

Quartus[®] Prime はじめてガイド Quartus Prime 簡易チュートリアル

Quartus Prime はじめてガイド

Quartus Prime 簡易チュートリアル

目次

1. はじめに.....	3
1-1. 必要な環境.....	3
1-2. サンプル・デザインの概要.....	4
2. Quartus Prime の設計.....	5
2-1. 論理回路を設計する.....	5
2-1-1. プロジェクトを作成する.....	6
2-1-2. 論理回路を作成する.....	9
2-1-3. Analysis and Elaboration の実行.....	21
2-2. 論理シミュレーションを実行する.....	22
2-3. 制約の設定をする.....	27
2-3-1. ユーザ I/O ピンのアサイン.....	28
2-3-2. タイミング制約.....	30
2-4. コンパイルを実行する.....	31
2-5. タイミングを検証する.....	32
2-6. プログラミング.....	33
2-6-1. sof ファイルを書き込む.....	34
2-6-2. コンフィギュレーション・デバイスに書き込む.....	37
2-6-2-1. プログラミング・ファイルを作成する.....	37
2-6-2-2. プログラミングする.....	39
2-7. 実機検証.....	43
3. ステップアップ.....	45
改版履歴.....	46

1. はじめに

この資料は、インテル® Quartus® Prime 開発ソフトウェアを使用して、デジタル論理回路の作成からインテル FPGA の搭載された評価キットや開発キットを動作させるまでをチュートリアル形式にまとめたものです。初めて FPGA 設計をするユーザが、開発ソフトウェアの基本的な操作フローを簡易的に理解することができます。

Notes: ・ 本チュートリアルは Quartus® Prime 18.0 以降の環境でご利用になることを推奨します。同様に、ModelSim® - Intel® FPGA Edition (以下、ModelSim - Intel FPGA Starter Edition を含む)は、ご利用になる Quartus Prime バージョンと同時提供されているバージョンをご利用ください。

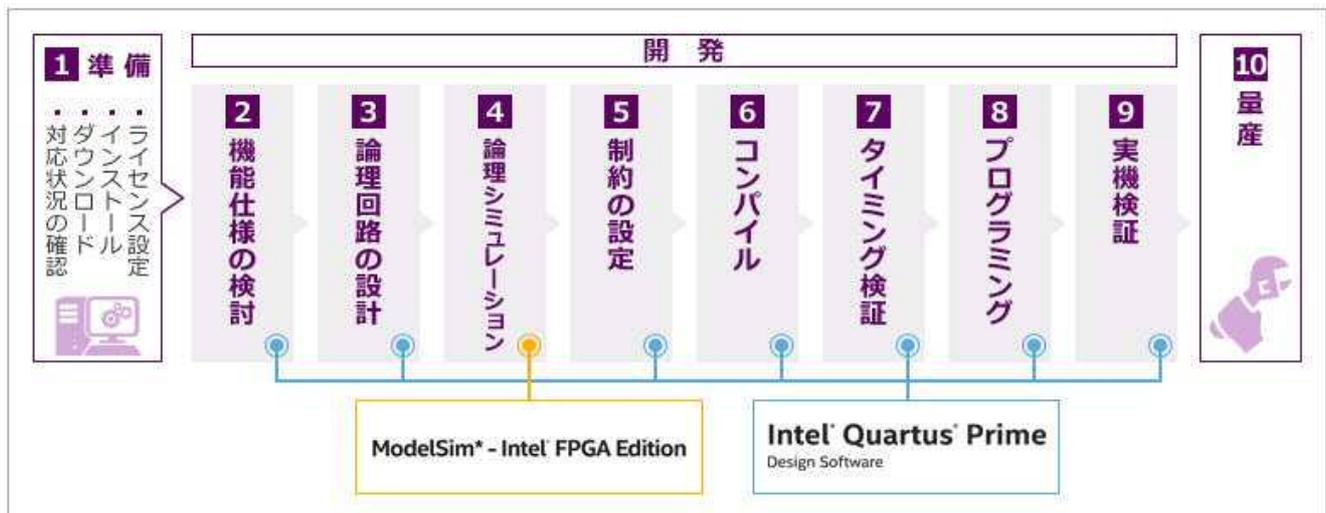
・ Quartus Prime および ModelSim - Intel FPGA Edition は、タイプ(Edition)により有償のライセンスが必要です。詳細は下記ページをご覧ください。

[ほんとのほんとの導入編 その2. FPGA 開発をはじめるために準備するもの](#)

下図は、FPGA の一般的な開発フローを示しています。

インテル FPGA を開発する場合、開発フローの [2]~[9] は Quartus Prime を使用し、そのうち開発フローの [4] では ModelSim - Intel FPGA Edition などのインテル FPGA をサポートする EDA シミュレーション・ツールを使用します。

このチュートリアルでは、開発フローの [3]~[8] を体験することができます。



なお、各工程の操作マニュアルは、下記ページよりご利用いただけます。

[FPGA/CPLD の一般的な開発フロー／トップページ](#)

1-1. 必要な環境

本チュートリアルを実施するためには、以下の環境が必要です。

- Quartus Prime (Lite あるいは Standard Edition のいずれか) 18.0 以降、および ModelSim - Intel FPGA Edition をインストールしたパソコン
- 使用可能な Windows オペレーティング・システムは、下記の対応表でご確認ください。

[「Quartus Prime - サポート Windows OS 対応表」](#)

Note: ・ Linux のディストリビューションについては、以下のページをご覧ください。

[Operating System Support](#)

- 各種ソフトウェアの[ダウンロードについてはこちら](#)、[インストールについてはこちら](#)をご覧ください。
- インテル FPGA ダウンロード・ケーブル II (旧 USB-Blaster™ II)※
 - インテル FPGA ダウンロード・ケーブル (旧 USB-Blaster)※、または インテル FPGA イーサネット・ケーブル (旧 EthernetBlaster II)でも可



Notes: ・ ご利用になるインテル製の FPGA 評価キットや開発キットにより、上記ダウンロード・ケーブルと同等の機能が内蔵されているボード(例: On-Board USB Blaster II 搭載タイプ)もあります。その場合は、これらのダウンロード・ケーブルは不要です。

- ・ 上記のダウンロード・ケーブルおよびオンボード・タイプをパソコンではじめて利用するには、専用ドライバのインストールが必要です。インストールの手順は、下記 TIPS をご参考ください。

[【TIPS】USB-Blaster™ II のドライバをインストールしてみよう](#)

[【TIPS】USB-Blaster™ のドライバをインストールしてみよう](#)

- インテル FPGA の搭載された開発キット
 - チュートリアルの対象ボードは下表のとおりです

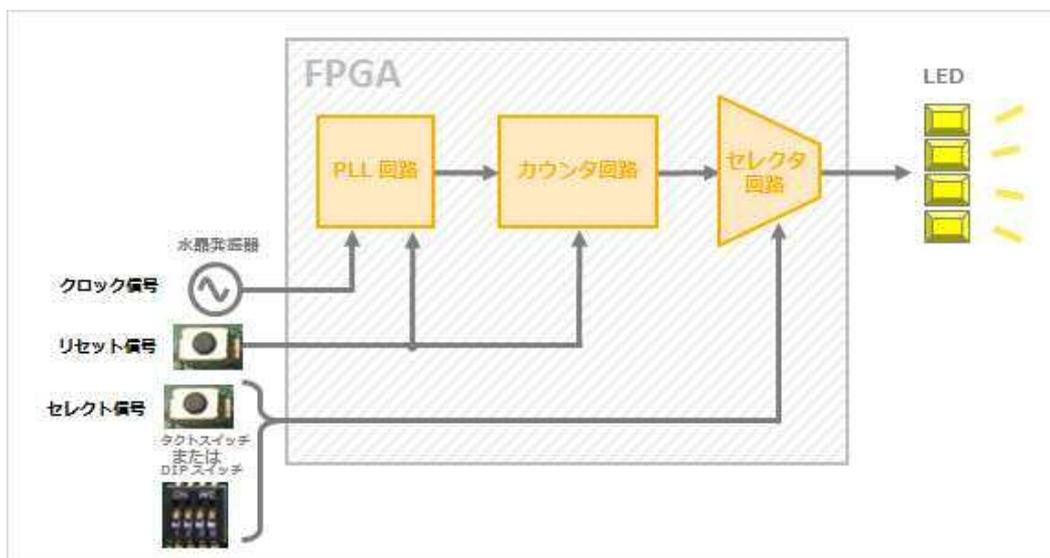
Board	Device Family	Available devices	ダウンロード・ケーブル
MAX® 10 FPGA Evaluation Kit	MAX 10	10M08SAE144C8G	必要
Cyclone® 10 LP Evaluation Kit	Cyclone 10 LP	10CL025YU256I7G	不要 On-Board USB-Blaster II
Beryll Cyclone V GX Base Board	Cyclone V	5CGXFC4C6F27C7	不要 On-Board USB-Blaster

1-2. サンプル・デザインの概要

このチュートリアルでは、デジタル論理回路をハードウェア記述言語(以下 HDL)で設計します。

開発ボード上の LED を、入力ボタンにより点滅する速度を制御するの論理回路を作成します。ブロック図は以下のとおりです。

この論理回路は、3つの機能で構成します。



- ・ PLL (Phase-Locked Loop) 回路

外部から入力されたクロック信号の周波数を逡倍や分周、およびクロックの位相をシフトすることが可能な回路です。

- ・ カウンタ回路

入力されたクロックの数を数えて処理を行う回路です。

- ・ セレクタ回路(マルチプレクサ回路)

複数の入力のうちのどれか 1 つを選んで取り出す回路です。

Note: ・ HDL 設計の経験がないユーザは、以下もご活用ください。

[はじめてみよう！ Verilog-HDL <演習問題つき>](#)

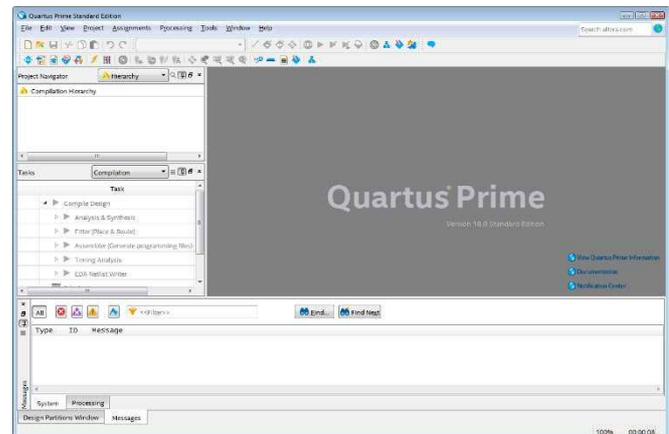
[はじめてみよう！ VHDL <演習問題つき>](#)

2. Quartus Prime の設計

Quartus Prime を起動します。

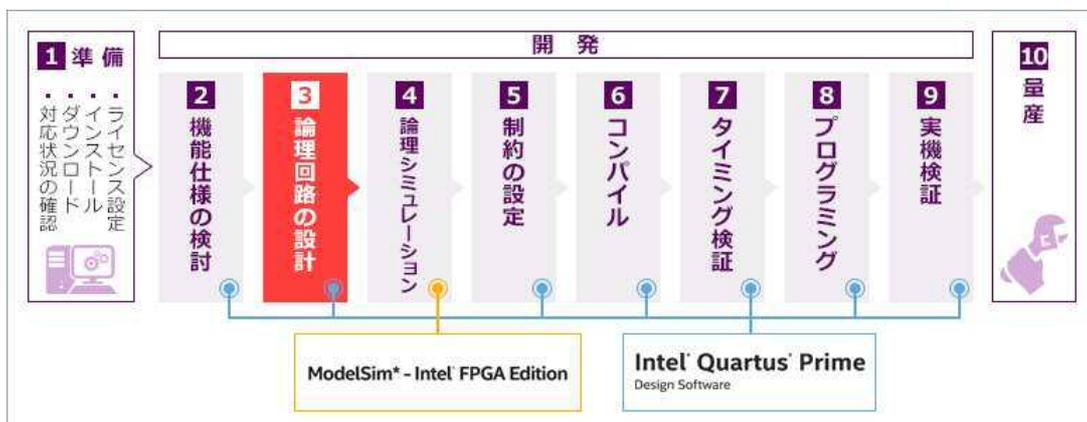
Windows OS の場合、インストール時に作成したデスクトップ上のショートカット・アイコンをダブルクリック、もしくは[スタート]メニュー (Windows 8.1 の場合はスタート画面) から Quartus Prime 開発ソフトウェアを起動させます。

Linux OS の場合には、quartus コマンドにより起動させます。



2-1. 論理回路を設計する

1 つの FPGA に構築する論理回路を作成します。(この作業は、開発フローの [3] にあたります。)

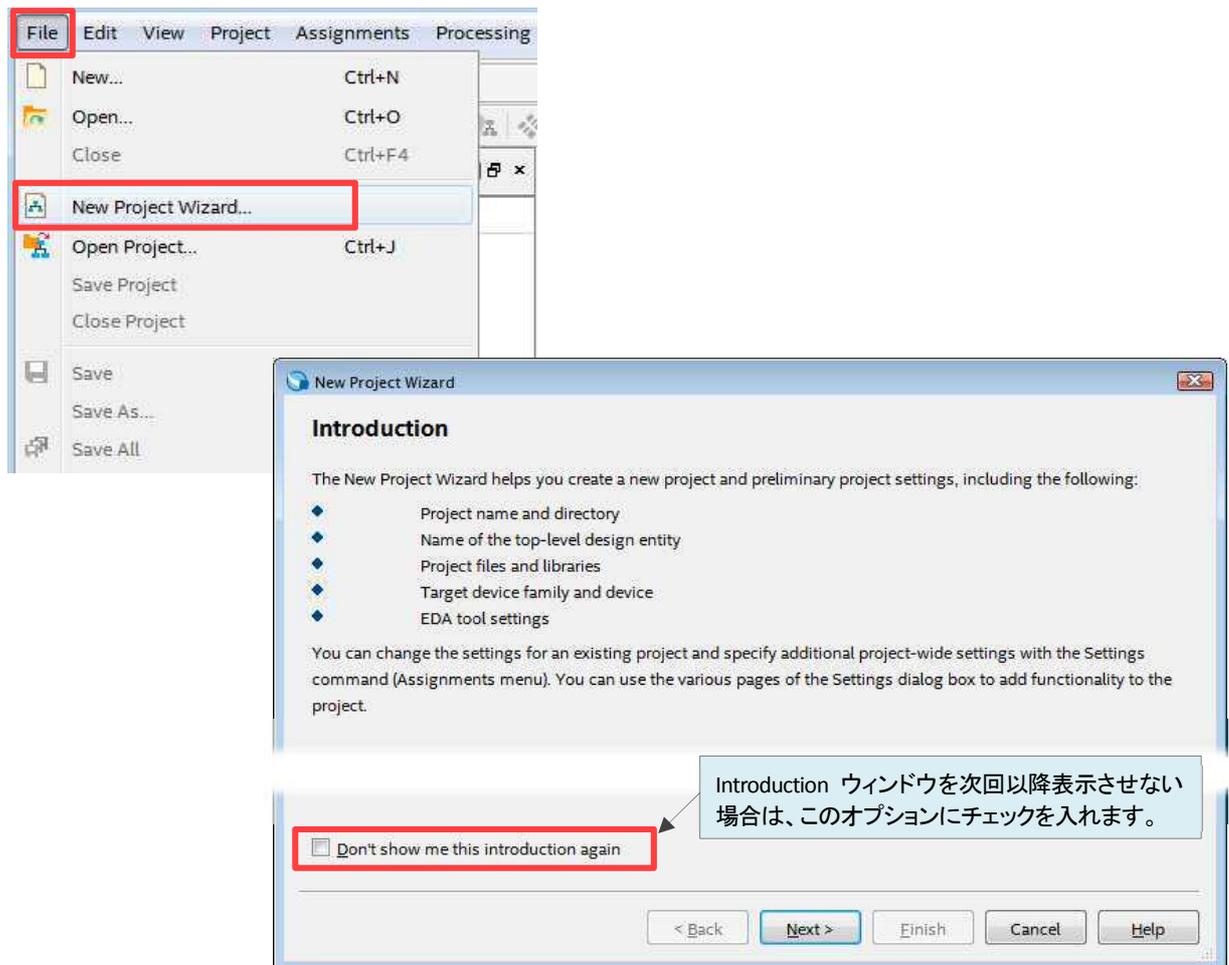


1 つの FPGA に対して開発を始めるには、Quartus Prime で 1 つのプロジェクトを構築する必要があります。そのプロジェクトに対して、HDL やスキマティックで論理回路を作成し、登録を行います。

2-1-1. プロジェクトを作成する

論理回路を作成するにあたり、プロジェクトを作成します。

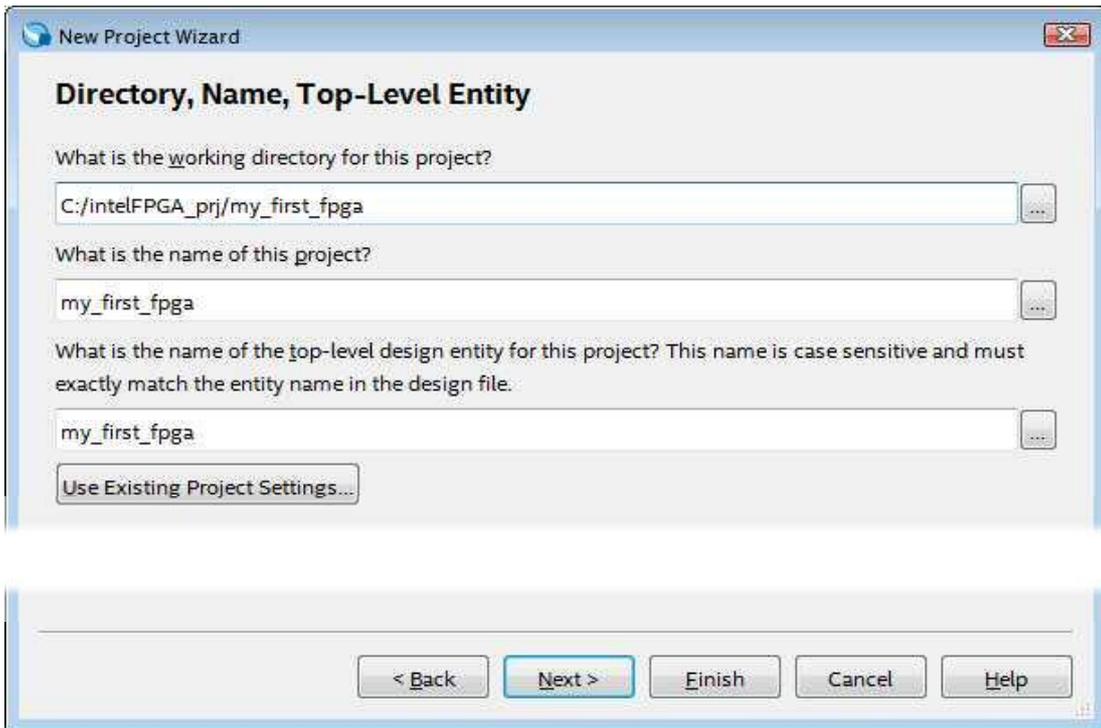
1. File メニュー ➤ New Project Wizard を選択します。New Project Wizard ダイアログボックスが起動し、初回は Introduction ページが表示されますので、Next ボタンをクリックし、次へ進みます。



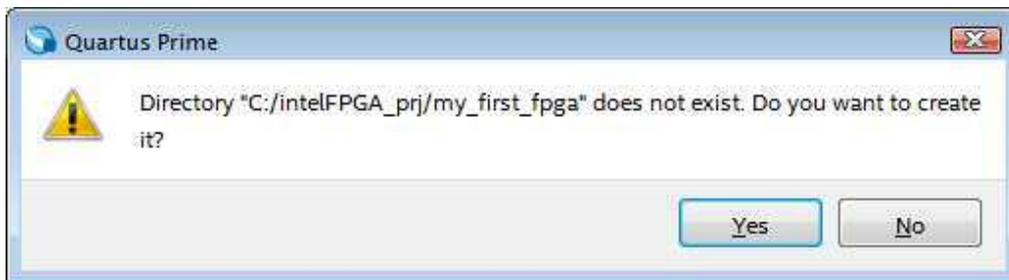
2. Directory, Name, Top-Level Entity ページでは、以下の情報を登録し、Next ボタンをクリックします。

