

第 10 章 論理シミュレーション

10.1 Diamond 環境での論理シミュレーション

Lattice Diamond 3.12 付属の ModelSim Lattice Edition” (以下 ModelSim LE) による論理シミュレーションの実行には三通りの方法があります。それぞれには長短があります。

1. Simulation Wizard GUI ツールによって ModelSim LE 方法 (第 10.2 節)
2. ModelSim LE を単独起動して、主にマウス操作によって行う方法 (第 10.3 節)
3. ModelSim LE を単独起動して、ユーザーが記述・作成する ”DO マクロ (TCL スクリプト) ” による方法 (第 10.4 節)

以下、各節でそれぞれの手順について記述します。

10.2 Simulation Wizard によるシミュレーション実行

10.2.1 Simulation Wizard の概要

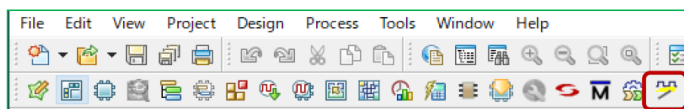
”Simulation Wizard” は Lattice Diamond から ModelSim LE を呼び出して、HDL ソースのコンパイルからシミュレーション実行までを簡易的に自動実行する GUI ツールです。ModelSim LE を単独起動して GUI 操作によるシミュレーション実行 (第 10.3 節) と、基本的には等価な機能ですが、ユーザー操作が極小化されています。

また、Diamond 3.11 まで付属している Active-HDL についても Simulation Wizard は同様な手順で実行できますので、本節記述をご参考ください。

10.2.2 Simulation Wizard の起動と操作ステップ

Simulation Wizard は、ツールバー上のアイコン  をクリックするか (図 10-1)、メニューバーから [Tools] → [Simulation Wizard] を選択して起動します。

図 10-1. Simulation Wizard の起動アイコン



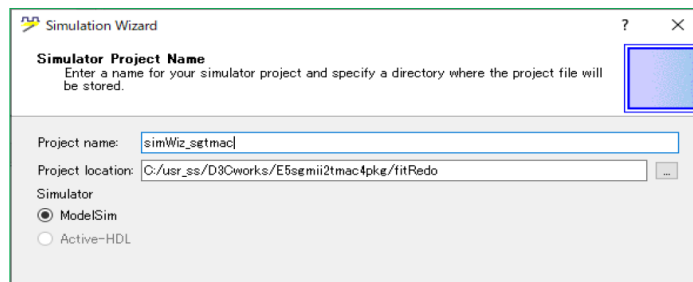
Simulation Wizard を起動すると、最初に ”Preparing the Simulator Interface” (Simulation Wizard の用途についてのメッセージ) が表示されます。設定項目はないので、ウィンドウ右下の 『Next>』 ボタンをクリックし次へ進みます。

次に表示されるウィンドウ (図 10-2) で、適切なシミュレーション・プロジェクト名を入力し、フォルダパスを指定後に、右下の 『Next>』 ボタンをクリックします (図 10-2 では非表示)。作業フォルダを作成するかどうかを確認するメッセージが出ますが、『Yes』 をクリックして先に進みます。

なお、図 10-2 におけるシミュレーターの選択について、Diamond 3.12 では常に ModelSim (LE) がデフォルトで選択されます。グレーアウトされている Active-HDL をイネーブルして選択できるようにするためには、Active-HDL をインストール後に Diamond 環境変数 Directories を設定しておきます (第 23.3.4 項参照)。

註: 本 Lattice Diamond 日本語マニュアルは、日本語による理解のため一助として提供しています。作成にあたっては各トピックについて可能な限り正確を期しておりますが、必ずしも網羅的あるいは最新でない可能性や、オリジナル英語版オンラインヘルプや各種ドキュメントと不一致がある可能性があり得ます。疑義が生じた場合は技術サポート担当者にお問い合わせ頂くか、または最新の英語オリジナル・ソースを参照するようお願い致します。

図 10-2. シミュレーション・プロジェクトの設定 1



次はシミュレーション種別の選択です (図 10-3)。選択肢とシミュレーション内容は表 10-1 の通りです。

図 10-3. シミュレーション種別の選択

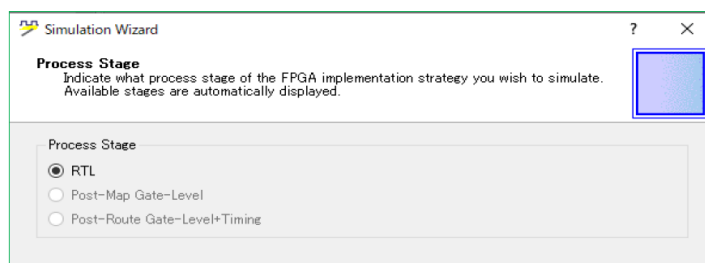
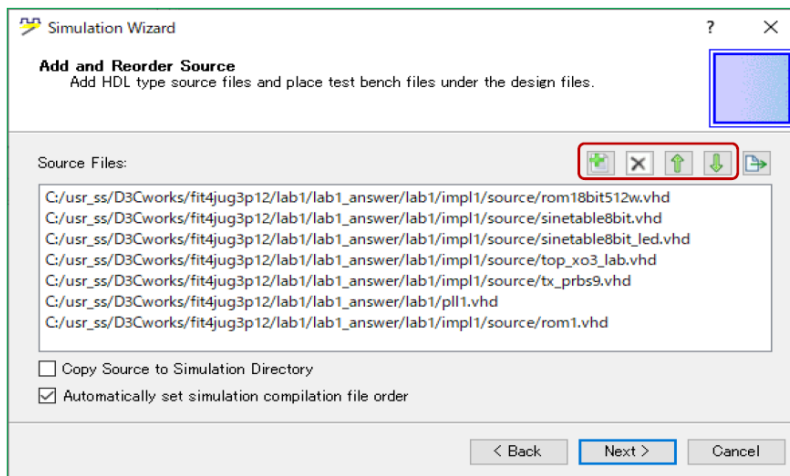


表 10-1. Process Stage とシミュレーション内容

Process Stage	シミュレーション対象	遅延情報 (sdf)
RTL	RTL ソースファイル	なし
Post-Map Gate-Level	マッピング後のネットリスト	なし
Post-Route Gate-Level+Timing	PAR 後ネットリスト (タイミングシミュレーション)	あり


”Post-Map Gate-Level” と ”Post-Route Gate-Level+Timing” は、シミュレーション実行に必要なネットリストが、当該サブプロセスを実行して生成されている場合のみ選択できます。シミュレーション内容を選択したら、右下の『Next>』ボタンをクリックして先に進みます (図 10-3 では非表示)。



図 10-4. RTL ファイルリスト表示



次はシミュレーションに使用するソースファイルの選択ウィンドウです (図 10-4)。デフォルト表示でシミュレーション種別に対応したソースファイルが選択された状態になっています。RTL シミュレーションの

場合はプロジェクトにインポートされているファイルが、ネットリストを使用するシミュレーションの場合は、対応するネットリスト・ファイル（と sdf ファイル）が表示されます。

シミュレーションの実行には、これにテストベンチを追加する必要があります。ウィンドウ上の  ボタンをクリックすると、HDL ソースを選択するウィンドウが開きますので、必要なテストベンチの HDL ソース全てを選択しインポートします。テストベンチなどシミュレーションのみに用いるファイルは、Diamond のファイルリスト・ビュー枠に ”simulation” 属性でインポートすることができます。この場合、図 10-4 に示すソースファイル一覧にはそうしたシミュレーション用ファイル一式も含まれますので、毎回新たにインポートする手間が省けます。

