

VHDL での FPGA 開発について

VHDL 論理合成の環境で最初に注意すべき点は

シミュレーション・タイミング検証の為の HDL 記述と論理合成の為の HDL 記述とを明確に区別して意識することが必要であること、

ライブラリーには、予めテクノロジー・ライブラリとして提供されるデバイス・ファミリーに固有の構成要素（たとえばグローバル・クロック・ドライバーなど）があり、一方ではアプリケーション側の RTL 記述によって生成される構成要素（たとえばパッケージ化された カウンタなど）があり、共にライブラリと呼ばれ、ワーク・ディレクトリにコンパイルしてから使用されるので混同し易いことであるように思います。

また、シミュレータとして `Vsystem` を使った場合のライブラリ/ワークの概念もはっきりと意識して使う必要があります。

小さなデバイスに多くを要求する場合

また、ACTEL - ACT 3 ファミリーなど配線資源に種々の制約があるようなデバイスでは、タイミング特性をチューンする為に意図的に階層構造をフラットにして最適化を図った場合に生じる LVT 資源の不足問題があります。

これは、ターゲット側のフィッター/コンパイラが、資源としてセル・ピンの利用率などを認識してフィット可能と判断し、長時間の配置配線の実施に入った後でギブアップする状況になりますが、事前の予測はかなり困難と言えます。

階層構造を保持して LVT 資源の利用を制限する事により、ルーターのギブアップを回避する事が出来る場合がありますが、このような場面に直面した場合の基本的な解決手法としては、設計サイズを小さく抑える工夫に尽きるものと思われます。

ただし、これらの状況は、ターゲットデバイスのフィッターの改版や論理合成プログラムの改版によって大きくその様相が変化します。

旧版での対応策が新版では逆にパフォーマンスを低下させる場合もありますので十分な注意が必要です。

これまでの基本的な設計方針として

子葉末節の部分については、今回の私共の設計方針では、使用するライブラリを極めて限定し、型変換についてのパッケージなどについても極力、ieee 標準ライブラリを直接参照するようにしています。

これらの型変換関数などは自分独自のパッケージを作成し、その中で新しい関数を定義して使えばよいのですが、このような手法は緊密な連絡を取り合っているプロジェクトの中で有効とされているもので、疎な結合連絡形態で作業を進める場合には、今回のような方法で混乱を避けるほうが良いように思っております。

双方向性のバス信号については、初期のレオナルドで発生した不具合経験から、複数の内部バスを階層最上位のマルチプレクサに接続した後に `std_logic_vector` に変換してから外部バスヘトライステート形式で接続するように統一してあります。

VHDL の処理系によっては（例えばアルテラ社）32ビットの取扱いを行わないものがありますが、このような場合には一旦レオナルドで EDIF に落としてインポートすれば済む内容なので、わざわざパッケージを作ってオーバーライドするような事はしておりません。

使用開始初期のレオナルド 4.0x では、バスのメンバーの取扱いに問題があり、バス・メンバーの並び方向指定に `downto` 構文を使用しても、ターゲット側に正しく伝達解釈されず、ビットの並びが逆転してしまうなどの問題がありましたが、現在の版 4.2x ではこれらの問題はひととおり解決されています。

同様に、ACTEL ターゲットでの負荷バランスの問題も現在のレオナルドでは解決されていますし、CLKINT マクロの多重生成の障害も発生しなくなっています

ザイリンクスの場合についても同様にレオナルドの改版とザイリンクスのフィッタの能力向上により大幅な能力の向上がありました。

xact6.0.2 版（`xfn` ネットを取り込んでいた）の時に入り切らなかった規模の設計が、m1.4 版（`edif` 形式ネットの取り込みができる）では殆ど苦労せずに同一のデバイスにフィットさせることができるようになっていきます。

ただし、ファウンデーションの m1.4 版は**基本的に回路図入力を前提**にしており、ネットリストを全て EDIF 形式で吐き出すレオナルドのようなサードパーティ製品からのエントリーではプロジェクトファイルの作成や、ファイル名ルールに少し注意が必要です。（おそらくアライアンスでの m1.4 には不必要な事だろうという事ですが私共では検証できません）

この資料の構成

添付した資料は、これからレオナルド / `Vsystem` を使って実際にシミュレーション / フィティングを行う為に必要な作業の流れと、余りマニュアルに書いていない `TIPS` を説明するもので、具体例としてアルテラの事例から入り、実例を示した上でザイリンスクでの事例を説明するようにしました。

設計の具体例として実例を利用するのには、余りにも大きな時間と労力が必要とされますので、

<code>TOP.VHD</code>	という簡単なターゲットの例
<code>TB.VHD</code>	という、この <code>TOP</code> 用のテストベンチ

を作成し、比較的詳しい解説資料を作成した上で、これを `leonardo` でコンパイルし、アルテラの `maxplusII` でフィットするまでの実例ファイルと、フロー解説を `A3` の解説図に纏めてみました。

次に、同じ設計をザイリンクスに適用した場合のフロー & `TIPS` を `A3` の解説図に作成し、実例ファイルを作成してあります。

特に、`Vsystem` 環境は `CRT` 上での解析を目的としたもので、詳細な波形情報や、カーソルで検査したタイミングデータなどの設計資料的な内容の保存はできませんので (`WAVE.PS` ファイルは作成できますがアウトラインしか判りません) おそらく、このシステムの設計意図にある利用法は、設計者が `CRT` 上で判断した事を証拠に残す必要はなく、そのような場合は `TEXTIO` を利用するものと想定しているのではないかと思います。

