# ModelSim を用いた回路シミュレーション

**学習内容**

STEP 08 で作成した回路は動作が速すぎて、実機を目で見て確認することはできませんでした。そこで回路シミュレーションを用いて動作を可視化しましょう。

シミュレーションは回路が複雑になるほど重要です。
本書では論理シミュレータに「ModelSim」使用します。

## 1. ModelSim とは

ModelSim は Mentor Graphics 社から提供されているシミュレータです。シミュレータでありながら、回路の動作を波形として観測することができます。回路の動作を波形として観測することで、設計した回路の問題点や通常人の目では観測できないような高速な信号の動作を知ることができます。

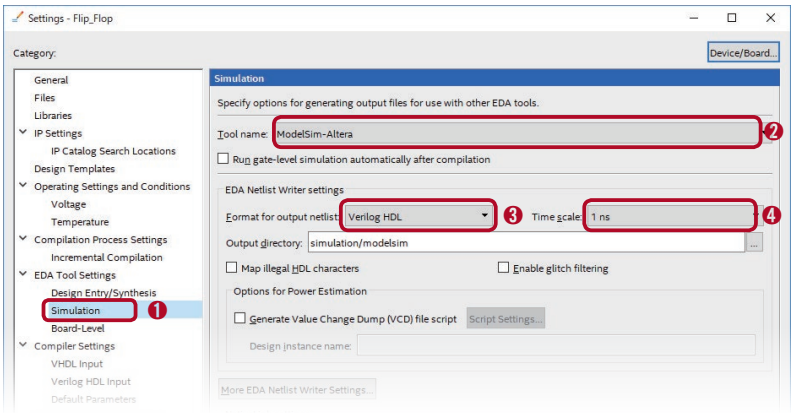
このステップでは Quartus Prime と連動した ModelSim-Altera を使用した回路シミュレーションを学んでいきます。

## 2. ModelSim の使用設定

ModelSim を使用できるようプロジェクトを設定しましょう。
まず、STEP 08-1 の Quartus Prime プロジェクトを開きます。

設定は ①②のいずれかの方法で Settings ウィンドウを開きます。
① Quartus Prime のメニューから Assignment > Settings を選択。
② ツールバーの「Settings」ボタン をクリック。

❶ Settings ウィンドウの Category 欄で EDA Tool Settings の Simulation を選択。
❷ Toolname 欄で ModelSim-Altera を選択。
❸ Format for output netlist 欄で Verilog HDL を選択。
❹ Time scale はシミュレーション計算を行う時間間隔です。シミュレーションで観測したい信号の速さに合わせて設定します。今回はクロック 50MHz で 1 周期 20.0ns なので 1ns 程度でいいでしょう。
❺ OK をクリックしてウィンドウを閉じます。

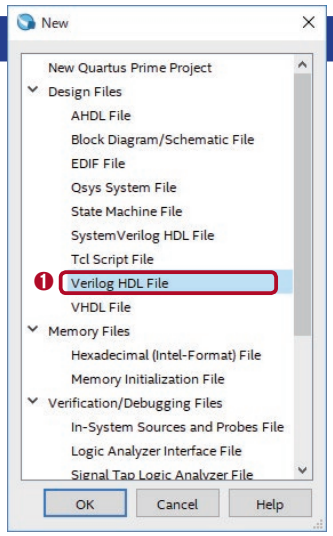


# ModelSim を用いた回路シミュレーション

## 3. テストベンチとは

回路シミュレーションは、擬似的に入力信号を作りそれに対する出力信号を観測します。このときのいくつもの入力信号のパターンを記述したものをテストベンチと呼びます。このテストベンチが適切でなければ正しいシミュレーション結果が出ないので注意して作成してください。

