VHDL-１０

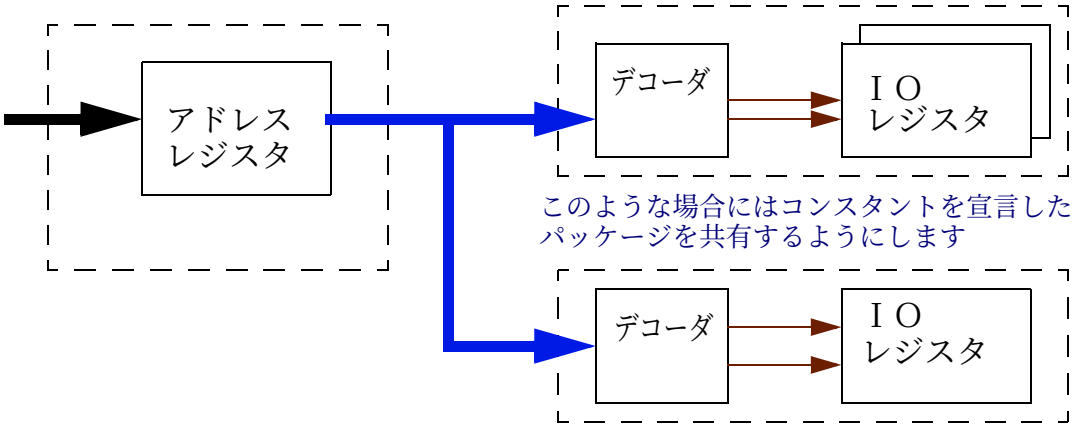
雑談　enum 型：デコーダでの利用について

この文書の目的

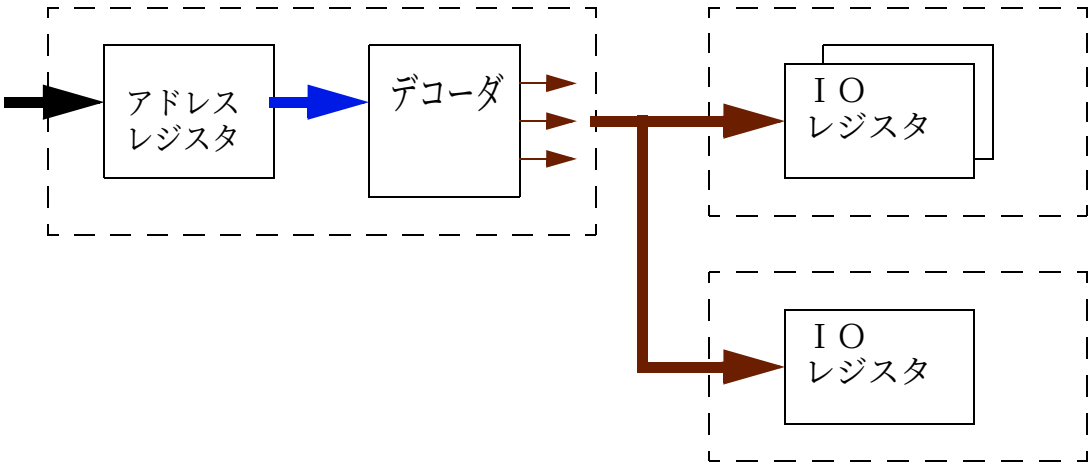
複数のユニット（エンティティ・アーキテクチャ・ペア）を構造記述で接続して回路を構成するとき、ＩＯ レジスタへなどの選択信号の数が増えてしまい、接続する信号の記述行数が増えてソースを読みにくくしてしまう事が多いのではないかと思います。（特にコンフィギュレーションを書く上で厄介になります）

デバイスの内部接続を少しだけすっきりと記述する方法として

1. 内部アドレス・バスを直接、全てのユニットに接続し、各ユニット毎にデコードして使用する場合と、



2. １つのユニットで一旦、完全なデコードを行っておき、これをバス信号あるいは ｅｎｕｍ 信号にまとめ直してから、それぞれのユニットに接続する方法



の２つを考えてみます。

ここでは、各ユニットはそれぞれ別のファイルに作成されるものとします。

１、各ユニット毎にデコードして使用する場合について

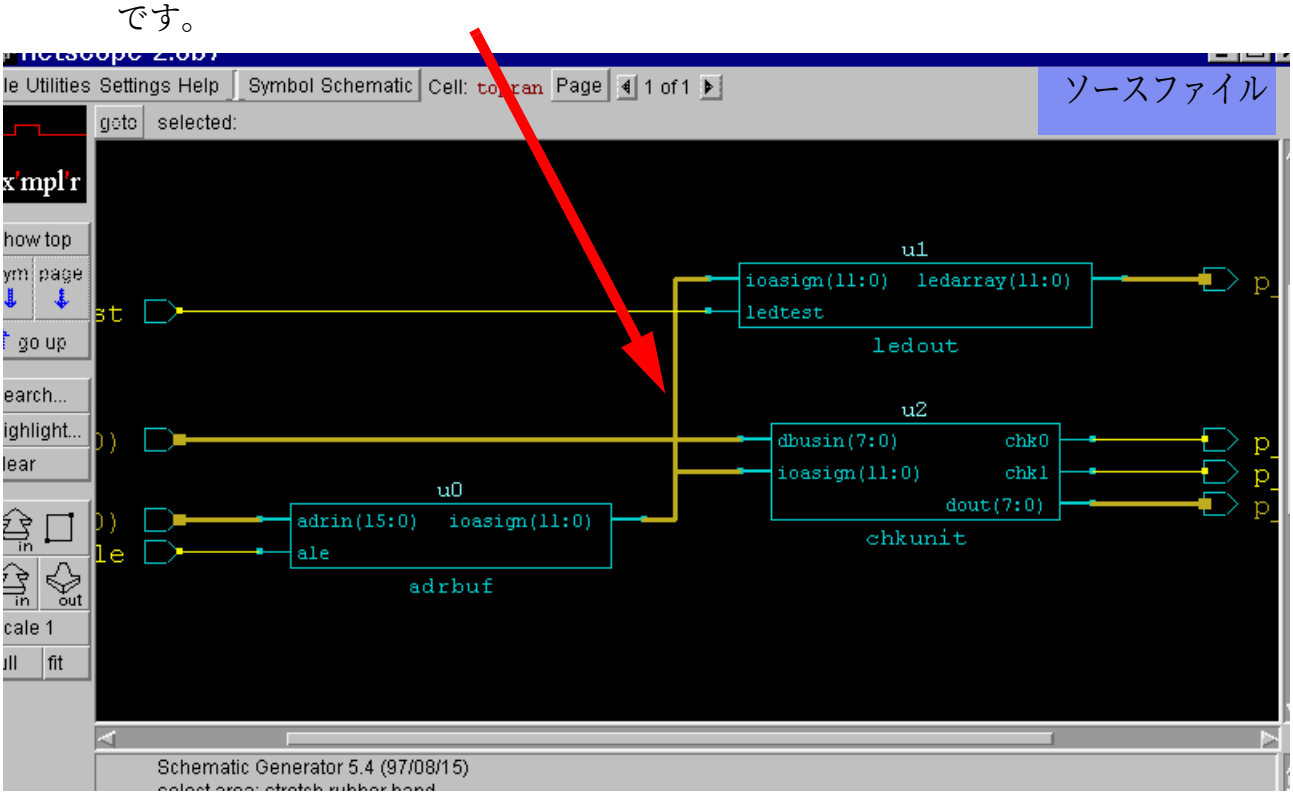
当然、デコードされるアドレス値はコンスタント定義のパッケージにして共有させます。（たとえ複数のモジュールに分割しない場合であっても、このような定数をパッケージにするのは良いアイディアだろうと思います）