VHDL の場合でユーザ定義パッケージがある場合は、当該ソースファイルがリストの上にくる（最初にコンパイルする）ように、ファイルを選択して  や  ボタンをクリックしてファイルの並び順を変えます。

ウィンドウ下部の「Automatically set simulation compilation file order」はチェックされたままにしておきます。「Copy Source to Simulation Directory」のチェックサムは任意です。『Next>』ボタンで次に進みます。

”Parse HDL files for simulation” ウィンドウが表示されます（図 10-5）。下部の「Simulation Top Module」に表示される、自動認識されたトップ・モジュール名が正しいことを確認します。異なる場合はクリックしてプルダウン表示される候補から選択します。意図するモジュールが選択できない場合は、ファイル構成や記述に問題のある可能性があります。問題がなければ『Next>』ボタンをクリックします。

図 10-5. トップ・モジュールの確認

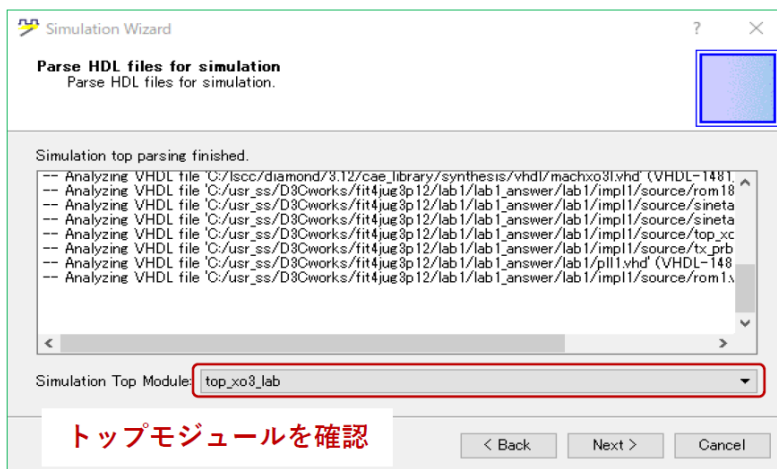
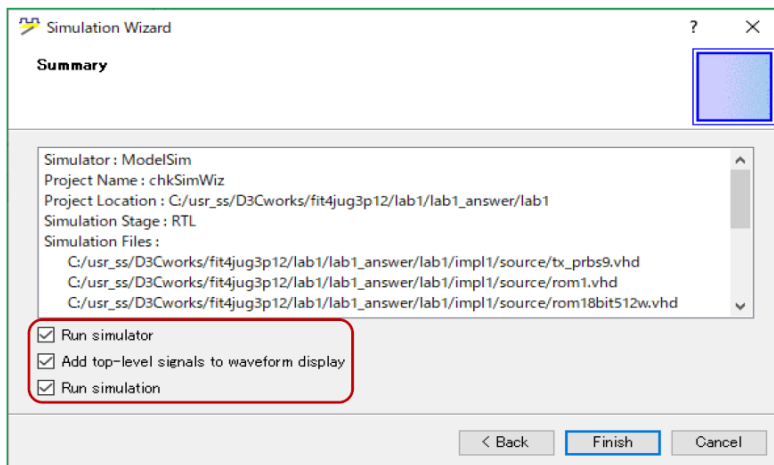


図 10-6. 設定確認のウィンドウ

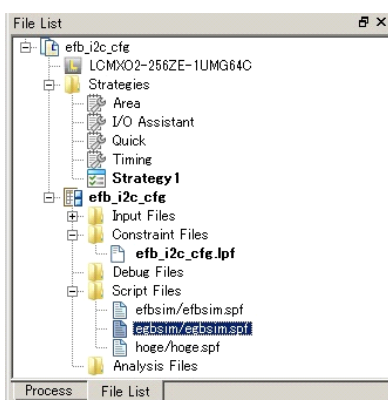


最後がこれまでの設定の確認画面です (図 10-6)。問題がなければ『Finish』ボタンをクリックします。変更の必要があれば『<Back』をクリックして該当するステップに戻って修正します。デフォルトでは下部三つのオプションにチェックが入っています。「Run Simulation」にチェックが入っている状態で『Finish』をクリックすると ModelSim LE が起動し、インポートしたソースのコンパイルが行われ、そしてシミュレーションが実行されます。実行時間も自動的にツールが設定します。「Add top-level signals to waveform display」にチェックが入っていると、波形表示信号はテストベンチのトップのみです。このチェックをはずすと「Run Simulation」はチェック (イネーブル) できません。

「Run Simulation」にチェックが入っていない場合は、コンパイルまでが行われます。ユーザーが ModelSim LE 上でシミュレーション初期化、波形表示信号の指定、シミュレーション実行を行います。GUI 起動後の操作については第 10.3 節をご参照ください。


作成されたシミュレーション・プロジェクトの "Simulation Wizard スクリプト" ファイル(*.spf)は、Diamond のファイルリスト・ビュー下の "Script Files" セクションに自動的にインポートされます (図 10-7)。

図 10-7. 作成されたスクリプト・ファイルのインポート例



*.spf がインポートされていれば、二回目以降はこれをダブルクリックすると、各ウィンドウが初めのステップから設定された状態が表示されます。複数の異なる *.spf をインポートすることができます。

10.3 GUI 起動・操作によるシミュレーション実行

Simulation Wizard ではなく、ModelSim LE GUI 起動でシミュレーションする場合、少なくとも本節で記述するような基本操作が必要となります。Simulation Wizard で最後のステップ (図 10-6) の左下オプション3つのうち、「Run Simulator」以外をチェックしていない状態で [Finish] をクリックする場合には、本節で記述する操作ステップの後半が必要となります (後述)。単独起動する場合は、アイコン  をクリックするか、Diamond メニューから [Tools] → [ModelSim Lattice-Edition] を選択します。

10.3.1 作成済みプロジェクトのオープン

作成済みのプロジェクトで再度シミュレーションをする場合は、メニューから [File] → [Recent Projects] → [プロジェクト・リスト] から選択します。リストには最大 8 つの履歴が表示されます (図 10-8)。[File] → [Recent Directories] の操作でも同様ですが、ディレクトリの移動のみができます。

なお、Diamond と Radiant 両環境で ModelSim LE のシミュレーションを行っている場合、環境は分別されずに、プロジェクト名が混在してリストされることにご留意ください。ツールが併存する環境下では、プロジェクト名を工夫して識別できるようにすることを推奨します。


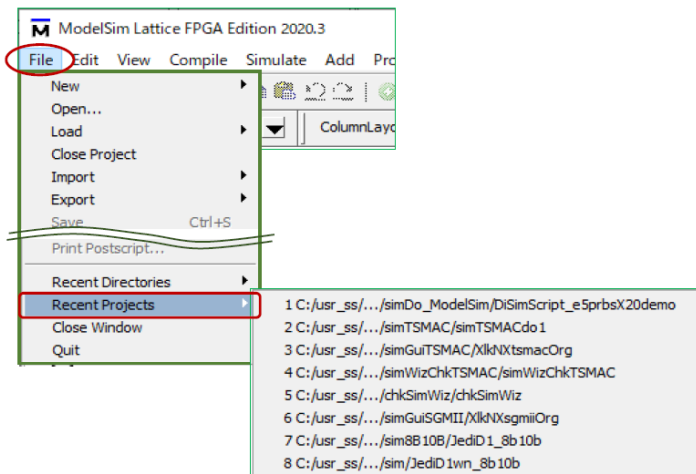
プロジェクト・リストに含まれていない場合、メニューバーの [File] → [Open...] を選択するか、 アイコンをクリックします。表示される “Open File” ウィンドウの右下ファイルタイプ “Project Files (*.mpf)” を指定し、ブラウザ・選択して『OK』をクリックすると、プロジェクトがオープンします。

