

Verilogの基礎その1

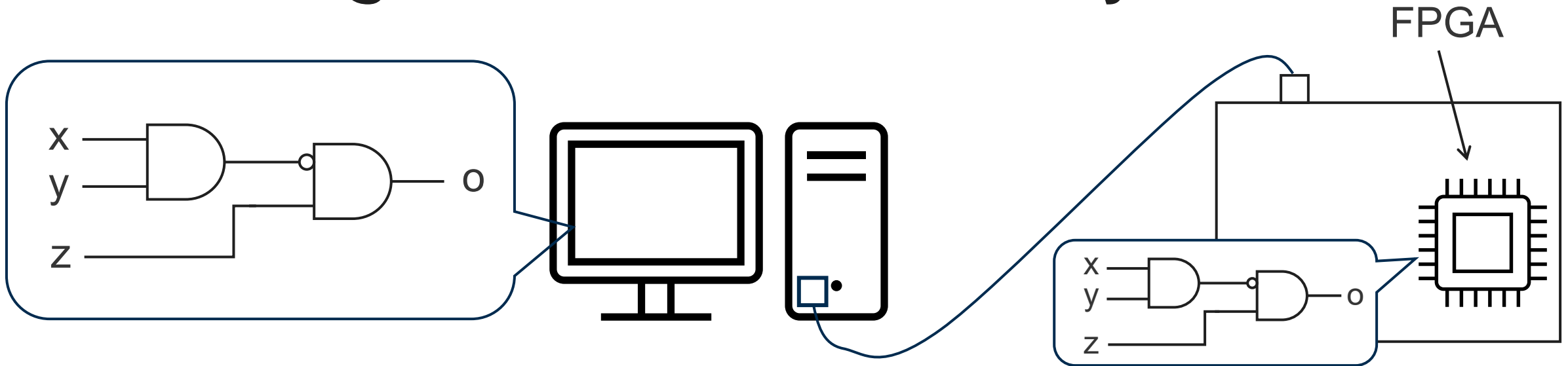
ISHI会

フルデジタルで音声を操るためのロジック回路 1回目

目次

- FPGAについて
- ボードの準備
- スイッチとLEDを使った回路を通したEfinix FPGAの開発方法
- 基本ゲートのVerilog記述
- 今日の練習問題

Field Programmable Gate Array

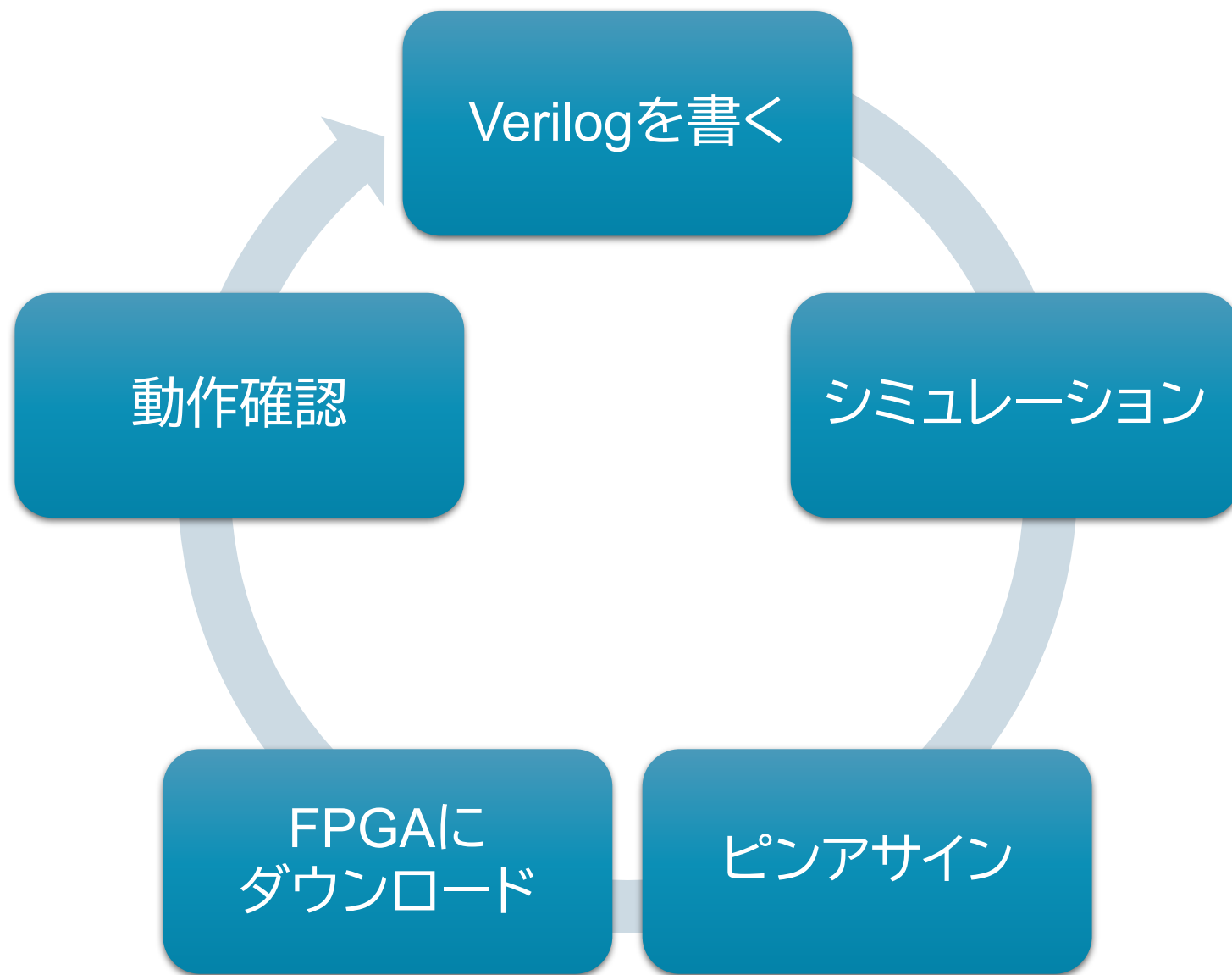


Field Programmable Gate Array(FPGA) はコンピュータでロジック回路を記述して、自由に回路を書き換えることができる集積回路(IC)。

Field (現場)でProgrammable(書き込める)特性を有している。

ICを作るより安価に自由なロジック回路を作り込めるだけでなく、何回でも回路を書き換えることができることから、頻繁に回路の変更が生じる試作や研究開発に広く使われている。

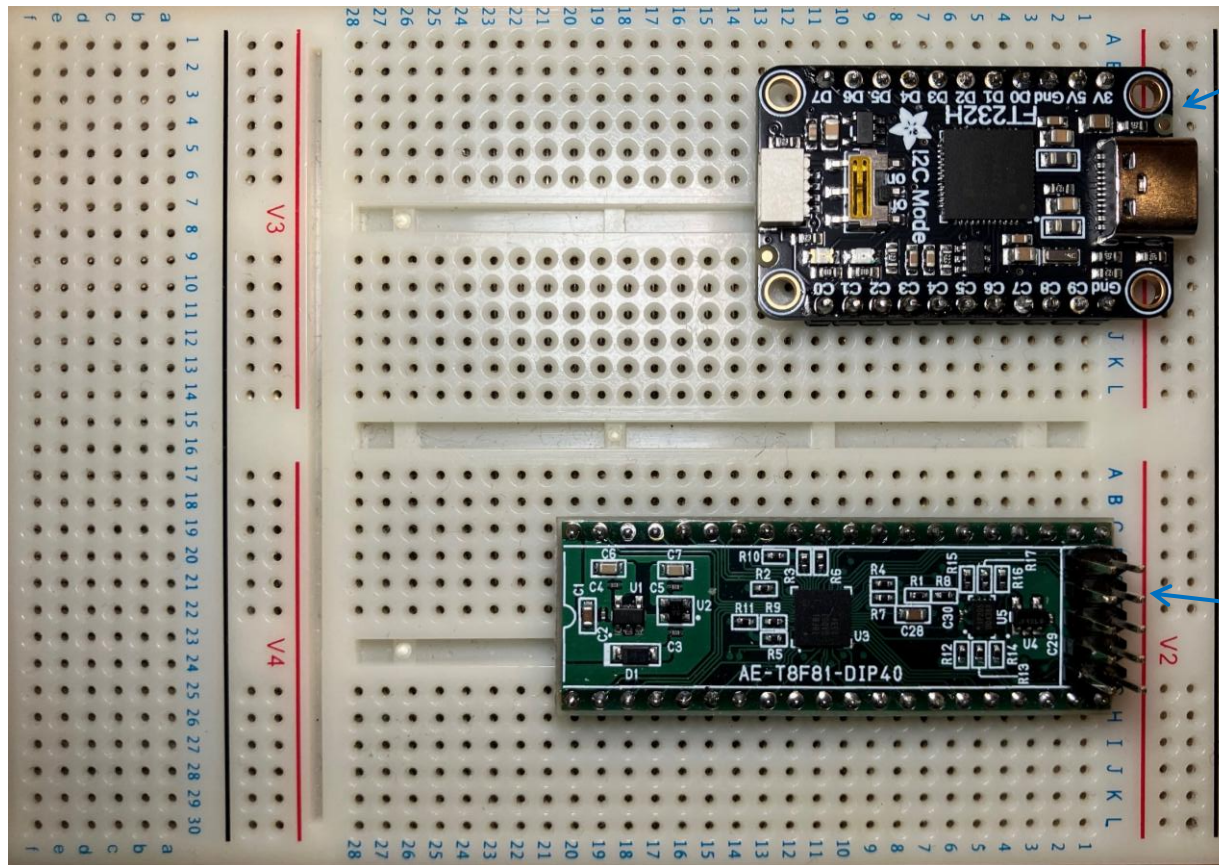
FPGAの開発手順



FPGAボードの準備

FPGAボード開発ボード

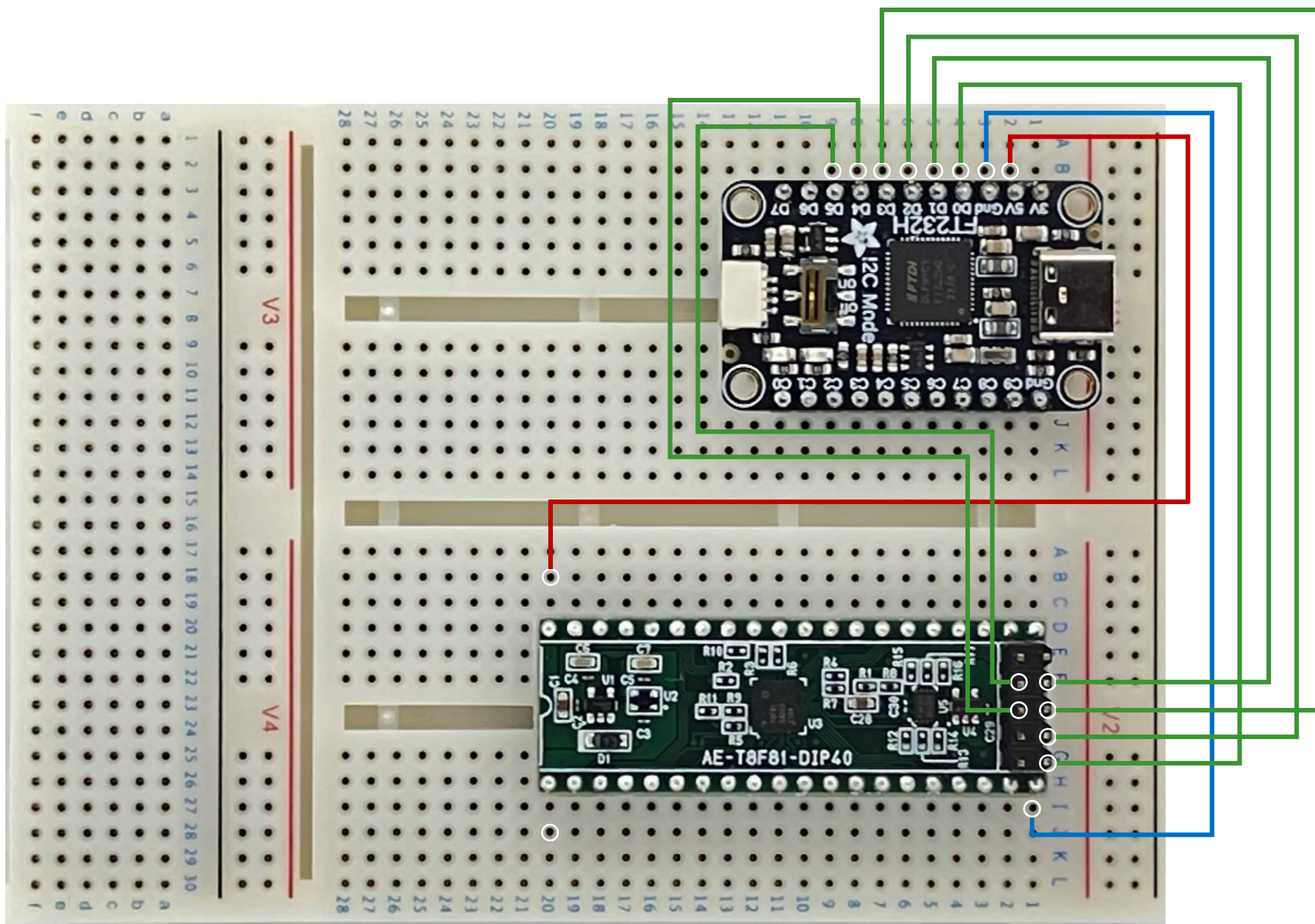
2つの基板が搭載されたブレッドボードをお渡しします。
一方がJTAG書き込み器で、他方がFPGAボードです。



書き込み器(JTAG)

FPGAボード

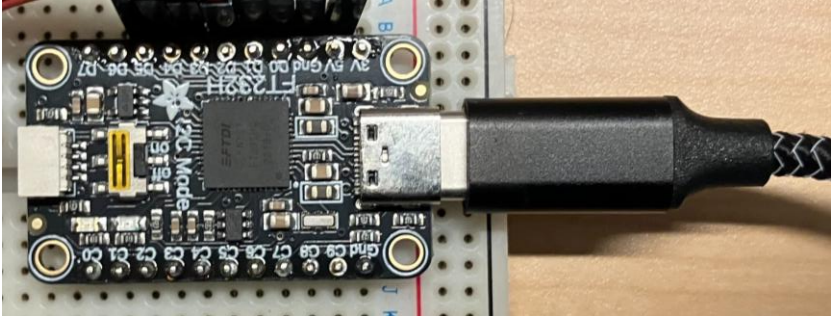
書き込み器とFPGAの接続



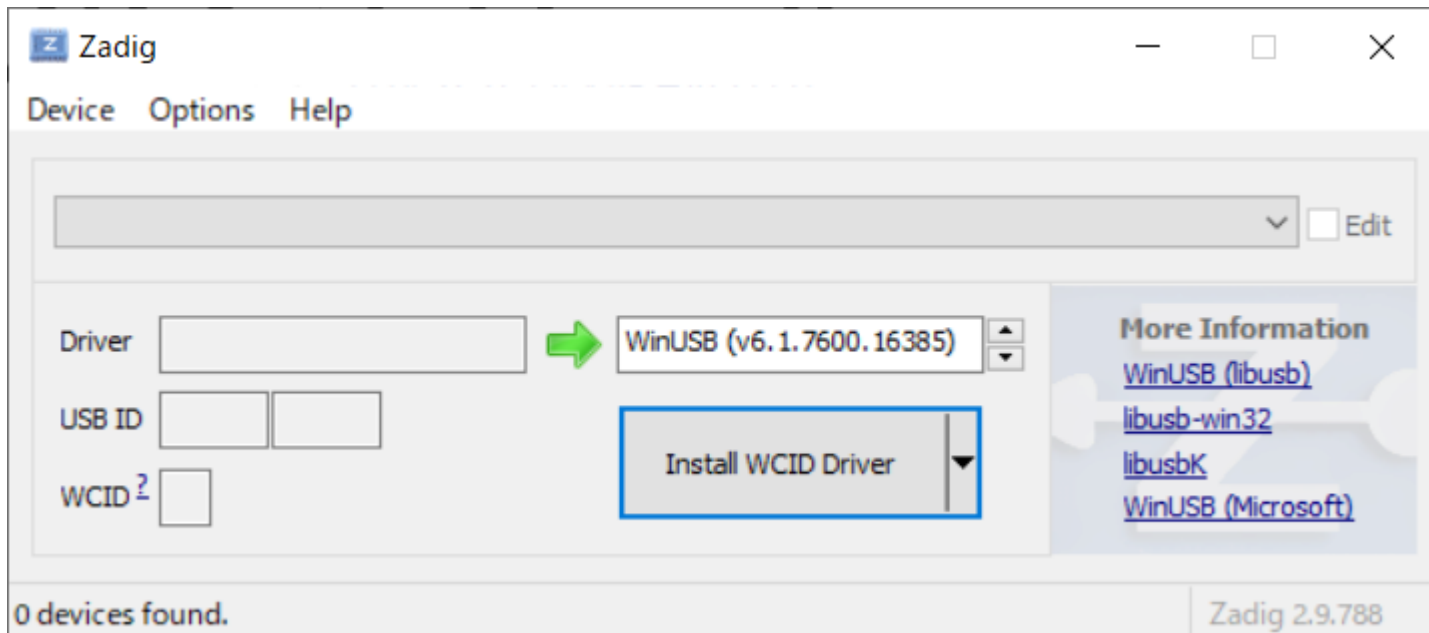
- 5V電源
- GND
- 信号線(JTAG)

