ASYL+/CQ 版チュートリアル

内容

本チュートリアルでは，VHDL シンセサイザの ASYL+/CQ 版と，パソコン用プログラマブル・ロジック開発パッケージの FLEX デザイン・キット/CQ 版とを組み合わせた場合の使い方について解説します．

基本的な使い方1　一つの VHDL ソースを処理する

ここでは，一つの VHDL ソース・ファイルを ASYL+/CQ 版に入力して論理合成を行い，生成されたネットリストを FLEX デザイン・キット/CQ 版に付属する MAX+plus II バージョン 6.2 で処理する方法を示します．

1.1　ASYL+/CQ 版での操作

VHDL で設計情報を記述したらこれを論理合成します．ASYL+/CQ 版(以下 ASYL)のアイコンをダブル・クリックして ASYL を起動します(図1)．**なお，VHDL ソース・ファイル名の拡張子はvhd** とします．

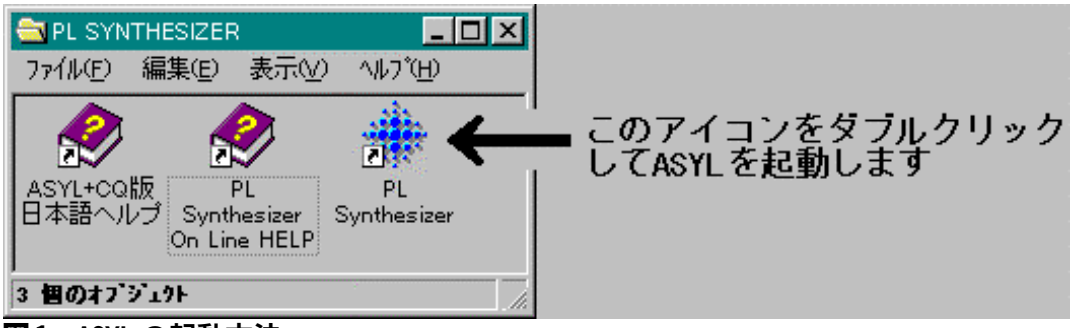


図1　ASYL の起動方法

ASYL が起動したら，[Command]-[Execute...]メニュを起動して論理合成パラメータを設定するウィンドウを開きます(図2)．

図2　論理合成パラメータ設定ウィンドウを開く

[Command]-[Execute...]を実行すると，図3 に示すウィンドウが開くので，以下のように設定します．

- **Input/Project File Name**
  このボタンをクリックするとファイル検索ができるので，作成した VHDL ソース・ファイル名を指定します．
- **Input Format**
  無条件に VHDL を選択します(CQ 版では VHDL 以外の入力フォーマットはサポートしていません)．

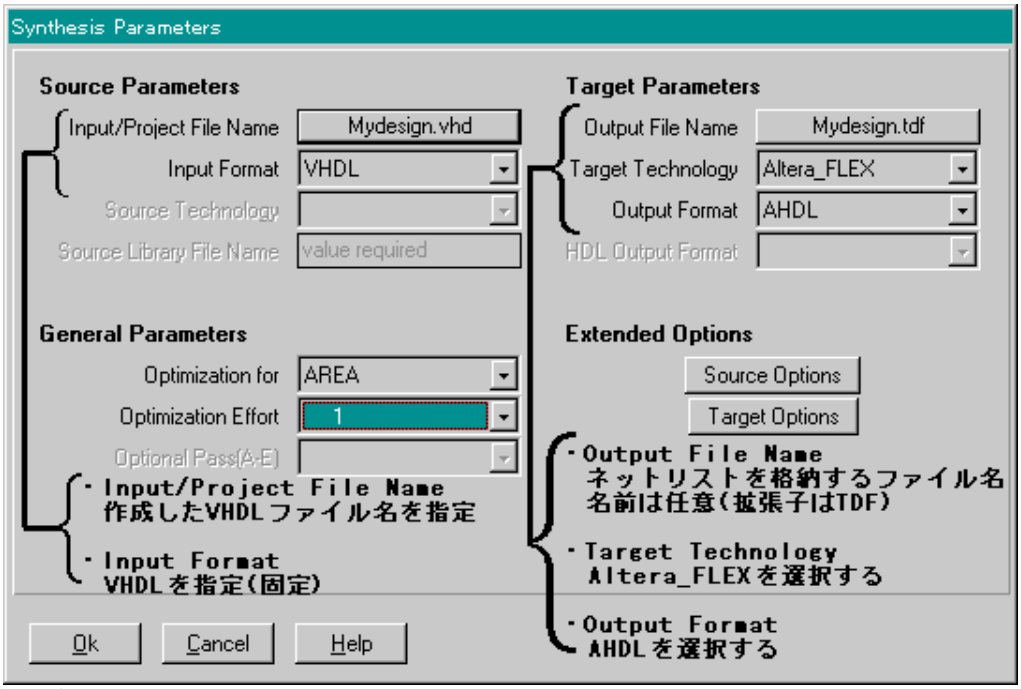


図3　各種パラメータの設定

- **Output File Name**
  ネットリストを格納するファイル名を入力します．ファイル名は任意ですが，Altera 社の FLEX8000 シリーズをターゲットとする場合，**拡張子は tdf** とします．
- **Target Technology**
  メニュから **Altera_FLEX** を選択します．
- **Output Format**
  メニュから **AHDL** を選択します．なお，ここでファイル・フォーマットを指定しても，Output File Name で出力ファイル・タイプを指定しないかぎり**拡張子が自動的に付かない**ので，注意してください．

以上の設定が終わったら Ok ボタンをクリックします．Ok ボタンをクリックすると同時に論理合成が始まります．

なおこのとき，入力したパラメータの誤りなどがあるとテキスト・ウィンドウ内にエラー・メッセージが表示されます．以下に二つのエラー・メッセージの実例を示します．

1.1.1 エラー・メッセージの例

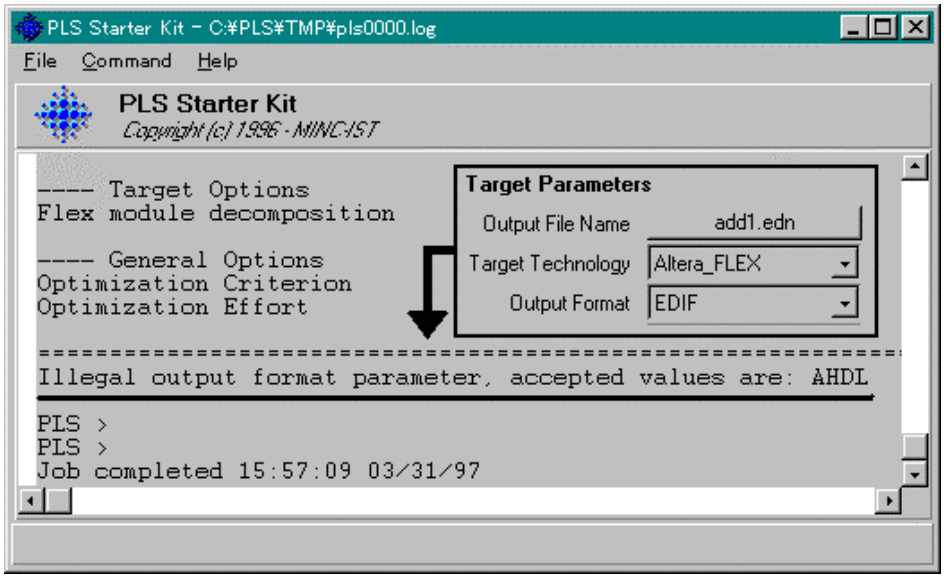
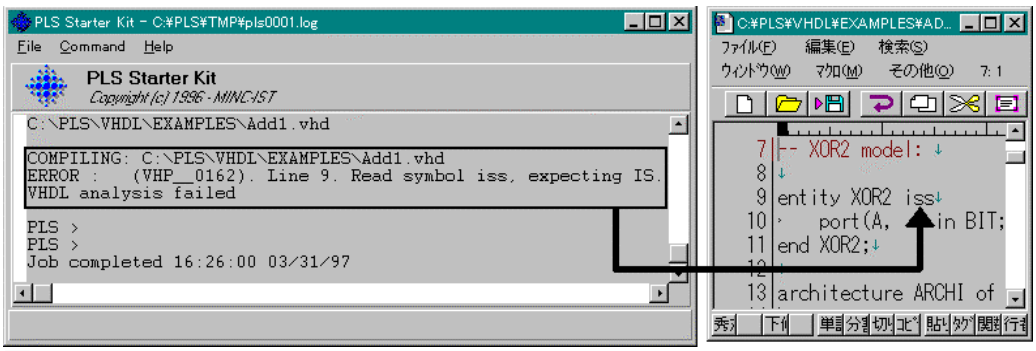


