第 2 章　プロジェクト構造

2.1 プロジェクト構造と特長

Lattice Radiant のプロジェクト構造として、以下のような特長があります。

1．単一ライセンスで複数のデザイン・プロジェクトを同時実行することができます

- 同一プロジェクトを対象にした複数実行は除きます
- Synplify-Pro も同様に GUI を複数立ち上げて実行することができます
- Radiant に付属する ModelSim Lattice Edition は、異なるシミュレーション・プロジェクトに対して GUI を複数立ち上げることはできますが、シミュレーションの複数実行はできません

2．デザイン・プロジェクトは大きくは ”ストラテジー ” および ”インプリメンテーション ” という二つの構成要素で管理されます。本章で詳細に記述します

3．本ユーザーガイドのカバーしていない項目やより詳細については以下をご参照ください

- スタートページ・ビューからリンクされているドキュメント類
- オンラインヘルプ（メニュー [Help] → [Lattice Radiant Software Help]）
- ラティスセミコンダクターのウェブサイト（https://www.latticesemi.com）
- 特定のトピックスについて知りたい場合、GUI 上でマウスポインターを当該アイテムに移動した状態でファンクションキー『F1』を押します。当該項目についてのオンラインヘルプ・ページを直接表示することができます（” コンテキスト・センシティブ・ヘルプ ”）

Lattice Radiant では、” プロジェクト ” ごとにデザインを構成しますが、主な管理機構は ” ストラテジー（Strategy）” と ” インプリメンテーション（Implementation）” という考え方です。プロジェクトには必ず一つの ”LPF 制約ファイル ” が存在し、ユーザーが用意する RTL ソースファイルがプロジェクトの入力になります（図 2-1）。

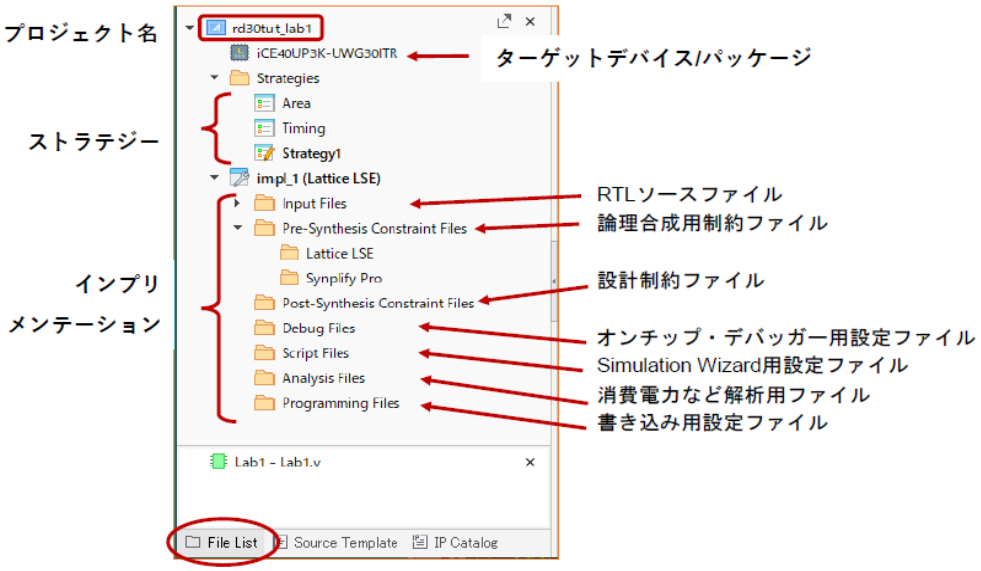
インプリメンテーションはデザインの管理単位であり、デザイン実装に関わる全てのファイルを含んだものです。論理合成や配置配線といった処理プロセスはインプリメンテーション単位で行われます。図 2-1 の表示構成で RTL ソースファイルは [Input Files] セクション、設計制約ファイルは [Pre-Synthesis Constraint Files] および [Post-Synthesis Constraint Files] セクション、これ以外にオンチップ・デバッガー Reveal 用は [Debug Files] セクション、書き込み用ファイルは [Programming Files] セクションに、などそれぞれ表示されます。インプリメンテーションは複数作成することができます。

ストラテジーは各プロセスの制約設定（Constraints、Preferences）やオプション設定を一元管理する機構です。定義済みストラテジーは三種類あり、プロジェクトとして管理されます。有効なストラテジーはインプリメンテーションごとに１つです。複数のインプリメンテーションで同じストラテジーを共用することも、或いはインプリメンテーションごとに異なるストラテジーを割り当てることも可能です（図 2-2）。ストラテジー・オプション個々の詳細についてはプロセス各章の記述をご参照ください。

デザインエントリーは RTL ソースファイル（VHDL/Verilog HDL/System Verilog 記述）で、インプリメンテーションごとに登録（インポート）します。複数のインプリメンテーションで同じソースファイルを参照して共用することが可能です（図 2-2）。モジュール生成（第 4 章参照）した場合は、RTL ソースファイルの代わりに、自動生成される定義ファイル（*.ipx）をインポートします。

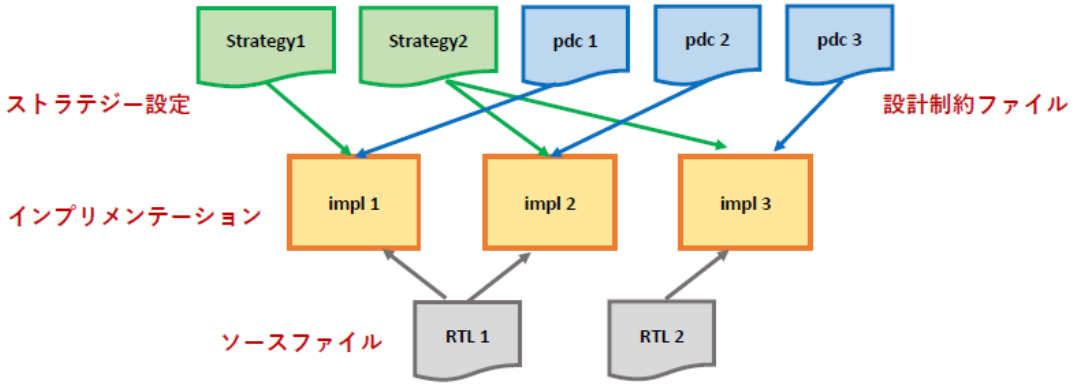
なお、デザインエントリーとして SynplifyPro の出力である ”*.vm” も使用することができます。

図 2-1. プロジェクト構造



各プロセスの処理において、デザインソース・ファイル（と設計制約ファイル）以外は必須ではありません。

図 2-2. 複数インプリメンテーションと制約設定やソース・ファイルの適用例



プロジェクトを構成する［File List］ビューの各ファイルについて、ボールド字体表示の意味することには二通りあります。［Input Files］セクションのソースファイルについては、トップモジュールとして認識されたファイルがボールド表示になります。ボールド表示が二つ以上あったり、或いは一つもない場合は、その状態を解消する必要があります。

インプリメンテーション下の［Input Files］以外の各セクションと、［Strategies］セクションについては”アクティブ（有効）”なファイルがボールド字体表示になります。これらは複数のファイルがインポート（登録）されることが可能ですが、各プロセスに適用されるアクティブな設定ファイルはそれぞれ一つのみです。

2.2 プロジェクト管理

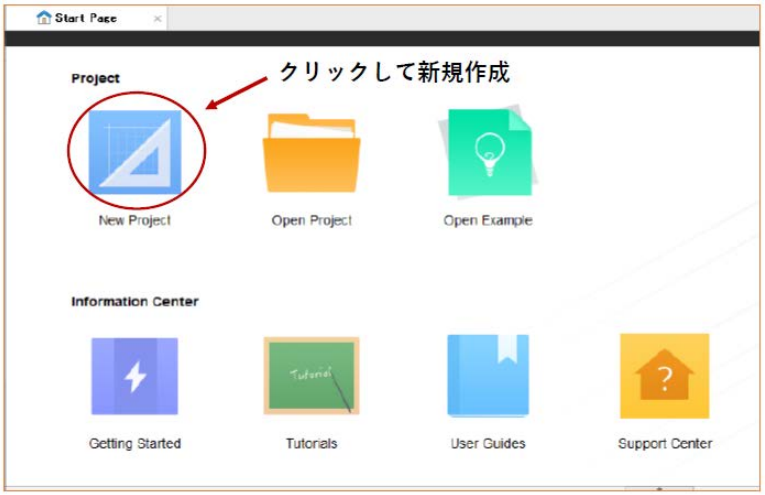
2.2.1 新規プロジェクトの作成

新規プロジェクトの作成は、メニューバーから［File］→［New］→［Project...］の順に選択するか、スタートページ・ビュー上部の Project セクションで大きな”New Project”アイコン をクリックします（図 2-3）。

これで ”New Project” ウィンドウが表示されますので、その指示に従ってデバイス等を選択すればプロジェクトが生成されます。

プロジェクト生成に伴って ”プロジェクト・フォルダ ” は自動作成されませんので、フィッティング用の作業フォルダを事前に作成しておく事を推奨します。

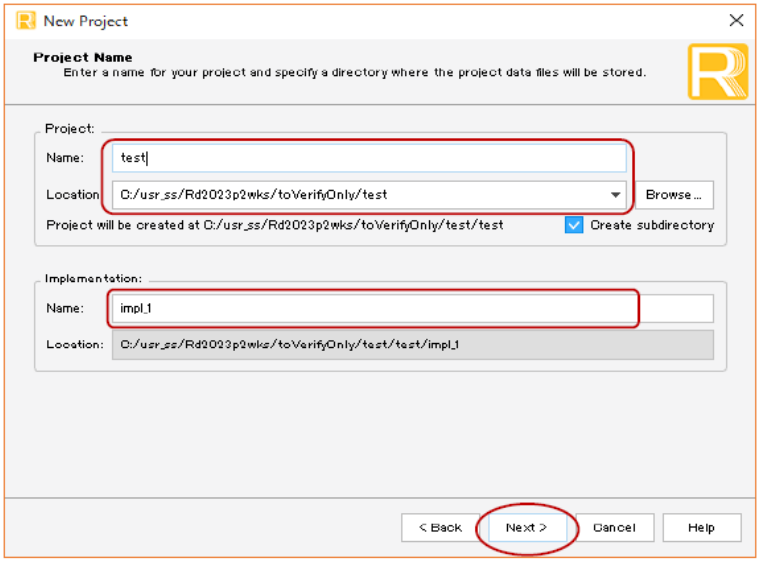
図 2-3. プロジェクトの新規作成開始



”New Project” ウィンドウの始めは、プロジェクトの生成に必要なフォルダやデバイスを指定する旨のメッセージが表示されています。そのまま『Next>』ボタンをクリックし、次に表示されるウィンドウでプロジェクト名とプロジェクトのフォルダパスを設定します（図 2-4）。