図 10-8. 作成済みプロジェクトのオープン



なお、この操作は”Transcript” 枠に次のようにコマンドを入力することと等価です。<location>は必ずフルパスで入力します。

```
ModelSim> project open C:/<location>/<project_name>.mpf
```

10.3.2 新規プロジェクトの作成

最初にシミュレーション作業を行うディレクトリに移動します。作業フォルダーを予め作成しておきます。移動方法は二つあり、第一の方法はメニュー [File] → [Change Directory...] と選択すると表れるファイル・ブラウザで所望のフォルダーを指定します。第二の方法は GUI 下部 ”Transcript” 枠に ”CD” コマンドを入力することです。例えば次のようになります。セパレータは ”¥” (バックスラッシュ) ではなく、必ず ”/” (スラッシュ) にして区切ります。

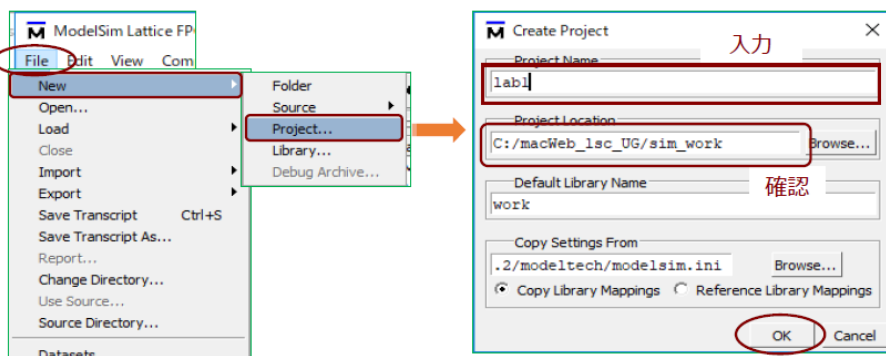
```
ModelSim> cd C:/xxx/yyy/zzz
```

次にシミュレーション・プロジェクトを作成します。メニュー [File] → [New] → [Project...] と選択すると表れるウィンドウの「Project Name」セルにプロジェクト名を入力します (図 10-9)。「Project Location」が意図する作業ディレクトリであることを確認します。デフォルトのシミュレーション・ライブラリー (Default Library Name) は ”work” になっていますが、敢えて変更する必要はありません。

なお、この操作は”Transcript” 枠に次のようにコマンドを入力することと等価です。<location>は作業ディレクトリです。以降、作業ディレクトリがカレント・ディレクトリになります。

```
ModelSim> project new <location> <project_name>
```

図 10-9. シミュレーション・プロジェクトの作成



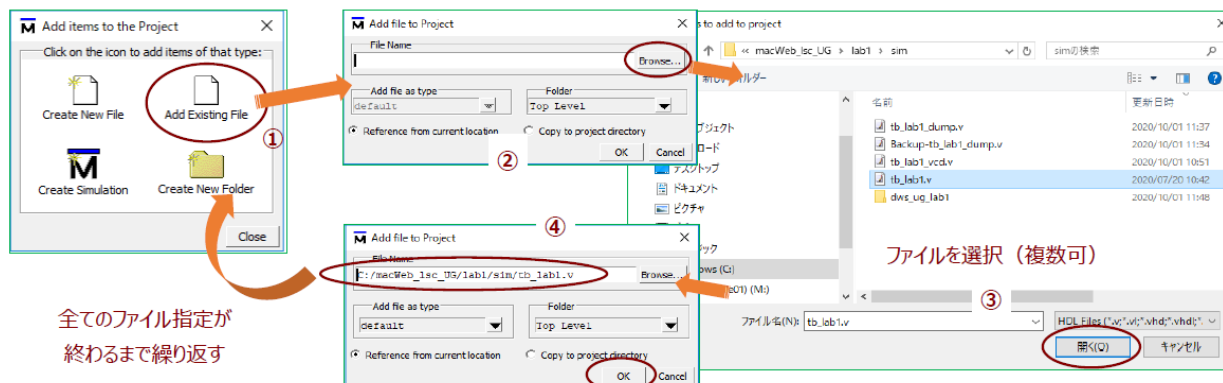
10.3.3 RTL ソースファイルのインポート

新規プロジェクトを作成すると、自動的に表示される ”Add items to the Project” ウィンドウでプロジェクトにインポートする RTL ソースファイルを指定します (図 10-10)。

- ① ”Add Existing File” をクリック
- ② ”Add file to Project” ウィンドウで 『Browse』 をクリック
- ③ ファイルを選択して 『開く』 をクリック
- ④ ”File Name” を確認して 『OK』 をクリック

すべてのファイルを指定するまで①～④を繰り返し、最後に①のウィンドウで 『Close』 をクリックします。

図 10-10. RTL ソースファイルのインポート

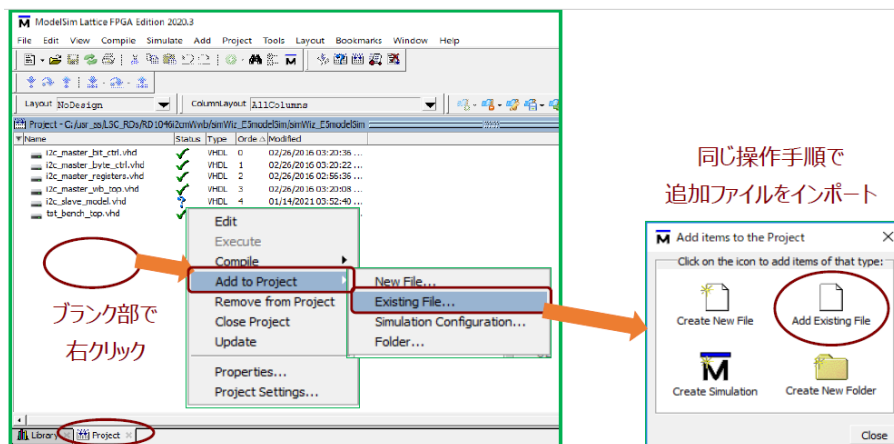


一旦ウィンドウを閉じた後や、プロジェクトを再オープンしてシミュレーション作業する場合にファイルの追加が必要な場合、”Project” ウィンドウ (”Project” タブを選択した状態) で、任意の表示が空白などこかを右クリックして、[Add to Project] → [Existing File] と選択すると、図 10-10 と同じウィンドウが現れます。またはメニューから [Project] → [Add to Project] → [Existing File] と選択しても同様です。

なお、この操作は ”Transcript” 枠に次のようにコマンドを入力することと等価です。<file_name> は作業フォルダー内に存在しない場合は相対パス、絶対パスいずれかの表記を伴う記述にします。

```
ModelSim> project addfile “<file_name>”
```