ドライバのインストール

書き込み器とPCをUSBケーブルで接続する

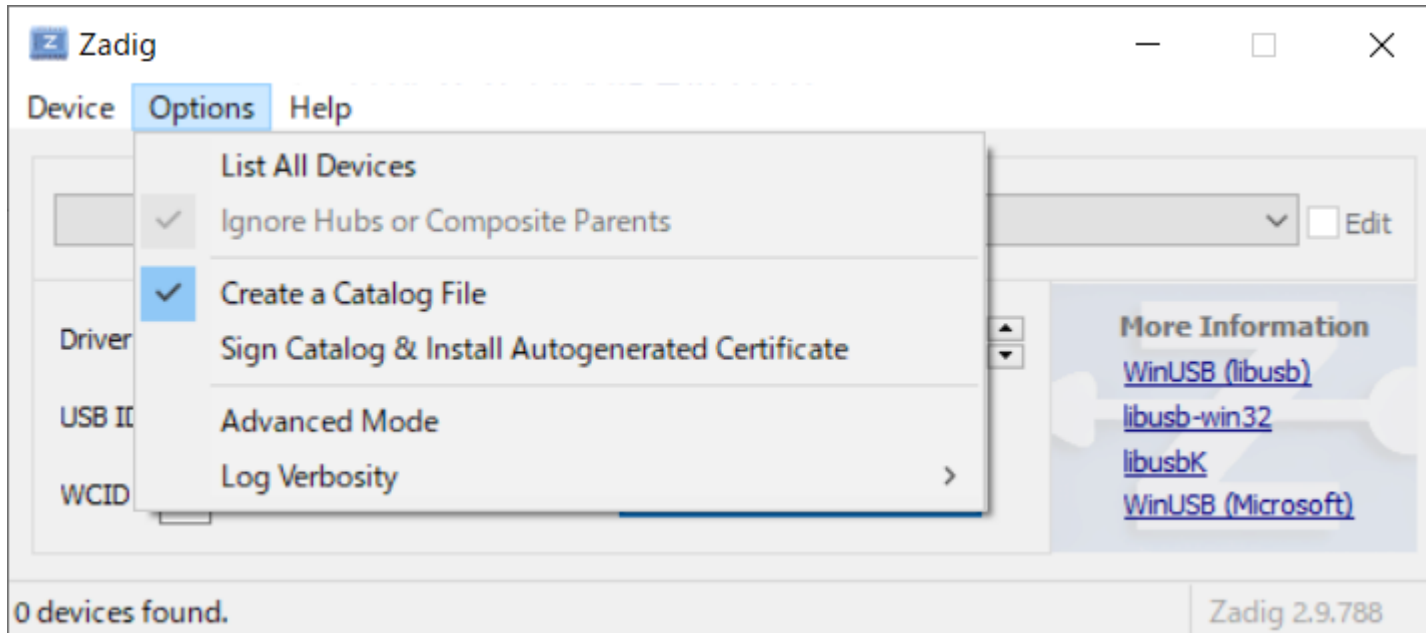


事前準備でダウンロードしたZadigを開く



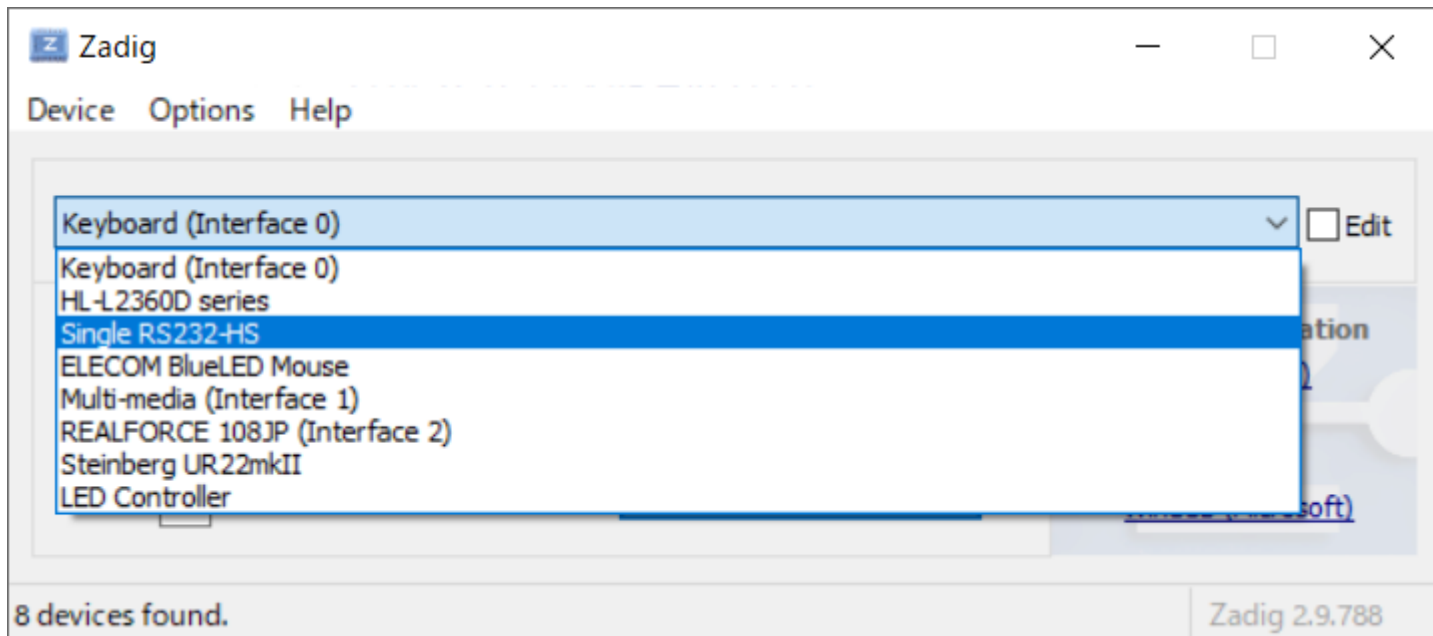
Zadigの操作1

Options→List All Devicesをクリックする



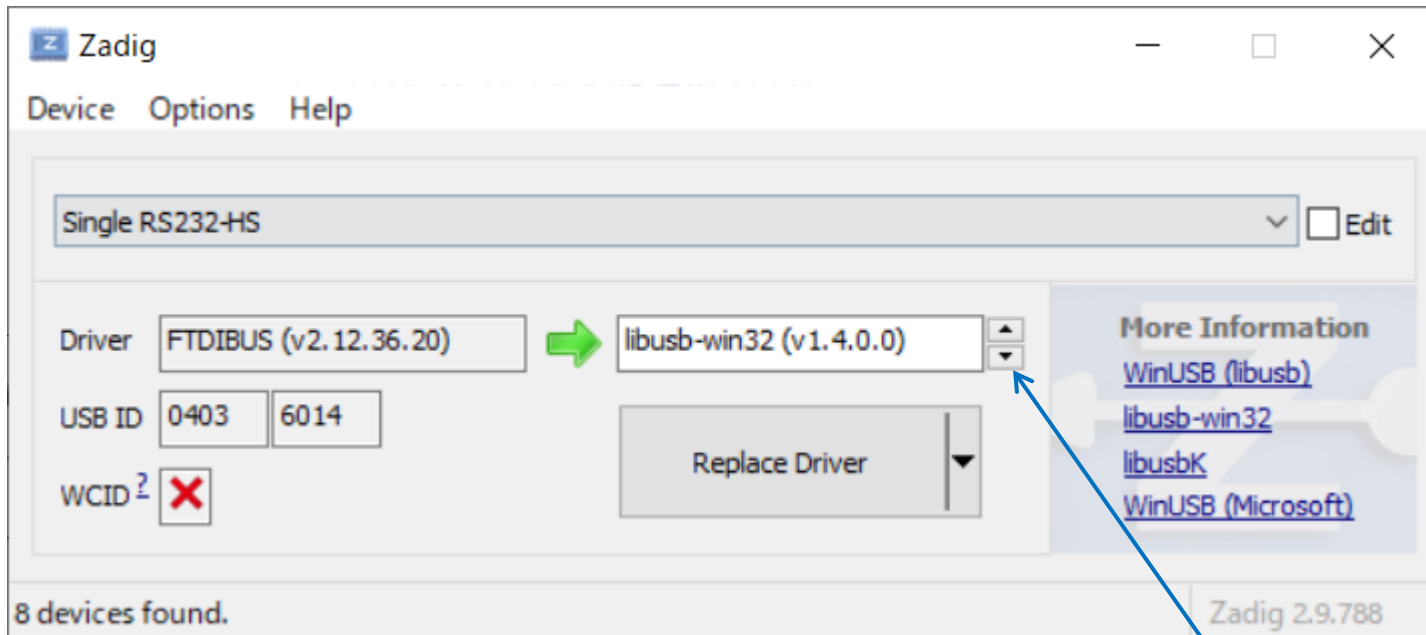
Zadigの操作2

リストから「Single RS232-HS」を選択する。



Zadigの操作3

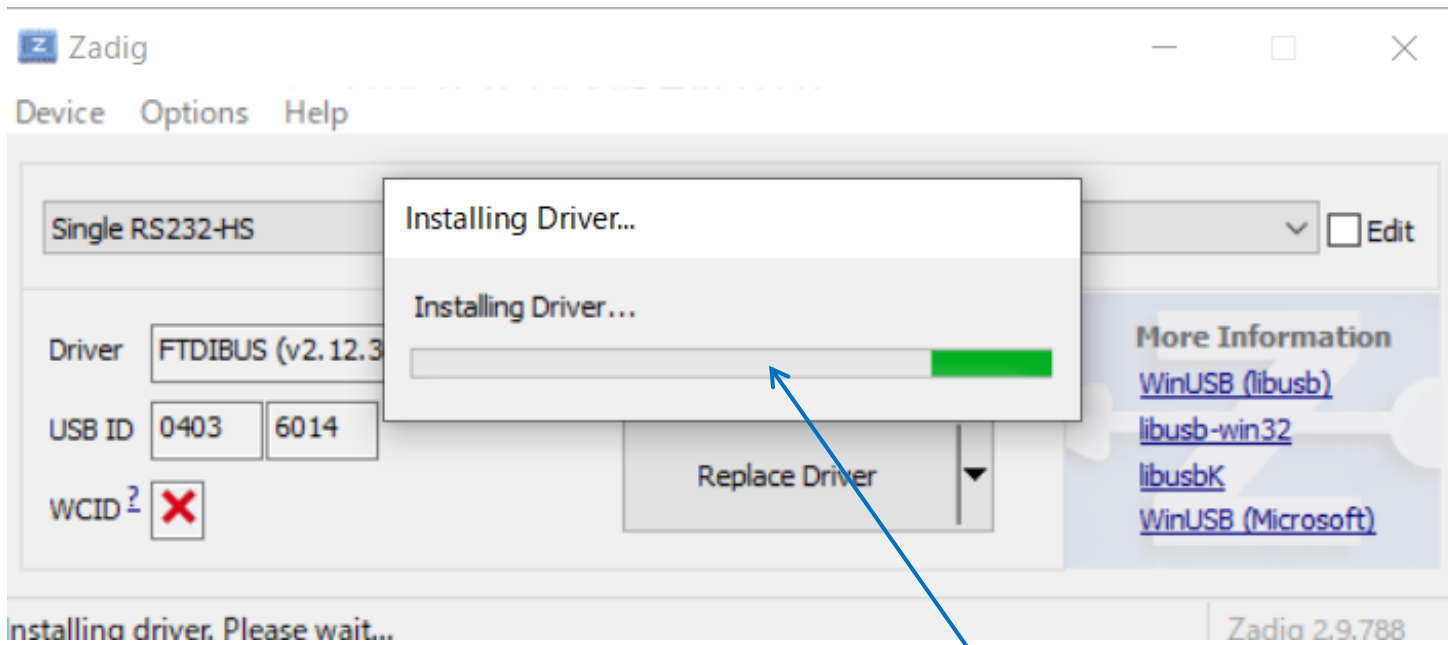
入れ替えるドライバーを「libusb-win32(v1.4.0.0)」を選択する。



