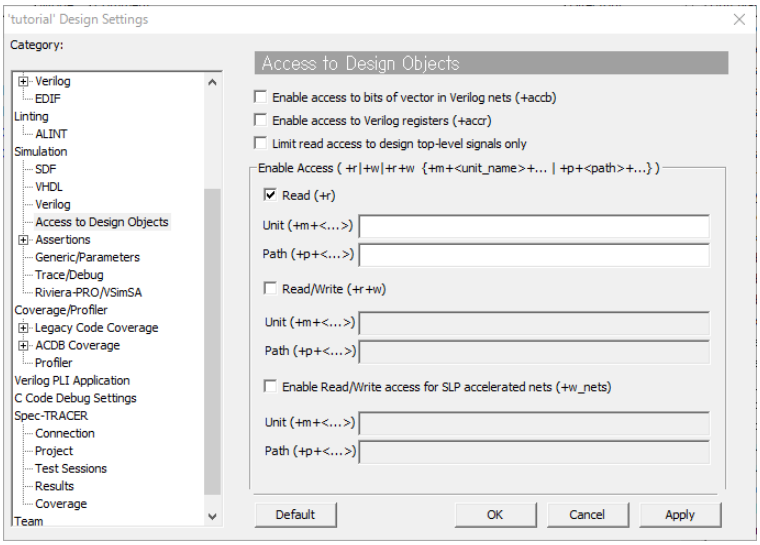


図 4-7

- Enable access to bits of vector in Verilog nets (+accb) :Verilog ベクタネットの各ビットへのアクセスを有効にします。
- Enable access to Verilog registers (+accr) ：Verilog レジスタへのアクセスを有効にします。
- Limit read access to design top-level signals only ：トップ階層の信号だけのリードアクセスを有効にします。デフォルト設定です。
- Enable access ：Limit read access to design top-level signals only のチェックをはずすと下記項目の設定が可能となります。
  - Read (+r) ：リードアクセスを有効化
  - Read/Write (+r+w) ：リード/ライトアクセスを同時に有効化
  - Enable Read/Write access for SLP accelerated nets (+w_nets) ：最適化処理をされているネットへのリード/ライトアクセスを有効化
  - 上記項目にチェックするとその下の Unit (+m+<>) および Path (+p+<>) の入力が可能となり、入力した Unit（モジュール）または Path（インスタンス）へのアクセスが有効となります。入力しない場合は、全てのモジュール/インスタンスへのアクセスが有効となります。アクセスが必要なモジュール/インスタンスを入力し、IP コアなどの必要ではないモジュール/インスタンスへのアクセスを行わないことで、シミュレーション速度が向上します。

参考）観測信号のリードアクセスが有効になっていないと、下記のようなワーニングがコンソールに出力されシミュレーション実行結果を波形ファイルに保存できません。

KERNEL: Warning: Cannot access SLP signal `UUT/HAND'. Use switch +access +r for this region.

5　ソースコードのコンパイル

5.1　Verilog の場合

デザインに登録された全ての HDL ファイルをコンパイルします。Design Browser Files タブにてデザイン tutorial を右クリックし、コンテキストメニューから Compile All を選択します(図 5-1)。

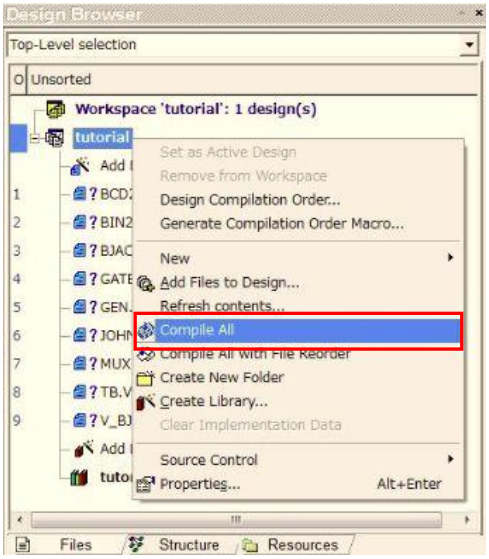


図 5-1

コンパイルの実行結果がコンソールに表示されます（図 5-2）。

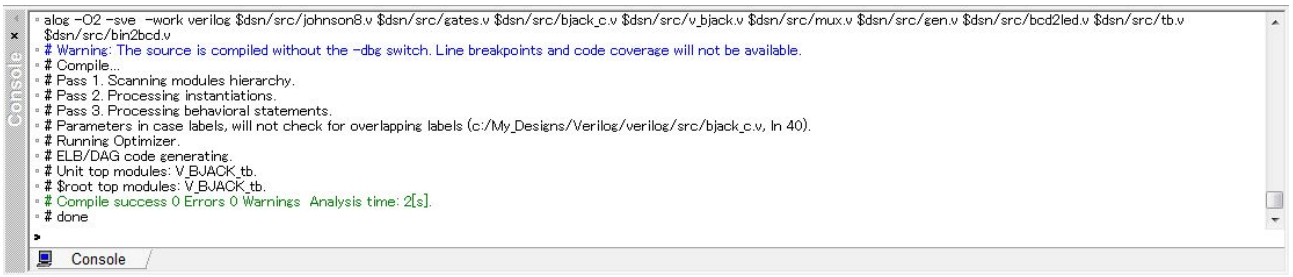


図 5-2

- # Compile success 0 Errors 0 Warnings　Analysis time: 0[s].　:コンパイルが成功し、エラーもワーニングも発生していません。

- # Warning: The source is compiled without the -dbg switch. Line breakpoints and code coverage will not be available.　:このワーニングメッセージは、「デバッグモードでコンパイルされていないため、ブレークポイントの設定やコード・カバレッジ収集が使用できない」という意味です。
  Active-HDL の設定に対して注意を促すワーニングであるため、実行上の問題はありません。

5.2 VHDL の場合

デザインに登録された全ての HDL ファイルをコンパイルします。Design Browser Files タブにてデザイン tutorial を右クリックし、コンテキストメニューから Compile All with File Reorder を選択します(図 5-3)。

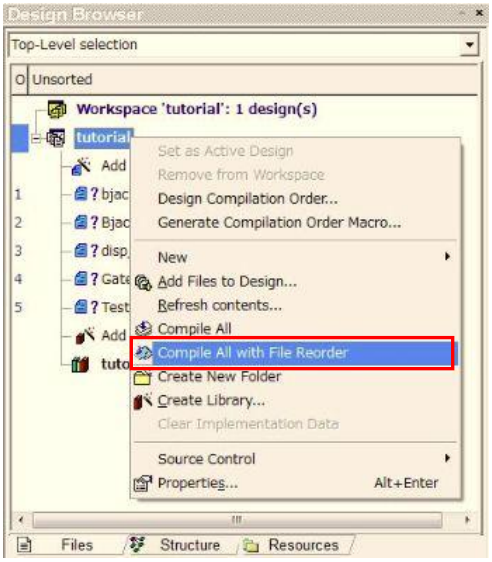


図 5-3

コンパイルは Design Browser Files タブに登録されている順番で実行されます。図 5-3 のようにファイルの順番が下位階層順に並んでいない場合は、Active-HDL が階層を判断し、コンパイル順を変えて実行するかどうか確認を促すウィンドウが表示されます(図 5-4)。はいをクリックしコンパイル順を変更して実行します。