図 10-11. RTL ソースファイルの追加インポート



10.3.4 コンパイル


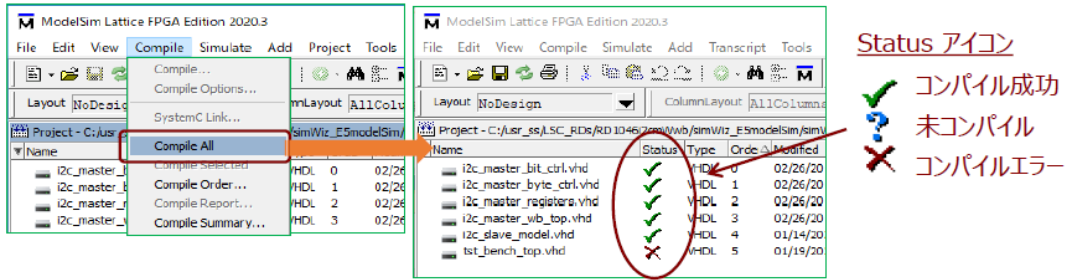
次のステップはソースファイルのコンパイルです。アイコンメニューからアイコン  をクリックするか、またはメニューから [Compile] → [Compile All] を選択します (図 10-12)。

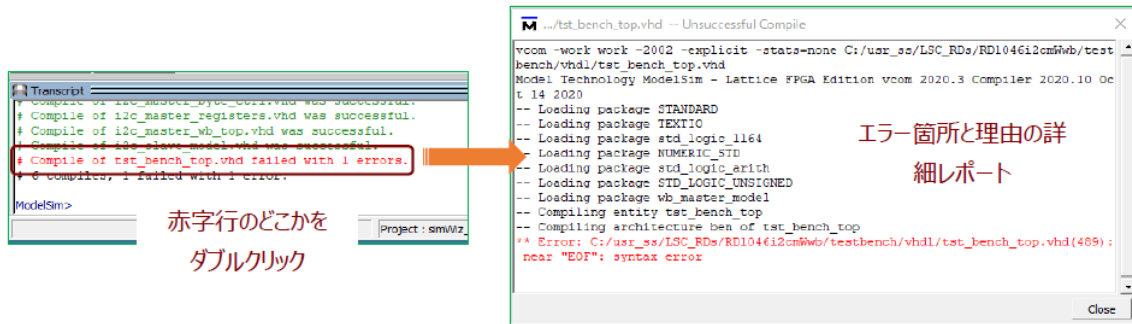
図 10-12. ソースファイルのコンパイル



ステータスが ? のソースファイルのみをコンパイルする場合は、アイコンを クリックします。

コンパイルエラーがある場合、Transcript ウィンドウの当該ログが赤字表示になります。赤字行のどこかをダブルクリックすると、別ウィンドウが開き (図 10-13 右)、エラーのより詳細な理由がレポートされますので、参考になります。

図 10-13. コンパイルエラーの詳細表示



Simulation Wizard 節でも記述したとおり、VHDL でユーザ定義パッケージがありこれを用いる場合は、最初に (参照される他のソースファイルの前に) その定義ファイルをコンパイルする必要があります。メニューから [Compile] → [Compile Order...] と選択すると表示されるウィンドウで、コンパイルするファイル順を指定・変更できます。

なお、コンパイルは "Transcript" 枠に次のようにコマンドを入力することと等価です。以下の例のように <file_name> を複数列記しても良いですが、全ての RTL ソースファイルについてコンパイルを行います。<file_name> は作業フォルダー内に存在しない場合は相対パス、絶対パスいずれかの表記を伴う記述とします。

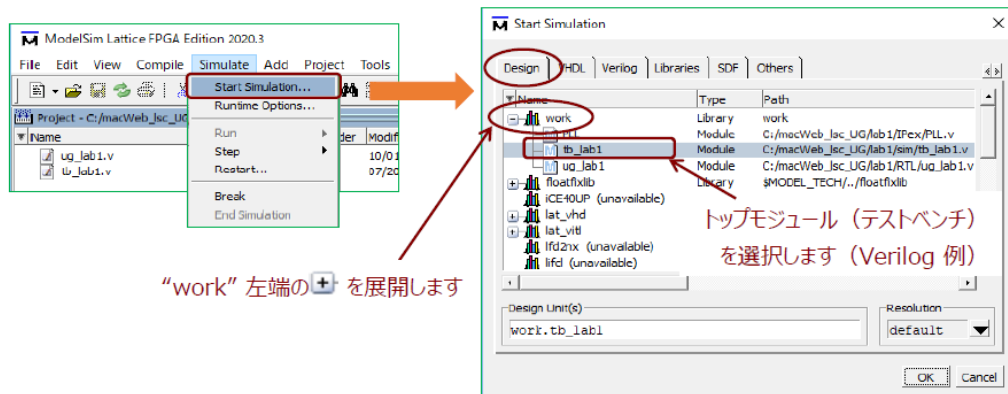
```
Verilog      ModelSim> vlog -work work <file_name> [<file_name>]
VHDL        ModelSim> vcom -work work <file_name> [<file_name>]
```

10.3.5 シミュレーションの初期化

次にシミュレーションの初期化を行います。アイコン をクリックするか、メニューバーで [Simulate] → [Start Simulation] を選択します。"Start Simulation" ウィンドウが表示されますので、"Design" タブで作業プロジェクトのライブラリー名 (デフォルト "work") を展開し、トップモジュール名=テストベンチ名を選択します。"Type" カラム表示が Verilog では "Module" (図 10-14 右例)、VHDL では "Entity" です。

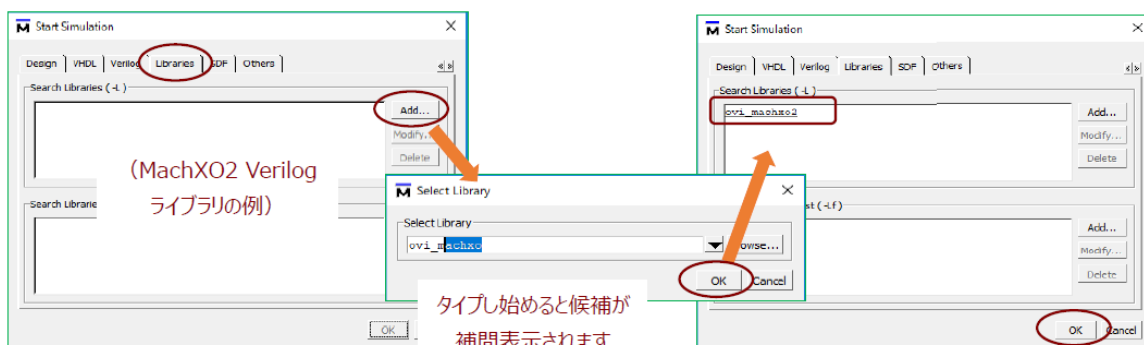
"Design" タブの次は "Libraries" タブです。「Search Libraries (-L)」枠の右にある『Add...』ボタンをクリックすると表示されるウィンドウ (図 10-15 中央) でターゲット・デバイスのライブラリー名入力して『OK』ボタンをクリックします。タイプし始めると候補名が自動補間表示されます。ライブラリー名がわからない場合は第 10.6.1 項をご参照ください。或いは図 10-11 左図の下部にある [Library] タブを選択すると、ツールにロードされているライブラリーの一覧が表示されていますので、事前に確認できます。"ovi" で始まるのが Verilog ライブラリー、"ovi" のないものが VHDL ライブラリーです。

図 10-14. シミュレーションの初期化 1



ECP5 ファミリーをターゲットとして SERDES (PCS) が実装されているデザインでは、これに加えて ”pmi_work” と ”pscd_work” という名称のライブラリーも必要です (第 10.5 節参照)。全てのライブラリーを指定したら、『OK』ボタンをクリックして (図 10-15 右) 戻ります。

図 10-15. シミュレーションの初期化 2



なお、この操作は ”Transcript” 枠に次のようにコマンドを入力することと等価です。〈TestBench_name〉は RTL 記述でのテストベンチ名です。