## 4. テストベンチファイルの作成

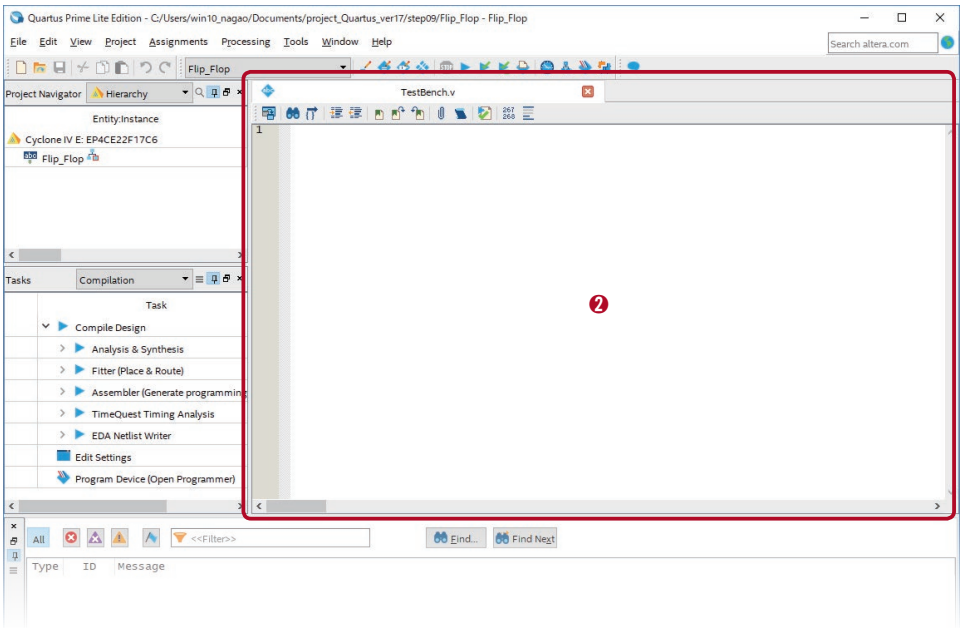



テストベンチも Verilog で記述していきます。

❶ File ＞ New で「Verilog HDL File」を選んで OK をクリック。

❷ すると、verilog1.v という新しい verilog ファイルが生成され画面が表示されます。

❸ このファイルを「TestBench.v」と名前をつけて保存しておきます。



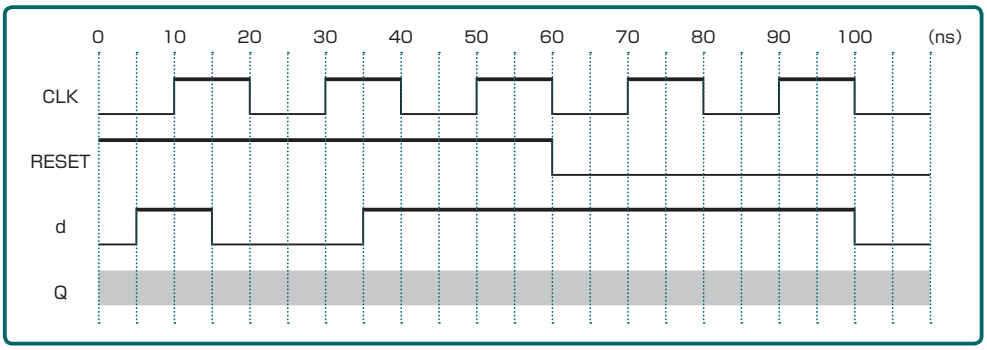
# ModelSim を用いた回路シミュレーション

テストベンチを記述していく前に、どんな入力信号を与えるか考えます。今回、入力信号には以下のような条件の信号を使用します。

・End Time（シミュレーション時間）：100ns
・Time Scale（シミュレーション計算間隔）：1ns
・CLK 信号：50MHz（20ns 周期）のクロック
・RESET 信号：0 ～ 60ns までは High(1)、60 ～ 80ns が Low(0)、80 ～ 100ns が High(1)
・d 信号：0 ～ 5ns が Low(0)、5 ～ 15ns が High(1)、15ns ～ 35ns が Low(0)、35 ～ 100ns が High(1)

シミュレーション時間を長く、シミュレーション計算間隔を細かくしすぎると、ModelSim がエラーを返す場合があります。設定時間は適切にしてください。

この条件を波形で表すと以下のようになります。
CLK、RESET、d の 3 つの信号によって 出力 Q がどのように変化するかシミュレーションするのです。




この条件のテストベンチを Verilog HDL で記述すると次ページのようになります。

# ModelSim を用いた回路シミュレーション

TestBench.v

弊社サイトに解答例ソースをご用意しています。http://www.adwin.com/product/AKE-1104.html

```
1  `timescale 1 ns/ 1 ns
2  module TestBench;
3
4      reg CLK, RESET, D;
5      wire Q;
6
7
8      Flip_Flop Test(.CLK(CLK), .RESET(RESET), .D(D), .Q(Q));
9      always
10         #10 CLK = ~CLK;
11
12     initial
13     begin
14         RESET = 1'b1; CLK = 1'b0; D = 1'b0;
15         #5  D = 1'b1;
16         #10 D = 1'b0;
17         #20 D = 1'b1;
18         #25 RESET = 1'b0;
19         #20 RESET = 1'b1;
20         #20 $stop;
21     end
22
23 endmodule
```