図4　Output　Format の設定誤りの例

これは Output Format の指定が誤っているために発生します．Target Technology に Altera_FLEX を指定したときは，Output Format として EDIF ではなく AHDL を選択します．



これは，VHDL ソースに文法エラーがあるために表示されるエラー・メッセージです(正しくは is であるが，VHDL ソースでは iss となっている)．

図5 文法エラーの例

1.1.2 このほかのエラー

・**Missing destination technology parameter**
ターゲット・デバイスが選択されていないと表示されます．使う FPGA の種類を選択してください．

・**Error - Illegal command usage : asyl**
スペースを含んだディレクトリやファイル名を使っていませんか？たとえば，"tset cir.vhd"．スペースは使わないでください．

・**Project file is either a VHDL file (.vhd[l]) or a project(.prj)**
Windows95/NT4.0 のエクスプローラを使う場合，拡張子の表示が省略される場合があるため，ファイルの種類を特定しにくい場合があります．ASYL には拡張子が vhd または prj(後述)のファイルだけを入力できるので，下記のような場合エラーとなります．

一見拡張子が vhd のファイルのように見えるが，ファイル名の先頭が"hadder.vhd"のテキスト・ファイル(メモ帳で見ることができる)なので，実際は"hadder.vhd.txt"というファイル名になっています(DOS プロンプト上では，"HADDER~1.TXT"と表示される)．

上記以外にも，パラメータの設定ミスなどによって対応するメッセージが表示されますが，エラー・メッセージは ASYL テキスト・ウィンドウの"PLS>"プロンプト付近に表示されるので，エラーが発生したらここを中心に検討するとよいでしょう．また ASYL を終了すると，セッション中のテキスト・ウィンドウの内容が plsxxxx.log のファイル名で tmp ディレクトリ(デフォルトでは\pls\tmp)に記録されるので，任意のテキスト・エディタでこれを開き，エラーや警告メッセージを確認することもできます．

処理が終了すると，エラーが発生して論理合成が最後まで行われなくても“ Job completed”と表示されます．
かならずテキスト・ウィンドウ内のメッセージをみて処理結果を確認しましょう．

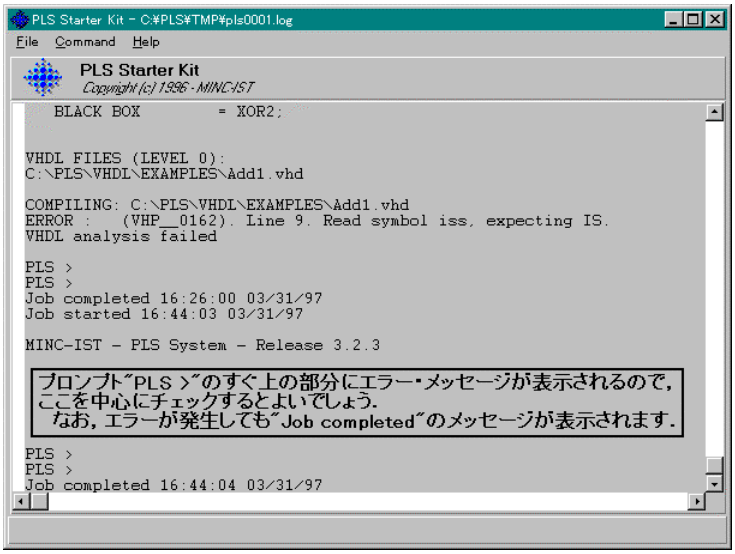


図6 エラー・メッセージの表示場所

1.2 簡単な回路の論理合成例

論理合成が行われるとネットリストが生成されます．使う FPGA ファミリとして Altera 社の FLEX を選択すると AHDL 形式のネットリストが生成されます．以下に，簡単な回路(半加算器)を ASYL で処理した結果を示します．

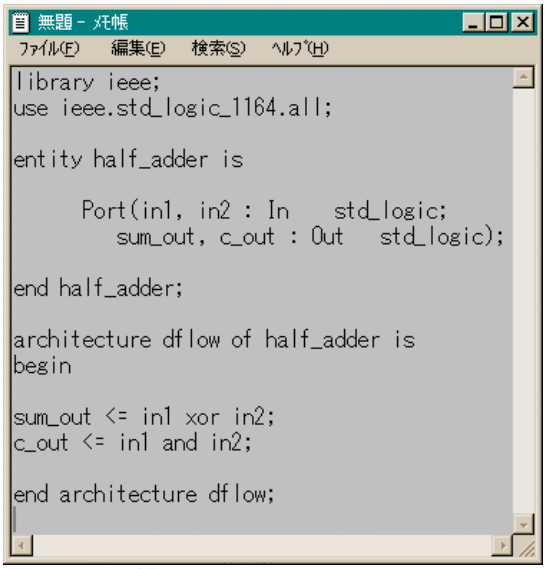


図7　半加算器の VHDL ソース

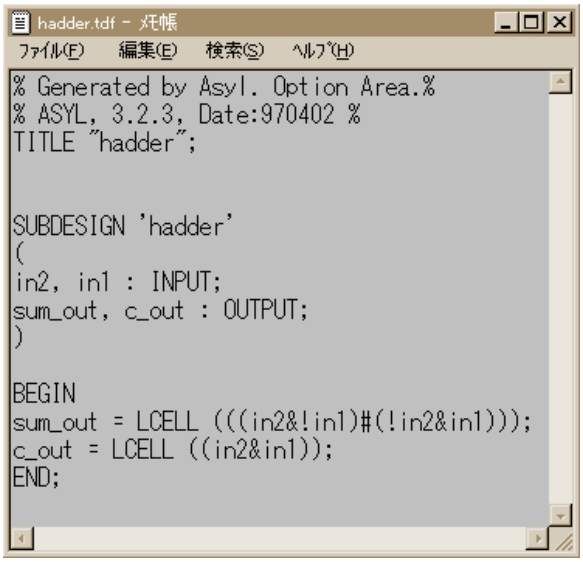


図8　図7 に示すソースの論理合成結果

図8 をみると，加算は入力 in1 と in2 の排他的論理和で，キャリは論理積で実現されていることがわかります．

1.3 ASYL で数値演算パッケージを使う場合

ASYL にはサンプル・ソースが多数付属しているので，これを参考に一部を書き換えてオリジナルの回路を作成できます．また，小社発刊の雑誌[Designwave Magazine]の付属 CD-ROM に VHDL シミュレータの Accolade PeakVHDL 規模制限版が収録されており，これをインストールすることによっても多数のサンプル・ソースが入手できます．

なお，ASYL で数値演算パッケージを含むソースの論理合成を行うときは，IEEE ライブラリではなく，ASYL 独自のライブラリを使います．Accolade PeakVHDL は IEEE の数値演算パッケージを使うので，もし VHDL ソースに IEEE の数値演算パッケージの記述がある場合は ASYL のライブラリに書き換えます．

Library ieee; Use ieee.std_logic_unsigned.all;	右のように書き換えます	Library asyl; Use asyl.arith.all;