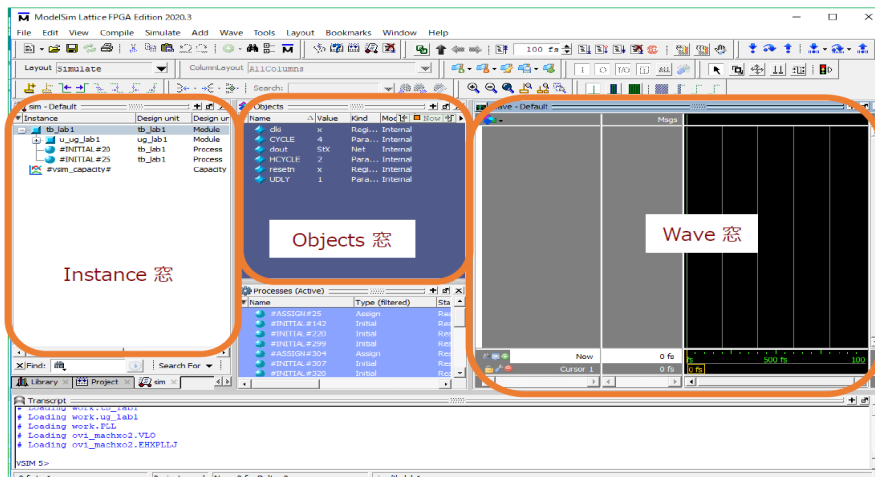
ModelSim > vsim -L work -L ovi_machxo2 <TestBench_name>

MachXO2 の場合

ModelSim > vsim -L work -L ovi_ecp5um -L pmi_work -L pscd_work <TestBench_name>

ECP5、SERDES 有り

図 10-16. シミュレーション初期化後の表示例

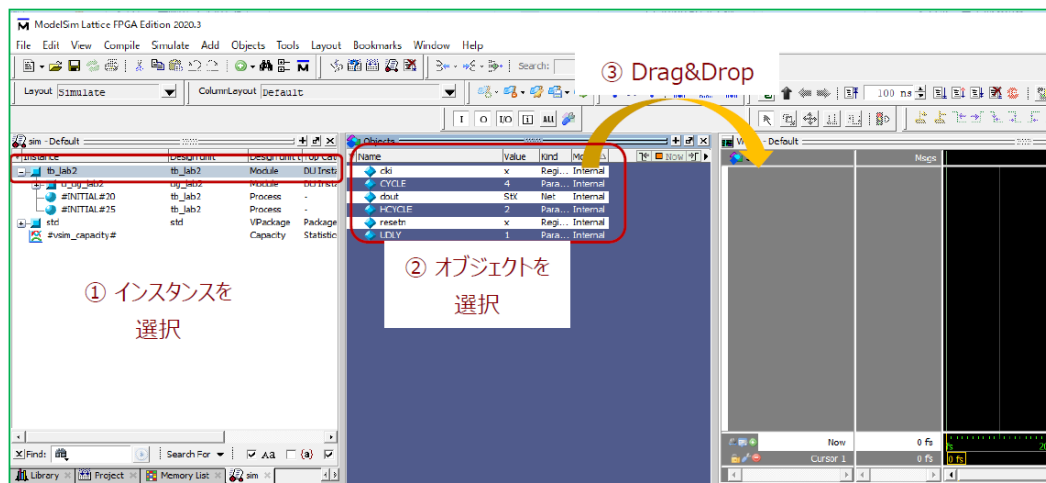


10.3.6 波形表示する信号の選択

シミュレーションの初期化が問題なく処理されると、初期表示は図 10-16 のようになります。まれに”Objects” 窓や”Wave” 枠が表示されないことがあります。その場合はメニューバーでそれぞれ [View] → [Objects]、[View] → [Wave] を選択すると表示されます。

表示（観測）する信号の指定方法は複数あります。

図 10-17. ドラッグ&ドロップによる表示波形の指定



ドラッグ&ドロップで行う場合（図 10-17）、①最初に左側”Instance” 窓で対象信号を含むモジュール（エンティティ）を選択し、②”Objects” 窓でオブジェクト（信号名）を選択し、③選択したオブジェクトを”Wave” 窓内にドラッグ&ドロップします。なお、オブジェクトとは、Verilog ではポートおよび wire や reg 宣言されているネット、VHDL ではポートおよび signal 宣言されているネット（またはノードと呼ぶ）です。


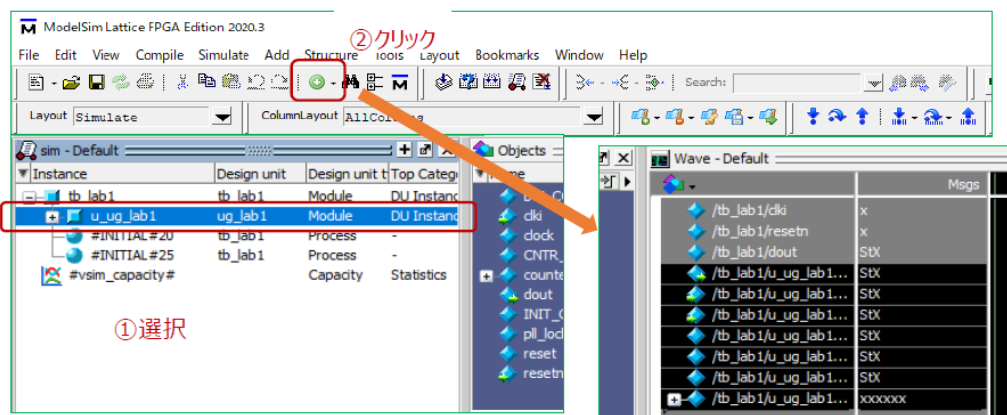
別の指定方法では、アイコンを用います。①左側”Instance” 窓で対象信号を含むモジュール（エンティティ）を選択するか、または”Objects” 窓で対象信号を選択し（『Ctrl』キーによる複数選択も可）、②アイコン  をクリックします（図 10-18）。”Instance” の場合は、選択されたモジュール（エンティティ）の全オブジェクトが追加されることにご留意ください。

図 10-18. アイコンによる表示波形の指定例



これ以外の方法として、”Instance” 窓でモジュール（エンティティ）を選択してメニューから [Add] → [To Wave] → [All Items in region] を選択したり、モジュール（エンティティ）を選択してその上で右クリックすると表示されるメニューで [Add to] → [Wave] → [All Items in region] を選択します。モジュール（エンティ

ティ) の全オブジェクトが追加されます。

或いは "Objects" 窓でオブジェクト (複数可) を選択し、その上で右クリックすると表示されるメニューで [Add to] → [Wave] → [Selected Signals] を選択しても追加できます。

各信号 (オブジェクト) を "Wave" ウィンドウに指定し終えたら、表示順を調整します。ウィンドウ内で変更する信号を選択してドラッグ&ドロップします。

なお、この操作は "Transcript" 枠に次のようにコマンドを入力することと等価です。種々表示オプションがありますが、ここでは割愛します (以下プロンプトの "x" はコマンド実行毎にインクリメントする数字)。


```
VSIM x> add wave -noupdate /*                テストベンチ階層の全オブジェクト
VSIM x> add wave -noupdate /<TestBench_name>/u_top/*   テストベンチ内インスタンス u_top の全て
VSIM x> add wave -noupdate /<TestBench_name>/u_top/u_abc/nodex   インスタンス u_abc の固有の信号
```