Project セクションの「Name」セルにプロジェクト名を入力します。プロジェクト名として使用できる文字は、アルファベット、数字と ”_” （アンダースコア）です（先頭の 1 文字目はアルファベットのみ）。

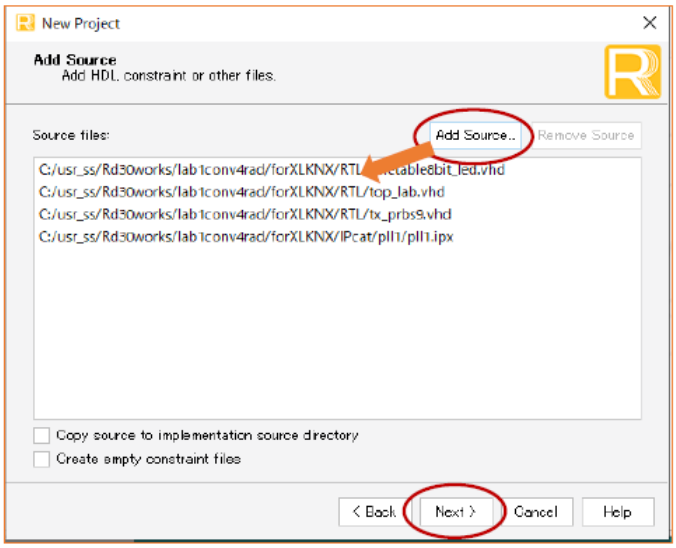
図 2-4. プロジェクトの新規作成～プロジェクト名とフォルダパスの設定



「Location」セルにはプロジェクトで使用するフォルダーを指定します。直接パスを入力するか、『Browse...』ボタンをクリックして立ち上がるウィンドウ上で適切なフォルダを選択します。フォルダーパスとして存在しないフォルダー名を直接入力すると、自動的にその名称の新しいフォルダーが生成されます。

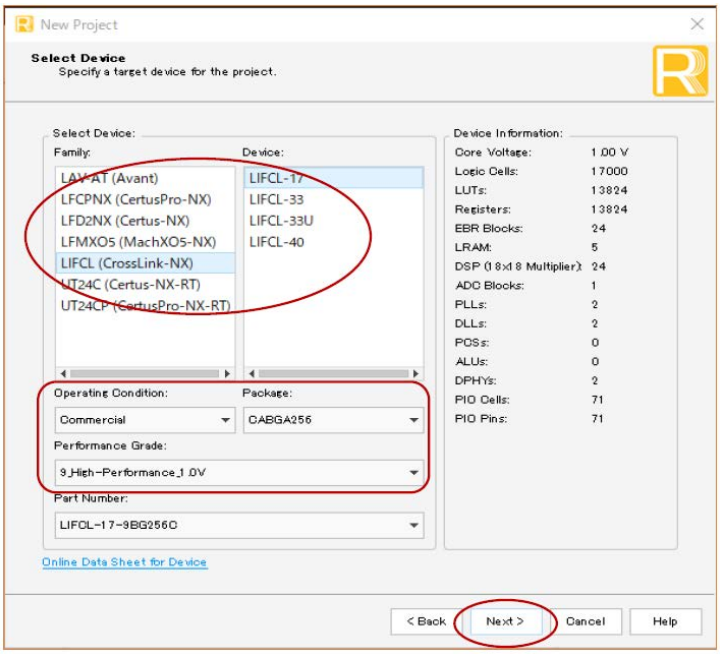
Implementation セクションの「Name」セルにはインプリメンテーション名を入力します。デフォルト名の”impl_1” が自動的に入力されていますが、変更できます。「Name」を変更すると「Location」セルのパス表示も自動的に更新されます。設定が完了後、『Next >』ボタンをクリックすると、次はソースをインポートするウィンドウ [Add Source] が表示されます（図 2-4）。

図 2-5. プロジェクトの新規作成～ソースファイルのインポート



『Add Source...』ボタンをクリックすると起動するファイル・ブラウザーで、必要なファイル（既存の RTL ソースや制約ファイル sdc/ldc/fdc/pdc）を選択して取り込みます。この際、ウィンドウ左下の「Copy source to...」チェックボックスにチェックが入っていると、選択したファイルがプロジェクトのフォルダーへコピーされ、それがインポートされます。チェックが入っていない場合は選択したファイルがインポートされます。

図 2-6. プロジェクトの新規作成～デバイス選択



各ソースファイルはプロジェクト作成後でもインポートできますので、このステップで全てをインポートする必要はありません。完了後『Next >』ボタンをクリックすると、次はターゲット・デバイスの選択ウィ

ンドウが表示されます（図 2-6）。

このウィンドウで、使用するデバイスやスピードグレード、パッケージを選択します。デバイスやパッケージはプロジェクト作成後でも変更できますので、確定していない条件については暫定的に適当なものを選択しておき、確定後に変更するようにします。

次に『Next >』ボタンをクリックすると、論理合成ツールの選択ウィンドウが開きます。2023.2 でのデフォルト論理合成ツールは Synplify Pro ですが、LSE（Lattice Synthesis Engine）を選択する事が可能です。また、プロジェクト生成後でも、変更することが可能です（第 2.2.4.2 項参照）。

図 2-7. 論理合成ツールの選択（ウィンドウの一部）

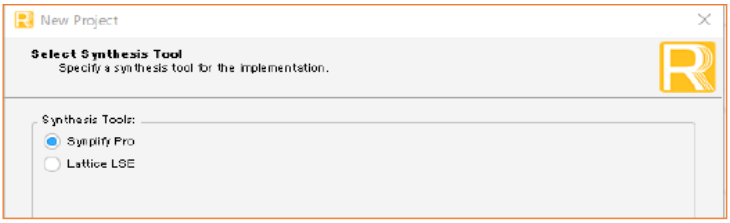
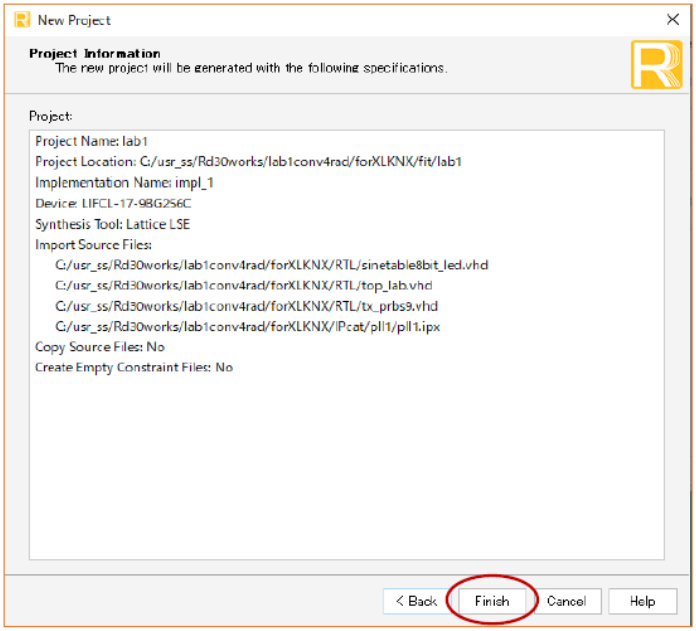


図 2-8. 新規プロジェクトの作成～設定確認



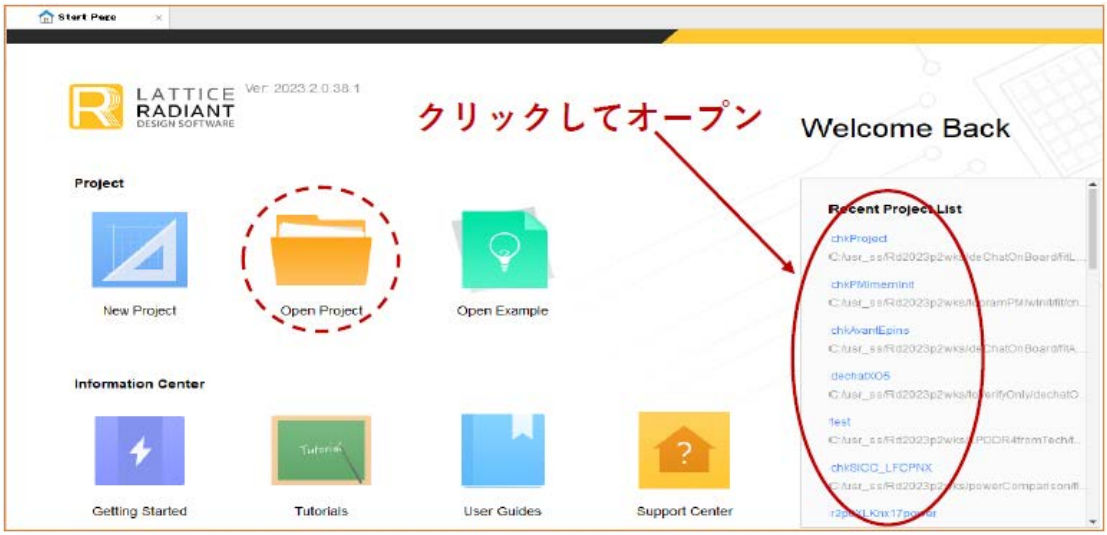
次に『Next >』ボタンをクリックすると、これまでの設定の確認ウィンドウが開きます。『Finish』ボタンをクリックすれば、設定は完了です。設定内容を変更する場合は、『< Back』ボタンをクリックして適切なウィンドウまで戻り、設定変更を行ってください。作成されたプロジェクト情報（インプリメンテーション名やインポートしたファイル等）は、プロジェクト・フォルダに”<プロジェクト名 >.rdf”というファイル名で保存されます。