しかし、個別のユニット毎にデコーダを設置する設計は（私の場合には）なるべく避けたほうが余計なミスを発生させないで済むように思います。

２、 下図のようにユニット間の接続をバスのようにまとめて配線する事を考えてみたとき、この接続線をバス信号として実現しても、ｅｎｕｍ信号として実現しても、レオナルドで論理合成すると同じような結果を得ます。
ＶＨＤＬ記述を始めた段階では、この ｅｎｕｍ型がなかなか直感的に分からなかった経験から、少し解説を試みてみます。

前回の例題が Soo でしたので、今回の例題は Ran 次回は miki ということで（^^;）よろしくお願いします。下の図は、例題 Ran をレオナルドに読込ませた直後、オプティマイズを掛ける前の状態を示しています。

図には３つのブロックがあり、この３つを繋いでいるバス信号がありますが、これが今回の目的の信号　IOAsign です。　それぞれのブロックには U0、U1、U2 と名前がつけられていますが、これはトップ階層で付けられたインスタンス・ラベルです。



これまでに幾度もパッケージを作成していますが、今回も新しくパッケージを作って、この enum 信号を作成してみます。

enum とはスカラタイプの列挙型の事で、などと言っても別に目新しい概念ではなく、単なる整数も考えてみれば enum の１つです。 加算回路を考えるときに適当に整数を定義して加えたり引いたりしている処理と同じようなものと考えてみれば、enum 定義された信号を回路図イメージで引き回す事にそれほど抵抗を感じなくなるのではないでしょうか。

例えば、アドレス信号を一旦整数に変換して、これを１つの信号として複数のユニットに配線するのと同じような処理になる訳ですが、回路図での設計と言語での設計の間に大きく違う点を挙げるとすれば、もしかしたらこの辺りにも１つのポイントがみつかるかも知れません。（ 図面に 1 本信号を引いて、これは整数！ なんて出来たかしら？）

単なる整数としてアドレス信号を配線しても別にいけない事はないのだろうと思いますが、ユーザー定義の enum 信号を使う事で目的に応じた制約条件をつけて、おかしな演算や間違えた代入などを防止する事ができるだけでなく、その属性が必ずついて回るため、そのメンバーにエイリアス名をつけたバス・シグナルのような使い方ができるようになります。

今回の例題ではアドレスのデコーダ用として以下の型を定義しています

```
Type DecEnumT is (   IOSEL0,   IOSEL1,   CSEL0, CSEL1, CSEL2,  CSEL3,
                     MSEL0,    MSEL1,    MSEL2, MSEL3,
                     EXTRST,   NotUsed   );
```

この信号には、ここで列挙したメンバー以外のものは存在し得ません。

特別に演算などを定義しなければ、この型に属する信号・変数は操作できませんので、同じパッケージとパッケージ・ボディの中にその「設定」についての関数を定義しておきます。

ここでは、１６ビット幅のビット・ベクターをアドレスとして受けて、これをターゲット・システム側の仕様として与えられるアドレス値、範囲値に対応させる関数 Word2DecEnumT を決めておきます。

```
--------------------------------------------------------------------------
function Word2DecEnumT(ADRS:WORD) return DecEnumT is
--------------------------------------------------------------------------
variable INDEX : INTEGER;
variable temp: DecEnumT;
Begin
        INDEX := CONV_INTEGER( TO_StdLogicVector('0' & ADRS) );
        case INDEX is
                when  aIOSEL0  =>                temp :=IOSEL0;
                when  aIOSEL1  =>                temp :=IOSEL1;
                when  aCSEL0   to  aCSEL1-1 =>   temp :=CSEL0;
                when  aCSEL1   to  aCSEL2-1 =>   temp :=CSEL1;
                when  aCSEL2   to  aCSEL3-1 =>   temp :=CSEL2;
                when  aCSEL3   to  aCSEL4-1 =>   temp :=CSEL3;
                when  aMSEL0   to  aMSEL1-1 =>   temp :=MSEL0;
                when  aMSEL1   to  aMSEL2-1 =>   temp :=MSEL1;
                when  aMSEL2   to  aMSEL3-1 =>   temp :=MSEL2;
                when  aMSEL3   to  aMSEL4-1 =>   temp :=MSEL3;
                when  aExtRst      =>            temp :=EXTRST;
                when  OTHERS   =>                temp :=NotUsed;
        end case;
        return temp;
end Word2DecEnumT;
```