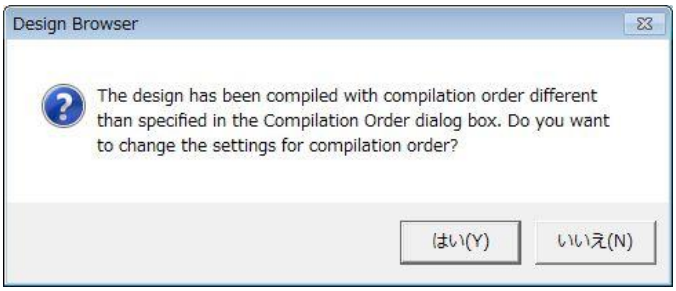


図 5-4

参考)ファイルの順番は、マウスでファイルを上下に移動することにより変更できます。

続いてトップ階層を指定する Top-level Selection ウィンドウが起動します(図 5-5)。ここでは表示されている testbench_cfg のまま OK ボタンをクリックします。

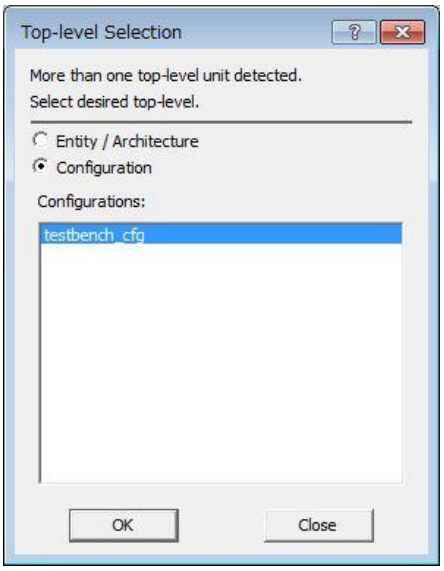


図 5-5

コンパイルの実行結果がコンソールに表示されます（図 5-6）。

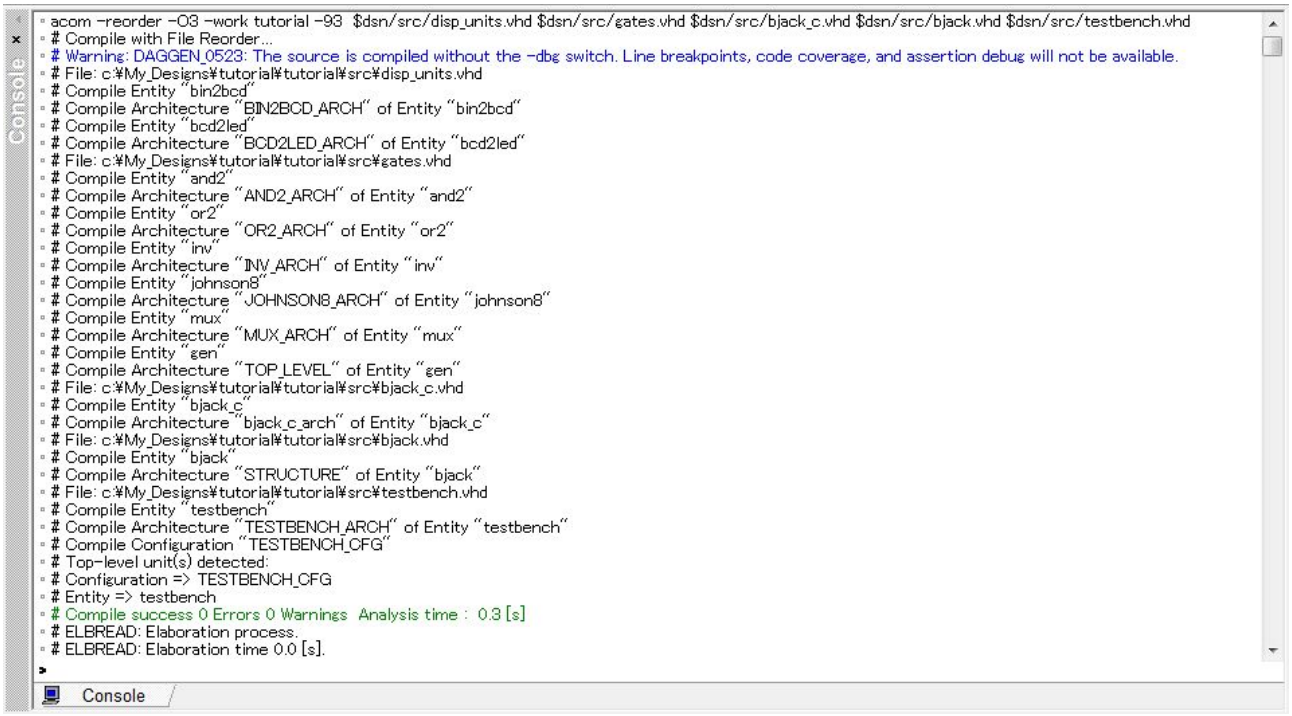


図 5-6

- # Compile success 0 Errors 0 Warnings　Analysis time: 0[s].　:コンパイルが成功し、エラーもワーニングも発生していません。

- # Warning: The source is compiled without the -dbg switch. Line breakpoints and code coverage will not be available.　:このワーニングメッセージは「デバッグモードでコンパイルされていないため、ブレークポイントの設定やコード・カバレッジ収集が使用できない」という意味です。
  Active-HDL の設定に対して注意を促すワーニングであるため、実行上の問題はありません。

5.3 Verilog/VHDL 共通

HDL ファイル右横の印は下記の状態を示します。

	コンパイルされていない、またはコンパイル後に編集された状態
	コンパイルが成功した状態
	コンパイルでワーニングが発生した状態
	コンパイルでエラーが発生した状態

図 5-7

参考）他のコンパイル方法

上述のコンテキストメニューから Compile All や Compile All with File Reorder のほかに、HDL ファイル単体で Compile も選択できます。それ以外にも下記の方法でコンパイルが実行できます。

操作	アクション
②  ボタン ② Design | Compile ③ F11	Design Browser Files タブで選択したファイル、またはドキュメントウィンドウに表示されたファイルをコンパイルします。
①  ボタン ② Design | Compile All	Design Browser Files タブに表示されたソースファイルを、ファイルの左側に表示された順番でコンパイルします。
②  ボタン ② Design | Compile All with File Reorder	VHDL ファイルの場合、Design Browser Files タブに表示された順番に関わらず Active-HDL がファイル関係を判別し、下位階層順に並べ替えてコンパイルを実行します。 Verilog ファイルの場合は、Compile All と同じです。