この資料がお役に立つ事を願っております。

```

1  -----
   -- N-bit COUNTER COMPARATOR   for testbench lessons.
   -----

5  library IEEE;
   use IEEE.std_logic_1164.all;
   use ieee.std_logic_ARITH.all ;
   use ieee.std_logic_signed.all ;

10 Entity TOP is
    port (DBUS   : inout Std_Logic_Vector(7 downto 0);
          CLK    : in Std_Logic;
          nRD    : in Std_Logic;
          nWT    : in Std_Logic;
15         nCLR   : in Std_Logic;
          CENBL  : in Std_Logic;
          RSEL   : in Std_Logic;
          CVAL   : out Std_Logic_Vector(7 downto 1);
          EQU    : out Std_Logic );

20 end TOP;

Architecture RTL of TOP is
    signal DRO      : Std_Logic_Vector(7 downto 0);
    signal CompReg  : Std_Logic_Vector(7 downto 0);
25     signal CntrReg : Std_Logic_Vector(7 downto 0);
    signal Counter  : Integer range 0 to 255;  --Integer'high;

begin
    GEN1:for I in 0 to 7 GENERATE
30         DBUS(I)<=DRO(I) when nRD='0' else 'z';
    end GENERATE GEN1;

    CntrReg<= Conv_Std_Logic_Vector(Counter,8);

35     DRO<= CompReg  when RSEL='0' else CntrReg;

   -----
   RegLdProc: Process (CLK,DBUS,nWT,CompReg)
   -----

40 begin
    if(CLK'EVENT and CLK='1') then
        if(nWT='0') then    CompReg<= DBUS;
        else                CompReg<= CompReg;
        end if;
45     end if;
    end Process RegLdProc;

   -----

50 CountProc: Process (CLK,CENBL,nCLR,nWT,Counter,CompReg)
   -----

begin
    if(nCLR='0' or nWT='0') then
55         Counter<=0;
    elsif(CLK'EVENT and CLK='1') then
        if(CENBL='1') then Counter<= Counter+1;
        else                Counter<= Counter;
        end if;
60     end if;
    CVAL<= Conv_Std_Logic_Vector(Counter,7);
    end Process CountProc;

   -----

65     EQU<='1' when (Counter= Conv_Integer(CompReg))
        else '0';

end RTL;

```

こういうパラメータはジェネリックとして与えるほうが良いかも知れません。

こういう記述は安易に使わないようにしたほうが無難でしょう。(きちんとした体系を作って設計するなら別) 実際にこれを試してみると分かりますが深い階層下で起きると厄介なバグになりかねません(--)

意図して遅延のある信号を取り出しています

altera tb.vhd

```
1 Entity TestBench is  
end;
```

```
5 library IEEE;  
use IEEE.std_logic_1164.all;  
use ieee.std_logic_ARITH.all ;  
use ieee.std_logic_signed.all ;
```

被テスト設計をコンポーネント宣言し
(実体は別にコンパイルされている)

```
10 Architecture TestMain of TestBench is
```

```
component TOP -- This is the TARGET design ----  
port (DBUS : inout Std_Logic_Vector(7 downto 0);  
CLK : in Std_Logic;  
nRD : in Std_Logic;  
nWT : in Std_Logic;  
nCLR : in Std_Logic;  
CENBL : in Std_Logic;  
RSEL : in Std_Logic;  
CVAL : out Std_Logic_Vector(6 downto 0);  
EQU : out Std_Logic  
);  
end component;
```

```
25 Type TestRecordT is record -- TestVector format definition.  
nRD,nWt : Std_Logic ;  
CE,RSEL : Std_Logic ;  
DBUS : Std_Logic_Vector(7 downto 0) ;  
30 EOVS : Std_Logic ;  
End record;
```

印加したいテストベクタ値を設定する

```
Type TestArrayT is array(positive range <>) of TestRecordT;
```

```
35 constant TestPattern : TestArrayT := (  
(nRD=>'1',nWT=>'1',CE=>'0',RSEL=>'0',DBUS => "00000000",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'0',RSEL=>'0',DBUS => "00000000",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'0',RSEL=>'0',DBUS => "00000000",EOVS=>'0'),  
40 (nRD=>'1',nWT=>'0',CE=>'1',RSEL=>'0',DBUS => "00000101",EOVS=>'0'),  
(nRD=>'0',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "00000000",EOVS=>'0'),  
(nRD=>'0',nWT=>'1',CE=>'1',RSEL=>'1',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1  
プリンタドライバのミス  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
45 (nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
50 (nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
55 (nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
60 (nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'1',nWT=>'1',CE=>'1',RSEL=>'0',DBUS => "ZZZZZZZZ",EOVS=>'0'),  
(nRD=>'0',nWT=>'1',CE=>'1',RSEL=>'1',DBUS => "ZZZZZZZZ",EOVS=>'1') );
```

