VHDL- 8

レオナルドでのライブラリー

この文書の目的

VHDL シミュレータ Vsys、論理合成ツール レオナルド は、一対のツールとして使われる場合が多いのではないかと思います。　しかし、この２つのツール間においてすら、そのライブラリの使われ方には大きな相違があります。

断片的ですが、基本的な事項でありながら、ユーザーズ・マニュアルなどに明確な定義・記述がみつかりませんので、ＶＨＤＬ 環境の説明の為に、私なりのアプローチをした結果を備忘録として文書に作成、保存しておきたいと思います。

レオナルドでのライブラリ

とは、基本的には論理合成を目的にしたもので、３つのカテゴリーがあると考えたほうが理解し易いように思います。

１ テクノロジ・ライブリー

これは、ターゲットデバイスの内容についてのライブラリで、各ＦＰＧＡ の構成要素に対応したものです。

オナルドの ＧＵＩ では ＩＯ メニューの LOAD LIBRARY によって選択、指定されるデバイス・ファミリ依存のライブラリー（.syn）です。

２ ＶＨＤＬ 言語基本仕様ライブラリー

ＶＨＤＬ 言語仕様では、TEXTIO と STD とが大前提として存在する事になっています。

レオナルドの ＧＵＩ では ＩＯ メニューの READ によって指定される VHDL ソースファイル（.vhd）です。

レオナルドでは、この２つの基本ライブラリに IEEE と EXEMPLAR ライブラリを加えた合計４つの基本ライブラリを、VHDL ソースファイル形式のままで、下に示した特定のディレクトリに保管・管理しています。

x: \ exemplar \ leonardo \ data

ユーザーが作成するソース文の中で、ライブラリ文節で指定される IEEE などの基本ライブラリは、このディレクトリに置かれた VHDL

ソースをその都度読み込んで、**ワーク**と呼ばれる揮発性のデータベース上にコンパイルして作成されます。
IEEE という名の、同じライブラリの構成要素であっても、それを構成するパッケージが全て一度に読込まれるのではなく、ユーザーのソース記述で use 文節が特定のパッケージを指定した時に、はじめてその部分だけが**ワーク**に読込まれる分割ソースファイル形式になっている、事に注意します。

例外的に、現在作業している（カレント）ディレクトリ内にこれらの基本ライブラリのソースファイルが予めコピーされていたような場合には、レオナルドはカレントに置かれているソースファイルを優先してコンパイルします。
ライブラリ名とは、レオナルドの場合、ディレクトリでもなく、ファイル名でもない、データベース領域の一部に付けられた「名前」の事であると解釈したほうが理解しやすいように思われます。

3 ワーク・ライブラリー

これも含めて、ライブラリー、と呼ぶのは、反って混乱を招くおそれが有るようにも思うのですが、論理合成作業ではすべての要素がこの**ワーク**内にコンパイルされてはじめて結果を出力できるようになります。
ライブラリと呼びにくいのは、この作業の結果なり、内容なりが、明確にある特定のディレクトリに保存され再利用可能となるわけではないからです。

比較的設計規模の小さな FPGA をターゲットにした時、レオナルドの ＧＵＩ 環境での論理合成作業では、先ほど指摘しておいた４つの基本ライブラリの内、STD、IEEE、EXMPLAR の３つ以外のものについては、全ての use 文節を work と言う名前に統一してで記述したほうが混乱や錯誤を避けられるのではないかとさへ思います。
（元々 TEXTIO は合成の対象になりませんので除外しています）
私なりのイメージ図を添付しておきますので参照してみてください。

Vsys の環境では、**ワーク**とは、ある特定のディレクトリに関連づけられていましたが、レオナルドでは違うということを見る為に、以下のように実例を作ってその結果をみてみましょう。

今回の例題

本例では、本文を FOO ディレクトリに置き、そこで使うパッケージ constpac は SUB ディレクトリに置いてあります。

本文の内容は以下のとおりです

```
------------------------------------------------------------
-- test of ieee library loading.
------------------------------------------------------------
LIBRARY ieee;
use ieee.std_logic_1164.all;

entity chk is
  port (data: in std_logic_vector(7 downto 0);
        clk : in std_logic;
        r   : out bit_vector(3 downto 0);
        q   : out std_logic_vector(7 downto 0));
end chk;

LIBRARY MYLIB;                    ①
use MYLIB.constpack.all;          ②

Architecture rtl of chk is
begin
    x0: process(clk,data)
    begin
    if(clk'event and clk='1') then
        q <= data;
        r <= Cnst_0010;
        end if;
    end process x0;
end rtl;
```