図 5-8

6　イニシャライズシミュレーション

下記の手順でイニシャライズシミュレーションを実行し、シミュレーション対象のデザインユニットをローディングします。

6.1　Verilog の場合

6.1.1　トップレベルの指定

ワーキングライブラリを展開し、トップレベルに指定するデザインユニットを右クリックし、コンテキストメニューから Set as Top-Level を選択します（図 6-1）。ここでは V_BJACK_tb を指定します。

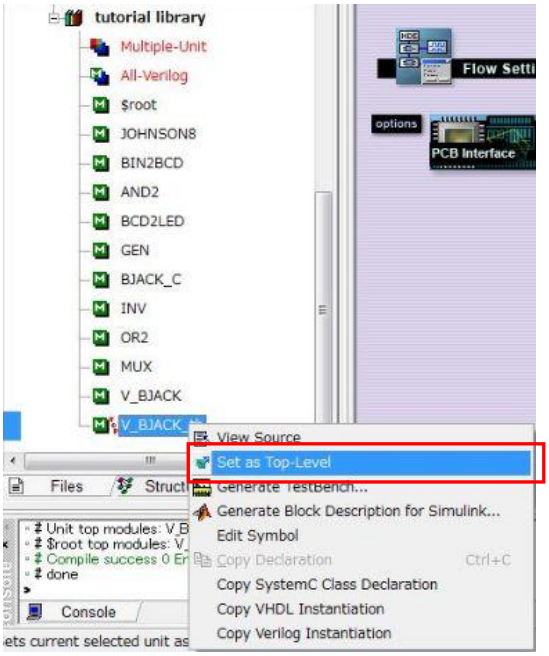


図 6-1

6.1.2　イニシャライズの実行

メニューから Simulation | Initialize Simulation を選択します（図 6-2）。

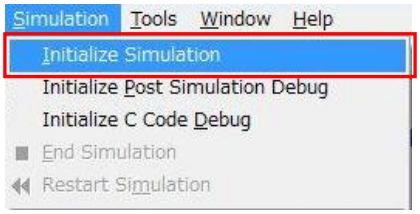
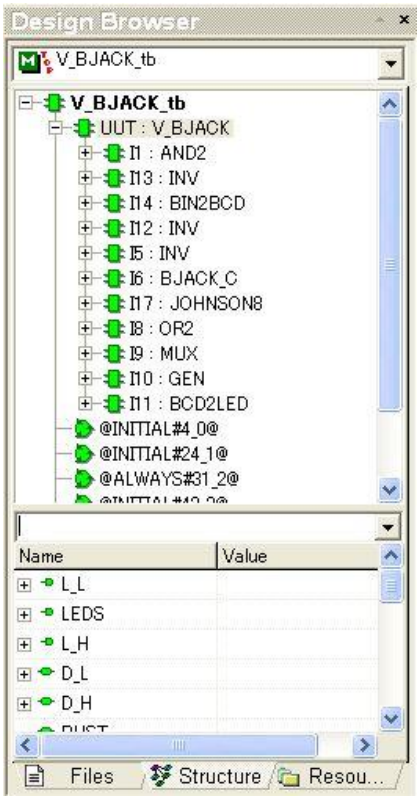


図 6-2

イニシャライズシミュレーションを実行すると、Design Browser が Structure タブへ自動的に切り替わります（図 6-3）。



トップレベルの指定、表示

検証対象デザインの階層表示

インスタンス　:インスタンス・モジュールを表現

プロセス　:モジュール内の Initial　文、Always　文、信号代入文を表現

信号検索ウィンドウ

選択した階層内の信号リスト

図 6-3

実行結果がコンソールに出力されます。ワーニングやエラーが出力されているかどうか確認します。ここでは、ワーニングもエラーも出力されていません（図 6-4）。

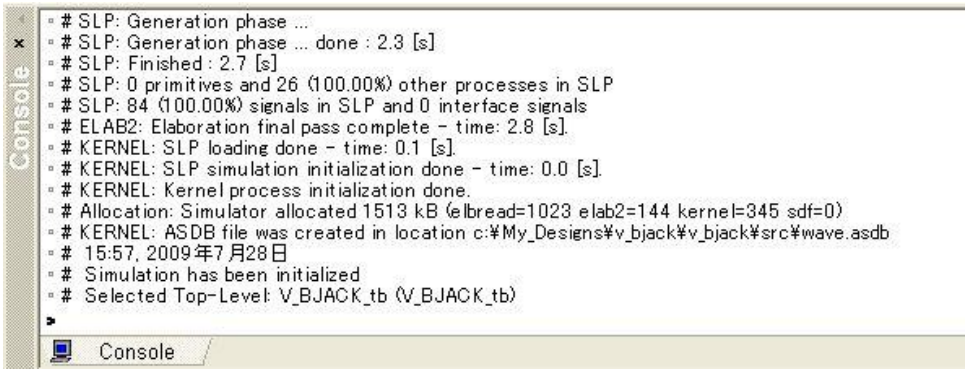


図 6-4

6.2　VHDL の場合

6.2.1　イニシャライズの実行

メニューから Simulation | Initialize Simulation を選択します（図 6-5）。

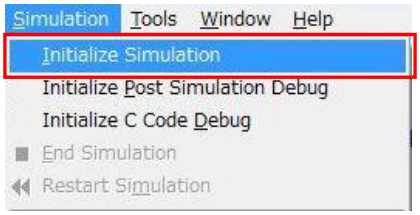
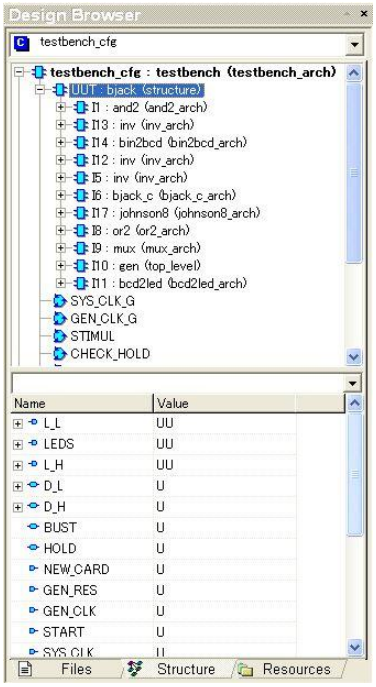