Directory, Name, Top-Level Entity ページ	
What is the working directory for this project?	C:\¥intelFPGA_prj¥my_first_fpga
What is the name of this project?	my_first_fpga
What is the name of the top-level design entity for this project? This name is ... (省略)	my_first_fpga

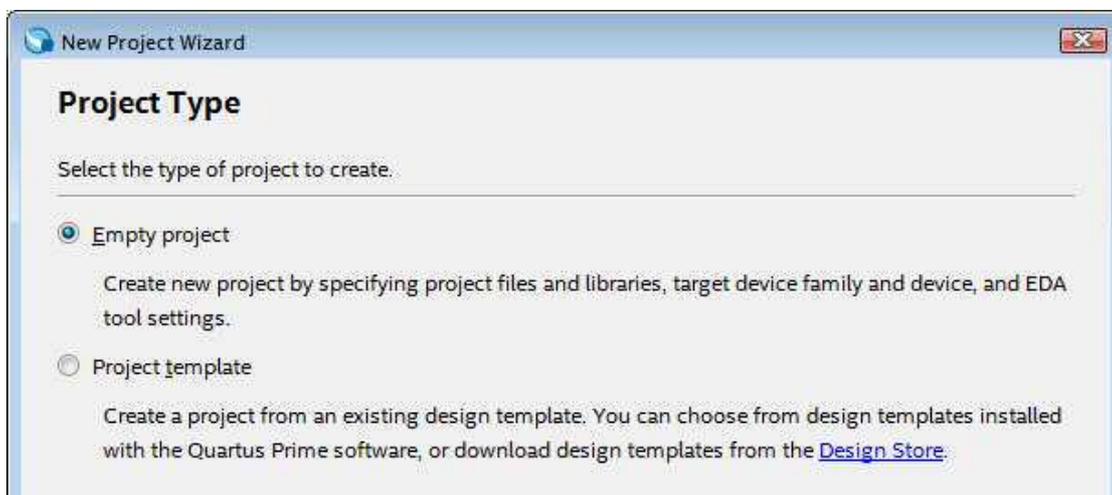
Note: ・ 作業ディレクトリのパス、プロジェクト名、ファイル名は、半角・英数字のみご利用ください。全角やスペースの使用はできません。-



パソコンにあらかじめ C:\intelFPGA_prj\my_first_fpga ディレクトリを作成していない場合は、入力したディレクトリの作成を促すメッセージ(下図)が表示されます。Yes を選択し、ワーキング・ディレクトリを作成します。



3. Project Type ページでは、Empty project を選択し、Next ボタンをクリックします。

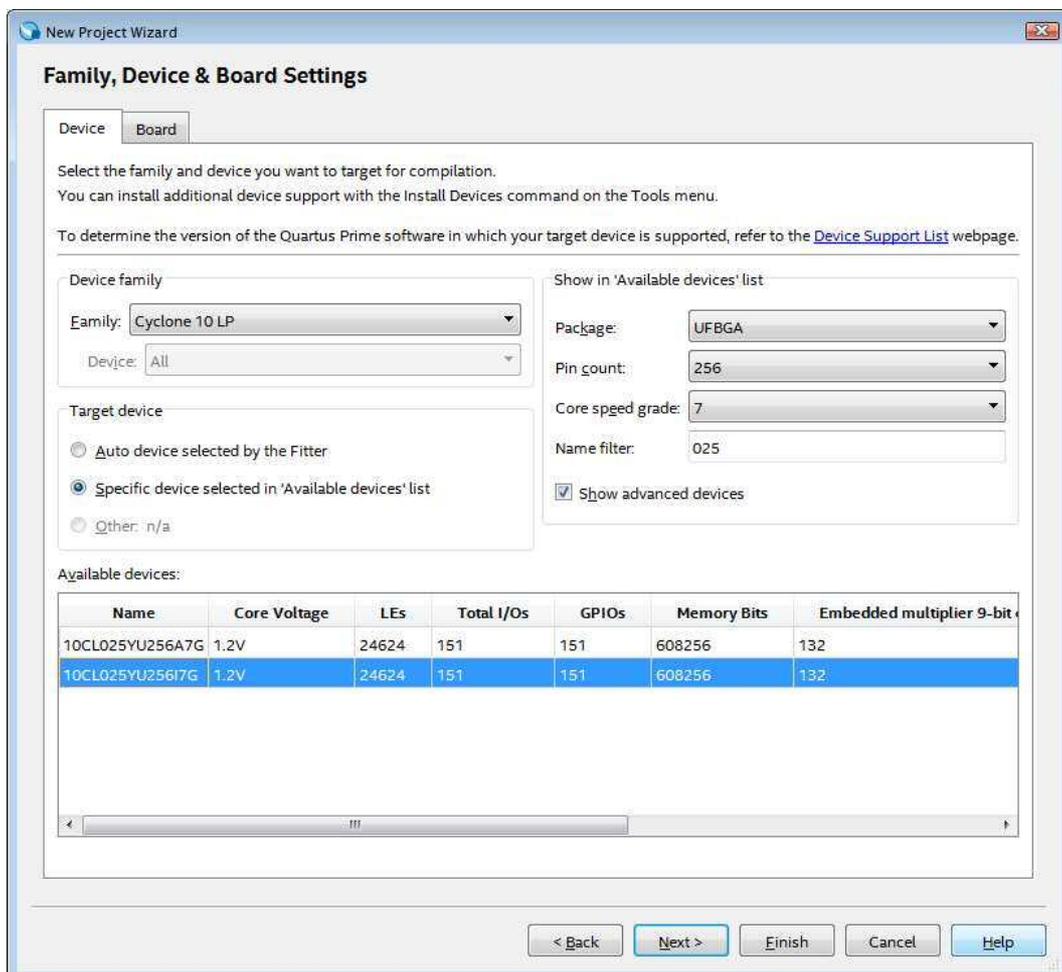


4. Add Files ページは、何も選択せずに Next ボタンをクリックします。
5. Family, Device & Board Settings ページでは、使用する開発キットに搭載されているインテル FPGA の型番を選択します。

Board	Device Family	Package	Pin count	Core speed grade	Available devices
MAX® 10 FPGA Evaluation Kit	MAX 10	EQFP	144	8	10M08SAE144C8G
Cyclone® 10 LP Evaluation Kit	Cyclone 10 LP	UFBGA	256	7	10CL025YU256I7G
Beryll Cyclone V GX Base Board	Cyclone V	FPGA	672	7_H6	5CGXFC4C6F27C7

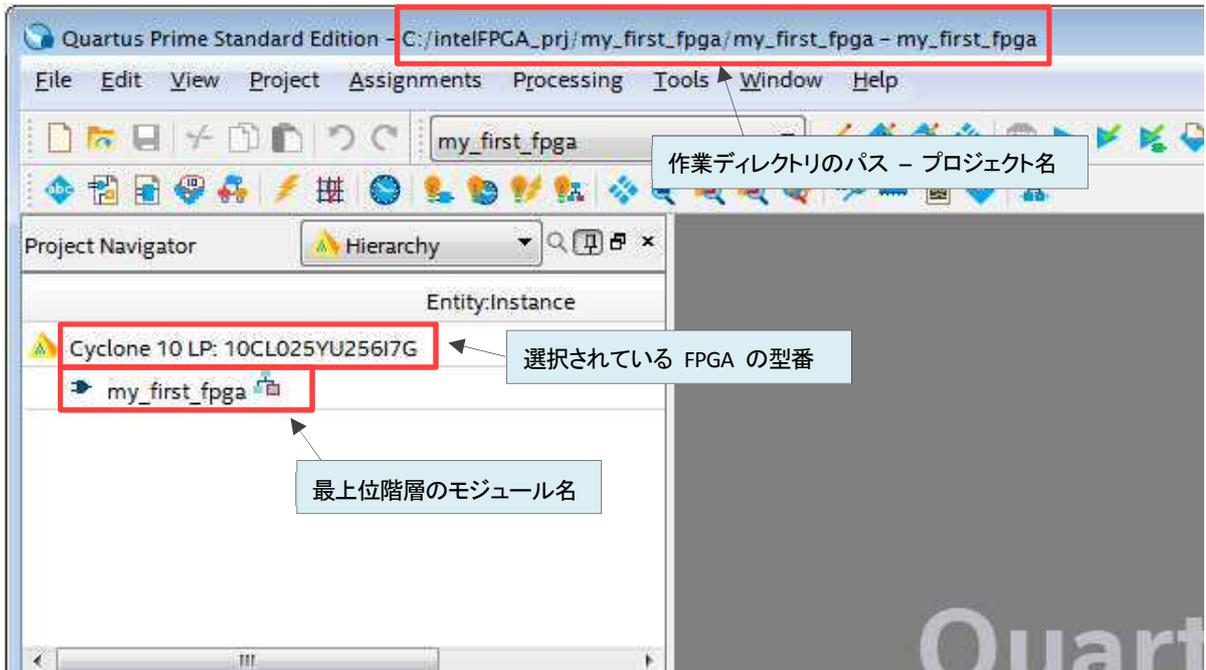
- ① Family (Device family エリア内)プルダウン・リストから、FPGA の デバイス・ファミリ を選択します。
- ② Show in 'Available devices' list エリアにおいて、上表を参考に Package、Pin count、Core speed grade の項目を選択します。選択した項目がフィルタ条件となり、Available devices 欄に該当する型番のみが表示されます。
- ③ Available devices 欄から対象の型番をクリックで指定し、ハイライトさせます。

Note: ・ 条件を入力しても Available devices 覧に多くの型番が検出され対象の FPGA が見つけにくいときは、Show in 'Available devices' list エリアの Name filter 欄にキーワードを入力して、さらに絞り込んでください。例えば、Cyclone 10 LP Evaluation Kit (10CL025YU256I7G) の場合、Name filter 欄に "025" を入力するとよいでしょう。



6. Finish ボタンをクリックします。

画面のツールバー、および Project Navigator ウィンドウに設定した情報が表示されることを確認します。



関連情報: 下記ドキュメントをご参考ください。

[「Quartus Prime はじめてガイド - プロジェクトの作成方法」](#)

2-1-2. 論理回路を作成する

作成したプロジェクトに、1-2. サンプル・デザインの概要 で示した論理回路(以下、デザイン)を作成します。

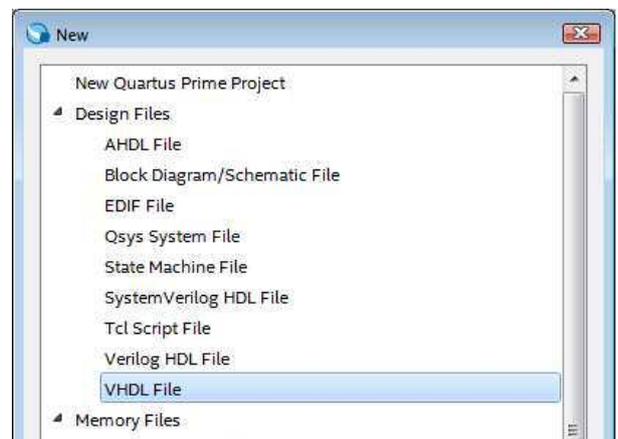
このチュートリアルでは、カウンタ回路およびセクタ回路は VHDL でコードを記述し、PLL 回路は IP Catalog という入力手法で設計します。また、この 3 つのモジュールを接続する最上位階層デザインは、VHDL で記述します。

■ カウンタ回路を設計する

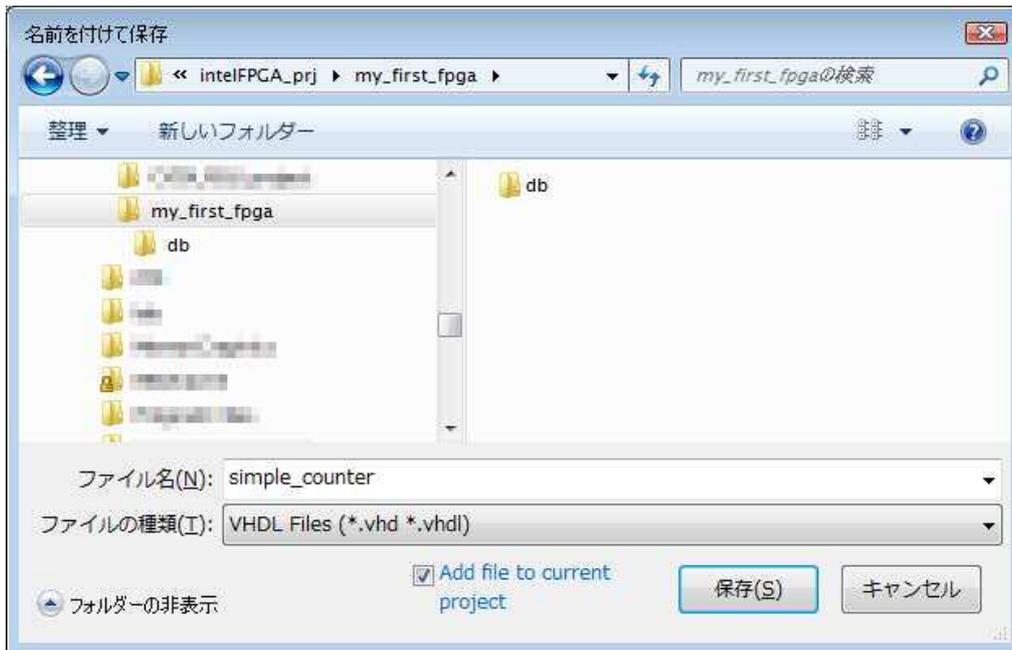
VHDL で作成します。

1. File メニュー ➤ New ➤ VHDL File (Design Files 内) をダブルクリックで選択します。

Vhdl1.vhd の新規シートが開きます。



2. File メニュー ➤ Save As... により、これから作成するデザイン・ファイルに名前を付け、保存するディレクトリを指定します。
 - ・ 保存するディレクトリ: C:\intelFPGA_prj\my_first_fpga
 - ・ ファイル名: simple_counter
 - ・ ファイルの種類: VHDL Files (*.vhd *.vhd)
 - ・ Add file to current project オプション = On



3. 次の HDL コードを simple_counter.vhd (または ファイル)に入力します。

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

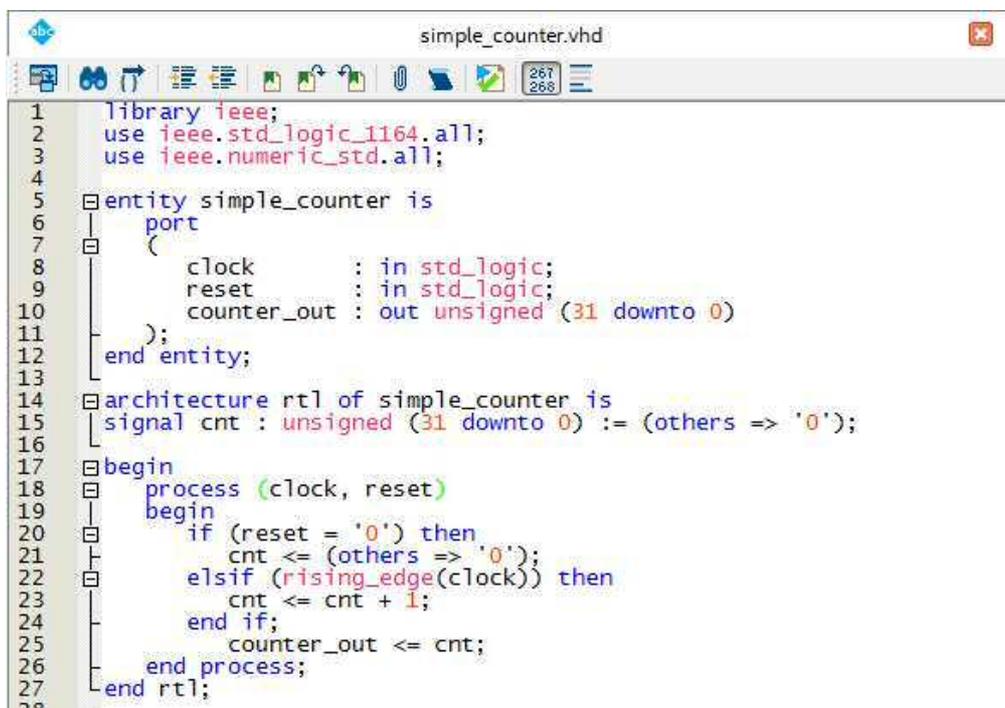
entity simple_counter is
    port
    (
        clock      : in std_logic;
        reset      : in std_logic;
        counter_out : out unsigned (31 downto 0)
    );
end entity;

architecture rtl of simple_counter is
    signal cnt : unsigned (31 downto 0) := (others => '0');

```

```

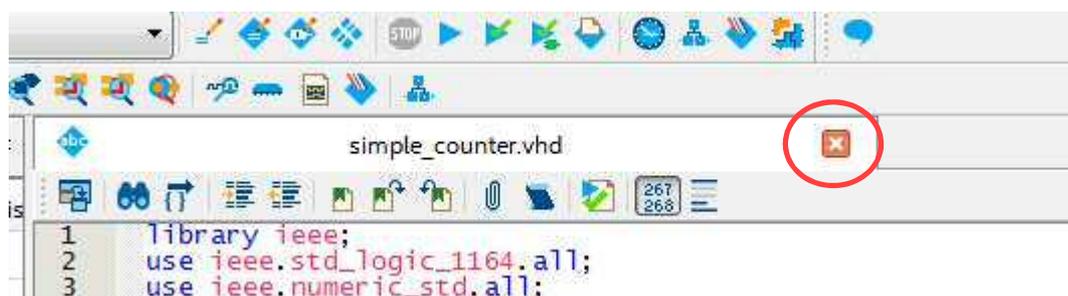
begin
  process (clock, reset)
  begin
    if (reset = '0') then
      cnt <= (others => '0');
    elsif (rising_edge(clock)) then
      cnt <= cnt + 1;
    end if;
    counter_out <= cnt;
  end process;
end rtl;
    
```



```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  entity simple_counter is
6  port
7  (
8      clock      : in std_logic;
9      reset      : in std_logic;
10     counter_out : out unsigned (31 downto 0)
11 );
12 end entity;
13
14 architecture rtl of simple_counter is
15     signal cnt : unsigned (31 downto 0) := (others => '0');
16
17     begin
18     process (clock, reset)
19     begin
20     if (reset = '0') then
21     cnt <= (others => '0');
22     elsif (rising_edge(clock)) then
23     cnt <= cnt + 1;
24     end if;
25     counter_out <= cnt;
26     end process;
27 end rtl;
28
    
```

4. File メニュー ➤ Save (または Ctrl+S) により simple_counter.vhd を保存します。
5. simple_counter.vhd のタブの端にある Close ボタンをクリックし、ファイルを閉じます。



■ セレクタ回路を設計する

VHDL で作成します。

1. File メニュー ➤ New ➤ VHDL File (Design Files 内) をダブルクリックで選択します。
2. File メニュー ➤ Save As... により、これから作成するデザイン・ファイルに名前を付け、保存するディレクトリを指定します。
 - ・ 保存するディレクトリ: C:\intel\FPGA_prj\my_first_fpga
 - ・ ファイル名: counter_bus_mux
 - ・ ファイルの種類: VHDL Files (*.vhd *.vhd)
 - ・ Add file to current project オプション = On
3. 次の VHDL コードを counter_bus_mux.vhd ファイルに入力します。

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_bus_mux is
    port
    (
        dataa    : in unsigned (3 downto 0);
        datab   : in unsigned (3 downto 0);
        sel      : in std_logic;
        result   : out unsigned (3 downto 0)
    );
end entity;

architecture rtl of counter_bus_mux is
begin
    process (dataa, datab, sel)
    begin
        case (sel) is
            when '1' => result <= dataa;
            when '0' => result <= datab;
            when others => result <= (others => '0');
        end case;
    end process;
end rtl;
    
```

