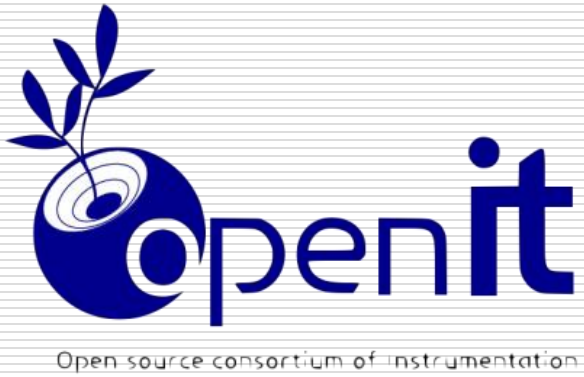
Open-It FPGAトレーニングコース（入門編）

4.2 論理シミュレーション（順序回路）



第3.3版

2016年06月22日

この節の概要

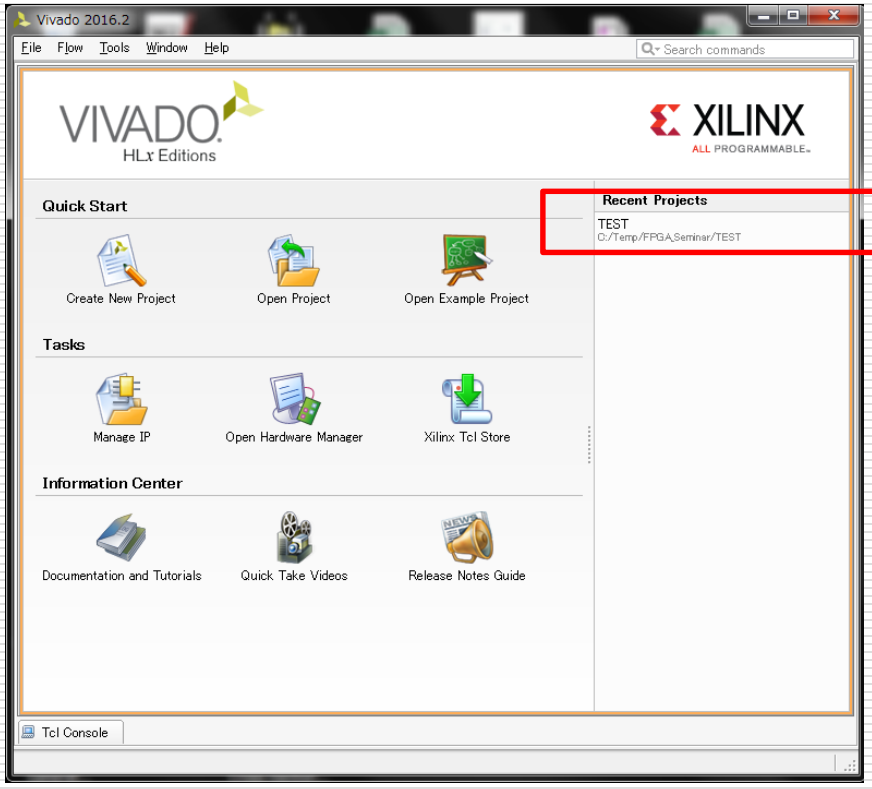
基本的な操作は組み合わせ回路と同じです

1. ソースコード修正
   - カウンターを追加する
2. テストベンチ修正
   - クロックやリセット信号の追加
3. シミュレーションの実行

ソースコード修正

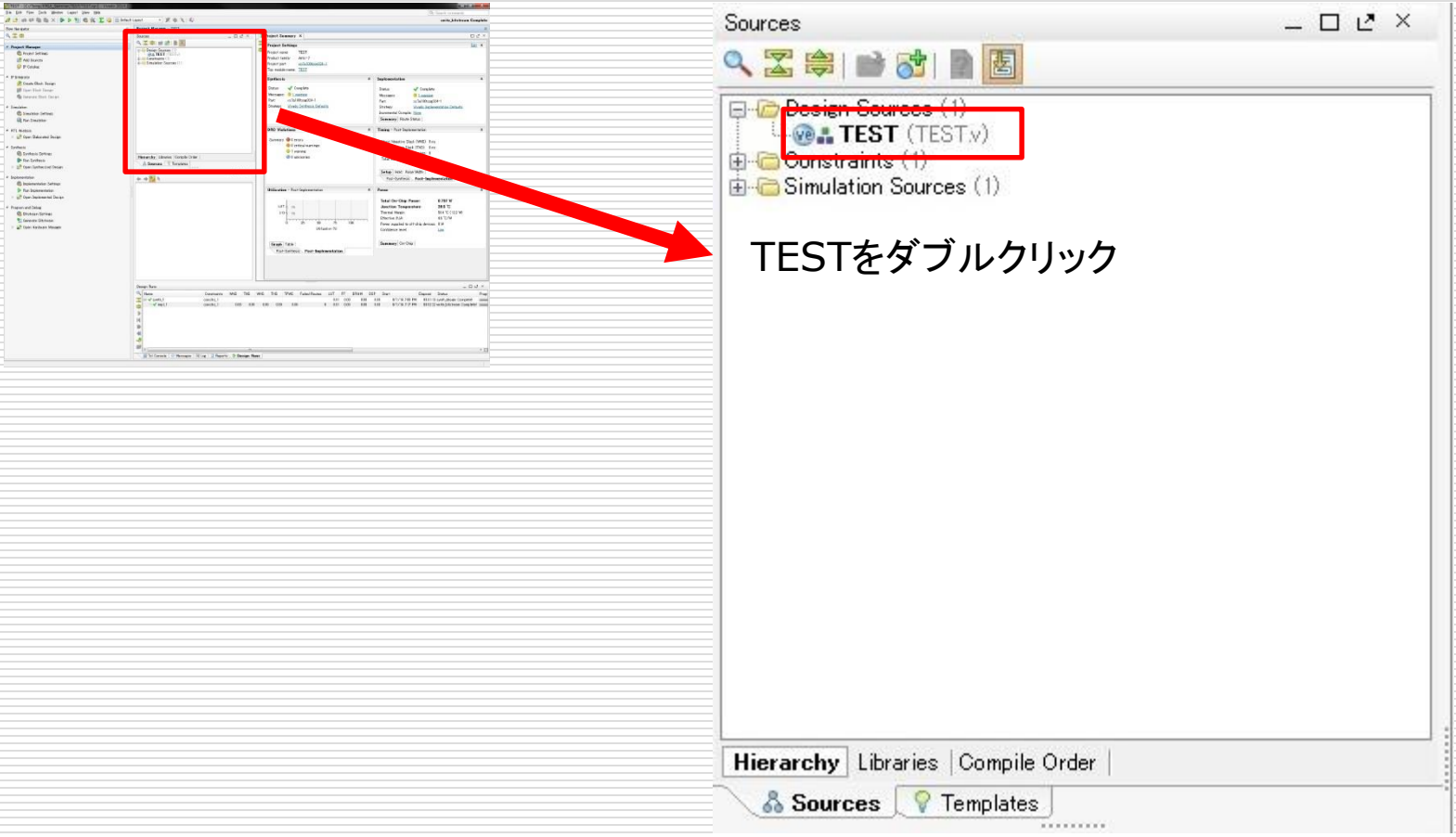
Vivado起動

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


Sources
Design Sources (1)
TEST (TEST.v)
Constraints (1)
Simulation Sources (1)
TESTをダブルクリック
Hierarchy
Libraries
Compile Order
Sources
Templates

コード修正

C:/Temp/FPGA_Seminar/TEST/TEST.srcs/sources_1/new/TEST.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2014/08/01 17:17:00
7  // Design Name:
8  // Module Name: TEST
9  // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 //////////////////////////////////////////////////////////////////////////////////
21
22
23 module TEST(
24     input SW_A,
25     input SW_B,
26     output LED0
27     );
28
29     assign LED0 = SW_A & SW_B;
30
31 endmodule
32
```

HDL内のコードは同時に動作しますのでどこに書いても構いません。
ここでは、説明のために下の指定箇所に追加します。

ここに同期カウンターを追加します

カウンターを追加