図 6-5

イニシャライズシミュレーションを実行すると、Design Browser が Structure タブへ自動的に切り替わります（図 6-6）。



トップレベルの指定、表示

検証対象デザインの階層表示

インスタンス　:インスタンス・コンポーネントを表現

プロセス　:Process　文、Procedure Call、信号代入文を表現

信号検索ウィンドウ

選択した階層の信号リスト

図 6-6

実行結果がコンソールに出力されます。ワーニングやエラーが出力されているかどうか確認します。ここでは、ワーニングもエラーも出力されていません（図 6-7）。

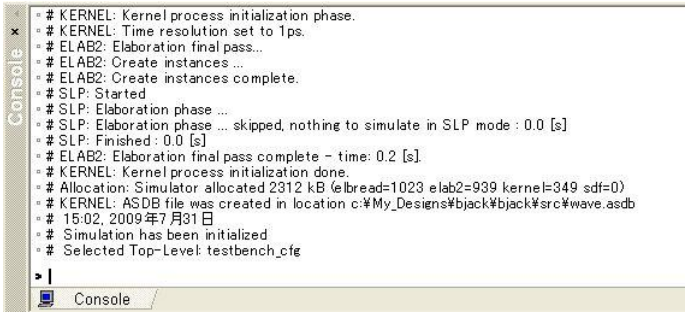


図 6-7

7　観測信号の指定

波形ファイルへの観測信号の指定は下記の手順で行います。指定した観測信号は ASDB データベースに保存されます。

7.1　波形ウィンドウの起動

ツールバー上の  ボタン（図 7-1 赤丸）をクリックし、波形ウィンドウを起動します。

7.1.1　ドラッグ&ドロップによる信号の追加

Design Browser Structure タブの階層表示のインスタンスや、下のウィンドウに表示されている信号リストから、波形ウィンドウへドラッグ&ドロップにより観測信号を追加します。

【Verilog の場合】

ここでは、V_BJACK_tb および UUT 階層の信号を観測信号に指定します(図 7-1)。

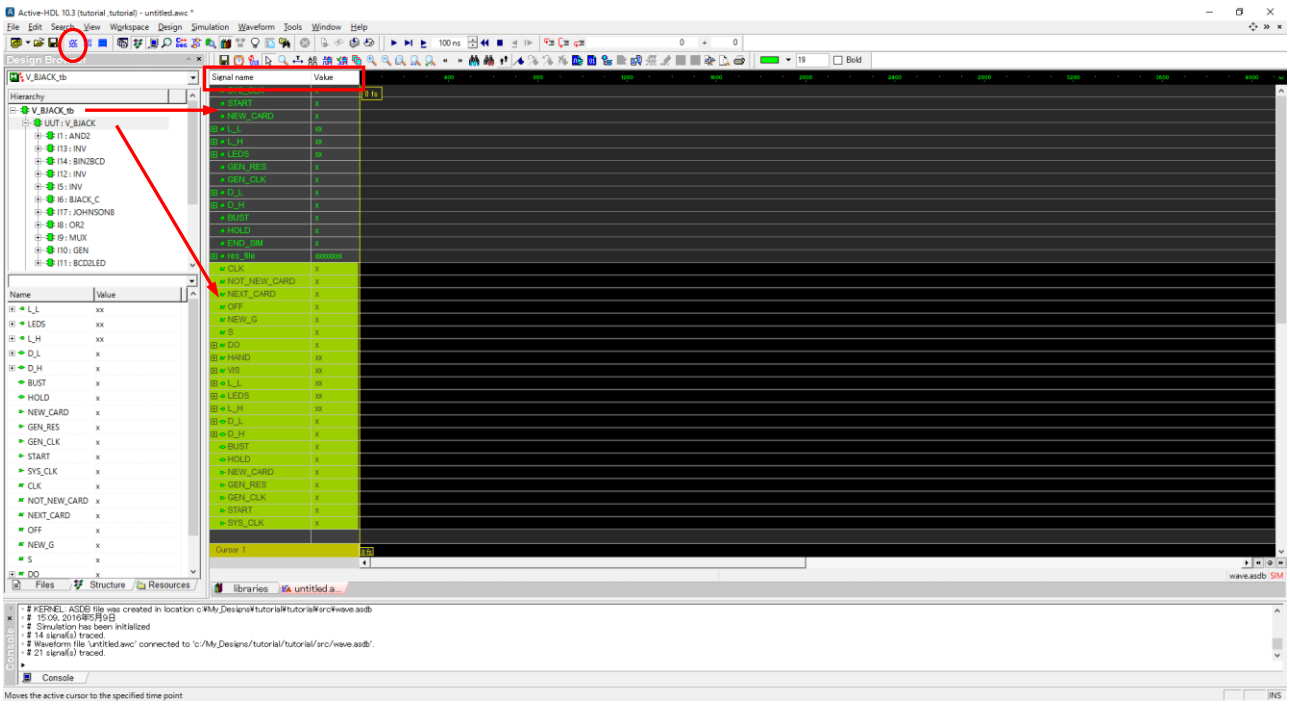


図 7-1

【VHDL の場合】

testbench_cfg および UUT 階層の信号を観測信号に指定します(図 7-2)。

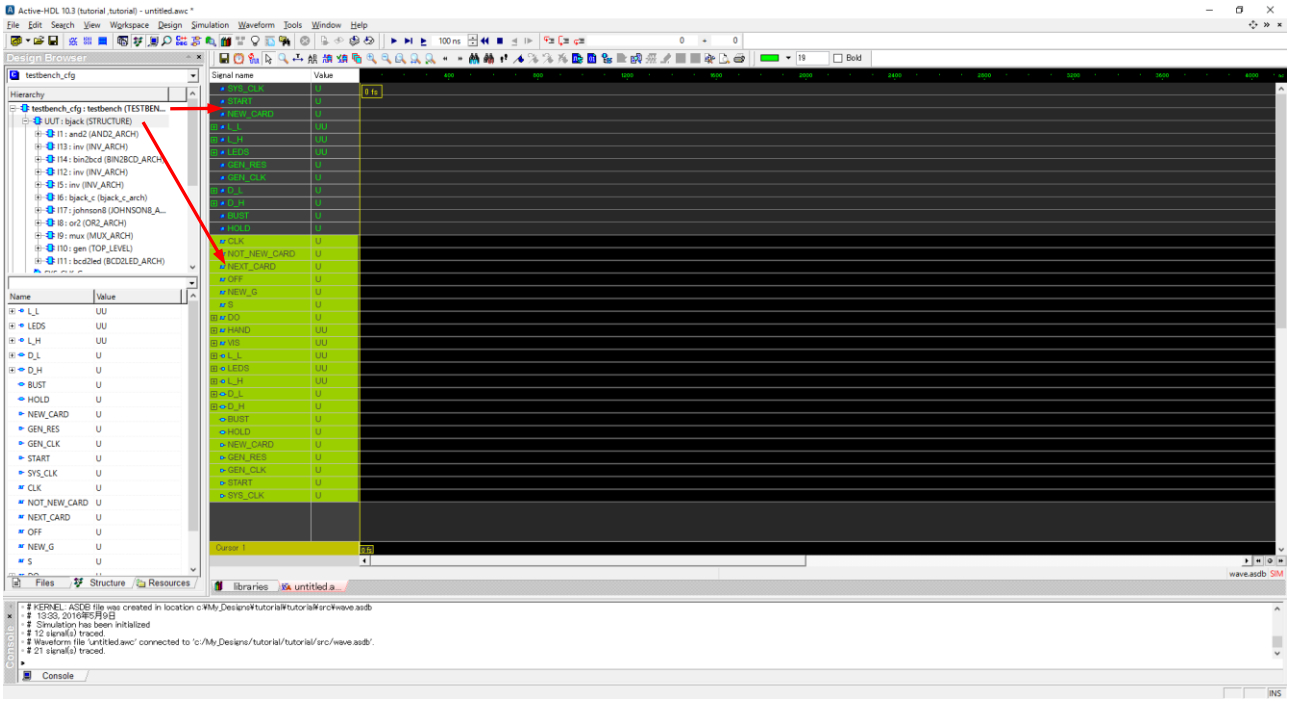


図 7-2

【Verilog/VHDL 共通】

参考）波形ウィンドウの Signal name には階層は表示されません。階層を表示するには図 7-1 の赤枠を右クリックし、コンテキストメニューから Hierarchy を選択すると階層が表示されます（図 7-3）。