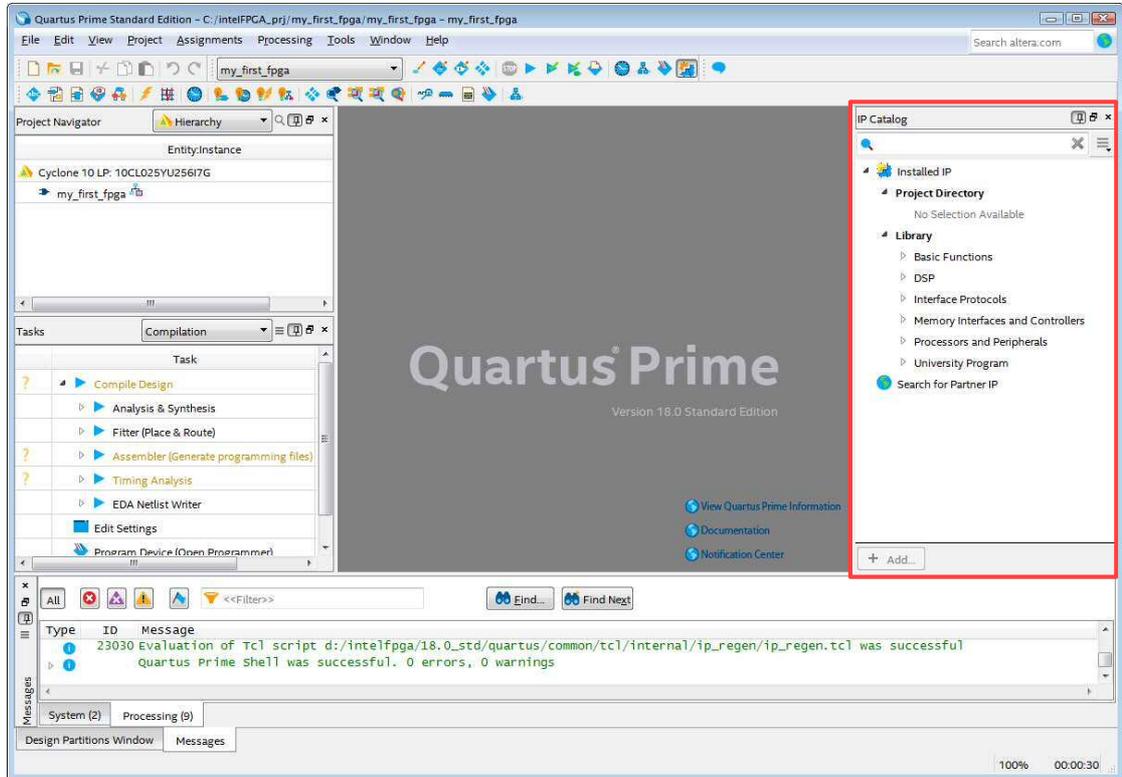
4. File メニュー ➤ Save (または Ctrl+S) により counter_bus_mux.vhd を保存します。
5. counter_bus_mux.vhd のタブの端にある Close ボタンをクリックし、ファイルを閉じます。

■ PLL 回路を設計する

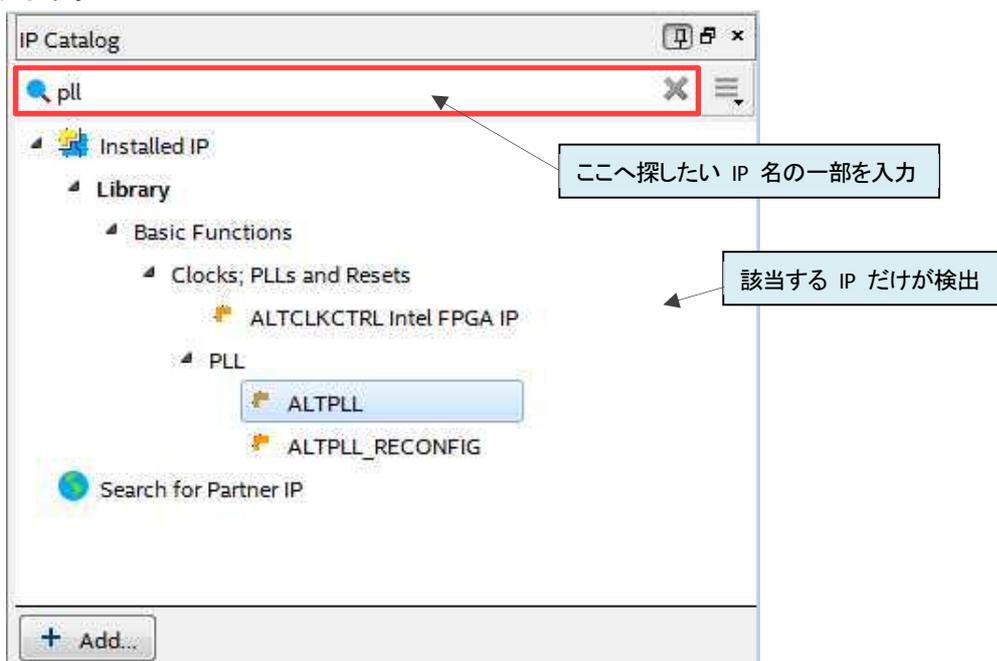
IP Catalog で作成します。

Note: ・ IP Catalog とは、インテル FPGA の IP コアを簡単にカスタマイズしてプロジェクトに組み込むことができるツールです。詳細は、[About the IP Catalog and Parameter Editor](#) をご参考ください。

1. Tools メニュー ➤ IP Catalog を選択し、IP Catalog ウィンドウを表示します。



2. 検索フィールドに *pll* と PLL IP の一部を入力すると、対象 FPGA ファミリで使用できる IP コア名が検索されます。



このチュートリアルでは、使用する開発キットにより IP コアが異なります。下表を参考にしてください。

Board	Device Family	PLL 回路の IP 名
MAX 10 FPGA Evaluation Kit	MAX 10	ALTPLL
Cyclone 10 LP Evaluation Kit	Cyclone 10 LP	ALTPLL
Beryll Cyclone V GX Base Board	Cyclone V	PLL Intel FPGA IP

Note: ・ デバイス・ファミリにより選択できる PLL IP が異なります。詳細は下記ページをご参考ください。

[IP の生成 \(「インテル® FPGA の PLL」より\)](#)

- 該当する PLL IP をハイライト選択し、+ Add ボタンをクリック(または IP コアをダブルクリックでも可)すると、Save IP Variation ダイアログボックスが表示されます。

下記の通り、これから作成する PLL IP の保存ディレクトリ、およびモジュール名を入力し、生成する HDL フォーマットを選択します。

このチュートリアルでは、以下のとおりに設定します。

- IP variation file name: C:\intelFPGA_prj\my_first_fpga\pll
- IP variation file type: VHDL

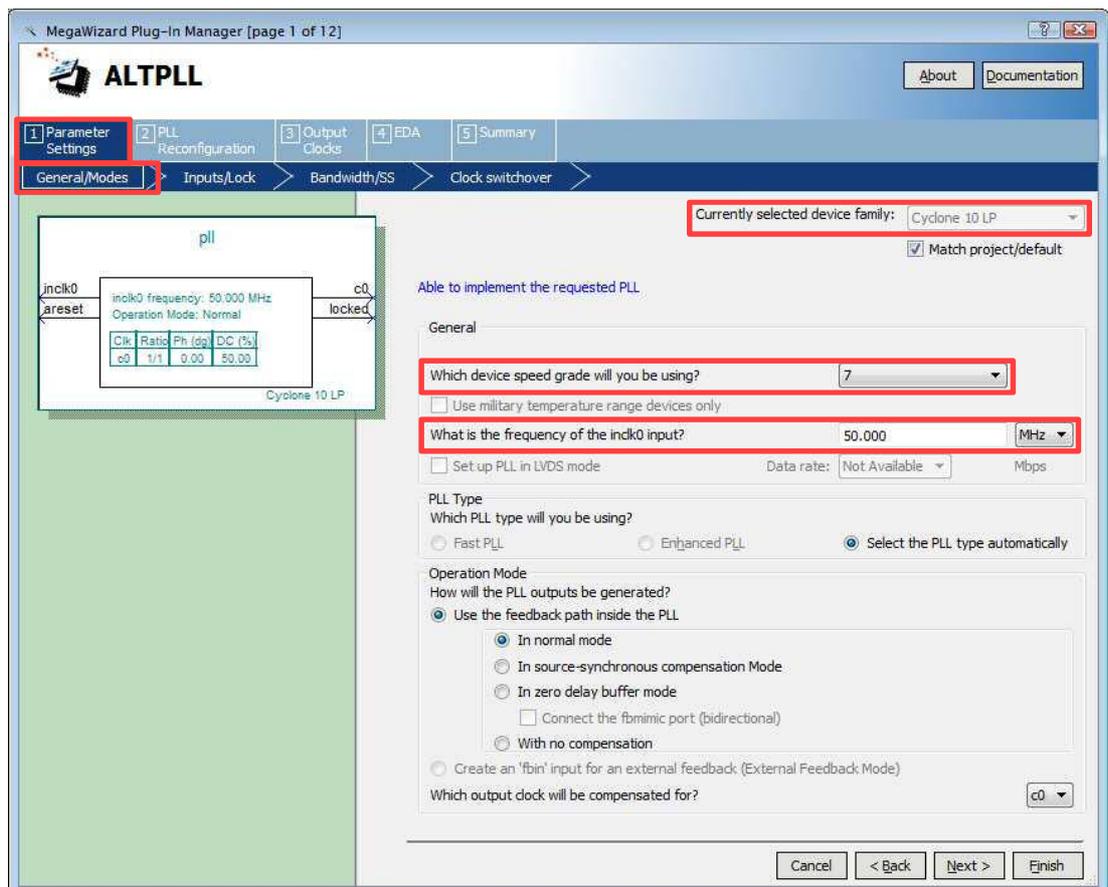


OK ボタンをクリックします。

- PLL IP のパラメータを設定します。PLL のタイプにより項目が異なりますので、使用するボードに応じて設定を行ってください。
 - [ALTPLL \(MAX 10 FPGA Evaluation Kit / Cyclone 10 LP Evaluation Kit\) の設定](#)
 - [PLL Intel FPGA IP \(Beryll Cyclone V GX Base Board\) の設定](#)

- ALTPLL (MAX 10 FPGA Evaluation Kit / Cyclone 10 LP Evaluation Kit) の設定
 - ① MegaWizard Plug-In Manager ウィンドウの **1** Parameter Settings ▶ General/Modes において、以下の設定をします。
 - a. Currently selected device family に使用するボードのデバイス・ファミリが表示されていることを確認します。
 - b. Which device speed grade will you be using? に下表の該当する値を選択します。
 - c. What is the frequency of the inck0 input? に下表の該当する値を入力し、単位を選択します。
 - d. Next ボタンをクリックします。

Parameter Settings ▶ General/Modes			
Board	Which device speed grade will you be using?	What is the frequency of the inck0 input?	その他の設定
MAX 10 FPGA Evaluation Kit	8	50 MHz	デフォルト
Cyclone 10 LP Evaluation Kit	7	50 MHz	デフォルト

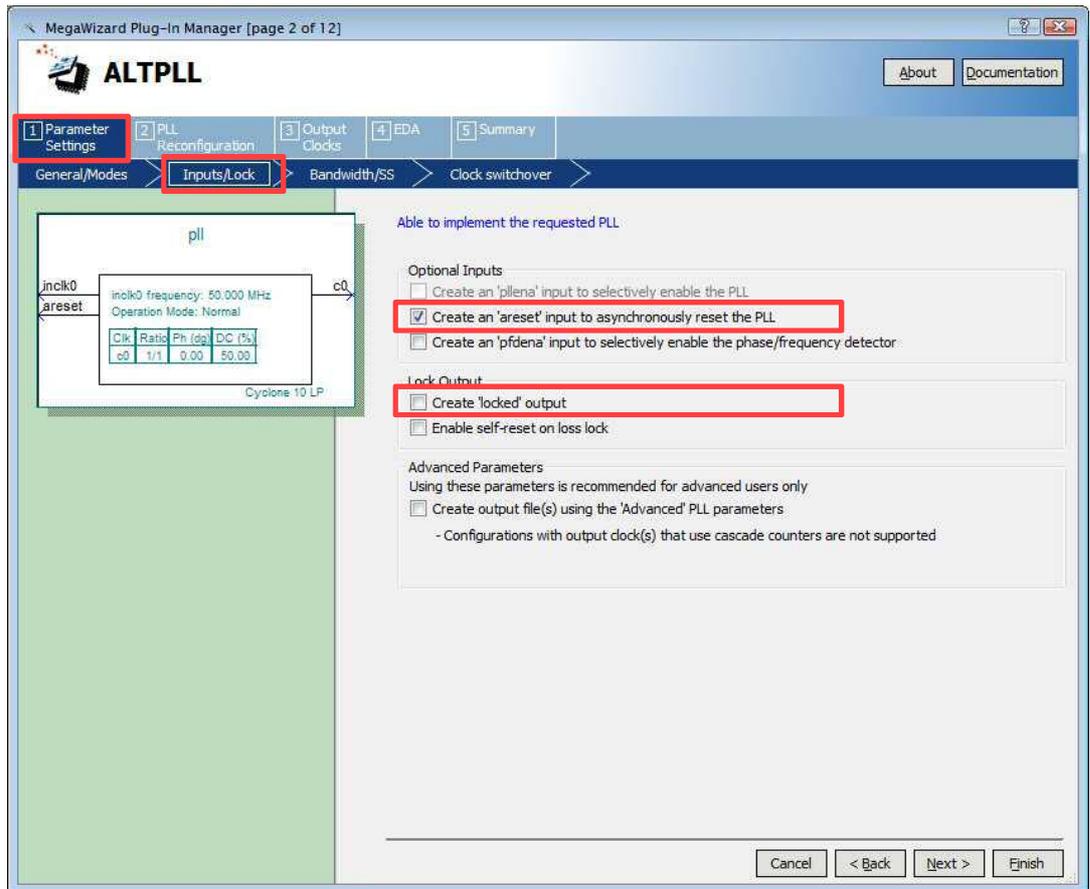


The screenshot shows the ALTPLL configuration window in the MegaWizard Plug-In Manager. The 'General/Modes' tab is active. The 'Currently selected device family' is 'Cyclone 10 LP'. The 'Which device speed grade will you be using?' dropdown is set to '7'. The 'What is the frequency of the inck0 input?' field is set to '50.000' MHz. The 'PLL Type' is set to 'Select the PLL type automatically'. The 'Operation Mode' is set to 'Use the feedback path inside the PLL' with 'In normal mode' selected. The 'Which output dock will be compensated for?' dropdown is set to 'c0'.

- ② 1 Parameter Settings > Inputs/Lock において、以下のとおりに設定します。

Parameter Settings > Inputs/Lock			
Board	Create an 'areset' input to asynchronously reset the PLL	Create 'locked' output	その他の設定
MAX 10 FPGA Evaluation Kit	On	Off	デフォルト
Cyclone 10 LP Evaluation Kit	On	Off	デフォルト

オフにしたポートは PLL ブロックのグラフィカルなプレビューから消えます。

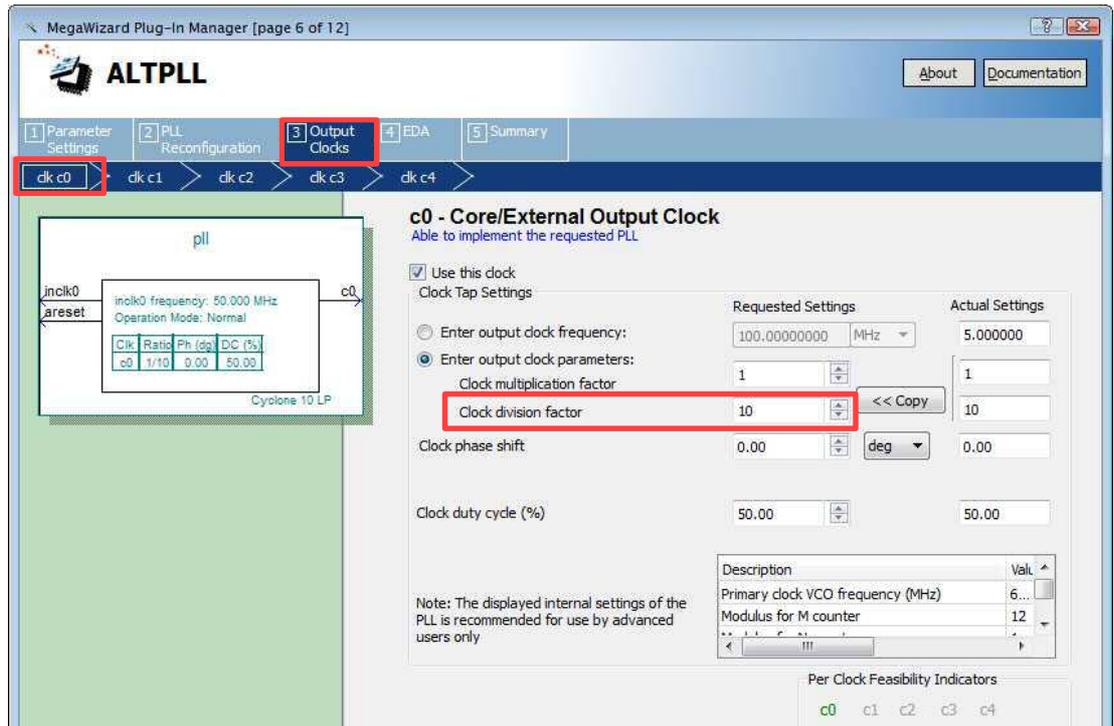


Next ボタンをクリックします。

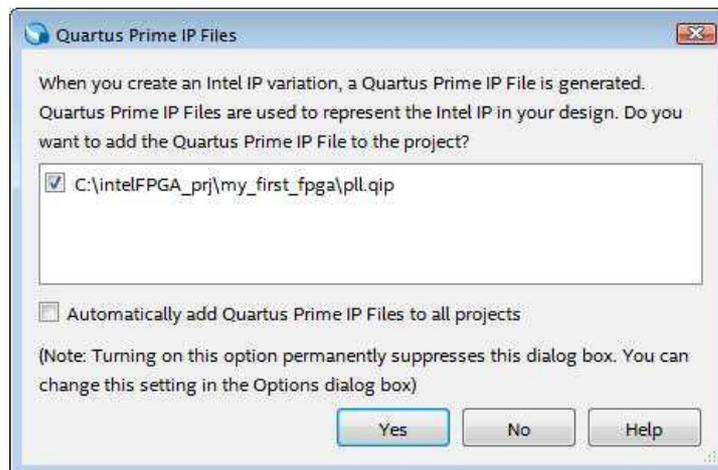
- ③ 3 Output Clocks をクリックし、clk c0 ページへジャンプします。

Clock division factor に下表の該当する値を入力します。(または上下ボタンで数値を設定します。)

Output Clocks > clk c0		
Board	Clock division factor	その他の設定
MAX 10 FPGA Evaluation Kit	10	デフォルト
Cyclone 10 LP Evaluation Kit	10	デフォルト



- ④ Finish ボタンをクリックします。
- ⑤ 5 Summary では、生成するファイルを選択できます。
このチュートリアルでは、*.cmp ファイル、*_inst.vhd ファイルを選択します。
- ⑥ Finish ボタンをクリックします。
以下のダイアログボックスが表示されますので、Yes ボタンをクリックします。