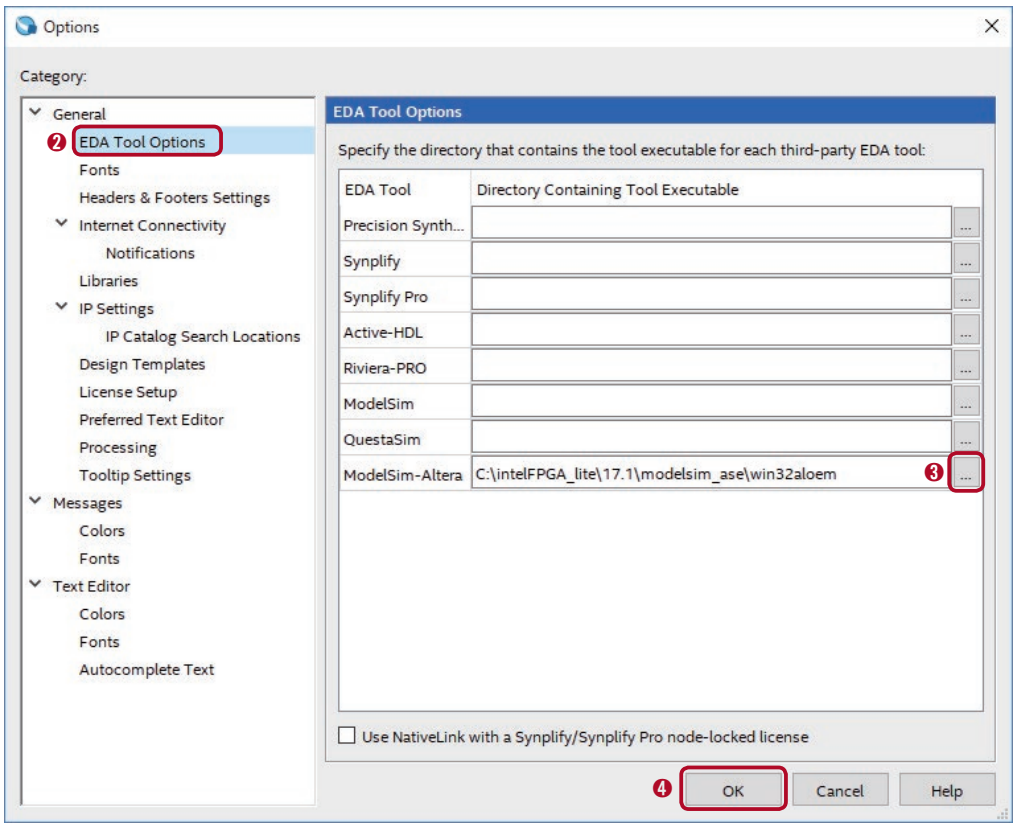
- 1行目：シミュレーションの 単位時間 / 精度 を指定。時間の単位は fs、ps、ns、us、ms、s が利用できる。
- 2行目：テストベンチもモジュールとして宣言する。ここではモジュール名を TestBench とした。テストベンチ用モジュールは入出力信号の記述は不要。
- 4行目：テスト入力信号は reg 宣言する。
- 5行目：テスト出力は wire 宣言する。出力信号を観測するだけなので wire 宣言で OK。
- 6〜7行目：step08-1.v で作成したモジュール Flip_Flop を呼び出し、名前を Test としている。モジュール Flip_Flop に与える信号を括弧で指定する。ここで与える信号の順番は Flip_Flop モジュールの信号順と必ずそろえること。この記述方法は階層化構造と呼ばれるもので詳しくは STEP14 で解説。
- 9行目：always 文で一定の繰り返し動作を記述している。
- 10行目：#10 と記述することで 10ns 後を表す（単位時間：1ns のため）10ns 毎に CLK 信号を反転。これによって 20ns 周期のクロック信号を生成。
- 12行目：initial 文は、シミュレーション開始直後に呼び出されて実行される。initial 文のなかにテスト信号の動きを記述。
- 14行目：initial 文が呼び出されたら、まず各信号の初期値を設定。このとき # を記述しないことでこの状態を 0ns（スタート）としてシミュレーションが実行される。
- 15〜19行目：初期値を設定したら先ほど考えた条件通りのテスト信号を生成するよう記述する。信号を記述するときは「# 時間 処理 ;」という形で記述。このとき注意しなければならないのが「# 時間」はスタートからの経過時間ではなく、ひとつ前の処理が実行されてからの遅延時間だということ。つまり # 時間の合計がシミュレーション時間となる。
- 20行目：$stop はシミュレーションの停止を示す命令。5+10+20+25+20+20 = 100 このテストベンチは 100ns で終了（単位時間：1ns のため）

# ModelSim を用いた回路シミュレーション

## 5. シミュレーションの設定

Quartus Prime と ModelSim-Altera との連動やシミュレーションのための設定をしていきます。

❶ Quartus Prime のメニューから Tool > Options をクリック。
❷ 表示された画面で EDA Tool Options を選択。
❸ Modelsim-Altera の参照ボタンから modelsim.exe の実行ファイルがあるフォルダを指定。
デフォルトでは C:\intelFPGA_lite\17.1\modelsim_ase\win32aloem にあります。
❹ OK をクリックしてウィンドウを閉じる。