特定の複数オブジェクトを指定する場合は、全て三番目のように与える必要があります。通常、波形表示するオブジェクトは多数になります。論理シミュレーションは通常は繰り返し作業であり、"*" による指定では順序や属性が制御できませんので、"DO" マクロを用いて実行することが良く行われます。

"Wave" ウィンドウでの表示波形リストと順序 (および属性) が指定できたら、DO マクロとして書き出しておくことを推奨します。メニューから [File] → [Save Format...] と選択すると、"Save Format" ウィンドウが表示されます。"Pathname" セルにディレクトリとデフォルトの DO マクロ・ファイル名がフィルされていますので、適宜編集して保存します。次回以降は "Transcript" 枠に次のように入力することで容易に波形表示設定を再現できます (波形表示マクロをカレント・ディレクトリに "waves.do" として保存した場合)。

```
VSIM x> do waves.do
```

10.3.7 シミュレーションの実行

シミュレーション実行は、ツールバーの "Run Length" セルに実行時間を単位と共に入力して  アイコンをクリックします (図 10-19)。これ以外にシミュレーション実行関連アイコンは幾つかあり、それぞれ以下のような動作になります。シミュレーション実行中は Break と Stop アイコンのみが有効です。





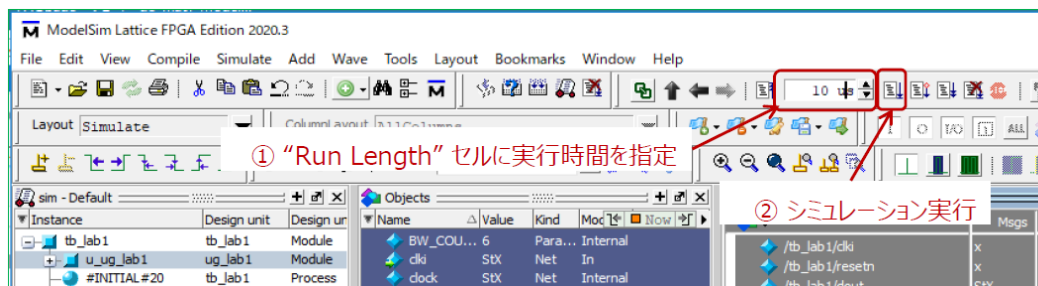
-  : Break (シミュレーションを一時中断)
-  : Continue Run (中断したシミュレーションを再開)
-  : Run All (シミュレーションをテストベンチ記述の最後まで実行)
-  : Stop (シミュレーション停止。再開不可)

図 10-19. シミュレーション実行



なお、この操作は "Transcript" 枠に次の例のようなコマンドを入力することと同等です。

```
VSIM x> run 10 us      或いは
VSIM x> run -all      など
```

10.3.8 シミュレーションの再実行


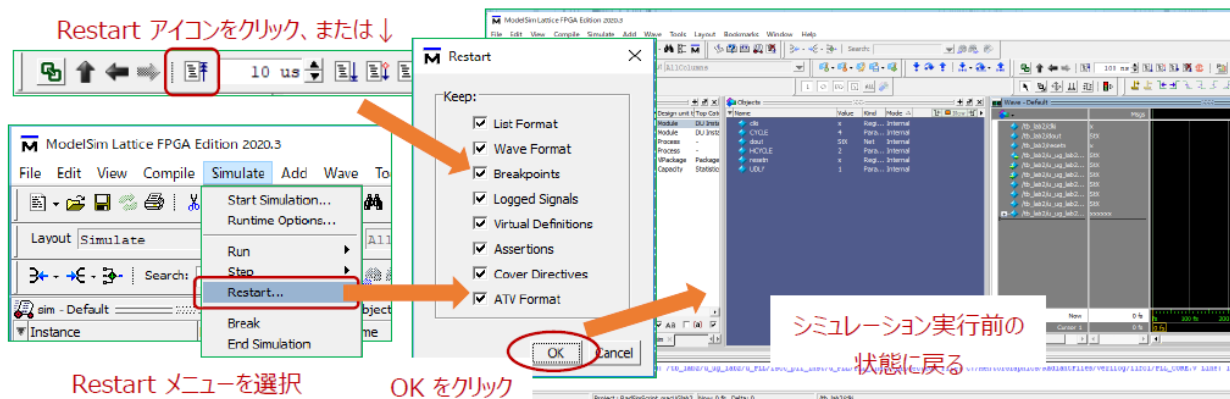
RTL ソースファイルの変更は一切なく、観測する波形の追加・削除や表示順の変更、或いは表示属性（2進、16進など）や表示名のみを操作・編集することは頻繁にあります。そのような場合のシミュレーション再実行は、のように行います。まずメニュー[Simulate] → [Restart...]を選択するか、アイコン  をクリックします。表示される”Restart” ウィンドウで『OK』ボタンをクリックすると、各ウィンドウがシミュレーション実行前の状態に更新されます（図 10-20）。

図 10-20. 波形表示のみを変更後のシミュレーション再実行



RTL ソースファイルの追加・削除や記述の編集など、何らかの変更がある場合の再実行には、一旦シミュレーション実行を終了し（第 10.3.9 項）、ソースファイルの再コンパイル（第 10.3.4 項）から行います。

なお、表示信号の属性変更は、”Wave” ウィンドウで当該信号を選択して右クリックすると表示されるメニューから、[Radix] などや [Properties...] を選択して行います（16 進表記の指定は add wave コマンドに -hexadecimal オプション指定でも可能です）。また、”Wave” ウィンドウでの信号名は階層が深いと冗長になりますので、add wave コマンドに -label オプションを追記して表示名を指定できます。また、インスタンス毎に分割表示する場合は、メニュー [Add] → [To Wave] → [Divider...] と選択してセパレータ行を追加し、表示名を入力します。

再実行の操作は”Transcript” 枠に次のようにコマンドを入力することと等価です。

```
VSIM x> restart           波形表示が変わらない場合の再実行
VSIM x> restart -f -nowave  波形表示を変更した後に再実行
```

10.3.9 シミュレーションの終了

RTL ソースファイルの追加・削除や記述の編集など、何らかの変更がある場合のシミュレーション再実行には、まず一旦シミュレーションを終了します。メニュー [Simulate] → [End Simulation] を選択します。

なお、この操作は”Transcript” 枠に次のようにコマンドを入力することと等価です。

```
VSIM x> quit -sim
```

ModelSim LE ではシミュレーションの終了、プロジェクトのクローズ（終了）、そして ModelSim LE のクローズ（終了）を区別する必要があります。プロジェクトのクローズは”Transcript” 枠に次のコマンドを入力します。

```
VSIM x> project close
```

ModelSim LE の終了は、メニュー [File] → [Quit] を選択するか、”Transcript” 枠に次のコマンドを入力します。

```
ModelSim > quit
```