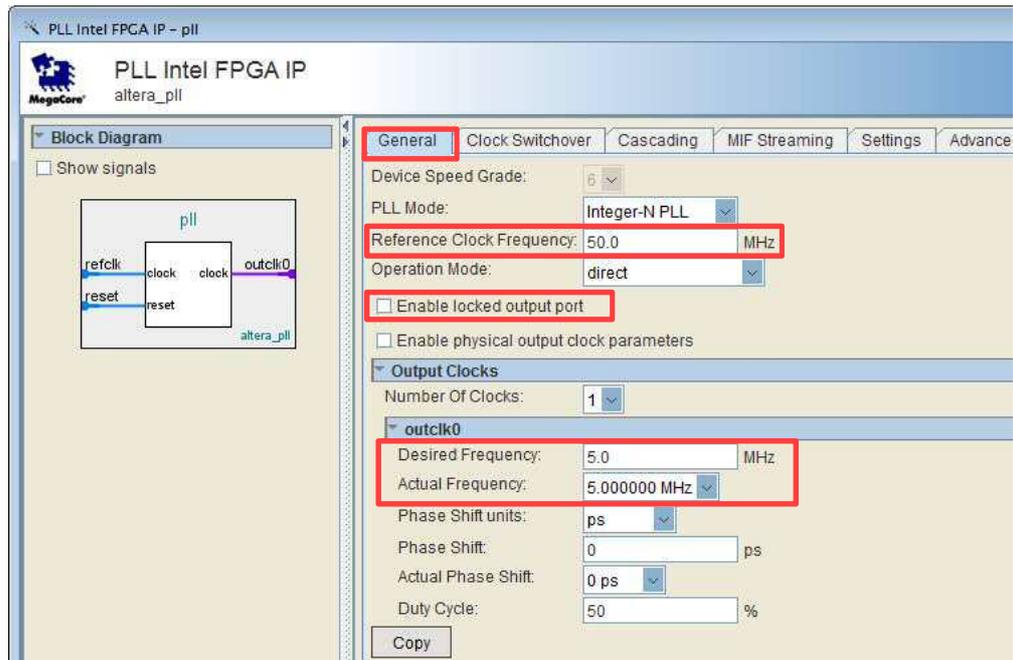


以上で ALTPLL の作成は完了です。 < 19 ページへ進みます。 >

- PLL Intel FPGA IP (Beryll Cyclone V GX Base Board) の設定

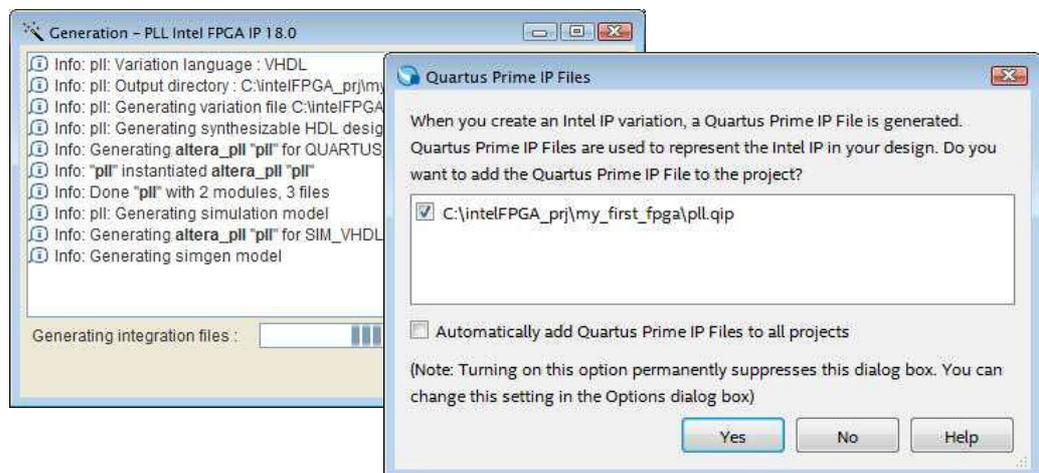
① General タブにおいて、以下の設定をします。

General タブ	
Reference Clock Frequency	50 MHz
Enable locked output port	OFF
Desired Frequency	5 MHz
Actual Frequency	5 MHz
その他の設定	デフォルト



② Finish ボタンをクリックすると、ファイル生成が開始されます。

Generation ダイアログボックスに Generation Successful が表示されたら Exit ボタンをクリックします。以下右のダイアログボックスが表示されますので、Yes ボタンをクリックします。



以上で PLL Intel FPGA IP の作成は完了です。

■ 最上位階層デザインを設計する

これまでで作成した各エンティティを接続し、最上位階層のエンティティを作成します。VHDL で回路を完成させます。

1. File メニュー ➤ New ➤ VHDL File (Design Files 内) をダブルクリックで選択します。
2. File メニュー ➤ Save As... により、これから作成するデザイン・ファイルに名前を付け、保存するディレクトリを指定します。
 - ・ 保存するディレクトリ: C:\¥intel\FPGA_prj¥my_first_fpga
 - ・ ファイル名: my_first_fpga
 - ・ ファイルの種類: VHDL Files (*.vhd *.vhd)
 - ・ Add file to current project オプション = On
3. 次の VHDL コードを my_first_fpga.vhd ファイルに入力します。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity my_first_fpga is
  port
  (
    OSC_CLK : in std_logic;
    RESET   : in std_logic;
    BUTTON  : in std_logic;
    LED     : out unsigned (3 downto 0)
  );
```

```
end entity;
```

```
architecture rtl of my_first_fpga is
```

```
  signal pll_c0 : std_logic;
  signal cnt    : unsigned (31 downto 0);
  signal tmp    : unsigned (3 downto 0);
```

```
  component pll
  port (
    areset : in std_logic := '0';
    inclk0  : in std_logic := '0';
    c0      : out std_logic
  );
end component;
```

-- PLL Intel FPGA IP デザイン (Beryll Cyclone V GX
-- Base Board) の場合は、左の点線枠内を下記に
-- 変更してください。

```
  component pll is
  port (
    refclk : in std_logic
    rst    : in std_logic
    outclk_0 : out std_logic
  );
end component pll;
```

```

component simple_counter
  port (
    clock      : in std_logic;
    reset      : in std_logic;
    counter_out : out unsigned(31 downto 0)
  );
end component;
    
```

```

component counter_bus_mux
  port (
    dataa      : in unsigned(3 downto 0);
    datab     : in unsigned(3 downto 0);
    sel        : in std_logic;
    result     : out unsigned(3 downto 0)
  );
end component;
    
```

```
begin
```

```

pll_inst : pll port map (
  areset => not RESET,
  inclk0 => OSC_CLK,
  c0     => pll_c0
);
    
```

-- PLL Intel FPGA IP デザイン (Beryll Cyclone V GX
 -- Base Board) の場合は、左の点線枠内を下記に
 -- 変更してください。

```

pll_inst : pll port map (
  rst      => not RESET,
  refclk   => OSC_CLK,
  outclk_0 => pll_c0
);
    
```

```

simple_counter_inst : simple_counter port map (
  clock      => pll_c0,
  reset      => RESET,
  counter_out => cnt
);
    
```

```

counter_bus_mux_inst : counter_bus_mux port map (
  dataa      => cnt(24 downto 21),
  datab     => cnt(26 downto 23),
  sel        => BUTTON,
  result     => tmp
);
    
```

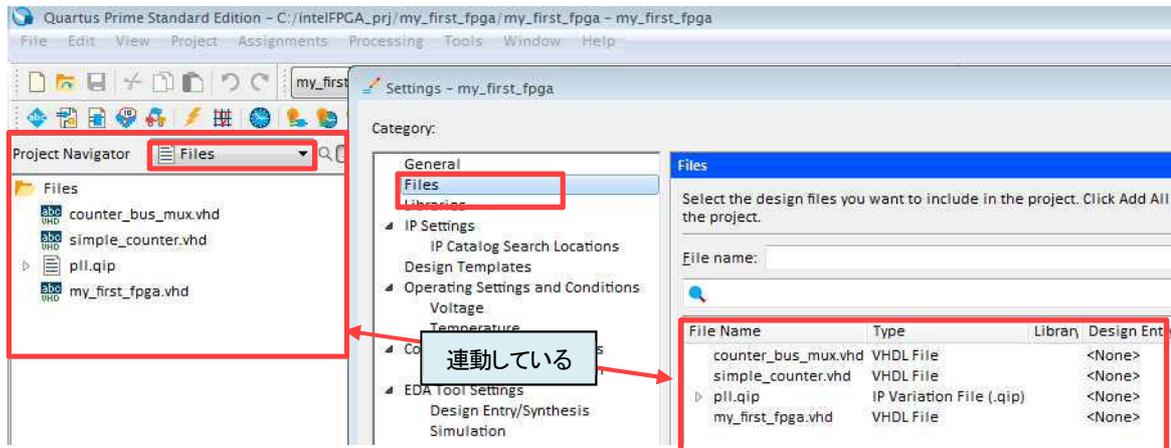
```

LED <= not tmp;
end rtl;
    
```

4. File メニュー ➤ Save (または Ctrl+S) により my_first_fpga.vhd を保存します。
5. my_first_fpga.vhd のタブの端にある Close ボタンをクリックし、ファイルを閉じます。

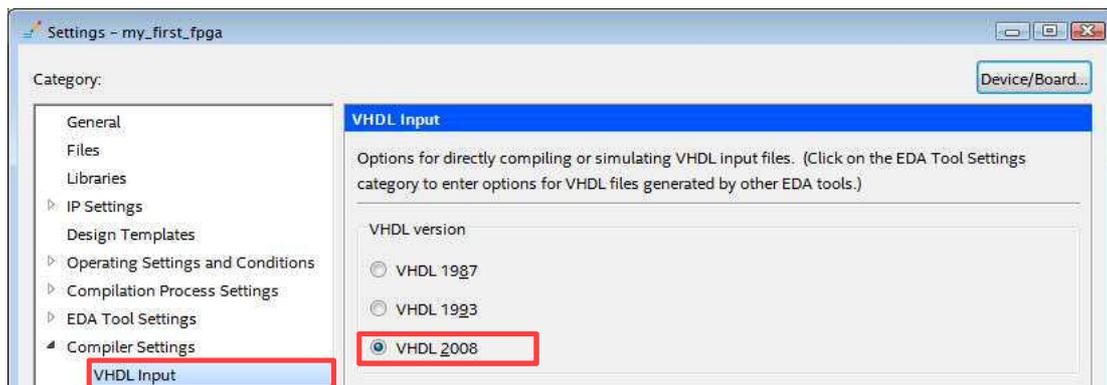
以上ですべてのデザイン作成が完了しました。

Project Navigator ウィンドウのプルダウン・リストを Files に切り替え、先ほど作成した4つのデザイン・ファイルが現在のプロジェクトに登録されていることを確認します。



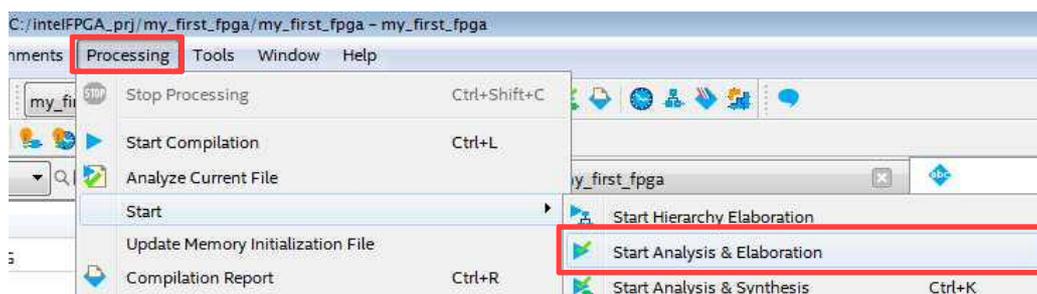
余分なファイルや不足があった場合には、Project メニュー ➤ Add/Remove Files in Project により追加あるいは削除を行ってください。

また、今回の VHDL デザインには、一部 VHDL-2008 に対応した記述があります。Assignments メニュー ➤ Settings ➤ Compiler Settings ➤ VHDL Input において、VHDL 2008 へ切り替えてください。



2-1-3. Analysis and Elaboration の実行

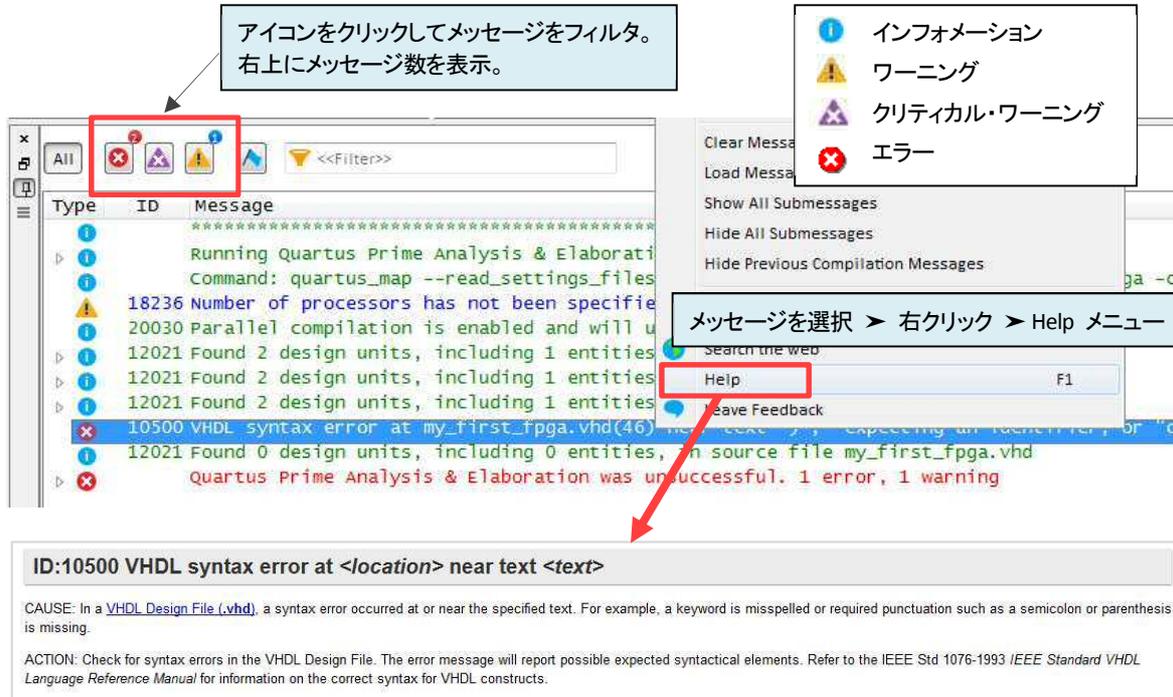
Processing メニュー ➤ Start ➤ Start Analysis and Elaboration をクリックし、コードのシンタックス・チェックや回路のデータベースを作成します。



もしエラーが発生した場合は、VHDL コードに問題があります。Messages ウィンドウに表示されたエラー・メッセージの内容をよく読み、指摘されている箇所付近の記述した内容を確認しましょう。

また、エラー・メッセージの他、ワーニング・メッセージが発生する場合があります。ワーニングは必ず内容を確認し、その内容を回避すべきか無視できるのかを仕様に応じて自己判断してください。

各メッセージ内容の詳細を確認するには、ヘルプ機能の活用が有効です。(次頁図を参照)ヘルプには、メッセージが発生した要因や回避するためのヒントが掲載されていますので、その情報からエラーなどを回避してください。

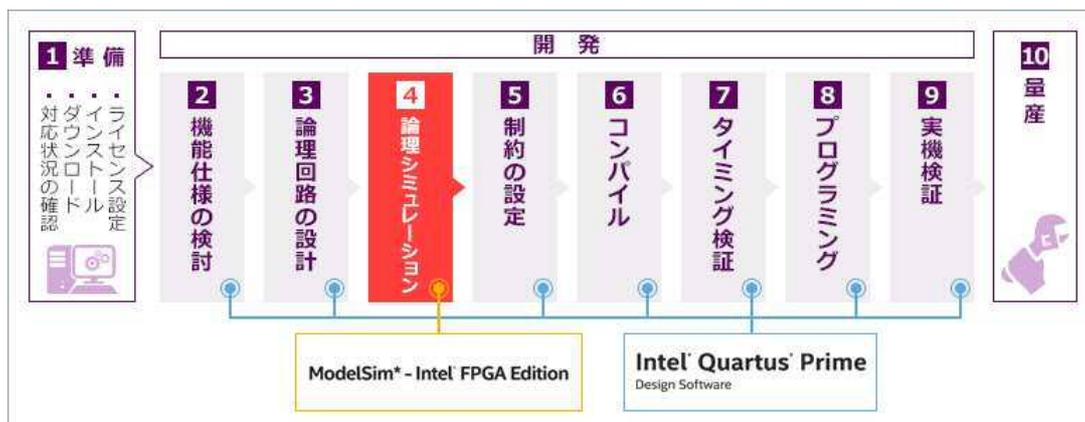


デザインの修正など、コード編集を行った場合はデザイン・ファイルを保存(File メニュー > Save)し、再び Start Analysis and Elaboration を実行します。エラーやワーニングを回避できたら、ここでの作業は完了です。

以上で論理回路の設計は完了です。

2-2. 論理シミュレーションを実行する

2-1-2 章で作成した論理回路の動作確認をするため、論理シミュレーションを行います。(この作業は、開発フローの [4] にあたります。)



シミュレーションを行うには、FPGA に対して入力する信号と出力した信号をモニタする内容を HDL 言語で表現した “テストベンチ” が必要です。テストベンチは、仕様に合わせてユーザが設計します。

ここでは、このチュートリアル用にサンプル提供されたテストベンチ・ファイルを使用します。このチュートリアルを入手した WEB ページからダウンロードしてください。その後、作業しているプロジェクトのディレクトリに保存してください。

Note: ・ テストベンチの設計経験がないユーザは、以下をご活用ください。

[はじめてみよう！テストベンチ](#)

シミュレーション検証するには、インテル FPGA をサポートしている他メーカーの言語シミュレータが必要です。このチュートリアルでは、ModelSim* - Intel® FPGA Edition を使用して作業を進めます。

通常は ModelSim - Intel FPGA Edition を起動しオペレーションしますが、このチュートリアルでは “NativeLink 機能” により ModelSim - Intel FPGA Edition にシミュレーションを実行させるフローで作業を行います。

Notes: ・ NativeLink とは、Quartus Prime のバックグラウンドで EDA ツールを操作・実行することを示します。例えば、ModelSim - Intel FPGA Edition の GUI オペレーションを知らなくてもユーザは Quartus Prime 上の GUI 設定をするだけでシミュレーションが実行できるため、作業フローが簡素化されます。また、IP を含んだデザインの論理シミュレーションをする場合、IP 用のシミュレーション・ライブラリのセットアップが必要になりますが、NativeLink でシミュレーションを実行するとそれらの環境構築もスクリプトにより自動的にセットアップされるため、ユーザへのストレスが軽減されます。

詳細は下記ドキュメントをご参考ください。

[Using NativeLink Simulation \(Intel Quartus Prime Standard Edition \)](#)

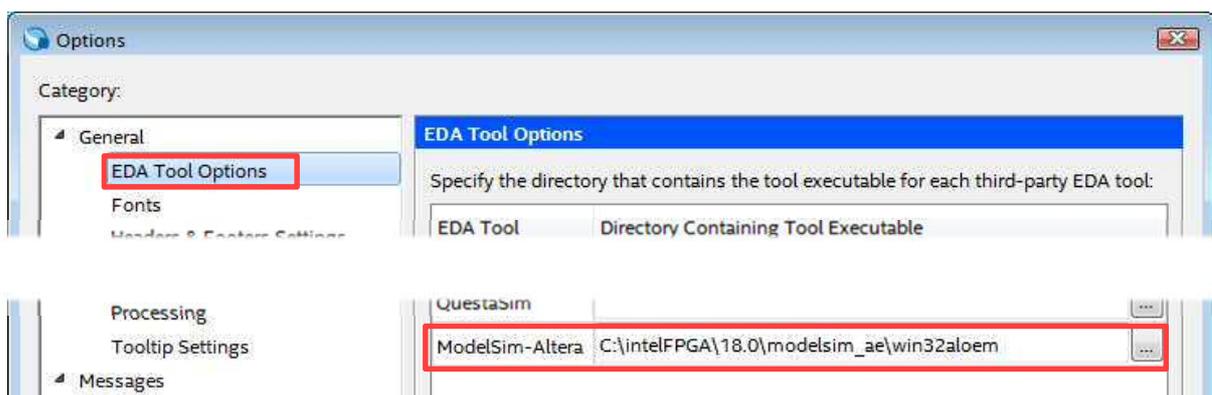
Intel Quartus Prime Standard Edition Handbook Volume 3 Verification より

- ・ ModelSim - Intel FPGA Edition を起動して GUI 操作によりシミュレーションを実行する場合は、下記ドキュメントをご参考ください。

[ModelSim - Intel FPGA Edition - RTL シミュレーションの方法](#)

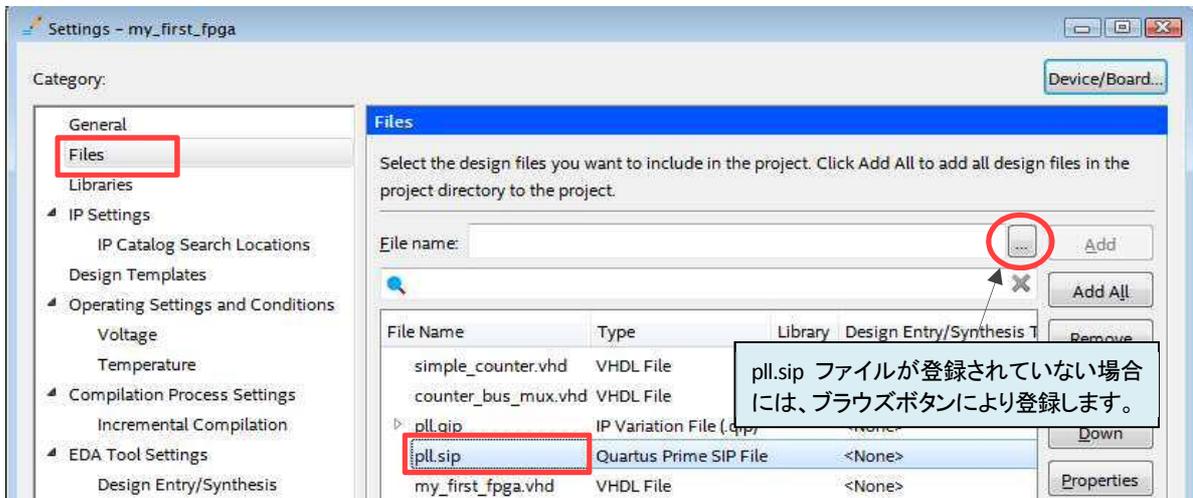
※ ModelSim - Intel FPGA Edition の GUI が起動している場合は、File メニュー ➤ Quit で終了しておきます。

1. Quartus Prime の Tools メニュー ➤ Options を選択し、Category 内の General ➤ EDA Tool Options をクリックします。ModelSim-Altera 欄に、ModelSim - Intel FPGA Edition の実行プログラム (modelsim.exe) が格納されているディレクトリのパスを設定します。(インストールされているディレクトリが異なる場合は、使用するパソコンに応じて変更してください。)



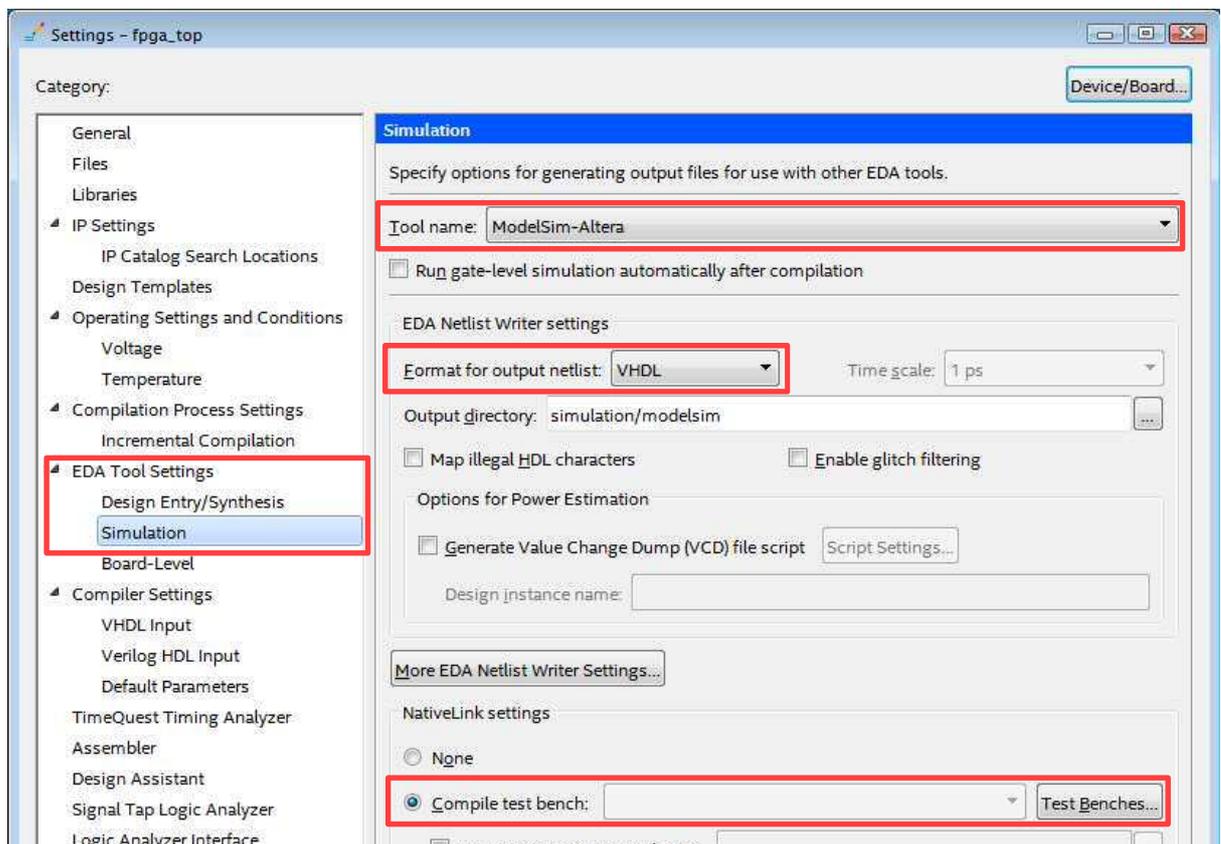
2. (Beryll Cyclone V GX Base Board ユーザのみ)

Assignments メニュー ➤ Settings ➤ Files (Category 内)を選択し、PLL IP のシミュレーションを NativeLink で行うため、ここへ pll.sip ファイルが登録されていることを確認します。

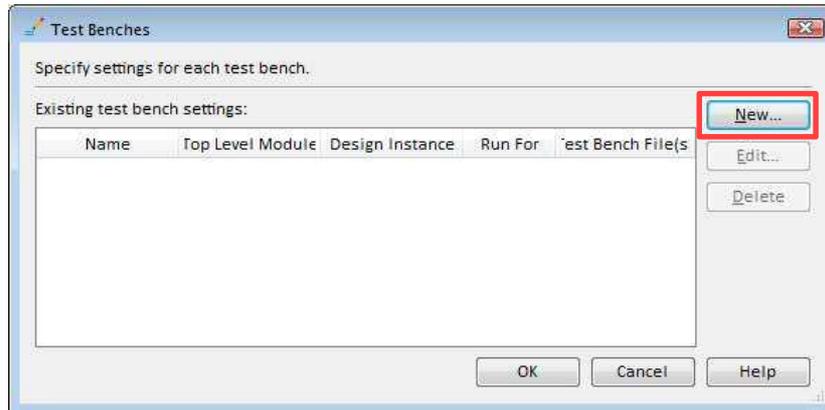


3. Assignments メニュー ➤ Settings ➤ EDA Tool Settings ➤ Simulation をクリックします。各項目を設定してください。

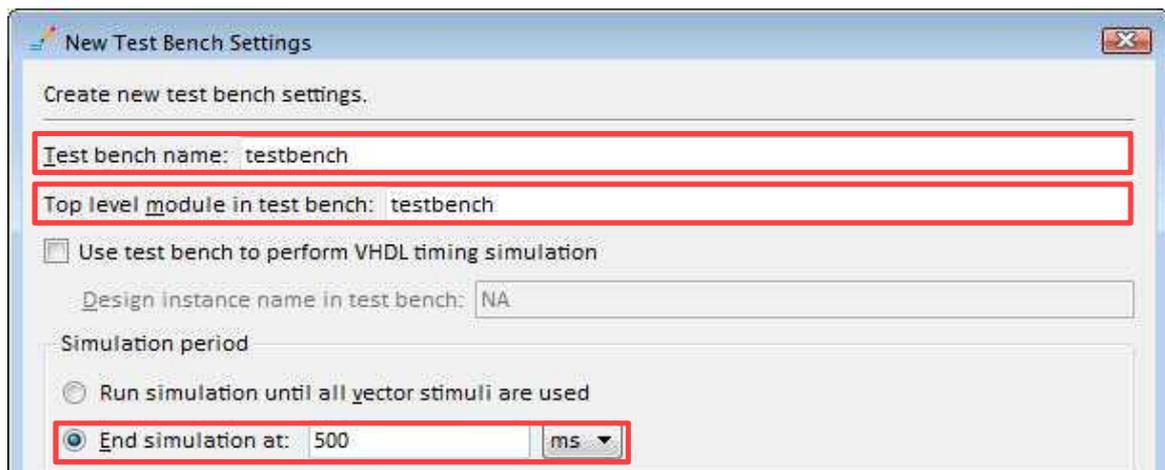
EDA Simulation Settings ➤ Simulation	
Tools name	ModelSim-Altera
Format for output netlist	VHDL
NativeLink settings セクション	Compile test bench を選択



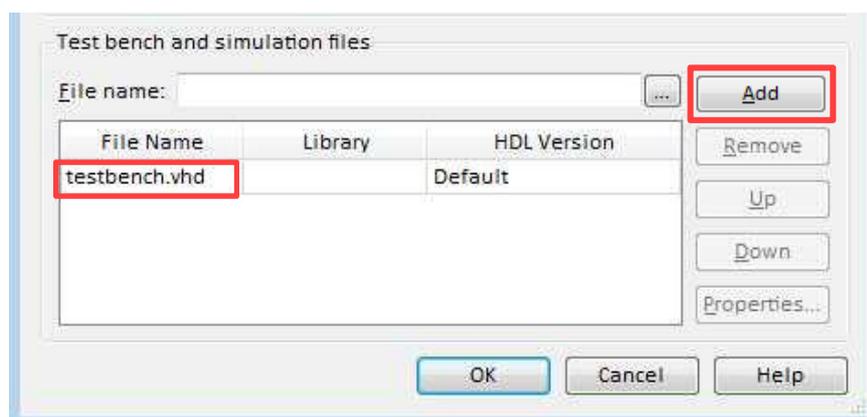
Test Benches ボタンをクリックします。Test Benches ウィンドウの New ボタンをクリックし、New Test Bench Settings ウィンドウにテストベンチ testbench.vhd の情報を入力します。



New Test Bench Settings	
Test bench name	testbench
Top level module in test bench	testbench
Simulation period セクション	End simulation at: 500 と入力 / 単位: ms を選択

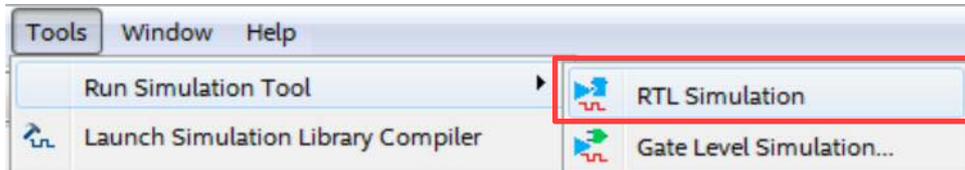


Test bench and simulation files セクションの File name 右横にあるブラウズボタンをクリックします。Select File ウィンドウにおいて作業ディレクトリを開き、WEB ページから入手したテストベンチ・ファイル (testbench.vhd) を選択後 Open ボタンをクリックし、Add ボタンをクリックして登録します



すべての設定画面を OK ボタンで閉じます。

- Tools メニュー ➤ Run Simulation Tool ➤ RTL Simulation を選択し、シミュレーションを実行します。この時、ModelSim - Intel FPGA Edition の GUI が起動していないことを確認してください



自動的に ModelSim - Intel FPGA Edition が起動し、シミュレーションが実行されます。画面の左下に、シミュレーションの実行時間が表示されます。このチュートリアルでは 500ms の実行が終了するまで待ちます。