矢印でドライバーを入れ替える。

Zadigの操作4

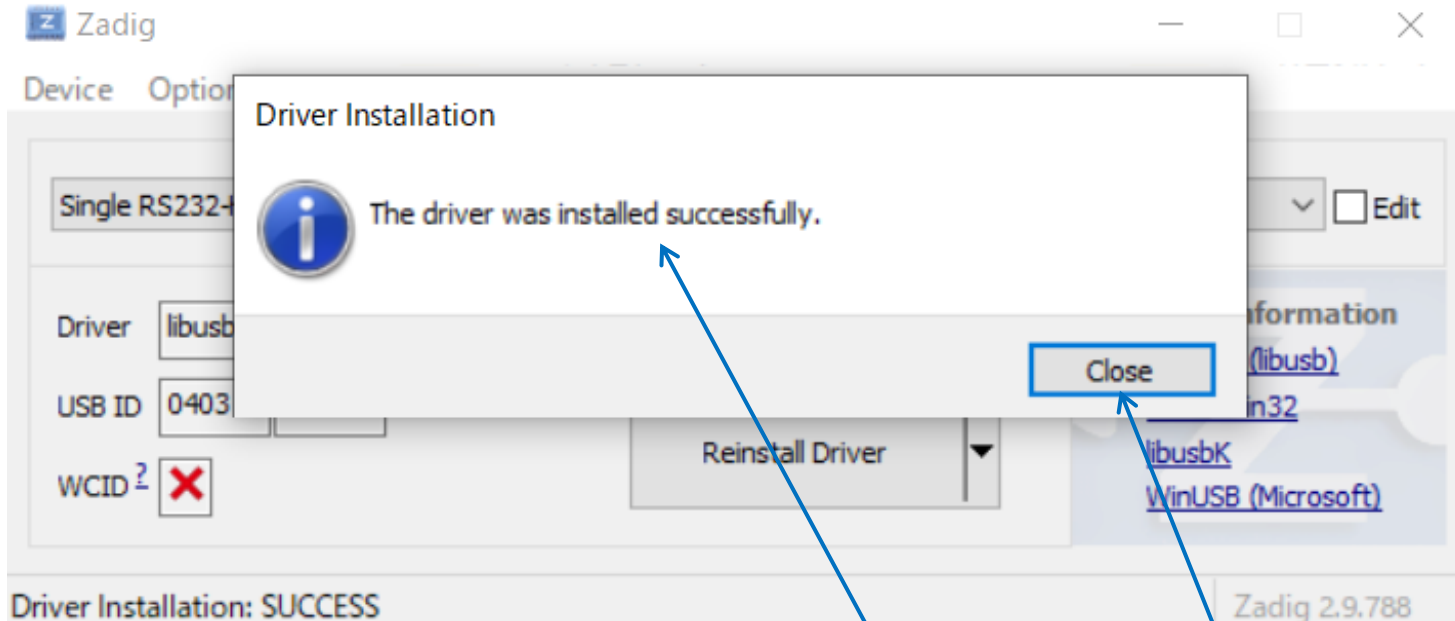
ドライバー入れ替え中



インストールが終わるまで待つ

Zadigの操作5

The driver was installed successfullyとなったらOK
右上Xボタンで終了する。

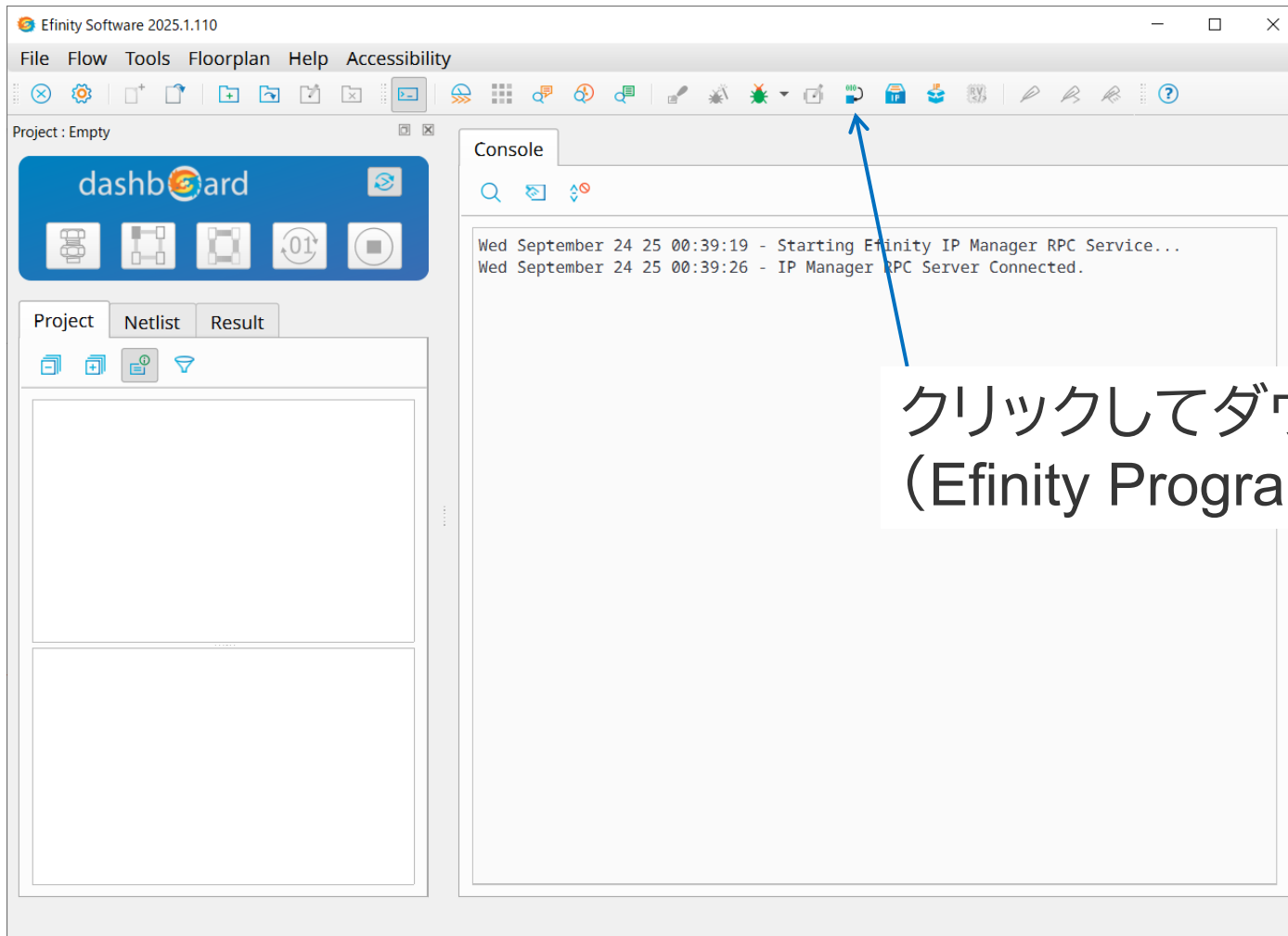


Closeをクリックして終了する

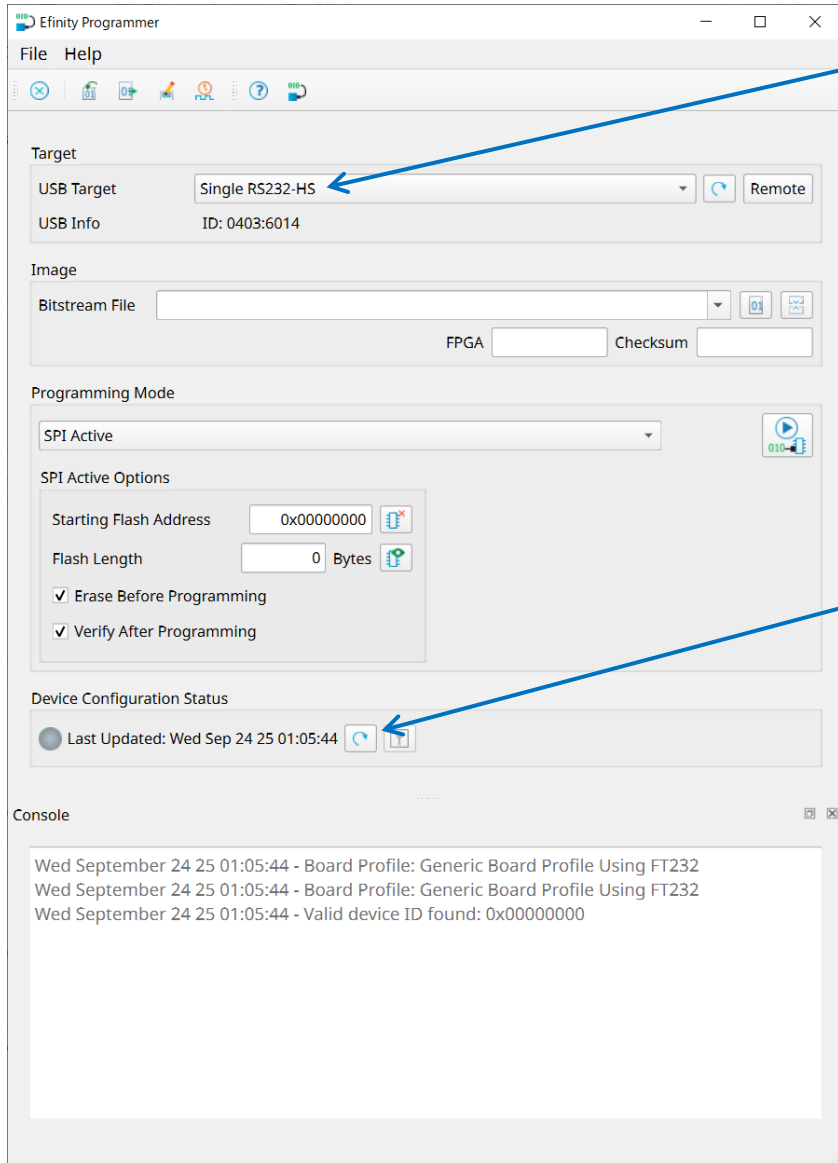
インストールが成功したときの表示

書き込み器とFPGAの接続確認

事前準備でインストールしたEfinityを起動する。



FPGAのチェック

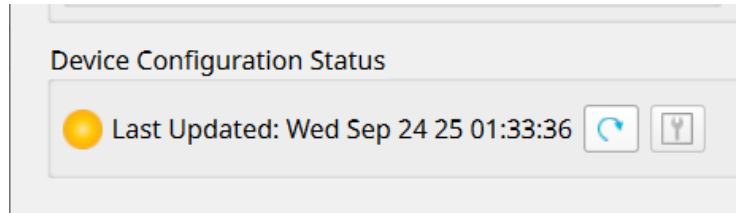


ドライバーが正しく割り当てられている場合、Single RS232-HSが選択される。

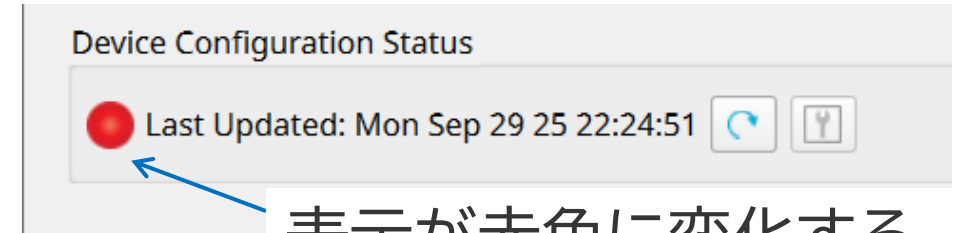
ここをクリックしてFPGAが認識するかチェックする

FPGAの認識

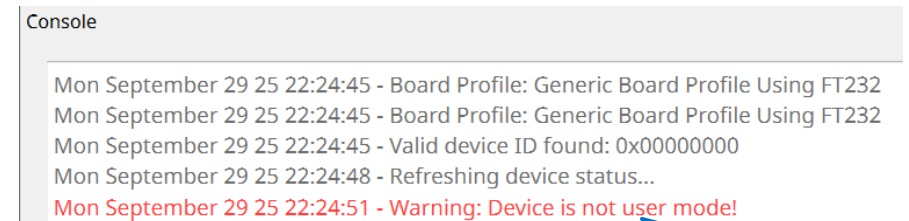
確認中...



認識できた場合

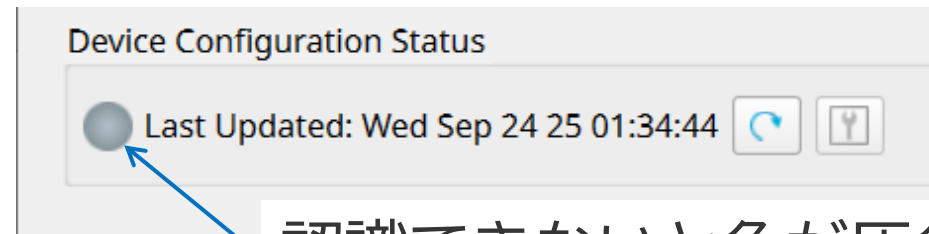


表示が赤色に変化する。



Warning: Device is not user mode!と
なっていればOK

認識できなかった場合



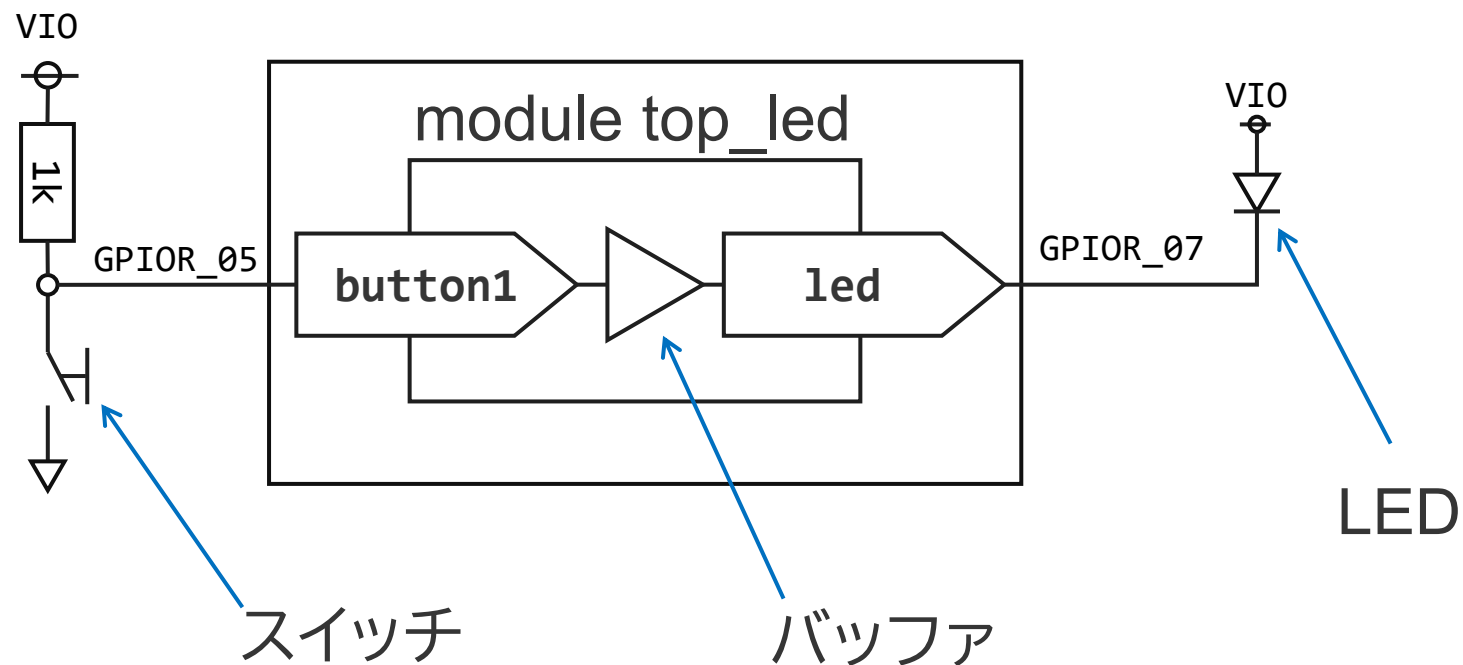
認識できないと色が灰色になる

認識できない場合配線が間違っているなので、よく確認する。

スイッチとLEDの回路と Efinityを用いたFPGA開発

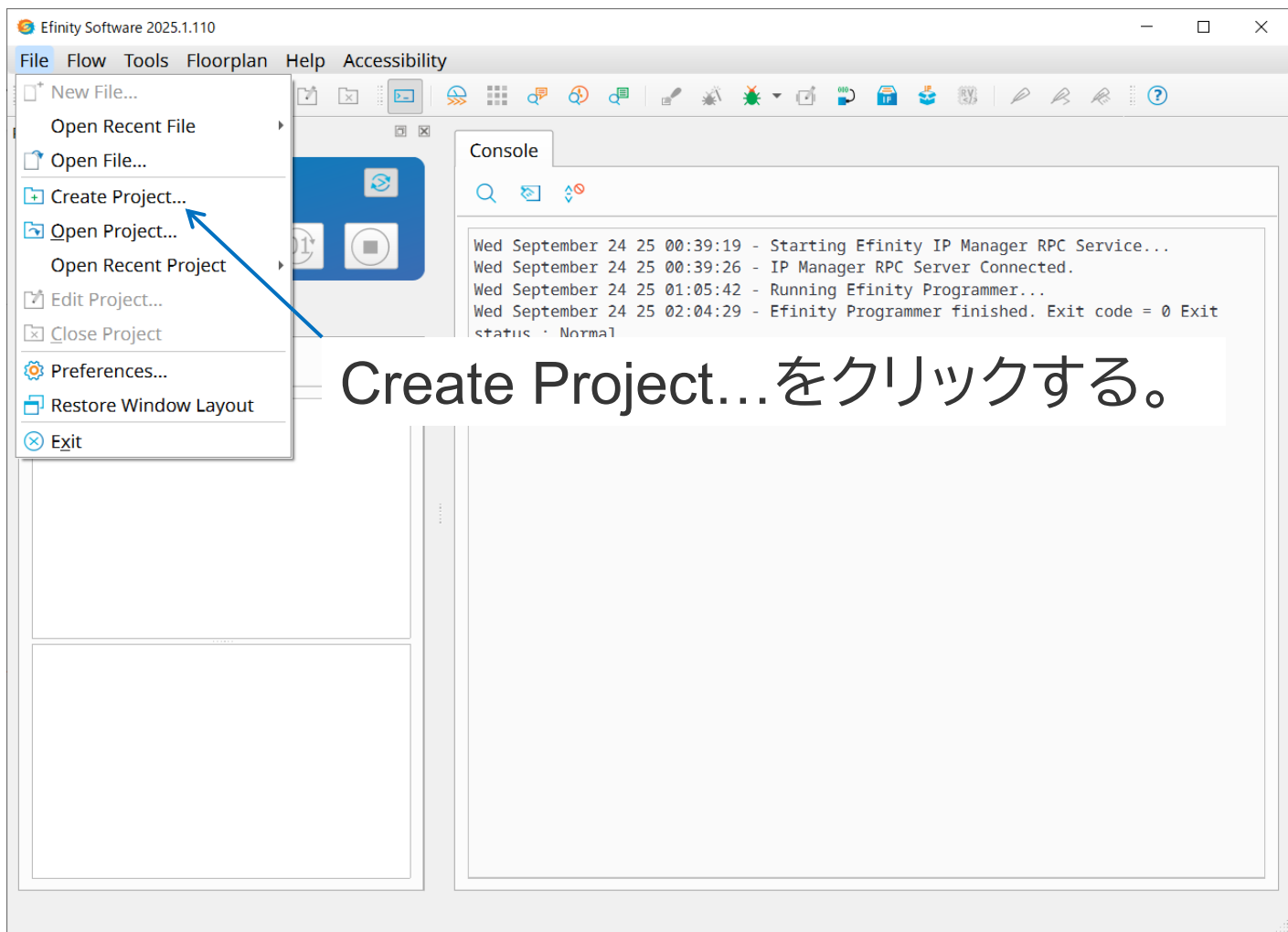
LEDとスイッチ回路

プルアップ抵抗



プロジェクト作成

File→Create Project...をクリックする。



プロジェクトウィザード

Create New Project

Project Design Synthesis Place and Route Bitstream Generation Deb

Name LED

Location C:/Users/Masahiro/Desktop/Efinix/LED

Description

Family Trion

Device T8F81 Select...

Timing Model C2

OK Cancel

Project tab contains compulsory inform

プロジェクト名はフォルダ名が自動でつく

①エクスプローラを開いてプロジェクトフォルダを作る

パスに日本語が入らないようにする

②Torionを選ぶ

③フォルダを作成したらOKをクリック

プロジェクトが開いた状態

この部分が色々表示が変わる

The screenshot shows the Efinity Software 2025.1.110 interface. The main window displays the 'Project' tab for a project named 'LED'. The interface includes a dashboard with various icons, a project tree, and a console window.

The project tree shows the following structure:

- LED
 - Design
 - Constraint
 - Simulation
 - Misc
 - IP

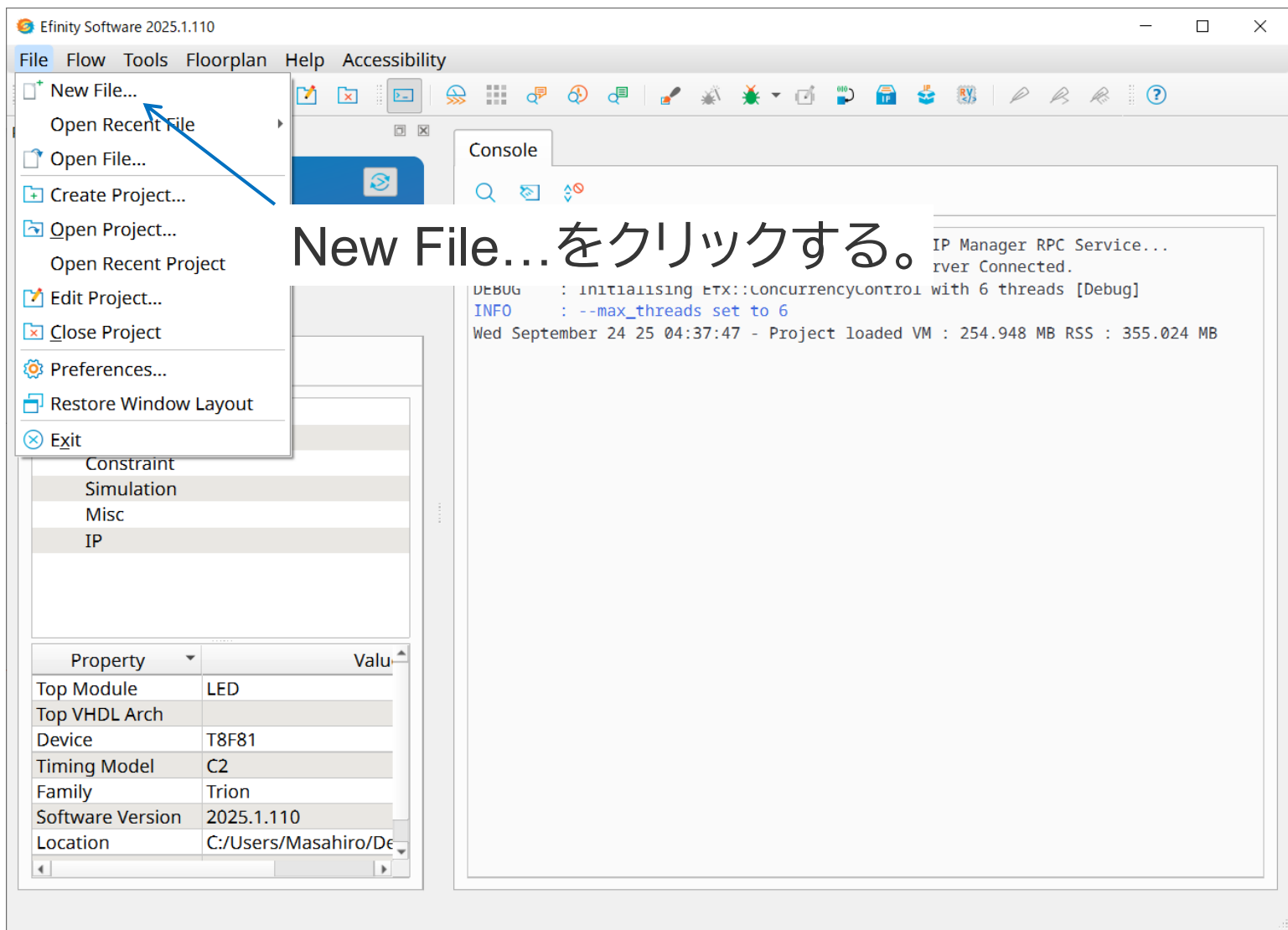
The console window displays the following output:

```
Wed September 24 25 00:39:19 - Starting Efinity IP Manager RPC Service...
Wed September 24 25 00:39:26 - IP Manager RPC Server Connected.
Wed September 24 25 01:05:42 - Running Efinity Programmer...
Wed September 24 25 02:04:29 - Efinity Programmer finished. Exit code = 0 Exit
status : Normal
DEBUG    : Initialising Efx::ConcurrencyControl with 6 threads [Debug]
INFO    : --max_threads set to 6
```

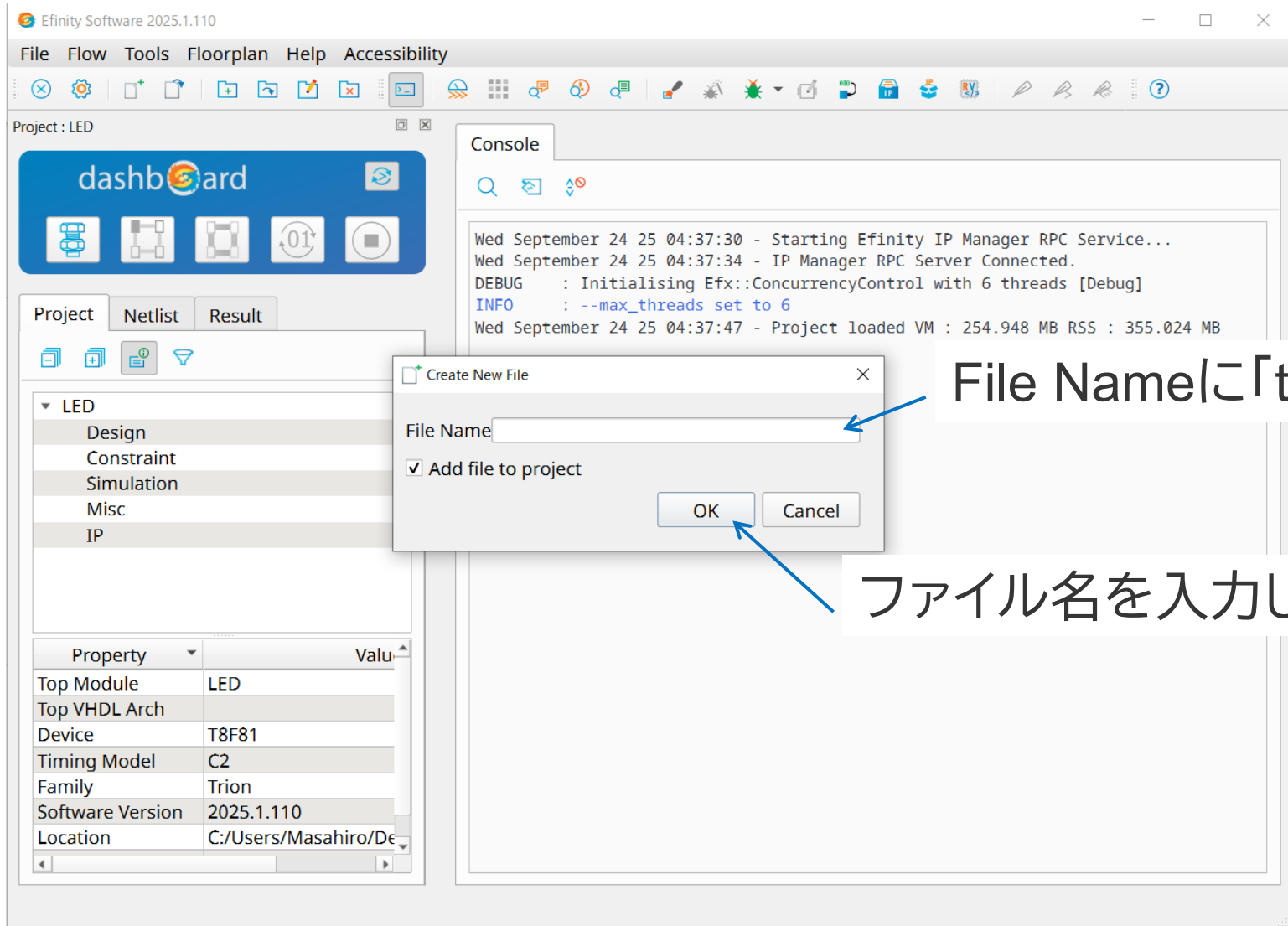
Property	Value
Top Module	LED
Top VHDL Arch	
Device	T8F81
Timing Model	C2
Family	Trion
Software Version	2025.1.110
Location	C:/Users/Masahiro/De

Verilogファイルの追加1

File→New File...をクリックする。



Verilogファイルの追加2



コード入力画面

The screenshot displays the Efinity Software 2025.1.110 interface. The main window is divided into several panes:

- Project: LED**: A sidebar on the left containing a "dashboard" and a project tree. The tree shows "LED" expanded to "Design", with "File: top_led.v (default)" selected. Below the tree is a "Property" table.
- Console**: A central pane showing system logs and debug messages.
- Code Editor**: A pane on the right showing the "top_led.v" file with a single line of code.

