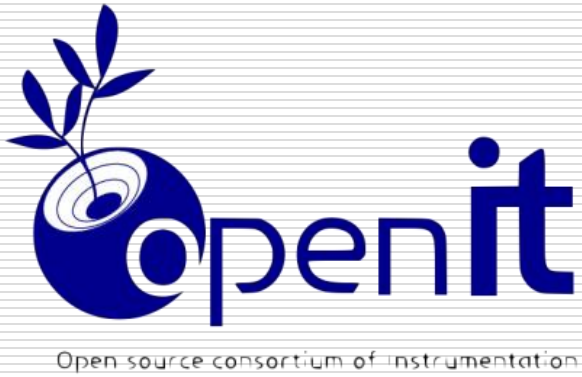
Open-It FPGAトレーニングコース（入門編）

6. 階層構造設計

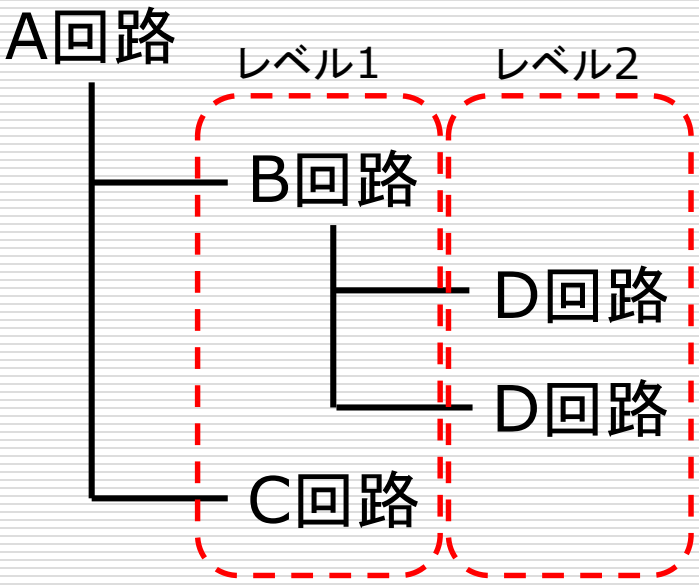


第3.4版

2016年06月22日

階層構造

- ある機能のまとまり毎に分ける
- さらに機能毎に分ける…



例えば、A回路の機能は大きく2つ（B回路とC回路）に分けることができる

B回路は2つのD回路から構成されている

機能単位で分かれているので理解しやすい

ブロック図で表現すれば、階層レベルごとに動作を理解できる

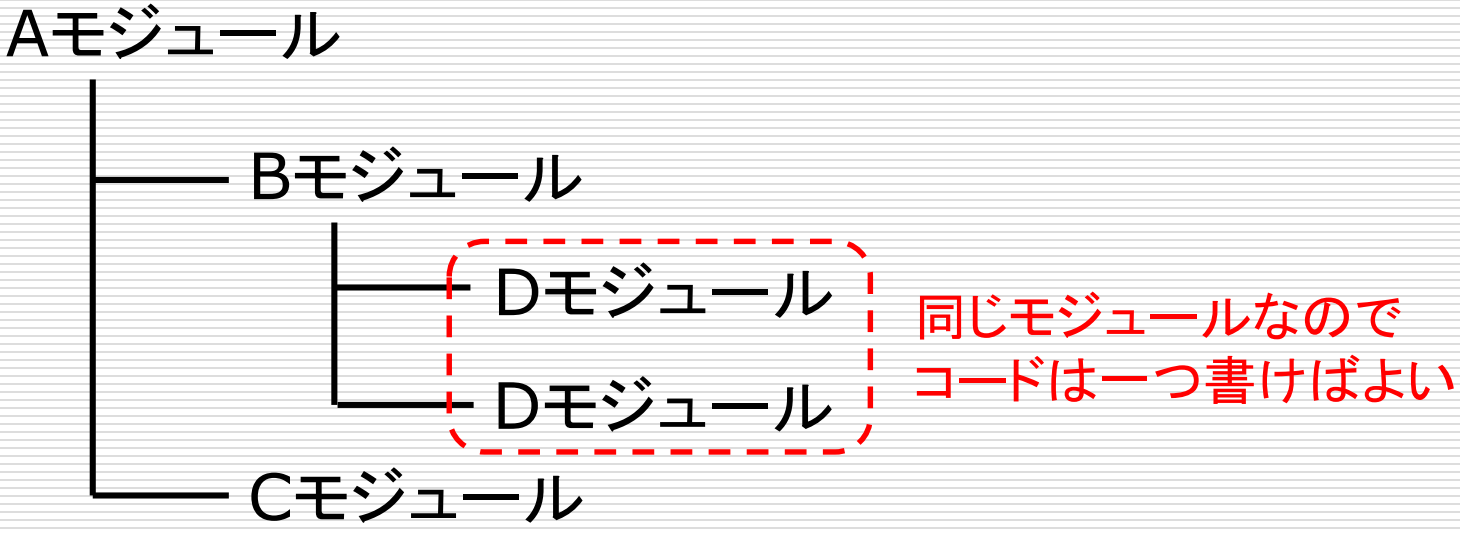
抽象度を上げるということ

抽象度

- 階層構造を採用すると上位層になる程、人に分かりやすい表現になる傾向がある
  - 細部を隠ぺいして、「抽象度を上げる」と言い換えることができる
- 例）ソフトウエアのプログラミング言語
  - CPUは2進数で動いているが、
  - 通常はプログラムを2進数で書くことは無い
  - 抽象度が高く人に分かりやすい言語で記述する
  - 階層構造を採用する事でハードウエアも同じように分かりやすくすることができる

Verilogコードでの階層表現

- モジュールを一つの回路ブロックと考える



以降で下位モジュールの記述方法を説明します。

下位モジュールの記述実習

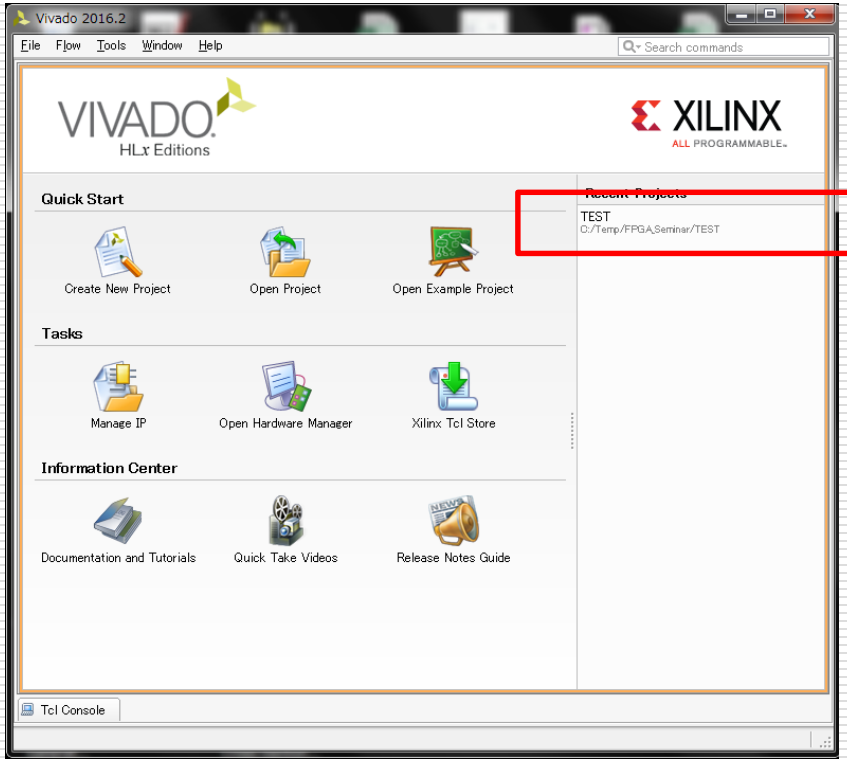
- TEST.vのカウンター部分を別のモジュールに分けて、TEST.vに組み込みます