```
# view signals
# .main_pane.objects.interior.cs.body.tree
# run 500 ms
# ** Note: Cyclone 10 LP PLL locked to incoming clock
# Time: 90 ns Iteration: 3 Instance: /testbench/ui/pll_inst
```

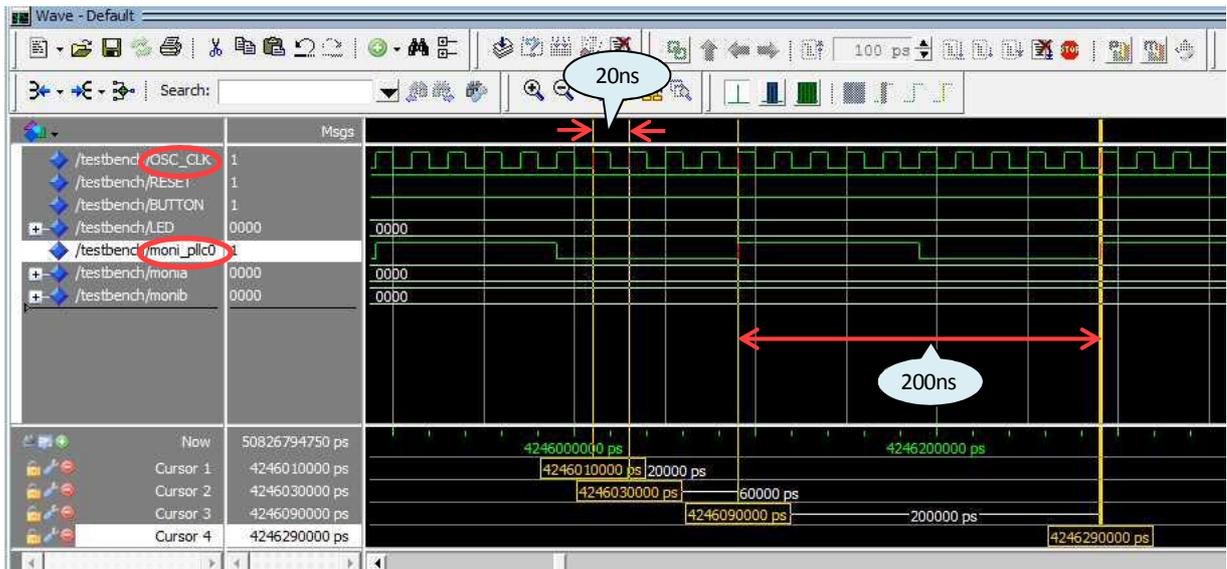
Now: 1,848,470 ns Delta: 5 sim:/testbench

もし HDL デザインのシンタックス・エラーなどで処理が停止してしまった場合は、一度 ModelSim - Intel FPGA Edition の GUI を閉じ、エラーの発生した HDL ファイルを修正および保存してから再度 Tools メニュー ➤ Run Simulation Tool ➤ RTL Simulation を選択し、シミュレーションを実行してください。

- シミュレーションが終了したら、Wave ウィンドウに表示された波形を確認します。moni_pll0、monia、monib 信号を見てください。これらは、my_first_fpga エンティティの内部信号をモニタしたものです。
 - moni_pll0 : PLL 回路の出力クロック
 - monia : カウンタ回路(32bit カウンタ)のレジスタ出力 [cnt(24 downto 21)]
 - monib : カウンタ回路(32bit カウンタ)のレジスタ出力 [cnt(26 downto 23)]

このチュートリアルで使用する開発キットに搭載されているクロックは 50MHz(20ns) です。デザインでは、50MHz を FPGA へ入力し、内部に構築した PLL のリファレンス・クロックとしています。また、PLL により 5MHz(200ns) のクロックを生成させ、カウンタ回路のクロックに用いています。

Wave ウィンドウにおいて、PLL の出カクロック(moni_pll0)が正常に生成されていることを確認します。



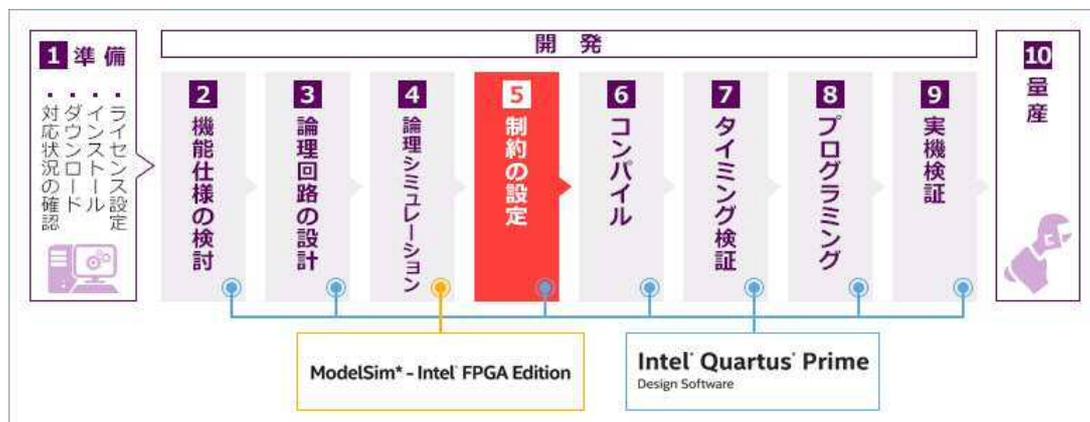
同様に、カウンタ回路(monია、monib)が PLL の出カクロックに同期して正常にカウントアップしていることを確認します。(ファンクション・シミュレーションは遅延を考慮しないため、クロックのエッジで出力信号が変化していることが確認できます。)

6. ModelSim - Intel FPGA Edition 上の Simulate メニュー ➤ End Simulation を選択し、シミュレーションを終了します。
7. File メニュー ➤ Quit を選択し、ModelSim - Intel FPGA Edition を終了します。

以上で、作成したデザインの論理シミュレーションは完了です。

2-3. 制約の設定をする

論理回路に対して各種オプションを設定します。(この作業は、開発フローの [5] にあたります。)



このチュートリアルでは、以下の内容に制約を行います。

- 論理回路の入出力ピンに、ターゲット・デバイスの I/O ピン番号へ割り当て、ボード仕様に応じた I/O 規格を設定 [ユーザ I/O ピンのアサイン]
- 論理回路に対して希望の動作スピードを設定 [タイミング制約]

2-3-1. ユーザ I/O ピンのアサイン

ピンの割り当てを行うためには、Analysis & Elaboration の実行が必要です。このチュートリアルでは [2-1-3 章](#) ですでに実行していますのでスキップできます。未完了の場合には、以下のメニューから実行してください。

1. Processing メニュー ➤ Start ➤ Start Analysis & Elaboration

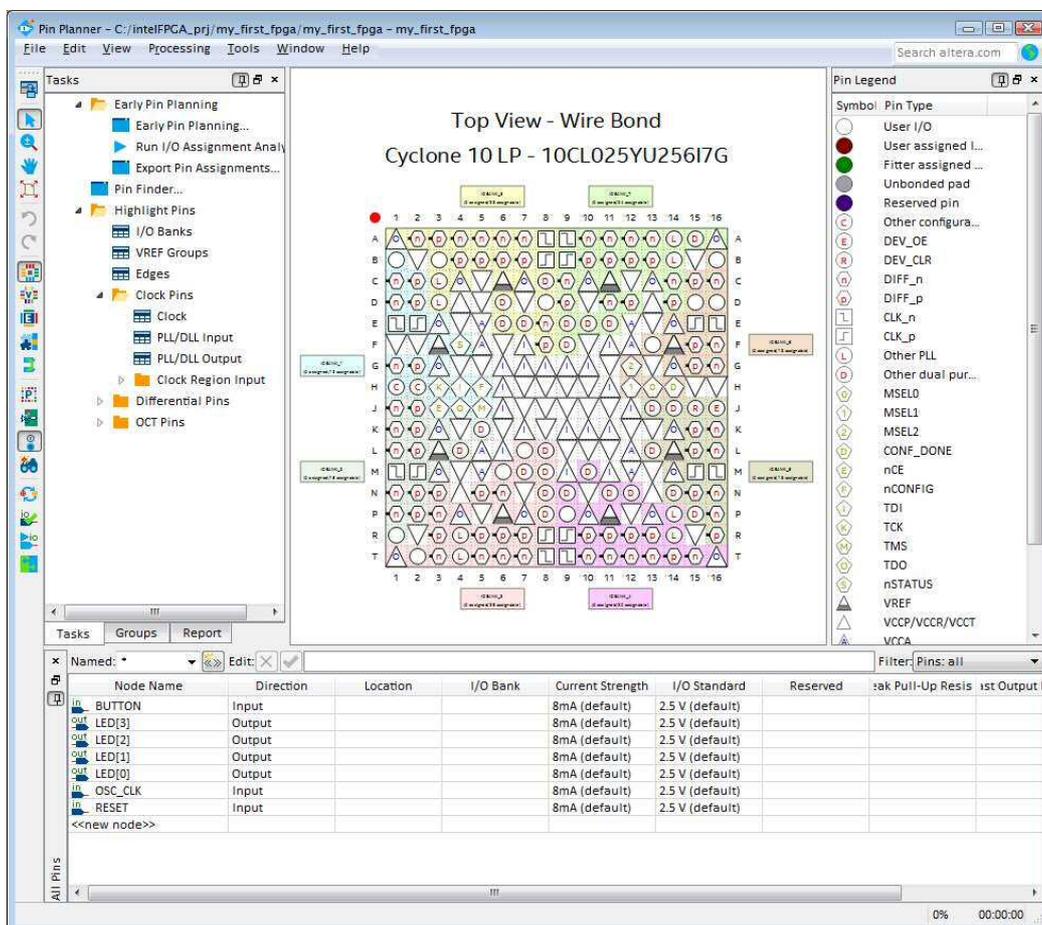
Analysis & Elaboration の実行後にエラーが発生した場合は、2-1-3 章を参考にエラーを回避してください。

デザイン上に存在するピンは、以下のとおりです。

入力ピン	OSC_CLK、RESET、BUTTON
出力ピン	LED[3]、LED[2]、LED[1]、LED[0]

2. これらにピン番号をアサインするため、Pin Planner を起動します。

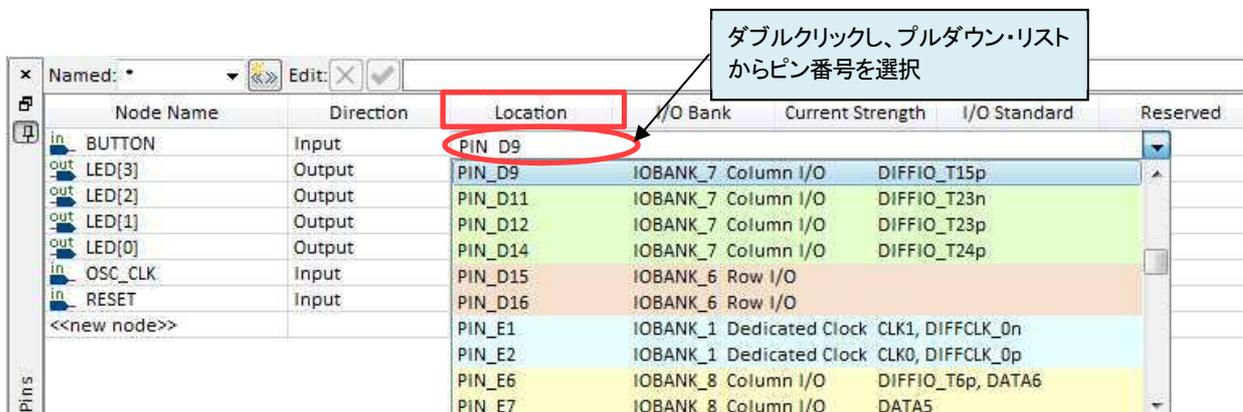
Assignments メニュー ➤ Pin Planner



3. All Pins リストにデザイン上のピンが表示されていることを確認してください。各ピン名の横にある Location カラムに、下表のとおりピン番号を指定します。

ピン・アサイン情報			
ピン名	MAX 10 FPGA Evaluation Kit	Cyclone 10 LP Evaluation Kit	Beryll Cyclone V GX Base Board
BUTTON	120	D9	V12
LED[3]	140	J13	J20
LED[2]	135	J14	H20
LED[1]	134	K15	G20
LED[0]	132	L14	H19
OSC_CLK	27	E1	U12
RESET	121	E15	AB24

Note ・ 各ピン番号が該当のボードのどのタクトスイッチや DIP スイッチ、LED なのかを、リファレンス・マニュアルやスキマティックで確認してください。



4. 各ピン名の横にある I/O Standard カラムに、下表のとおり I/O 規格を指定します。

I/O 規格情報			
ピン名	MAX 10 FPGA Evaluation Kit	Cyclone 10 LP Evaluation Kit	Beryll Cyclone V GX Base Board
BUTTON	3.3-V LVTTTL	3.3-V LVTTTL	1.5 V
LED[3]	3.3-V LVTTTL	3.3-V LVTTTL	3.3-V LVTTTL
LED[2]	3.3-V LVTTTL	3.3-V LVTTTL	3.3-V LVTTTL
LED[1]	3.3-V LVTTTL	3.3-V LVTTTL	3.3-V LVTTTL
LED[0]	3.3-V LVTTTL	3.3-V LVTTTL	3.3-V LVTTTL
OSC_CLK	3.3-V LVTTTL	3.3-V LVTTTL	1.5 V
RESET	3.3-V LVTTTL	3.3-V LVTTTL	3.3-V LVTTTL

関連情報: 下記ドキュメントをご参考ください。

[「Quartus Prime はじめてガイド - ピン・アサインの方法」](#)

[「Quartus はじめてガイド - デバイス・オプションの設定方法」](#)

2-3-2. タイミング制約

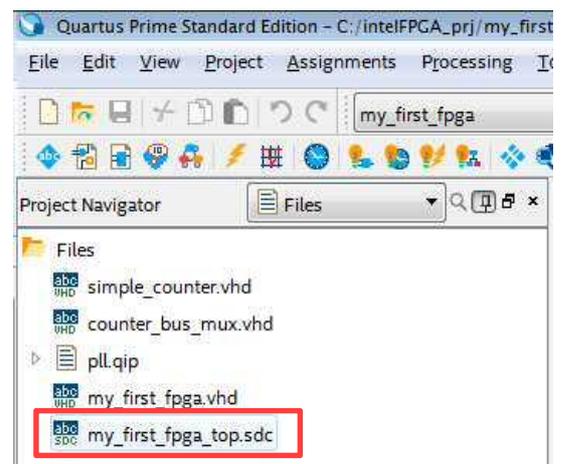
論理回路に対して希望の動作スピードを設定するため、タイミング制約を作成します。Quartus Prime では、ASIC 業界で標準的に使用されている Synopsys Design Constraints (SDC) ファイル・フォーマットでタイミングの制約を行います。

1. Tools メニュー ➤ Timing Analyzer を選択し、Timing Analyzer ウィンドウを表示させます。
2. File メニュー ➤ New SDC File を選択すると、Quartus Prime のメイン・ウィンドウに SDC ファイル・エディタが開きます。
3. SDC ファイルに、以下のコードを入力してください。