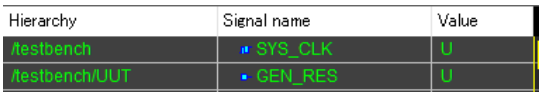


図 7-3

参考）Design Browser Structure タブのインスタンスまたは信号リストを右クリックし、コンテキストメニューの Add to Waveform または Add to Waveform Recursively を選択して、波形ウィンドウへ信号を追加できます。

- Add to Waveform ：選択したインスタンスの信号を追加します。
- Add to Waveform Recursively ：選択したインスタンスの信号とその下位階層のすべての信号を追加します。

全ての信号を指定する場合は、最上位階層から Add to Waveform Recursively を選択します（図 7-4）。

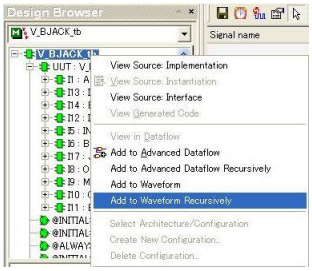


図 7-4

8　シミュレーションの実行

ツールボタン ▶ をクリックし、シミュレーションを実行します。シミュレーションが終了すると図 8-1 のウィンドウが表示されますので、OK ボタンをクリックします。

図 8-1

参考）他のシミュレーション実行方法は下記の通りです。

操作	アクション
② ▶ ボタン ② Simulation | Run ③ Alt+F5	シミュレーションイベントがなくなるまで、シミュレーションを進めます。
① ▶I ボタン ② Simulation | Run Until	Run Until ウィンドウが起動（図 8-3）。ボックスに指定した絶対時間までシミュレーションを進めます。
② ▶ ボタン ② Simulation | Run For ③ F5	右横のボックスに指定した時間だけシミュレーションを進めます。

図 8-2

図 8-3

9　波形データベースと信号リストの保存

9.1　波形データベースの保存

デフォルトでは波形データベースは wave.asdb という名前で、デザイン・ディレクトリ内の src ディレクトリに保存されます。ツールボタン  をクリックして表示される Save ウィンドウにて、図 9-1 のように波形名に first を指定し、Save ボタンをクリックします。Design Brower Files タブに保存した波形ファイルが追加されます。

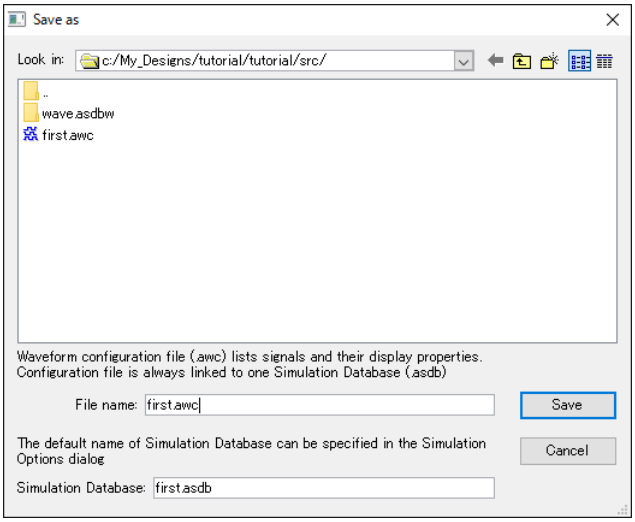


図 9-1

- Look in　：波形データベースの保存先を表示・指定します。
- File name：AWC ファイル名を指定します。ここでは first.awc とします。
- Simulation Database　：データベース名を指定します。ここでは first.asdb とします。

参考）ASDB と AWC について

ASDB（Aldec Simulation DataBase）には、指定した観測信号全ての情報が記録されます。ASDB から全てまたは一部の信号を AWC（Aldec Waveform Configuration）に選択・保存することができます。1つの ASDB から複数の AWC ファイルを作成できます。

注）wave.asdb はイニシャライズシミュレーションを実行するたびに上書きされます。前回のデータベースを保存するには、wave.asdb 以外の名前をつけて保存してください。

9.2　信号リストの保存

波形ウィンドウに表示されている信号を観測信号として再利用するには、波形マクロを生成します。first.awc に表示されている信号を下記の手順でファイルに保存します。first.awc を表示させた状態で、メニューから Waveform | Save to Macro を選択し、表示された Choose a filename to save ウィンドウにて、任意のファイル名を入力し保存ボタンをクリックします。ここでは signals と入力します。

Design Browser Files タブに生成されたマクロファイル signals.do が追加されます。このファイルをダブルクリックするとドキュメントウィンドウに内容が表示されます（図 9-2）。

```
onerror { resume }
transcript off
add wave -noreg -logic {/testbench/SYS_CLK}
add wave -noreg -logic {/testbench/START}
add wave -noreg -logic {/testbench/NEW_CARD}
add wave -noreg -hexadecimal -literal {/testbench/L_L}
add wave -noreg -hexadecimal -literal {/testbench/L_H}
:
:
```

図 9-2 (Verilog/VHDL 共通)

9.3 1 回目シミュレーションの終了

シミュレーションを停止するには、End Simulation アイコン  をクリックするか、メニューから Simulation | End Simulation を選択します。

10　デザインの修正とシミュレーションの再実行

仕様変更または不具合を見つけた場合には、ソースコードを修正し再度シミュレーションを実行します。

10.1　波形によるデバッグ

出力信号の値やイベントのタイミングが、期待通りであるかどうか確認します。

【Verilog の場合】

シミュレーション波形 first.awc にて、トップレベルユニットの出力信号 L_H を選択し、カーソルを 1.76ns にあわせ、値が 01 から 79 に変化していることを確認します（図 10-1）。