もし，以下のようなエラー・メッセージが表示されたら VHDL ソースのライブラリ宣言部分を上記のように書き換えてください．

Library unit std_logic_unsigned is not available　in library ieee

1.4 MAX+plus II での操作

ASYL が生成した AHDL 形式のネットリストを MAX+plus II に入力して配置/配線を行いますが，**回路図入力，VHDL 入力など，デザインのエントリ方法が異なっても，MAX+plus II でのネットリスト以降の処理手順は共通**です．単に回路の機能を定義する方法が回路図入力方式から VHDL 入力に変わっただけです．

それでは，確認のために MAX+plus II での操作方法を以下に示します．

1.4.1 新規デザインの登録

MAX+plusⅡを起動したら[File]-[Project]-[Name...]コマンドを使って,ASYL が生成した AHDL 形式のネットリスト・ファイルを指定して，新規プロジェクトを登録します(図9)．

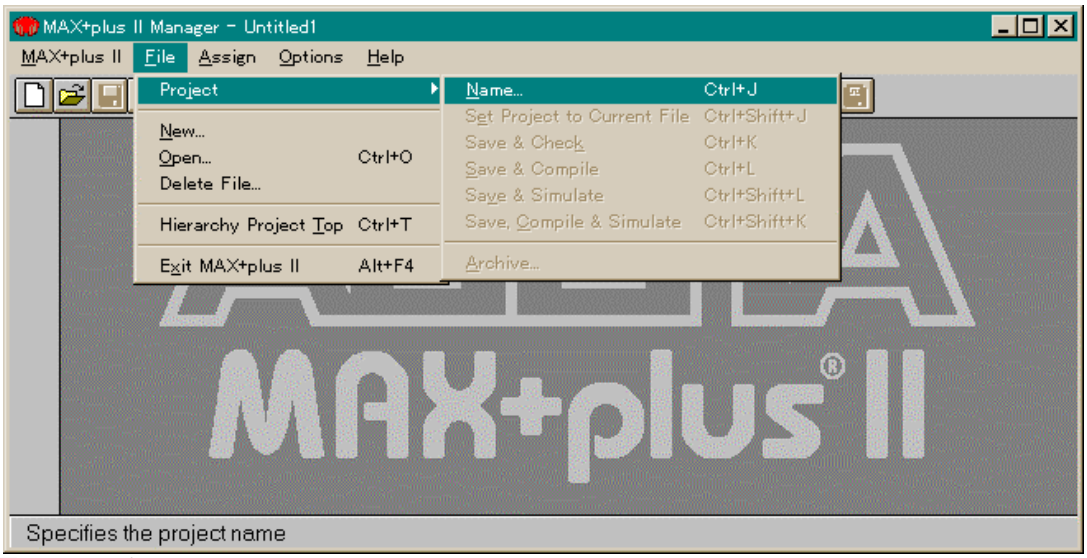


図9 新規プロジェクトの設定

1.4.2 配置/配線の実行

プロジェクトの登録が終わったらコンパイラを起動し，使うデバイスとして EPF8282 を指定します．

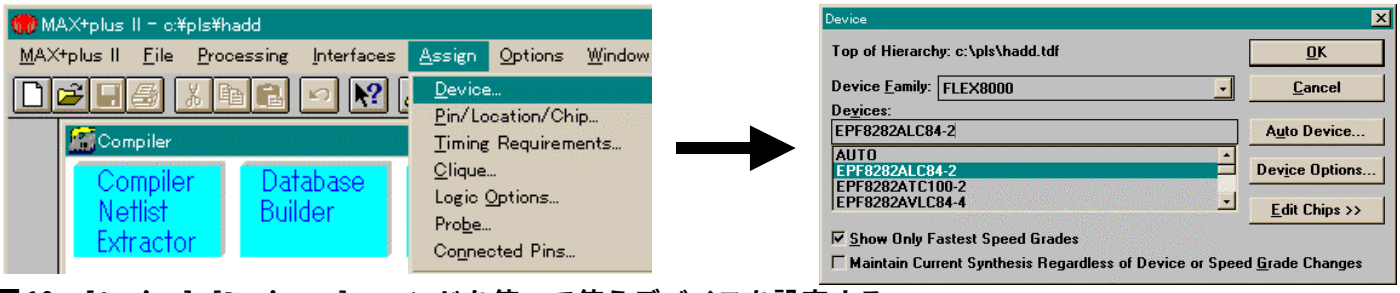


図10 [Assign]-[Device...]コマンドを使って使うデバイスを設定する

最後に配置/配線を行えばコンフィグレーション・データが生成されるので，ピン割り当てを確認したのちこれをダウンロードします．また，必要に応じて処理パラメータを変更してコンパイルを行うこともできます．

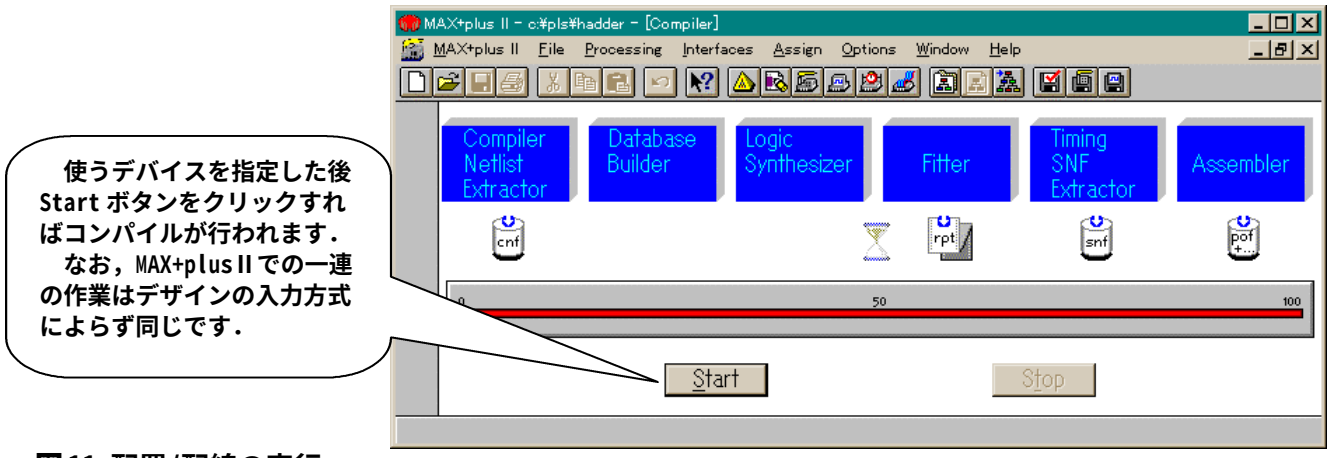


図11 配置/配線の実行

基本的な使い方2　複数の VHDL ソースを処理する

大規模な回路の設計を行う場合，機能別に分けて VHDL ソースを記述しておき，必要に応じてリンクすることが必要になります．ここでは，複数の VHDL ソース・ファイルからなるデザインを ASYL で処理する方法について解説します．