この部分をTEST_SYNC_COUNTERモジュールとして分けます。

```
34    reg [31:0]  sync_counter;
35
36    always @(posedge OSC or negedge RST_SWn)begin
37        if(!RST_SWn)begin
38            sync_counter[31:0] <= 32'd0;
39        end else begin
40            sync_counter[31:0] <= sync_counter[31:0] + 32'd1;
41        end
42    end
43
44    assign LED15 = sync_counter[28];
```

Vivado起動

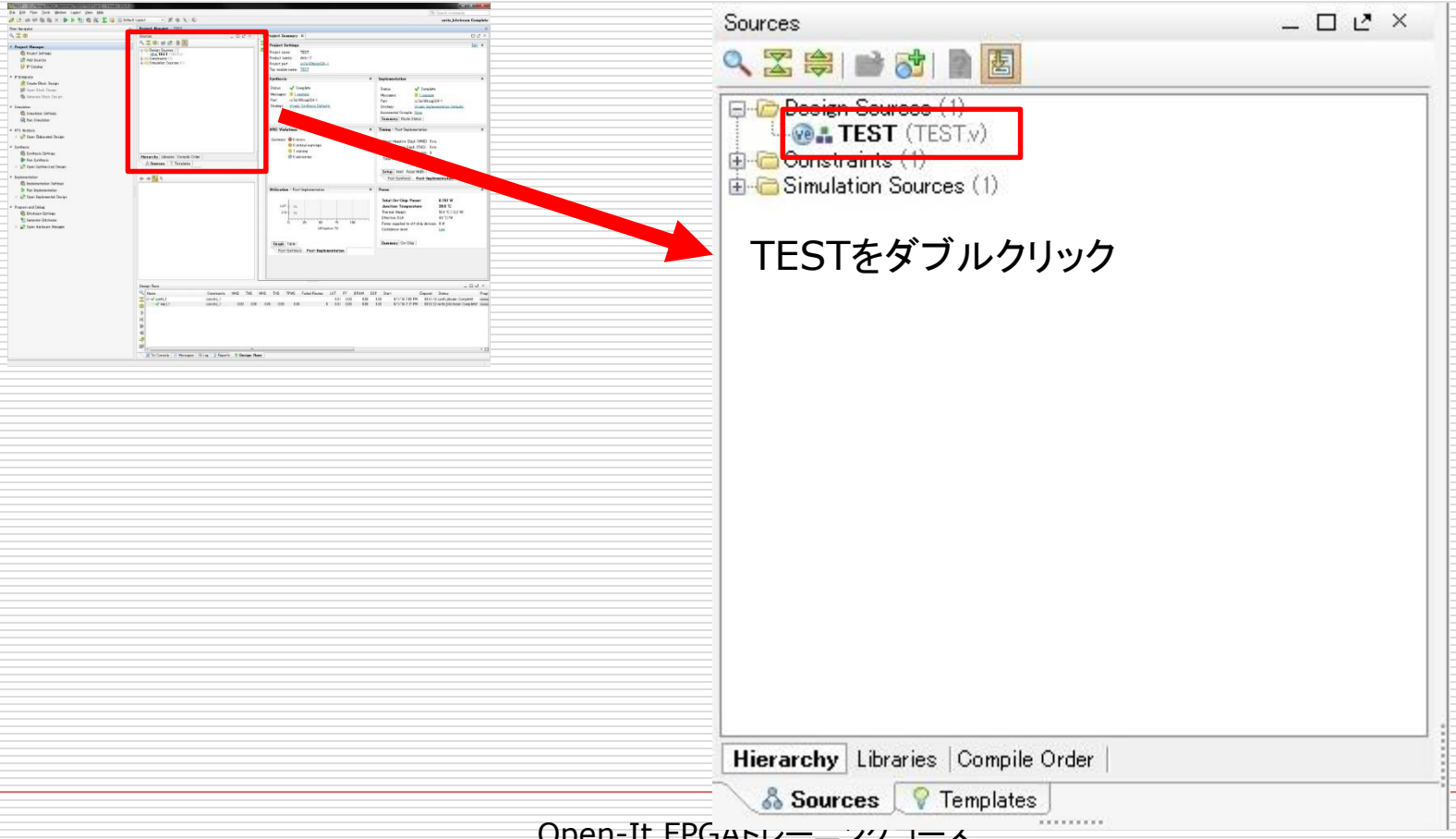
ソースコードを編集するためにVivadoを起動してください



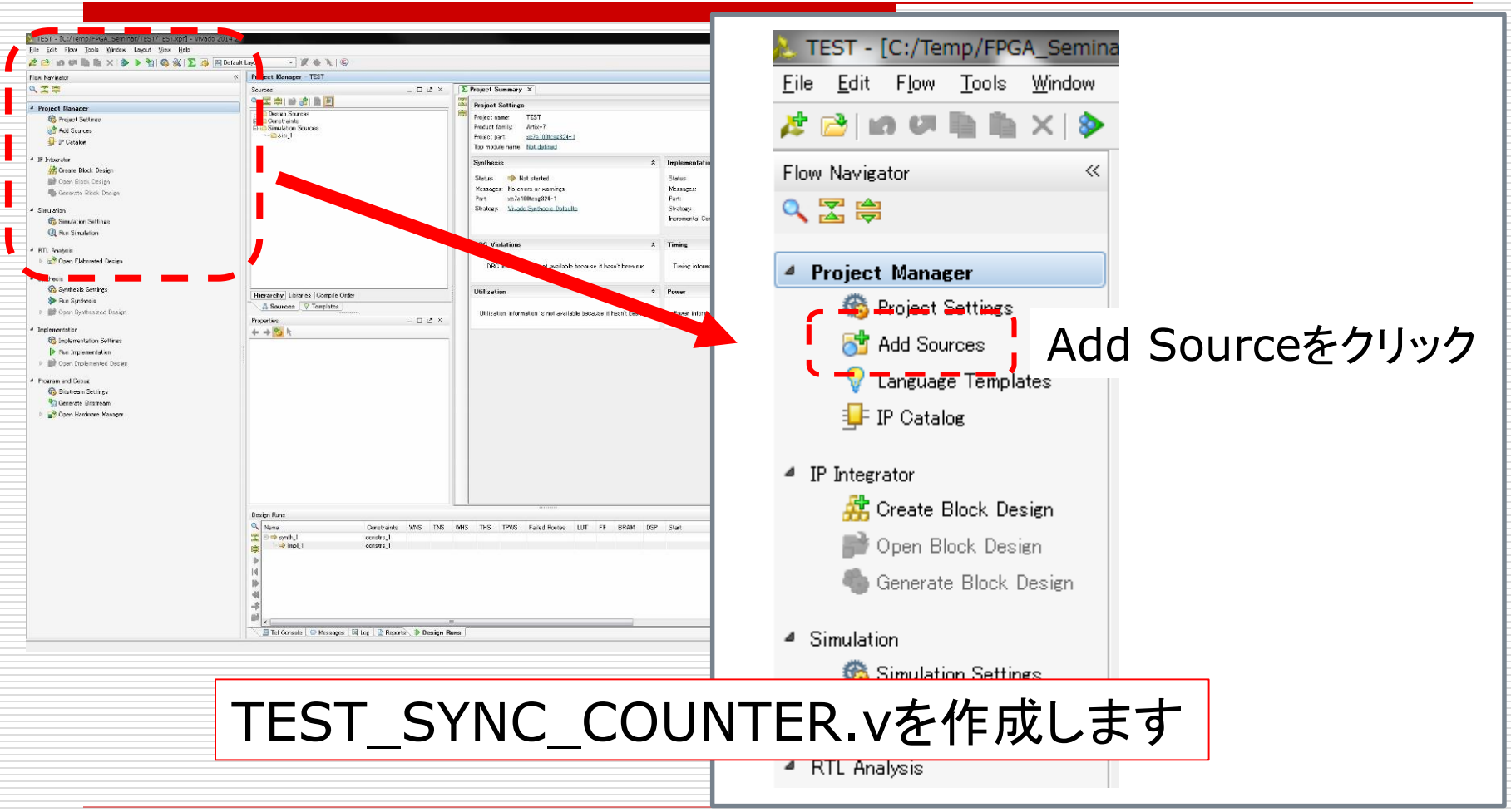
TESTをクリック

ファイルを開く

カウンター部分をコピーするためにTEST.vを開きます

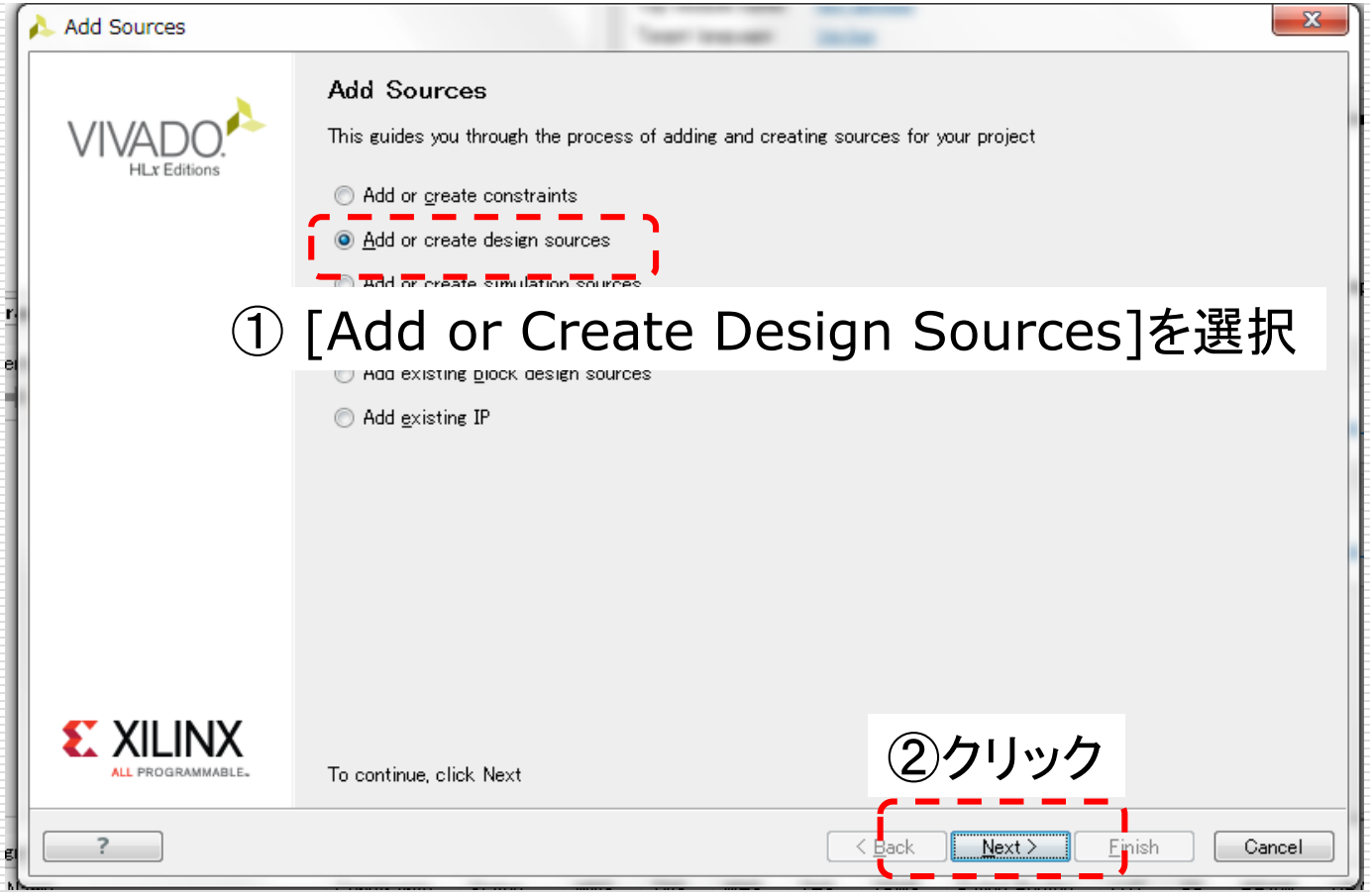


新規Verilogモジュール作成



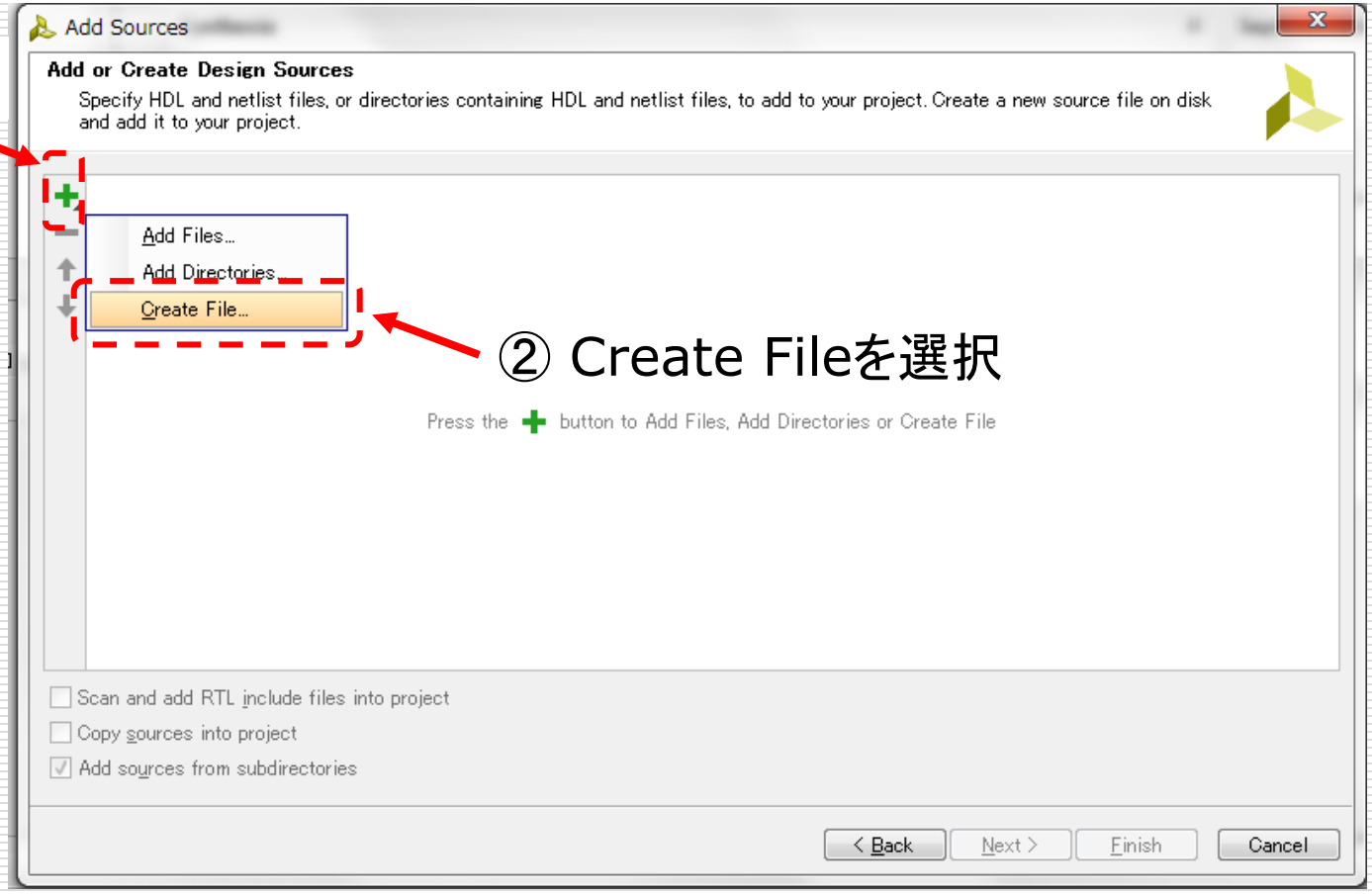
TEST_SYNC_COUNTER.vを作成します

ウィザード開始

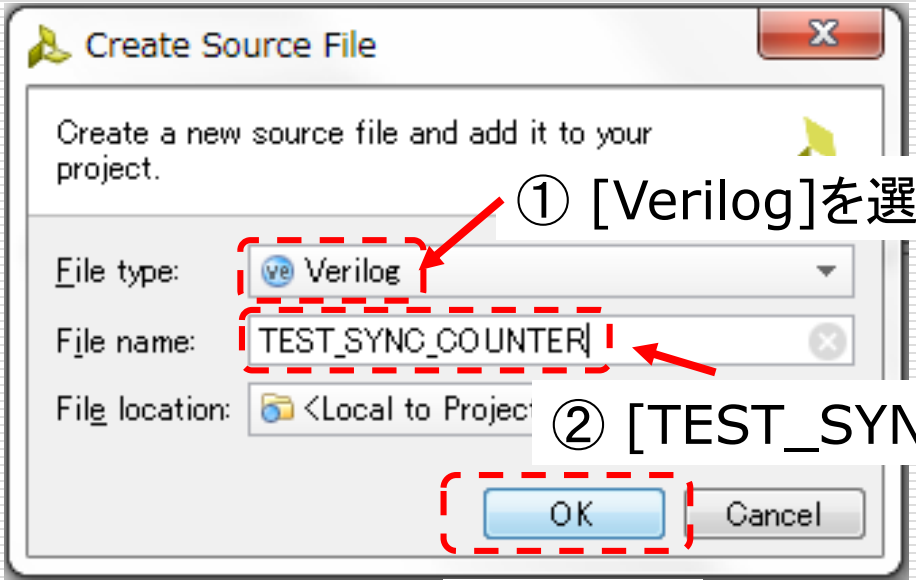

Add Sources
Add Sources
This guides you through the process of adding and creating sources for your project
VIVADO
HLx Editions
Add or create constraints
Add or create design sources
Add existing block design sources
Add existing IP
① [Add or Create Design Sources]を選択
XILINX
ALL PROGRAMMABLE
To continue, click Next
②クリック
?
< Back
Next >
Finish
Cancel