2.2.2 既存プロジェクトのオープン

既存の Lattice Radiant プロジェクトを開くには、メニューバーから [File] → [open] → [Project…] の順に選択するか、スタートページ・ビュー上部の Project セクションで大きな ”Open...” アイコン をクリックします。”Recent Project List” セクションには直近に作業したプロジェクト名が表示されています（図 2-9 の右枠）ので、開きたいプロジェクトがこの中にあれば、そのプロジェクト名をクリックしてもオープンできます。表示するプロジェクト履歴数は、オプション設定で指定することができます（第 3.5.1 項参照）。アイコ

ンをクリックした場合は、立ち上がるファイルブラウザーで *.rdf ファイルを指定します。

図 2-9. 直近プロジェクトのオープン



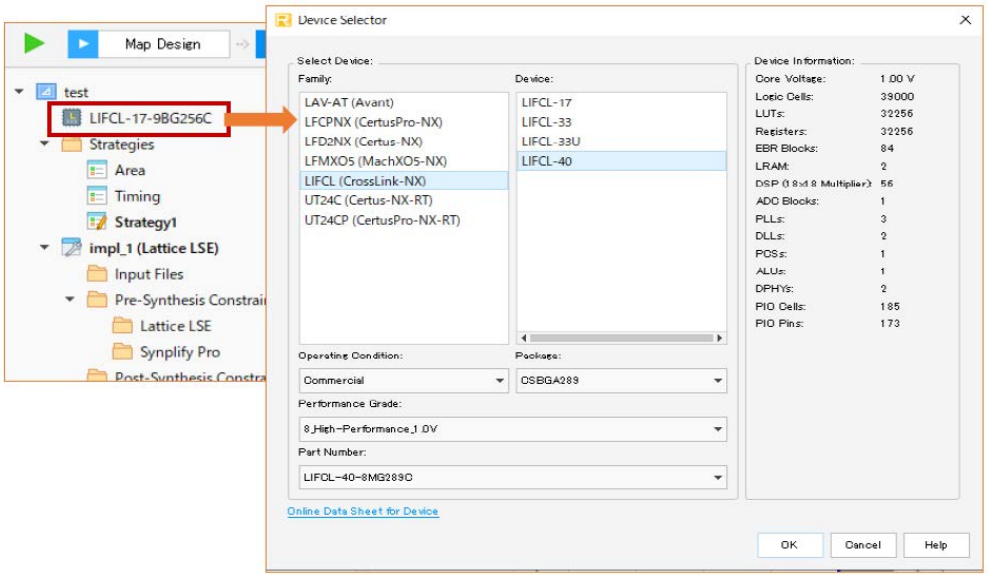
2.2.3 プロジェクトのクローズ

プロジェクトをクローズする場合は、メニューバーから [File] → [Close Project] を選択します。他のプロジェクトをオープンするか、Lattice Radiant を終了しても自動的にクローズされます。メニューの [File] → [Close] や [File] → [Close All] ではプロジェクトはクローズされません。

2.2.4 プロジェクト生成後の設定変更

2.2.4.1 ターゲット・デバイスの変更

図 2-10. ターゲットデバイスの変更方法 1



ターゲットとするデバイスや、デバイスの論理規模（バルク）、或いはパッケージやスピードグレード（SG）など、このような変更の必要がある場合の操作方法は二通りあります。

第一の方法はファイルリスト・ビュー内でプロジェクト名の下に表示されているデバイス名をダブルクリックすることです（図 2-10）。これでプロジェクト作成時のデバイス選択と同じ ”Select Device” ウィンドウを表示させて変更します。第二の方法は Radiant のメニューから [Project] → [Device...] と選択することで同じウィンドウを表示させます。

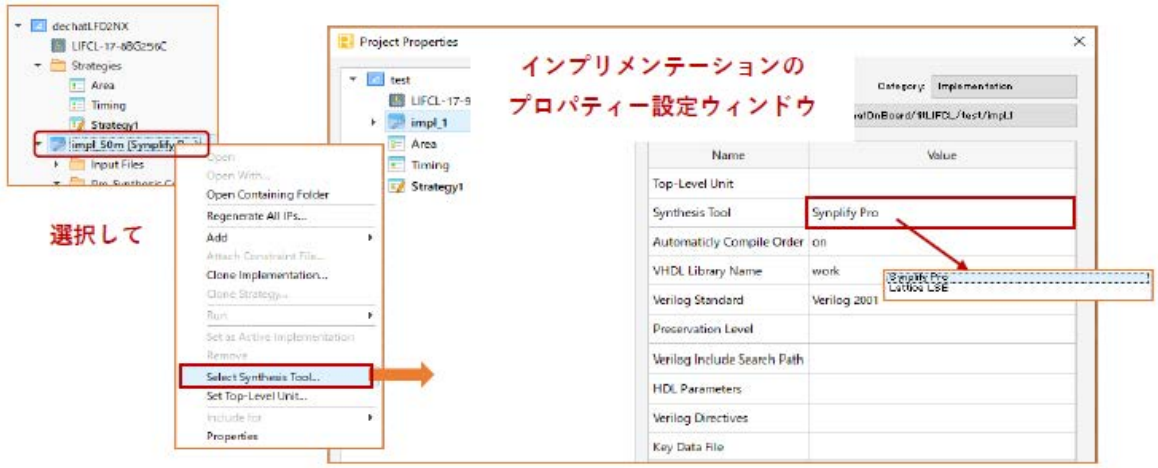
ターゲット・デバイス / パッケージ /SG はプロジェクトで一つのみ指定可能です。インプリメンテーション毎に異なる設定はできません。変更すると、論理合成プロセスからの再実行になります。また、モジュール生成ツール（IP Catalog）で生成したマクロや IP がインポートされている場合、再生成を求められるケースがあります。

2.2.4.2 プロジェクト属性の変更

” アクティブ ” なインプリメンテーションについては、プロジェクト生成時に選択した論理合成ツールなどの属性を変更することができます。Radiant のメニューから [Project] → [Property Pages] と選択する、或いはインプリメンテーション名を選択して右クリックすると表示されるメニューから ”Properties” を選択し、”Project Properties” ウィンドウを表示させます。「Top-Level Unit」はトップレベルの指定、「Verilog Include Search Path」は Verilog RTL 記述内の ” ‘include” 指定しているファイルの所在の設定、「HDL Parameter」は RTL 記述内のパラメータ指定、「Verilog Directive」は ” ‘define” ディレクティブで定義するキーワードの指定と同等のことがそれぞれ可能です。

論理合成ツールの変更は、「Synthesis Tool」セルを二回クリックすると現れるプルダウンから選択します（図 2-11 右）。また、Radiant のメニューから [Project] → [Active Implementation] → [Select Synthesis Tool...] と選択するか、インプリメンテーション名の右クリック表示メニューから [Select Synthesis Tool...] を選択しても同様です（図 2-11 左）。

図 2-11. 論理合成ツールの変更



論理合成ツールはインプリメンテーション毎に異なる指定ができます。また、変更すると論理合成プロセスからの再実行になります。また、モジュール生成ツール（IP Catalog）で生成したマクロや IP がインポートされている場合、再生成を求められるケースがあります。

2.2.5 プロジェクトの複製

プロジェクトを複製するためには、メニューから [File] → [Save Project As...] を選択して表示されるウィンドウで、保存フォルダと別名のプロジェクト名称を指定します。

2.2.6 アーカイブ・プロジェクトの作成

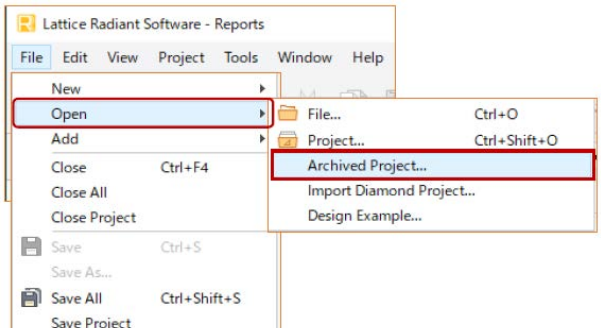
プロジェクトを保存する場合や何らかの都合でフォルダを移動させる場合等に、プロジェクトのアーカイブを作成することができます。メニューバーから [File] → [Archive Project...] を選択すると、”Archive Project”

ウィンドウが開きます。このウィンドウでアーカイブ・ファイル（.zip）の保存先と名称を指定します。アーカイブ・ファイルであることが容易に判別できるような名称、例えば ”_archv.zip” のようにすることを推奨します。

Radiant の環境設定オプションの一つである「Archive all files under the Project directory when archving project」がデフォルトでイネーブルされています（図 3-13、および第 3.5.1 項参照）。この状態、または ”Archive Project” ウィンドウ左下にある「Archive all files under the Project directory.」にチェックが入れると、プロジェクトとの関連に関わらずプロジェクト・フォルダ下にあるフォルダとファイル全てが含まれるアーカイブが作成されます。チェックが入っていない場合で、環境設定オプションがデフォルトからディセーブルに変更された場合は、プロジェクトの復元に必要な関連フォルダとファイルのみの、サイズの小さいアーカイブ・ファイルが作成されます。FAE のテクニカルサポートにプロジェクトを送付して技術支援を受ける場合などに、後者のアーカイブ機能を用いると、小さいファイルサイズでやりとりが容易になります。