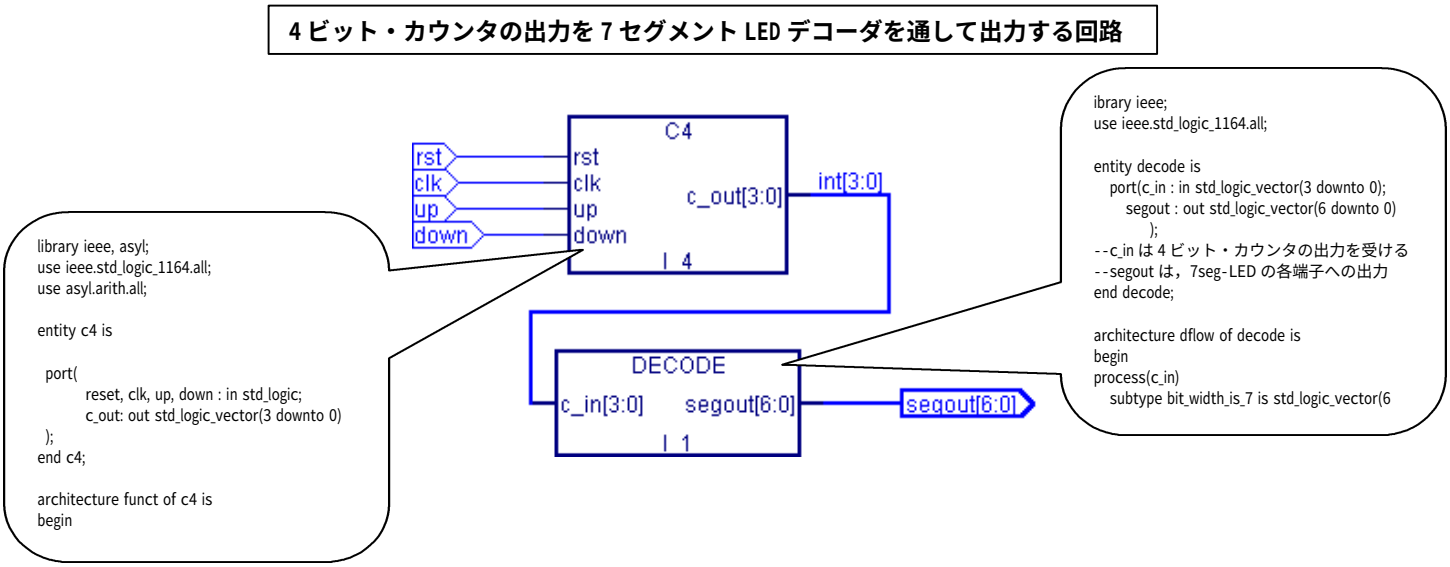


図 12　4 ビット・カウンタと 7 セグメント LED デコーダを分けて設計した例

2.1 半加算器を使って全加算器を作成する

ここでは，半加算器を使って全加算器を作成してみます．ブロック図は以下の通りです．

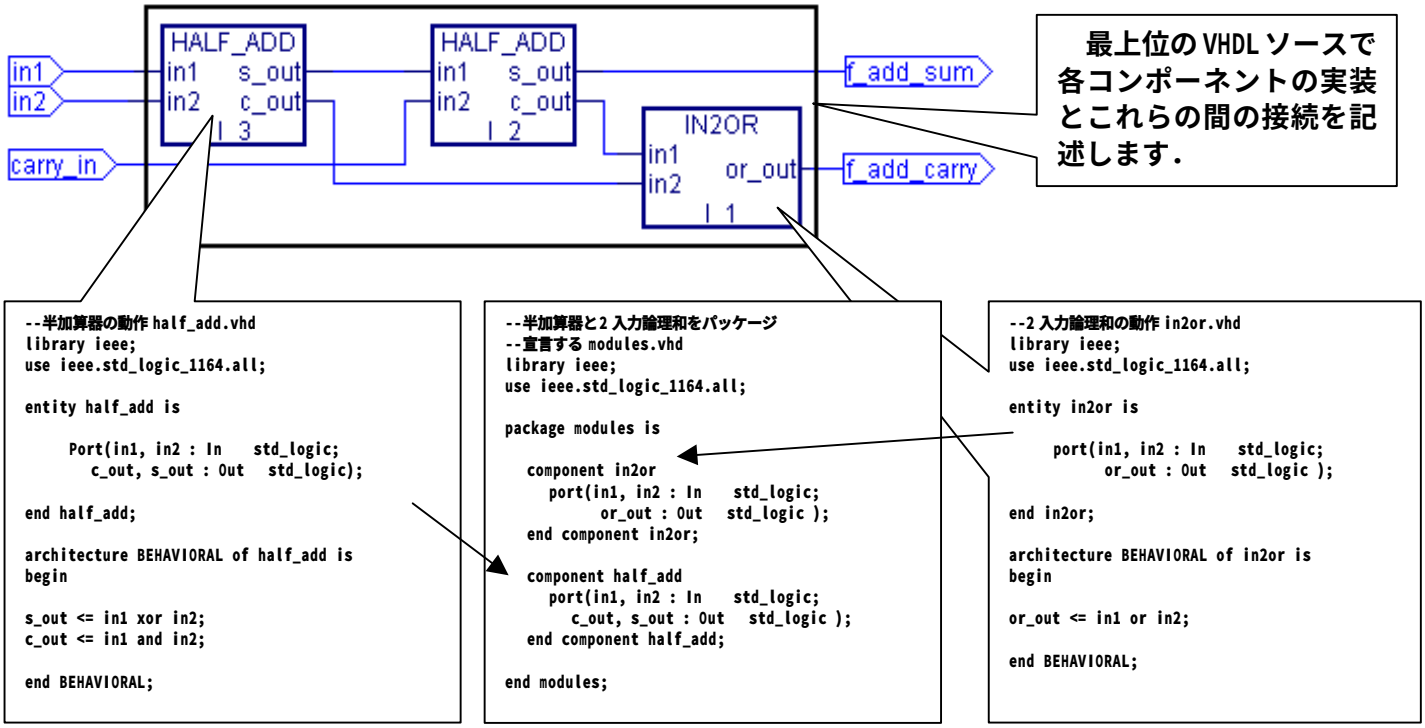


図 13　全加算器のブロック図と各コンポーネントの VHDL ソース

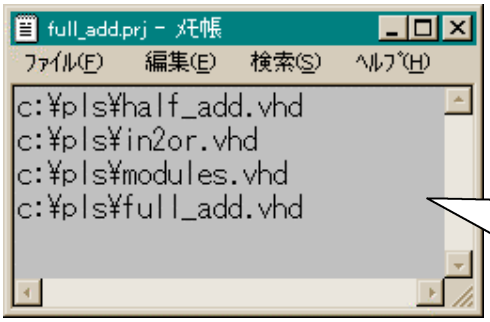
図 13 に示すように，全加算器を構成する各コンポーネント(半加算器と 2 入力の論理和)を作成し，これらをパッケージ化します．最後に，全加算器を実現するために各コンポーネントの実装とこれらの間の接続を記述します．

なお，最上位の VHDL ソースは以下の通りです．

```
-- 全加算器の実現 full_add.vhd
  library ieee, work;
  use ieee.std_logic_1164.all;
  -- パッケージ化したコンポーネント群を使う
  use work.modules.all;

  entity full_add is

        port(in1, in2, carry_in : In      std_logic;
         f_add_sum, f_add_carry : Out     std_logic);

  end full_add;

  architecture struct of full_add is

  -- N_3, N_4, N_5 は内部信号(ネット名)
  signal        N_3, N_4, N_5 : std_logic;

  begin

  -- 半加算器と二入力論理和の実装(instance)とこれらの間の接続

     instance_of_in2or : in2or
        Port Map ( in1=>N_3, in2=>N_4, or_out=>f_add_carry );
     instance_of_half_add1 : half_add
        Port Map ( in1=>N_5, in2=>carry_in, c_out=>N_3, s_out=>f_add_sum );
     instance_of_half_add2 : half_add
        Port Map ( in1=>in1, in2=>in2, c_out=>N_4, s_out=>N_5 );

end struct;
```

ASYL は各 VHDL ソースのコンパイル結果をデフォルトでは work ライブラリに格納します．
また，Modules.vhd で二つのコンポーネント(半加算器と二入力の論理和)を modules のパッケージ名でパッケージ化しているので，これらすべてを利用するために work.modules.all と宣言します．

それでは，全加算器を論理合成してみます．ASYL では個々のコンポーネントを個別にコンパイルした後最上位の VHDL ソースでこれらを使うこともできますが，コンポーネントの数だけ ASYL を操作しなければならないので，コンポーネント数が多い場合は操作が大変です．そこで，**使う VHDL ソース・ファイルの一覧を記したプロジェクト・ファイル**を作成してこれを ASYL に入力することにより，これらを一度に処理させます．
プロジェクト・ファイルは以下のように記述します．



設計で使う VHDL ソース・ファイル名を絶対パスを含めて記述します．
記述の順序は任意ですが，下位層→上位層というように，記述の順番に規則を作っておくと管理が容易になるでしょう．
プロジェクト・ファイルは任意のテキスト・エディタを使って作成できます．
ファイル名は任意ですが，拡張子は prj とします．