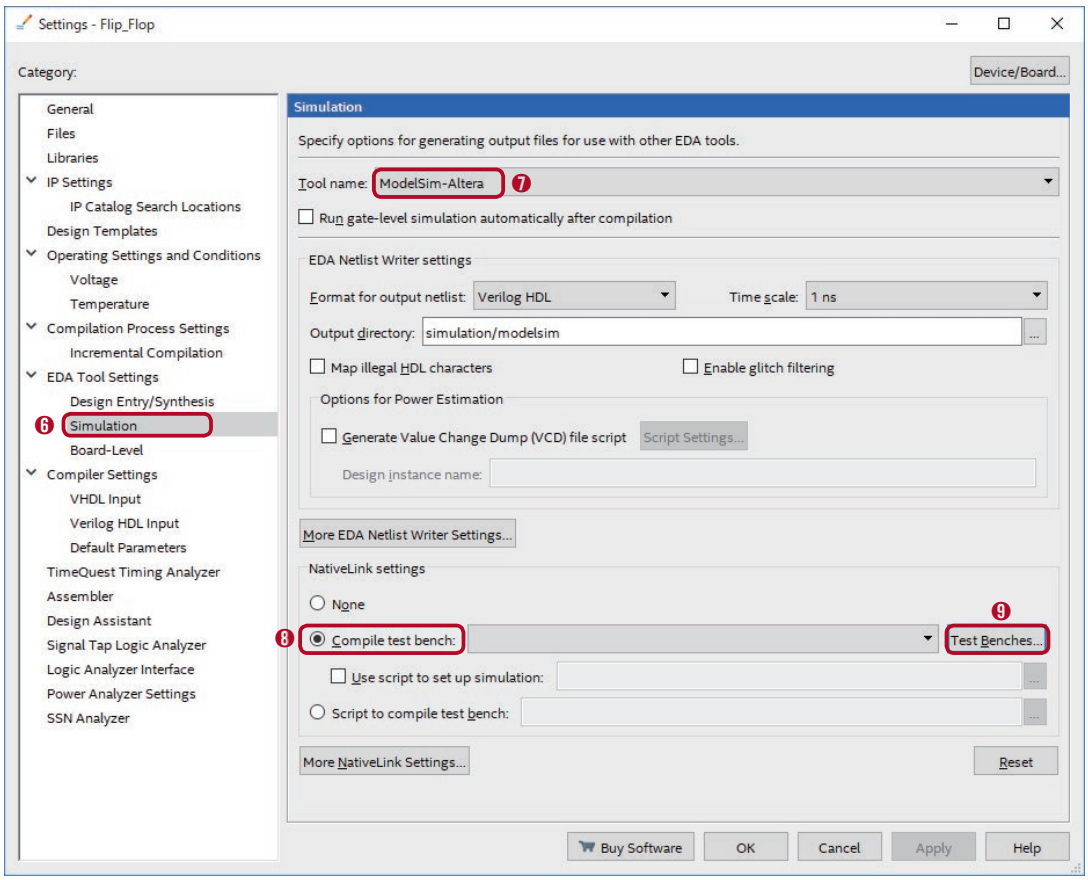

次のページへ

この設定は Quartus Prime の設定なので、全プロジェクトに反映されます。
プロジェクトごとに設定する必要はありません。

# ModelSim を用いた回路シミュレーション

❺ ①②のいずれかの方法で Settings ウィンドウを開き、
① Quartus Prime のメニューから Assignment ＞ Settings を選択。
② ツールバーの「Settings」ボタン をクリック。

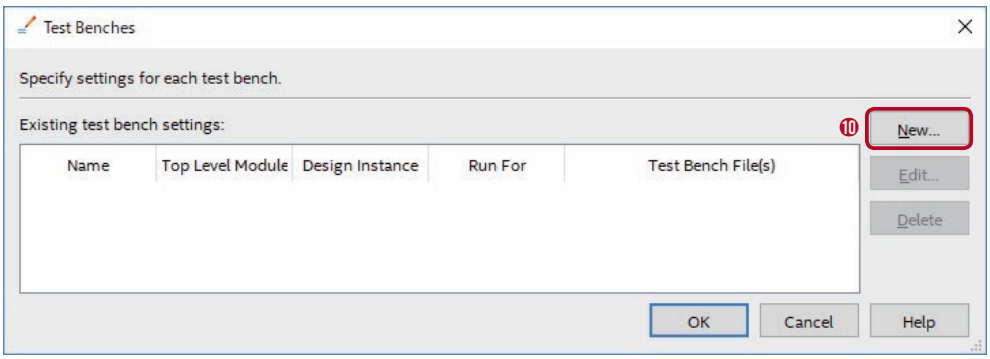
❻ 表示される左側の Category にて EDA Tool Settings の Simlation をクリック。
❼ Tool name で ModelSim-Altera を選択。
❽ NativeLink settings 内の Compile test bench をチェック。
❾ Test Benches をクリック。