```
create_clock -period 20.000 -name osc_clk OSC_CLK
derive_pll_clocks
derive_clock_uncertainty
```



4. File メニュー ➤ Save As... により、SDC ファイルに名前を付け、保存します。
 - ・ 保存するディレクトリ: C:\intelFPGA_prj\my_first_fpga
 - ・ ファイル名: my_first_fpga_top.sdc
 - ・ ファイルの種類: Synopsys Design Constraints Files (*.sdc)
 - ・ Add file to current project オプション = On
5. Project Navigator ウィンドウのプルダウン・リストを Files に切り替え、my_first_fpga_top.sdc が現在のプロジェクトに登録されていることを確認します。



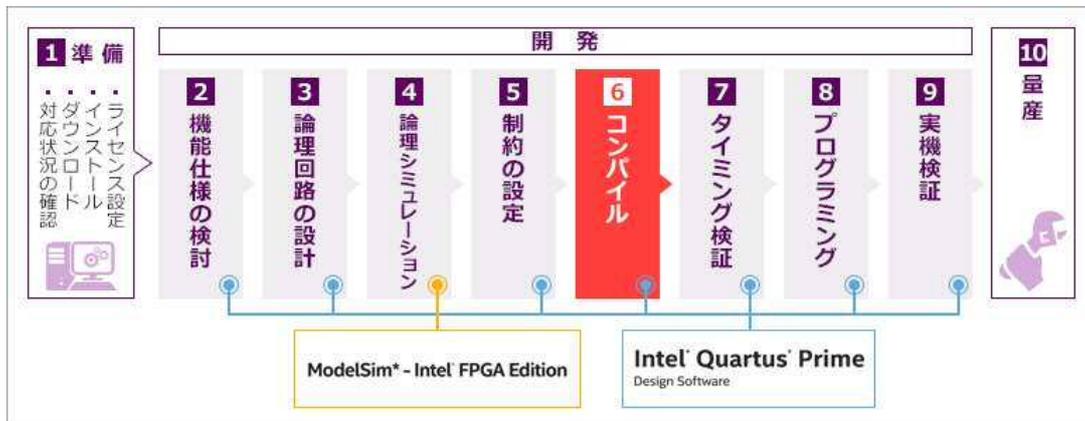
関連情報 下記ドキュメントをご参考ください。

[「Quartus はじめてガイド - タイミング制約の方法」](#)

以上で、デザインに対する制約設定は完了です。

2-4. コンパイルを実行する

完成したデザインをコンパイルします。(この作業は、開発フローの [6] にあたります。)



1. デザインが完成したらコンパイルを実行します。

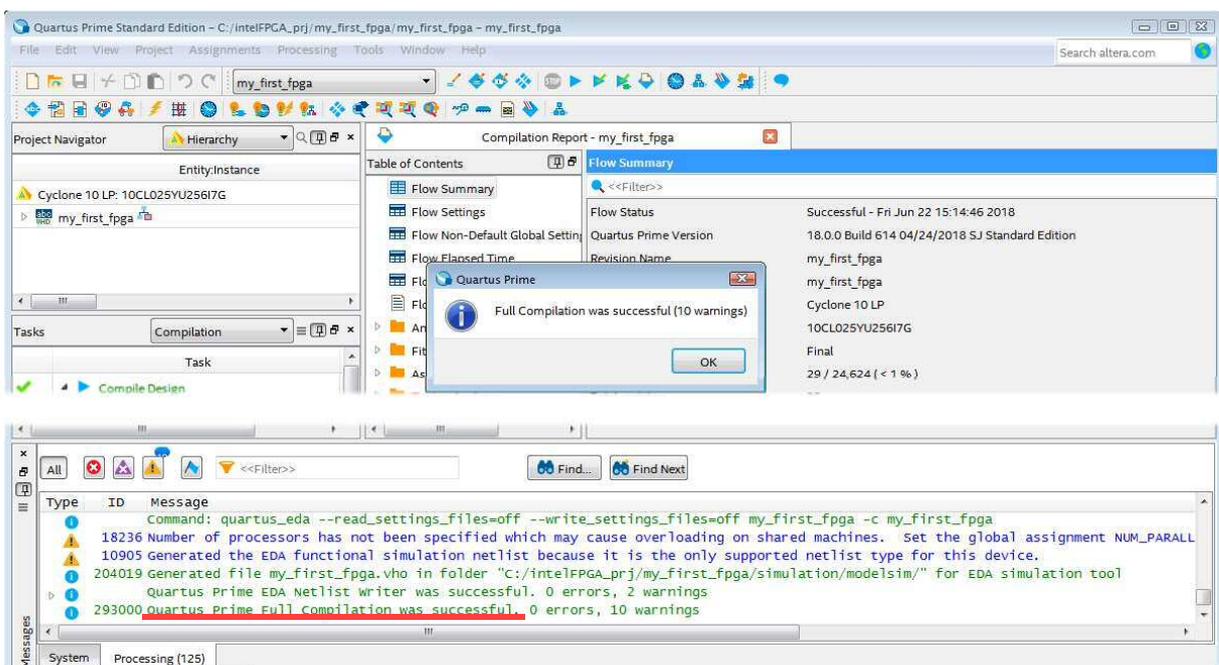
Processing メニュー ➔ Start Compilation

コンパイルでは、論理合成や配置配線、タイミング解析、プログラミング・ファイルの作成が自動で行われます。

コンパイルを実行すると、Messages ウィンドウに様々な情報が表示されます。これらメッセージに関しては、本チュートリアル 2-1-3. Analysis and Elaboration の実行(21 ページ)を参考にして、対処してください。



2. Full Compilation was successful のメッセージが確認できたら、コンパイル終了です。



コンパイルが終了すると、コンパイル・レポート(Compilation Report)にコンパイル結果が表示されます。

コンパイルでは、Analysis & Synthesis、Fitter、Assembler、Timing Analyzer の4工程が基本的に実行され、工程ごとにレポートが生成されます。ユーザは、自分の仕様に適切な処理が行われたかを確認するため、コンパイル・レポートを必ず見る必要があります。

詳細は、下記ドキュメントをご覧ください。

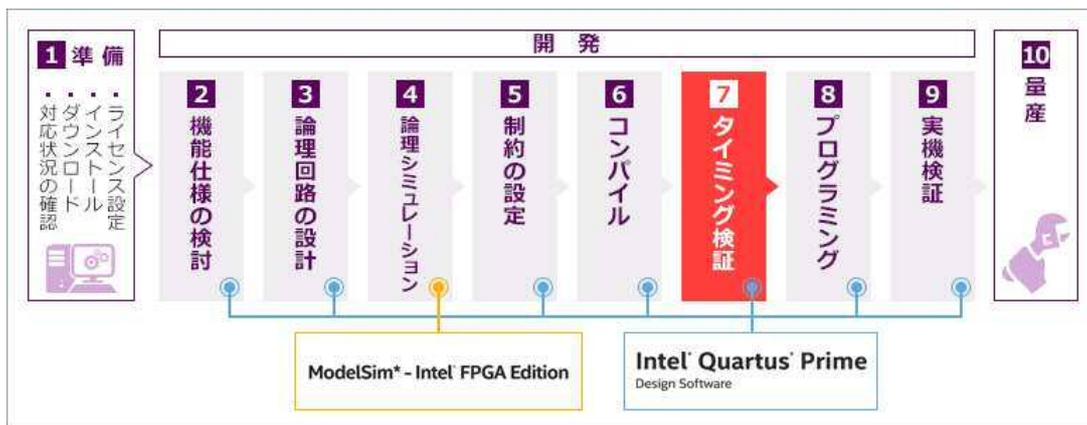
コンパイル・レポートを閉じた後に再び表示させるには、Processing メニュー ➤ Compilation Report を選択してください。

関連情報: 下記ドキュメントをご参考ください。

[「Quartus Prime はじめてガイド - コンパイル・レポート・ファイルの見方」](#)

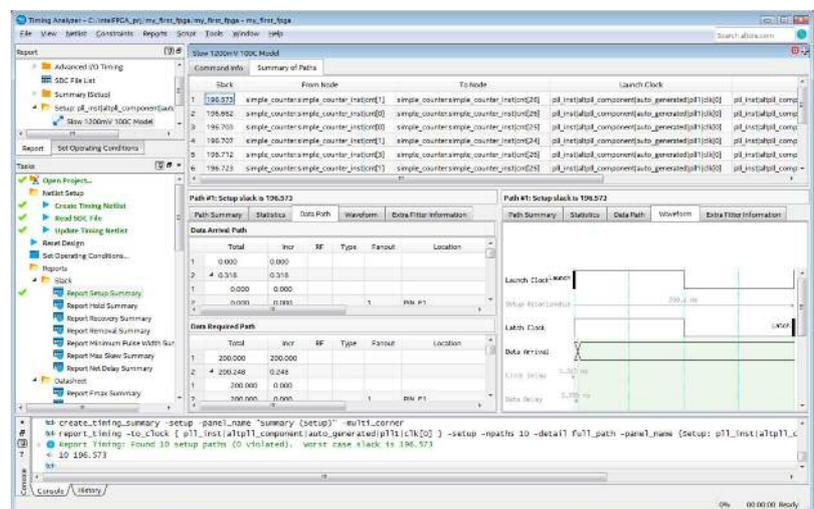
2-5. タイミングを検証する

コンパイルの配置配線結果がタイミング要件を満足したかを確認します。(この作業は、開発フローの [7] にあたります。)



Quartus Prime は、コンパイル実行と同時に、タイミング検証も実施しています。ユーザが事前に登録したタイミング制約と配置配線した際のタイミング情報とを比較検証し、ユーザの要求値を満足(メット)しているかどうかを Timing Analyzer レポート(コンパイル・レポート内)で確認することができます。

実際の設計においては、タイミングがユーザの要求値を満足していないと実機でも動作しないため、非常に重要な作業工程です。まずコンパイル・レポートで検証結果のサマリを確認し、メットできていない箇所があった場合にはそのパスを特定します。その後 Timing Analyzer を別途起動して要因を追究し、回避しなければなりません。



今回は、タイミング制約内容を満足する仕様になっていることと、Quartus Prime の基本的なオペレーションを体験するためのチュートリアルなので、タイミング検証は割愛します。

下記に関連情報をご案内しますので、次のステップとして習得してください。

関連情報: 下記ドキュメントをご参考ください。

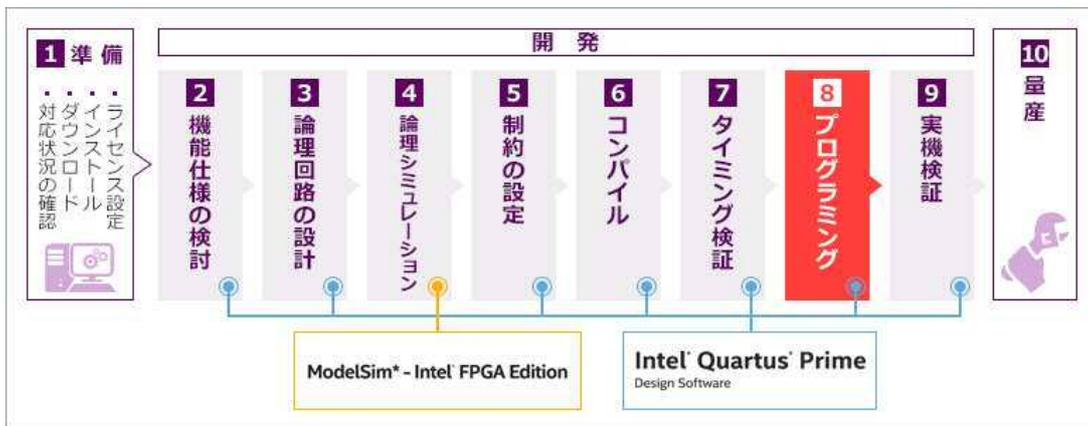
[「Quartus Prime はじめてガイド - TimeQuest によるタイミング解析の方法」](#)

タイミング検証のトレーニングも開催されています。

[インテル FPGA テクニカル・トレーニング「Quartus Prime パーフェクト・コース II : タイミング解析」](#)

2-6. プログラミング

FPGA ヘータを書き込みます。(この作業は、開発フローの [8] にあたります。)



デザインをコンパイルしてタイミングを検証したら、開発ボード上の FPGA を実際に動作させる準備が整いました。作成したファイルをダウンロードします。

Note: ・ Cyclone 10 LP Evaluation Kit をご利用の場合は、事前に DIP スイッチ [SW1] を下図のとおりに変更してから作業を行ってください。



プログラミング・データは、コンパイル実行時に作成され、FPGA の場合 .sof ファイルが生成されます。

sof ファイルは、FPGA の SRAM 部分に書き込むための形式です。SRAM にデータを書き込んだ場合、FPGA へ供給される電源が OFF になるとデータは消去されます。そのため、開発ボードの電源を OFF にしても次の電源投入時に FPGA が動作をするためには、インテル FPGA 用のコンフィギュレーション・デバイス (FPGA が起動するための ROM) に開発したデザインのデータを書き込む必要があります。

コンフィギュレーション・デバイスへ書き込んで実機検証を行っても良いのですが、FPGA への書き込み時間と比べて時間がかかることと、書き込み/書き換え/消去に対する実行回数に上限があるため、一般的には先に FPGA の実機検証を行い、修正が無いことを確認した上でコンフィギュレーションにプログラムすることを推奨しています。

このチュートリアルにおいても、まずは FPGA に sof ファイルを書き込み、実機検証を行った後にコンフィギュレーション・デバイスへ書き込みを行います。

2-6-1. sof ファイルを書き込む

1. ダウンロード・ケーブルとパソコンを接続します。

Note: ・ 開発ボードの JTAG ヘッダの位置は、各開発ボードのユーザ・ガイドやスキマティックをご確認ください。

- MAX 10 FPGA Evaluation Kit の場合

インテル FPGA ダウンロード・ケーブル II と USB ケーブルを接続し、インテル FPGA ダウンロード・ケーブル II を開発ボードの JTAG ヘッダに接続します。その後 USB ケーブルのもう一方をプログラミングに使用するパソコンの USB ポートに接続します。

- Cyclone 10 LP Evaluation Kit の場合

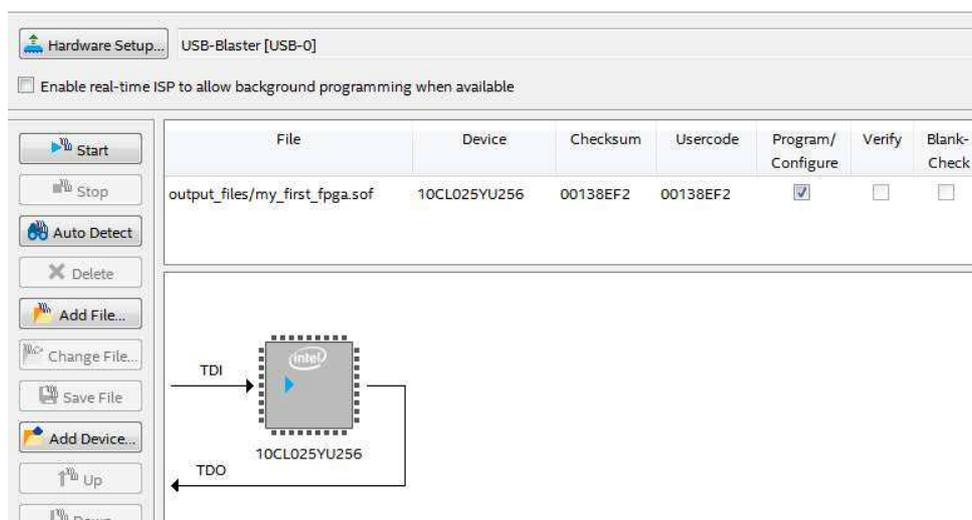
ここでの作業はありません。次のステップへ進んでください。

(Cyclone 10 LP Evaluation Kit は、ダウンロード・ケーブルと電源供給ケーブルが同じ USB ケーブルで賄われています。)

- Beryll Cyclone V GX Base Board の場合

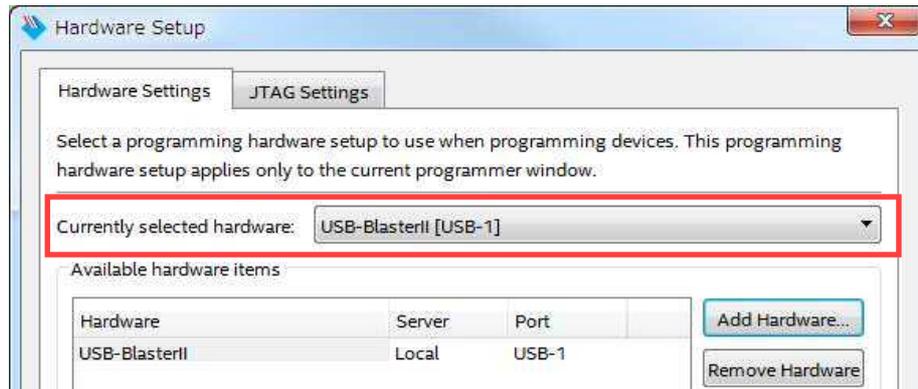
付属の USB ケーブルで開発ボードとパソコンを接続します。開発ボード側のコネクタ位置は、本開発ボードの Getting Started をご覧ください。

2. 開発ボードと電源ケーブルを接続します。(開発ボードによっては、電源ケーブルを接続すると電源が供給されます。)電源スイッチがある開発ボードの場合は、スイッチを入れて電源を供給します。
3. Tools メニュー ➤ Programmer により、Programmer ウィンドウを起動します。



4. Hardware Setup ボタン(Programmer 画面の左上)をクリックします。

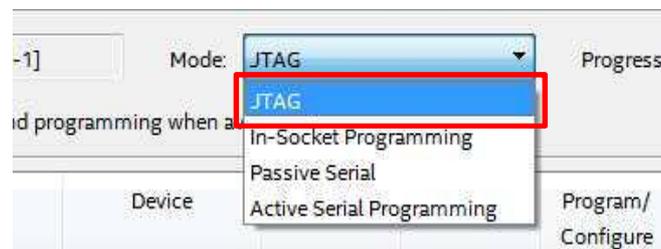