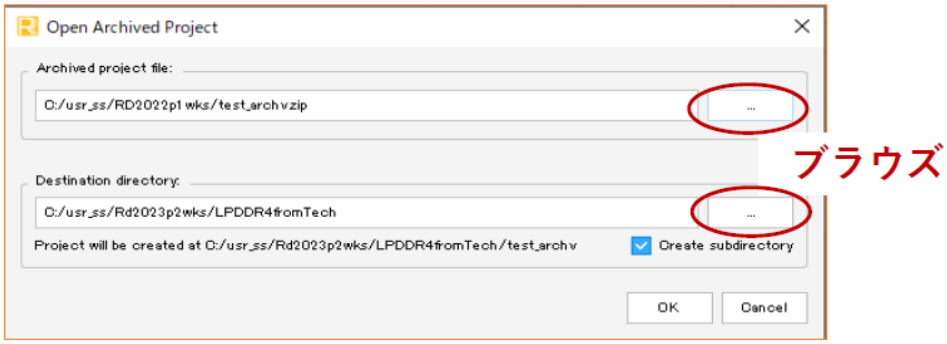
なお、予期しないトラブルを防ぐため、アーカイブ後の .zip ファイルは Windows 対応の解凍ツールを用いて unzip するのではなく、メニューから [File] → [Open] → [Archived Project...] を選択して（図 2-12）復元するようにします。立ち上がるウィンドウで（図 2-13）、アーカイブ・ファイルと復元先のフォルダーを「Destination directory」セルにブラウズ後指定して『OK』をクリックします。

図 2-12. アーカイブ・プロジェクトの復元操作 1



復元先のファイル / フォルダ構造は、アーカイブ元のそれとは異なり、ツール所定の様式で展開されます。

図 2-13. アーカイブ・プロジェクトの復元操作 2



なお、当該デザインにモジュール生成ツールで作成した ”ROM モジュール ” が含まれている場合は注意が必要です。ROM 生成には必ずメモリ初期化用のテキスト・ファイル ”.mem” が必要です（第 4.6.1 項参照）。元々のプロジェクト・フォルダー下に *.mem が存在し、かつ全ファイルを全てアーカイブに含める設定以外では、アーカイブ zip には含まれないことにご留意ください。

2.3 インプリメンテーションの管理

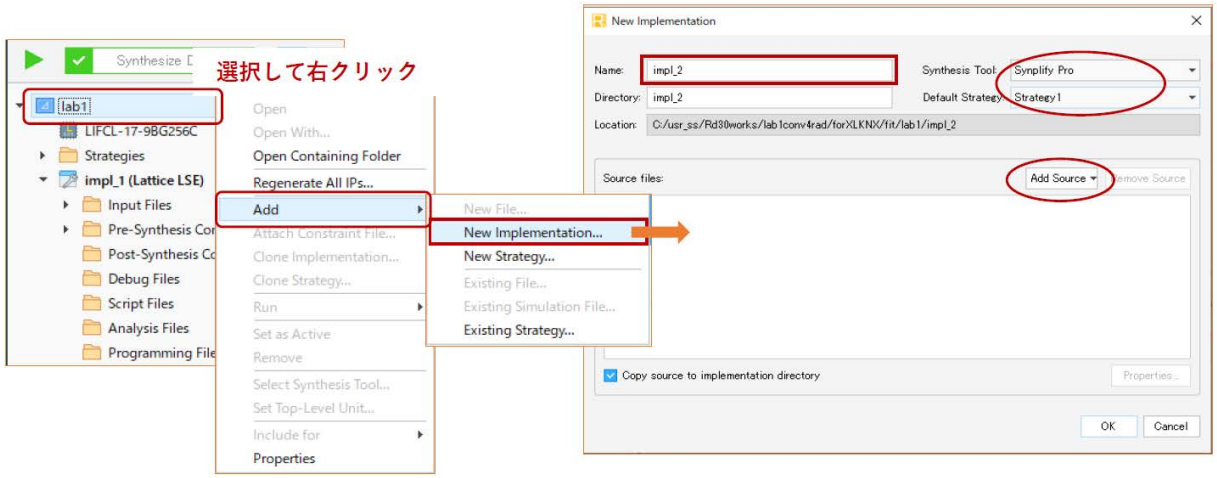
「インプリメンテーション」とは、前述のとおりデザインの管理単位であり、デザイン実装に関わる全てのファイルを含んだものです。Radiant では、単一プロジェクト内に複数のインプリメンテーションを定義することができます。それぞれのインプリメンテーションで異なるソースファイルや制約ファイル、あるいはオ

プションを用いて論理合成や配置配線を行うことができます。Radiant では単一プロジェクト内で最適な実装の推敲を行うことが容易です。

2.3.1 新規インプリメンテーションの追加

インプリメンテーションは新規プロジェクトを作成した際に必ず 1 つ作成されます（ユーザーが命名）。追加の新規インプリメンテーションが必要な場合、メニューバーから [File] → [New] → [Implementation...] の順に選択するか、ファイルリスト・ビューの一番上に表示されているプロジェクト名を右クリックし [Add] → [New Implementation...] の順に選択します（図 2-14 左）。これで ”New Implementation” ウィンドウが表示されます（図 2-14 右）。

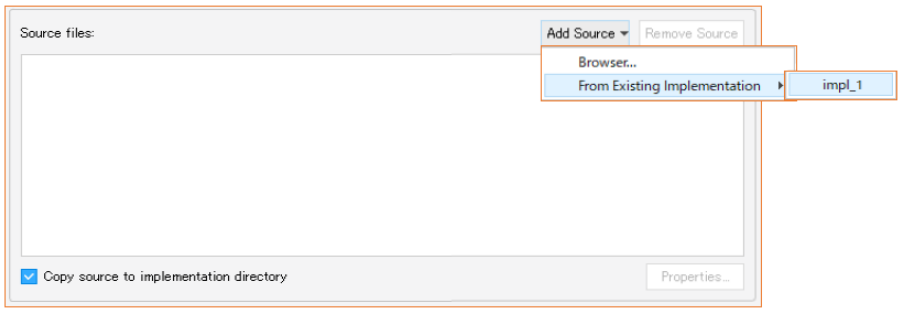
図 2-14. インプリメンテーションの追加



「Name:」セルは追加するインプリメンテーションの名称、「Directory:」セルはインプリメンテーションで使用するファイルを格納するフォルダ名です。名称を先に入力するとフォルダ名も自動的に同じものが入力されますが、変更することもできます。

「Synthesis Tool」セルと「Default Strategy:」セルは、それぞれ右側の▼アイコンをクリックして適切なものを選択します。インプリメンテーション作成後に変更することも可能です。

図 2-15. 既存インプリメンテーションからのソースのインポート例



インプリメンテーション作成時に『Add Source』ボタンをクリックしてソースファイルのインポートができます。プルダウンメニューで [Browser...] と [From Existing Implementation] が表示されます。ブラウザを使用して既存のファイルを選択する場合は前者を選択します。既存のインプリメンテーションにインポートされているソースを選択する場合は、後者を選択しますが、すると既存のインプリメンテーション名が全て候補としてリストされますので、その中から 1 つを選択します（図 2-15。この図では一つのみ）。

この際、ウィンドウ左下の「Copy source to implementation directory」にチェックが入っていると、選択したソースは新規に作成したインプリメンテーションのフォルダにコピーされ、それがインポートされます。

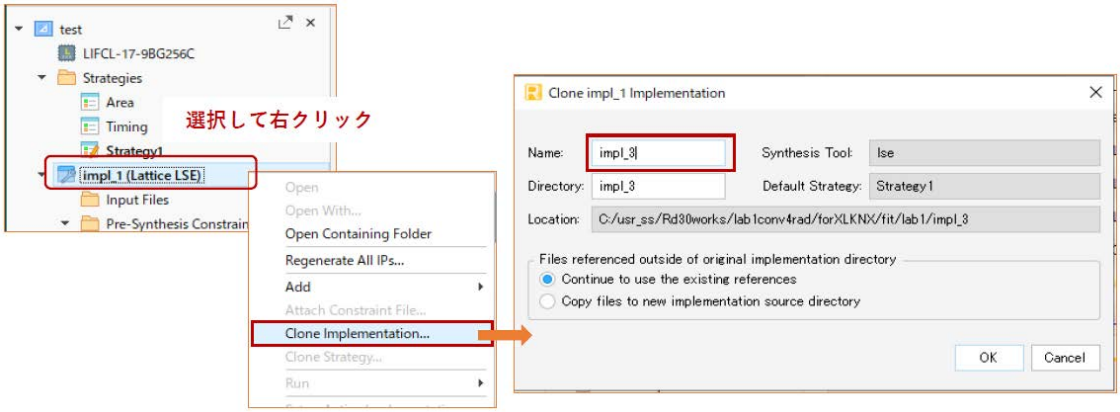
チェックが入っていないと選択したオリジナルのソースファイルがインポートされます。コピーしたものをインポートする場合、オリジナルのソースファイルは変更されませんので、他のインプリメンテーションのソースには影響しません。逆にオリジナルのソースファイルをインポートする場合、複数のインプリメンテーションが同一ソースを参照していれば、更新された際に全てが影響されます。

必要なソースを選択後、『OK』ボタンをクリックして設定は完了です。”インプリメンテーション・フォルダ”が自動作成され、各種ファイル / サブフォルダがその下に置かれます。