altera tb.vhd

```
65     signal INDX: positive :=1;           -- Index of TestPattern(INDX) 0 not allowed

constant MAXCOUNT : INTEGER:= 3; -- for Patern-Gen repeat count value
signal   CLKCOUNT : INTEGER:= 0; -- CLOCK counter to increment Pattern-Gen

70     -- Internal signals for Test-Bench operations -----

signal DBUS      : Std_Logic_Vector(7 downto 0);
signal CVAL      : Std_Logic_Vector(7 downto 1);
signal CLK,nRD   : Std_Logic;
signal nWt,nCLR  : Std_Logic;
75     signal CE,RSEL : Std_Logic;
signal EQU       : Std_Logic;

-----

80     Begin  -- Test Bench MAIN.  Connect target entity with test vector signals --
-----

nCLR    <= '0', '1' after 200 ns,'0' after 1 us,'1' after 1.2 us;

85     -----
CLK_P : Process  -- Clock pulse generator
Begin          CLK<='0';  wait for 25 ns;
              CLK<='1';  wait for 25 ns;
90     End Process CLK_P;
-----

-----
95     CLKCOUNT_P : process(CLKCOUNT,CLK)  -- @CLKCOUNT=0 increment (INDX)
-----
-- You can modify TestStim process to control INDX increment timing NOT
-- with CLKCOUNT=0 event BUT with some responce signal from the Target.
-- This will provide a TestBench-Target handshaking.
-----

100    Begin
        if(CLK'EVENT and CLK='1') then
            if(CLKCOUNT=MAXCOUNT) then  CLKCOUNT<=0;
            else                            CLKCOUNT<=CLKCOUNT+1;
            end if;
105    end if;
End Process CLKCOUNT_P;
-----

-- provide stimulus -----
110    TestStim: process(CLKCOUNT)  -- Read Test-Vectors into signals.
-----
variable Vector : TestRecordT;
variable EOVS   : Std_Logic;
Begin
115    if(CLKCOUNT=0) then
        Vector := TestPattern(INDX);
        nRD    <= Vector.nRD;           -- apply the stimuli
        nWT    <= Vector.nWT;
        CE     <= Vector.CE;
120    RSEL    <= Vector.RSEL;
        DBUS   <= Vector.DBUS;
        EOVS   := Vector.EOV;
        INDX   <= INDX+1;
        if(EOV='1') then assert FALSE report "End.";
125    end if;
    end if;
End Process TestStim;
-----
```

クロックの発生例、当然ジェネ
リックで与えてもよい

ここではテストベクターを CLKCOUNT_P に
合わせて順次読込んでいるが、任意の方法
をとればよい。

被テスト設計をインスタンス化してテストベンチの信号とを接続する。

```

130  UUT: TOP    -- Notice this is the sdf applicant REGION
-----
135  PORT MAP(  DBUS    =>DBUS,
              CLK     =>CLK,
              nRD     =>nRD,
              nWT     =>nWT,
              nCLR    =>nCLR,
              CENBL   =>CE,
              RSEL    =>RSEL,
              CVAL    =>CVAL,
              EQU     =>EQU
              );
140
145  ENL:
-----

```

.sdfのデータを（個別に）適用する対象、レジヨン名がUUTとなる。

```

150  configuration RTL of TestBench is
      for TestMain
        for all: TOP
          use entity work.TOP(RTL);
        end for;
      end for;
155  end RTL;

```

アルテラのバック・アノテート出力 .vho の場合は、フィットしたデバイス名がアーキテクチャ名になるので、これを指定したコンフィギュレーションを作っておく。

```

160  configuration ALTERA_TEST of TestBench is
      for TestMain
        for all: TOP
          use entity work.TOP(EPM7032SLC44_a10);  -- From Top.vho output
        end for;
      end ALTERA_TEST;
165

```

```

170  configuration XILINX_FIT of TestBench is
      for TestMain
        for all: TOP
          use entity work.TOP(STRUCTURE);
        end for;
      end XILINX_FIT;

```

ザイリンクスでのバックアノテートファイルは time_sim.vhd time_sim.sdf の2つになる（バイタル）

time_sim.vhd : フィットしたデバイス内部のネットリスト
time_sim.sdf : 使用されているプリミティブの遅延データ

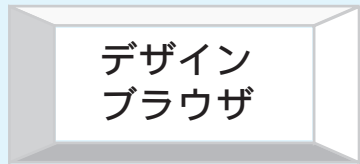
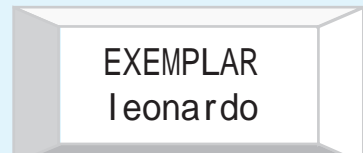
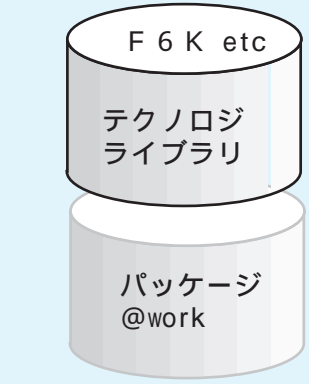
vsimするとき、エンティティを指定する方法と、コンフィギュレーションを指定する方法のどちらも使えるので、ワーク内にコンパイルしてあれば差はない。エンティティから指定する方法を取った場合には、アーキテクチャ名も指定する事になる。

leonardoのワーク

! 注意 top.vhdなどのように設計と同じ名前でラッパーを作成しないようにします

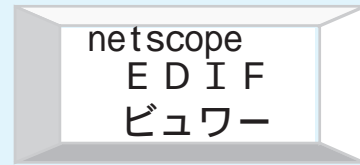
WRAPPER

最上位のエンティティをStdLogicで再定義
目的はシミュレーション。手書きでも可。



階層の組み替え (最適化対象の構成変更)
インスタンス・ポート名・プリミティブの確認
特に配線資源が微妙な場合に有効

タイミング制約を与え最適化し、故意にスラックさせ、REPORT_DELAYをViewCriticalPathチェックで実行してハイライトファイル (ex temp.hlt) しながらネットスコープ起動。ハイライトボタンで目的ネットを検索すると便利です。