上で、aIOSEL0 などのようにその値を決定する定数は、別途コンスタント定義として、パッケージ宣言部にまとめておくと良いかもしれません。（ただし、ここでインテジャーを下のようにして使っているのは悪い考えかも知れません (^^)。

```
constant    aIOSEL0   :   INTEGER  := 16#0002#;
constant    aIOSEL1   :   INTEGER  := 16#0004#;
constant    aCSEL0    :   INTEGER  := 16#0010#;
constant    aCSEL1    :   INTEGER  := 16#0018#;
constant    aCSEL2    :   INTEGER  := 16#0020#;
constant    aCSEL3    :   INTEGER  := 16#0028#;
constant    aCSEL4    :   INTEGER  := 16#0030#;
constant    aMSEL0    :   INTEGER  := 16#4000#;
constant    aMSEL1    :   INTEGER  := 16#5000#;
constant    aMSEL2    :   INTEGER  := 16#6000#;
constant    aMSEL3    :   INTEGER  := 16#7000#;
constant    aMSEL4    :   INTEGER  := 16#8000#;
constant    aExtRst   :   INTEGER  := 16#ffff#;
```

U0 ユニットは、外部からのアドレス信号を受けて上の仕様表に従ったアドレスのデコードを行い、このように定義された enum 信号に変換して U1、U2、のユニットに対して、あたかも１本の信号線であるかのような書き方で、全ての情報を送信しています。これはコンフィギュレーションを記述するときに相当すっきり

とした記述ができるようになったことを意味しています。これは実際の開発ではかなり大きな利点になるものと思います。

外部から受け取ったアドレス論理信号をこのタイプ宣言された信号載せる記述は以下のとおりです。

```
-- ADRBUF ----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use work.MyDecPack.all;
--------------------------------------------------------------------------
ENTITY    ADRBUF is
PORT (    ALE       : in   BIT;
          ADRIN     : in   WORD;
          IOAsign   : out  DecEnumT);
signal ADRLATCH : WORD;
end ADRBUF;
--------------------------------------------------------------------------
ARCHITECTURE a of ADRBUF is
Begin
      Process (ALE,ADRIN)
          Begin
              if(ALE='1')   then     ADRLATCH <= ADRIN;
              else                   ADRLATCH <= ADRLATCH;
              end if;
          IOAsign <= Word2DecEnumT(ADRLATCH);
      end Process;
end a;
```

つまり、ここで使っている IOAsign という信号は、構造記述の面から見るとあくまで 1 本の信号なのですが、これを別の見方からすると、**サイズを指定しないバス信号を接続している事とほぼ等価な記述方法**になっている事に気がつきます。

つまり、後からこの DecEnumT についてそのメンバーに追加・変更を行った場合にも、これらの内部配線記述自体には手を加えずに、関数 Word2DecEnumT とその参照場所だけを修正すれば良いことになります。

受け取る側の処理は非常に簡単になりますし、これはパッケージにまとめてある定義と関数・プロシージャですから、このパッケージを使っている全てのモジュールに変更内容が確実に伝達されます。

念のため、次ページに受け取る側での参照方法の例を幾つか示します。

受信するユニット側での参照例１

```
-- LEDOUT ----------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use work.MyDecPack.all;
-------------------------------------------------------------------------
ENTITY LEDOUT is
PORT ( IOAsign    : in   DecEnumT;
       LEDTEST    : in   BIT;
       LEDArray   : out  Bit_Vector(11 downto 0));
signal LED    : Bit_Vector(11 downto 0);
end LEDOUT;
-------------------------------------------------------------------------
ARCHITECTURE a of LEDOUT is
Begin
  DecEnumT2LED( IOAsign, LED );
    orgate:  for I in 0 to 11  generate
          LEDArray(I) <= LED(I) or LEDTEST;
    end generate orgate;
end a;
-------------------------------------------------------------------------
```

受信するユニット側での参照例２

```
-- CHKUNIT2 --------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use work.MyDecPack.all;
-------------------------------------------------------------------------
ENTITY CHKUNIT is
PORT (    IOAsign    : in   DecEnumT;
          DBUSIN     : in   BYTE;
          DOUT       : out  BYTE;
          CHK0       : out  BIT;
          CHK1       : out  BIT      );
end CHKUNIT;
-------------------------------------------------------------------------
ARCHITECTURE a of CHKUNIT is
Begin
    Process(IOAsign,DBUSIN)
      variable temp: BYTE;
    Begin
          DOUT <= temp;
          if(IOAsign=IOSEL0)   then temp := DBUSIN;
          else                      temp := temp;
          end if;
    end Process;
CHK0 <='1'  when  IOAsign=IOSEL1  else '0';
CHK1 <='1'  when  IOASign=MSEL1   else '0';
end a;
-------------------------------------------------------------------------
```

ここで、DecEnumT2LED として使用されているプロシージャは以下のとおり同じパッケージ・ボディ部分にて定義してありますが、この程度の内容であれば別にわざわざプロシージャにしなくても済みます。

```
-------------------------------------------------------------------------------
PROCEDURE DecEnumT2LED(DecEnumIn : in     DecEnumT;
                       signal       LED  : out  Bit_Vector(11 downto 0) ) is
-------------------------------------------------------------------------------
Begin
case DecEnumIn is
      when IOSEL0     => LED <= B"0000_0000_0001";
      when IOSEL1     => LED <= B"0000_0000_0010";
      when CSEL0      => LED <= B"0000_0000_0100";
      when CSEL1      => LED <= B"0000_0000_1000";
      when CSEL2      => LED <= B"0000_0001_0000";
      when CSEL3      => LED <= B"0000_0010_0000";
      when MSEL0      => LED <= B"0000_0100_0000";
      when MSEL1      => LED <= B"0000_1000_0000";
      when MSEL2      => LED <= B"0001_0000_0000";
      when MSEL3      => LED <= B"0010_0000_0000";
      when EXTRST     => LED <= B"0100_0000_0000";
      when OTHERS     => LED <= B"1000_0000_0000";
   end case;
end DecEnumT2LED;
end MyDecPack;
-- -------------------------------------------------------------------------------
```