2.3.2 インプリメンテーションの複製（クローン）

Radiant にはインプリメンテーションの複製（”クローン・インプリメンテーション”）を容易に生成する機能があります。

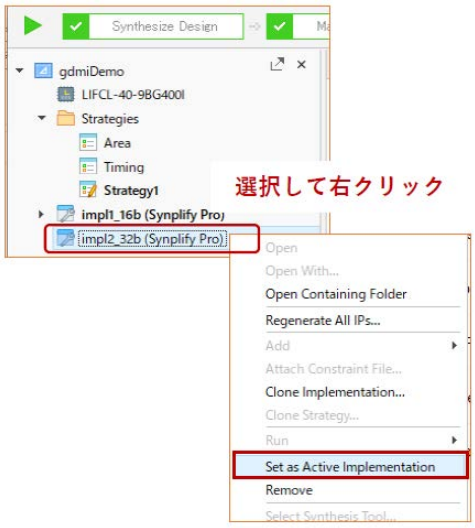
図 2-16. インプリメンテーションの複製



複製の元になるインプリメンテーションを選択後、右クリックで[Clone Implementation...]を選びます（図 2-16 左）。”アクティブ”（次項で記述）でないインプリメンテーションでも選択は有効です。その後表示されるウィンドウ（図 2-16 右）で複製後の名称を「Name:」セルに入力します。下部でソースファイルを参照するか、コピーするかの指定も行います。インプリメンテーションがフォルダ一括で複製され、付与した名称になります。

2.3.3 インプリメンテーションのアクティブ化

図 2-17. インプリメンテーションのアクティブ化



複数のインプリメンテーションを持つプロジェクトでは、”アクティブ”なインプリメンテーションは一つだけです。アクティブとは、論理合成以下各処理プロセスの対象ということです。従って意図する処理対象のインプリメンテーションは、初めにアクティブ化してから、またはアクティブであることを確認してからすべての操作を行う必要があります。

ファイルリスト・ビューで、アクティブ化したいインプリメンテーション名を選択後に右クリックし、メニューから [Set as Active Implementation] を選択します。インプリメンテーションがアクティブになったことを示すボールド字体になります。

2.3.4 インプリメンテーションの削除

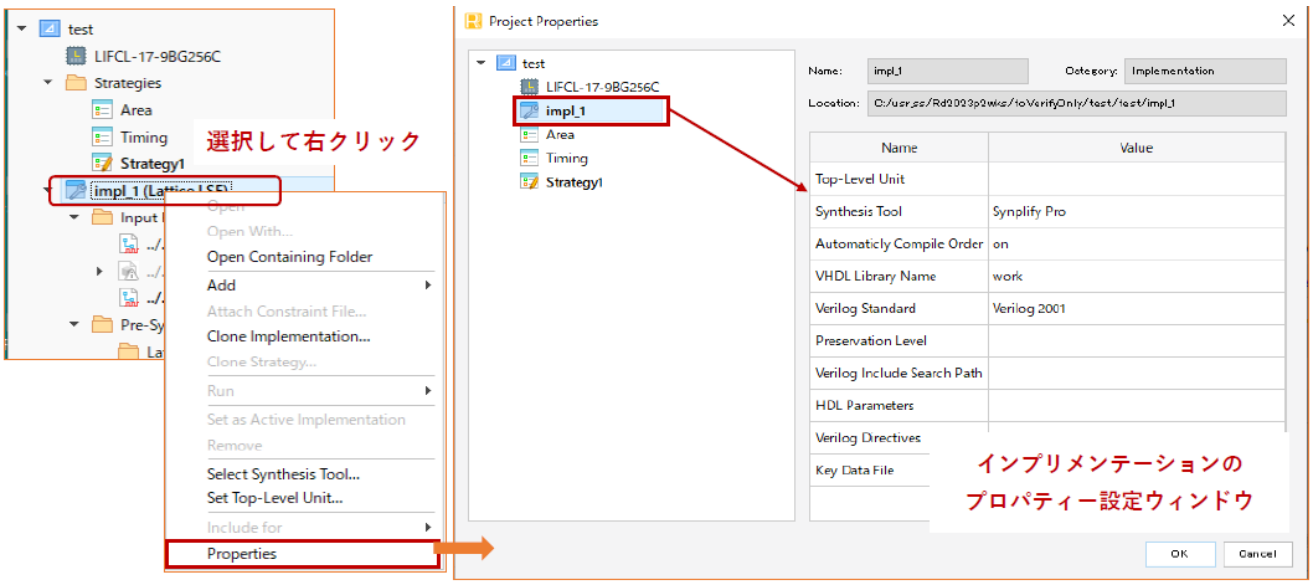
不要なインプリメンテーションをプロジェクトから削除するには、まずそのインプリメンテーションを非アクティブ化（他のインプリメンテーションをアクティブ化）します。次にそのインプリメンテーション名を右クリックし、[Remove] を選択するか（図 2-17 で赤枠の下）、またはインプリメンテーションを選択した状態でキーボードの [Delete] キーを押します。

アクティブなインプリメンテーションは削除できません。なお、”削除”はインポート情報の削除であり、ファイルやフォルダ自体は削除されません。

2.3.5 インプリメンテーションのプロパティ設定

プロパティ設定はメニューバーから [Project] → [Property Pages] の順に選択するか、ファイルリスト・ビューでインプリメンテーション名を右クリックして [Properties] を選択します（図 2-18 左）。これにより ”Project Properties” ウィンドウが立ち上がります（図 2-18 右）。

図 2-18. インプリメンテーションのプロパティ設定



プロパティ項目の意味は以下の通りです。インプリメンテーション名が左枠で選択された状態で、右枠内でそれぞれ確認・変更します。なお、非アクティブなインプリメンテーションを選択した場合は、何も表示されませんので、変更は一切できません。

Top Level Unit

デザインソースの最上位階層のエンティティ（VHDL）/ モジュール（Verilog）名を入力できます。ほとんどの場合、最上位階層名はインポートしたソースから自動的に検出します。しかし VHDL/Verilog HDL 混在の場合など、最上位階層が認識されないケースがあります。そのような場合にここで指定します。セルをクリックすると、右端に▼印が現れ、プルダウンでその候補が表示されます。

Synthesis Tool

選択されている論理合成ツールが表示されますが、変更することができます。

Automaticly Compile Order

2023.2 から追加されました。デフォルトは 'on' で、ソースファイルのコンパイル順序を自動的に設定します。このウィンドウで 'off' にすることもできますが、Input Files セクションでファイルを上下にドラッグして順序を変える操作をすると、オフにするかどうかのメッセージがポップアップします。

VHDL Library Name

インプリメンテーションで、デフォルトで使用する VHDL ソースのコンパイル先ライブラリー名の設定です。デフォルトでは work ライブラリーとしてコンパイルされます。VHDL ソース単位でコンパイル先のライブラリー名が異なる場合は、デザインソースのプロパティで設定します（第 2.6.6 項参照）

Verilog Standard

デフォルトは Verilog 2001 で、Verilog 95 と SystemVerilog が選択できます。

Preservation Level

2023.1 から追加されたブロックベース・デザイン（Block Based Design）に関するものですが、2023.2.1 時点では何も設定できません。

Verilog Include Search Path

Verilog HDL ソース内で相対パス記述でインクルードしているファイルを検索するパスの設定です。インクルードされるファイルが、ソース内にフルパスでファイル名が記述されていたり、プロジェクト内の適切なフォルダに保存されていたりする場合は、設定する必要はありません。

HDL Parameters

HDL ソースにグローバルで作用するパラメータ（generic/VHDL、parameter/Verilog）を指定することができます。VHDL ソースによるプロジェクトで Reveal デバッガを使用する場合、generic 宣言しているパラメータがあれば、全てここに列記する必要があります（第 11.2.1 項参照）。複数定義する場合は以下のようにセミコロンで分離します（改行はできません）。

-- 記述例　　g_bus_width=16;comm_id=8;

Verilog Directives

Verilog HDL の "define" に相当するコンパイラ指示子を指定できます。複数定義する場合はセミコロンで分離します（改行はできません）。

2.4 ストラテジーの管理

ストラテジーは前述のように各プロセスの制約設定（Constraints、Preferences）やオプション設定を一元管理する機構です。Radiant では、プロジェクト内に複数のストラテジーを持ち、それぞれのストラテジーでは異なるオプション設定を定義することができます。インプリメンテーションごとに "アクティブ" な（有効な）ストラテジーは一つだけです。

2.4.1 デフォルト・ストラテジー

プロジェクトを作成した際に、以下の 3 つのストラテジーが作成されています。Strategy1 以外は編集することができません。

Area　　：必要なリソース（スライス）数が最小になるように設定されたストラテジー
Timing　　：タイミング要求を優先するよう設定されたストラテジー（全てデフォルト）
Strategy1　　：Timing と同じ（ユーザーが編集可能）

ストラテジーを追加するには、新たに作成する方法と、既存のストラテジーを複製する方法があります。

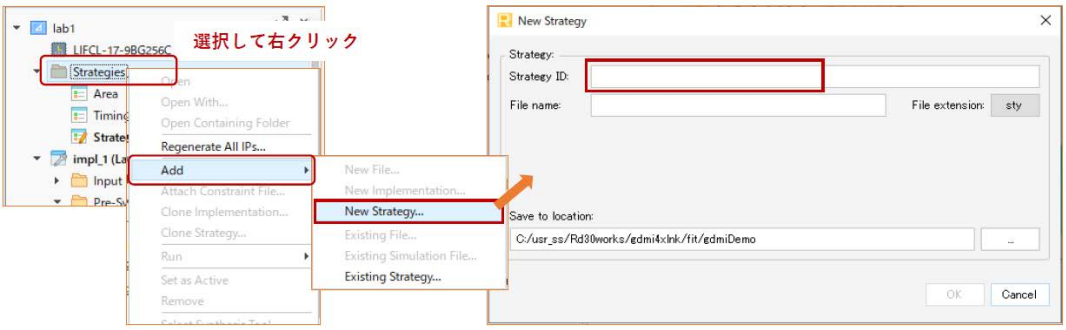
2.4.2 新規ストラテジーの作成

新規作成する場合は、メニューバーから [File] → [New] → [Strategy...] の順に選択するか、ファイルリスト・ビュー内の [Strategies] 表記を選択して右クリックし、[Add] → [New Strategy...] と選択します（図 2-19 左）。これで、”New Strategy” ウィンドウが表示されます（図 2-19 右）。