図 14 プロジェクト・ファイルの作成方法

ASYL での処理は以下の通りです．

VHDL ファイルの代わりにプロジェクト・ファイル名を指定します．
このほかの項目の設定は**一つの VHDL ソースを処理する**場合と同じです．

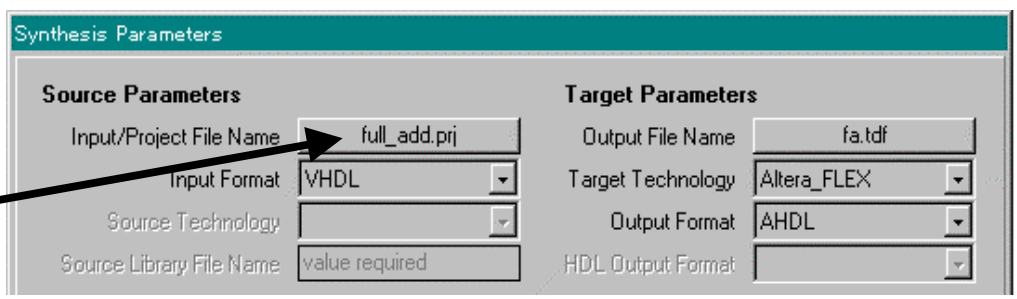


図 15 プロジェクト・ファイルの入力

最後に MAX+plus II で配置/配線を行います．MAX+plus II での処理も単一の VHDL ソースを処理する場合と同じ操作で行えます(1.4.1 を参照)．

2.2 論理合成パラメータの保存

VHDL ソースの文法エラーなどにより，一度で論理合成が完了しない場合があります．このようなとき ASYL を起動するごとに同じパラメータを入力するのはわずらわしいものです.しかし ASYL には論理合成パラメータを保存するオプションがあるので,これを利用すれば ASYL を起動するごとに保存された論理合成パラメータが読み込まれるので，何度も同じ設定を行わずにすみます．

なお，保存できるパラメータは一組のみで，新しいパラメータの保存を行うと前回のそれは消去されます．

また，保存にあたり保存ファイル名は必要なく，Save Configuration の実行だけでパラメータが保存されます．

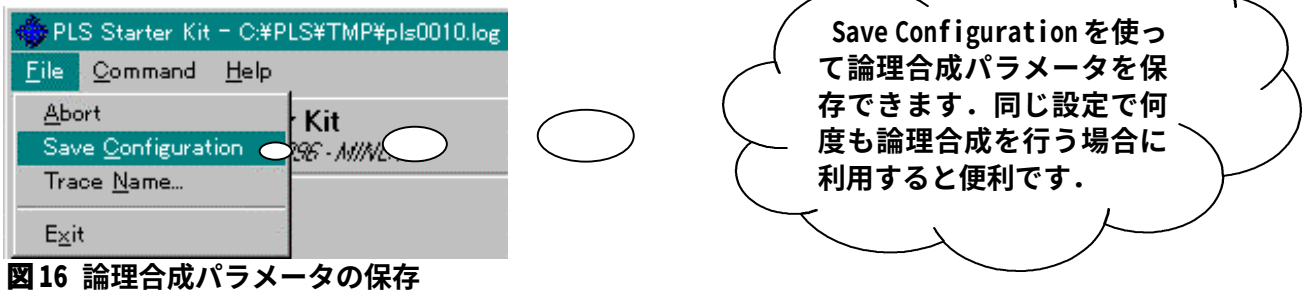


図 16 論理合成パラメータの保存

応用例　コンポーネント化設計

ここでは複数の VHDL ソースよりなる設計の別の処理方法について説明します．”基本的な使い方 2”で示した処理方法では，VHDL ソースは機能別に分かれているが，AHDL ネットリストはそれらの機能を結合したものとなっています．これでは，ASYL の機能制限である 1000 ゲート相当，64I/O を超えてしまう場合があります．そこで，ネットリストもコンポーネント化し，これらを MAX+plus II 上で結合する方法について説明します．

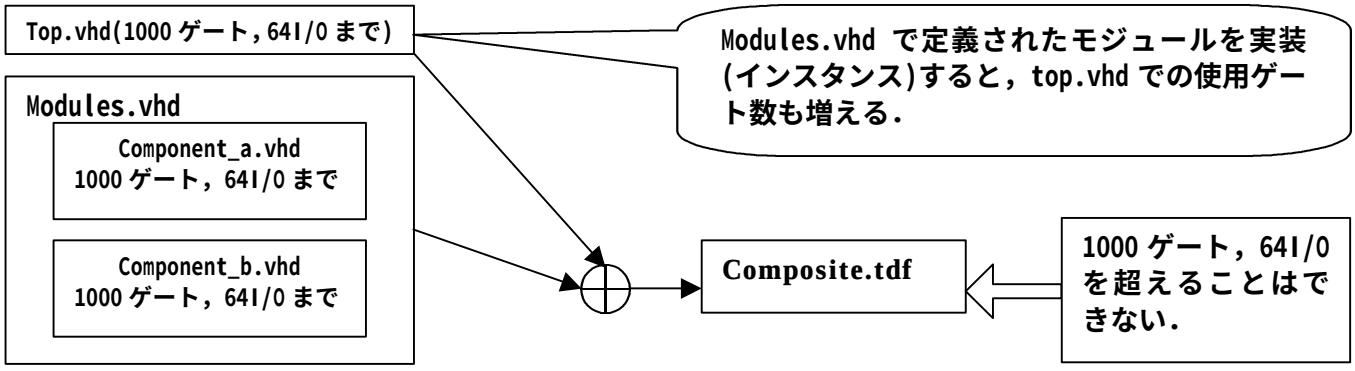


図 17 ASYL の制限に達しやすい設計

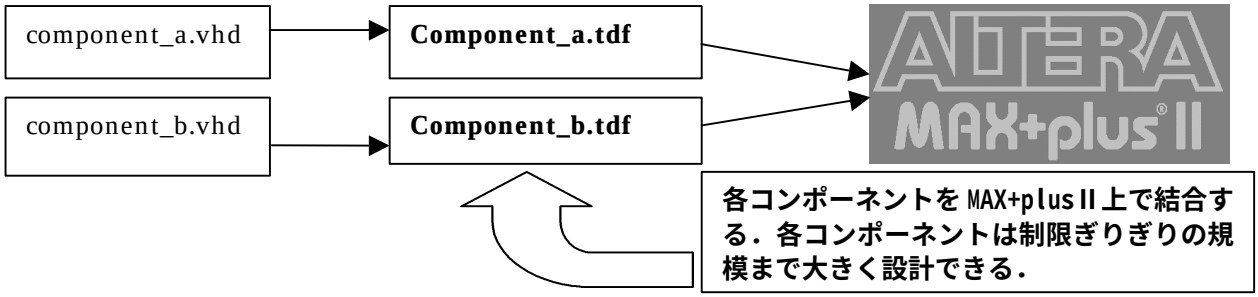


図 18 ネットリストをコンポーネント別に出力し，MAX+plus II 上で結合する設計

3.1 各モジュールを論理合成する

それでは，具体的な操作方法を示します．まず，各モジュールを論理合成します．1.1 節を参考に各モジュールをあらわす VHDL ソースを論理合成し，AHDL ネットリストを出力します．図 12 に示す回路を作成する場合はまず，C4 をあらわすモジュールと DECODE をあらわすモジュールをそれぞれ論理合成し，AHDL ネットリストを出力させます(ソース・ファイル名はそれぞれ c4.vhd と c_decode.vhd)．ネットリスト・ファイル名は任意ですが，ここではそれぞれ c4.tdf および c_decode.tdf とします．

3.2 MAX+plus II での操作

ASYL を使って生成したネットリストから各ネットリスト(機能)に対応するシンボルを作成し，これをグラフィック・エディタ(gdf)上で使えるようにします．最後に FPGA への配置/配線を行います．