```
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     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];
45
46 endmodule
47
```

OSC(クロック)とRST_SWn(リセット)を追加

LED15(出力)追加

カウンター追加
(次ページで詳細説明)

出力をLEDに接続
ここではsyn_counter[28]を出力している

カウンター記述詳細

DFF出力信号はregで宣言

カッコ中の条件が成立したら実行する

```
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];
```

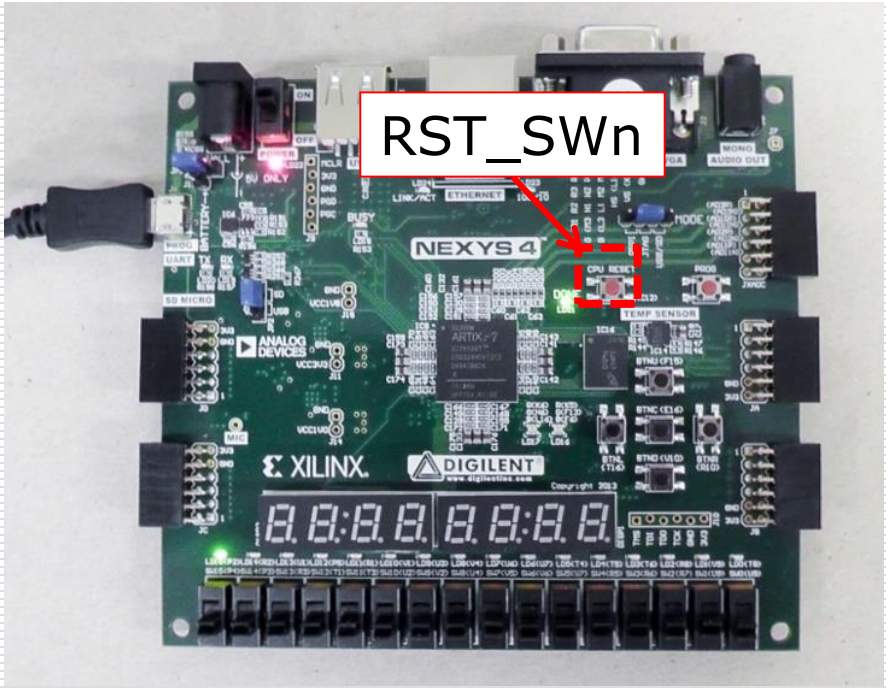
RST_SW==0の時に実行

RST_SW!=0の時に実行

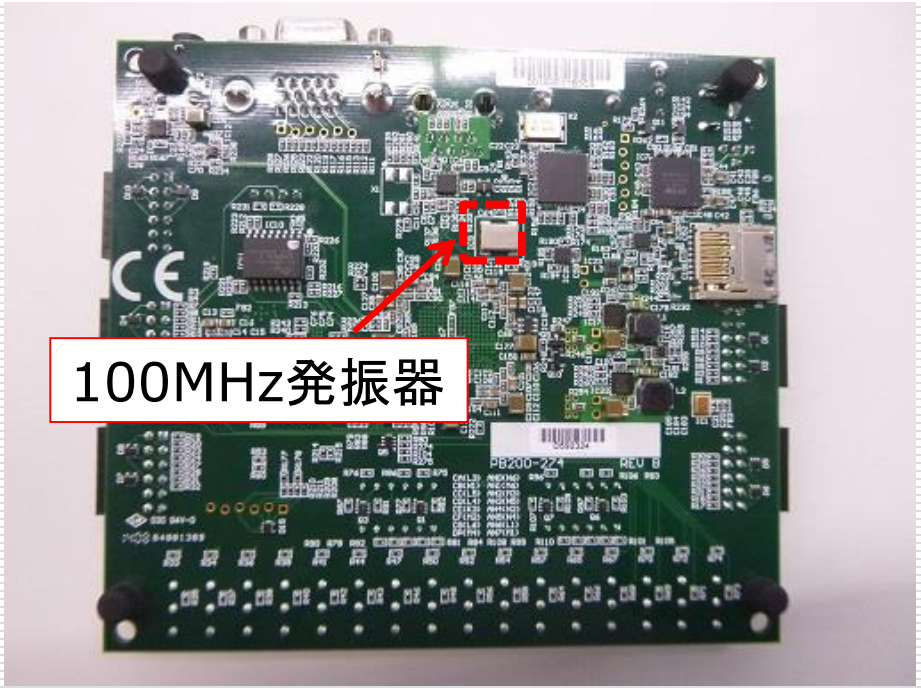
修正が終わったらセーブしてファイルを閉じてください

ボード上のリセットスイッチと発振器

表



裏

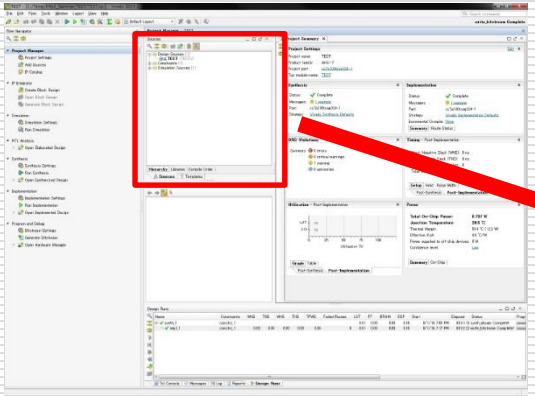


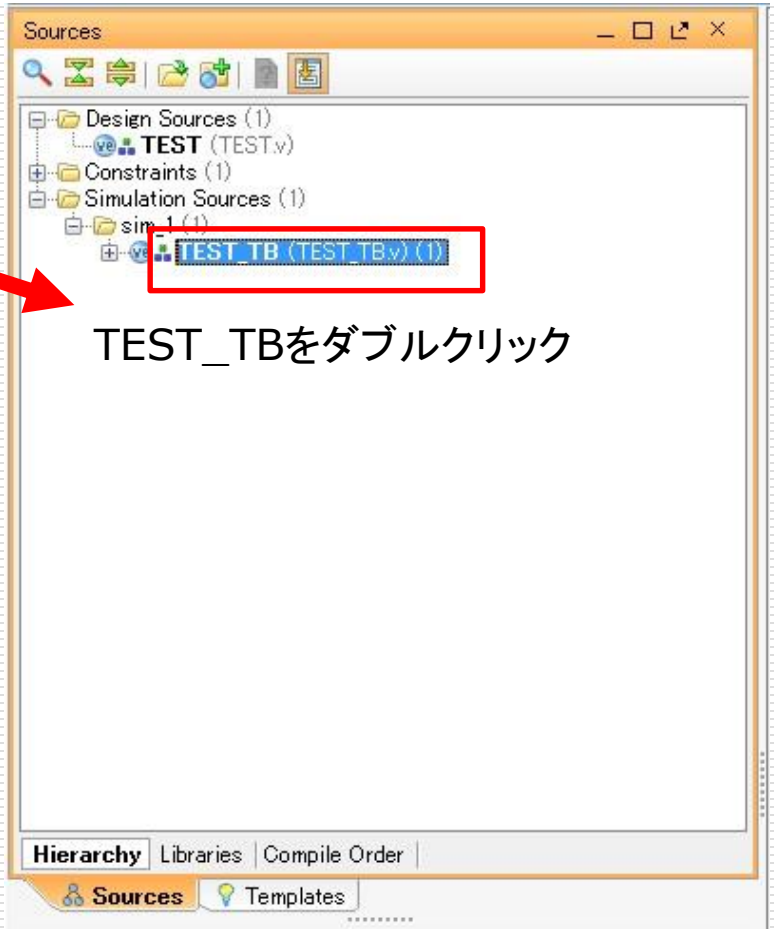
テストベンチ修正

TEST_TB.vのコードの修正

- 追加する信号
  - CLK100M
    - 100MHzのクロック
    - デューティ比が50%（1と0の割合が1:1）
  - RST_SWn
    - 0の時にリセットがかかる
  - LED15
    - Counterの29ビット目の出力
      - counter[0]を1ビット目としてcounter[28]は29ビット目

ファイルを開く




Sources
Design Sources (1)
TEST (TEST.v)
Constraints (1)
Simulation Sources (1)
sim_1 (1)
TEST_TB (TEST_TB.v) (1)
TEST_TBをダブルクリック
Hierarchy
Libraries
Compile Order
Sources
Templates

コード修正場所