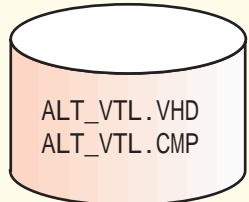


この段階で、すでにRTL記述ではなくなっている事に注意します

コンパイラ/インターフェース/EDIFネットリーダ設定でEXEMPLARを指定
SHOW_LMF マッピングをチェックしておくでOPミスに気が付く事が有ります。

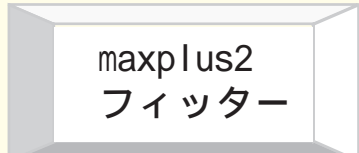
メニュー/コンパイラ/インターフェース/VHDLネットライタの設定で参照されるバイタル・ライブラリ。
インストールしてあればよい(善)です

アルテラの場合、バイタルはmaxplusIIに読込ませて利用できます

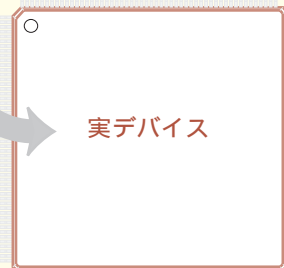


V8.3までのmaxplusIIのVHDLリーダはパーサが貧弱なので使用しないほうがよいと思いますが、コンパイラがハングした時はVsystemやleonardoで調べると文法ミスなどのハング原因が判る事が有ります

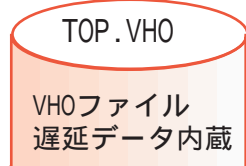
アルテラでは同じデバイス・パッケージであっても速度ランクが異なる場合には再度コンパイルする事になるので、スマート・コンパイルオプションを使っておくともいいかも知れません。(ザイリンクスの場合はPAR以後の処理で済みます)



基本的にはWYSIWYGでフィットさせます



コンパイラ/インターフェース/VHDLライタセッティングでVHOを選択、同じくインターフェース/VHDLライタをセットしてコンパイルします。
アルテラの場合のシミュレーションはこのVHOファイルだけで十分です



SDFファイルを使う場合はVSIM/SDFで適用するREGIONを指定して使います。

実フィットデバイス名がアーキテクチャ名として出力されるので、TBではこの新しいバインディングを使います。

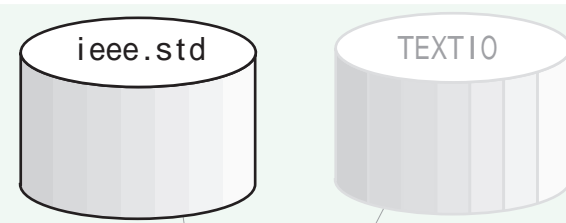
このファイルは使用プリミティブの個別タイミングデータを返すものでネット情報は持っていません。
どうしてもsdfを使いたい場合には、maxplusII/vhdl87/vital/v3_0でvlibしてworkを作成し、alt_vtlを全てvcomし、これをvlibでマッピングして使います。



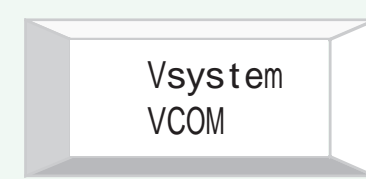
LEONARDO-ALTERA-VSYSTEM 簡略化作業フロー概念図

Systems Workshop Inc 1998/7/25

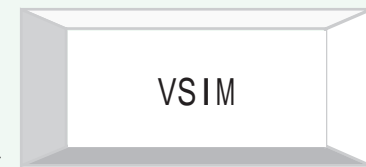
VSYSTEMのワーク



ライブラリマッピングしておく



RTLモデル

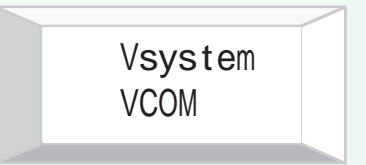


テキスト記述を修正してバインディングを切替えます。

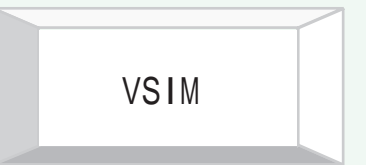
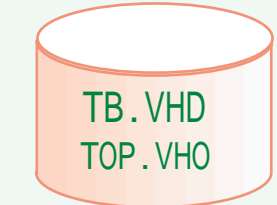
エンティティ・コンフィギュレーション(RTL)を指定してシミュレートしますが、論理合成とは関係なく処理されます。