- Hardware Settings タブの Currently selected hardware 項のプルダウン・リストから使用するハードウェアを選択し、Close ボタンをクリックします。



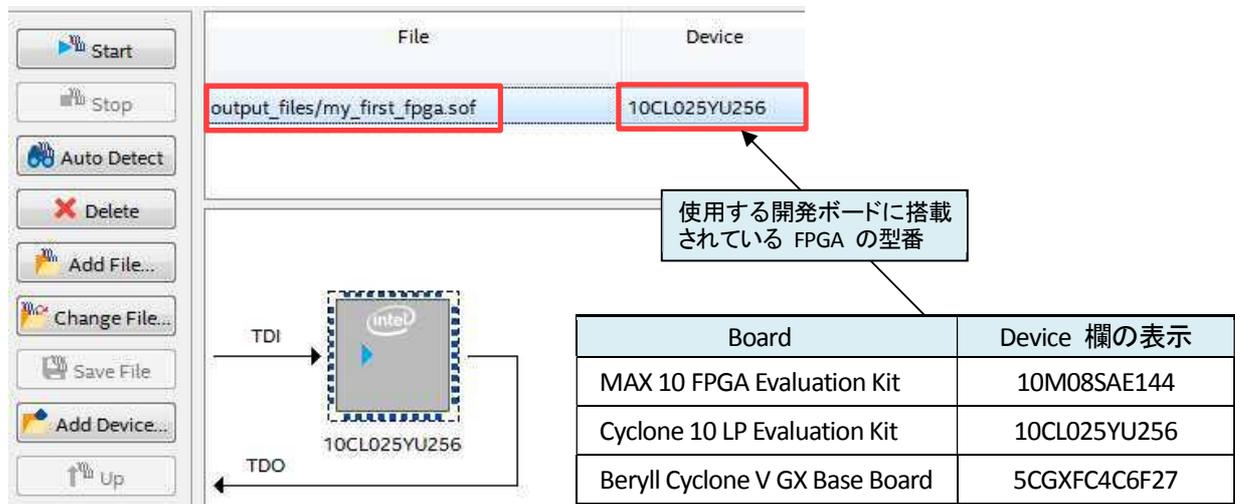
Board	Currently selected hardware に検出される名前
MAX 10 FPGA Evaluation Kit	使用するダウンロード・ケーブル名
Cyclone 10 LP Evaluation Kit	Cyclone 10 LP Evaluation Kit
Beryll Cyclone V GX Base Board	USB-Blaster

- Mode (Programmer 画面の右上) のプルダウン・リストより、デバイスへの書き込みを実行する手法を選択します。

このチュートリアルでは、JTAG を選択します。



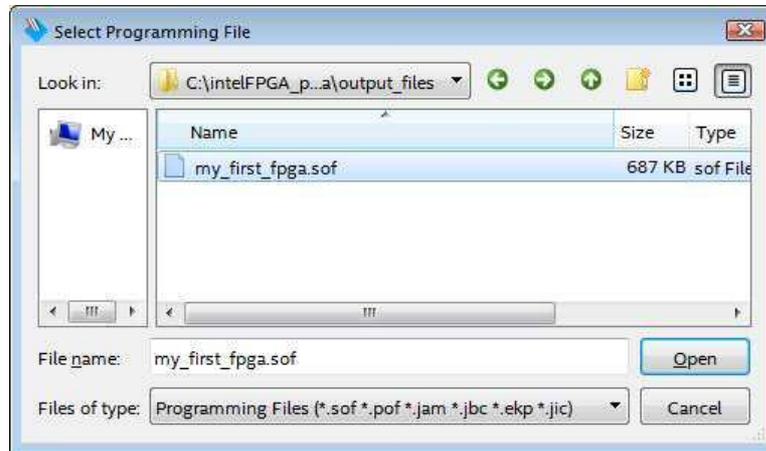
- Programmer 画面上部の File 欄には、セットしているプロジェクト用の my_fast_fpga.sof ファイルがすでに登録されています。また Device 欄には、開発している FPGA の型番が表示されています。



Board	Device 欄の表示
MAX 10 FPGA Evaluation Kit	10M08SAE144
Cyclone 10 LP Evaluation Kit	10CL025YU256
Beryll Cyclone V GX Base Board	5CGXFC4C6F27

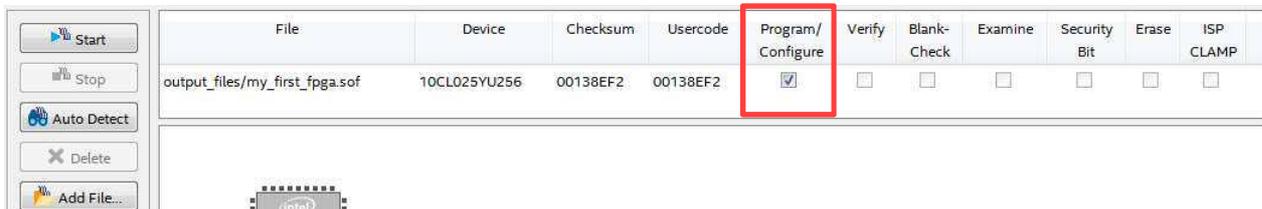
もし別の .sof ファイルが登録されている場合には、.sof ファイルをハイライト選択し Delete ボタンをクリックして削除(空欄)してください。

その後 Add File ボタンをクリックし、作業ディレクトリ内の output_files フォルダ内に生成されている my_first_fpga.sof ファイルを選択し、Open ボタンをクリックして File 欄に登録します。



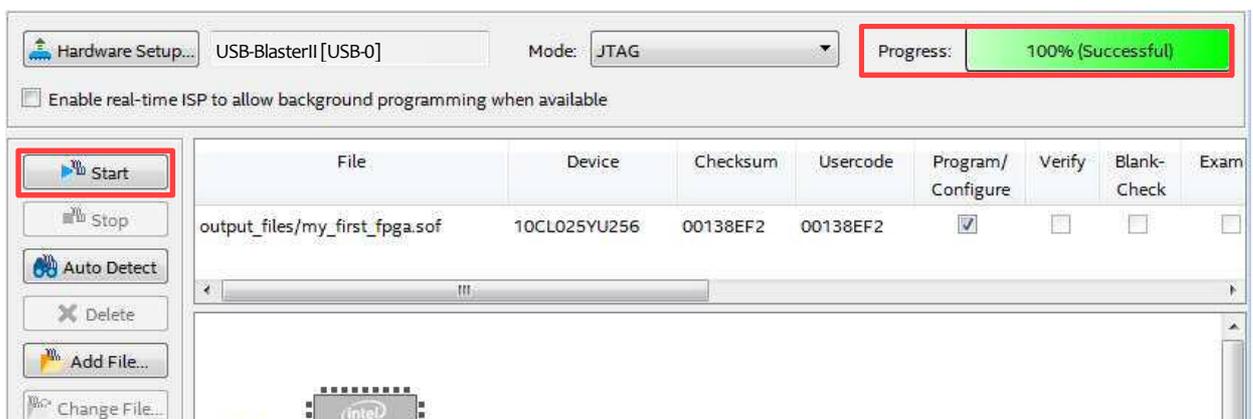
8. my_first_fpga.sof ファイル行の右側にあるプログラミング・オプションにおいて、実行する項目にチェックを入れます。

このチュートリアルでは、FPGA に .sof データを転送するので、Program/Configure オプションにチェックを入れてください。



9. Start ボタンをクリックし、データの書き込みをスタートさせます。

データのダウンロードが開始されます。Progress バーが 100% になり、Messages ウィンドウ(System タブ)に Info: Successfully performed operation(s) のメッセージを確認できたら書き込み完了です。



続いて、[2-7. 実機検証](#) を行い FPGA の動作を確認します。

2-6-2. コンフィギュレーション・デバイスに書き込む

この作業は、先に次項 2-7. 実機検証を行い、FPGA の動作に修正の必要が無いことを確認した上で行ってください。

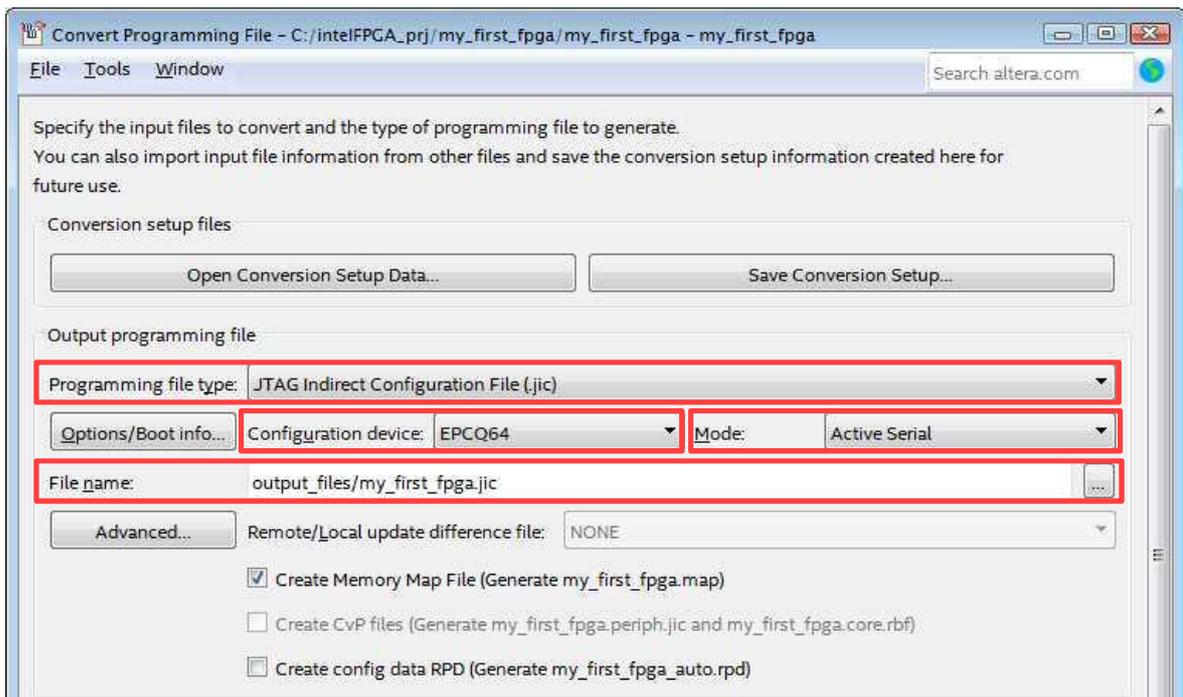
2-6-2-1. プログラミング・ファイルを作成する

使用する開発ボードに応じたコンフィギュレーション・デバイス用のプログラミング・ファイルを sof ファイルから生成します。

Board	Configuration Device	File Format
MAX 10 FPGA Evaluation Kit	MAX 10 内蔵 CFM	pof
Cyclone 10 LP Evaluation Kit	EPCQ64	jic
Beryll Cyclone V GX Base Board	EPCS128	jic

なお、MAX 10 用の pof ファイルは、コンパイル実行時に同時生成されていますので、MAX 10 FPGA Evaluation Kit ユーザは [2-6-2-2. プログラミングする](#) へ進んでください。

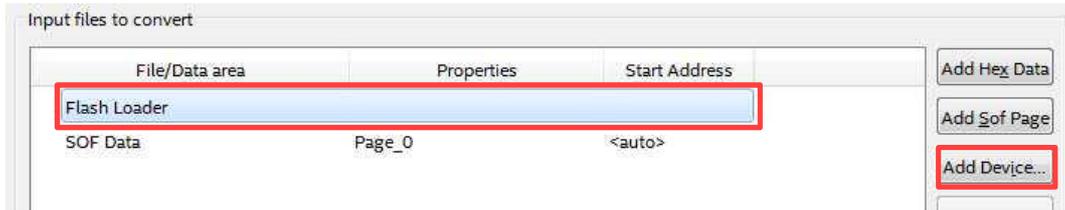
1. File メニュー ➤ Convert Programming Files を起動します。
2. Output programming file 枠内を設定します。



設定名	内容
Programming file type	JTAG Indirect Configuration File (.jic)
Configuration device	<i>Cyclone 10 LP Evaluation Kit の場合</i> : EPCQ64 <i>Beryll Cyclone V GX Base Board の場合</i> : EPCS128
Mode	Active Serial
File name	output_files/my_first_fpga.jic

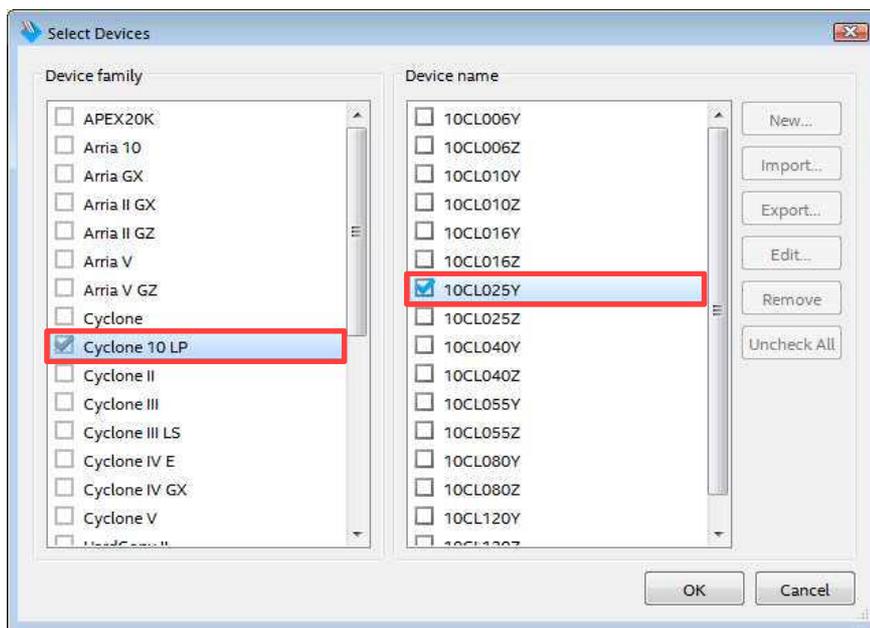
3. Input files to convert 枠内を設定します。

① Flash Loader 行をハイライト選択し、Add Device ボタンをクリックします。

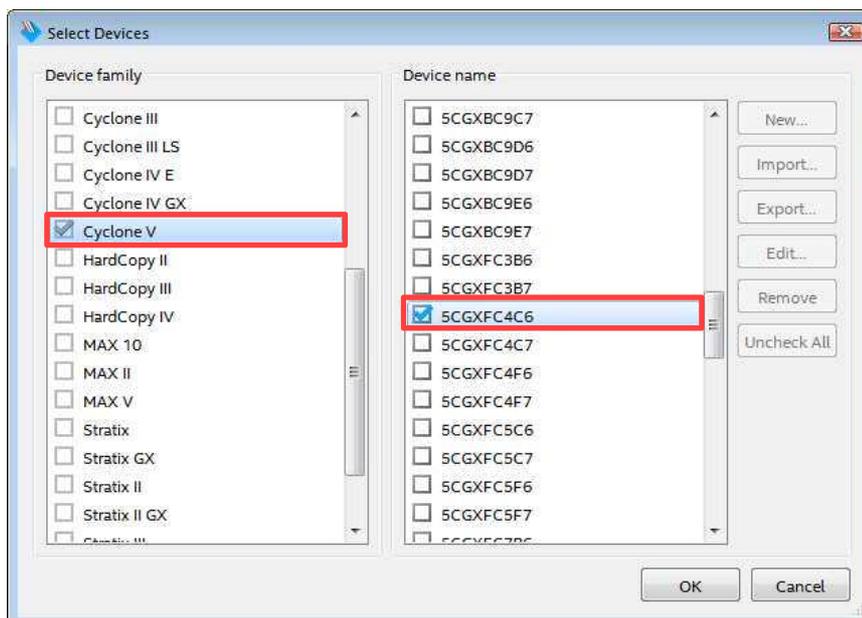


② Select Devices ダイアログボックスから、開発ボードに応じたデバイス・ファミリとデバイス名を選択し、OK ボタンをクリックします。

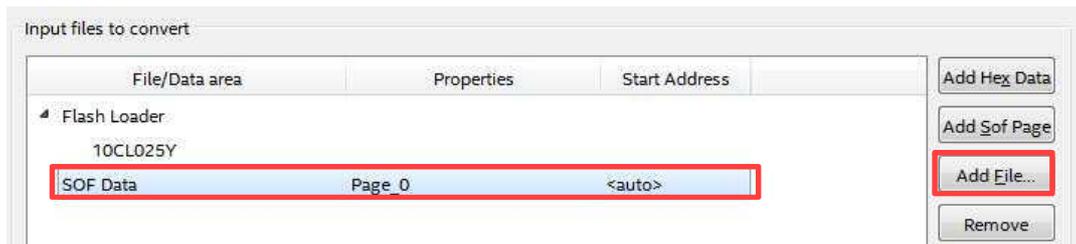
▼ Cyclone 10 LP Evaluation Kit の場合



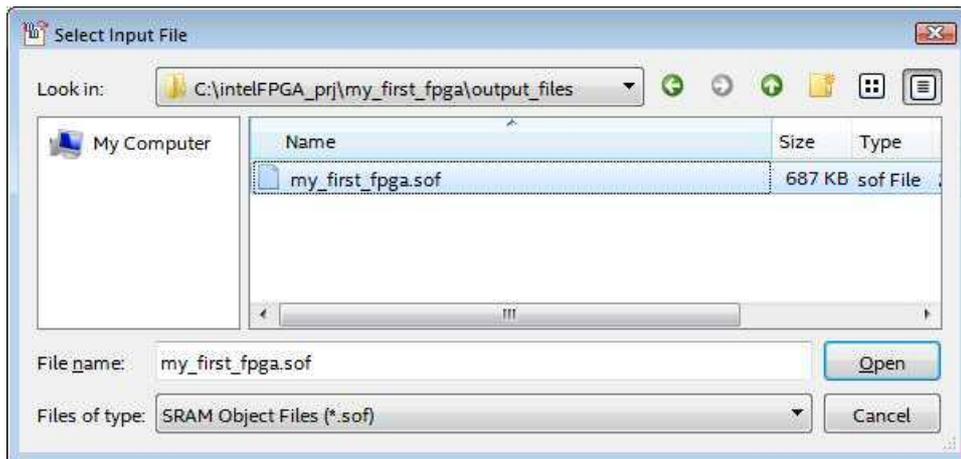
▼ Beryll Cyclone V GX Base Board の場合



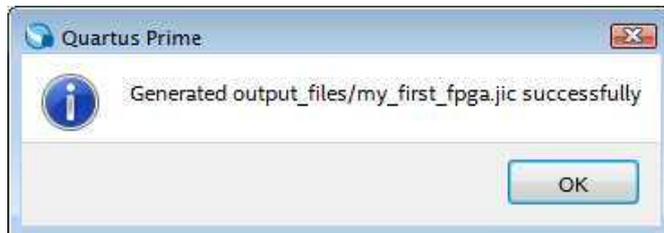
- ③ SOF Data 行をハイライト選択し、Add File ボタンをクリックします。



- ④ 変換元の my_first_fpga.sof ファイルを指定します。



4. Generate ボタンをクリックして JIC ファイルを生成させます。



5. Close ボタンをクリックし、Convert Programming File ダイアログボックスを閉じます。

以上で、コンフィギュレーション・デバイス用のプログラミング・ファイルが完成しました。

2-6-2-2. プログラミングする

1. ダウンロード・ケーブルとパソコンを接続します。

Note: ・ 開発ボードの JTAG ヘッダの位置は、各開発ボードのユーザ・ガイドやスキマティックをご確認ください。

- MAX 10 FPGA Evaluation Kit の場合

インテル FPGA ダウンロード・ケーブル II と USB ケーブルを接続し、インテル FPGA ダウンロード・ケーブル II を開発ボードの JTAG ヘッダに接続します。その後 USB ケーブルのもう一方をプログラミングに使用するパソコンの USB ポートに接続します。

- Cyclone 10 LP Evaluation Kit の場合

ここでの作業はありません。次のステップへ進んでください。

(Cyclone 10 LP Evaluation Kit は、ダウンロード・ケーブルと電源供給ケーブルが同じ USB ケーブルで賄われています。)

- Beryll Cyclone V GX Base Board の場合

付属の USB ケーブルで開発ボードとパソコンを接続します。開発ボード側のコネクタ位置は、本開発ボードの Getting Started をご覧ください。

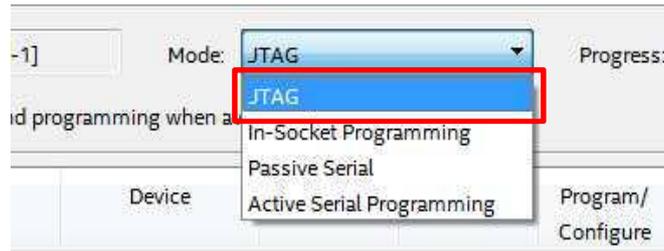
- 開発ボードと電源ケーブルを接続します。(開発ボードによっては、電源ケーブルを接続すると電源が供給されます。)電源スイッチがある開発ボードの場合は、スイッチを入れて電源を供給します。
- Tools メニュー ➤ Programmer により、Programmer ウィンドウを起動します。
- Hardware Setup ボタン(Programmer 画面の左上)をクリックします。
- Hardware Settings タブの Currently selected hardware 項のプルダウン・リストから使用するハードウェアを選択し、Close ボタンをクリックします。



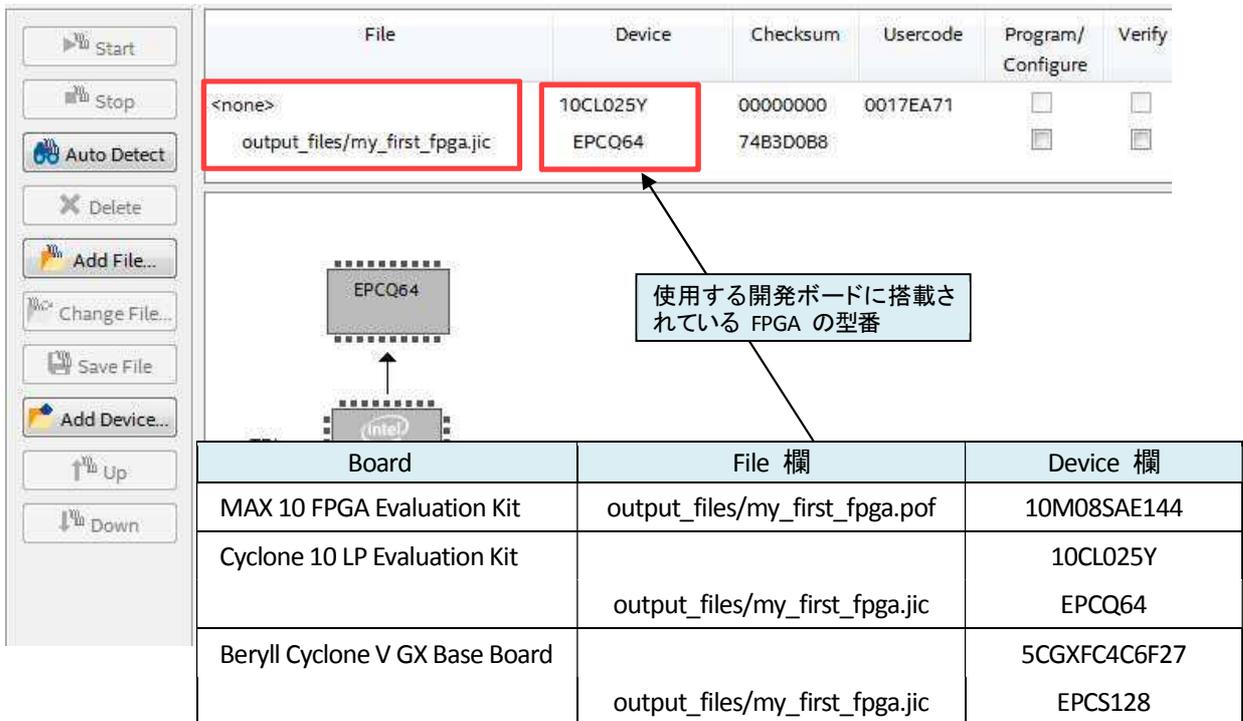