```
22
23 module TEST_TB;
24
25     reg SW_A;
26     reg SW_B;
27     wire LED0;
28
29     TEST uut(
30         .SW_A(SW_A),
31         .SW_B(SW_B),
32         .LED0(LED0)
33     );
34
35     initial begin
36         SW_A = 0;
37         SW_B = 0;
38
39         #100 SW_A = 1'b1;
40         #300 SW_B = 1'b1;
41         #200 SW_A = 1'b0;
42     end
43
44 endmodule
```

信号追加（CLK100M, RST_SWn, LED15）

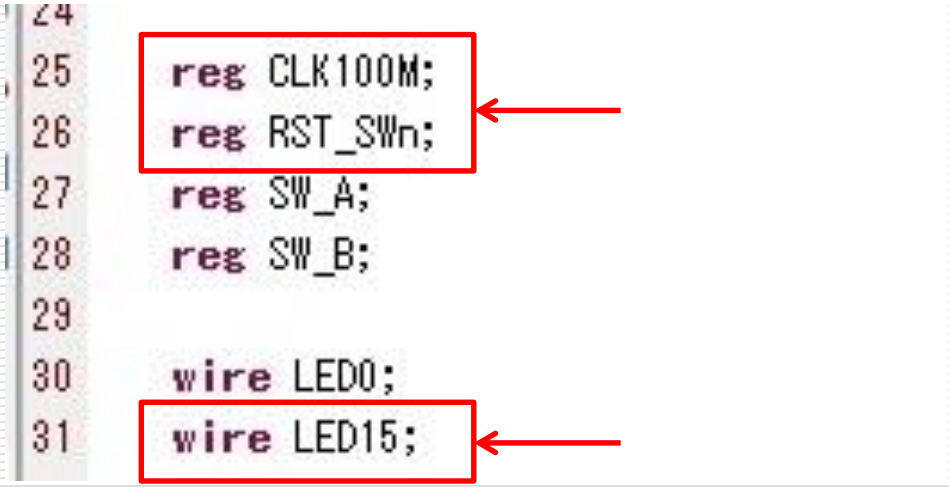
ポート追加（CLK100M, RST_SWn, LED15）

HDL内のコードは同時に動作しますのでどこに書いても構いません。
ここでは、説明のために下の指定箇所に追加します。

クロックとリセット動作の記述します

信号定義の追加

信号追加(CLK100M, RST_SWn, LED15)



ポートの追加

ポート追加（CLK100M, RST_SWn, LED15）

```
33    TEST uut(
34        .OSC(CLK100M),
35        .RST_SWn(RST_SWn),
36        .SW_A(SW_A),
37        .SW_B(SW_B),
38        .LED0(LED0),
39        .LED15(LED15)
40    );
```

ここに注意！！

モジュールのポート名はOSC、
テストベンチの信号名はCLK100M

クロックとリセット信号動作の記述

```
51  parameter PERIOD = 10;
52
53  always begin
54      CLK100M = 1'b0;
55      #(PERIOD/2);
56      CLK100M = 1'b1;
57      #(PERIOD/2);
58  end
59
60  initial begin
61      RST_SWn = 1'b0;
62      #700 RST_SWn = 1'b1;
63  end
```

クロック記述

変数を使うことができる。
HDLでの変数はコードにより動的変更されるものではなく、設定値を変更しやすくするために使用する。ここでは、周波数変更が簡単になる

100MHzクロックの周期＝10ns

動作条件なしalways文は繰り返し実行される
文中の命令は上から下に順に実行する

リセット記述

Initial文は一度だけ実行される
文中の命令は上から下に順に実行する

クロックの書き方

- 書き方はいろいろあります
  - 今回は一例を紹介します

定数宣言

周期

```
parameter PERIOD = 20;
always begin
    CLK50M = 1'b0;
    #(PERIOD/2) CLK50M = 1'b1 ;
    #(PERIOD/2);
end
```

  - 現在の設定では50MHzのクロックになります

修正したテストベンチ