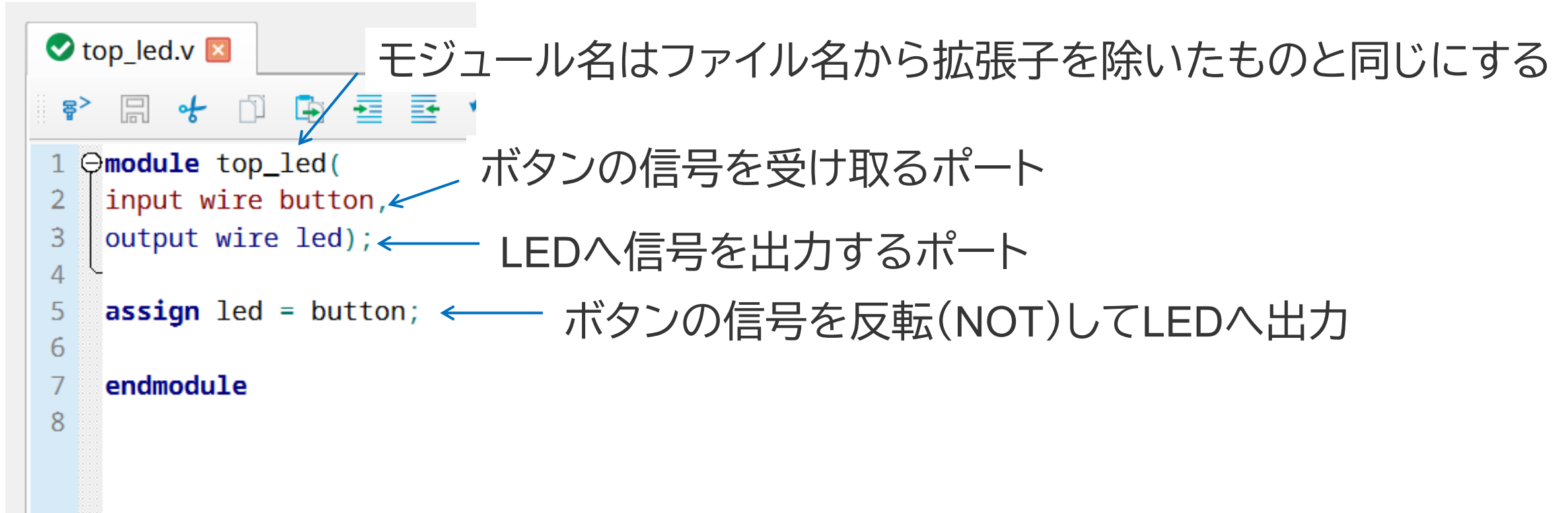
Property	Value
Top Module	LED
Top VHDL Arch	
Device	T8F81
Timing Model	C2
Family	Trion
Software Version	2025.1.110
Location	C:/Users/Masahiro/De

```
Wed September 24 25 04:37:30 - Starting Efinity IP Manager RPC Service...
Wed September 24 25 04:37:34 - IP Manager RPC Server Connected.
DEBUG    : Initialising Efx::ConcurrencyControl with 6 threads [Debug]
INFO     : --max_threads set to 6
Wed September 24 25 04:37:47 - Project loaded VM : 254.948 MB
RSS : 355.024 MB
```

```
1
```

ここにコードを入力する。
真ん中のConsoleを縮めて、
入力画面を広げても良い。

入力するコード



The image shows a screenshot of a Verilog code editor window titled 'top_led.v'. The code is as follows:

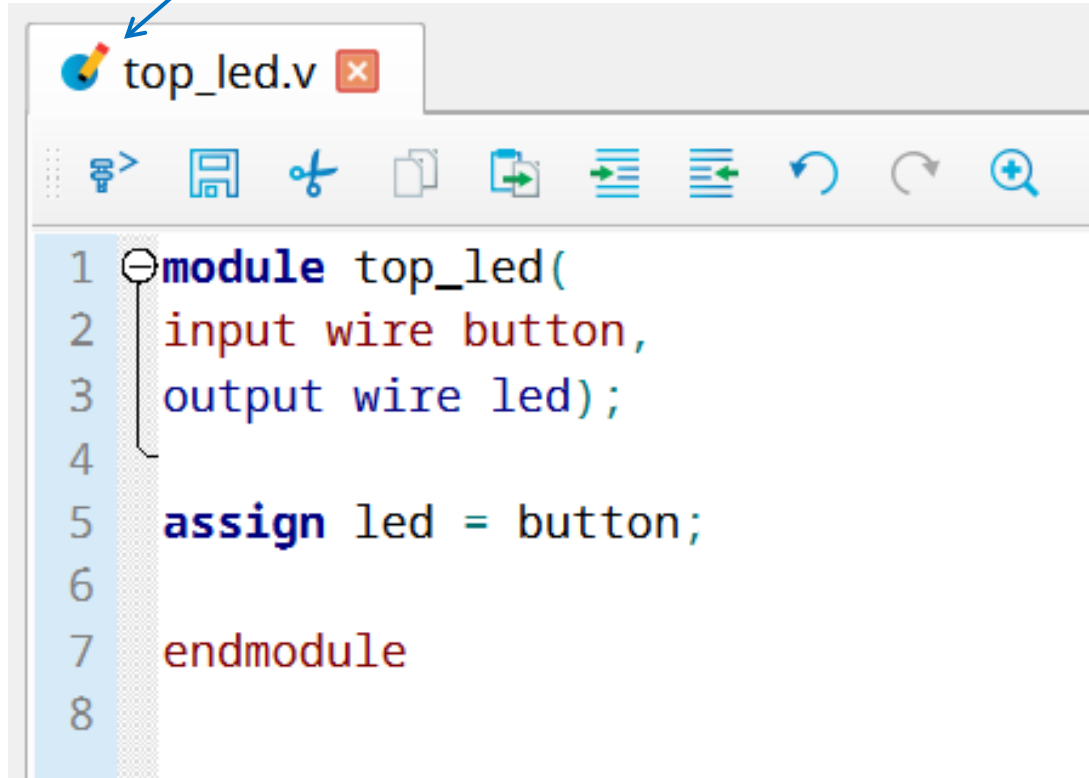
```
1 module top_led(  
2   input wire button,  
3   output wire led);  
4  
5   assign led = button;  
6  
7   endmodule  
8
```

Annotations with arrows point to specific parts of the code:

- An arrow points from the text "モジュール名はファイル名から拡張子を除いたものと同じにする" to the module name 'top_led' in the code.
- An arrow points from the text "ボタンの信号を受け取るポート" to the 'input wire button' declaration.
- An arrow points from the text "LEDへ信号を出力するポート" to the 'output wire led' declaration.
- An arrow points from the text "ボタンの信号を反転(NOT)してLEDへ出力" to the 'assign led = button;' statement.

コードの保存

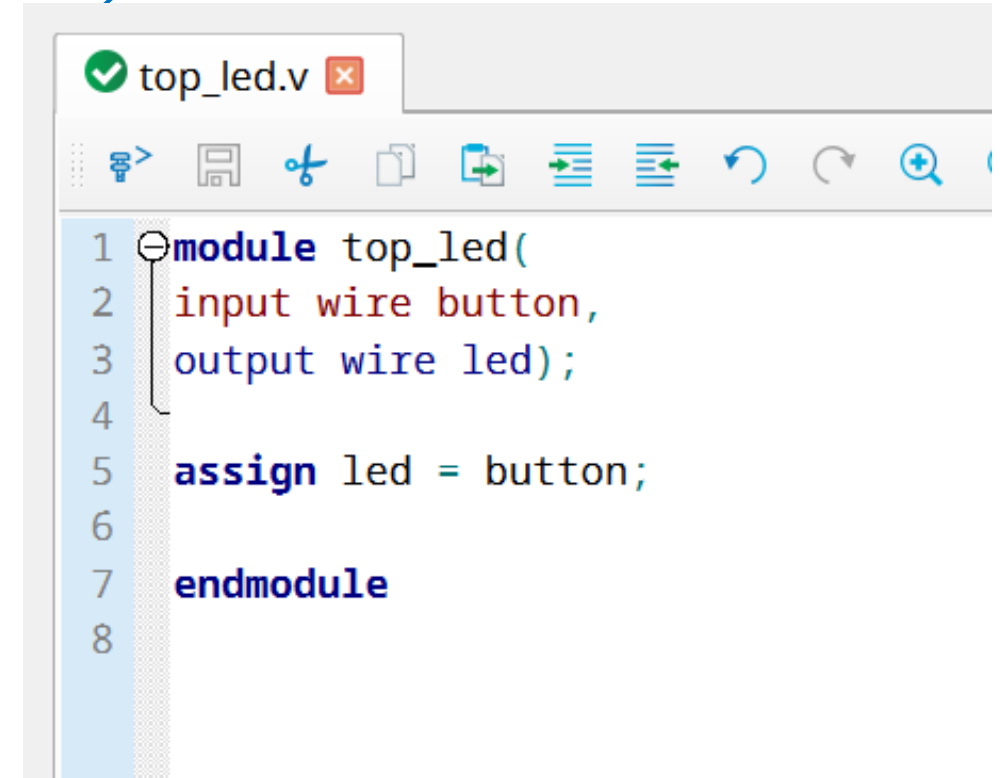
コード保存前は鉛筆アイコン



```
1 module top_led(  
2   input wire button,  
3   output wire led);  
4  
5   assign led = button;  
6  
7   endmodule  
8
```

Ctrl + s

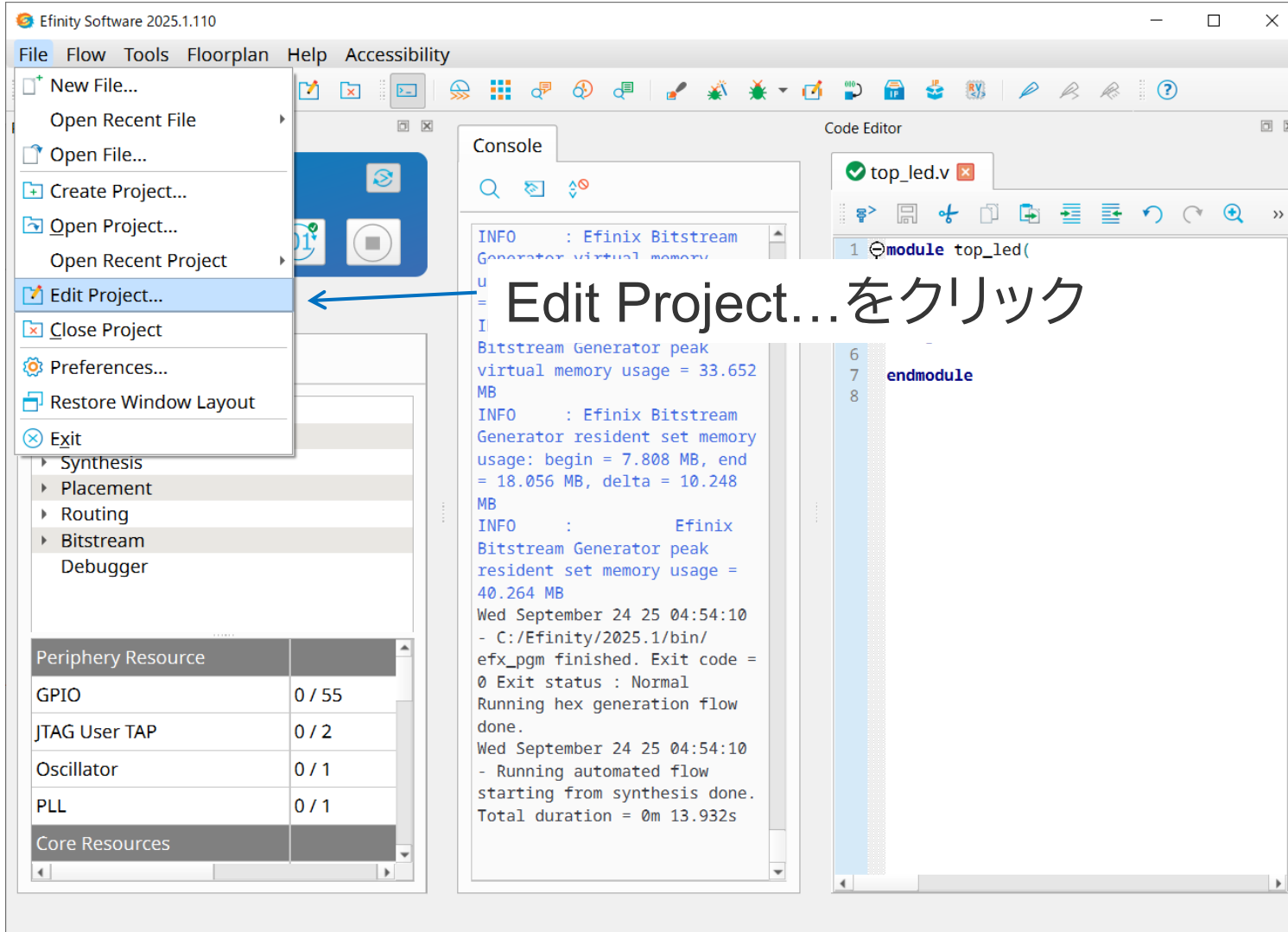
保存前の編集がなければ
チェックアイコンがつく



```
1 module top_led(  
2   input wire button,  
3   output wire led);  
4  
5   assign led = button;  
6  
7   endmodule  
8
```

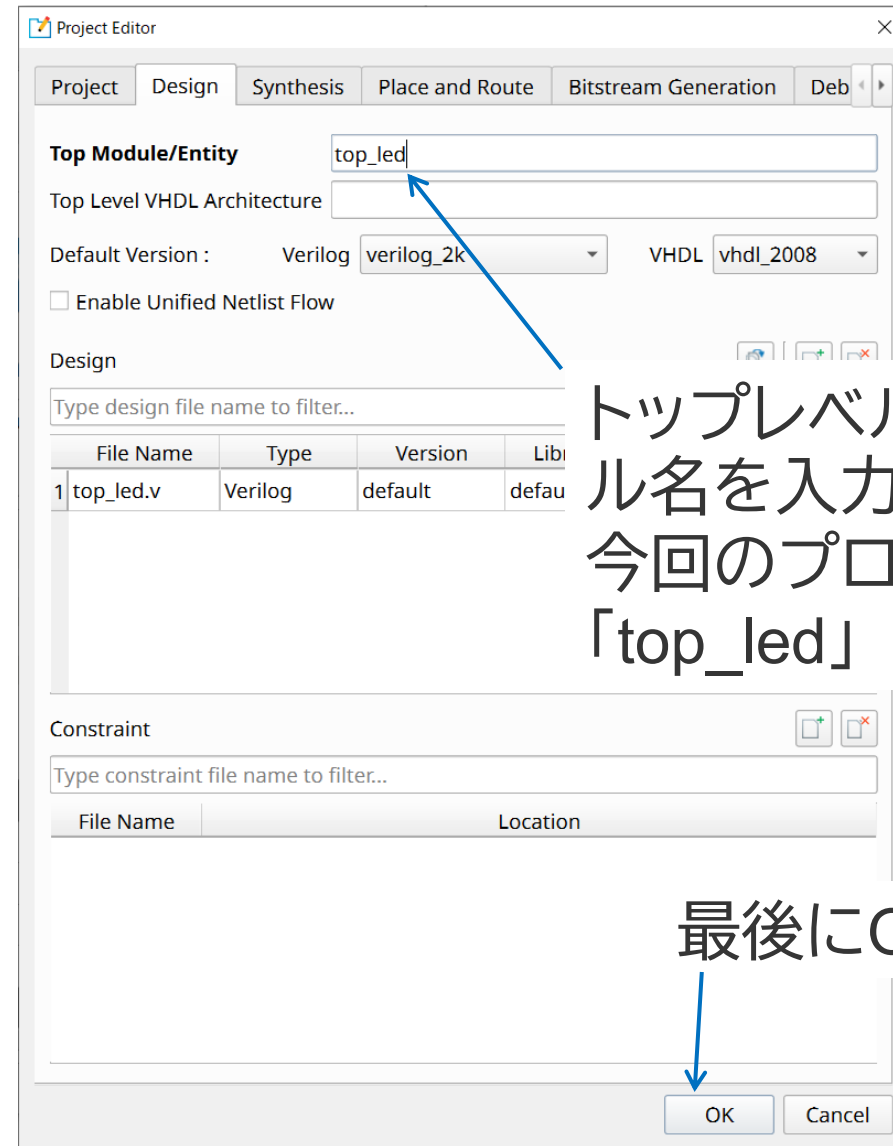
プロジェクト設定の変更(Top Level Moduleの設定)

File→Edit Project...をクリックする。



複数のVerilogコードがあるときに、
最上位になるVerilogコード
(Top level module)を指定する。
Verilogコードが1つであったとしても、
トップレベルモジュールは設定が必要。

Projectの設定



Verilogコードの文法チェック

Project : LED

File Flow Tools Floorplan Help Accessibility

Console

0 Exit status : Normal
Running synthesis flow done.

Code Editor

```
1 module top_led(  
2   input wire button,  
3   output wire led);  
4  
5  
6  
7 endmodule
```

dashboard

Project Netlist Result

LED

- Design
- File : top_led.v (default)
- Constraint
- Simulation
- Misc
- IP

Property	Value
Flow	Synthesis
Path	top_led.v
Exist	Yes
Link	No
Last Changed ...	Wed September 24 2025

Placement and Routing

Wed September 24 25 20:56:03
- Placement and routing flow
working directory is C:/
Users/Masahiro/Desktop/
Efinix/LED/work_pnr

INFO : --max_threads set
to 6

Wed September 24 25 20:56:04
- C:/Efinity/2025.1/
python311/bin/python.exe
"C:/Efinity/2025.1/scripts/
efx_run_pt_unified.py" "LED"
"Trion" "T8F81"

Wed September 24 25 20:56:04
- Running C:/Efinity/2025.1/
python311/bin/python.exe...