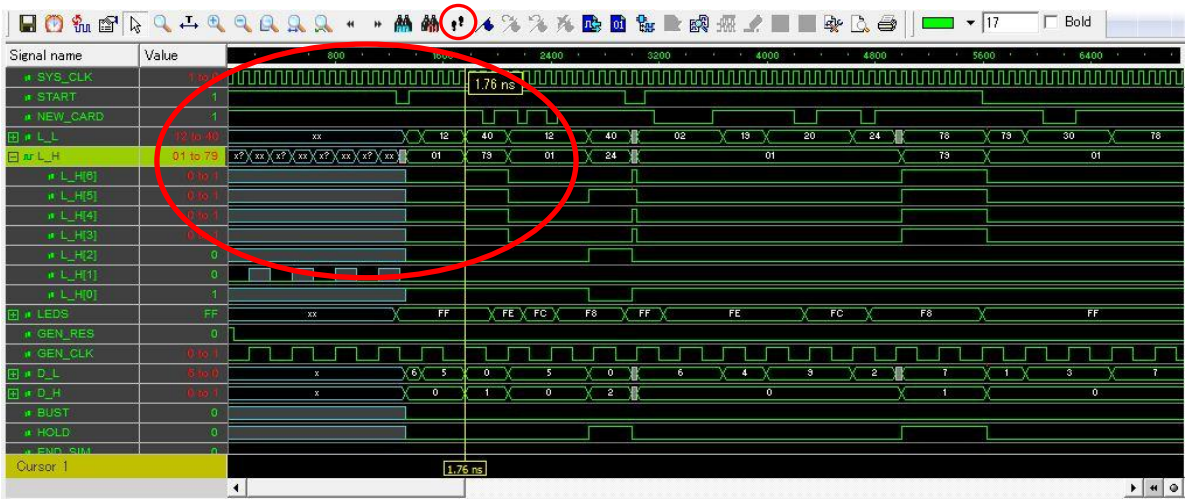


図 10-1　(Verilog)

【VHDL の場合】

シミュレーション波形 first.awc にて、トップレベルユニットの出力信号 L_H を選択し、カーソルを 1.04us にあわせ、値が 01 から 4F に変化していることを確認します（図 10-2）。

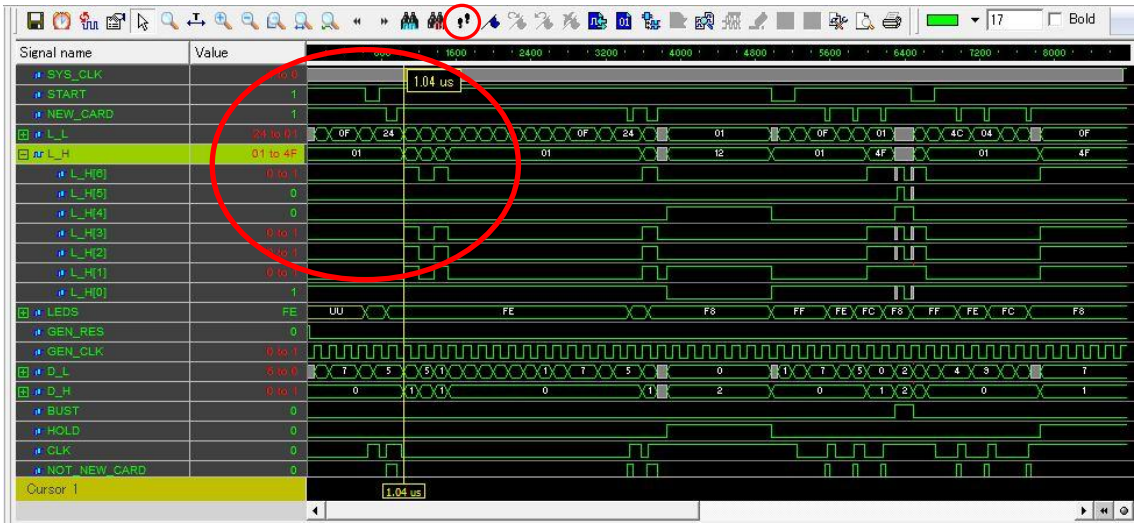


図 10-2　(VHDL)

10.2　ソースコードの修正

BCD 変換を通してバイナリデータを 7 セグ LED に変換表示するユニットである bcd2led.v / disp_units.vhd において、BCD から 7 セグ LED 表示変換の記述の一部に誤りがあったと仮定して、次のように修正します。
修正対象ファイルをダブルクリックすると HDL エディタが開きます。オリジナルの記述をコピーして修正する際に、comment　uncomment アイコンを利用すると便利です。

【Verilog の場合】

bcd2led.v　34　行目:

```
//assign H_TMP = (DIGIT_H == 2'b01) ? 7'b1111001 :                  // 1
//                              (DIGIT_H == 2'b10) ? 7'b0100100 :   // 2
//                              (DIGIT_H == 2'b11) ? 7'b0110000 :   // 3
//                              7'b0000001;                         // 0
assign H_TMP = (DIGIT_H == 2'b10) ? 7'b1111001 :                    // 1
                              (DIGIT_H == 2'b01) ? 7'b0100100 :     // 2
                              (DIGIT_H == 2'b00) ? 7'b0110000 :     // 3
                              7'b0000001;                           // 0
```

【VHDL の場合】

disp_units.vhd　76　行目:

```
--H_TMP<=          "1001111" when (DIGIT_H="01") else    --1
--                 "0010010" when (DIGIT_H="10") else    --2
--                 "0000110" when (DIGIT_H="11") else    --3
--                 "0000001";        --0
H_TMP<="1001111" when (DIGIT_H="10") else    --1
                   "0010010" when (DIGIT_H="01") else    --2
                   "0000110" when (DIGIT_H="00") else    --3
                   "0000001";        --0
```

10.3　修正したデザインの再コンパイルとシミュレーションの再実行

bcd2led.v / disp_units.vhd を save ボタン で保存します。修正したファイルは、コンパイルが必要なことを示すステータス？がデザインファイルの右横についています。コンテキストメニューからコンパイルを実行します。

1 回目のシミュレーション波形ウィンドウの右上の をクリックして、first.awc 波形を閉じます（図 10-3）。

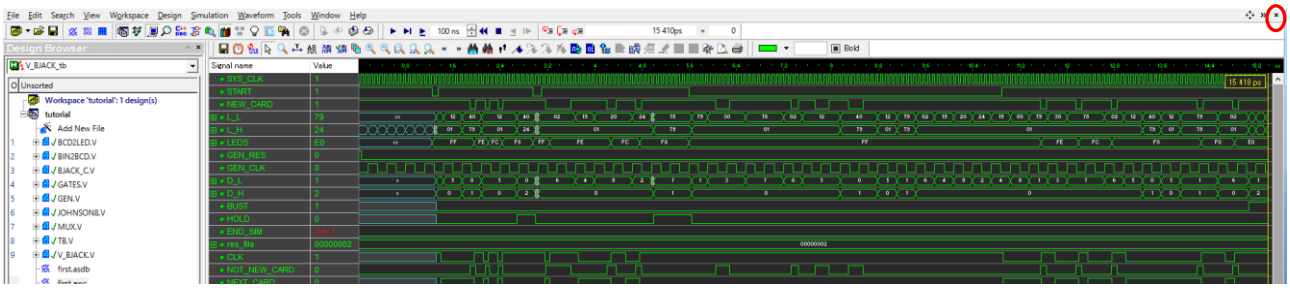


図 10-3

メニューの Simulation | Restart Simulation を選択し、イニシャライズシミュレーションを実行します。

1 回目のシミュレーションで観測した信号リストを再利用するため、Design Browser Files タブにて signals.do を選択し、コンテキストメニューの Execute をクリックします。続いて Run（run -all）を実行すると、デザインの修正内容が反映されたシミュレーション結果が表示されます。2 回目のシミュレーションの波形名を second として保存します。