```
22
23 module TEST_TB;
24
25     reg CLK100M;
26     reg RST_SWn;
27     reg SW_A;
28     reg SW_B;
29
30     wire LED0;
31     wire LED15;
32
33     TEST uut(
34         .OSC(CLK100M),
35         .RST_SWn(RST_SWn),
36         .SW_A(SW_A),
37         .SW_B(SW_B),
38         .LED0(LED0),
39         .LED15
40     );
```

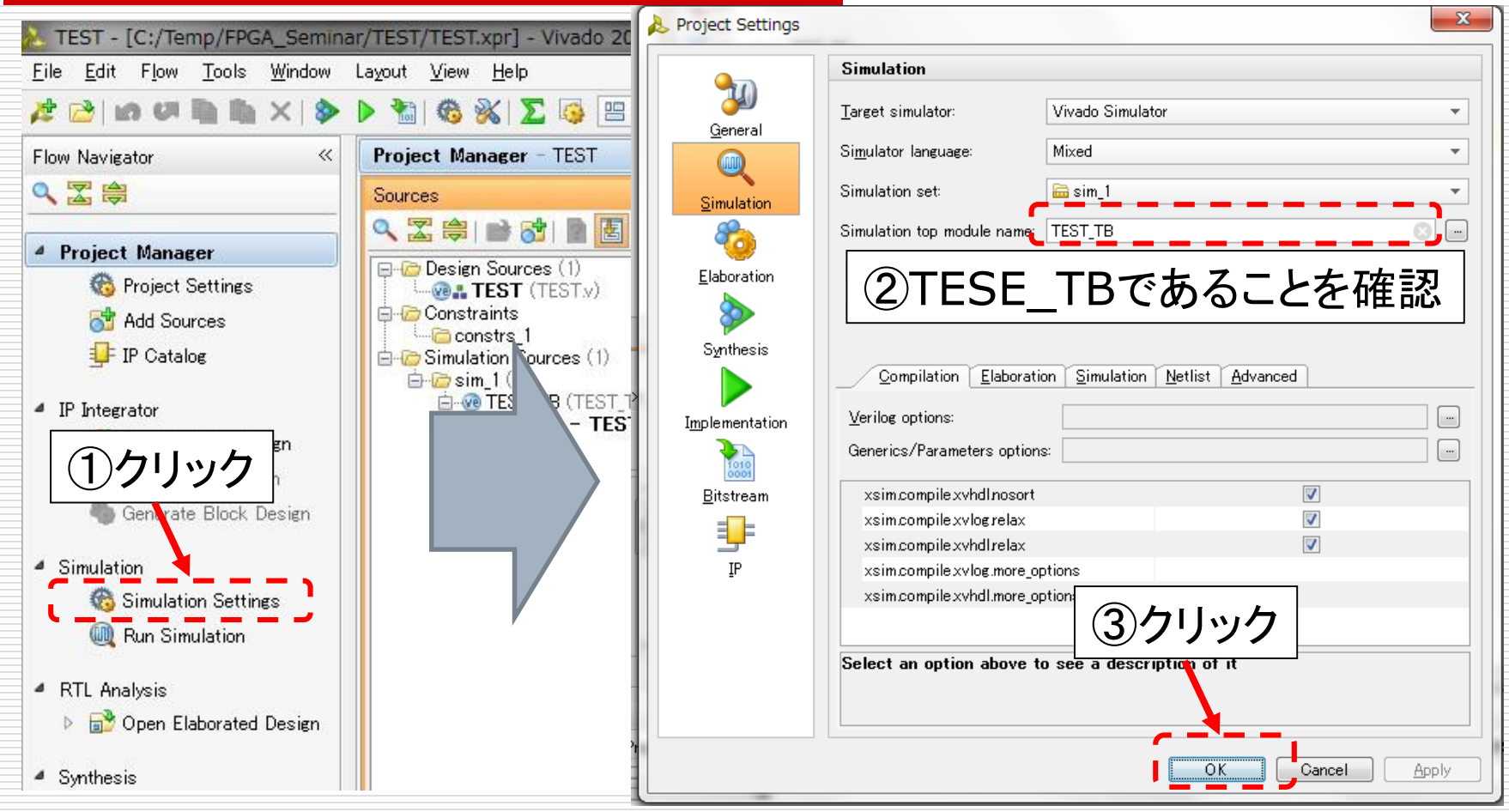
```
42     initial begin
43         SW_A = 0;
44         SW_B = 0;
45
46         #100 SW_A = 1'b1;
47         #300 SW_B = 1'b1;
48         #200 SW_A = 1'b0;
49     end
50
51     parameter PERIOD = 10;
52
53     always begin
54         CLK100M = 1'b0;
55         #(PERIOD/2);
56         CLK100M = 1'b1;
57         #(PERIOD/2);
58     end
59