新規ファイル作成

①クリック



ファイル名の指定

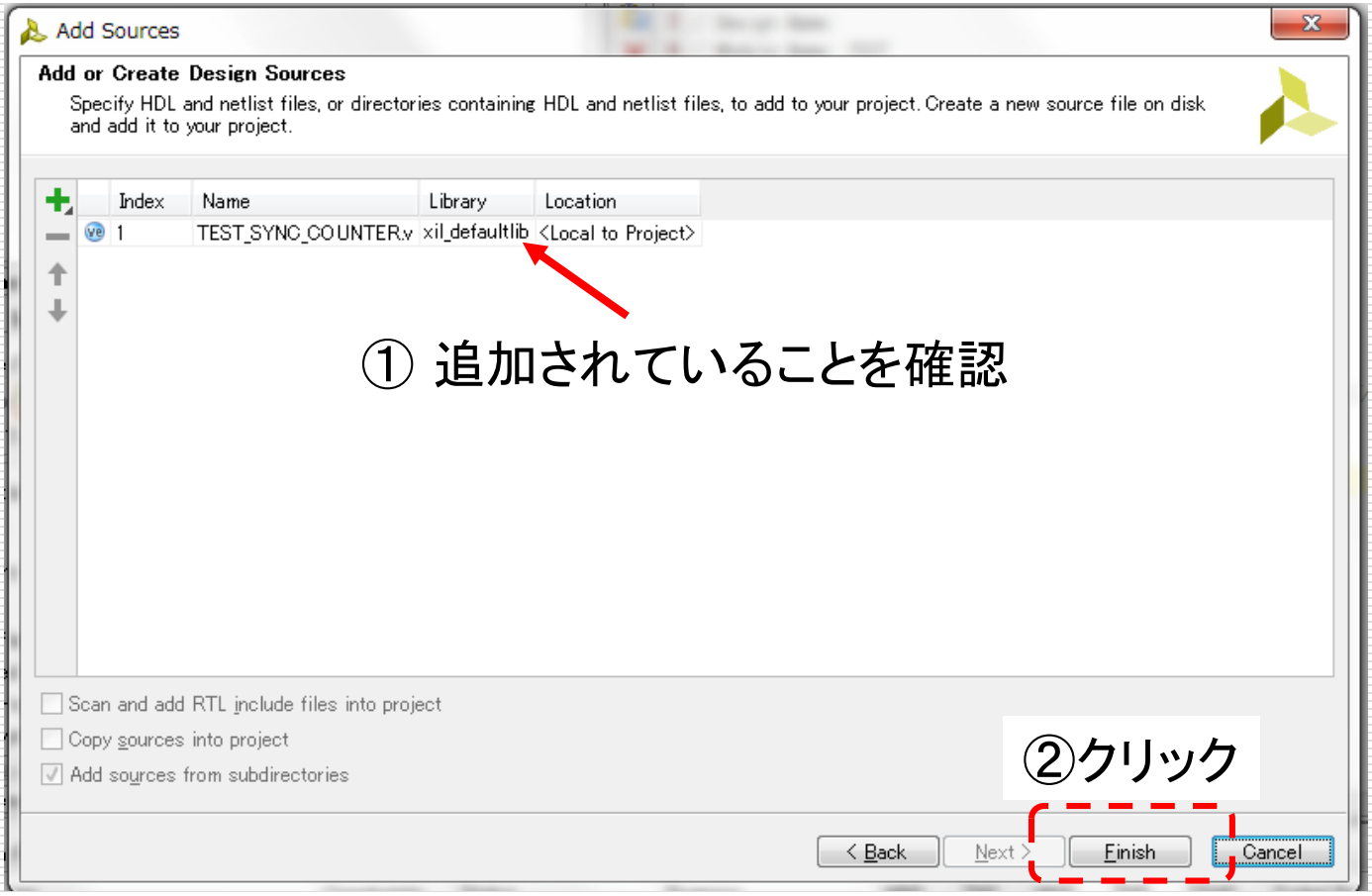


① [Verilog]を選択

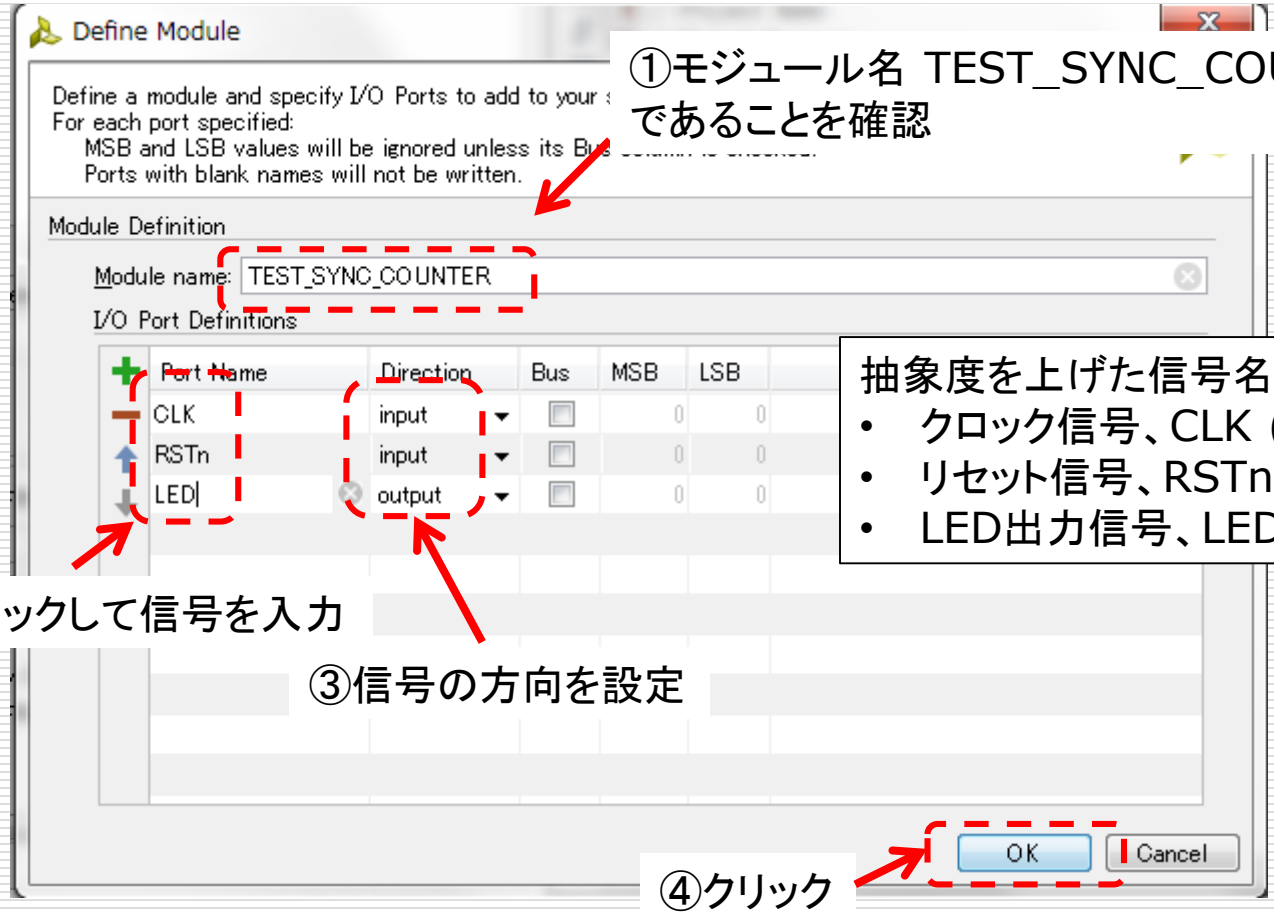
② [TEST_SYNC_COUNTER]と入力

③クリック

ファイル名などの確認

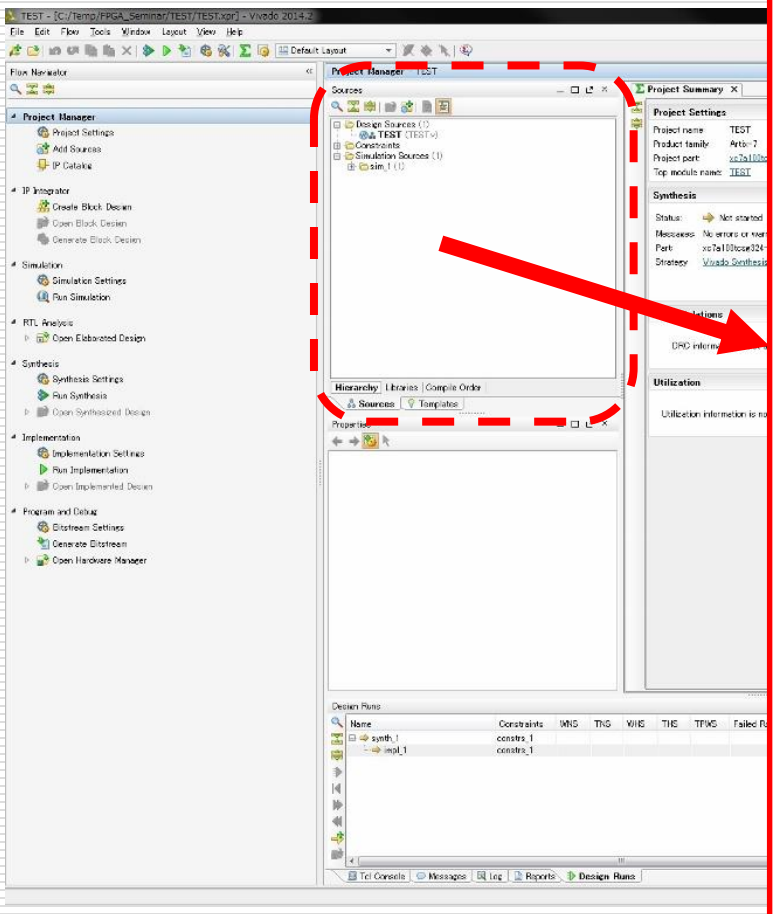

Add Sources
Add or Create Design Sources
Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project.
Index
Name
Library
Location
1
TEST_SYNC_COUNTER.v
xil_defaultlib
<Local to Project>
① 追加されていることを確認
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
②クリック
< Back
Next >
Finish
Cancel