【Verilog の場合】

second.awc にて、トップレベルユニットの出力信号 L_H を選択し、カーソルを 1.76ns にあわせ、値が 30 から 24 に変化していることを確認します（図 10-3）。

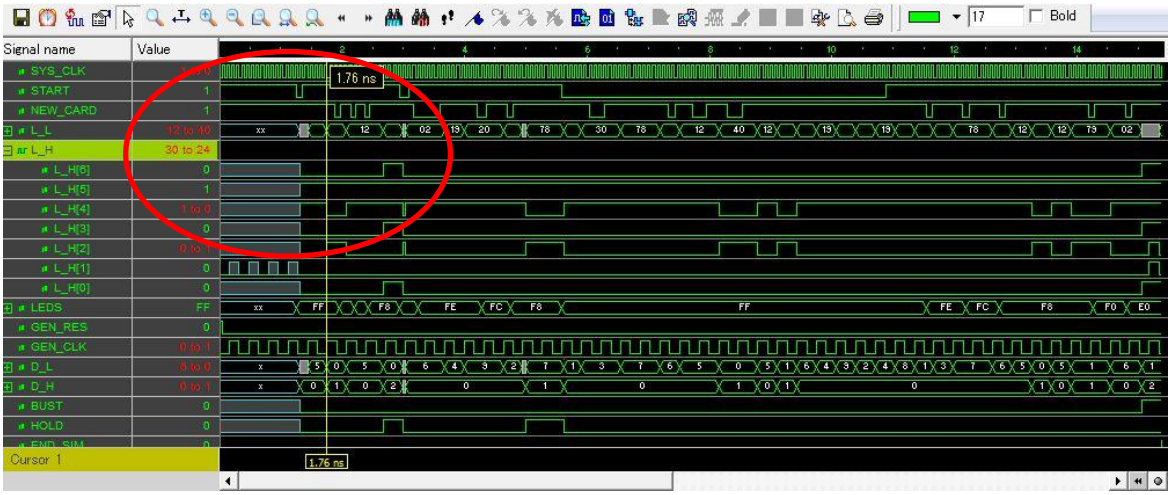


図 10-3　(Verilog)

【VHDL の場合】

second.awc にて、トップレベルユニットの出力信号 L_H を選択し、カーソルを 1.04us にあわせ、値 06 から 12 に変化していることを確認します（図 10-4）。

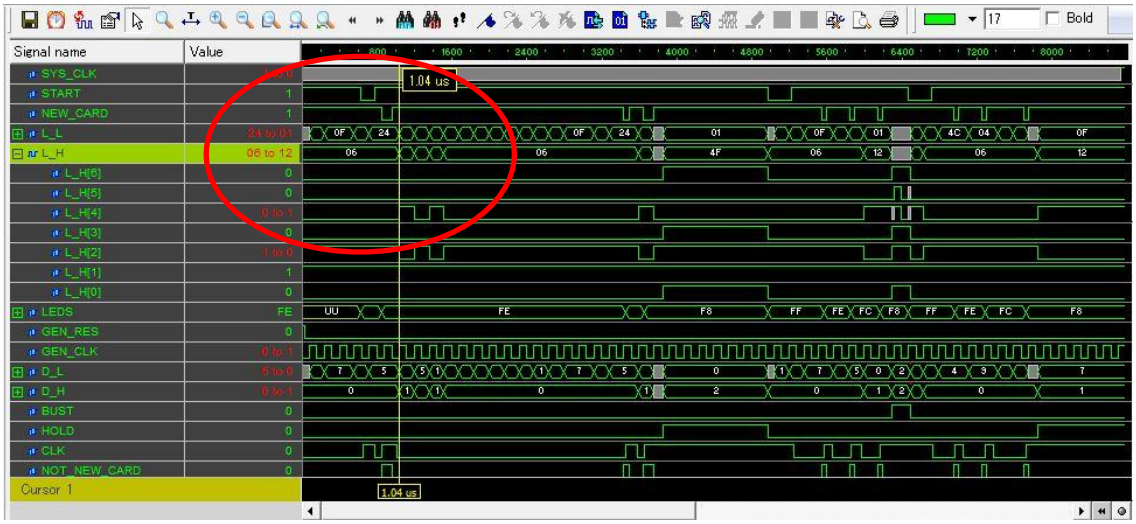


図 10-4 (VHDL)

10.4 波形の比較

second.awc を開いた状態でメニューの Waveform | Compare Waveforms を選択し、デザインファイルの修正前後の波形 first.asdb と second.asdb を比較します。Compare waveforms ウィンドウにて、比較対象である波形 first.asdb を選択して OK をクリックします (図 10-5)。