60     initial begin
61         RST_SWn = 1'b0;
62         #700 RST_SWn = 1'b1;
63     end
64
65 endmodule
```

これらの2つのinitia文は同時に動く

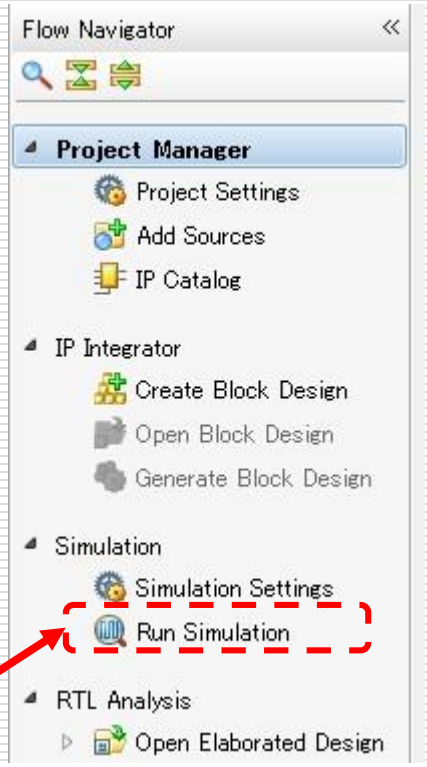
修正が終わったらセーブしてファイルを閉じてください

論理シミュレーションの実行

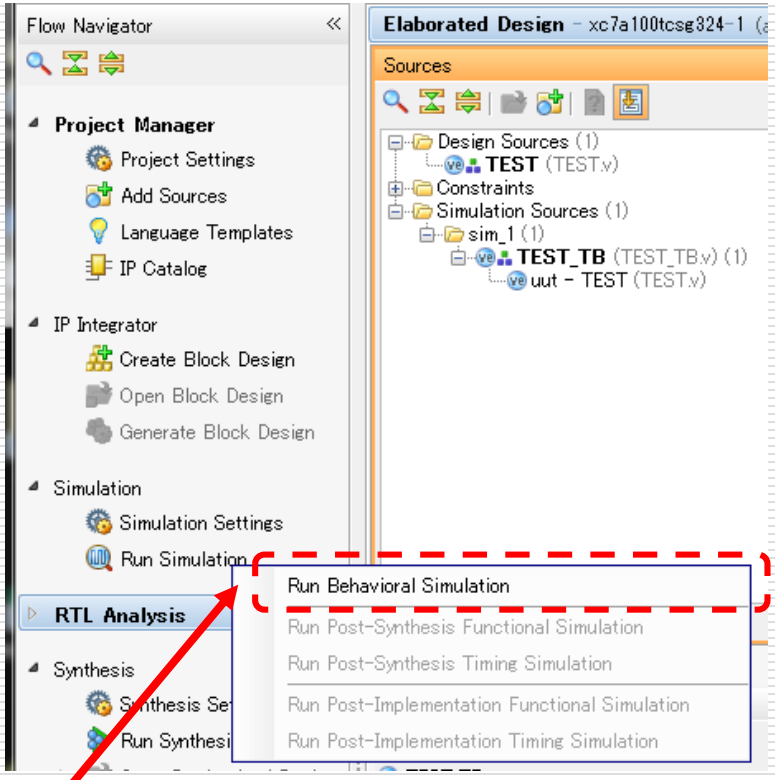
シミュレーション設定確認



シミュレータ起動



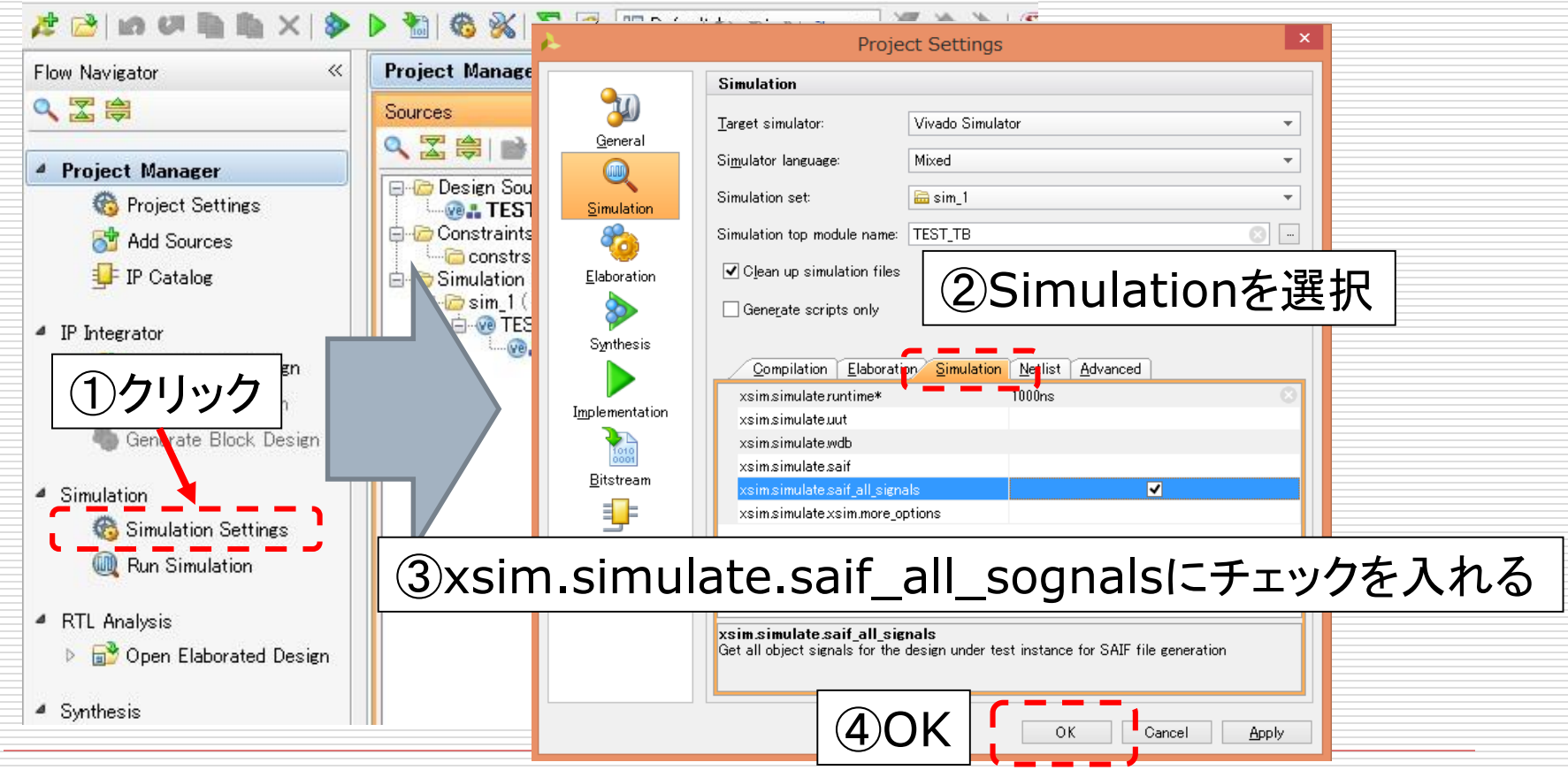
①クリック



②「Run Behavioral Simulation」クリック

シミュレーターが起動しない場合

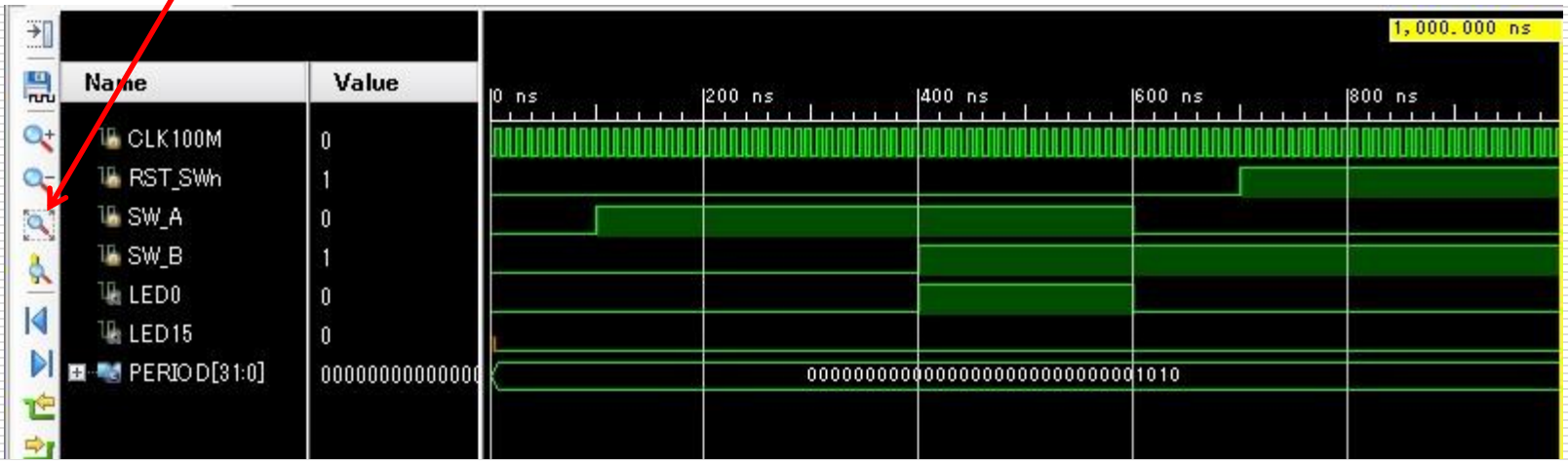
エラーも何も表示されずにシミュレータが起動しない場合は下の設定を試してください



結果画面

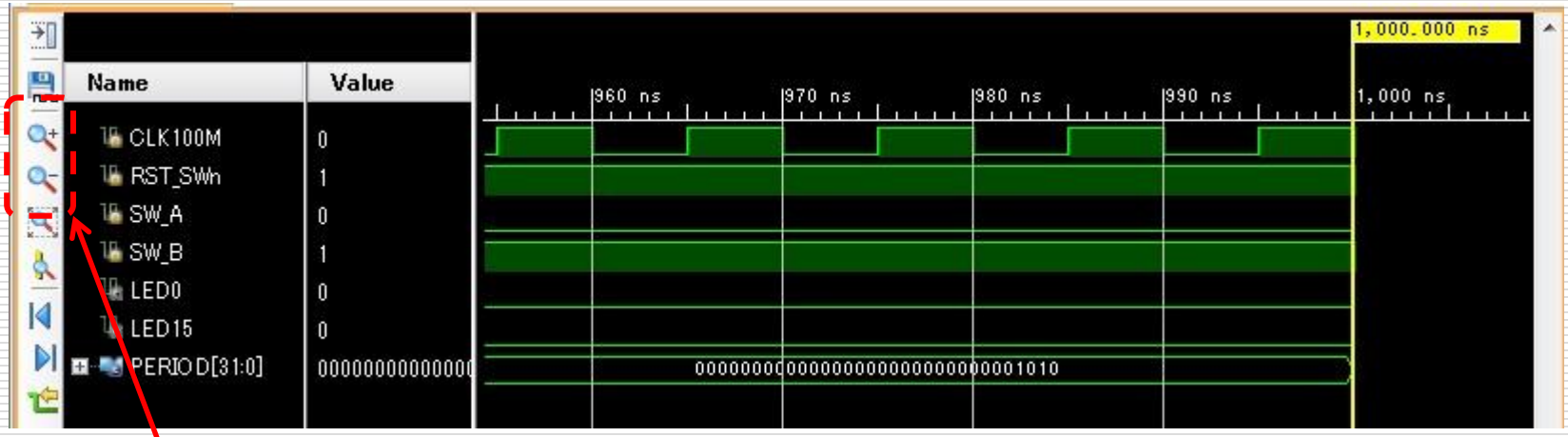
ここをクリック
Zoom-Fit

全体を見てください



クロック周期の確認

- クロック周期の確認
  - 昨日と同様の手順で確認します

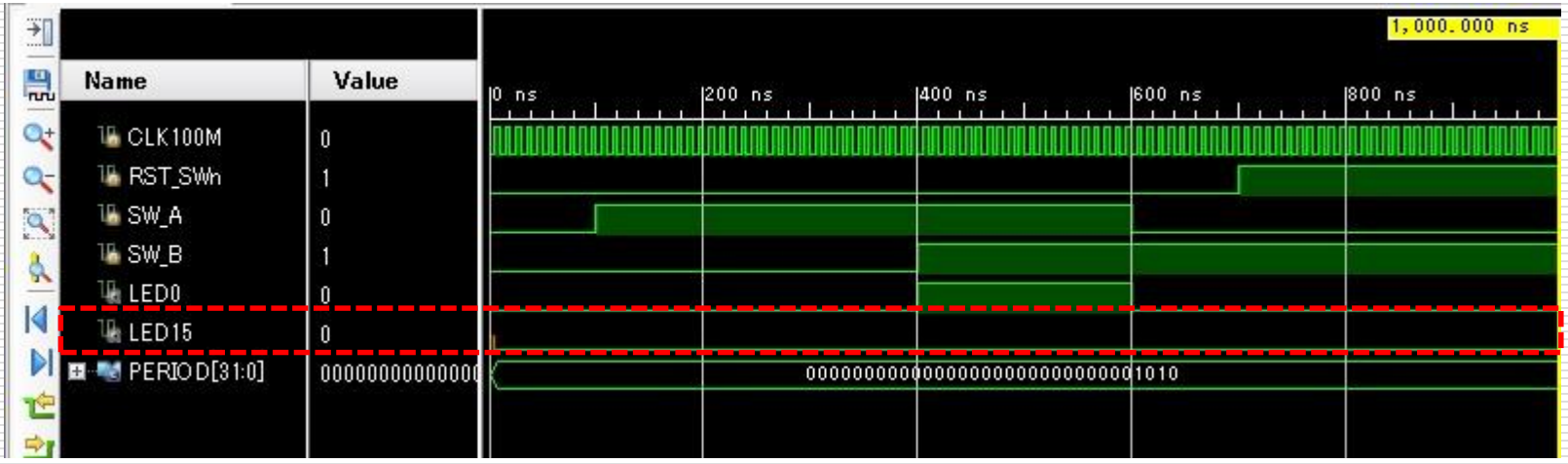


表示の拡大・縮小

10nsでしたか？

LED15の確認

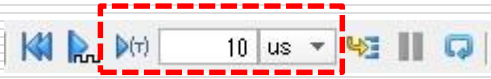
- LED15（Counter[28]の出力）の確認