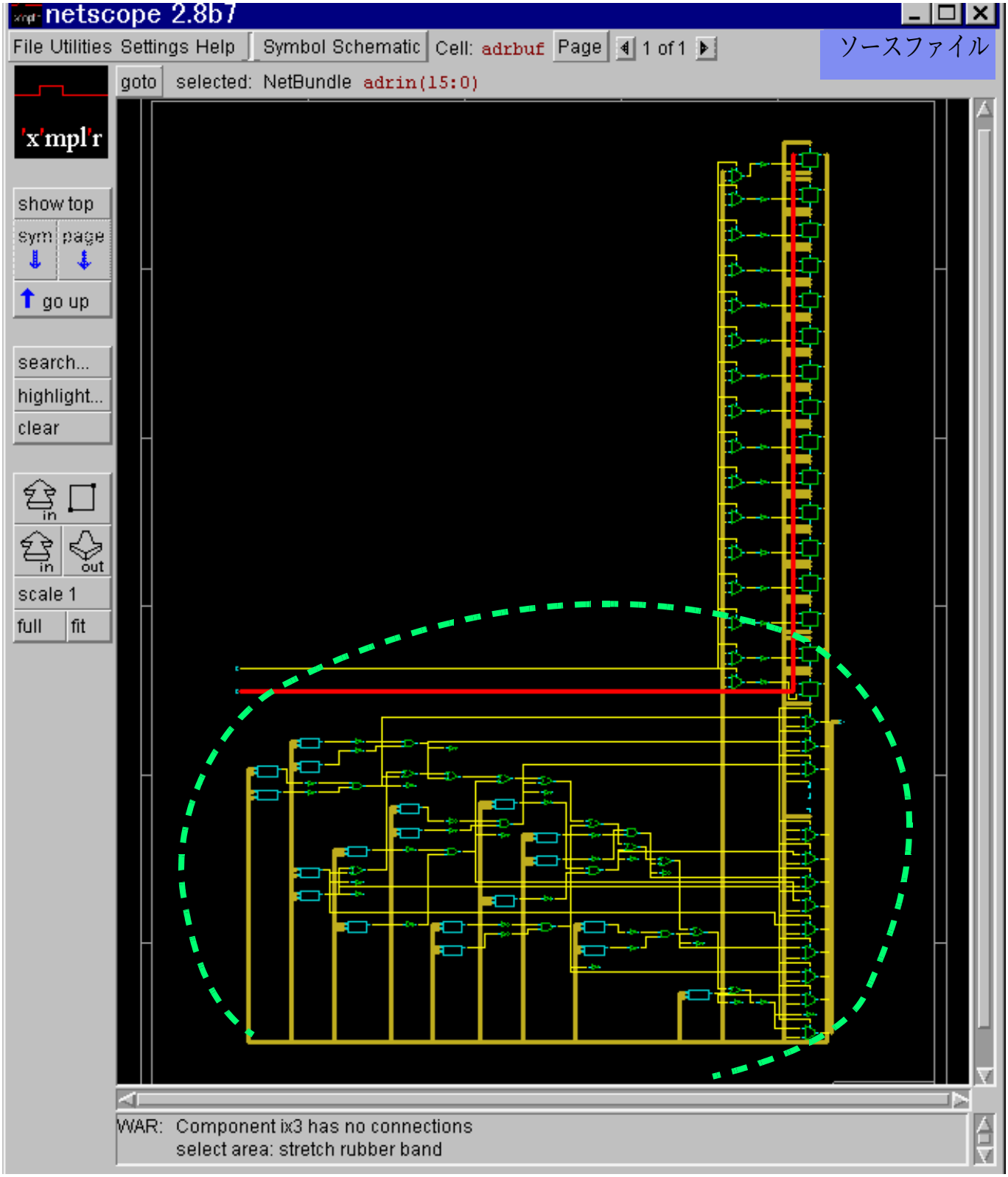
この例について次ページ以下にレオナルドでの読込み結果を添付しますのでよろしければ参照してみてください。