Console panel

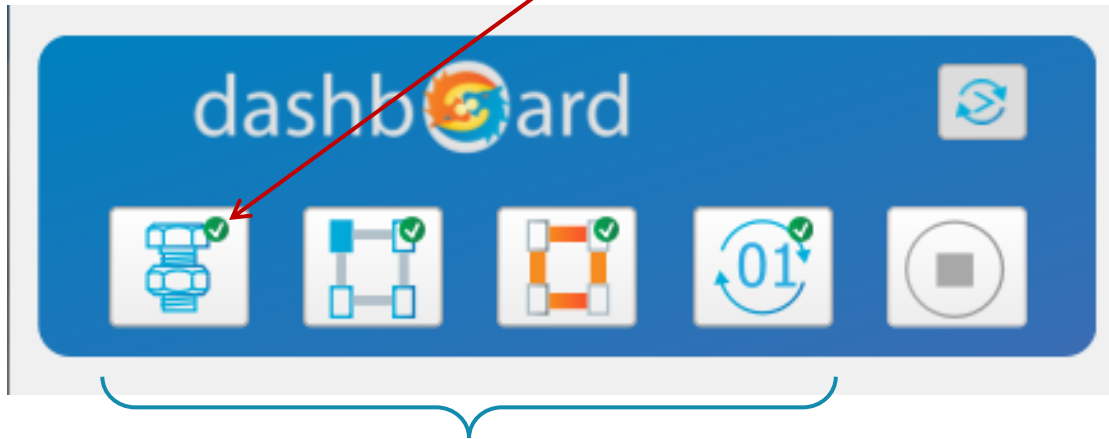
Running automated flow starting from synthesis...placement is in progress...

dashboardのボルトアイコンをクリック

実行後

エラーがない場合

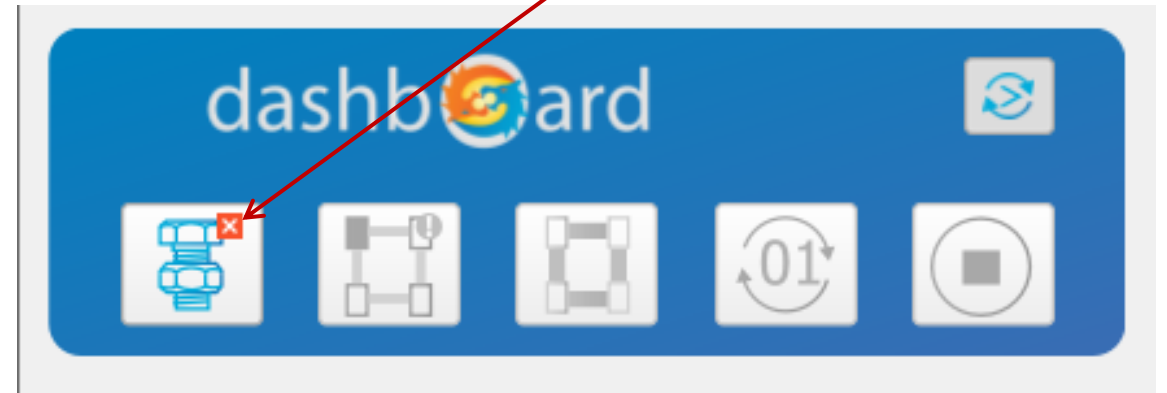
チェックマーク



すべてのアイコンの右上に緑で白抜き
のチェックが付けばOK

エラーがある場合

バツマーク



```

30.00 MB
Wed September 24 25 21:01:28
- C:/Efinity/2025.1/bin/
efx_map finished. Exit code =
3 Exit status : Normal
Running synthesis flow fail.
See exit code and exit
status.
Wed September 24 25 21:01:28
- Running automated flow
starting from synthesis done.
Total duration = 0m 1.887s

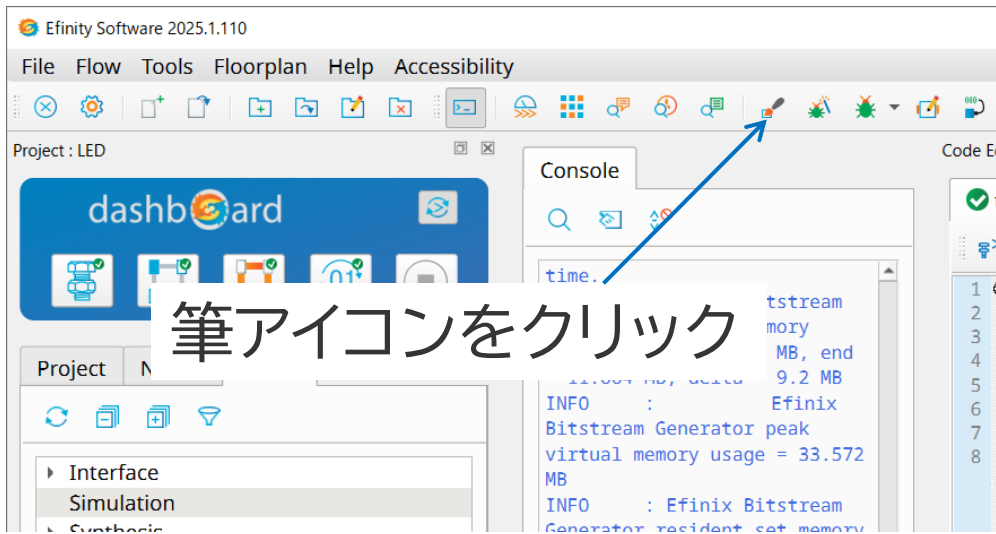
ERROR      : C:
\Users\Masahiro\Desktop\Efiniti\LED\top_led.v(7): syntax
error near 'endmodule'
[VERI-1137]
ERROR      : C:
\Users\Masahiro\Desktop\Efiniti\LED\top_led.v(7): Verilog
2000 keyword 'endmodule' used
in incorrect context
[VERI-2344]

```

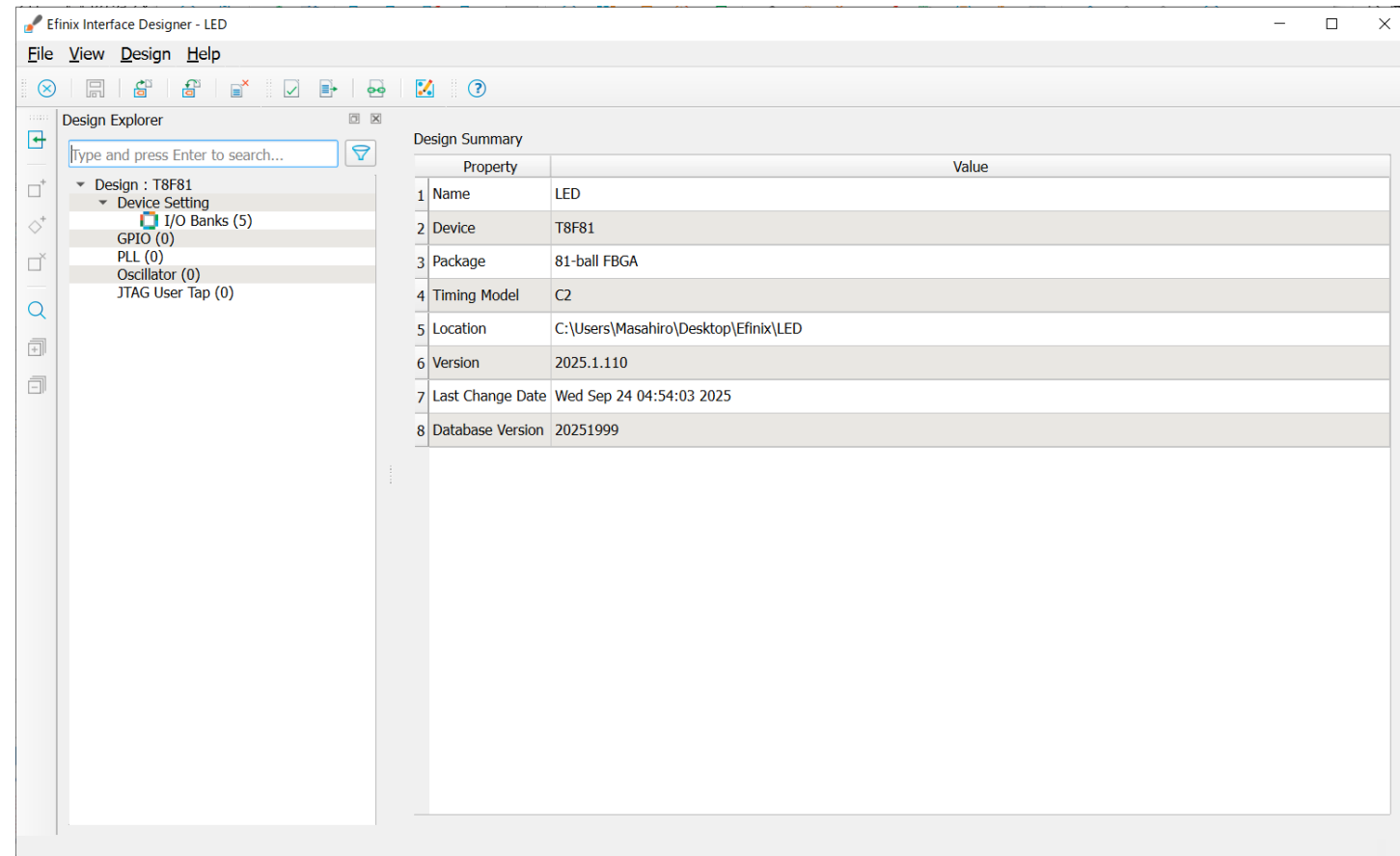
Console画面にエラーの詳細が出るので
よく読む

ピンアサインの手順

Verilogコードのネットリストに宣言したポートをFPGAのピンに割り当てる作業を行う。

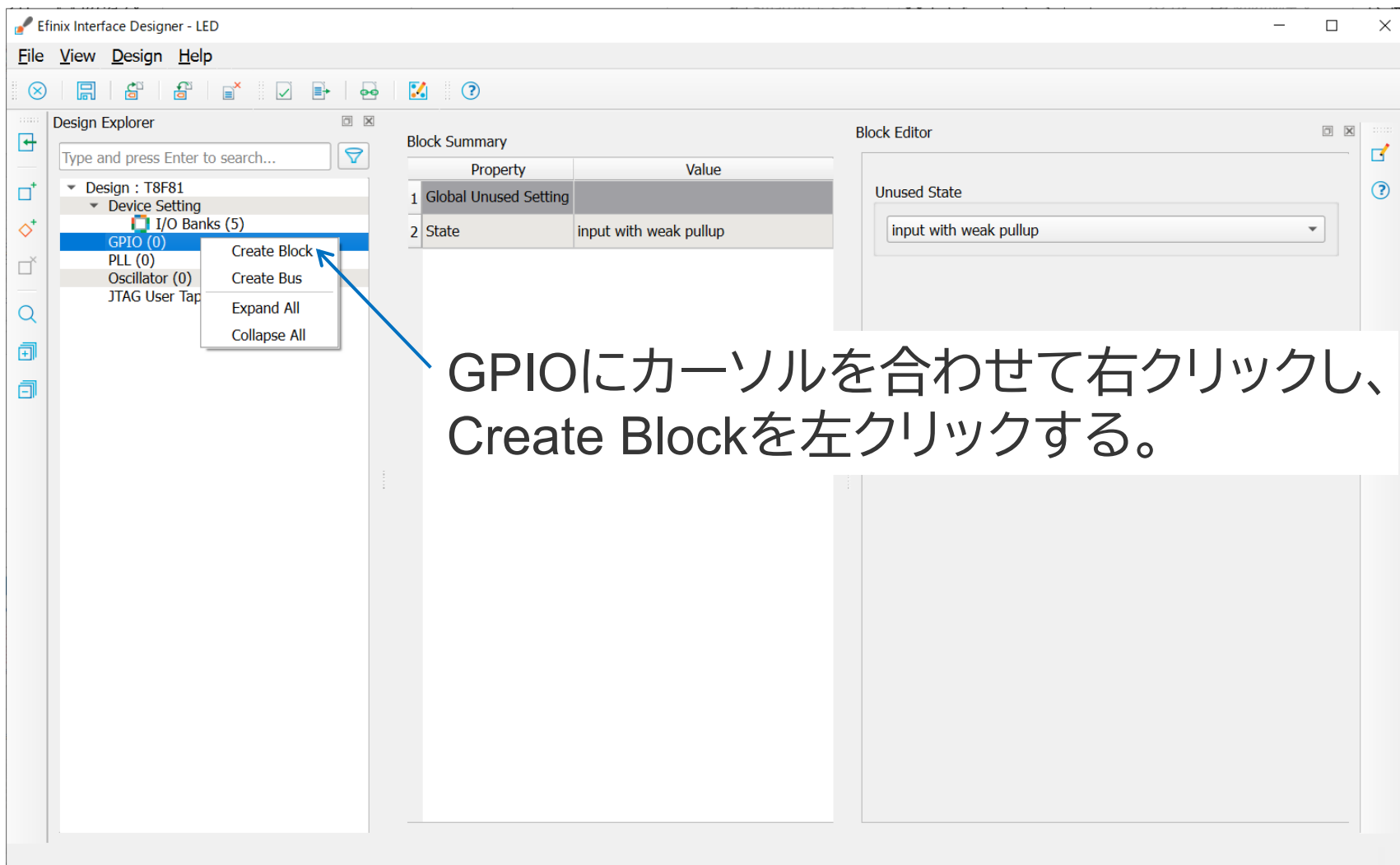


ピンアサインは
Efinity Interface Designer
より行う。



GPIOブロックの作成

GPIO → Create Blockを選択する。



The screenshot displays the Efinix Interface Designer interface. In the Design Explorer on the left, the 'GPIO (0)' component is selected, and a context menu is open with 'Create Block' highlighted. A blue arrow points from the text overlay to this menu item. The Block Editor on the right shows the 'Unused State' dropdown menu set to 'input with weak pullup'. The Block Summary table in the center shows the following properties:

Property	Value
1 Global Unused Setting	
2 State	input with weak pullup

GPIOにカーソルを合わせて右クリックし、
Create Blockを左クリックする。

必要な数だけCreate Blockをする

ポートはbuttonとledがあるので、2回Create Blockをする。

The screenshot shows the Efinix Interface Designer interface. The Design Explorer on the left shows a project named 'T8F81' with two GPIO instances, 'gpio_inst1' and 'gpio_inst2', highlighted. The Block Editor on the right shows the configuration for 'gpio_inst2'. The Block Summary table is as follows:

Property	Value
1 Instance Name	gpio_inst2
2 GPIO Resource	
3 Mode	input
4 I/O Standard	3.3 V LVTTTL / LVCMOS
5 Unused State	NA
6 Input	
7 Pin Name	
8 Connection Type	
9 Register Option	none
10 Pull Option	none
11 Enable Schmitt Trigger	false

A blue arrow points from the 'gpio_inst2' entry in the Design Explorer to the 'gpio_inst2' entry in the Block Summary table. A white text box with the Japanese text 'ポートの数だけ用意する。' (Prepare only the number of ports) is overlaid on the Block Editor area.

他のFPGAと違って、Verilogからの自動生成はありません...

buttonポートの設定

gpio_inst1のInstance Nameをbuttonに変更する。

The screenshot shows the Efinix Interface Designer interface. The Design Explorer on the left shows the project structure for 'Design : T8F81', including 'Device Setting', 'I/O Banks (5)', and 'GPIO (2)'. The 'GPIO (2)' section is expanded, showing 'button :', 'gpio_inst2 :', 'PLL (0)', 'Oscillator (0)', and 'JTAG User Tap (0)'. The Block Summary table in the center lists the properties of the 'button' instance. The Block Editor on the right shows the configuration for the 'button' instance, with the Instance Name field set to 'button'. A blue arrow points to the Instance Name field, and a text box highlights the instruction 'Instance Nameをbuttonに変更'.

Property	Value
1 Instance Name	button
2 GPIO Resource	
3 Mode	input
4 I/O Standard	3.3 V LVTTTL / LVCMOS
5 Unused State	NA
6 Input	
7 Pin Name	button
8 Connection Type	normal
9 Register Option	none
10 Pull Option	none
11 Enable Schmitt Trigger	false

Instance Nameをbuttonに変更

ledポートの設定

gpio_inst2のInstance Nameをledに、Modeをoutputに変更する。

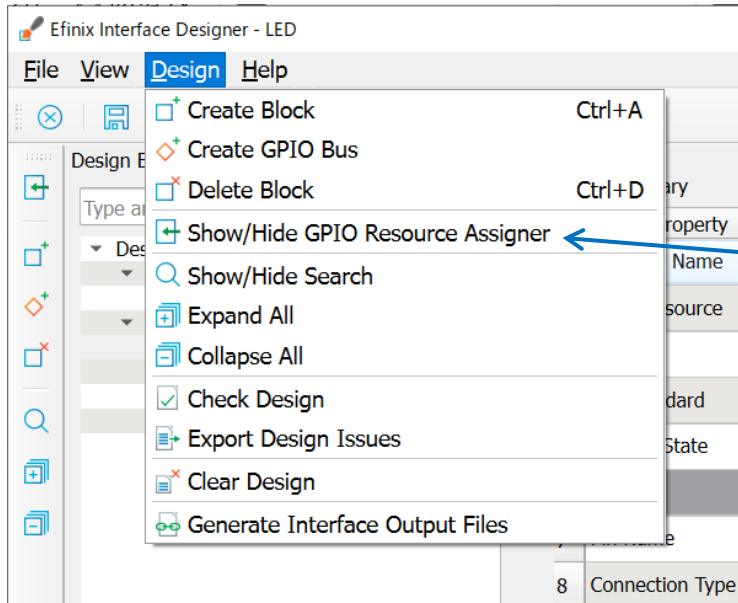
The screenshot shows the Efinix Interface Designer interface for an LED project. The Design Explorer on the left shows the hierarchy: Design: T8F81 > Device Setting > I/O Banks (5) > GPIO (2) > led : GPIOR_06. The Block Summary table in the center lists the properties and values for the selected block. The Block Editor on the right shows the configuration for the 'led' instance, with the Instance Name field set to 'led' and the Mode dropdown set to 'output'. Two blue arrows point from text annotations to these fields.