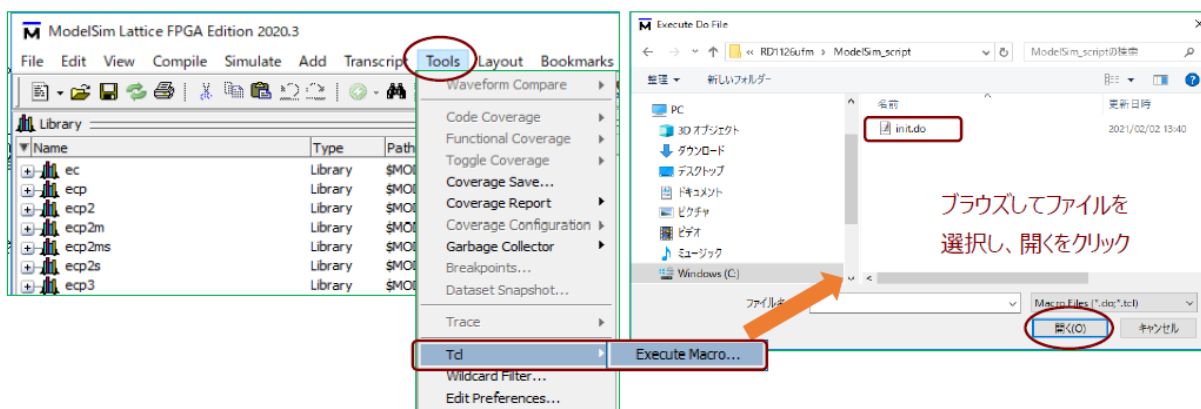
10.4 TCL スクリプトによるシミュレーション実行

これまでに記述した Simulation Wizard や GUI 起動後のマウス操作によるシミュレーションは、簡易的に実行する場合は有用ですが、やや煩雑な操作を伴います。TCL スクリプト (DO マクロ) を用いた作業にすると、一連の操作の殆どを自動実行できます。ユーザーが用意するスクリプトの拡張子は任意ですが、慣例的に ".do" としますので、ここでは "DO マクロ" と呼びます。

10.4.1 スクリプトの実行方法

既に DO マクロが用意されていれば、メニューから [Tools] → [Tcl] → [Execute Macro...] と選択し、ファイルブラウザでファイルを選択後に『開く』をクリックして実行できます (図 10-21)。

図 10-21. メニューからの DO マクロの実行例



或いは "Transcript" 枠にコマンド入力することでも実行できます。

```
ModelSim > cd <work_directory>          まず DO マクロのある作業ディレクトリに移動
ModelSim > do init.do                    マクロ "init.do" を実行
```

作業ディレクトリとは別の所に DO マクロがある場合は、do コマンドに続くファイル名を相対パスや絶対パス記述で指定します。カレント・ディレクトリがどこかを知りたい場合は、"pwd" と入力します (print working directory)。

10.4.2 DO マクロの典型的な記述例

ここでの記述例に示す、頻繁に用いるコマンドの文法は、前項で例示されているものに相当します。コマンド記述の順序・内容は、概ね次のようになります。Simulation Wizard によるシミュレーションを実行すると、そのフォルダーに拡張子 "*.mdo" のテキスト・ファイルが生成されます。これをガイド (参考) として意図するプロジェクト用の DO マクロを作成する方法もあります。

- ・ マクロ内で使用する変数の定義と新規プロジェクト作成
- ・ RTL ソースファイルのインポート
- ・ RTL ソースファイルのコンパイル
- ・ シミュレーションの初期化
- ・ 表示波形の指定
- ・ シミュレーション実行

マクロ記述例です。文頭は説明のための行番号 (マクロ記述では不要)、<>印はユーザー固有の環境や名称を意図しています。ここでのファイル名は例であり、特に意味はありません。

```

1  # "init.do"                コメント行は "#" で始めます
2  set WORK_DIR C:/<work_directory>    "作業ディレクトリ" をマクロ変数として定義します
3  set RTL_DIR C:/<RTL_source_directory>  RTL ソースのあるディレクトリを変数として定義します
4  cd $WORK_DIR                作業ディレクトリに "Change Directory" で移動します
5  #
6  project new $WORK_DIR <project_name>  作業ディレクトリに新規プロジェクトを作成します
7  #
8  project addfile $RTL_DIR/jk.vhd      定義済み変数 RTL_DIR 下の VHDL ソースをインポートします
9  project addfile $RTL_DIR/mn.v       同、Verlog ソースをインポートします
10 project addfile tb_zz.v             カレント（作業）ディレクトリにあるテストベンチを加えます
11 #
12 vlib work                          作業ライブラリ名を "work" とします
13 vdel -lib work -all                 念のため古い（前の作業時の）work ライブラリを削除します
14 vlib work                          再度宣言します
15 #
16 # compile Verilog source files      RTL ソースファイルのコンパイルです
17 vcom $RTL_DIR/jk.vhd               VHDL ソースをコンパイルします
18 vlog $RTL_DIR/mn.v tb_zz.v         Verilog ソースとテストベンチと一緒にコンパイルします
19 #
20 # initialize simulation              シミュレーションを初期化します
21 vsim -L work -L ovi_machxo2 <tb_name>  ターゲットが MachXO2 の例です
22 #
23 # set up signals to display          表示波形の指定例です
24 add wave /<tb_name>/<instance_top_name>/*  TB 内 DUT トップのインスタンス下全オブジェクトを追加
25 #do waves.do                       (該当時の例) 作業ディレクトリにある波形表示指定のみの DO マクロを実行
26 #
27 # run the simulation                 シミュレーションを実行します
28 run 100 us                          実行時間は 100us。数値と単位の間にはスペースを入れます

```

以下に何点かコメントを追記します。

- 8～10行目 ファイルのあるフォルダー記述を伴う場合は、相対パスか絶対パスで記述します
- 17～18行目 同様です
- 21行目 “-t” オプションで時間精度を指定することがあり得ます。遅延（タイミング）シミュレーションについては、次項で追記します
- 25行目 波形表示 DO マクロについては、次項で具体例を示します
- 4～12行目 一度この DO マクロを実行した後、繰り返しコンパイルからシミュレーション実行まで行う場合に、これらの行は不要です。コメントアウトして別ファイルとして保存し、それを繰り返し実行用 DO マクロにすると便利です

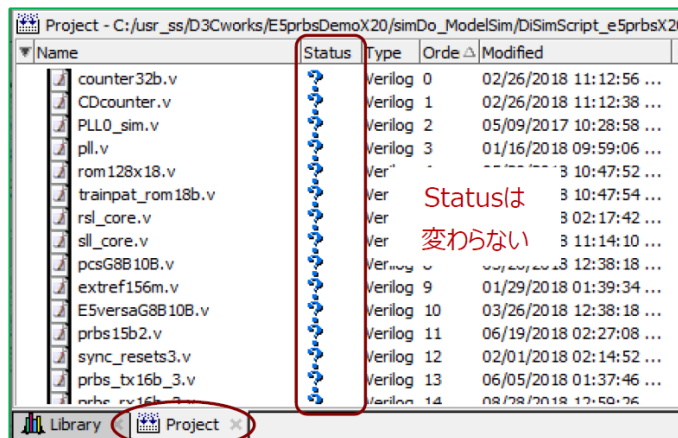
なお、各コマンドに続くオプションやファイル名などが長くなる場合、複数行に亘って記述することも可能です。その場合は文末で改行する直前に “¥” 文字（半角バックスラッシュ）を記述します。この特殊文字はコマンドが次の行まで継続することをツールに指示します。“¥” 後は空白も何も付加しないようにします。

繰り返し実行用 DO マクロを作成して、これを実行する場合は “Transcript” 枠で次のように入力します（マクロ名を “run.do” とした場合の例）。

```
VSIM x> do run.do
```


補足として、ソースファイルをインポートすると [Project] タブにファイルがリストされますが、DO マクロの実行によってコンパイルが成功している場合でも、”Status” カラムの記号は未コンパイル状態表示のままです(図 10-22)。しかし、”Transcript” 枠のログでエラー・メッセージが出ていなければ、問題はありません。