3.2.1 シンボルの作成

1. 生成したネットリストを適当なディレクトリにまとめる

処理されるネットリストは HDD 内の任意の位置に置けますが,特定のディレクトリ内にまとめておくと管理しやすいでしょう．ここでは，\maxplus2\max2lib 内の mylib ディレクトリにまとめておくことにします．

2. シンボルを作成する

MAX+plus II の[File]-[Open...]メニュから c4.tdf を選択すると，ネットリストが画面に表示されます．

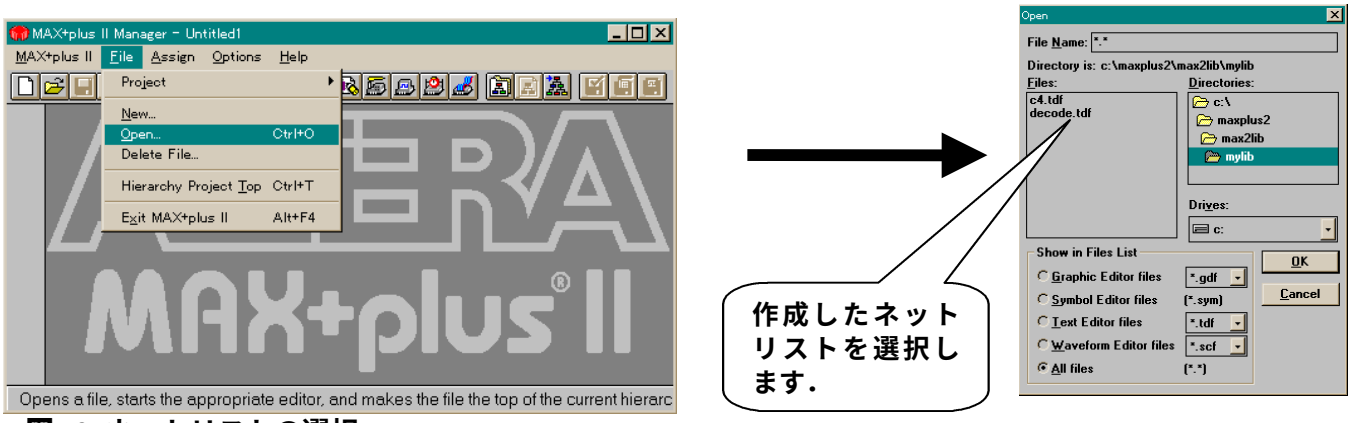


図 19 ネットリストの選択

次に，[File]-[Create Default Symbol]を選択します．"OK to change project to \maxplus2\max2lib\mylib\c4.tdf"と表示されたウィンドウが開くので OK ボタンをクリックします．MAX+plus II は選択されたネットリストのチェックを行ったのちシンボルを作成します．ここで，シンボルの作成を告げるメッセージ・ウィンドウが開くので，エラー，警告がともにないことを確認して OK ボタンをクリックします．

最後に[File]-[Open…]を使ってモジュール C4 のシンボル・ファイルである c4.sym を開くと，図 20 に示すシンボルが作成されていることがわかります．

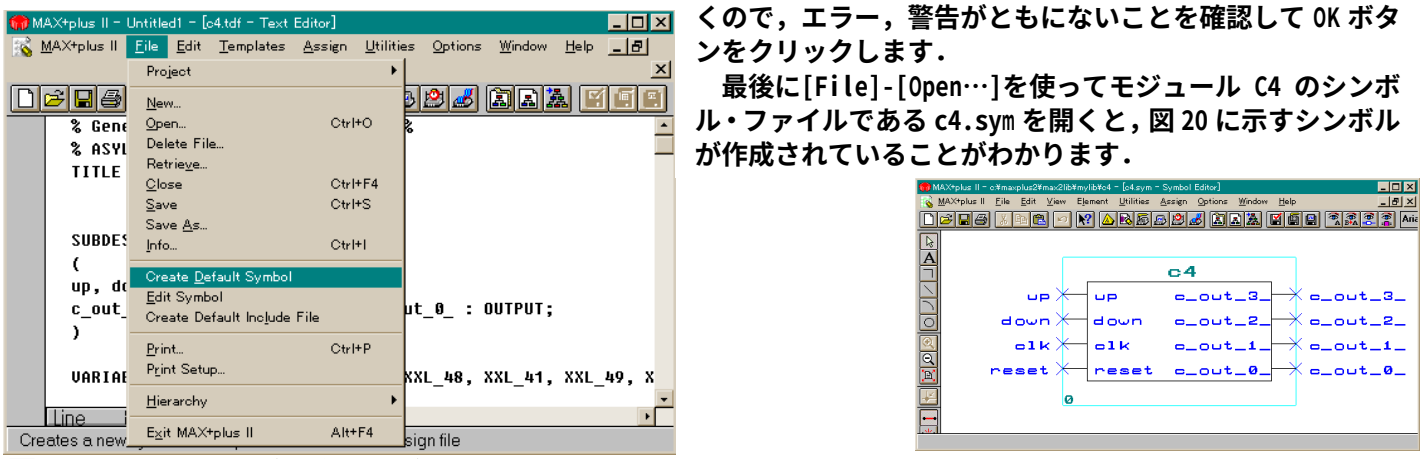


図 20 シンボルの作成と自動作成されたシンボル

なお，モジュール C4 の VHDL ソース(c4.vhd)では出力をベクタ・タイプで定義しましたが，作成されたシンボルの出力はベクタを構成する各信号線として定義されています．これは，ASYL がベクタを，これを構成する各信号線に分解したためです.出力をバス形式であらわしたいときはネットリストを以下のように書き換えてください(太い斜字をそれぞれ囲み文字で示すように書き換える)．

```
SUBDESIGN 'c4'
(
up, down, clk, reset : INPUT;
c_out_3_, c_out_2_, c_out_1_, c_out_0_ : OUTPUT;
)

VARIABLE XXL_50, XXL_46, XXL_42, XXL_48, XXL_41, XXL_49, XXL_47, XXL_45,
XIL_50, XIL_51, XIL_52, XIL_53 : NODE;

BEGIN
c_out_3_ = (XXL_42);
c_out_2_ = (XXL_46);
c_out_1_ = (XXL_48);
c_out_0_ = (XXL_50);
XIL_50 = LCELL
(((XXL_50&((XXL_46&((XXL_42&!XXL_48)#(!XXL_42&XXL_48)))#(!XXL_46&
XXL_42)))#(!XXL_50&XXL_42)));
```

→

```
SUBDESIGN 'c4'
(
up, down, clk, reset : INPUT;
c_out[3..0] : OUTPUT;
)

VARIABLE XXL_50, XXL_46, XXL_42, XXL_48, XXL_41, XXL_49, XXL_47, XXL_45,
XIL_50, XIL_51, XIL_52, XIL_53 : NODE;

BEGIN
c_out3 = (XXL_42);
c_out2 = (XXL_46);
c_out1 = (XXL_48);
c_out0 = (XXL_50);
XIL_50 = LCELL
(((XXL_50&((XXL_46&((XXL_42&!XXL_48)#(!XXL_42&XXL_48)))#(!XXL_46&
```

図 21 ベクタ形式のネットリストへの書き換え

書き換えた AHDL ネットリストからシンボルを作成すると図 22 に示すようになります．なお，AHDL の文法の詳細については MAX+plus II バージョン 6.2 日本語ヘルプなどを参照してください(図 23)．