GATEモデル



ここで読み込むのはtop.VHOファイルです (top.vhdをVCOMするとRTLモデルになります) その次にtb.vhdをVCOMしてVSIMします。



VSIMのwave/メニュー/signals/addtowaveformでsignals in designを選択すると全信号がモニターできます

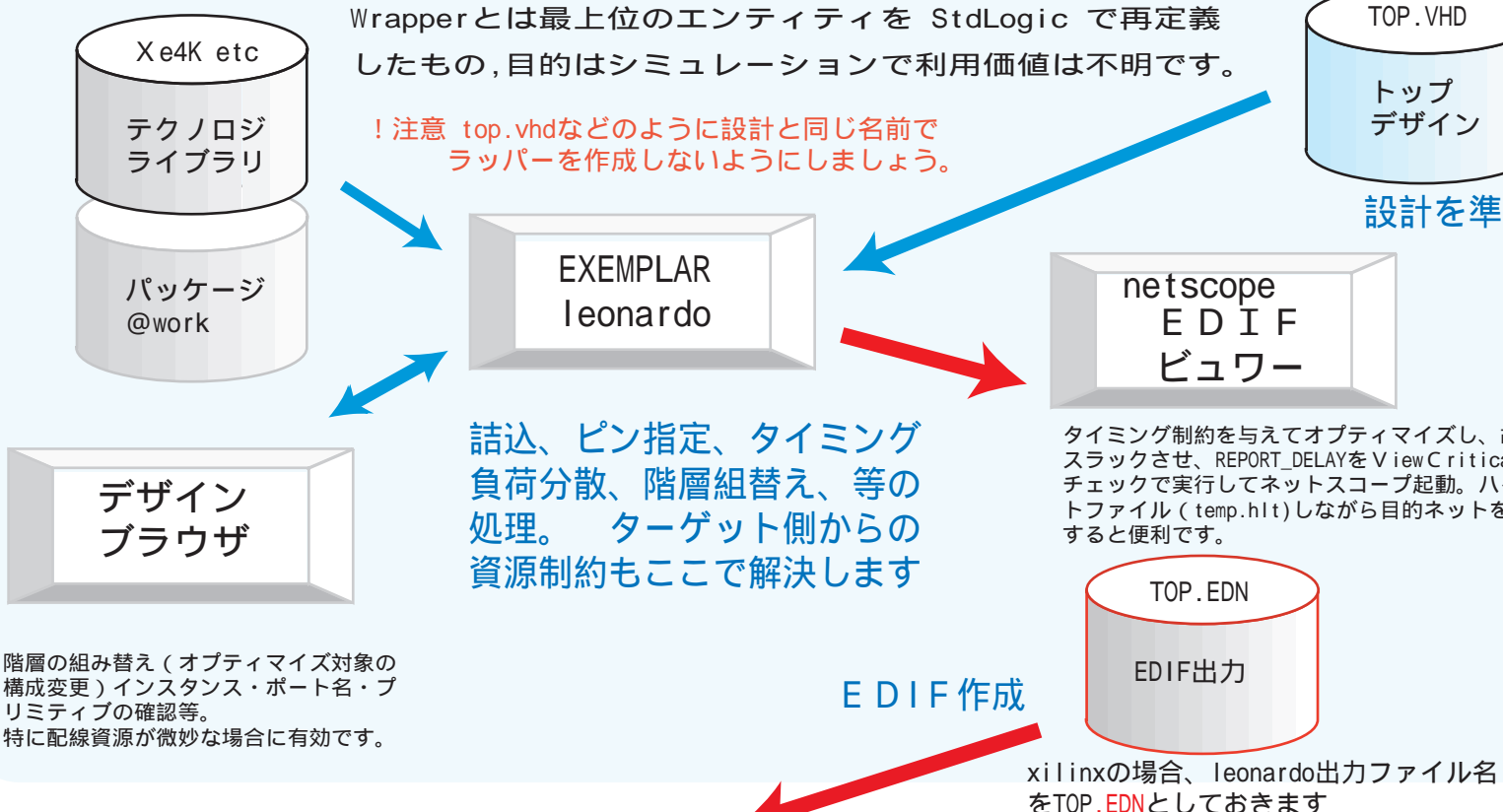
適用すべきバインディングはもはやRTLではない事に注意します。アルテラの場合、対象となるアーキテクチャ名=デバイス名になります。