図 10-22. DO マクロによるコンパイル・ステータス記号



10.4.3 波形表示 DO マクロ

以下は波形表示 DO マクロのごく基本的な記述例です：

```

1 add wave -noupdate /tb_lab/resetn
2 add wave -noupdate /tb_lab/clki
3 add wave -noupdate /tb_lab/dout
4 add wave -divider "u_ug_lab"
5 add wave -noupdate -label pll_lock /tb_lab/u_ug_lab/pll_lock
6 add wave -noupdate -label counter -hexadecimal /tb_lab/u_ug_lab/counter
    
```

以下に何点かコメントを追記します。

- 1～3 行目 テストベンチ直下の信号 resetn を表示、等々
- 4 行目 波形の間にセパレータ行を挿入し、次の行から続く信号のあるモジュール名を明記
- 5 行目 表示名を短縮するために”-label” オプションを使用して表示名を指定
- 6 行目 -label オプションと共に、-hexadecimal オプションで 16 進表記を指定

第 10.3.6 項で記述したような手順で波形表示 DO マクロを書き出した場合は、これをそのまま用いるか、これを編集して使用することになります。

10.4.4 タイミング・シミュレーション

遅延付きタイミング・シミュレーションをする場合の初期化は、以下のようになります。Diamond で生成するネットリストのシミュレーション用 RTL (Verilog) を”*.vo”、その遅延ファイルを”*.vo.sdf” とし、カレント (作業) ディレクトリにコピーされているものと仮定します。*.vo/.sdf の生成方法については、第 8 章をご参照ください。

```
vsim -t 10ps -L ovi_machxo2 <tb_name> -sdfmax <dut_top>=<design_example>.vo.sdf
```

”-t” は時間精度指定 (数値と単位の間スペースなし)、<tb_name> はテストベンチ名、-sdfmax は条件の指定 (sdfmin/sdftyp/sdfmax のいずれか)、<dut_top> はテストベンチでインスタンスされているシミュレーション対象デザインのインスタンス名です。

なお、タイミング・シミュレーションでコンパイルする RTL はテストベンチ関連以外は唯一”*.vo”のみ

です。また、*.vo/.sdf がカレント・ディレクトリにない場合は、相対 / 絶対パス記述を伴う指定にします。

10.5 PCS/SERDES のシミュレーション

インプリメンテーションで SERDES (PCS) を用いている場合のシミュレーション実行について、ECP5UM ターゲットを例にして、留意点を補足記述します。

ECP5 ファミリーでは Clarity Designer で PCS モジュールを生成・プランニングしますが、リセットシーケンスを実行するソフトロジックのモジュール "RSL" が必ず PCS と同時に生成されます (生成しないオプションもあり)。これは生成する (トップ) モジュールの言語指定に拘わらず、必ず Verilog で、ファイル名は <PCS モジュール名>_rsl.v です。個の場合の RSL のコンパイル・コマンドは以下のようにします。リセットシーケンスは実シリコンの動作としては数十 msec 以上の規定時間を待つためのカウンタが幾つか含まれています。これらはシミュレーション時間的には許容できない程度に長いので、短縮するディレクティブを明示します。二つの方法があります。

- ① コマンドライン (スクリプト内) : `vlog +define+RSL_SIM_MODE <WORK_DIR>/<PCS mod>_rsl.v`
- ② パラメータ指定ファイル等を用意して 'define RSL_SIM_MODE を記述

ここで、<WORK_DIR> はスクリプトやコマンドを実行する作業フォルダーからの相対パス、或いはフルパス指定です。RSL 内記述は "RSL_SIM_MODE" というパラメータ定義があると短縮する動作になっています。

次に、シミュレーション実行時のライブラリー指定方法例です。<tb_top> はテストベンチのトップモジュール名を示すものとします。ECP5UM のデバイス内にあるモジュール (マクロ) の記述は全て Verilog です。vsim コマンドは次のようになります。

```
VSIM x> vsim -L ovi_ecp5um -L pmi_work -L pcsd_work <tb_top>
```

10.6 シミュレーション・ライブラリー

10.6.1 ライブラリー名

シミュレーション初期化 (第 10.3.5 項など) において指定するターゲット・デバイスのライブラリー名は表 10-2 の通りです。

表 10-2. ライブラリー名

デバイス	Verilog ライブラリー名	VHDL ライブラリー名
MachXO2	ovi_machxo2	machxo2
MachXO3L, LF	ovi_machxo3l	machxo3l
MachXO3D	ovi_machxo3d	machxo3d
ECP5U	ovi_ecp5u	ecp5u
ECP5UM	ovi_ecp5um	ecp5um
Crosslink	ovi_lifmd	lifmd
Crosslink Plus	ovi_lifmdf	lifmdf
Mach-NX	ovi_lfmnx	lfmnx

10.6.2 VHDL のライブラリー呼び出し記述

VHDL の RTL ソースファイル内では、それぞれ冒頭で使用ライブラリーを明記する必要があります。Simulation Wizard からシミュレータを起動する場合は必須ではありません。

MachXO2 の場合の例を以下に示します。デバイス固有のマクロをインスタンスする場合は、必ず宣言する必要があります。他のファミリーの場合も同様です。

```
--VHDL 記述例 (合成対象のモジュール記述に含める場合)
-- synopsys translate_off
library MACHXO2;
use MACHXO2.components.all;
-- synopsys translate_on
```

また、MachXO2/3 の EFB のように、Verilog ビヘービア記述のマクロ・モジュールを呼び出す必要があるデザインの場合は、以下の Verilog ライブラリーも含めます (MachXO2 の例)。

```
library ovi_machxo2;
use ovi_machxo2.all;
```

10.7 GSR/PUR のインスタンス

MachXO2 など殆どのデバイスファミリーでは、以下に示すデバイス初期化マクロに相当する GSR (Global Set/Reset) と PUR (Power-Up Reset) のインスタンス記述を、テストベンチに含める必要があります (コンパイルエラーが出なければ不要です)。特にインスタンス名については、この記述例の通りでないとは期待動作しません (エラーになる) ので、ご注意ください。

GSR の "リセット信号名" は実デザインでの信号名です (有効極性は任意で、ツールが自動判定します)。実装している特定デバイスのマクロ・モジュールによってはリセット信号ではなく、"1'b1" とする必要がありますので、シミュレーションで不可解な振る舞いが見られた場合は、書き換えて試すことを推奨します。

// Verilog HDL 記述例 (テストベンチ内)

```
PUR PUR_INST (1'b1) ;
GSR GSR_INST (<リセット信号名>) ;      問題がある場合は GSR_INST(1'b1) と記述変更して試す
```

--VHDL 記述例 (テストベンチ内)

```
-- コンポーネント宣言部
COMPONENT GSR
  PORT ( GSR: IN std_logic ) ;
END COMPONENT;
COMPONENT PUR
  PORT ( PUR: IN std_logic ) ;
END COMPONENT;
...
Begin -- モジュールボディ内
...
GSR_INST: GSR
  port map ( GSR => <リセット信号名> ) ;
PUR_INST: PUR
  port map ( PUR => c_vcc ) ;
```

ここで、VHDL の場合、ECP5 以前のデバイス・ファミリーではここに示す方法で問題ありませんが、ECP5 と Crosslink ファミリーでは VHDL の GSR/PUR プリミティブがないため、ライブラリー宣言に留意する必要があります。ECP5 の例を以下に示しますが、Verilog ライブラリの使用を追加するようにします。

```
library ovi_ecp5u;
use ovi_ecp5u.all;
```

なお、GSR はどのデバイスにも組み込まれているマクロ (グローバル・バッファ) ですが、PUR は実在する訳ではなく、論理シミュレーション用のみのマクロです。

--- *** ---