New Strategy ウィンドウの「Strategy ID:」セルにストラテジー名を入力します。Radiant 上にはこの名称で表示されます。「File name:」セルにはデフォルトでストラテジー ID と同じものが自動入力されますが、変更できます。「Save to Location:」セルはストラテジー設定ファイルを保存するフォルダです。デフォルトでプロジェクトフォルダが設定されていますが、変更することも可能です。

全ての設定完了後、OK ボタンをクリックすると、新規ストラテジー（設定は全てデフォルト）が作成され、”< ストラテジー名 >.sty” というファイル名で指定フォルダに保存されます。

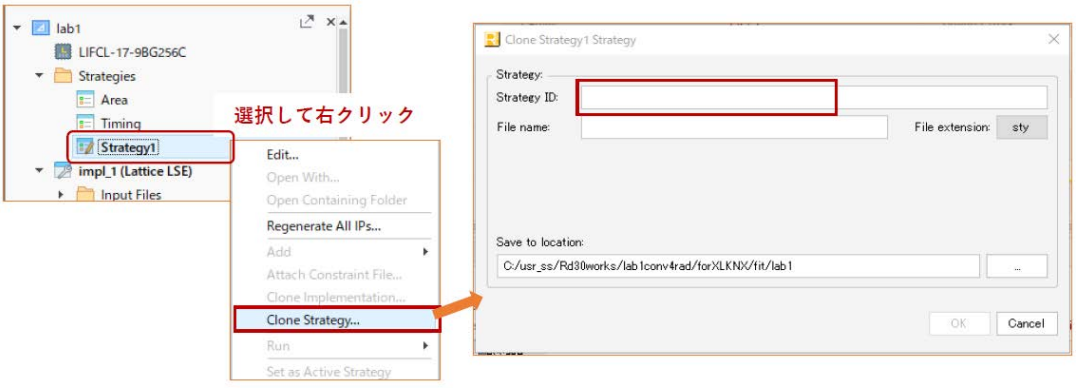
図 2-19. 新規ストラテジーの追加



2.4.3 ストラテジーの複製

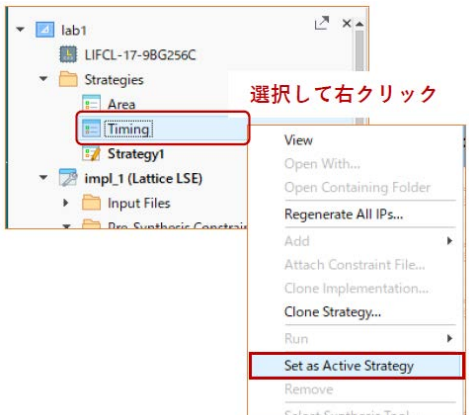
Lattice Radiant には既存ストラテジーの複製 ” クローン（Clone）” を作成する機能があります。複製したいストラテジーを選択して右クリックし、表示されるメニューから [Clone Strategy...] を選択します（図 2-20 左）。これで、”Clone *** Strategy” ウィンドウ（’***’ は複製元のストラテジー名）が立ち上がりますので、新ストラテジー名を入力します（図 2-20 右）。

図 2-20. ストラテジーの複製



Clone Strategy ウィンドウの設定は、新規ストラテジーを追加する場合と同じですが、作成されるストラテジーの設定値が複製元と同じになっています。プロジェクト作成時にデフォルトで生成される設定変更できないストラテジーである ”Area” や ”Timing” の場合でも、複製版は変更が可能です。

図 2-21. ストラテジーのアクティブ化



2.4.4 適用するストラテジーの選択（アクティブ化）

インプリメンテーションでは、”アクティブ化”されている１つのストラテジーのみが有効です。このため”ストラテジーのアクティブ化”がすなわち適用するストラテジーの選択になります。

ストラテジーをアクティブ化するには、ファイルリスト・ビューで当該ストラテジーをクリックして選択後、右クリックすると表示されるメニューから [Set as Active Strategy] を選択します（図 2-21）。アクティブ化されたストラテジーは太字で表示され、以後のプロセス処理に適用されます。

2.4.5 ストラテジーの削除

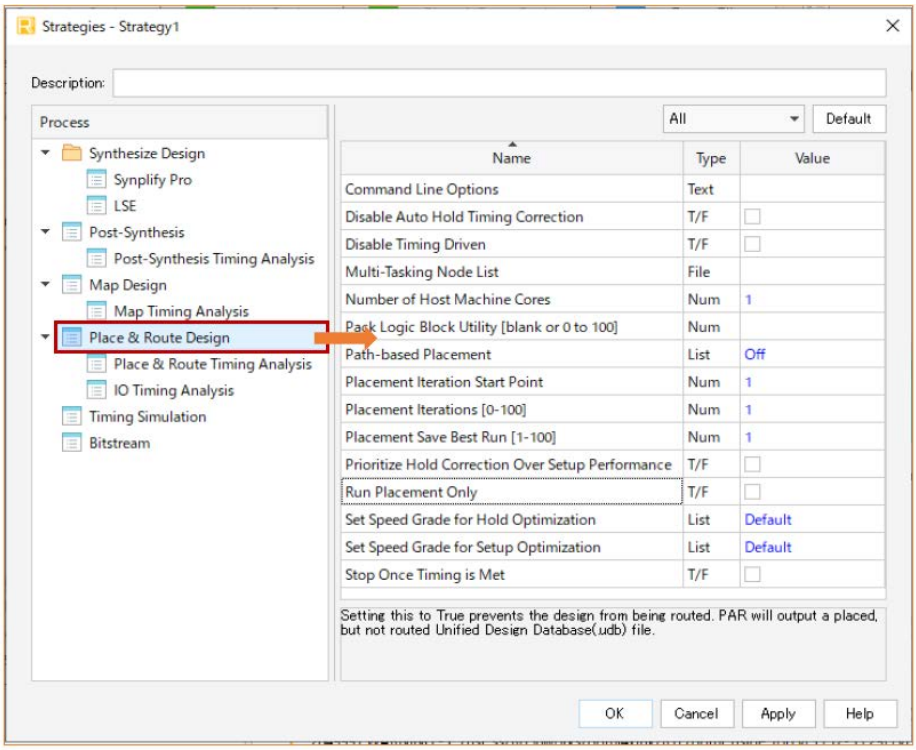
プロジェクトで不要になったストラテジーは削除できます。ファイルリスト・ビューで削除したいストラテジーを右クリックして選択後、表示メニューから [Remove] を選択するか、キーボードの [Delete] キーを押すと、ストラテジーがプロジェクトから削除されます。”プロジェクトから削除”することはインポート情報の削除であり、ストラテジー・ファイルの削除ではありません。

なお、非アクティブなインプリメンテーションでもアクティブなストラテジーは削除できません。また、プロジェクト作成時に自動生成されているものは、全て削除できません。

2.4.6 ストラテジー・オプションの編集

ストラテジー・オプション項目を変更・編集するためには、まずファイルリスト・ビューでストラテジー名をダブルクリックするか、または意図するストラテジーを右クリックして選択後、表示されるメニューから [Edit] を選択します。ストラテジー・オプションの設定ウィンドウが立ち上がります（図 2-22）。

図 2-22. ストラテジー・オプション設定ウィンドウ（PAR）



このウィンドウで各プロセスの定義済みオプション設定を変更することができます。左枠で意図するプロセスかサブプロセスを選択します。オプション項目の詳細については、各プロセス章の記述をご参照ください。

2.5 設計制約ファイルの管理

Radiant の制約ファイルに三つのタイプがあります。Radiant プロジェクトではインプリメンテーションごとに 1 つ以上の sdc/pdc 制約ファイルが存在できますが、″アクティブ″なものは一つです。或いは制約ファイルが一切なくても処理プロセスに問題はありませんが、実デザインの実装ではピン配置指定や動作クロックに対するタイミング制約は必要ですので、実質的に最低限 *.sdc/*.pdc 各一つはインポートして適用するのが通常です。

・Pre-Synthesis Design Constraints (*.sdc)

論理合成に適用される業界標準の Synopsys 設計制約書式の制約です。論理合成ツールとして Synplify Pro を選択している場合、*.fdc を指定することもできます。

・Post-Synthesis Design Constraints (*.pdc)

論理合成後の処理に適用されるタイミング制約や、ピン配置していなどの Radiant 固有の設計制約です。

・Lattice Design Constraints (*.ldc)

Lattice 独自の制約で sdc (ver.2.0) と論理合成アトリビュートの組み合わせたものです。IP などを生成すると付随して生成されます。それらに含まれるピン配置やピン属性指定、一部のタイミング制約についてはプロジェクトとしての制約に自動的に反映されます。

それぞれの作成方法等については、第 15 章、第 16 章をご参照ください。

2.5.1 既存の制約ファイルのインポート

既存の制約ファイルをインポートする場合は、メニューバーから [File] → [Add] → [Existing File...] と選択するか、ファイルリスト・ビューでインプリメンテーション・セクション下 [Pre-Synthesis Constraint Files] や[Post-Synthesis Constraint Files]行を選択後右クリックし、表示されるメニューから[Add] → [Existing File...] の順に選択します。図 2-23 は sdc のインポート例ですが、他も同様です。