maxplusIIでは処理が困難な外部遅延素子(メモリなど)のモデル化が容易です。

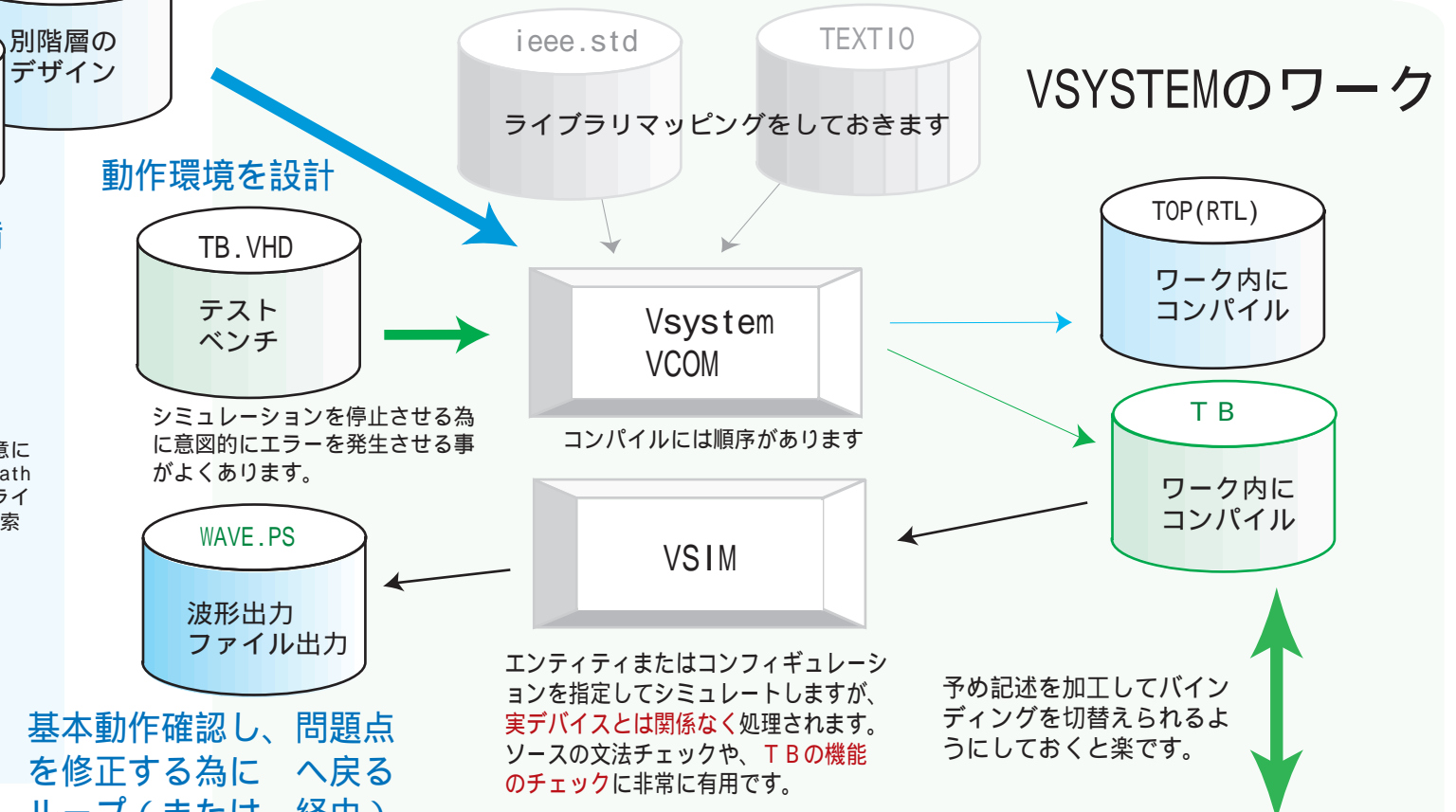


maxplusIIのワーク

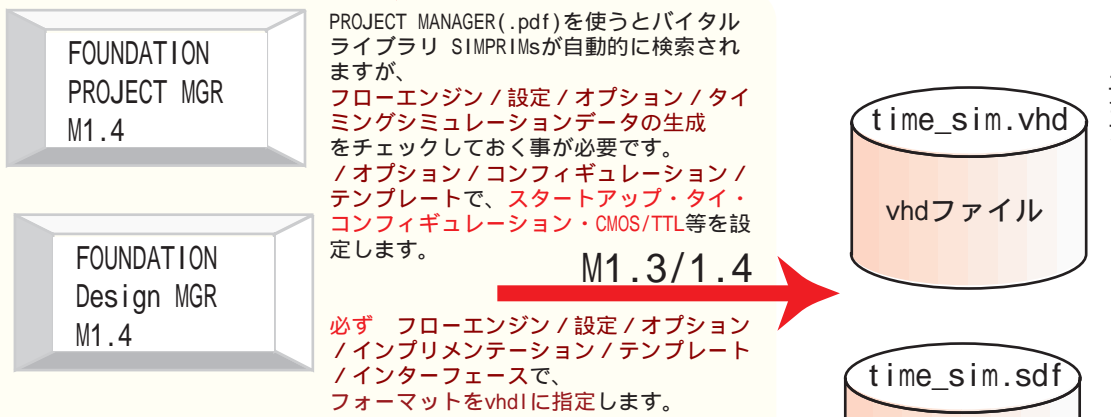
leonardoのワーク



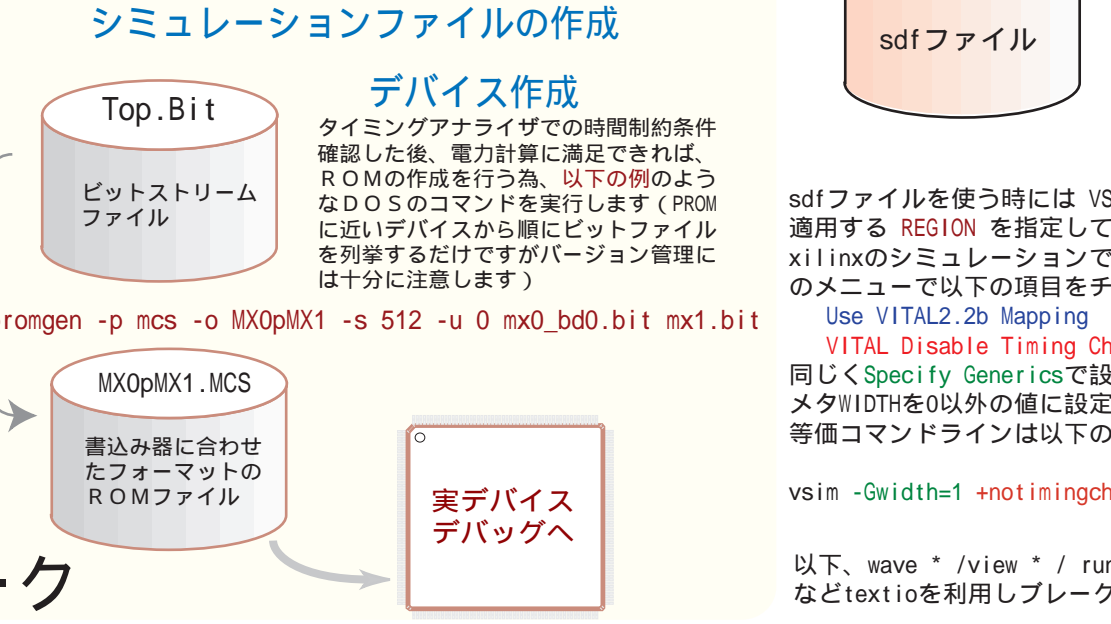
leonardo-Xilinx-VSYSTEM 簡略化作業概念図



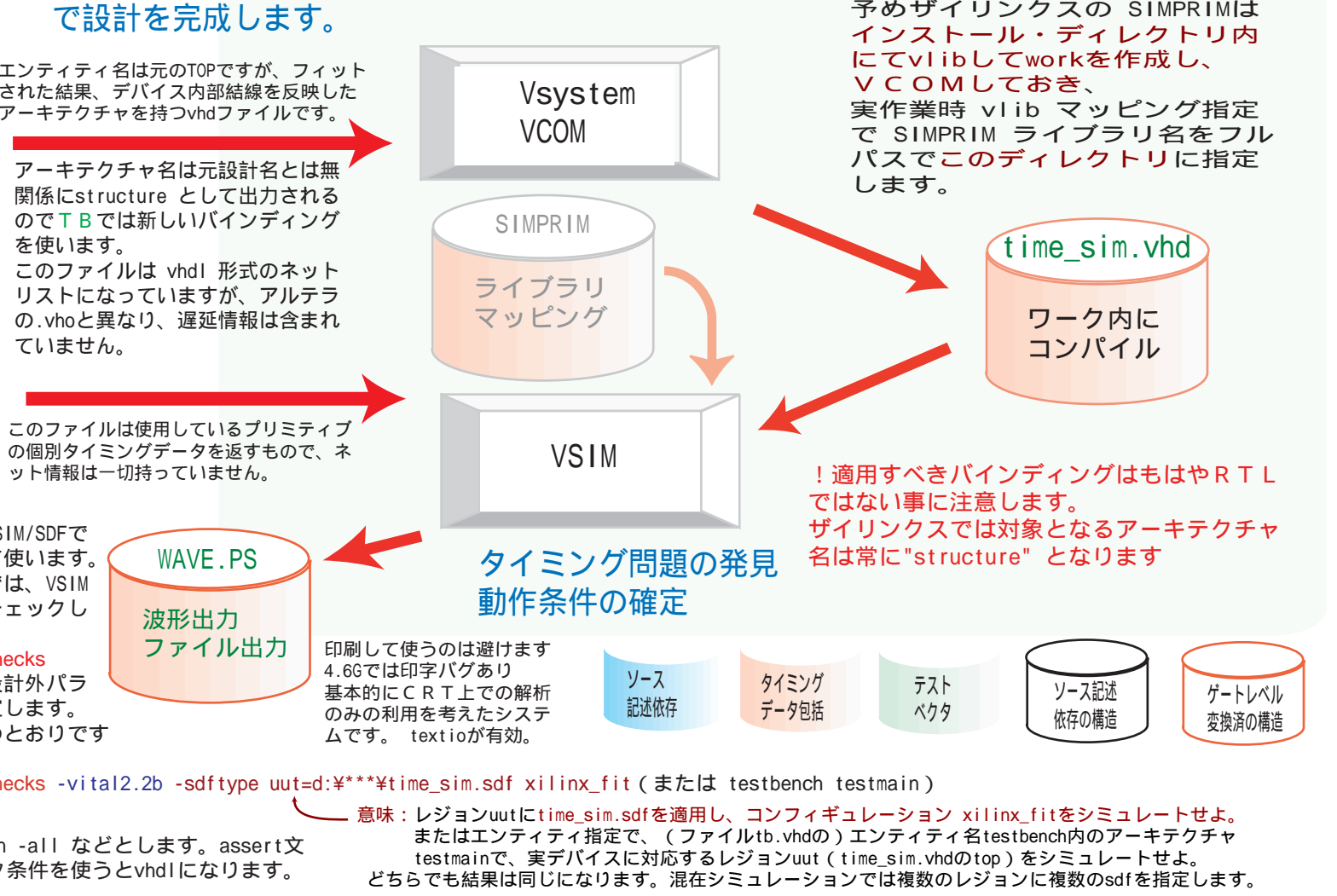
ファウンデーションでの処理の場合にはプロジェクトマネージャからメニューで /NEW PROJECT/としてフォルダを作成しますが、この操作でディレクトリ階層が1つ下がりますのでDOCUMENT/ADDで1つ上の階層の TOP.EDN ファイルを指定し、インプリメントアイコンでデザインマネージャを起動します。この時、EDIFネットがSCHネットより古いから更新するか?と問い合わせて来ますので必ずNOとします。



もし、ツールがうまく動作しない場合フローエンジンのコマンドヒストリをテキストで開き、同じ内容をDOS窓から実行すると処理できる場合があります (トランスレート/マップまで) このような場合には、インストールの問題か、プロジェクトファイルの問題か、環境変数の設定が原因かをつきとめて対策とします。(当然、ルータはDOS窓からは実行できません) 問題の発生した場所がトランスレートやマッピングの段階であれば、ログファイルをよく検討して、プリミティブの使い方などの基本的な間違いを検出できることがあります。