この枠の中を変更してみます

パッケージの内容は以下のとおりです

```
-- PACKAGE.
package constpack is
    constant Cnst_0010 : Bit_Vector( 3 downto 0) := B"0010";
end constpack;
```

ライブラリ文節を削除して

最初の例では、本文中の枠で示した部分を書き換えて

LIBRARY 文節は削除し、

パッケージは、以下の文で参照するようにします。

use work.constpack.all;

これをレオナルドに読込ませる時、ダイアログボックスからワーク・ライブラリを指示してコンパイルさせると以下の経過をたどります。

応答メッセージで、std、ieee ライブラリがデフォルトの読み出しになっている点にも注意しておいてください。

```
LEONARDO{1}: read -work WORK -format VHDL D:/FOO/SUB/libtest.vhd
  -- Reading file C:\Exemplar\Leonardo\data\standard.vhd for
           unit standard
  -- Loading package standard into library std
  -- Reading vhdl file D:/FOO/SUB/libtest.vhd into library WORK
  -- Loading package constpack into library WORK
  Info: "D:/FOO/SUB/libtest.vhd" Nothing to synthesize
LEONARDO{2}: read -work WORK -format VHDL D:/FOO/chk.vhd
  -- Reading vhdl file D:/FOO/chk.vhd into library WORK
  -- Reading file C:\Exemplar\Leonardo\data\std_1164.vhd for
           unit std_logic_1164
  -- Loading package std_logic_1164 into library ieee
  -- Loading entity chk into library WORK
  -- Loading architecture rtl of chk into library WORK
  -- Compiling root entity chk(rtl)
  LEONARDO{3}:
```

上の例では、２つのソースを共に同一のワーク・ライブラリ（というよりも、ワーク・データベース）WORK に読込んでコンパイルしています。

ライブラリ文節を使う

次の例では、このプログラム本文ソースの LIBRARY 文節を書き換えて MYLIB を指定し、その中のパッケージ constpack を use 文節で呼び出すように変更します。

```
library MYLIB;                ①
use MYLIB.constpack.all;      ②
```

レオナルドでのソース読込みは以下のような過程をたどります。

```
LEONARDO{1}:read -work MYLIB -format VHDL D:/FOO/SUB/libtest.vhd
  -- Reading file C:\Exemplar\Leonardo\data\standard.vhd for
         unit standard
  -- Loading package standard into library std
  -- Reading vhdl file D:/FOO/SUB/libtest.vhd into library MYLIB
  -- Loading package constpack into library MYLIB
  Info:"D:/FOO/SUB/libtest.vhd" Nothing to synthesize
LEONARDO{2}: read -work MYLIB -format VHDL D:/FOO/chk.vhd
  -- Reading vhdl file D:/FOO/chk.vhd into library MYLIB
  -- Reading file C:\Exemplar\Leonardo\data\std_1164.vhd for
          unit std_logic_1164
  -- Loading package std_logic_1164 into library ieee
  -- Loading entity chk into library MYLIB
  -- Loading architecture rtl of chk into library MYLIB
  -- Compiling root entity chk(rtl)
LEONARDO{3}:
```

この操作ではエラーを発生せず、論理合成も正しく行われています。

ワークライブラリを指定する

この同じソースファイルを、レオナルドのファイル読込み手順でワーク・ライブラリ名の指定を変えてみると、以下のような結果になります。

```
LEONARDO{1}:read -work MYLIB -format VHDL D:/FOO/SUB/libtest.vhd
  -- Reading file C:\Exemplar\Leonardo\data\standard.vhd for
     unit standard
  -- Loading package standard into library std
  -- Reading vhdl file D:/FOO/SUB/libtest.vhd into library MYLIB
  -- Loading package constpack into library MYLIB③
  Info: "D:/FOO/SUB/libtest.vhd" Nothing to synthesize
LEONARDO{2}: read -work WORK -format VHDL D:/FOO/chk.vhd
  -- Reading vhdl file D:/FOO/chk.vhd into library WORK
  -- Reading file C:\Exemplar\Leonardo\data\std_1164.vhd for
     unit std_logic_1164
  -- Loading package std_logic_1164 into library ieee
  -- Loading entity chk into library WORK
  "D:/VSYS_DOC/ieeetest/chk.vhd",line 16: Warning, constpack is
     not declared in library mylib.④
  "D:/VSYS_DOC/ieeetest/chk.vhd",line 24: Error, cnst_0010 is
     not declared.
  Error in file D:/FOO/chk.vhd.-- Error found in VHDL source
LEONARDO{3}:
```