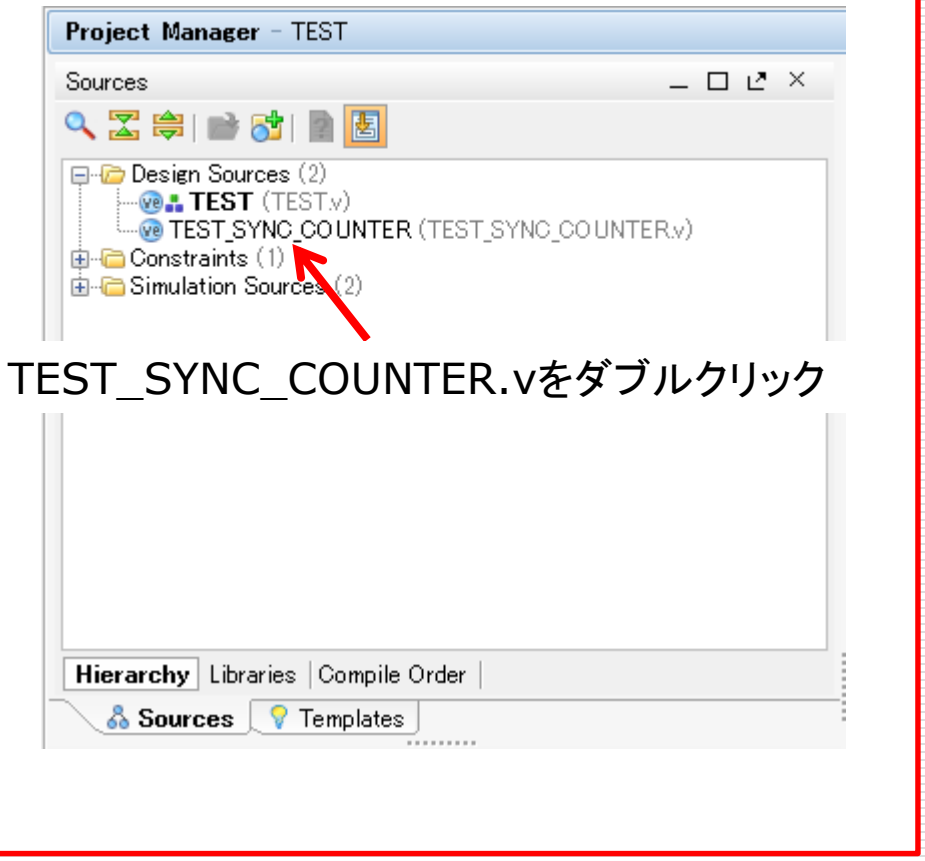
モジュール名とポート名の指定



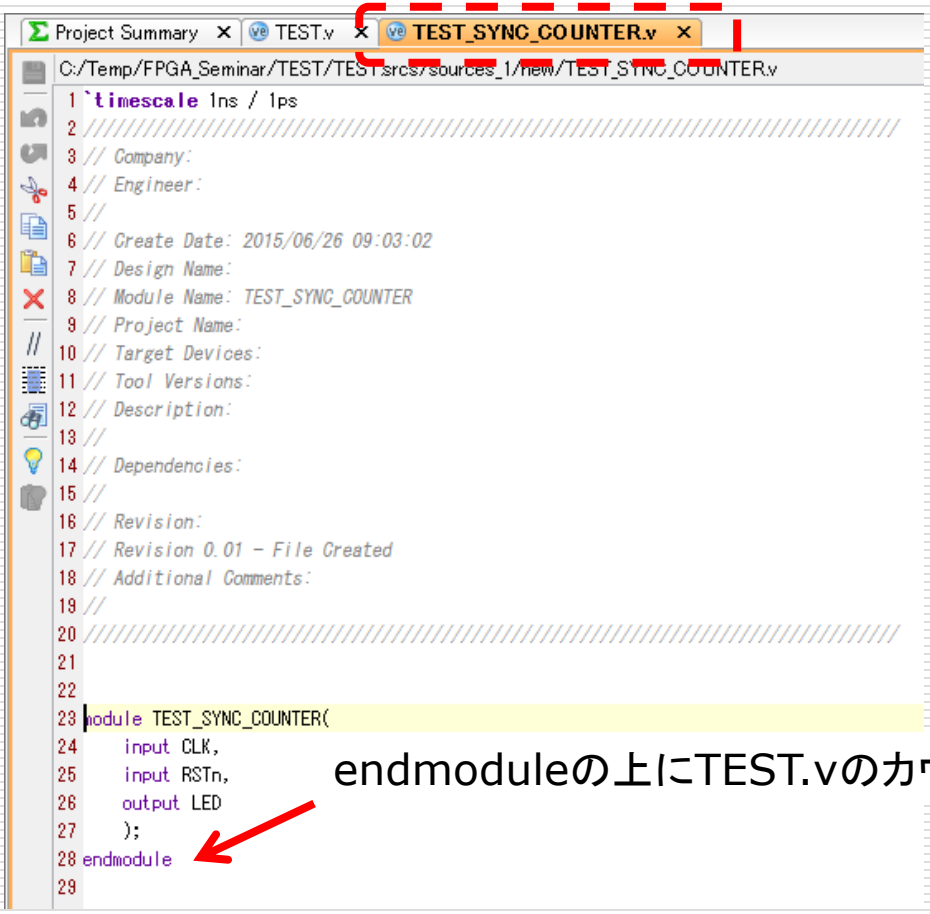
抽象度を上げた信号名を採用します
- クロック信号、CLK (IN)
- リセット信号、RSTn (IN)
- LED出力信号、LED (OUT)