Board	Currently selected hardware に検出される名前
MAX 10 FPGA Evaluation Kit	使用するダウンロード・ケーブル名
Cyclone 10 LP Evaluation Kit	Cyclone 10 LP Evaluation Kit
Beryll Cyclone V GX Base Board	USB-Blaster

- Mode(Programmer 画面の右上)のプルダウン・リストより、デバイスへの書き込みを実行する手法を選択します。

このチュートリアルでは、JTAG を選択します。



7. Programmer 画面上部の File 欄に my_first_fpga.sof ファイルがすでに登録されている場合には、sof ファイルをハイライト選択し Delete ボタンをクリックして削除(空欄)してください。
8. Add File ボタンをクリックし、作業ディレクトリ内の output_files フォルダ内に生成されている my_first_fpga.pof あるいは my_first_fpga.jic ファイルを選択し、Open ボタンをクリックして File 欄に登録します。



Board	File 欄	Device 欄
MAX 10 FPGA Evaluation Kit	output_files/my_first_fpga.pof	10M08SAE144
Cyclone 10 LP Evaluation Kit	output_files/my_first_fpga.jic	10CL025Y EPCQ64
Beryll Cyclone V GX Base Board	output_files/my_first_fpga.jic	5CGXFC4C6F27 EPCS128

9. my_first_fpga.pof あるいは my_first_fpga.jic ファイル行の右側にあるプログラミング・オプションにおいて、実行する項目にチェックを入れます。

ここでは コンフィギュレーション・デバイスにプログラミングするので、Program/Configure オプションにチェックを入れてください。

- MAX 10 FPGA Evaluation Kit の場合は、pof ファイルの下にある CFM0 側にチェックをしてください。



- Cyclone 10 LP Evaluation Kit / Beryll Cyclone V GX Base Board の場合は、jic 側にチェックしてください。自動的に上側の枠にもチェックが入力され、File 欄には “Factory default enhanced SFL image” が表示されます。



10. Start ボタンをクリックし、データの書き込みをスタートさせます。

データのダウンロードが開始されます。Progress バーが 100% になり、Messages ウィンドウ (System タブ) に Info: Successfully performed operation(s) のメッセージを確認できたら書き込み完了です。

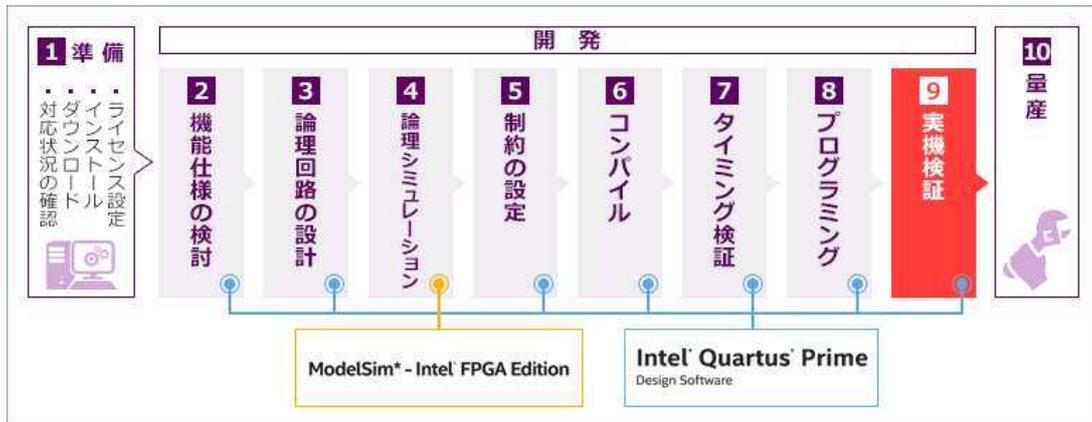
関連情報: 下記ドキュメントおよび WEB コンテンツをご参考ください。

[「Quartus Prime はじめてガイド - デバイス・プログラミングの方法」](#)

[FPGA 経由で EPCQ デバイスへプログラミング \(JIC プログラミング\)](#)

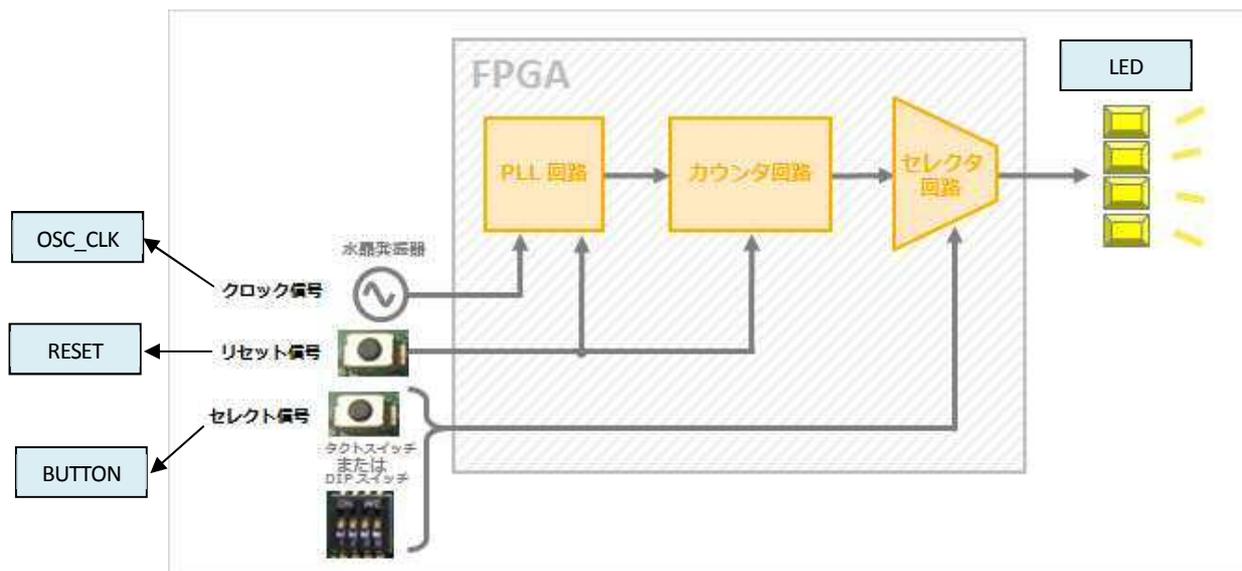
2-7. 実機検証

実機(ボード)上で FPGA が正しく動作しているかを確認します。(この作業は、開発フローの [9] にあたります。)



チュートリアルでは、L チカ(LED チカチカ)動作を作成しました。

すでに開発ボード上の LED が点灯しているはずです。セレクト信号を割り当てたピンに接続されているタクトスイッチを押す、あるいは DIP スイッチを切り替えると、点灯する速さが変わります。



<FPGA の動作仕様>

- LED が点灯および点滅します。
- BUTTON に割り当てたピンに接続されているタクトスイッチを押している間、LED の点滅および点滅する速さが変わり(遅くなります)。
- RESET に割り当てたピンに接続されているタクトスイッチを押している間、PLL 回路とカウンタ回路がリセットされ、LED が全消灯します。

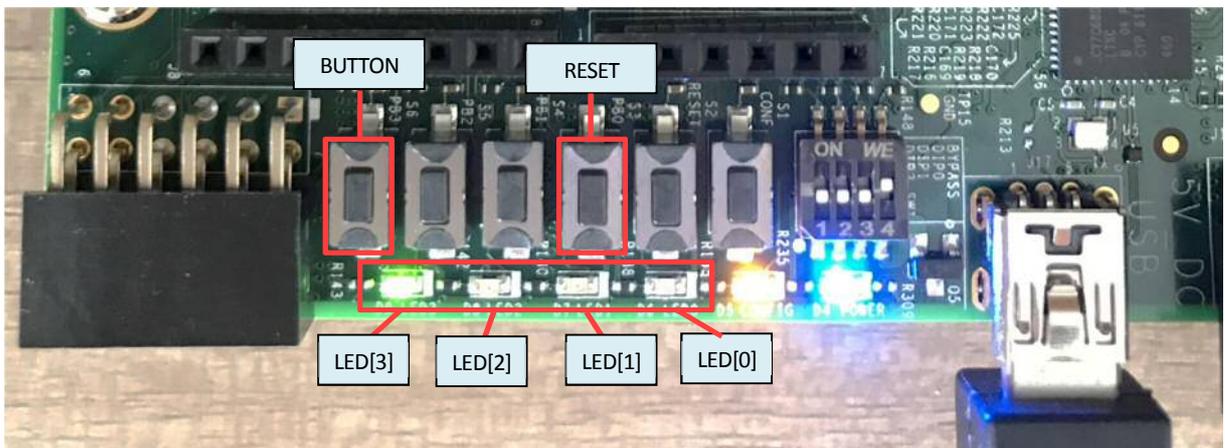
L チカは成功しましたか？

なお、各開発ボードのスイッチや LED の位置は、このとおりです。

▼ MAX 10 FPGA Evaluation Kit の場合



▼ Cyclone 10 LP Evaluation Kit の場合



▼ Beryll Cyclone V GX Base Board の場合



以上で、FPGA 開発における Quartus Prime の基本的な操作は終了です。

3. ステップアップ

FPGA や CPLD を勉強するために便利な WEB サイトをご紹介します。

サイト名	概要
インテル PSG ホームページ	メーカーのホームページです。 英語 日本語
FPGA 入門 (インテル PSG HP より)	FPGA について分かりやすく紹介しています。
CPLD 入門 (インテル PSG HP より)	CPLD について分かりやすく紹介しています。
はじめての CPLD 設計 (インテル PSG HP より)	CPLD 設計に役立つデザイン例
サポート情報 (インテル PSG HP より)	開発ソフトウェアやトランシーバ、外部メモリ・インタフェースなど、カテゴリ別に情報がまとめられたページです。
オンライン資料 (インテル PSG HP より)	デバイスやツールのデータシートやユーザ・ガイドなどが閲覧できます。PDFとしてダウンロードすることも可能です。
ダウンロード・センター (インテル PSG HP より)	インテル FPGA 開発に必要なソフトウェア各種がダウンロードできます。
アルテラ フォーラム	他の FPGA ユーザに質問をしたり、問題を解決したり、アイデアを共有したりできます。
アルテラ Wiki	組み込み設計者がデザイン・ガイドやツール、デザイン例等を共有できるコミュニティです。
Design Store (インテル PSG HP より)	インテル FPGA を使い始めるのに役立つサンプル・デザインの検索ページです。 ★関連情報★ 【TIPS】 サンプル・デザインを無料で手に入れよう
インストラクター・トレーニング (インテル PSG HP より)	インテル 公式 FPGA テクニカル・トレーニングです。講師がレクチャーするハンズオン・トレーニングです。[有料]
オンライン・トレーニング (インテル PSG HP より)	インテル公式オンライン・トレーニングです。インテル FPGA 製品に関する各種トレーニングをいつでもどこでも受講できます。[無料] ★関連情報★ 【TIPS】 インテル(旧アルテラ) FPGA テクニカル・トレーニングを無料で受講しよう
MACNICA フォーラム	マクニカ在籍のエンジニアに質問したり、メンバー同士のアイデアや意見交換ができます。投稿および閲覧の際は、カテゴリより “Intel FPGA (旧 ALTERA)” を選択してご利用ください。 ※ はじめにお読みください

この情報は下記 WEB コンテンツでも公開しています。

[ほんとのほんとの導入編 その5. FPGA 役立つコンテンツの紹介](#)

改版履歴

Revision	年月	概要
1	2018年7月	初版

免責およびご利用上の注意

弊社より資料を入手されましたお客様におかれましては、下記の使用上の注意を一読いただいた上でご使用ください。

1. 本資料は非売品です。許可無く転売することや無断複製することを禁じます。
2. 本資料は予告なく変更することがあります。
3. 本資料の作成には万全を期していますが、万一ご不明な点や誤り、記載漏れなどお気づきの点がありましたら、本資料を入手されました下記代理店までご一報いただければ幸いです。
株式会社マクニカ アルティマ カンパニー <https://www.alt.macnica.co.jp/> 技術情報サイト アルティマ技術データベース <http://www.altima.jp/members/>
4. 本資料で取り扱っている回路、技術、プログラムに関して運用した結果の影響については、責任を負いかねますのであらかじめご了承ください。
5. 本資料は製品を利用する際の補助的な資料です。製品をご使用になる際は、各メーカー発行の英語版の資料もあわせてご利用ください。