U0 ユニット：アドレス入力ユニットの内部です。

赤で示すのは、ADRIN 信号がラッチに接続されている部分を示しています。

緑の破線で囲んだ部分で示すのは、実際のデコーダ記述が構文解釈され、最適化前の状態になっているところです。

最適化後

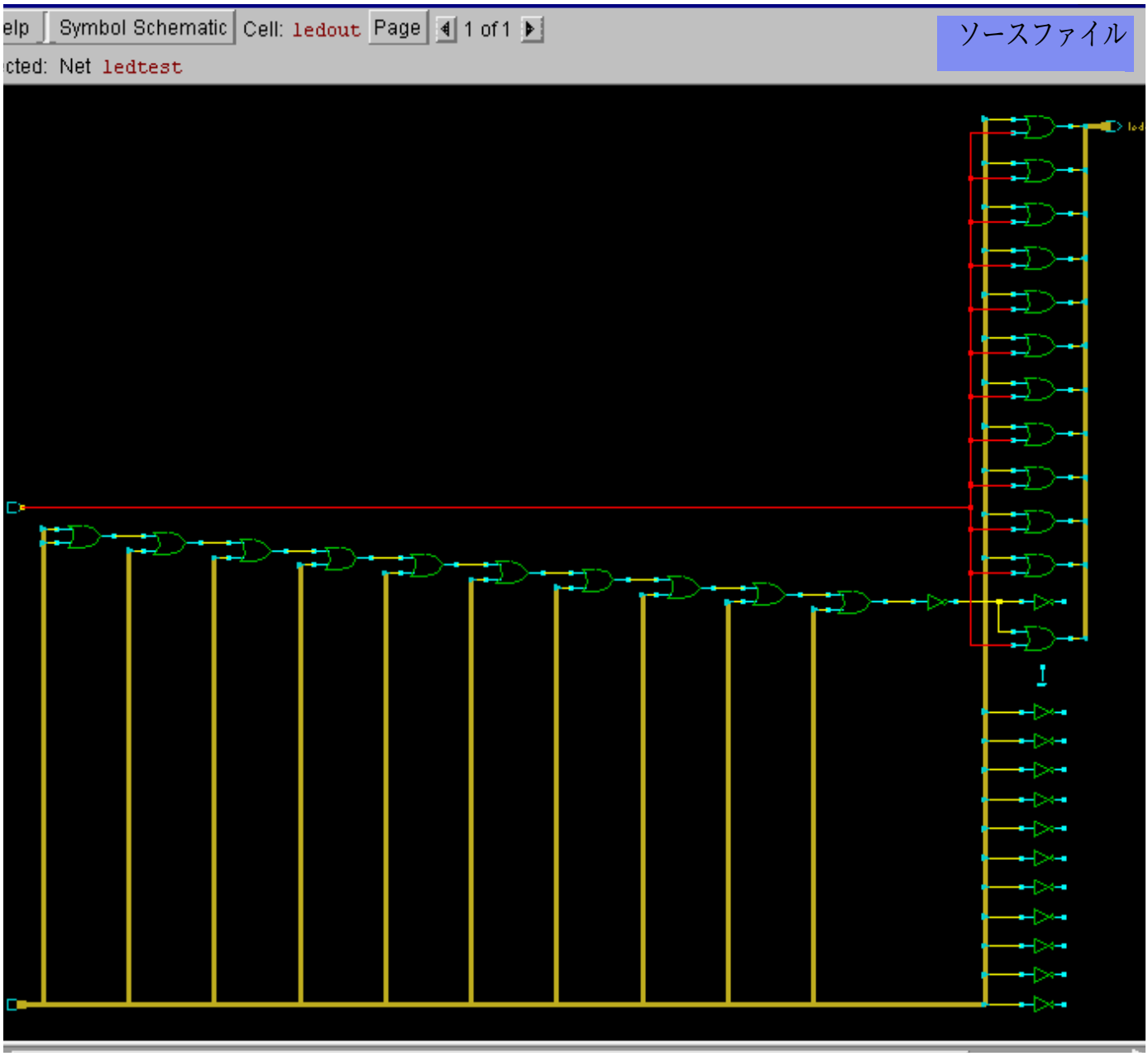


U1 ユニット： 受信側の回路は以下のように解釈されています。

赤で示したのは ＬＥＤ テスト入力信号の接続です

ＯＲ ゲートが延々と接続され、OTHERS 文節による選択がこのように実現されていることが判ります。

最適化後

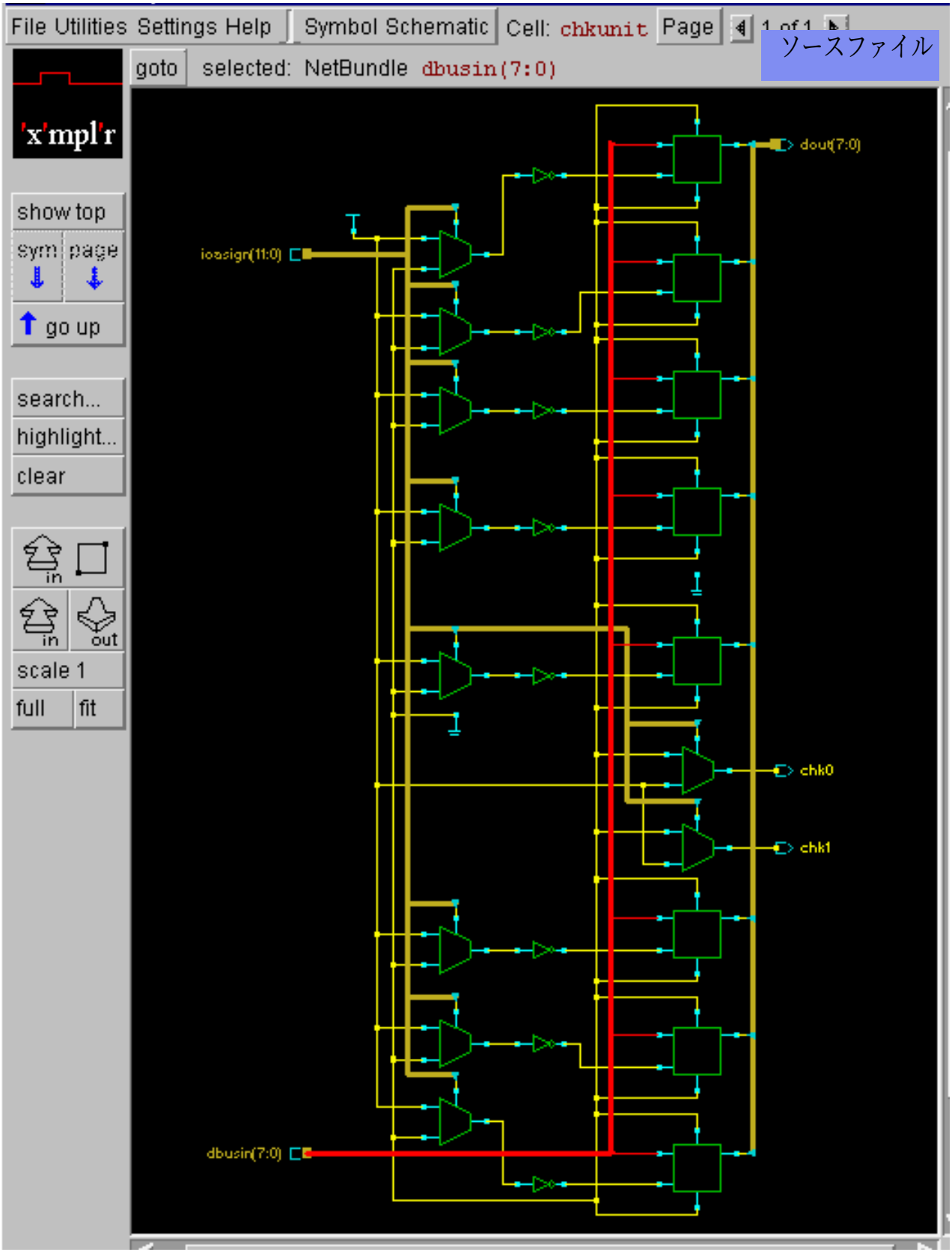


回路図面でリダンダントな部分は、最適化過程ですべて削除されます。

U2 ユニット ：同じく受信側の回路の解釈結果です