モジュールがDesign Sourceに追加された事を確認



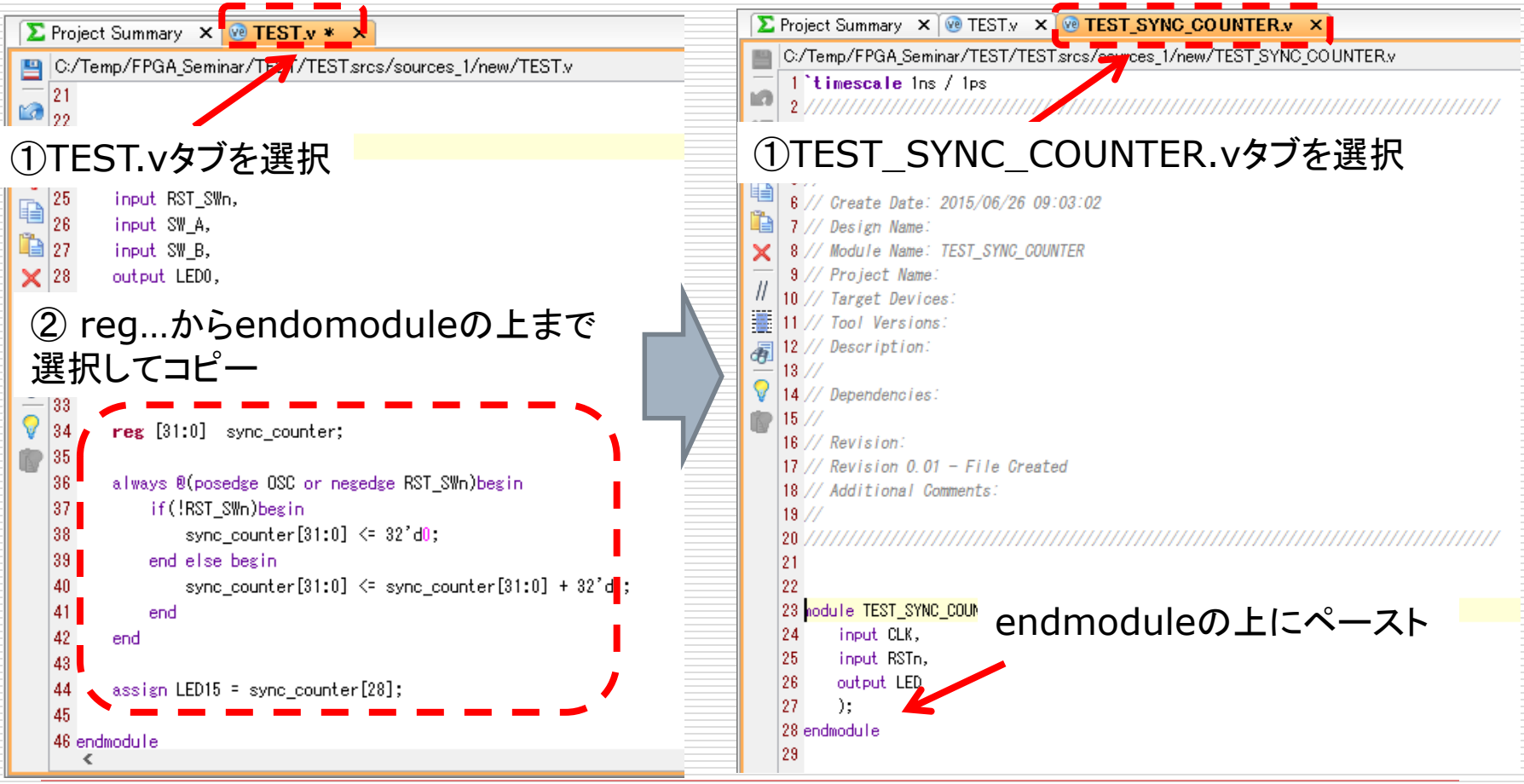


TEST_SYNC_COUNTER.vの内容表示



endmoduleの上にTEST.vのカウンタ記述をコピペする

カウンタ記述のコピー


Project Summary
TEST.v *
C:/Temp/FPGA_Seminar/TEST/TEST.srcs/sources_1/new/TEST.v
①TEST.vタブを選択
② reg…からendomoduleの上まで
選択してコピー
reg [31:0] sync_counter;
always @(posedge OSC or negedge RST_SWn)begin
if(!RST_SWn)begin
sync_counter[31:0] <= 32'd0;
end else begin
sync_counter[31:0] <= sync_counter[31:0] + 32'd1;
end
end
assign LED15 = sync_counter[28];
endmodule
Project Summary
TEST.v
TEST_SYNC_COUNTER.v
C:/Temp/FPGA_Seminar/TEST/TEST.srcs/sources_1/new/TEST_SYNC_COUNTER.v
①TEST_SYNC_COUNTER.vタブを選択
`timescale 1ns / 1ps
// Create Date: 2015/06/26 09:03:02
// Design Name:
// Module Name: TEST_SYNC_COUNTER
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
module TEST_SYNC_COUN
input CLK,
input RSTn,
output LED
);
endmodule
endmoduleの上にペースト

ペーストしたコードの修正
TEST_SYNC_COUNTER.v

TEST_SYNC_COUNTER.vを開いてください

```
23 module TEST_SYNC_COUNTER(
24     input CLK,
25     input RSTn,
26     output LED
27     );
28
29     reg [31:0]  sync_counter;
30
31     always @(posedge OSC or negedge RST_SWn)begin
32         if(!RST_SWn)begin
33             sync_counter[31:0] <= 32'd0;
34         end else begin
35             sync_counter[31:0] <= sync_counter[31:0] + 32'd1;
36         end
37     end
38
39     assign LED15 = sync_counter[28];
40
41 endmodule
```

入出力信号に合わせて修正します

- OSC → CLK
- RST_SWn →RSTn
- LED15→LED

修正したコード

```
23 module TEST_SYNC_COUNTER(
24     input CLK,
25     input RSTn,
26     output LED
27     );
28
29     reg [31:0]  sync_counter;
30
31     always @(posedge CLK or negedge RSTn)begin
32         if(!RSTn)begin
33             sync_counter[31:0] <= 32'd0;
34         end else begin
35             sync_counter[31:0] <= sync_counter[31:0] + 32'd1;
36         end
37     end
38
39     assign LED = sync_counter[28];
40
41 endmodule
```

修正が終わったら保存してください

上位モジュールでの下位モジュールの組み込み

- TEST.vを開いてください
- 先ほどコピーした部分を削除してください

```
23 module TEST(
24     input OSC,
25     input RST_SWn,
26     input SW_A,
27     input SW_B,
28     output LED0,
29     output LED15
30     );
31
32     assign LED0 = SW_A & SW_B;
33
34    この部分に記述されていたカウンターを削除した
35
36 endmodule
```

下位モジュールの組み込み方 1/2

```
23 module TEST(
24     input OSC,
25     input RST_SWn,
26     input SW_A,
27     input SW_B,
28     output LED0,
29     output LED15
30     );
31
32     assign LED0 = SW_A & SW_B;
33
34     TEST_SYNC_COUNTER U1(
35         .CLK    (OSC),
36         .RSTn   (RST_SWn),
37         .LED    (LED15)
38     );
39
40 endmodule
```