Property	Value
1 Instance Name	led
2 GPIO Resource	GPIOR_06
3 Mode	output
4 I/O Standard	3.3 V LVTTTL / LVCMOS
5 Unused State	NA
6 Alternate Connection	None
7 Features	None
8 Clock Region	R1
9 I/O Bank	2A
10 Pad	GPIOR_06
11 Package Pin	C6
12 Output	
13 Pin Name	led
14 Constant Output	none
15 Drive Strength	1
16 Enable Slew Rate	false
17 Register Option	none

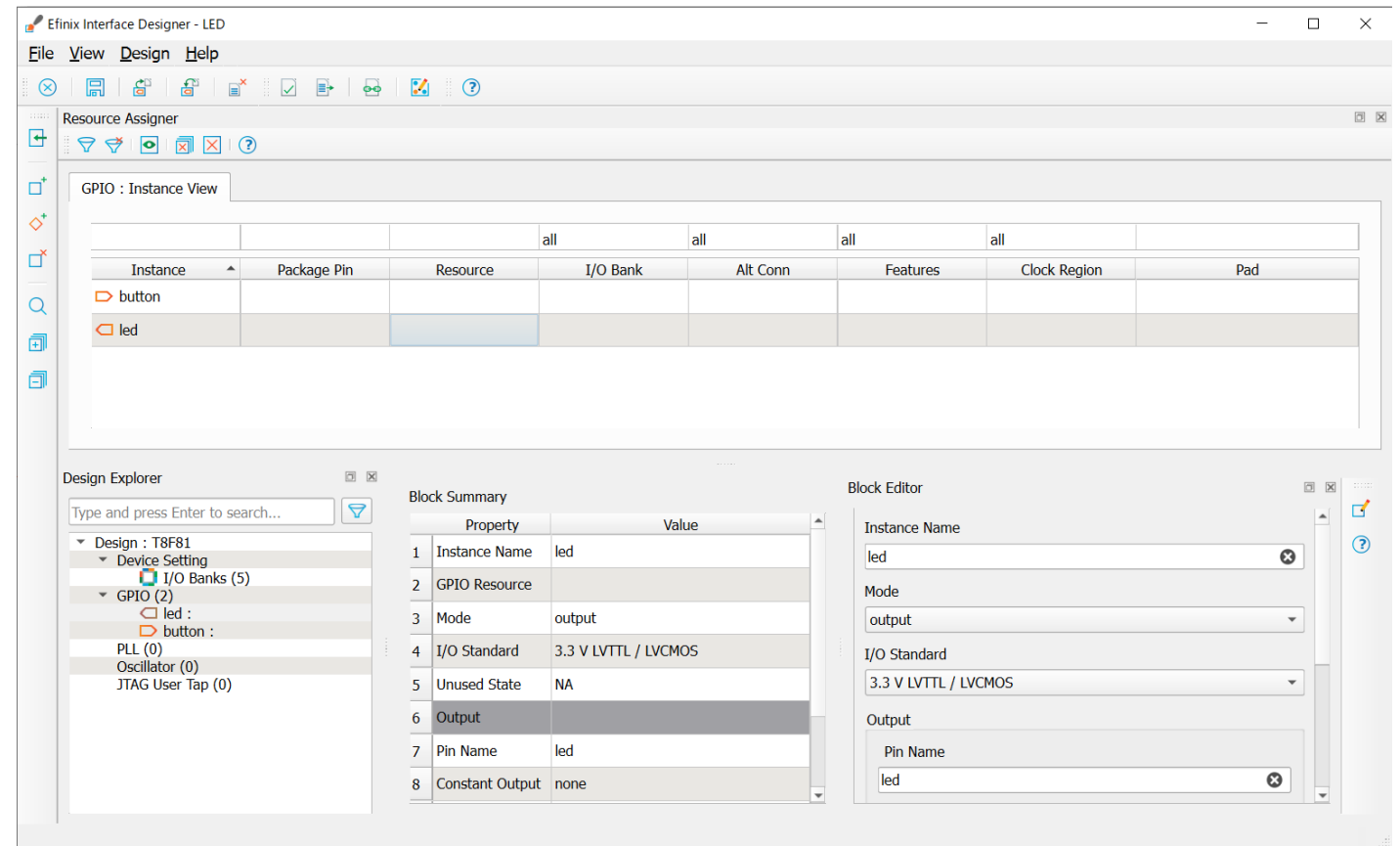
Instance Nameをledに変更

Modeをoutputに変更

ポートのピンアサイン



Show/Hide GPIO Resource Assignerを左クリックする。



Resource Assignerウィンドウが追加され、GPIO: Instance Viewが現れる。

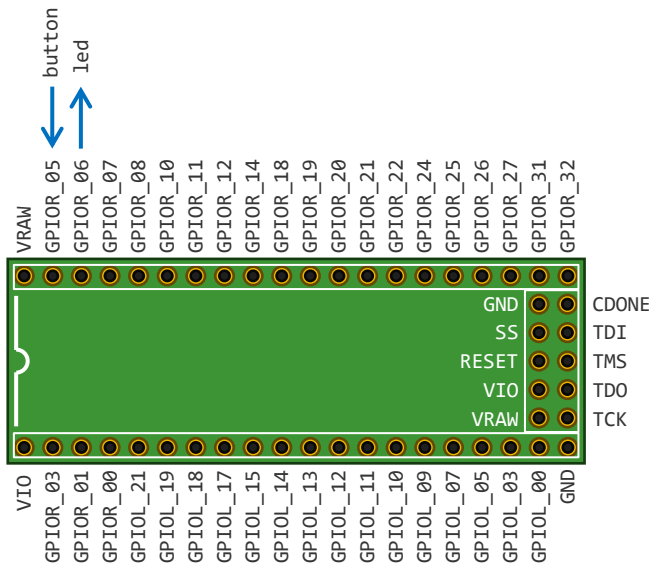
ピンアサイン

VerilogのポートとICパッケージのピンを結びつける。

GPIO : Instance View

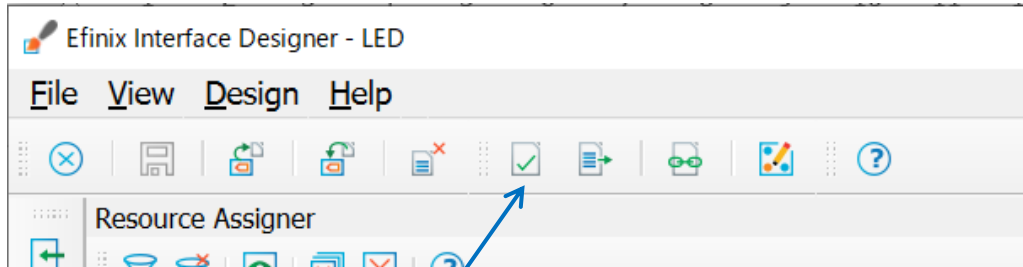
Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
button	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
led	C6	GPIOR_06	2A	None	None	R1	GPIOR_06

buttonに**GPIOR_05**、
ledに**GPIOR_06**を指定する。



設定ファイルの作成

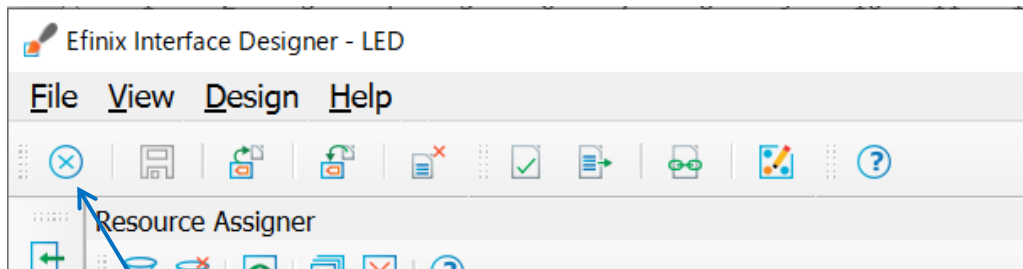
デザインに矛盾がないかチェックする。



チェックアイコンをクリックする。



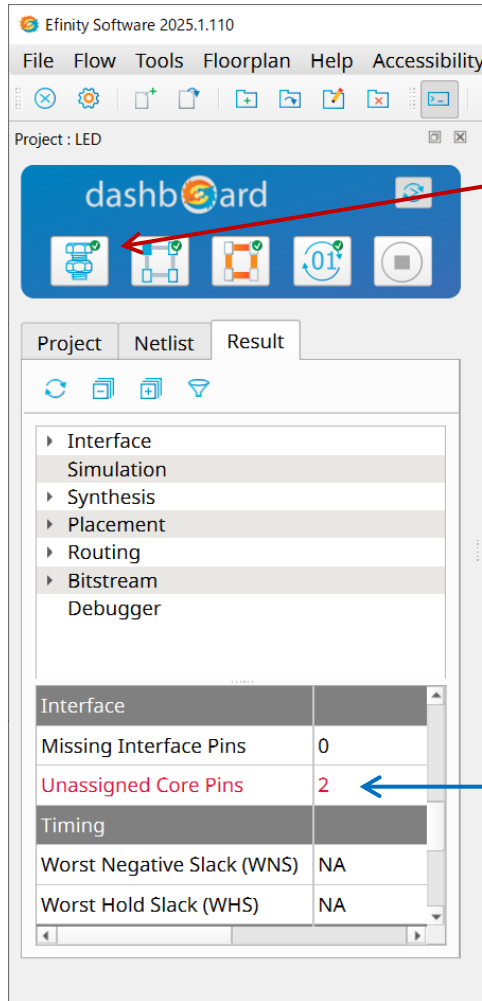
ウィンドウ左下部に0 issueと表示されればOK
これで自動的に設定ファイルが生成される。