最初にパッケージを読込んだレオナルドは、この結果をキーボードから打ち込まれたとおりに大文字の MYLIB データベースに収納③しています。

そして次のソース・ファイルを読込んだ時、ソース内①②で、いくら大文字のMYLIB を指定していても、ＶＨＤＬではソース記述中の大文字／小文字は無視されることになっていますので、これは無視されてしまい、レオナルドは小文字のmylib ライブラリから目的のパッケージを探そうとしてエラーを発生している④事が分かります。

キーボードからライブラリ名を指定する時に、大文字小文字に注意して処理し以下のように無事に期待どおりのコンパイルを完了します。

```
LEONARDO{3}:read -work mylib -format VHDL D:/foo/SUB/libtest.vhd
  -- Reading vhdl file D:/foo/SUB/libtest.vhd into library mylib
  -- Loading package constpack into library mylib
  Info: "D:/foo/SUB/libtest.vhd" Nothing to synthesize
LEONARDO{4}: read -work work -format VHDL D:/foo/chk.vhd
  -- Reading vhdl file D:/foo/chk.vhd into library work
  -- Loading entity chk into library work
  -- Loading architecture rtl of chk into library work
  -- Compiling root entity chk(rtl)
LEONARDO{5}:
```

まとめ

ＶＨＤＬ ソース記述の中で、ライブラリ名などに使用されている大文字・小文字の違いは無視され、基本的に小文字に変換されてレオナルドのインターフェースに返されます。

レオナルドのコマンド体系では、大文字／小文字は区別されるので、このようなオペレーション・ミスを発生することがあります。

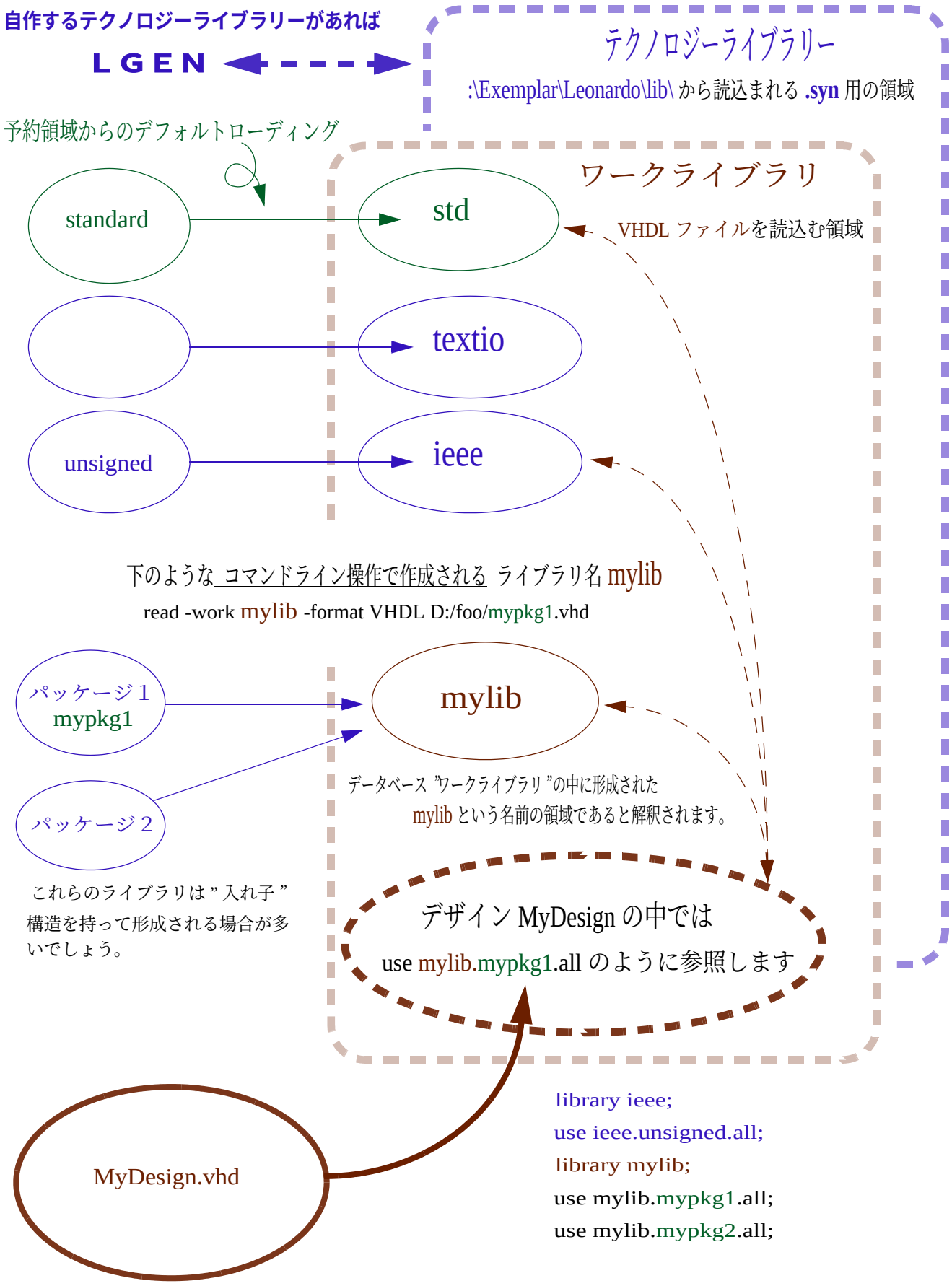
ライブラリ文節を使用し、オペレーションで正しくワークライブラリを指定してコンパイルができても、その過程で作成されたライブラリはレオナルドの場合揮発性のもので保存して再利用するという方法はありません。