赤で示すのはデータバス入力経路です。

最適化後



ソースファイル

試しにアルテラの７ K シリーズに対して速度での最適化を行って読込んだ時との変化をみてみましょう

U0 ユニットは以下のようになりました。

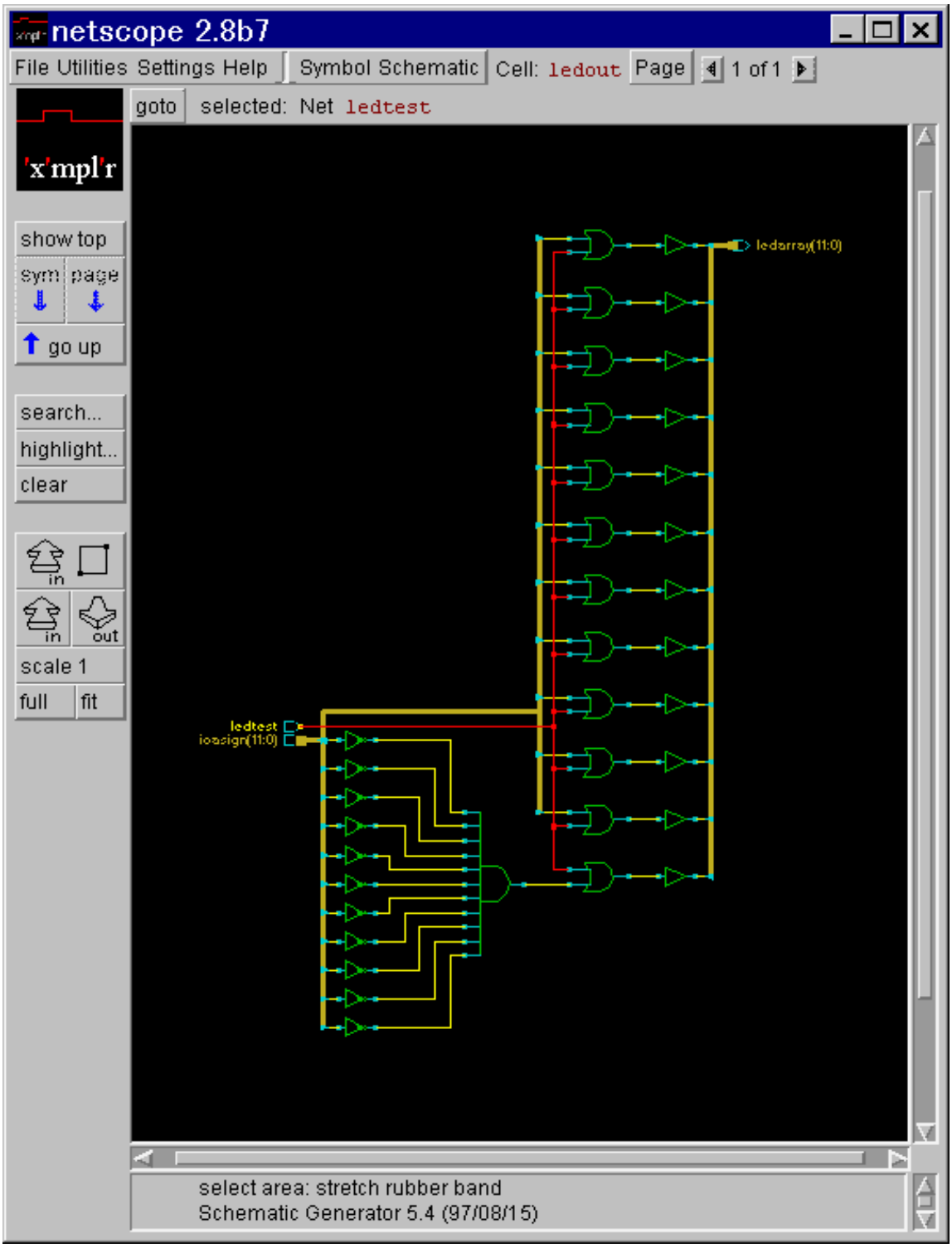
赤で示したのは最初の例と同じく、ADRIN 信号がラッチに接続されている部分を示しています。デコーダの展開状態が観察できます。

最適化前

U1 ユニット： 受信側の回路は以下のように最適化されています。

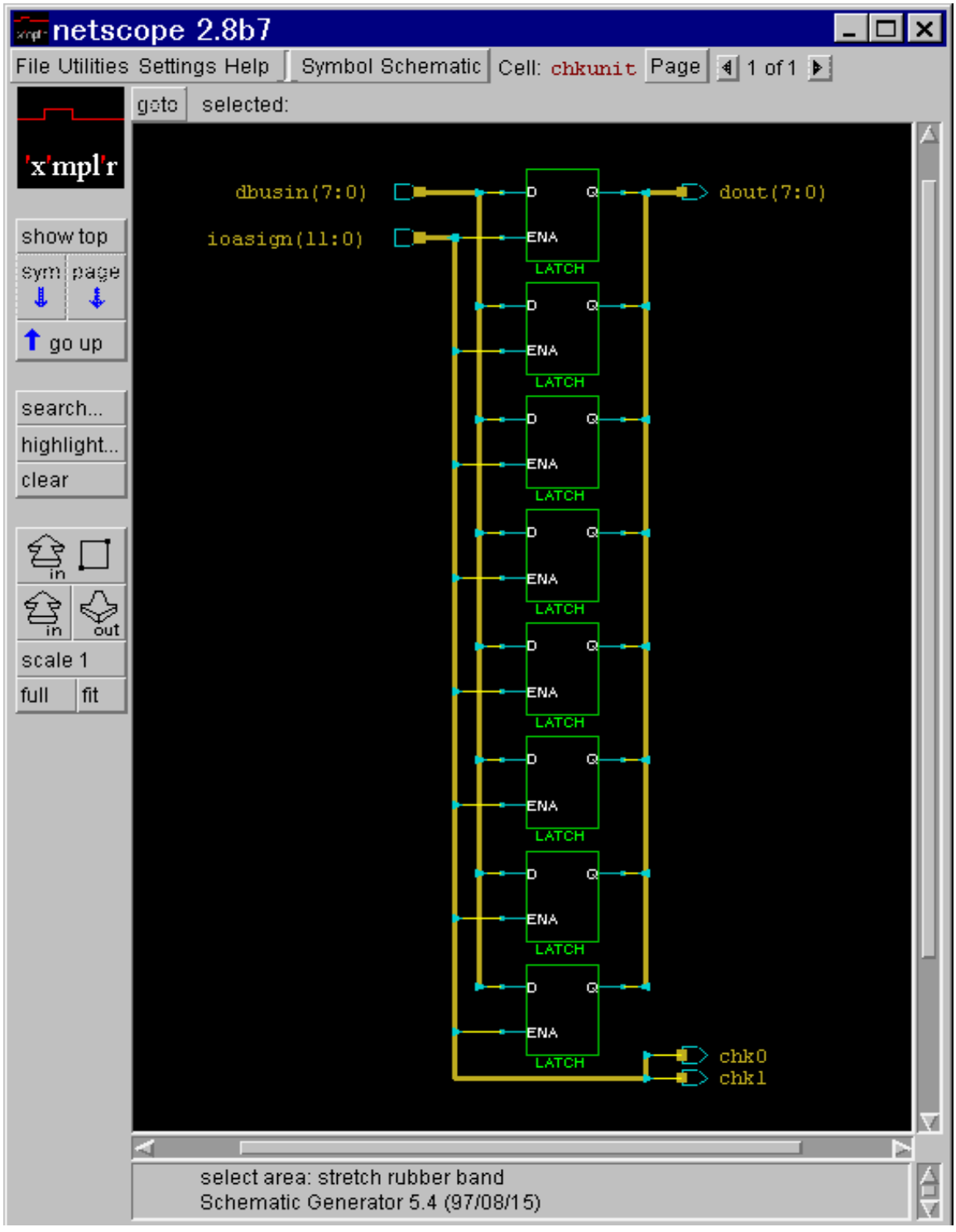
赤で示したのは前と同じく ＬＥＤ テスト入力信号の接続です
OTHERS 文節による選択が ＯＲ ゲートの延々とした接続として解釈されていましたが、最適化後はこのように実現されてます。