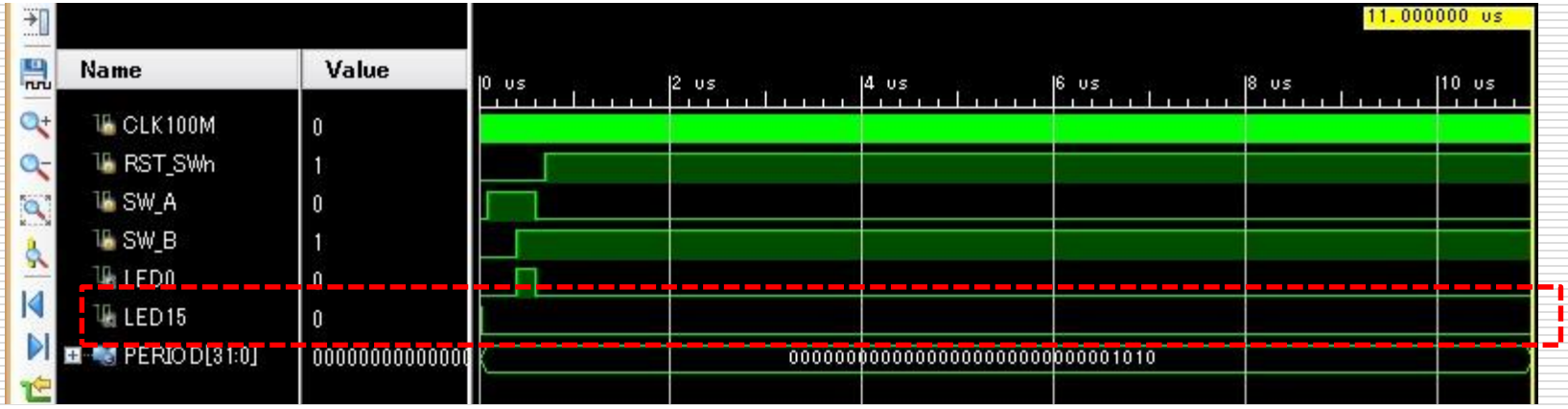
変化がないので正しいかどうかわかりません
シミュレーション時間を追加してみます

LED15の確認

上部のツールバーを使って



10usシミュレーションを追加実行してみます



まだ変化はありません

カウンターの29bit目が動くのは

$$2^{28} \times 10ns(\text{周期}) \approx 2.7s$$

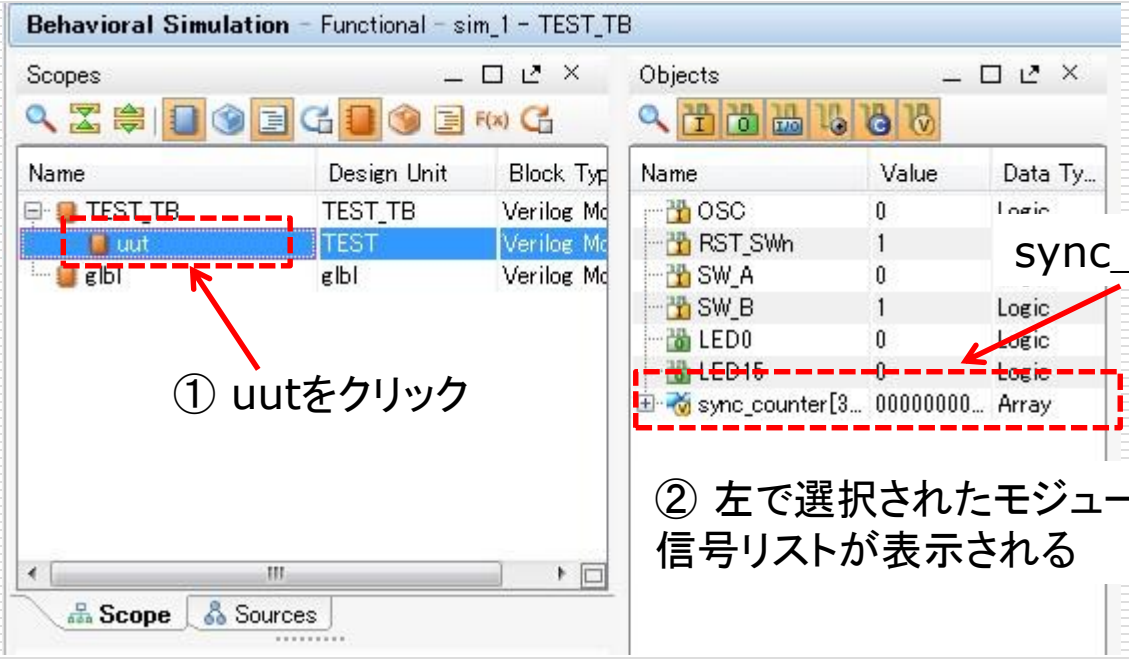
LED15の確認

- 時間を掛ければシミュレーションはできますが数分、数十分、数時間と長時間かかります
- この様な場合は他の方法で動作していることを確認します
- ここではカウンターの下位ビットが正しく変化していることを確認する事でLED15を確認したと見なします

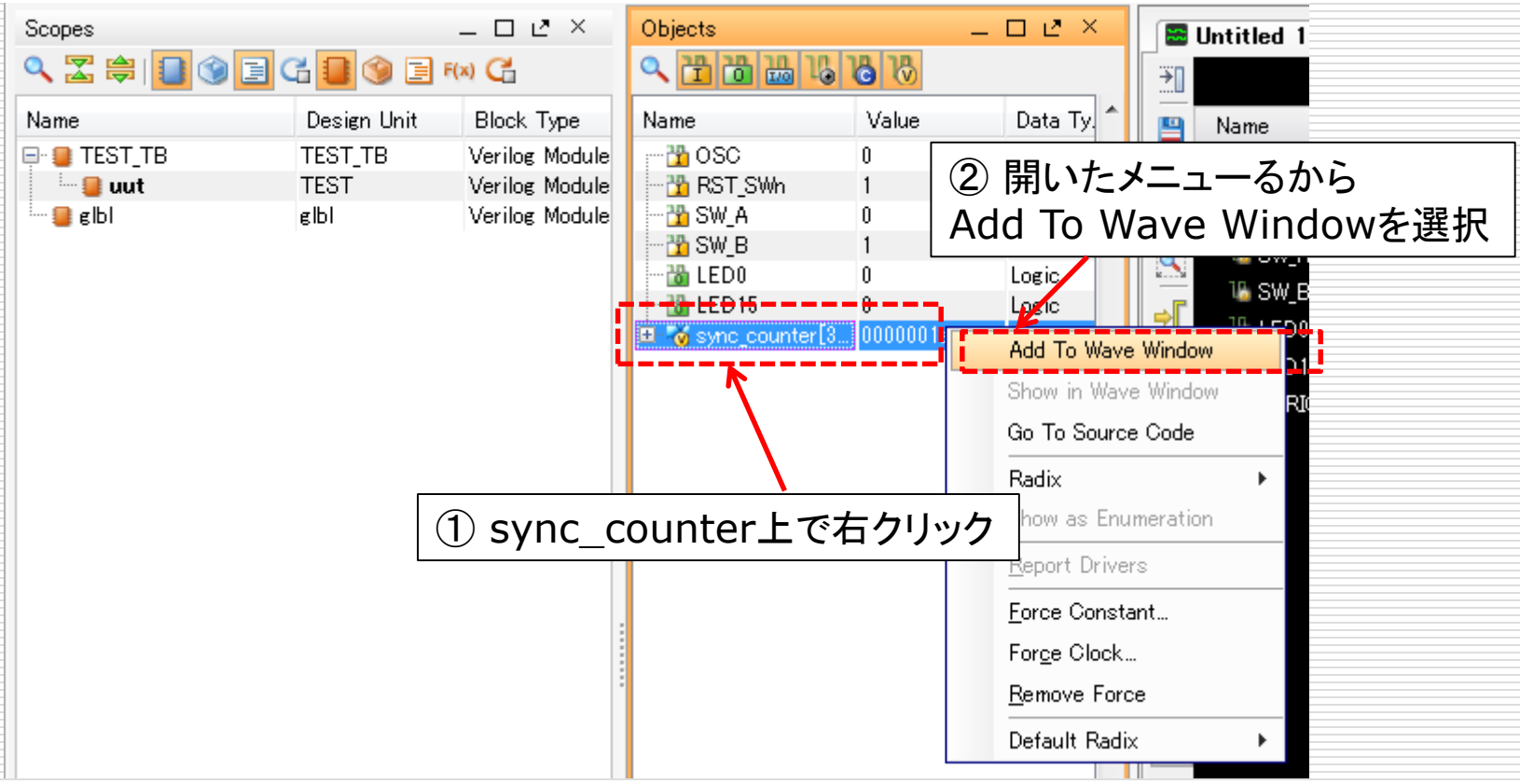
下位モジュール信号の確認

- テストベンチの信号のみ波形として表示されています
- 下位モジュールの信号も観測する事ができます
  - シミュレーションの良い点は全ての信号を簡単に観測する事ができることです
- sync_counter信号を観測します

下位モジュール信号の表示



sync_counter信号の波形表示

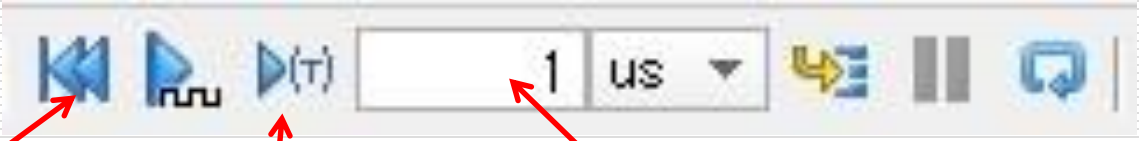


または、①でsync_counterをクリックしたまま波形窓にドラックアンドドロップしても追加できる

再シミュレーション