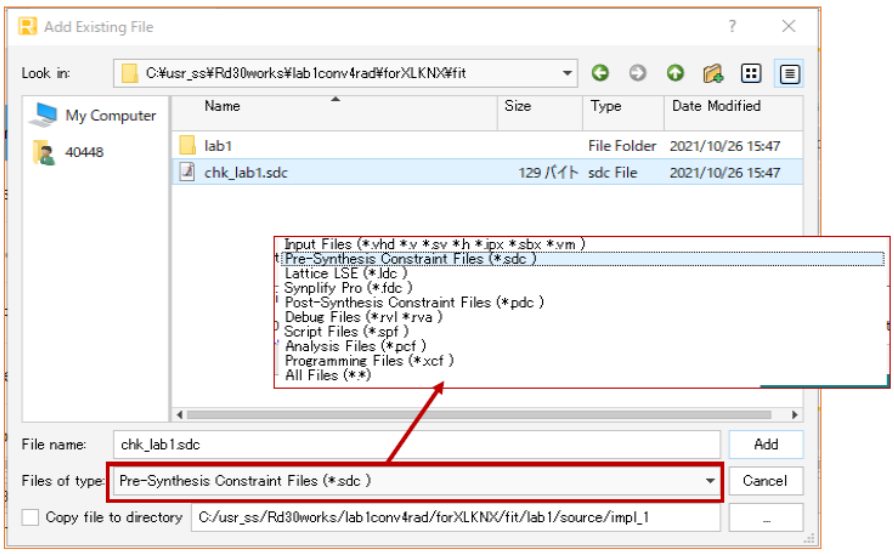
図 2-23. 既存 sdc 制約ファイルのインポート例

”Add Existing File” ウィンドウ（図 2-24）が表示されます。”Files of Type” として [Pre-Synthesis Constraint Files（*.sdc）] が選択されて立ち上がりますので（必要に応じてタイプを変更した後）、ブラウズして適切なファイルを選択し、『Add』ボタンをクリックします。

ここで、ウィンドウ左下の [Copy file to Implementation’s Source directory] にチェックが入っていると、インプリメンテーション・フォルダにコピーが作成され、これがインポートされます。チェックが入っていない場合は、選択したファイルがインポートされます。

なお、*.fdc をインポートする場合は、[Pre-Synthesis Constraint Files] セクション下にある [Synplify Pro] を選択後右クリックして同様に指定します。

図 2-24. インポートする制約ファイルの選択



2.5.2 適用する制約ファイルの選択（アクティブ化）

インプリメンテーションに複数の制約ファイルが Pre-/Post- それぞれにインポートされている場合でも、有効（アクティブ）なものは一つです。アクティブ化するためには、ファイルリスト・ビューでアクティブでない制約ファイルを選択後右クリックし、表示されるメニュから [Set as Active Preference File] を選択します。アクティブ化された制約ファイルは太字で表示され、以後のプロセス処理に適用されます。

複数のファイルがインポートされていても、すべて非アクティブにできます。

2.5.3 制約ファイルの削除

不要になった制約ファイルは、プロジェクトから削除することができます。ファイルリスト・ビューで削除したい制約ファイルを選択後右クリックし、表示されるメニューから [Remove] を選択するか、選択した状態でキーボードの [Delete] キーを押します。”プロジェクトから削除 ” はインポート情報の削除であり、ファイル自体は削除されません。

アクティブな制約ファイルでも削除できますので、ご注意ください。

2.6 RTL ソースファイルの管理

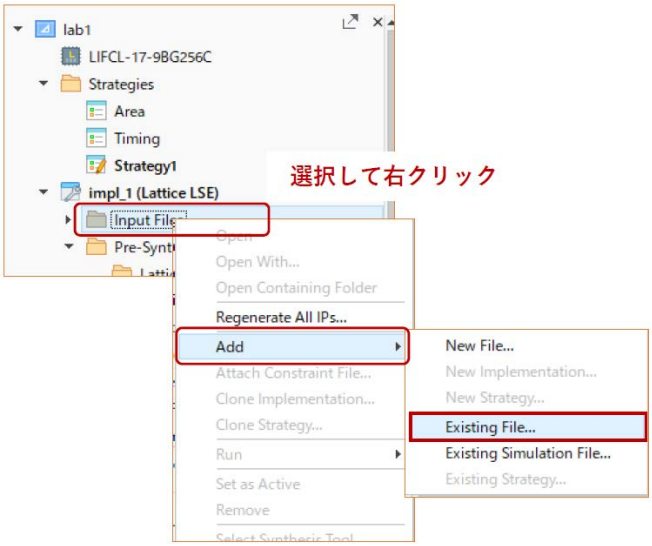
2.6.1 新規 RTL ソースファイルの作成

新規 RTL ソースファイルを作成する場合、メニューバーから [File] → [New] → [File...] と選択するか、もしくはファイルリスト・ビューでインプリメンテーション・セクション下の [Input Files] 行を選択後右クリックし、表示されるメニューから [Add] → [New File...] の順に選択します。”New File” ウィンドウが起動しますので、適宜意図するタイプを選択して作業を開始します。

一般的には、RTL ソースファイルを Radiant のようなフィッティング・ツール上で新規作成するのは非常にまれですので、ここでの詳細説明は割愛します。

2.6.2 既存の RTL ソースファイルのインポート

図 2-25. 既存のソースファイルのインポート



作成済みの RTL ソースファイルをインプリメンテーションにインポートする場合は、メニューバーから [File] → [Add] → [Existing File...] の順に選択するか、もしくはファイルリスト・ビューでインプリメンテーション・セクション下の [Input Files] 行を選択後右クリックし、表示されるメニューから、[Add] → [Existing

File...] の順に選択します（図 2-25）。”Add Exiting File” ウィンドウが表示されますので、ブラウズして指定します。

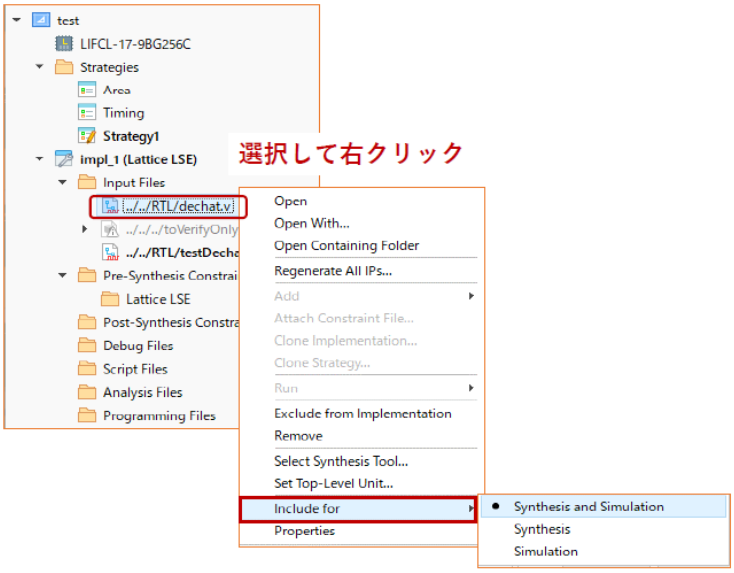
”Add Exiting File” ウィンドウでは、必要に応じて適宜 ”Files of Type” を選択し直します。同ウィンドウ左下の「Copy file to Implementation’s Source directory」にチェックが入っていると、インプリメンテーション下の ”source フォルダ ” にコピーが作成され、これがインポートされます。チェックが入っていない場合は、選択したファイルがインポートされます。

なお、モジュール生成ツール IP Catalog（参照）で生成済のモジュールをインポートする場合は、RTL 記述のトップファイル（.v）ではなく、それらの定義ファイル（拡張子 ”.ipx”）を指定します。

2.6.3 対象プロセスの指定

インポートした RTL ソースファイルは、デフォルトで ” シミュレーションおよび論理合成 ” の対象として設定されますが、これをシミュレーションのみや論理合成のみの対象に変更できます。

図 2-26. RTL ソースファイルの対象プロセスを指定



当該 RTL ソースファイルを選択後、右クリックして表示されるメニューから [Include For] を選択するとリストが表示されますので、この中から 1 つを選択します。

[Synthesis] または [Synthesis and Simulation] を選択したソースは、論理合成の対象となります。[Simulation] または [Synthesis and Simulation] を選択したソースは、”Simulation Wizard”（第 10 章参照）で論理シミュレーション用のコンパイルスクリプトが自動作成される時に、デフォルトでコンパイル対象として含まれます。

使用例

- テストベンチをインポートして [Simulation] に設定
- 論理合成用のブラックボックス・モデル（ファイル）をインポートして [Synthesis] に設定

2.6.4 使用しないソースファイルの扱い

RTL ソースファイルは、インポートしていてもインプリメンテーションごとに処理の対象から除外することができます。例えば（内部の記述が異なるが）同じ entity/module 名のファイルを複数インポートしておき、インプリメンテーション毎に使用するファイルを切り替えて使用することができます。

ファイルリスト・ビューの [Input Files] セクションで、処理対象から外したいファイルを選択した後、右クリックして表示されるメニューから [Exclude from Implementation] を選択すると、そのファイルは処理対象から除外されます。対象から除外されたファイルは、薄い文字で表示されます。

この設定は選択されているインプリメンテーションでのみ有効で、他のインプリメンテーションには反映されません。また”Remove”とは異なり、インポート自体は解除されません。

一度除外したファイルを再び処理対処に戻す場合は、同様に右クリックして表示されるメニューから[Include in Implementation]を選択します。

2.6.5 RTL ソースファイルの削除

不要になった RTL ソースファイルは、プロジェクトから削除できます。[Input Files]セクションで削除したいソースファイルを選択後、右クリックして表示されるメニューから[Remove]を選択するか、選択した状態でキーボードの[Delete]キーを押すと、プロジェクトから削除されます。ファイル自体は削除されません。