下位モジュールの組み込み

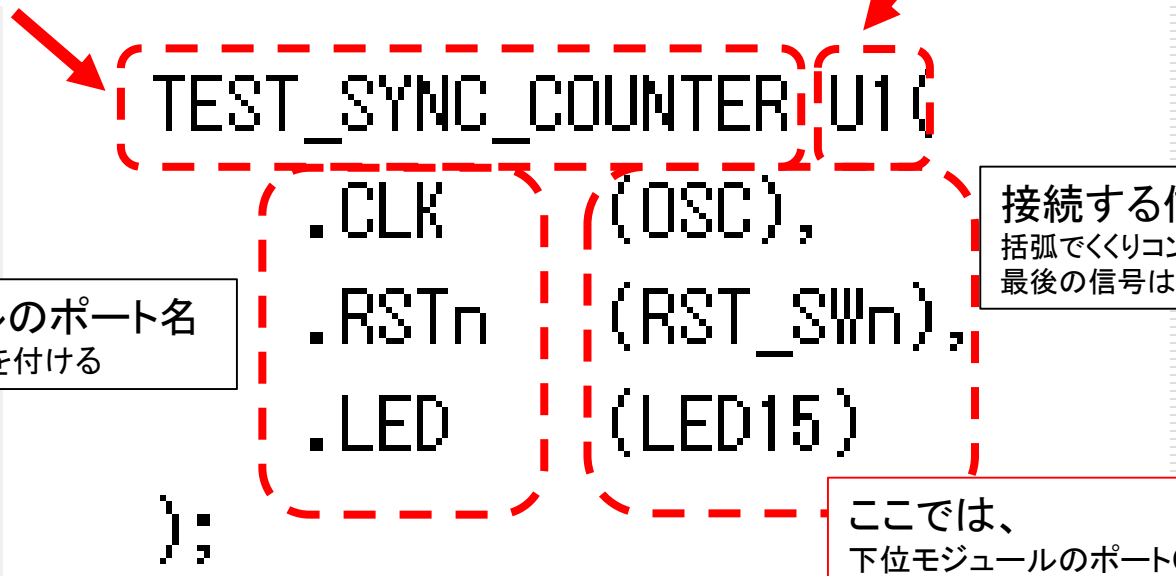
次ページで拡大

下位モジュールの組み込み方 2/2

拡大する為assign文の下からendomoduleの上までの部分を示します

インスタンス名：
設計者が自由な名前を付けることができる
複数の同一モジュールを組み込んだときに区別できるように

下位モジュールのモジュール名



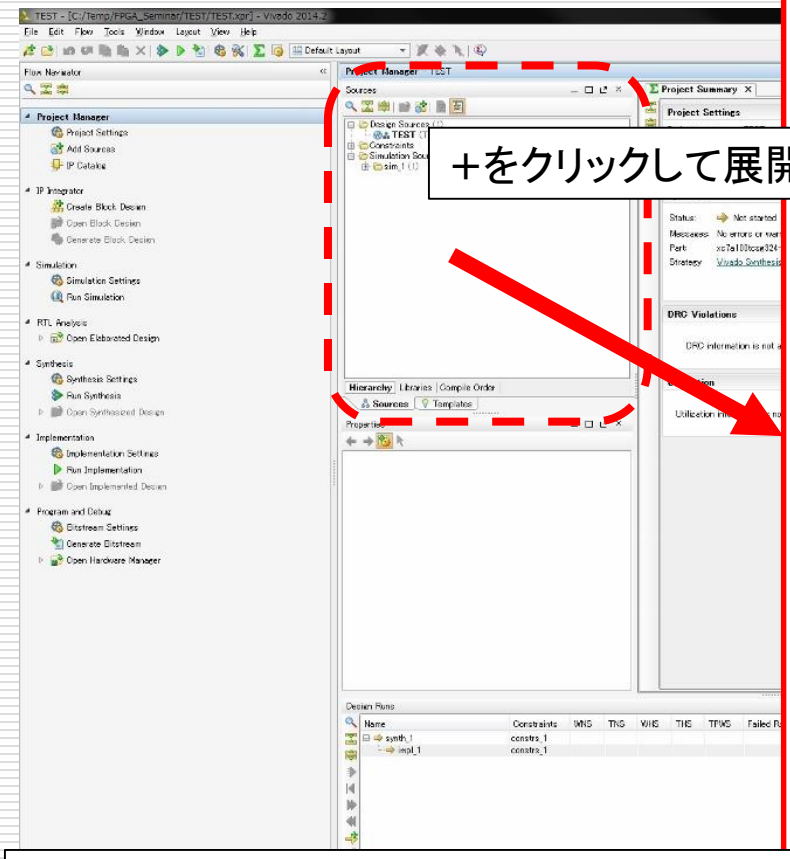
接続する信号名
括弧でくくりコンマを打つ、
最後の信号はコンマ不要なので注意

下位モジュールのポート名
信号名の前にドットを付ける

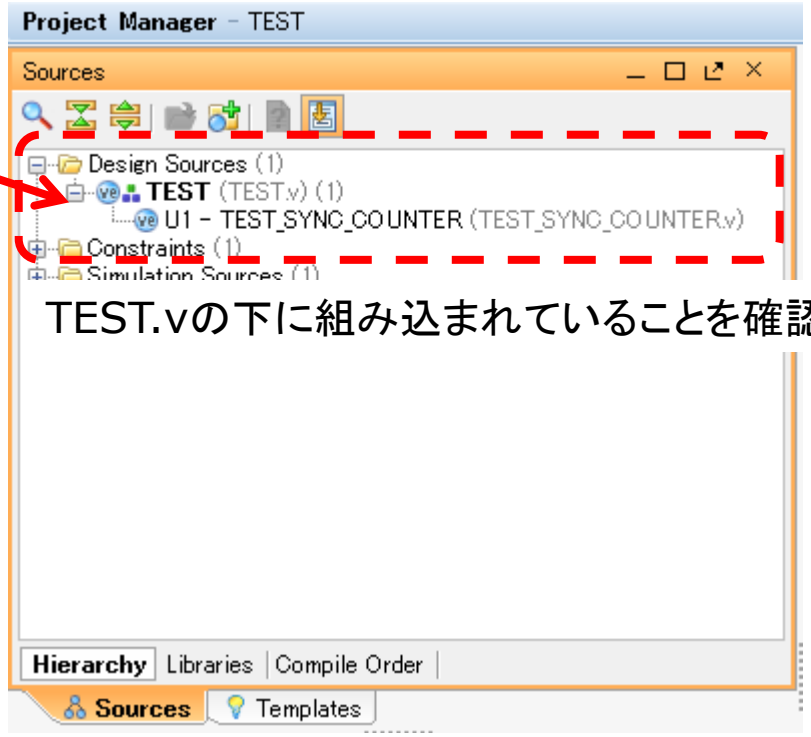
ここでは、
下位モジュールのポートCLKをOSCに接続
下位モジュールのポートRSTnをRST_SWnに接続
下位モジュールのポートCLKをOSCに接続

終わったら保存してください

組み込まれた事を確認



+をクリックして展開



TEST.vの下に組み込まれていることを確認

組み込まれていない、エラーがある場合は記述が間違っているので確認して修正してください

合成し動作確認してください

1. RTL解析し回路図確認
   - 講義資料「2.2 HDL入力とRTL解析」参照
2. シミュレーション実行
   - 講義資料「 4.2 論理シミュレーション」参照
   - テストベンチ等の変更は不要
     - テストベンチから見た回路はまったく同じなので。
3. 論理合成、配置配線、データ生成
   - 講義資料「 4.3 FPGAへの実装」参照
4. 実機による動作確認
   - LEDが点滅すればOK

履歴

- 2015/7/31　第3.0版 本章追加　内田智久(Esys, KEK/総研大)
- 2015/8/7　第3.1版 本章追加　内田智久(Esys, KEK/総研大)
- 2015/12/04 第3.2版 コード中のリセットを同期から非同期へ変更　内田智久(Esys, KEK/総研大)
- 2016/01/27 第3.3版 Vivado2015.4対応　内田智久(Esys, KEK/総研大)
- 2016/06/22 第3.4版 Vivado2015.4対応　内田智久(Esys, KEK/総研大)