以上の作業を C_decode.tdf についても行います．必要に応じてシンボルの形状や，シンボル・ピンの編集などを行えば作業は完了です．

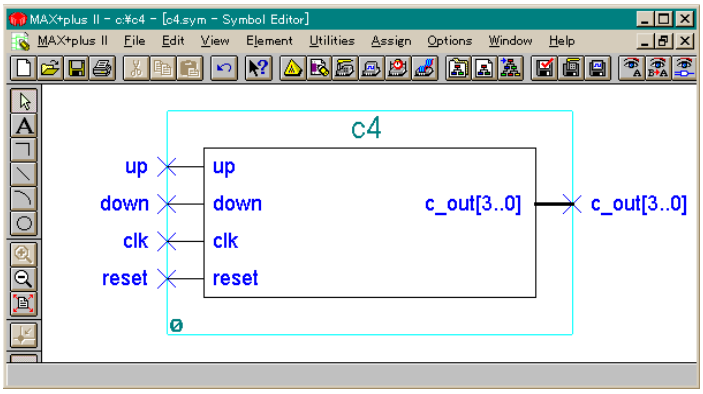


図 22 ベクタ形式を使ったシンボル

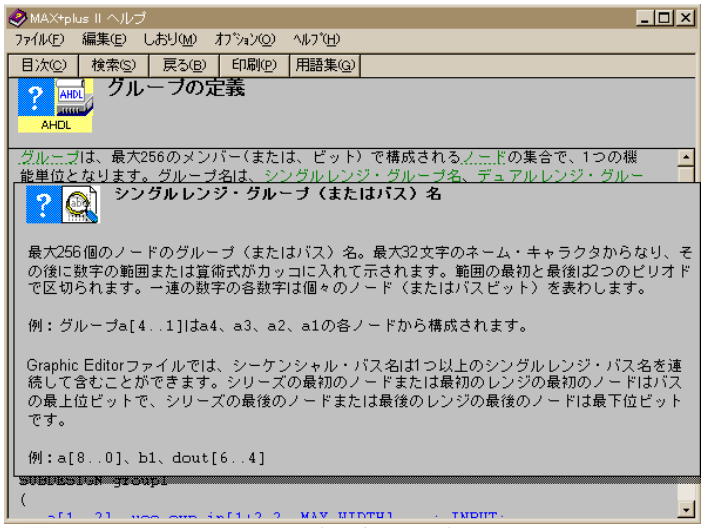


図 23 MAX+plus II ヘルプ(グループの定義について)

3．作成したシンボルの位置を登録する

新規作成したシンボルを使ったプロジェクトのコンパイルを行うために，それらの置かれているディレクトリ名を登録します．登録は図 24 に示すように，maxplus2.ini ファイルに新規作成したシンボルの置かれているディレクトリ名を記入します．

Maxplus2.ini 内の MAXPLUS_LIB の項に新規作成したシンボルが置かれているディレクトリ名をセミコロンで区切って記入します．

なお，maxplus2.ini は MAX+plus II プログラムをインストールしたディレクトリ(デフォルトでは\maxplus2)にあります．

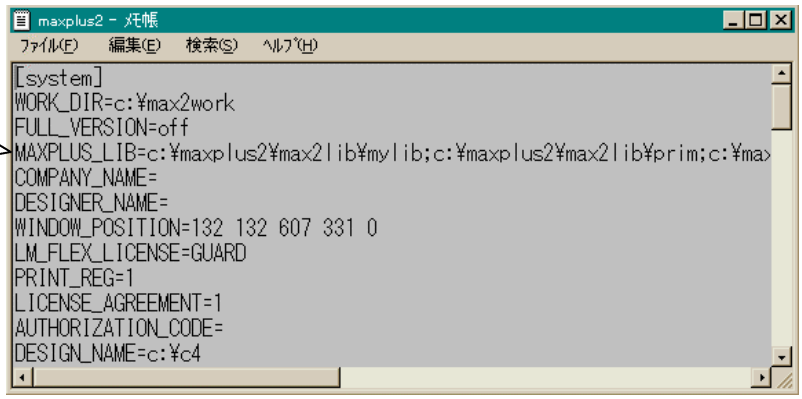


図 24 シンボル位置の登録

4．作成したシンボルを使ってブロック図を作成する

作成したシンボルを使ってブロック図を描きます.具体的には,新規プロジェクトを登録したのちグラフィック・エディタを使ってシンボルを配置してこれらの間を配線します．最後に I/O パッドつければブロック図は完成です．

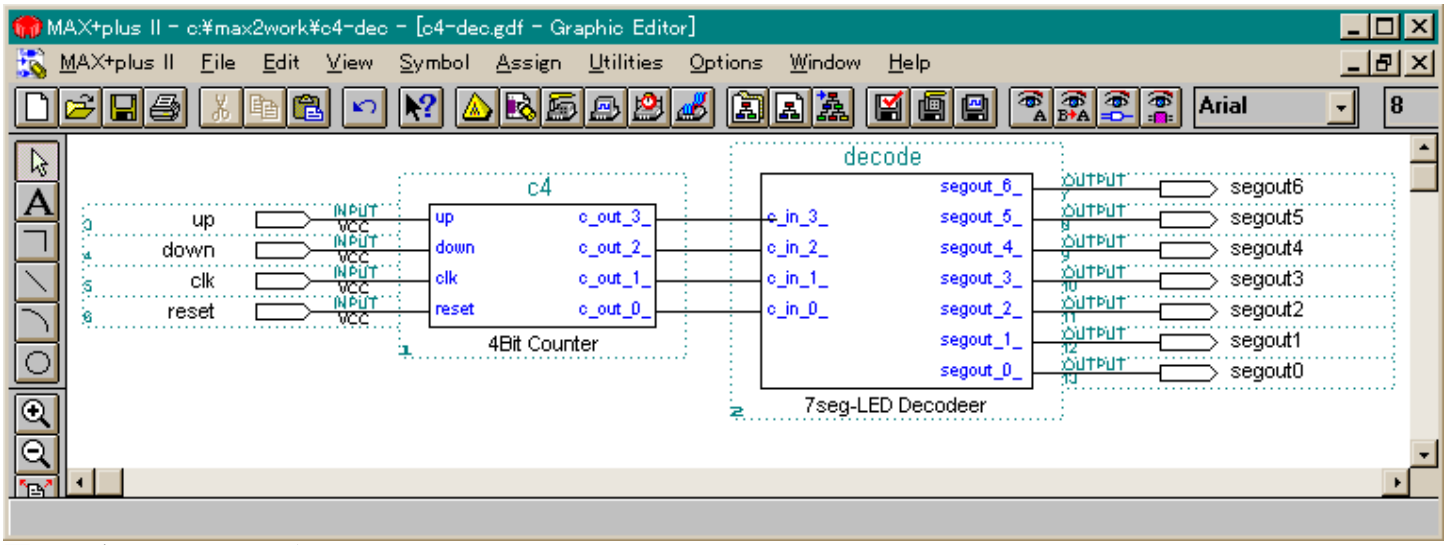


図25 ブロック図の作成例

3.2.2 EPF8282 への配置/配線

図 25 に示すようなブロック図を作成したら MAX+plus II コンパイラを起動します．ターゲット・デバイスとして FLEX デザイン・キット/CQ 版付属基板に実装されている EPF8282 を選択して配置/配線を行います．コンフィグレーション・データが生成されたらこれをダウンロードして FPGA を動作させることができます．

ソース・ファイルの使い方

ここでは，本チュートリアルで紹介した各 VHDL ソース・ファイルや関連ファイルの使い方について説明します．

1．ディレクトリ構成

SOURCE.EXE をダウンロードして C ドライブのルートで実行すると五つのディレクトリが現れます．各ディレクトリの内容は以下の通りです．

```
Asyl_tut
 |
 ├12：1.2 節で紹介した半加算器の VHDL ソース・ファイルと AHDL ネットリスト
 |    └Accsim：半加算器のシミュレーション・プロジェクト
 ├21：2.1 節で紹介した全加算器の VHDL ソースとプロジェクト・ファイルおよび AHDL ネットリスト
 |    └Accsim：全加算器のシミュレーション・プロジェクト
 ├31：3.1 節で紹介した 7seg-LED デコーダ付き 4 ビット・カウンタの VHDL ソース・ファイル
 |    └Accsim：7seg-LED デコーダ付き 4 ビット・カウンタのシミュレーション・プロジェクト
 ├Maxplus：MAX+plus II で使うファイル群
 |    ├C4-dec：7seg-LED デコーダ付き 4 ビット・カウンタのブロック図(gdf と acf ファイル)
 |    ├Mylib：プロジェクト c4-dec および rotate で使うライブラリ群
 |    └Rotate：FLEX デザイン・キット/CQ 版に付属のサンプル・デザイン"Rotate"のブロック図(gdf と acf ファイル)
 └Rotate：FLEX デザイン・キット/CQ 版に付属のサンプル・デザイン"Rotate"の VHDL ソース・ファイル．
      └Accsim：デザイン"Rotate"のシミュレーション・プロジェクト
```