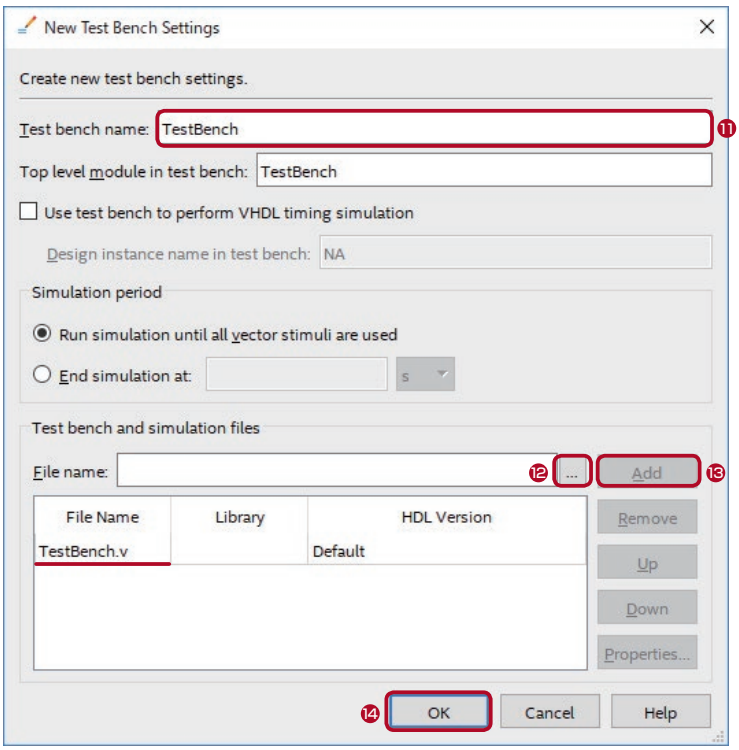

次のページへ

# ModelSim を用いた回路シミュレーション

⑩ New をクリック。



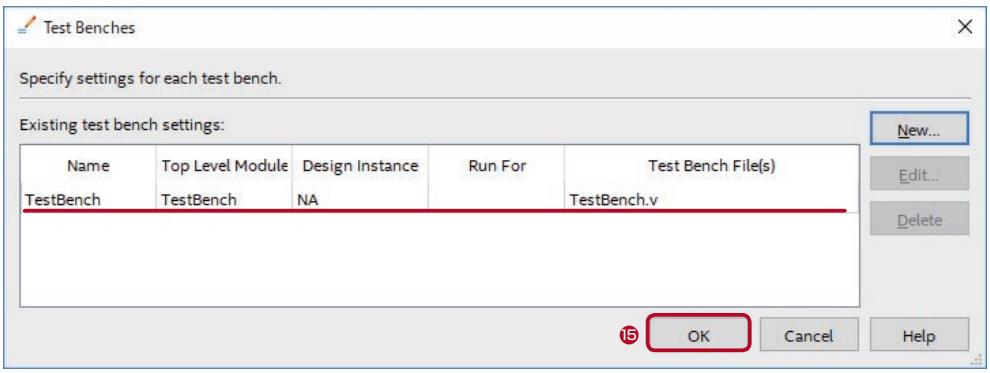

⑪ 表示されたウインドウの Test bench name に先ほど作成したテストベンチのモジュール名を指定。
今回の場合「TestBench」。
⑫ Test bench files の欄は File name の…をクリックして先ほど作成した「TestBench.v」を選択。
⑬ Add をクリックすると Test Benches ウィンドウに作成したファイルが表示される。
⑭ OK をクリックして New Test Bench Settings ウィンドウを閉じる。