最適化前



U2 ユニット ：は、無駄な回路が削除され、ごく単純な回路に整理されています。
赤で示すのはデータバス入力経路です。

最適化前



まとめ

と言っても、今更のことばかりなので特にあたらしい事はもうありません。
開発の担当者間でよく打ち合わせた上、パッケージを作成し、その内容と変更情報を常に綿密に更新しながらテーマを追いかけてゆけば、協調開発環境としてのＨＤＬ手法は回路図での管理よりかなり楽になる部分がある筈です。

ran MyDec.vhd (package)

```
1       -- PACKAGE MyDecPack DEF ------------------------------------------------------
        library ieee;
        use ieee.std_logic_1164.all;
        -- ------------------------------------------------------------------------------
5       package MyDecPack is
        -- ------------------------------------------------------------------------------

        subtype DecLEDOut is  Bit_Vector(11 downto 0);
        subtype WORD is  BIT_VECTOR(15 downto 0);
10      subtype BYTE is  BIT_VECTOR( 7 downto 0);
        subtype NBBL is  BIT_VECTOR( 3 downto 0);

        Type DecEnumT is (  IOSEL0,IOSEL1, CSEL0, CSEL1, CSEL2, CSEL3,
                            MSEL0, MSEL1,  MSEL2, MSEL3,
15                          EXTRST,NotUsed      );

        constant aIOSEL0:   INTEGER := 16#0002#;
        constant aIOSEL1:   INTEGER := 16#0004#;
        constant aCSEL0 :   INTEGER := 16#0010#;
20      constant aCSEL1 :   INTEGER := 16#0018#;
        constant aCSEL2 :   INTEGER := 16#0020#;
        constant aCSEL3 :   INTEGER := 16#0028#;
        constant aCSEL4 :   INTEGER := 16#0030#;
        constant aMSEL0 :   INTEGER := 16#4000#;
25      constant aMSEL1 :   INTEGER := 16#5000#;
        constant aMSEL2 :   INTEGER := 16#6000#;
        constant aMSEL3 :   INTEGER := 16#7000#;
        constant aMSEL4 :   INTEGER := 16#8000#;
        constant aExtRst:   INTEGER := 16#ffff#;
30
        -- ------------------------------------------------------------------------------
        function Word2DecEnumT(ADRS:WORD) return DecEnumT;
        -- usage will be as follows ------------------------------------------------------
        --  Entity ADRUNIT  is
35      --  PORT(   P_ADRWD : WORD;
        --          ALE     : Bit;
        --          IntnlDecSignal: DecEnumT -- Now U don't have to describe details.);
        --  end ADRUNIT;
        --
40      --  Archtecture exmpl of ADRUNIT is
        --  variable ADRWD: WORD;
        --  Begin
        --      if(ALE='1') then    ADRWD := P_ADRWD;
        --      else                ADRWD := ADRWD;
45      --      end if;
        --      IntnlDecSignal <= BitVec2DecEnum(ADRWORD);
        --  end exmpl;
        -- ==============================================================================--
        --      BdSel      <= '0' when IntnlDecSignal = NotUsed else '1';
50      --      MemSelOut1 <= '1' when IntnlDecSignal = IOSEL1  else '0';
        -- ------------------------------------------------------------------------------
        PROCEDURE DecEnumT2LED(     DecEnumIn : in  DecEnumT;
                             signal LED       : out Bit_Vector(11 downto 0) );
        -- ------------------------------------------------------------------------------

55      end MyDecPack;
        ------------------------------------------------------------------------------



60



65
```

ran MyDec.vhd (package)

70

```
-- PACKAGE BODY ------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all ;

-- ------------------------------------------------------------------------
package Body MyDecPack is
-- ------------------------------------------------------------------------

------------------------------------------------------------------------------
function Word2DecEnumT(ADRS:WORD) return DecEnumT is
------------------------------------------------------------------------------
    variable INDEX : INTEGER;
    variable temp: DecEnumT;
Begin
    INDEX := CONV_INTEGER( TO_StdLogicVector('0' & ADRS) );
    case INDEX is
        when  aIOSEL0            => temp := IOSEL0;
        when  aIOSEL1            => temp := IOSEL1;
        when  aCSEL0 to aCSEL1-1 => temp := CSEL0;
        when  aCSEL1 to aCSEL2-1 => temp := CSEL1;
        when  aCSEL2 to aCSEL3-1 => temp := CSEL2;
        when  aCSEL3 to aCSEL4-1 => temp := CSEL3;
        when  aMSEL0 to aMSEL1-1 => temp := MSEL0;
        when  aMSEL1 to aMSEL2-1 => temp := MSEL1;
        when  aMSEL2 to aMSEL3-1 => temp := MSEL2;
        when  aMSEL3 to aMSEL4-1 => temp := MSEL3;
        when  aExtRst            => temp := EXTRST;
        when  OTHERS             => temp := NotUsed;
    end case;
    return temp;
end Word2DecEnumT;

------------------------------------------------------------------------------
PROCEDURE DecEnumT2LED(     DecEnumIn : in  DecEnumT;
                    signal  LED       : out Bit_Vector(11 downto 0) ) is
------------------------------------------------------------------------------
Begin
    case DecEnumIn is
        when IOSEL0 => LED <= B"0000_0000_0001";
        when IOSEL1 => LED <= B"0000_0000_0010";
        when CSEL0  => LED <= B"0000_0000_0100";
        when CSEL1  => LED <= B"0000_0000_1000";
        when CSEL2  => LED <= B"0000_0001_0000";
        when CSEL3  => LED <= B"0000_0010_0000";
        when MSEL0  => LED <= B"0000_0100_0000";
        when MSEL1  => LED <= B"0000_1000_0000";
        when MSEL2  => LED <= B"0001_0000_0000";
        when MSEL3  => LED <= B"0010_0000_0000";
        when EXTRST => LED <= B"0100_0000_0000";
        when OTHERS => LED <= B"1000_0000_0000";
     end case;
end DecEnumT2LED;
end MyDecPack;
-- ------------------------------------------------------------------------
```