Xilinxのワーク



```

1  -----
   -- N-bit COUNTER COMPARATOR   for testbench lessons.
   -----

library IEEE;
5  use IEEE.std_logic_1164.all;
   use ieee.std_logic_ARITH.all ;
   use ieee.std_logic_signed.all ;

Entity TOP is
10  port (DBUS   : inout Std_Logic_Vector(7 downto 0));
       P_CLK   : in Std_Logic;
       nRD    : in Std_Logic;
       nWT    : in Std_Logic;
       P_nCLR : in Std_Logic;
15  CENBL    : in Std_Logic;
       RSEL   : in Std_Logic;
       CVAL   : out Std_Logic_Vector(7 downto 1);
       EQU   : out Std_Logic );

end TOP;

20  Architecture RTL of TOP is
   -- for Xilinx fitting -----
   COMPONENT BUFG  PORT( I: IN Std_Logic;  O: OUT Std_Logic ); END COMPONENT ;
   -----

25  signal    DRO      : Std_Logic_Vector(7 downto 0);
   signal    CompReg  : Std_Logic_Vector(7 downto 0);
   signal    CntrReg  : Std_Logic_Vector(7 downto 0);
   signal    Counter  : Integer range 0 to 255; -- Integer'high;
   signal    CLK,nCLR : Std_Logic;

30  Begin
   -- for Xilinx fitting -----
   ux0: BUFG  PORT MAP( I => P_CLK, O => CLK  );
   ux1: BUFG  PORT MAP( I => P_nCLR,O => nCLR );
   -----

   GEN1:for I in 0 to 7 GENERATE
       DBUS(I)<=DRO(I) when nRD='0' else 'Z';
   end GENERATE GEN1;

40  CntrReg<= Conv_Std_Logic_Vector(Counter,8);
   DRO<= CompReg  when RSEL='0' else CntrReg;

   -----

45  RegLdProc: Process(CLK,DBUS,nWT,CompReg)
   -----
   begin
       if (CLK'EVENT and CLK='1') then
           if(nWT='0') then    CompReg<= DBUS;
50           else              CompReg<= CompReg;
           end if;
       end if;
   end Process RegLdProc;

   -----

55  CountProc: Process(CLK,CENBL,nCLR,nWT,Counter,CompReg)
   -----
   begin
60       if(nCLR='0' or nWT='0') then
           Counter<=0;
       elsif(CLK'EVENT and CLK='1') then
           if(CENBL='1') then Counter<= Counter+1;
           else              Counter<= Counter;
65           end if;
       end if;
       CVAL<= Conv_Std_Logic_Vector(Counter,7);
   end Process CountProc;

   -----

70  EQU<='1' when (Counter= Conv_Integer(Compreg))
       else '0';

end RTL;

```



```

70      -- Internal signals for Test-Bench operations -----
signal DBUS      : Std_Logic_Vector(7 downto 0);
signal CVAL      : Std_Logic_Vector(7 downto 1);
signal CLK,nRD   : Std_Logic;
signal nWT,nCLR  : Std_Logic;
75 signal CE,RSEL  : Std_Logic;
signal EQU       : Std_Logic;

-----

80 Begin  -- Test Bench MAIN.  Connect target entity with test vector signals --
-----

nCLR  <= '0', '1' after 200 ns,'0' after 1 us,'1' after 1.2 us;

85 -----

CLK_P : Process      -- Clock pulse generator
-----
Begin      CLK<='0';   wait for 25 ns;
           CLK<='1';   wait for 25 ns;
90 End Process CLK_P;
-----

CLKCOUNT_P : process(CLKCOUNT,CLK)  -- @CLKCOUNT=0 increment (INDX)
-----
-- You can modify TestStim process to control INDX increment timing NOT
-- with CLKCOUNT=0 event BUT with some response signal from the Target.
-- This will provide a TestBench-Target handshaking.
-----

100 Begin
      if(CLK'EVENT and CLK='1') then
          if(CLKCOUNT=MAXCOUNT) then CLKCOUNT<=0;
          else CLKCOUNT<=CLKCOUNT+1;
          end if;
105     end if;
End Process CLKCOUNT_P;
-----

-- provide stimulus -----
110 TestStim: process(CLKCOUNT)  -- Read Test-Vectors into signals.
-----
variable Vector : TestRecordT;
variable EOVS   : Std_Logic;
Begin
115     if(CLKCOUNT=0) then
          Vector := TestPattern(INDX);
          nRD    <= Vector.nRD;           -- apply the stimuls
          nWT    <= Vector.nWT;
          CE     <= Vector.CE;
120         RSEL  <= Vector.RSEL;
          DBUS   <= Vector.DBUS;
          EOVS   := Vector.EOV;
          INDX   <= INDX+1;
          if(EOVS='1') then assert FALSE report "End.";
125         end if;
      end if;
End Process TestStim;
-----

130 -----

UUT: TOP  -- Notice this is the sdf applicant REGION
-----

```

```
135     PORT MAP(   DBUS    =>DBUS ,
                 P_CLK   =>CLK ,
                 nRD     =>nRD ,
                 nWT     =>nWT ,
                 P_nCLR  =>nCLR ,
                 CENBL   =>CE ,
                 RSEL    =>RSEL ,
140                 CVAL   =>CVAL ,
                 EQU     =>EQU -- ,
                 );
```

```
145     END;
```

```
-----
-----
150     configuration RTL of TestBench is
         for TestMain
             for all: TOP
                 use entity work.TOP(RTL);
             end for;
         end for;
155     end RTL;
```

```
-----
-----
160     configuration ALTERA_FIT of TestBench is
         for TestMain
             for all: TOP
                 use entity work.TOP(EPF6016TC144_a3);
             end for;
         end for;
165     end ALTERA_FIT;
```

```
-----
-----
170     configuration XILINX_FIT of TestBench is
         for TestMain
             for all: TOP
                 use entity work.TOP(STRUCTURE);
             end for;
         end for;
     end XILINX_FIT;
```