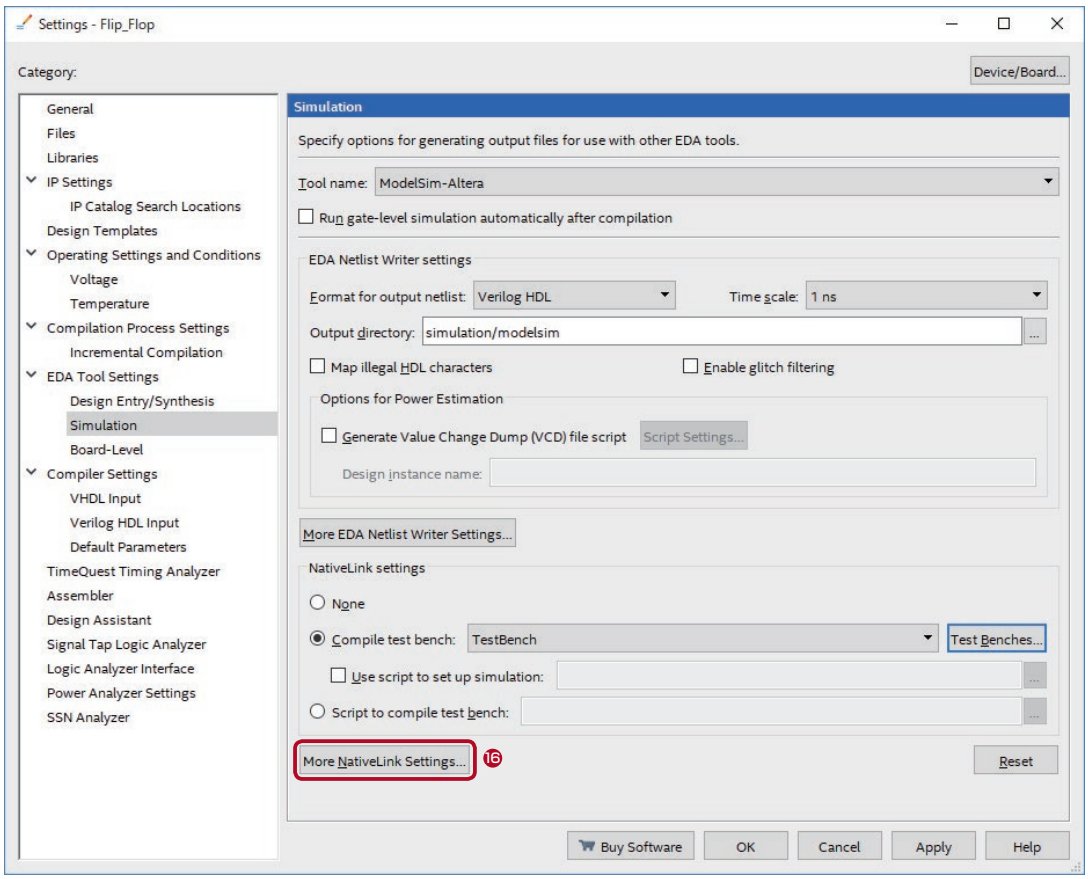

次のページへ

# ModelSim を用いた回路シミュレーション

⓯ OK をクリックして Test Benchs ウィンドウを閉じる。



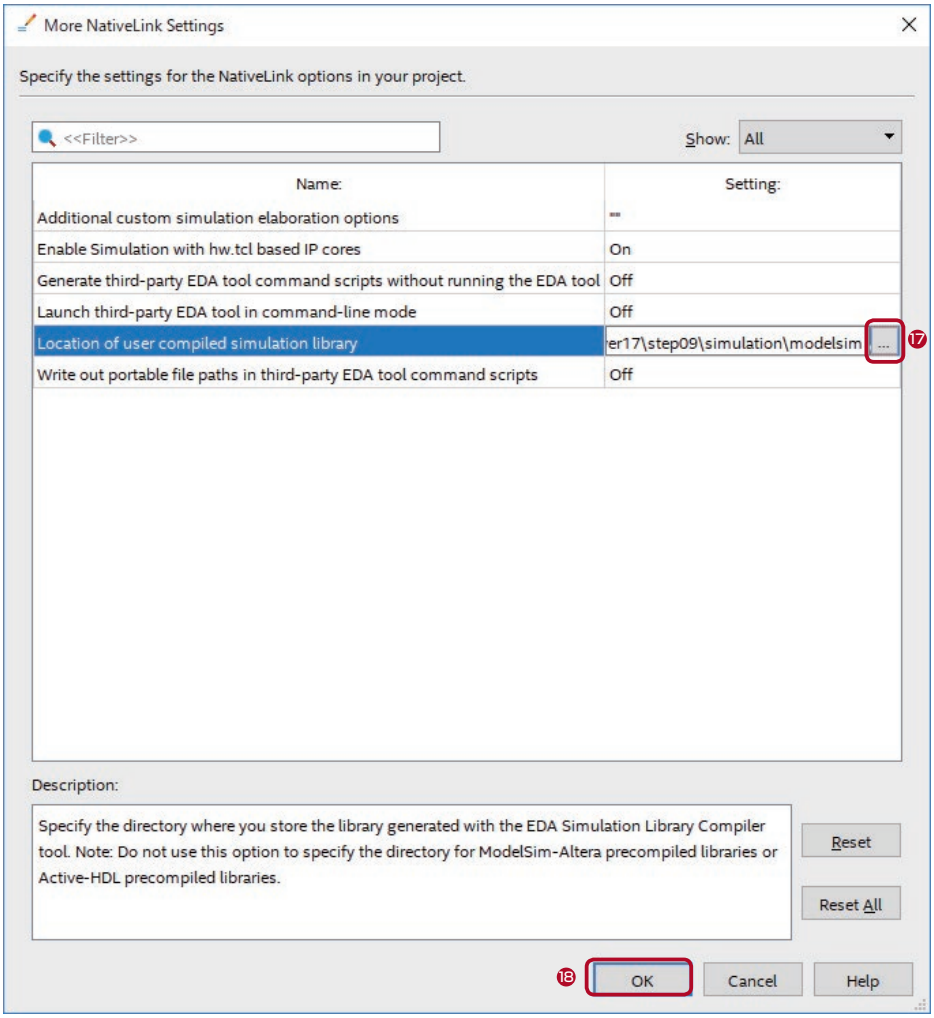

⓰ More NativeLink Settings をクリック。




次のページへ

# ModelSim を用いた回路シミュレーション

⑰ Location of user compiled simulation library 行の Setting の…をクリックしてフォルダを選択。
プロジェクトフォルダ内の \simulation\modelsim フォルダを指定。
もし simulation フォルダが作成されていない場合、一旦 Settings を閉じてコンパイルを行う。
⑱ OK をクリックして More NativeLink Settings ウィンドウを閉じる。