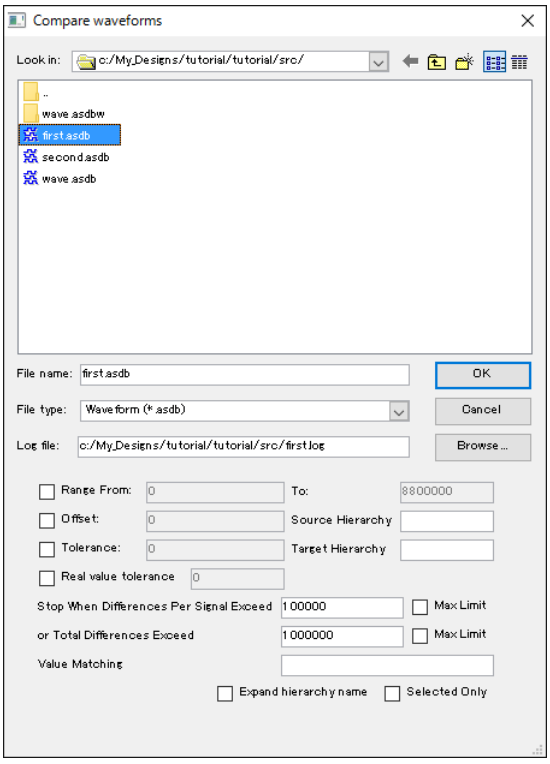


図 10-5

同名信号の値が異なる場合には、異なる個所が青色でマークされます。デザインの修正前後で、信号 L_H の値が変わったことが確認できます(図 10-6/10-7)。

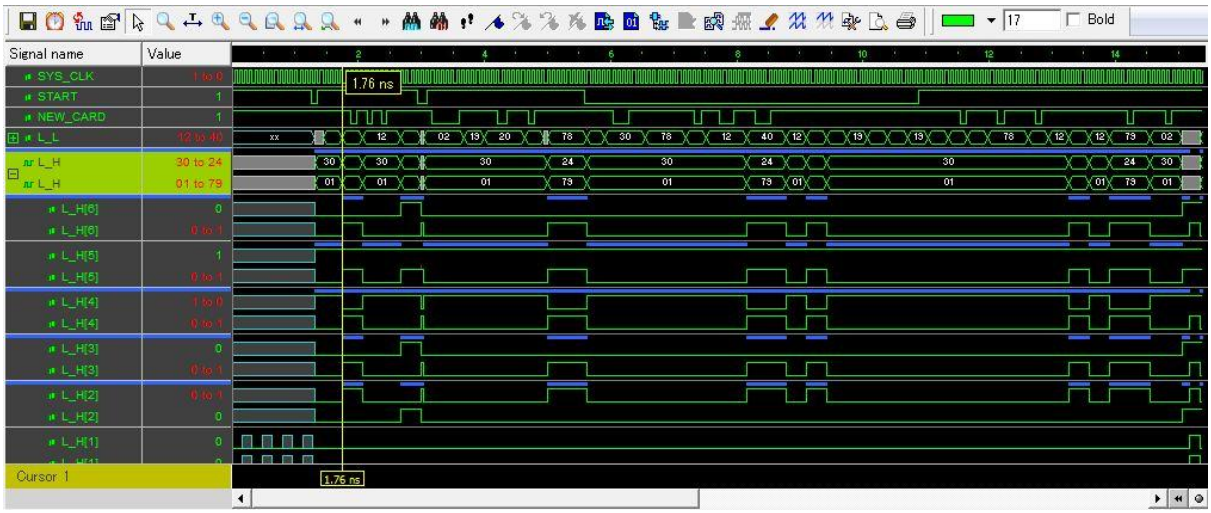


図 10-6 (Verilog)

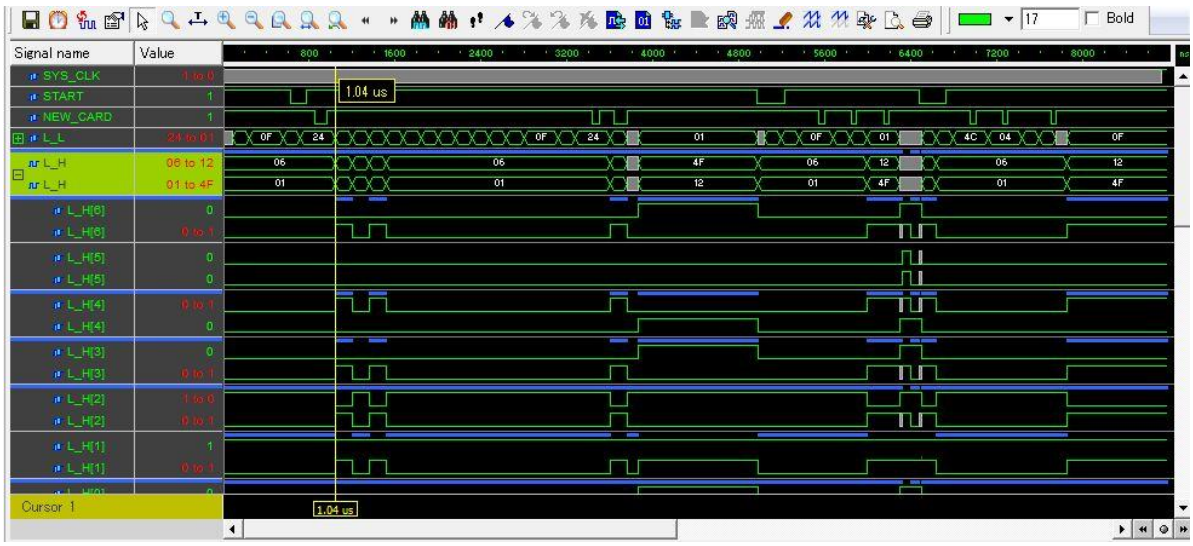


図 10-7 (VHDL)

メニューの Waveform | Remove Difference Marks を選択し、波形の差分表示を解除します。

10.5 2 回目シミュレーションの終了

End Simulation アイコン ■ をクリックして、2 回目のシミュレーションを終了します。

11 コンパイル・シミュレーション実行マクロの生成

同じデザインのシミュレーションを繰り返し行う場合には、GUI で実行したコンパイル・シミュレーション操作をマクロファイルに保存して、次回のシミュレーションで使用すると便利です。

11.1 カレントデザインのコンパイル・シミュレーションマクロの生成

メニューから Design | Generate Macro を選択し、表示された Generate Macro ウィンドウにて、マクロ名 run_sim を入力し、Macro type として Generate simulation macro based on compilation order を選んで、Generate ボタンをクリックします(図 11-1)。

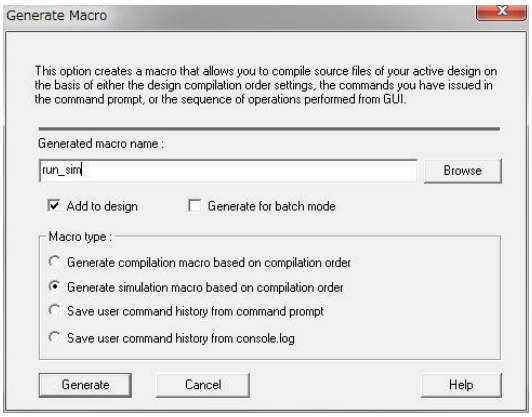


図 11-1

Design Browser Files タブに、生成されたマクロファイル run_sim.do が追加されます。このファイルをダブルクリックして、マクロファイルの内容を修正します。コンパイとイニシャライズシミュレーション用のコマンドが記載されているので、波形上に観測信号を追加しシミュレーションを実行するコマンドを追加します。波形への観測信号の追加は 9.2 で生成した波形マクロを使用します（図 11-2/11-3）。

```
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/bcd2led.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/bin2bcd.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/bjack_c.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/gates.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/gen.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/johnson8.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/mux.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/tb.v"
alog -O2 -protect 0 -msg 5 -sve "$dsn/src/v_bjack.v"
asim -stack 32 -retval 32 -O5 +access +r V_BJACK_tb
do signals.do
run –all
```

図 11-2（Verilog）

```
acom -2002 -O3 -e 100 -protect 0 -reorder "$dsn/src/bjack_c.vhd"
acom -2002 -O3 -e 100 -protect 0 -reorder "$dsn/src/disp_units.vhd"
acom -2002 -O3 -e 100 -protect 0 -reorder "$dsn/src/gates.vhd"
acom -2002 -O3 -e 100 -protect 0 -reorder "$dsn/src/bjack.vhd"
acom -93 -O3 -e 100 -protect 0 -reorder "$dsn/src/testbench.vhd"
asim -stack 32 -retval 32 -O5 +access +r testbench_cfg
do signals.do
run -all
```

図 11-3（VHDL）

11.2　マクロ実行

11.2 で追加修正した run_sim.do ファイルを保存後、Execute アイコン  をクリックして実行します。
コンパイル・イニシャライズシミュレーションに続いて、波形に観測信号が追加され、シミュレーションが実行されます。
End Simulation でシミュレーションを終了します。

12　Active-HDL の終了

右上の  ボタンをクリックするか、メニューから File | Exit を選択し、Active-HDL を終了します。