バツアイコンをクリックして
Interface Designerを終了する。

合成のやり直し

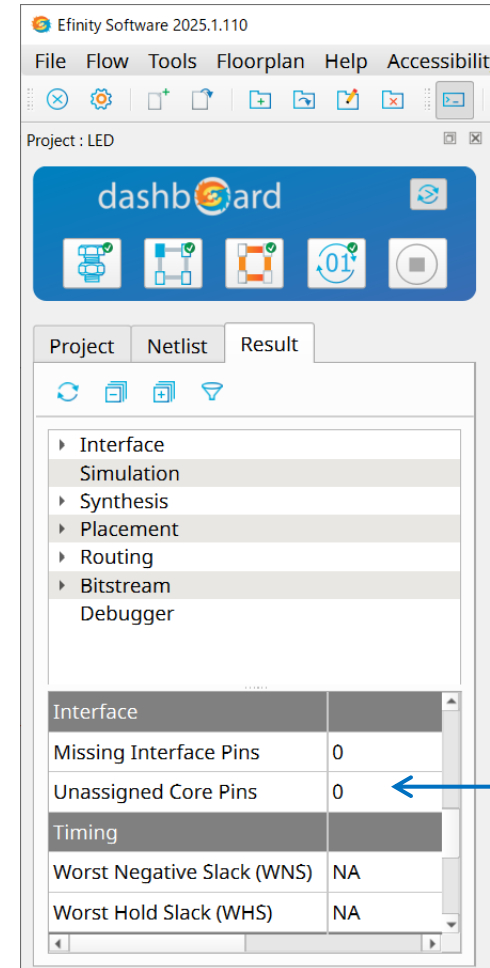
やり直し前



ボルトアイコンをクリックして合成をやり直す

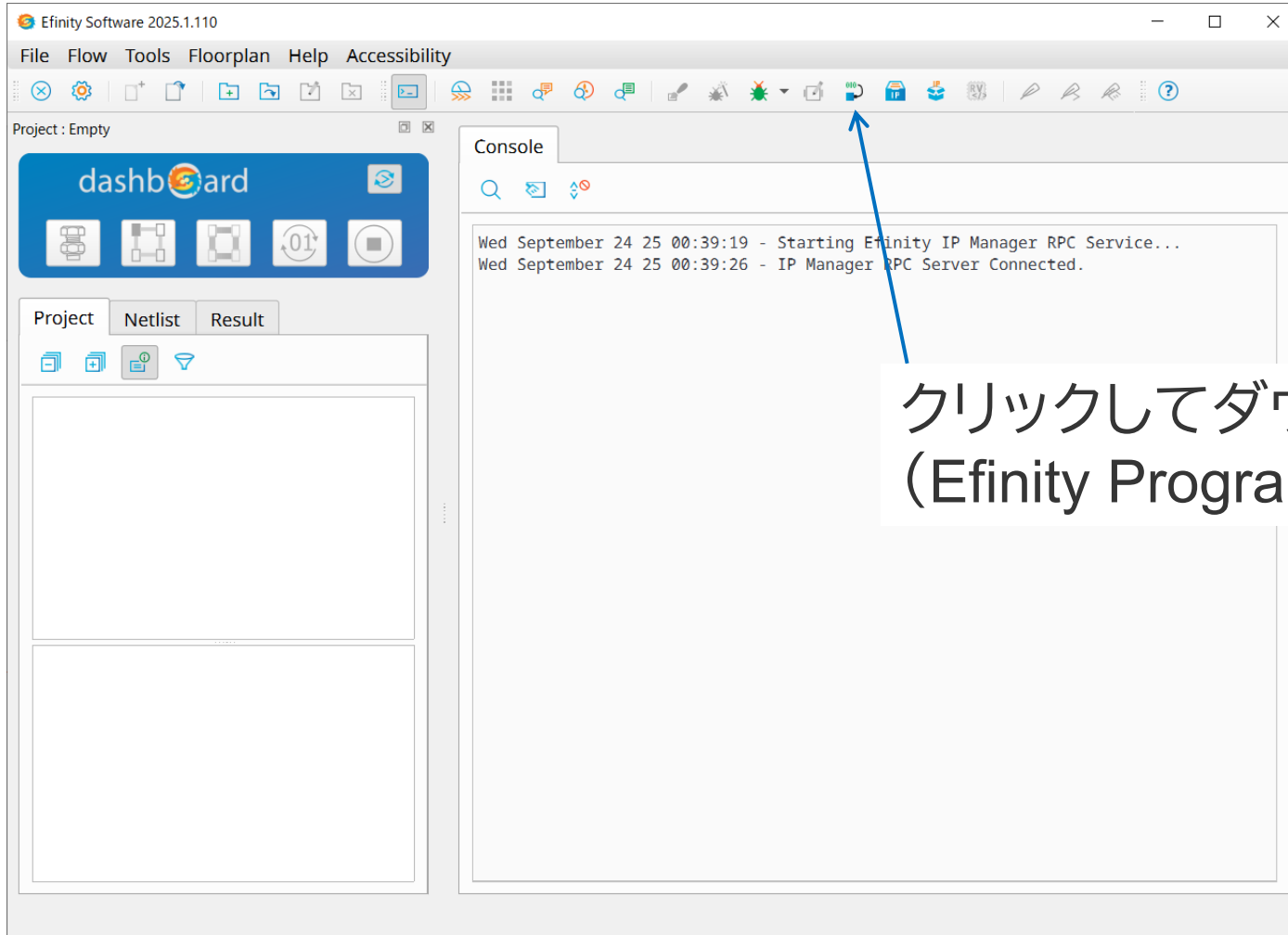
トップレベルからIC外部への接続がないことを意味するエラー

やり直し後



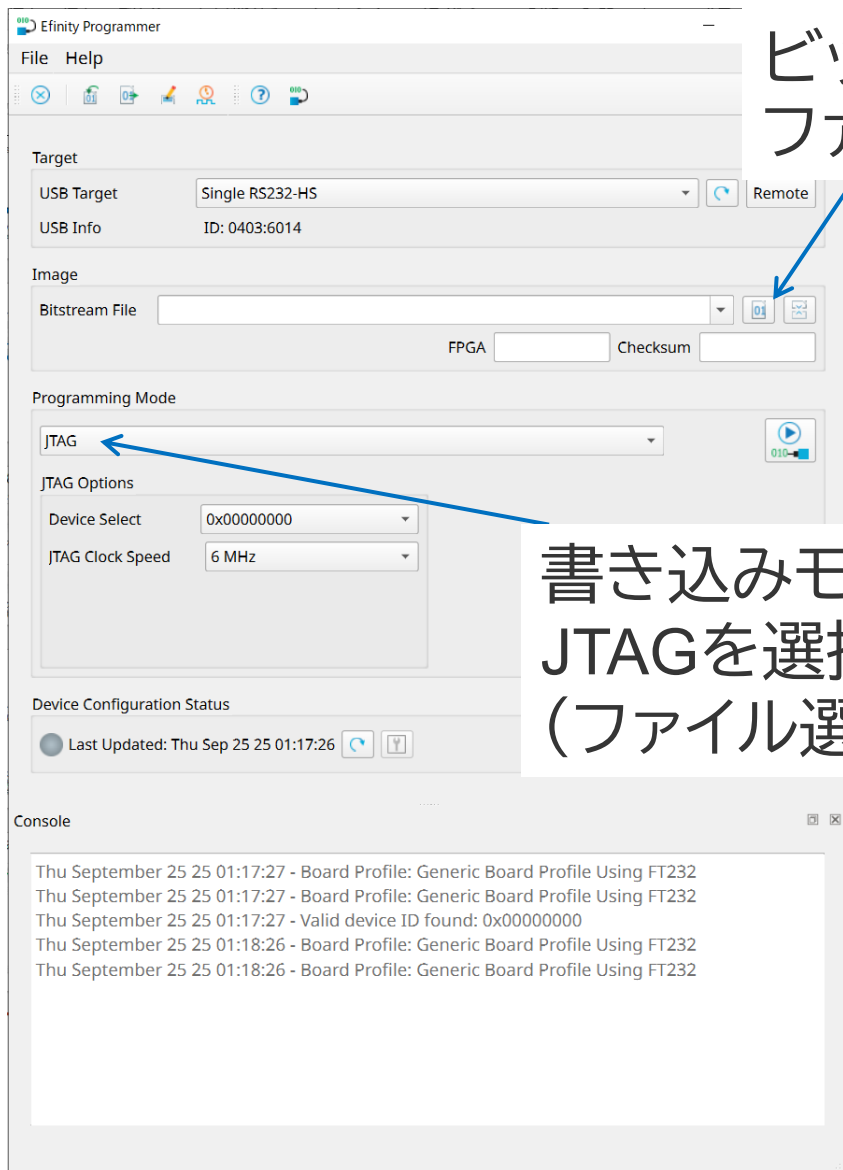
すべてのポートが外部と接続される

FPGAへの書き込みソフトを起動



クリックしてダウンロードソフト
(Efinity Programmer)を起動

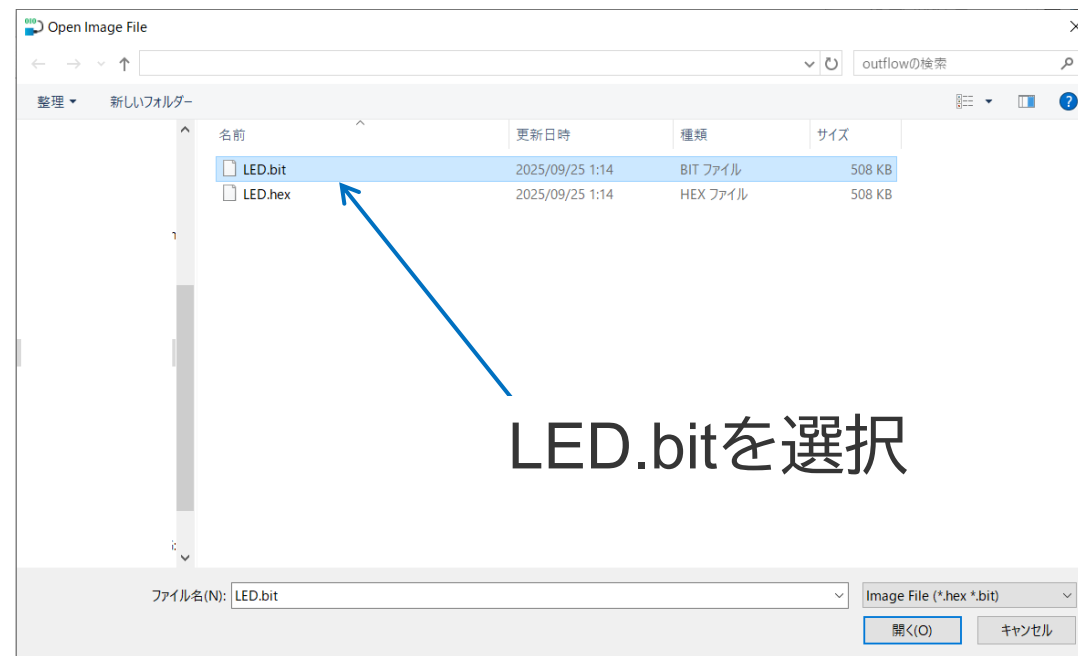
書き込みの設定



ビットストリーム
ファイルの選択

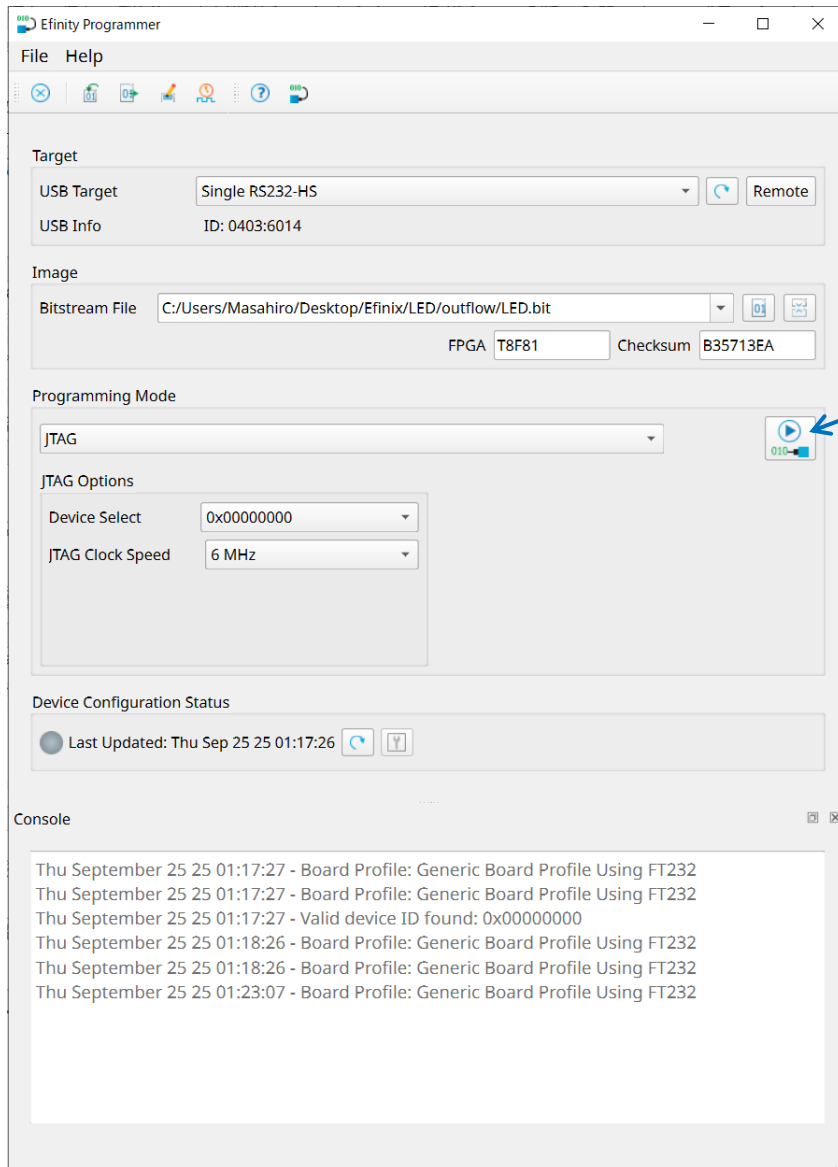


書き込みモードに
JTAGを選択
(ファイル選択前に)

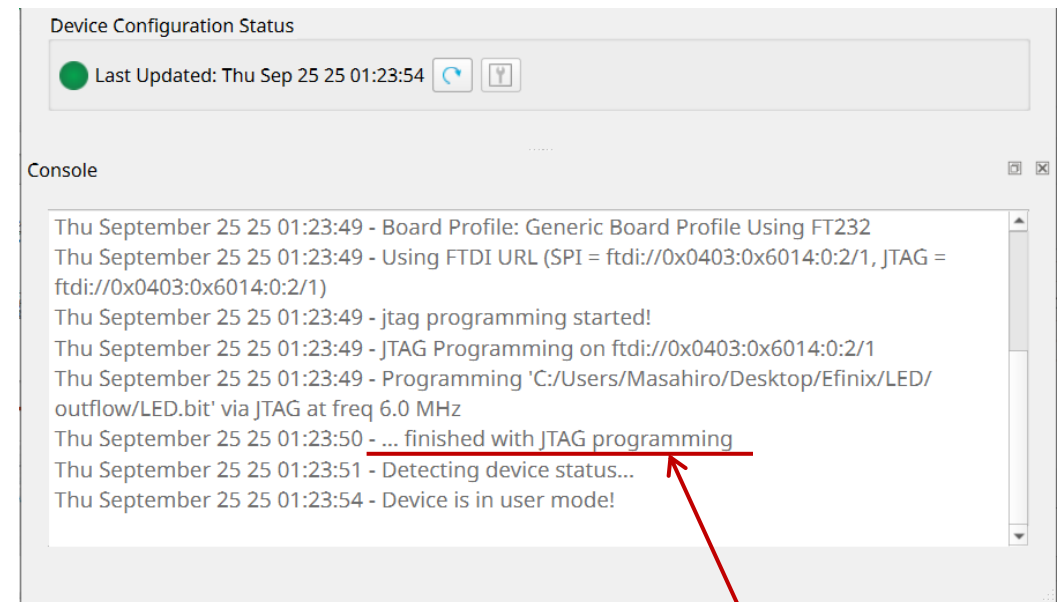


LED.bitを選択

書き込み



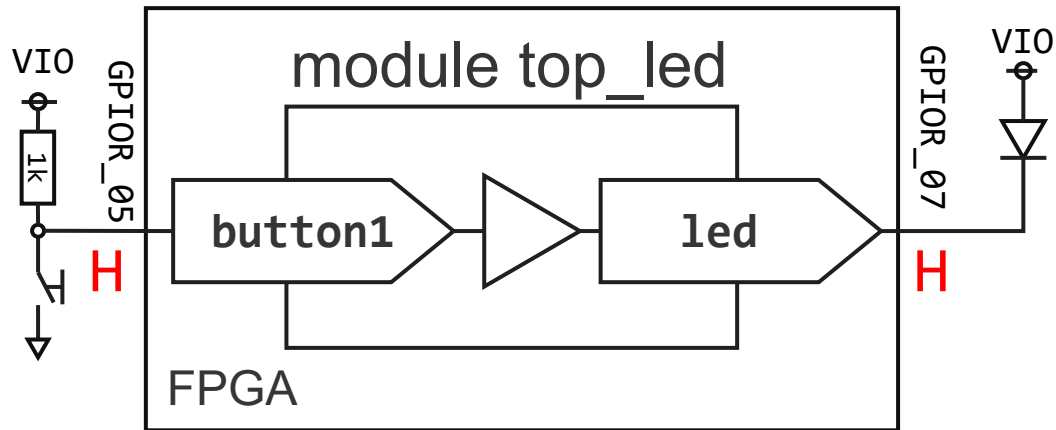
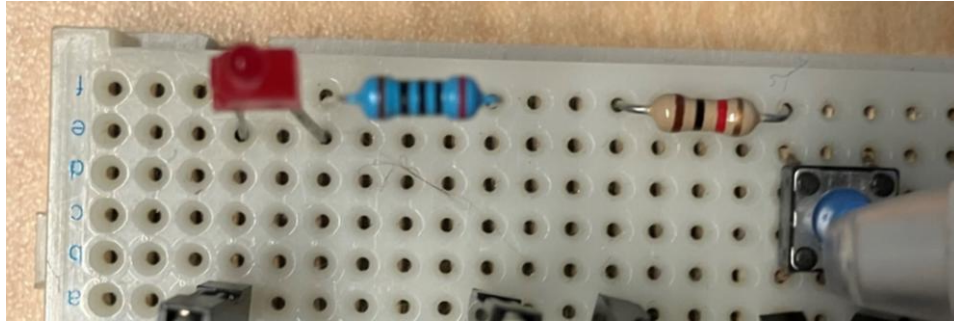
アイコンをクリックして書き込み



これが出ればOK

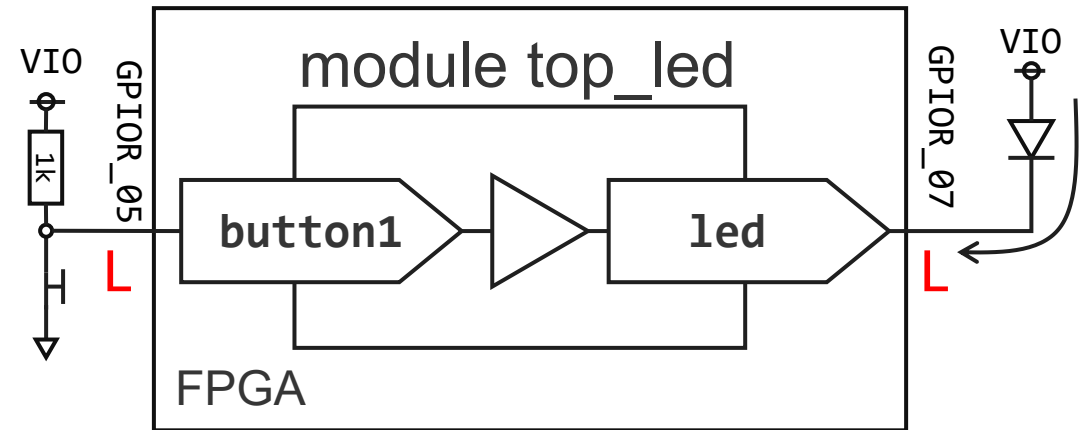
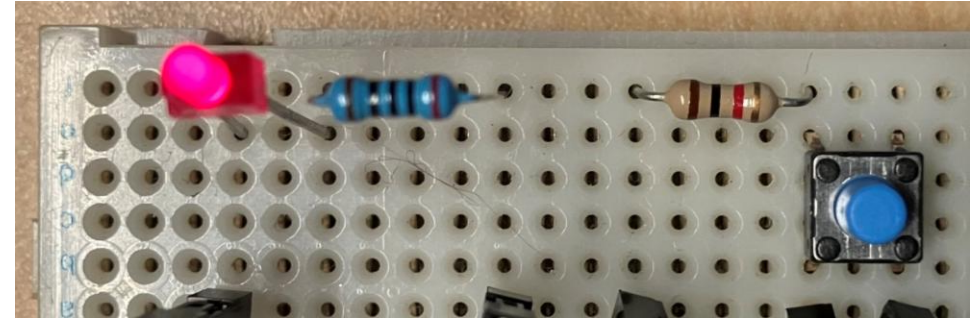
動作

ボタンを押していない状態



- 入力:プルアップ抵抗を介してH論理が入る
- 出力:HなのでLED両端に電位が生じず電流が流れないので消灯

ボタンを押した状態



- 入力:スイッチを介してL論理が入る
- 出力:LなのでLED両端に電位が生じる。電流が流れることで点灯

入力・出力ともに**負論理**なので注意する

Verilog解説その1

Verilogコードの説明

module定義の最後は必ず
セミコロン忘れずに

```
module direct;
```

moduleの中に回路を
記述する

```
endmodule
```

- ファイルの中には必ず1つ以上のモジュールがあります。
- モジュールはmodule ~ endmoduleの範囲です。
- 回路はmodule ~ endmoduleの範囲に書きます。
- moduleのあとにはモジュール名が入ります。モジュール名には算術演算子、スペース、その他予約語は入れられません。
- モジュール定義の終わりはセミコロン(;)が必要です。

Verilogコードの説明

```
module direct(  
    input wire a,  
    output wire b  
);
```

ここにセミコロンを忘れない

```
endmodule
```

- モジュール名のあとには回路の入力と出力を定義します。
- 入力は、出力はoutput指示をつけます。
- **wire**は配線であることを意味します。
- 入出力にはそれぞれを区別するラベルが必要です。
- ラベルを区切るために,(カンマ)が必要です。(改行は必須ではありません)
- モジュール定義のセミコロンはカッコの外に配置します。

Verilogコードの説明

```
module direct(  
    input wire a,  
    output wire b  
);
```

このセミコロンを忘れない

```
assign b = a;
```

```
endmodule
```

- ラベルaとラベルbの関係を結びつける方法の1つがassign文です。
 - 組み合わせ回路の記述方法の1つです
- assignのあとに結びつける対象 = 結びつけるルールの順に記述します。
- 配線aを配線bに接続するようなイメージです。代入ではありません。
- 文の最後には必ず「;」(セミコロン)が必要です。

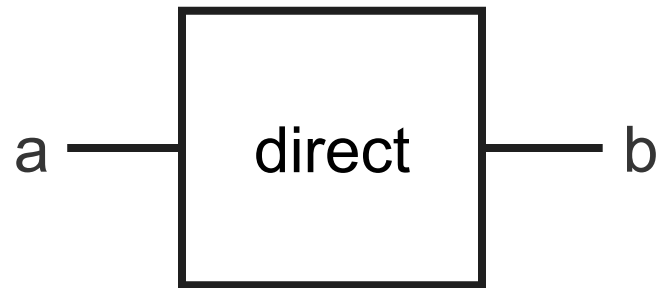


assignは配線をつなぐイメージ

初めてのVerilogの動作

```
module direct(  
    input wire a,  
    output wire b  
);  
  
    assign b = a;  
  
endmodule
```

これまでの説明から左の回路Verilogは次の動作になる。



真理値表

入力a	出力b
0	0
1	1

bはaをそのまま出力するので、

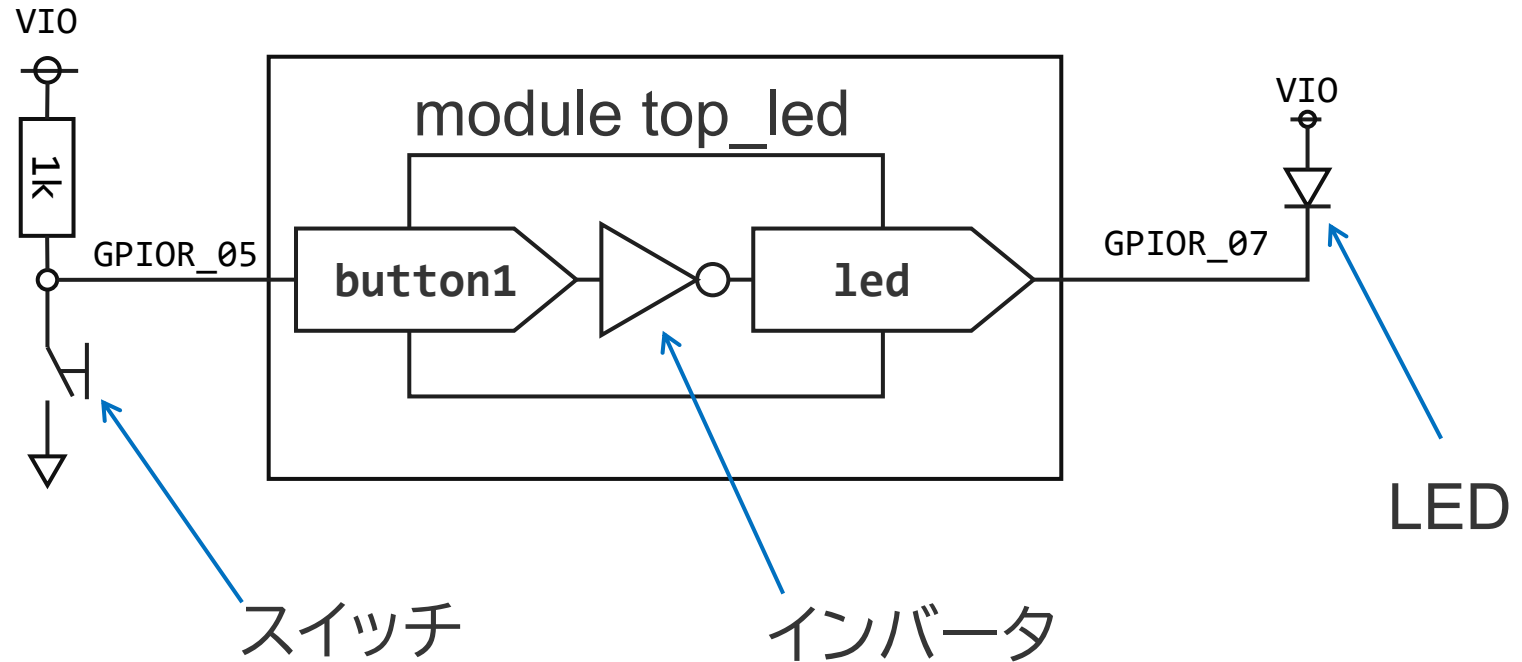
- 入力0のときは出力0
- 入力1のときは出力1

となる。

いろいろな演算子の回路

NOTゲートの動作確認

プルアップ抵抗



インバータのコード

top_ledのときと同様にFile -> New Fileからファイルを作成。ファイル名はmy_inv。

```

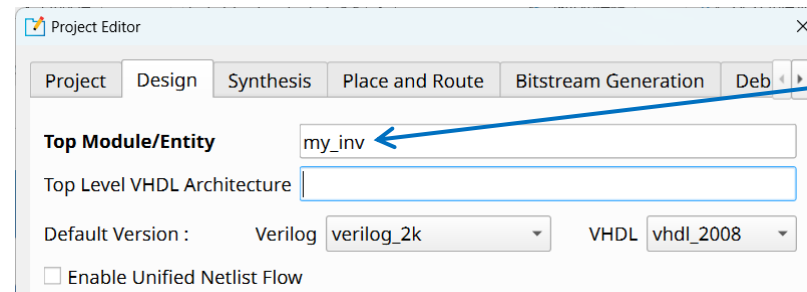
1 module my_inv(
2   input wire button,
3   output wire led);
4
5   assign led = ~button;
6
7   endmodule