sync_counterの波形を表示させます

上部のツールバーから



① Restart

② 1 usを設定

③ Run for x sec

sync_counterの波形が表示されました。

リセット解除付近を拡大

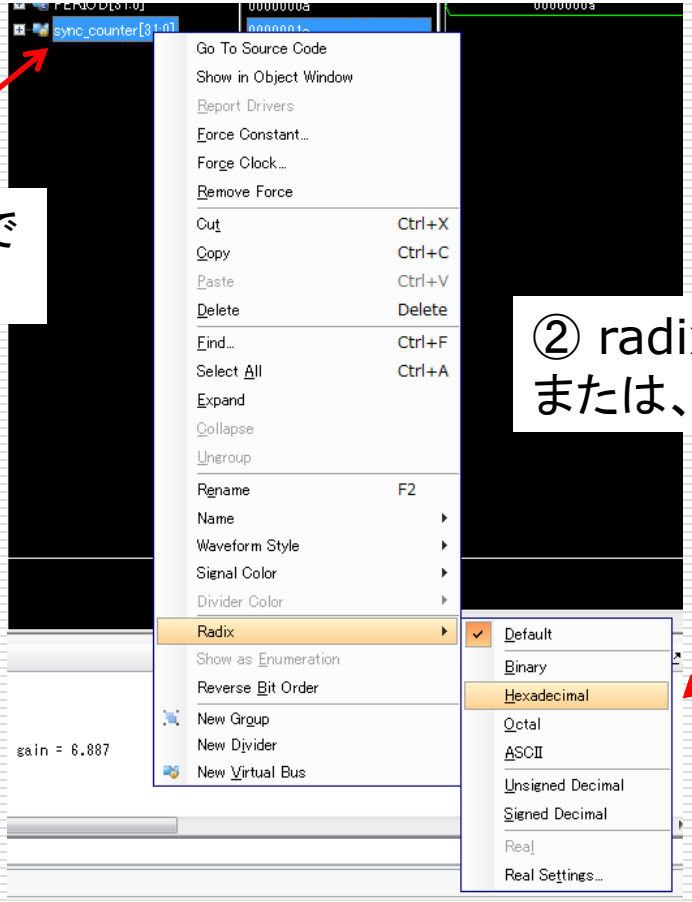
700ns付近を拡大してください

① カーソルを700ns付近に置く

② 拡大アイコンを何度かクリック

マルチビット信号の表示形式変更方法

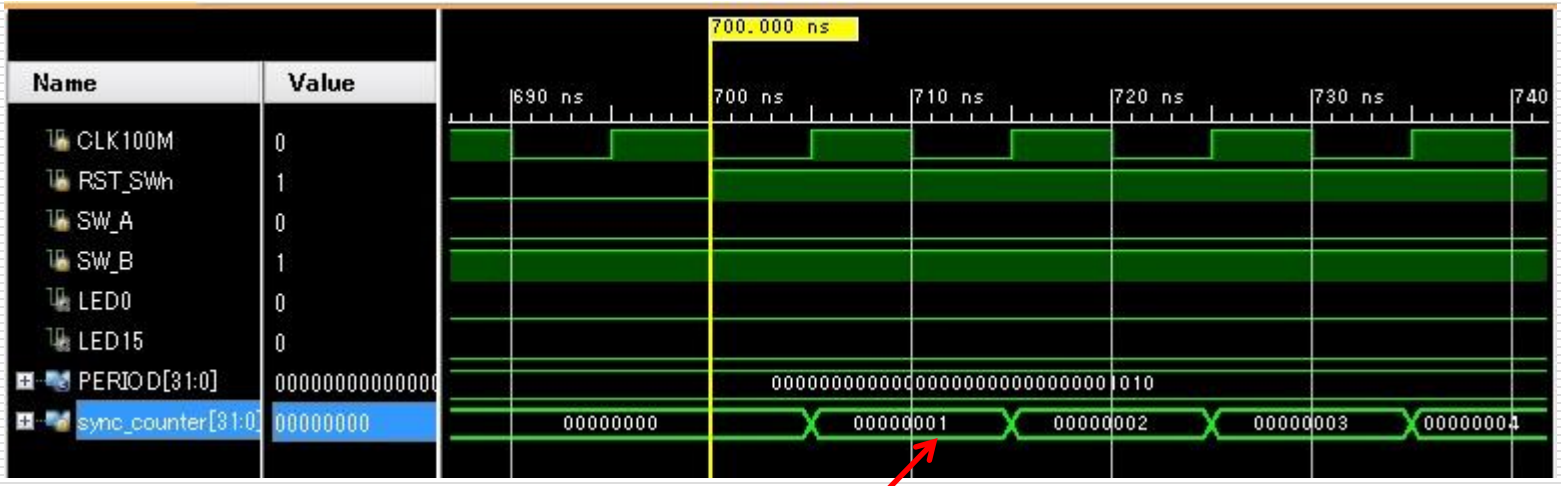
① sync_counterの上で
右クリック



② radixの中のHexadecimal
または、Unsigned Decimalを選択

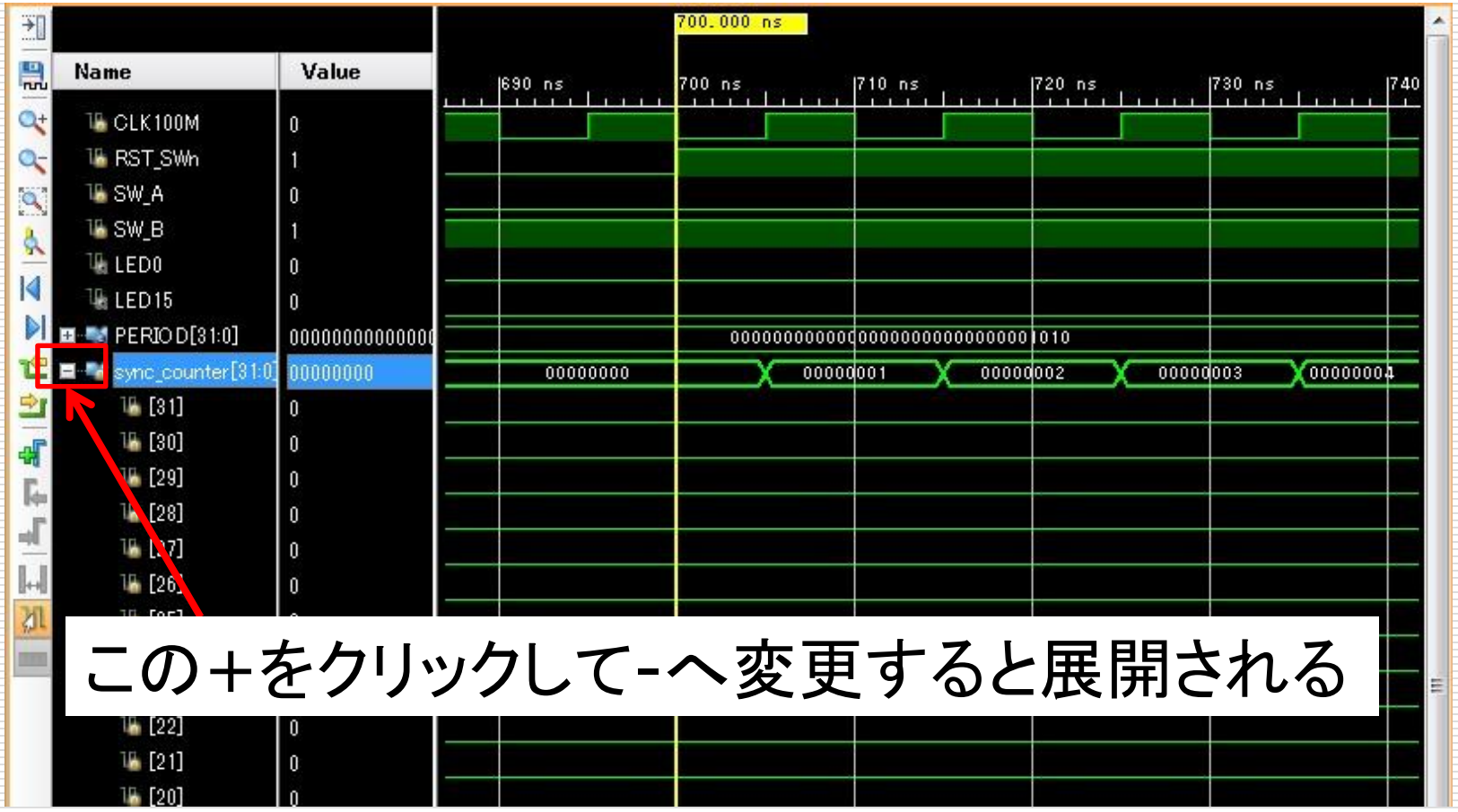
16進表示結果

クロックの立ち上がりの時にリセットが
　0の時はカウンターは0に
　1の時はカウンターが正常に動いていることが確認できます



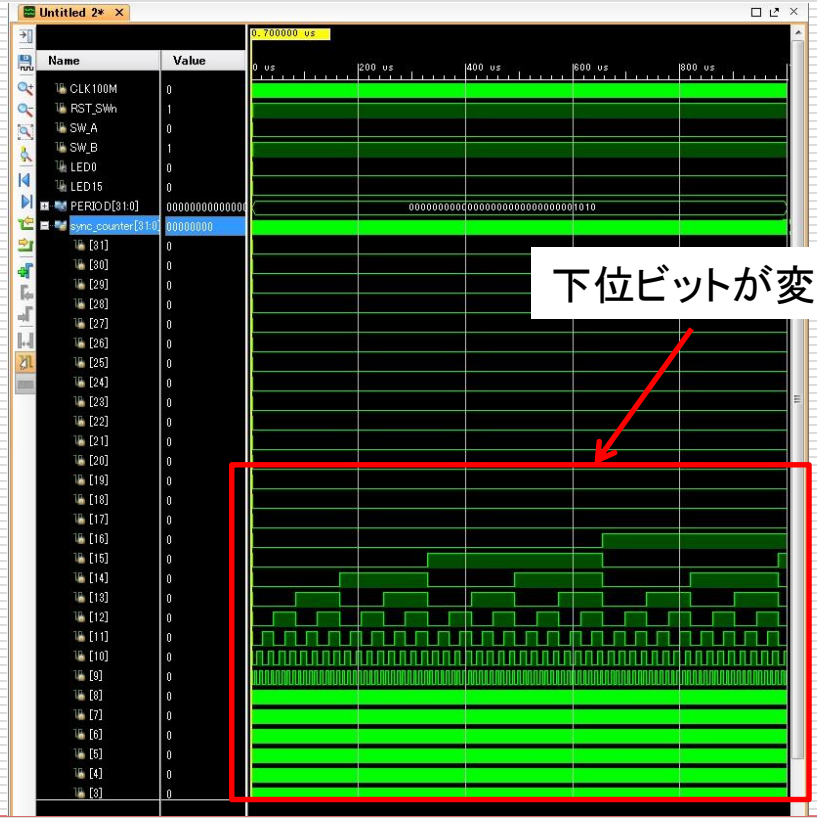
希望する表示になった

各ビットの波形表示


Name
Value
CLK100M
RST_SWn
SW_A
SW_B
LED0
LED15
PERIOD[31:0]
sync_counter[31:0]
[31]
[30]
[29]
[28]
[22]
[21]
[20]
700.000 ns
690 ns
700 ns
710 ns
720 ns
730 ns
740
00000000
00000001