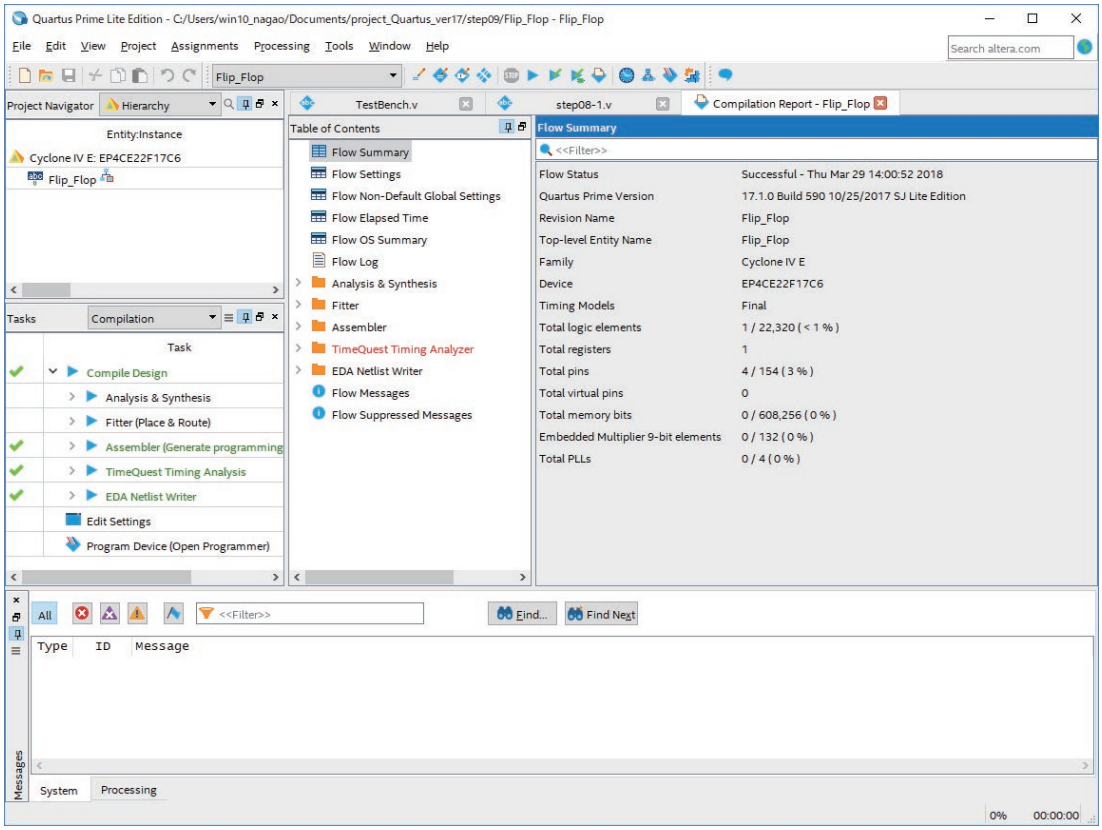



⑲ OK をクリックして Settings ウィンドウを閉じる。

# ModelSim を用いた回路シミュレーション

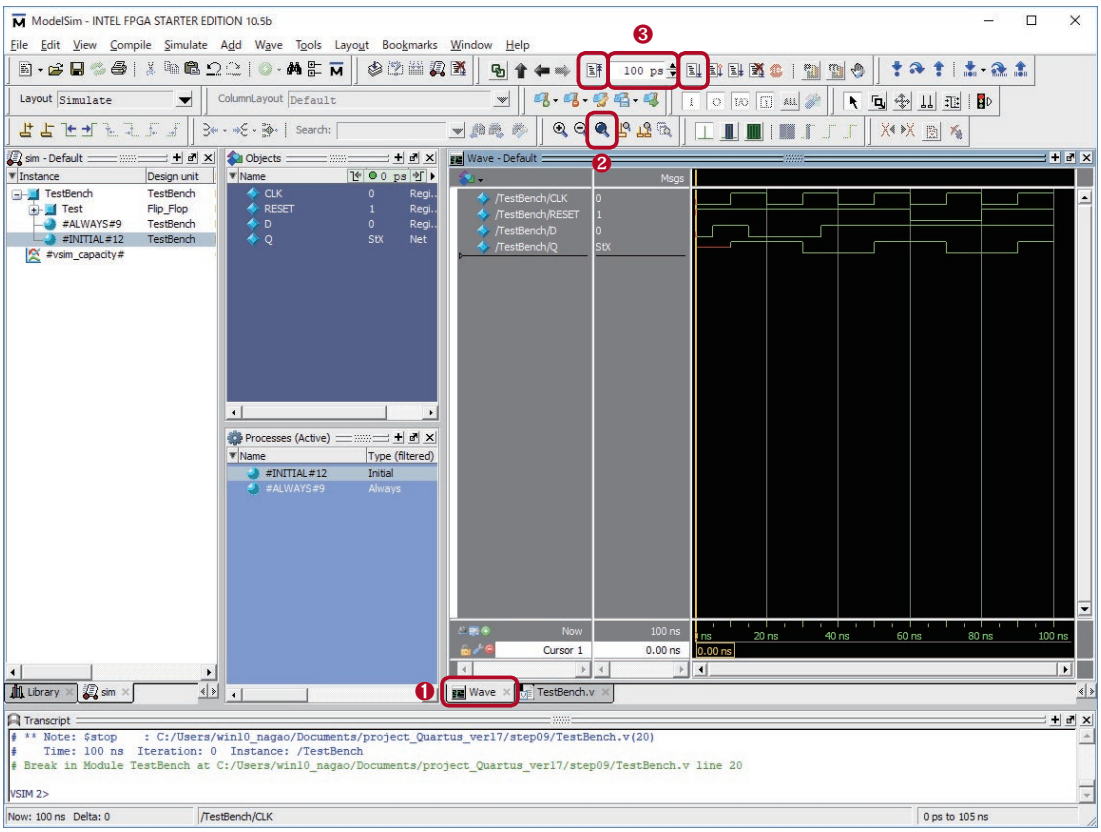
## 6. シミュレーションの実行

❶ シミュレーションの設定が完了したら再度コンパイルを行います。
❷ Quartus Prime のメニューから Tools ＞ Run Simulation Tools ＞ RTL Simlation を選択。
すると ModelSim-Altera が立ちあがりしばらくするとシミュレーションが完了します。