ran decode.vhd

```
1       -- Example: ran ------------------------------------------------------------
        --
        -- This example is to demonstrate a case of enum signal usage.
        -- WATCH Entity descriptions are much simplified when using some enum signals.
5       --
        ------------------------------------------------------------------------------

        -- ADRBUF --------------------------------------------------------------------
        library ieee;
10      use ieee.std_logic_1164.all;
        use work.MyDecPack.all;
        ------------------------------------------------------------------------------
        ENTITY ADRBUF is
        PORT (  ALE     : in    BIT;
15              ADRIN   : in    WORD;
                IOAsign : out   DecEnumT    );
        signal ADRLATCH : WORD;
        end ADRBUF;
        ------------------------------------------------------------------------------
20      ARCHITECTURE a of ADRBUF is
        Begin
            Process (ALE,ADRIN)
            Begin
                if(ALE='1') then    ADRLATCH <= ADRIN;
25              else                ADRLATCH <= ADRLATCH;
                end if;
            IOAsign <= Word2DecEnumT(ADRLATCH);
            end Process;
        end a;
30
        -- LEDOUT --------------------------------------------------------------------
        library ieee;
        use ieee.std_logic_1164.all;
        use work.MyDecPack.all;
35      ------------------------------------------------------------------------------
        ENTITY LEDOUT is
        PORT (  IOAsign : in    DecEnumT;
                LEDTEST : in    BIT;
                LEDArray: out   Bit_Vector(11 downto 0) );
40          signal LED: Bit_Vector(11 downto 0);

        end LEDOUT;
        ------------------------------------------------------------------------------
        ARCHITECTURE a of LEDOUT is
45      Begin
            DecEnumT2LED( IOAsign,LED );
            orgate:for I in 0 to 11  generate
                LEDArray(I) <= LED(I) or LEDTEST;
            end generate orgate;
50      end a;
        ------------------------------------------------------------------------------


        -- CHKUNIT2 ------------------------------------------------------------------
55      library ieee;
        use ieee.std_logic_1164.all;
        use work.MyDecPack.all;
        ------------------------------------------------------------------------------
        ENTITY CHKUNIT is
60      PORT (  IOAsign : in    DecEnumT;
                DBUSIN  : in    BYTE;
                DOUT    : out   BYTE;
                CHK0    : out   BIT;
                CHK1    : out   BIT         );
65      end CHKUNIT;
        ------------------------------------------------------------------------------
        ARCHITECTURE a of CHKUNIT is
        Begin
            Process(IOAsign,DBUSIN)
70              variable temp: BYTE;
```

ran decode.vhd

```
        Begin
            DOUT <= temp;
            if(IOAsign=IOSEL0) then     if(IOAsign=IOSEL0) then temp := DBUSIN;
            else                    temp := temp;
            end if;
        end Process;
        CHK0 <='1' when IOAsign=IOSEL1  else '0';
        CHK1 <='1' when IOASign=MSEL1   else '0';
    end a;
    ------------------------------------------------------------------------------


    -- TOPUNIT -----------------------------------------------------------------
    library ieee;
    use ieee.std_logic_1164.all;
    use work.MyDecPack.all;
    ------------------------------------------------------------------------------
    ENTITY TOPRAN is
    PORT (  P_ALE       : in    BIT ;
            P_ADRIN     : in    WORD;
            P_LEDTEST   : in    BIT ;
            P_LEDArry   : out   Bit_Vector(11 downto 0);
            P_DBUSIN    : in    BYTE;
            P_DOUT      : out   BYTE;
            P_LED0      : out   BIT ;
            P_LED1      : out   BIT      );
    end TOPRAN;

    ------------------------------------------------------------------------------
    ARCHITECTURE RTL of TOPRAN is
    ------------------------------------------------------------------------------
    component ADRBUF    ---------------------------------------------------------

    PORT (  ALE     : in    BIT;
            ADRIN   : in    WORD;
            IOAsign : out   DecEnumT        );
    end component;
    component LEDOUT    ---------------------------------------------------------

    PORT (  IOAsign : in    DecEnumT;
            LEDTEST : in    BIT;
            LEDArray: out   Bit_Vector(11 downto 0) );
    end component;
    component CHKUNIT   ---------------------------------------------------------
    PORT (  IOAsign : in    DecEnumT;
            DBUSIN  : in    BYTE;
            DOUT    : out   BYTE;
            CHK0    : out   BIT;
            CHK1    : out   BIT        );
    end component;

    signal  IntnlEnumSig :DecEnumT;
    ------------------------------------------------------------------------------

    Begin
    ------------------------------------------------------------------------------
        U0: ADRBUF  PORT MAP(   ALE    => P_ALE,
                                ADRIN  => P_ADRIN,
                                IOAsign=> IntnlEnumSig  );

        U1: LEDOUT  PORT MAP(   IOAsign => IntnlEnumSig,
                                LEDTEST => P_LEDTEST,
                                LEDArray=> P_LEDArry    );

        U2: CHKUNIT PORT MAP(   IOAsign => IntnlEnumSig,
                                DBUSIN  => P_DBUSIN,
                                DOUT    => P_DOUT,
                                CHK0    => P_LED0,
                                CHK1    => P_LED1       );
```

ran decode.vhd

```
end RTL;
-------------------------------------------------------------------------------
```