00000002
00000003
00000004
この+をクリックして-へ変更すると展開される

シミュレーション時間延長

上部ツールバーにより追加で1msecシミュレーションしてください



拡大し正しくカウンター動作していることを確認してください

シミュレータ終了

- 正しく動作していることを確認する事ができたらシミュレータを終了してください
- 終了するときに、波形情報を保存するか聞かれますが、今回は保存しない「No」を選択してください。
  - 表示する波形、表示順、数字の表現形式などを保存する事ができる。保存すると次回以降設定ファイルを読み込むことで同じ波形表示にすることができる

参考文献

- Xilinx, UG937, Vivado Design Suite Tutorial:Logic Simulation

履歴

- 2012/5/17 第1.0版 ISE13.4対応　内田智久(Esys, KEK/総研大), 林達也（大阪大学）
- 2014/8/7　第2.0版 Vivado2014対応、章構成変更　内田智久(Esys, KEK/総研大)
- 2015/7/31　第3.0版 Vivado2015対応、章構成変更　内田智久(Esys, KEK/総研大)
- 2015/12/04 第3.1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3.2版　Vivado2015.4対応　内田智久(Esys, KEK/総研大)
- 2016/06/22 第3.3版　Vivado2016.2対応　内田智久(Esys, KEK/総研大)