これは別にユーザー・ライブラリだけに限った事ではなく、std ／ ieee ライブリーの読込み過程でも、レオナルドは毎回、ソース・ファイルからの読込みからコンパイルを開始している点に注意します。

効果的にワーク・ライブラリを管理するには、Ｔｃｌ スクリプトあるいはそれに準じたメイクファイルのような適切なバッチ処理の手段を考える必要があると思われます。

これまでに提出してきた私共の ＶＨＤＬ ソースでは、全て単一の work しか使わないようにして来ていますので、このような混乱は回避できていると思います。

イメージ図


自作するテクノロジーライブラリーがあれば
LGEN
テクノロジーライブラリー
:\Exemplar\Leonardo\lib\ から読込まれる .syn 用の領域
予約領域からのデフォルトローディング
ワークライブラリ
VHDL ファイルを読込む領域
standard
std
textio
unsigned
ieee
下のような コマンドライン操作で作成される ライブラリ名 mylib
read -work mylib -format VHDL D:/foo/mypkg1.vhd
パッケージ１
mypkg1
パッケージ２
mylib
データベース "ワークライブラリ"の中に形成された
mylib という名前の領域であると解釈されます。
これらのライブラリは" 入れ子 "
構造を持って形成される場合が多
いでしょう。
デザイン MyDesign の中では
use mylib.mypkg1.all のように参照します
MyDesign.vhd
library ieee;
use ieee.unsigned.all;
library mylib;
use mylib.mypkg1.all;
use mylib.mypkg2.all;

備考

ライブラリとして、どのような ＦＰＧＡ 内部資源（エレメント）を提供するか、これはザイリンクスの場合とアルテラの場合を比較してかなりの違いがあるように思われます。

例えばバウンダリスキャンなどの ＪＴＡＧ エレメントは、アルテラではフィッター側によって全て管理されるのに対して、ザイリンクスでは ＶＨＤＬ ソースでの記述が行われるようになっています。（コンフィギュレーション用のエレメントについては、どちらもフィッタ側のセッティングだけで処理が可能です）

ＪＴＡＧ／ＢＳＴ は元々は ＶＨＤＬ 言語のサブセットとして規格化された経緯があったのですが、現在では ＪＡＭ 規格（ＩＥＥＥ １１４９．１）の中にその一部として統合されています。

基板の履歴やバージョン情報などをシステム側から管理する為のウエス・コードなどの名前は ＪＴＡＧ／ＢＳＴ 規格のころに設定されたものですが、基本的な内容はすべて ＪＡＭ に引き継がれているようです。

ＦＰＧＡ ベンダーのウエブでの Ｑ ＆ Ａ をチェックしてみますと、現実には ＪＴＡＧ／ＪＡＭ の活用は今始まったばかりのように（１９９７年度のあるロットまではＪＴＡＧピンのテストをせずに出荷されていた部品ファミリーがあった、など）見受けられますが、ＩＳＰ やテストが ＪＡＭ 規格に包括され、ＪＡＭ プレイヤーなどのソースが無料でウエブサイトからダウンロードできるようになっている現在では、VHDL の立場からは、何も触れないほうが正しいありかたであるように思われます。