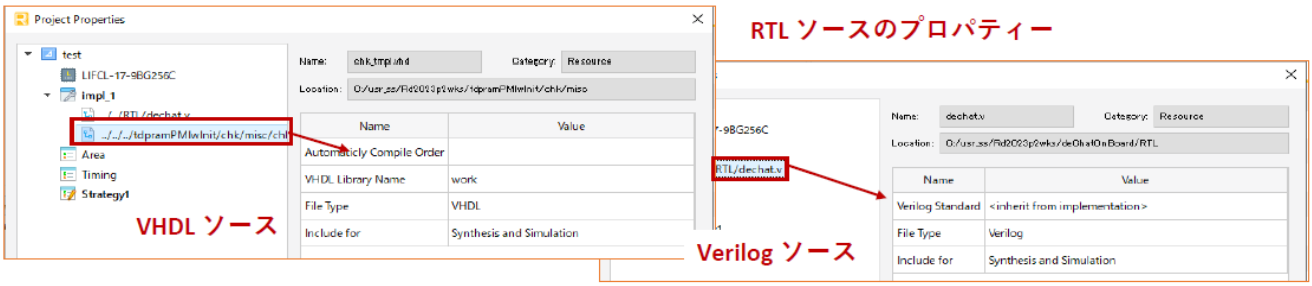
また、インポート済みの RTL ソースファイルを移動した場合など、ツールが見つからないと判断すると、当該ファイル名を取り消し線で表示します。削除するか、移動したものを元に複製するか戻すなどの対処をします（任意）。

2.6.6 RTL ソースファイルのプロパティ設定

インポート済み RTL ソースファイルのプロパティとして、個別に以下の設定を行うことができます。

- VHDL ソースのコンパイル先ライブラリー名
- Verilog HDL ソース内でインクルードされているファイルの検索パス

図 2-27. Project Property ウィンドウ～RTL ソースファイルの設定



第 2.3.5 項と同様の手順で”Project Properties”ウィンドウを表示します。ソースファイルを選択してから表示しても同様です。ウィンドウ左枠でプロパティを設定したいソースファイルを選択すると、ウィンドウ右側に設定可能な項目が表示されます。図 2-27 は VHDL ソースを選択した場合の表示例です。

なお、インプリメンテーション作成時に既存のインプリメンテーションにインポートされているソースを参照した場合、これらのプロパティ設定も反映されます。従ってインプリメンテーション毎に設定し直す必要はありません。

2.6.7 RTL ソースファイルの暗号化

Radiant はインプリメンテーションにエントリーする RTL ソースファイルのモジュール（エンティティ）単位での暗号化に対応しています。

ラティスの公開鍵を用いる場合を Verilog を例にした手順は次の通りです。Verilog の”pragma”は VHDL では”protect”が相当します。

1．RTL ソース先頭に次のような pragma（Verilog）/protect（VHDL）属性の宣言文を入れます。

```
`pragma protect version=1
`pragma protect encoding=(enctype="base64")
```

テンプレートにある次の宣言は任意です。

`pragma protect author="<Your Name>"

`pragma protect author_info="<Your info>"

2．以下を RTL ソース内に記述するか、またはテキストファイル "key.txt" として用意します。

`pragma protect key_keyowner= "Lattice Semiconductor"

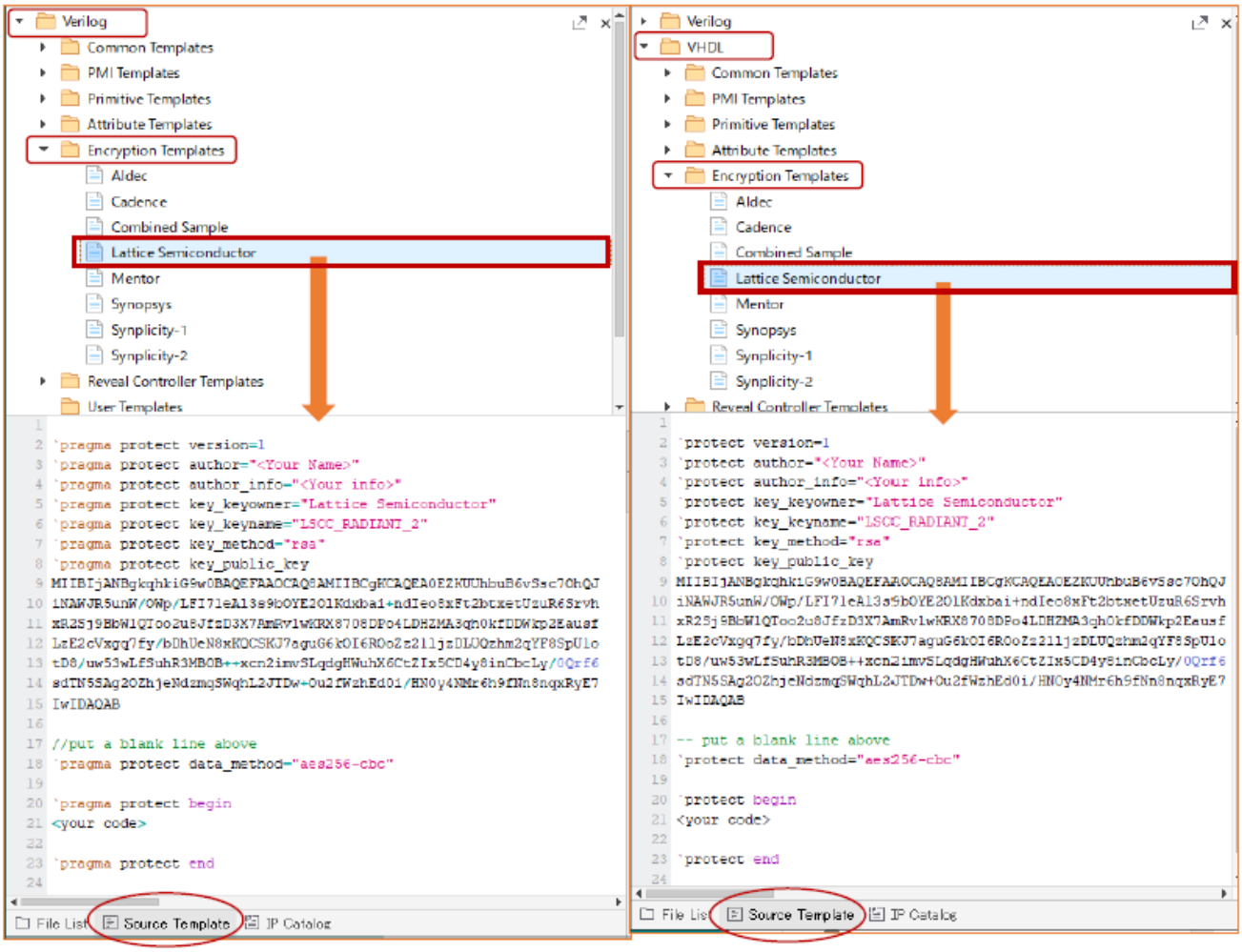
`pragma protect key_keyname= "LSCC_RADIANT_2"

`pragma protect key_method="rsa"

`pragma protect key_public_key

MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0EZKUUhbuB6vSsc7OhQJiNAWJR5unW/OWp/LFI71eAl3s9bOYE2OlKdxbai+ndIeo8xFt2btxetUzuR6SrvhxR2Sj9BbW1QToo2u8JfzD3X7AmRvlwKRX8708DPo4LDHZMA3qh0kfDDWkp2EausfLzE2cVxgq7fy/bDhUeN8xKQCSKJ7aguG6kOI6ROoZz21ljzDLUQzhm2qYF8SpU1otD8/uw53wLfSuhR3MBOB++xcn2imvSLqdgHWuhX6CtZIx5CD4y8inCbcLy/0Qrf6sdTN5SAg2OZhjeNdzmqSWqhL2JTDw+Ou2fWzhEd0i/HN0y4NMr6h9fNn8nqxRyE7IwIDAQAB

図 2-28. 暗号化記述テンプレート （Lattice Semiconductor）



3．その後に RTL ソース内に一行空けて使用するアルゴリズムを宣言します。対応するのは標準的な AES の CBC モードで、キーワードは "aes256-cbc"（デフォルト）または "aes128-cbc" です。

`pragma protect data_method="aes256-cbc"

4．RTL ソース内で暗号化する対象の記述範囲の前後に次のようなキーワードを記述します。

```
`pragma protect begin
  <your code>
`pragma protect end
```

5．TCL コンソールで以下のようなコマンドを実行します（<> 内は実際のファイル名や言語を記述）。上ステップ２でキーファイルを用意している場合はこのように ”-k” オプションで指定します。RTL 内に記述している場合は不要です。

encrypt_hdl -k <keyfile> -l <verilog | vhdl> -o <output_file> <input_file>

出力ファイル名の指定は任意です。その場合は ”<input_file>_enc.v” のような名称で生成されます。

暗号化された RTL ソースファイルは、もちろんインプリメンテーションにインポートして使用できます。

RTL ソースファイルにコピー＆ペーストできるテンプレートは Radiant GUI 操作で容易に入手できます。まず GUI 左枠のウィンドウ下部にある三つのタブから、[Source Template] を選択します。表示されるセクションから [Verilog] か [VHDL] を展開し、[Encryption Templates] を展開すると、Radiant の対応しているベンダーを含めた８つのテンプレートが用意されていることがわかります（図 2-28）。

なお、RTL ソースの暗号化では、Radiant フロー以外で論理合成として Synplify Pro 対応にする場合は、RTL ソース内に ”key_keyowner” としては ”Synplicity”、”key_keyname” として ”SYNP15_1” および ”SYNP05_001” の二つについてそれぞれの ”key_public_key” を RTL ソースファイルの冒頭に追加しておくようにします。

関連情報についてはオンラインヘルプで [User Guides] → [Securing the Design] → [HDL File Encryption Steps] をご参照ください。

更　新　履　歴

リビジョン	ページ	内　　容
2022.1	6	Rev.3.2.1 からバージョン・アップに伴い、図2-9を更新
	19	RTL暗号化について、LSEのみの対応の注記を追加
(r1)	1	デザインエントリーとして*.vmが可能なことを追記
	7	2.2.4.2項を”プロジェクト属性の変更”に変更し、論理合成ツールの変更以外についても追記
2023.2.1		バージョンアップに伴い見直し・更新
	12	VHDLでのHDL Parameters属性入力について追記

--- *** ---