# ModelSim を用いた回路シミュレーション

❶ Wave タブをクリックすると、波形でシミュレーションを確認できます。
❷ Zoom Fill などを使って表示サイズを調節し、波形を確認します。
❸ Restart をクリックしてシミュレーションしたい時間を入力し 100 ps 、Run で少しずつ実行することもできます。




シミュレーションが実行されない場合はテストベンチもしくは設計した回路に何らかの問題があります。
ModelSim 画面下にある Transcript にエラーメッセージが出ていないか確認してください。

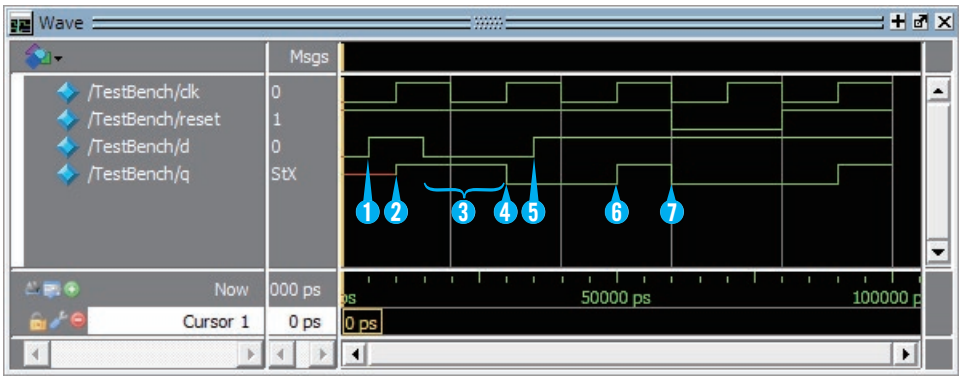
課題 08-2 も同様にシミュレーションしてみましょう。

# ModelSim を用いた回路シミュレーション

## 7. シミュレーション結果

シミュレーションの結果を見てみると、❶入力 D を 1 にすると ❷ CLK の立ち上がりで出力 Q の値は 1 になります。❸その後入力 D が 0 になっても出力 Q は 1 のままです。❹ CLK が再び立ち上がると出力 Q は 0 になります。この間、出力 Q は 1 が保持されていました。
RESET 信号の動作を確認するため出力 Q の値を 1 にします。（❺入力 D を 1 にすると ❻ CLK の立ち上がりで出力 Q の値は 1 になります）❼このとき RESET 信号が立ち下がると Q の値も 0 になっているのが分かります。

### 非同期リセット



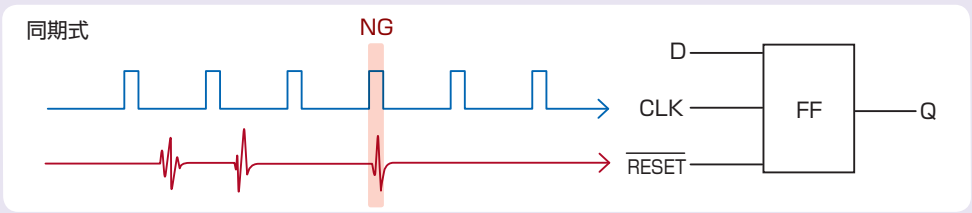

### 同期リセット


一見、非同期リセットと同様に見えますが、RESET 信号があったときの動作に違いがあります。先ほどは RESET 信号があった直後に Q の値が 0 になりましたが、今回は RESET 信号が入力されても Q の値は 0 にならず、次に CLK が立ち上がったとき Q の値が 0 になっています。
同期リセットでは、出力 Q の変化が必ず CLK の立ち上がりで起こります。つまり、同期式で設計すると 1 つのクロックですべての回路が駆動するのです。これを 単相同期回路 といいます。

# ModelSim を用いた回路シミュレーション

## 同期回路はノイズに強い？

同期式では、入力信号にノイズが載ったとしても、ノイズのタイミングがクロックのタイミングと一致しなければ誤動作にはなりません。たまたまクロックのタイミングと一致したノイズだけが、誤動作につながります。





今回の例で RESET 信号にノイズが載ってしまった場合、非同期リセットは誤動作して保持している値が勝手にリセットされてしまいます。しかし、同期リセットはクロック信号のタイミングにノイズが一致しない限り誤動作しないため、非同期回路に比べてノイズに強いといえます。

しかし、同期式がすべてにおいて優れているという訳ではありません。同期式、非同期式それぞれメリットデメリットがあり、回路規模や要求される仕様によって考慮する必要があります。

| 同期式 | |
|---|---|
| メリット | タイミング設計が容易.<br>比較的ノイズに強い |
| デメリット | 生成される回路規模が肥大化する.<br>消費電流が大きくなる. |

⟷

| 非同期式 | |
|---|---|
| デメリット | 各ゲートでの遅延時間が蓄積していくため、回路規模が大きくなるとタイミング設計が難しくなる. |
| メリット | 生成される回路規模が小さくなる.<br>消費電流も一般に小さい. |