図26　サンプル・ファイルの内容

2. 使い方

2.1 1.2 節の関連ファイル

2.1.1 半加算器のVHDL ソース・ファイル

本文 1.2 節を参考に C:\Asyl_tut\12 ディレクトリ内の half_add.vhd の論理合成を行います．参考に論理合成した結果も収録しています(hadder.tdf)．

2.1.2 半加算器のシミュレーション・プロジェクト

半加算器の Accolade PeakVHDL によるシミュレーション・プロジェクトです．Accolade PeakVHDL を起動し，C:\Asyl_tut\12\Accsim ディレクトリ内の half_add.acc を指定してください．テスト・ベンチが付属しているのですぐにシミュレーションが行えます．

2.2 2.1 節の関連ファイル

2.2.1 全加算器のVHDL ソースとプロジェクト・ファイル

本文 2.1 節を参考に C:\Asyl_tut\21 ディレクトリ内の full_add.prj の論理合成を行います．参考に論理合成した結果も収録しています(full_add.tdf)．

2.2.2 全加算器のシミュレーション・プロジェクト

全加算器の Accolade PeakVHDL によるシミュレーション・プロジェクトです．Accolade PeakVHDL を起動し，C:\Asyl_tut\21\Accsim ディレクトリ内の full_add.acc を指定してください．テスト・ベンチが付属しているのですぐにシミュレーションが行えます．

2.3 3.1 節の関連ファイル

2.3.1 7seg-LED デコーダ付き4 ビット・カウンタのVHDL ソース・ファイル群

本文 3.1 節を参考に C:\Asyl_tut\31 ディレクトリ内の c4.vhd と c_decode.vhd の論理合成を行います.生成されたネットリストの取り扱いについては本文 3.2 節を参照してください．

2.3.2 7seg-LED デコーダ付き4 ビット・カウンタのシミュレーション・プロジェクト

7seg-LED デコーダ付き 4 ビット・カウンタの Accolade PeakVHDL によるシミュレーション・プロジェクトです．Accolade PeakVHDL を起動し，C:\Asyl_tut\31\Accsim ディレクトリ内の c4_dec.acc を指定してください．テスト・ベンチが付属しているのですぐにシミュレーションが行えます．

2.4 MAX+plus II で使うファイル

2.4.1 7seg-LED デコーダ付き4 ビット・カウンタのブロック図

本文 3.1 節で生成した c4.tdf と c_decode.tdf を基に作成したブロック図です．MAX+plus II の[File]-[Project]-[Name...]コマンドを使って C:\Asyl_tut\Maxplus\C4-dec ディレクトリ内の c4-dec.gdf を指定した後コンパイルを行えば c4-dec.ttf が生成されるので，FLEX デザイン・キット/CQ 版に付属の実験基板に実装された EPF8282 にダウンロードすれば実際に 16 進カウンタとして動作します．

なお，7seg-LED デコーダ(c_decode)の出力は実験基板上の 7seg-LED の各端子に接続された EPF8282 の各ピンに，4 ビット・カウンタの up,down 入力は EPF8282 の 22,24 ピンに割り当てずみですが(詳細については c4-dec.gdf を参照のこと)，clk,reset 入力は未定義です．コンパイルを行った後これらに割り当てられたピン番号を確認してクロックおよびリセット信号を与えてください．入力されたクロック信号はそのままカウンタの駆動に使われます．
また，up=1，down=0 でアップ・カウンタ，up=0,down=1 でダウン・カウンタとなり，reset=1 でカウンタはリセットされます．

以上の作業は2.4.2 節で示す作業を完了してから行ってください．

注：
コンパイルを行うと，マニュアルでのピン割り当てを警告するメッセージが表示されますが，無視してください．
ピン割り当てを行うには，使うデバイスが選択されている必要があります．
Maxplus2.ini に本文 3.2.1 節の 3 で示す編集作業を行っておかないと，コンパイル時にシンボルが見つからないというエラー・メッセージが表示されます．

2.4.2 プロジェクト c4-dec と rotate で使うライブラリ群

本文 3.1 節で紹介した 7seg-LED デコーダ付き 4 ビット・カウンタと FLEX デザイン・キット/CQ 版に付属のサンプル・デザイン"rotate"で使うライブラリ群で，本文 3.2 節で示す作業を行って生成されるものです．
これらをすぐに使う場合は，C:\Asyl_tut\Maxplus\Mylib 内の全ファイルを，\Maxplus2\Max2lib 内に Mylib というディレクトリを作成してこの中にコピーしてください(MAX+plus II を Maxplus2 以外のディレクトリにインストールしている方は上記の Maxplus2 をインストールしたディレクトリに読み替えてください)．この後，本文 3.2.1 節の 3 を参考に maxplus2.ini を編集してください．

2.4.1 節および2.4.3 節で示す作業は本節で示す作業を完了してから行ってください．

2.4.3 サンプル・デザイン"rotate"のブロック図

FLEX デザイン・キット/CQ 版に付属のサンプル・デザイン"rotate"のブロック図です．MAX+plus II の[File]-[Project]-[Name...]コマンドを使って C:\Asyl_tut\Maxplus\Rotate ディレクトリ内の rot.gdf を指定した後コンパイルを行えば rot.ttf が生成されるので，FLEX デザイン・キット/CQ 版に付属の実験基板に実装された EPF8282 にダウンロードすれば実際に動作します(8 の字を描くように 7seg-LED の各エレメントが点灯する)．
なお，7seg-LED デコーダ(r_decode)の出力は実験基板上の 7seg-LED の各端子に接続された EPF8282 の各ピンに，8 ビット・シフタの dir 入力は EPF8282 の 22 ピンに割り当てずみですが(詳細については rot.gdf を参照のこと)，clock,reset 入力は未定義です．コンパイルを行った後これらに割り当てられたピン番号を確認してクロックおよびリセット信号を与えてください．入力されたクロック信号はそのままシフタの駆動に使われます．また，dir の値によって 8 の字を描く方向を変えることができ，reset=1 でシフタはリセットされます．

以上の作業は2.4.2 節で示す作業を完了してから行ってください．

注：
コンパイルを行うと，マニュアルでのピン割り当てを警告するメッセージが表示されますが，無視してください．
ピン割り当てを行うには，使うデバイスが選択されている必要があります．
Maxplus2.ini に本文 3.2.1 節の 3 で示す編集作業を行っておかないと，コンパイル時にシンボルが見つからないというエラー・メッセージが表示されます．

2.5　FLEX デザイン・キット/CQ 版に付属のサンプル・デザイン"Rotate"の関連ファイル.

2.5.1 VHDL ソース・ファイル

FLEX デザイン・キット/CQ 版に付属のサンプル・デザイン"Rotate"の VHDL ソース・ファイルです．2.4.2 節および 2.4.3 節にしたがって作業を行えばすぐに実際の動作を見ることができます．自分で論理合成からの構築作業を行う場合は，C:\Asyl_tut\Rotate 内の rotate.vhd と r_decode.vhd の二つの VHDL ソース・ファイルを本文 3.1 節以降を参照して処理してください．

2.5.2 "Rotate"の VHDL シミュレーション・プロジェクト

"Rotate"の Accolade PeakVHDL によるシミュレーション・プロジェクトです．Accolade PeakVHDL を起動し，C:\Asyl_tut\Rotate\Accsim ディレクトリ内の rot.acc を指定してください．テスト・ベンチが付属しているのですぐにシミュレーションが行えます．

本文についてのご質問などは以下の電子メール・アドレスまでお願いいたします．
メール送付先：***edasupport@cqpub.co.jp***