```

モジュール名はmy_invに

NOTゲートは~(チルダ)を用いて表す。

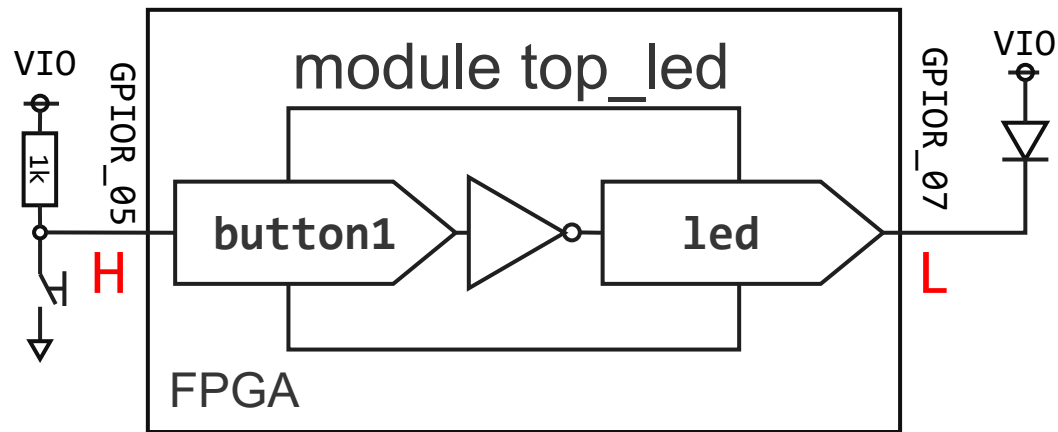
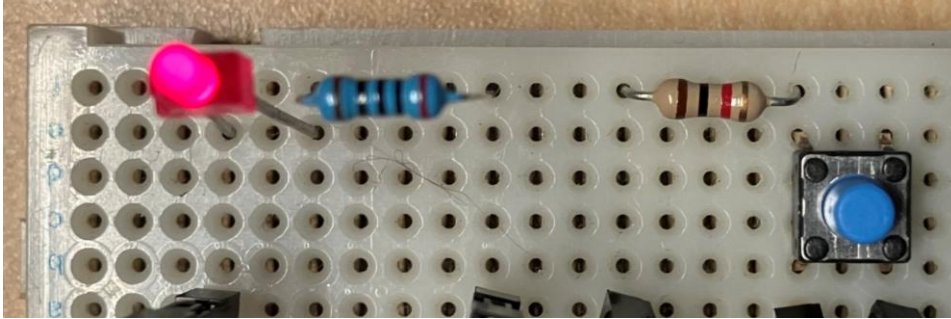
File -> Edit projectからProject Editorを開き、Designタブ内の
Top Module/Entityをmy_invに変更



my_invに変更

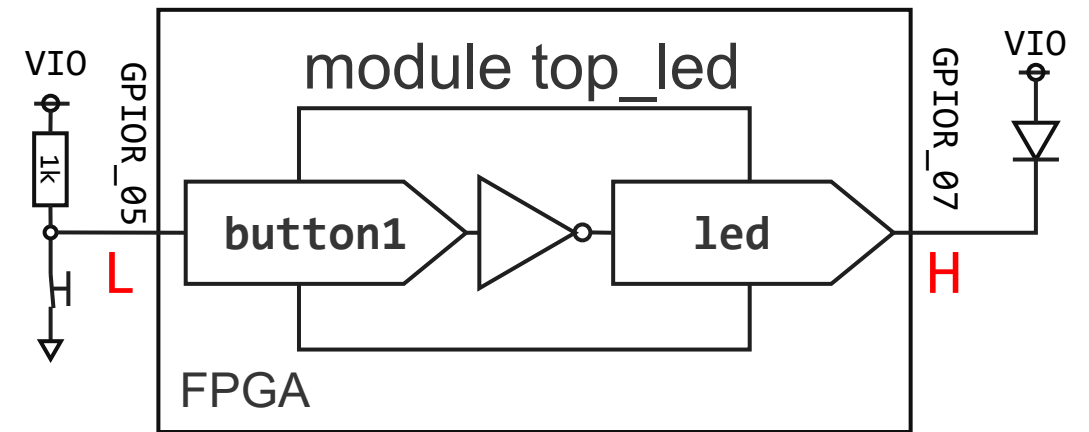
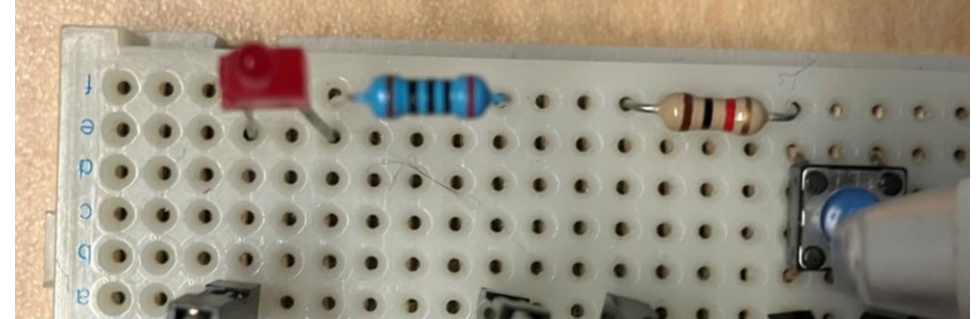
インバータの動作

ボタンを押していない状態



- 入力:プルアップ抵抗を介してH論理が入る
- 出力: LなのでLED両端に電位が生じる。電流が流れることで点灯

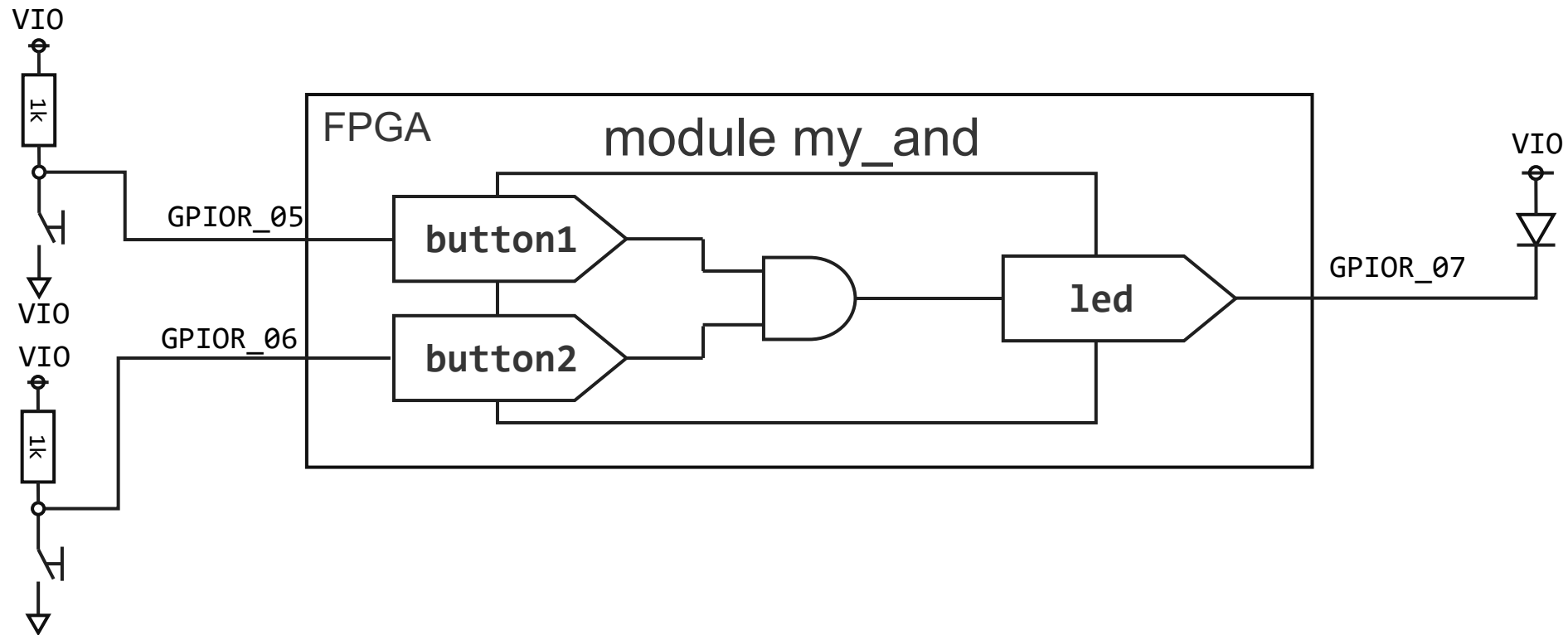
ボタンを押した状態



- 入力:スイッチを介してL論理が入る
- 出力:HなのでLED両端に電位が生じず電流が流れないので消灯

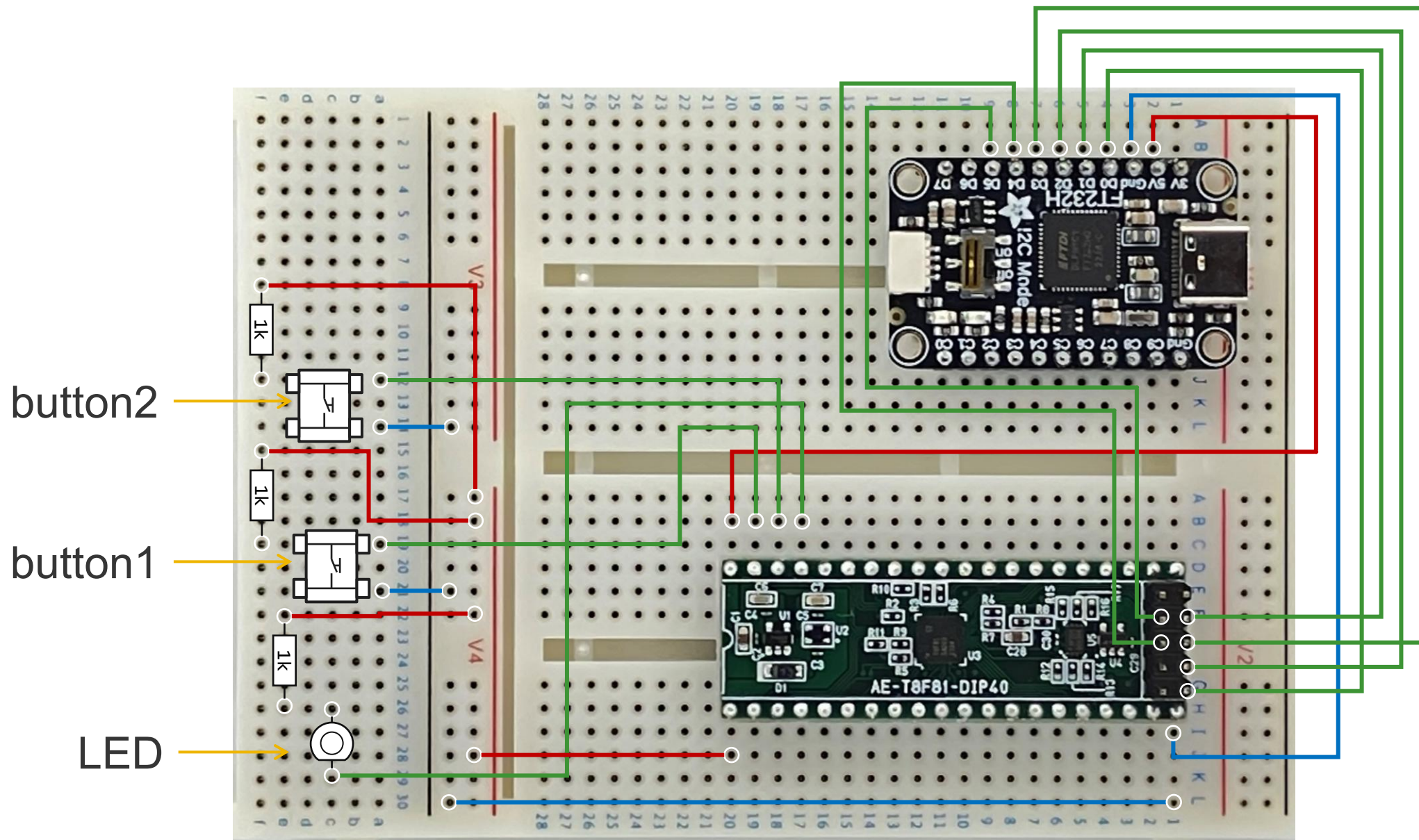
ANDゲートの動作確認

2入力ANDゲートとして動作する回路をFPGAに実装する。
入力として2つのボタンと1つのLEDを有する。



2個のボタンはGPIOR_05とGPIOR_06、LEDはGPIOR_07に接続する。

配線図



Verilogコードとピンアサイン

※Top Module/Entityをmy_andに変更するのを忘れずに

Code Editor

my_and.v

```

1 module my_and(
2   input wire button1,
3   input wire button2,
4   output wire led);
5
6   assign led = button1 & button2;
7
8   endmodule
9

```

モジュール名はmy_and

ANDゲートは&を用いて表す。

GPIO : Instance View

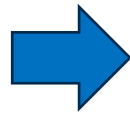
Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
button1	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
button2	C6	GPIOR_06	2A	None	None	R1	GPIOR_06
led	C7	GPIOR_07	2A	None	None	R1	GPIOR_07

ANDゲートの動作

どちらのスイッチを押してもLEDが点灯するORのような動作をしていると思います。

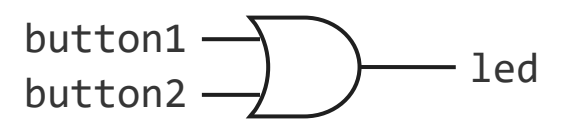
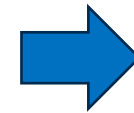
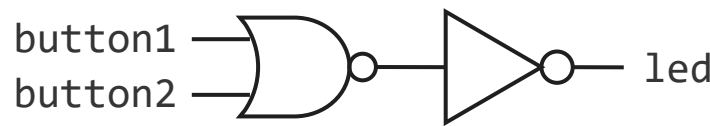
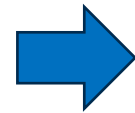
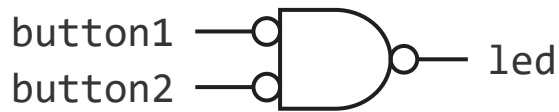
button1の状態	button2の状態	button1ピンの入力論理	button2ピンの入力論理	ledピンの出力論理	ledの状態
離す	離す	H	H	H	消灯
押す	離す	L	H	L	点灯
離す	押す	H	L	L	点灯
押す	押す	L	L	L	点灯

- ボタンを押したときにピンがL
- 出力がLのときにLEDが点灯



入出力がLのときに動作
→ 負論理という

なぜ入出力両方が負論理のときにANDはORになるのでしょうか？



$$led = \overline{\overline{button1} \& \overline{button2}}$$

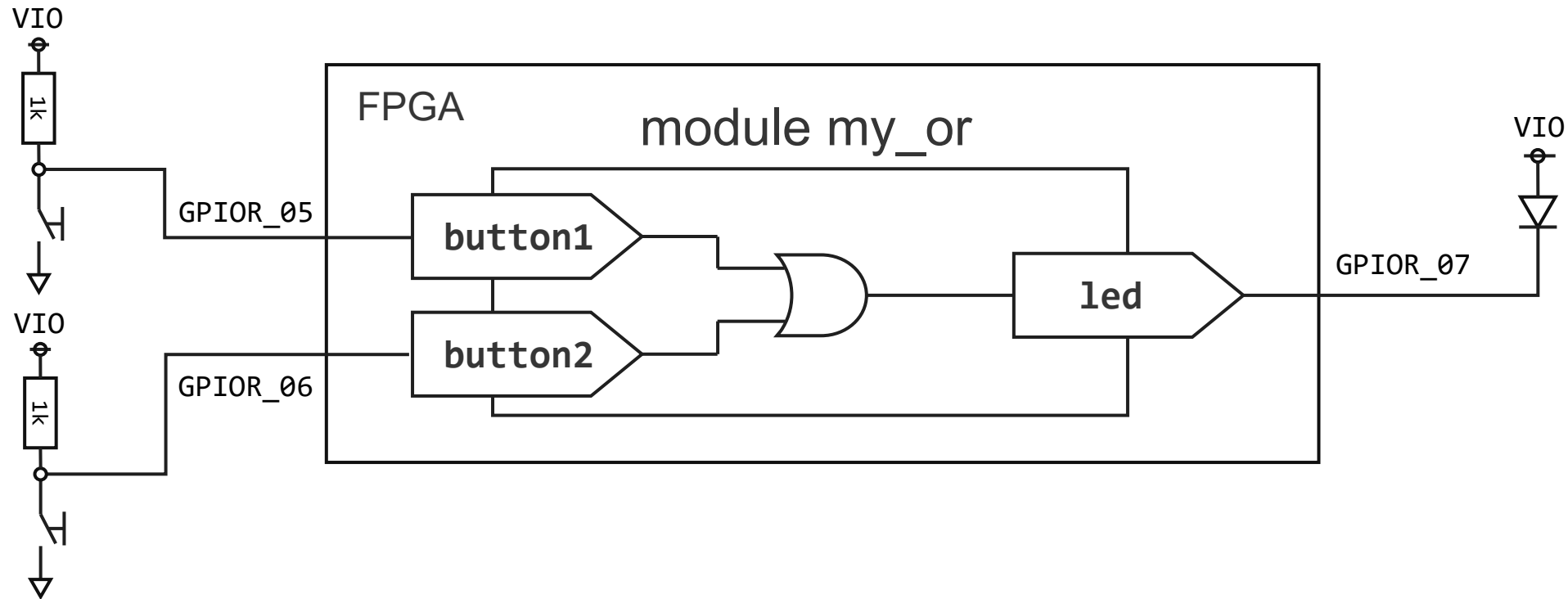
$$led = \overline{\overline{button1} | \overline{button2}}$$

$$led = button1 | button2$$

負論理のANDはこのように変形されるので、結果としてORのように振る舞います。

ORゲートの確認回路

2入力ORゲートとして動作する回路をFPGAに実装する。
入力として2つのボタンと1つのLEDを有する。



2個のボタンはGPIOR_05とGPIOR_06、LEDはGPIOR_07に接続する。
配線はANDゲートの実験と同じものを使用する。

Verilogコードとピンアサイン

※Top Module/Entitiyをmy_orに変更するのを忘れずに

Code Editor

my_or.v

```

1 module my_or(
2   input wire button1,
3   input wire button2,
4   output wire led);
5
6   assign led = button1 | button2;
7
8   endmodule
9

```

モジュール名はmy_orに

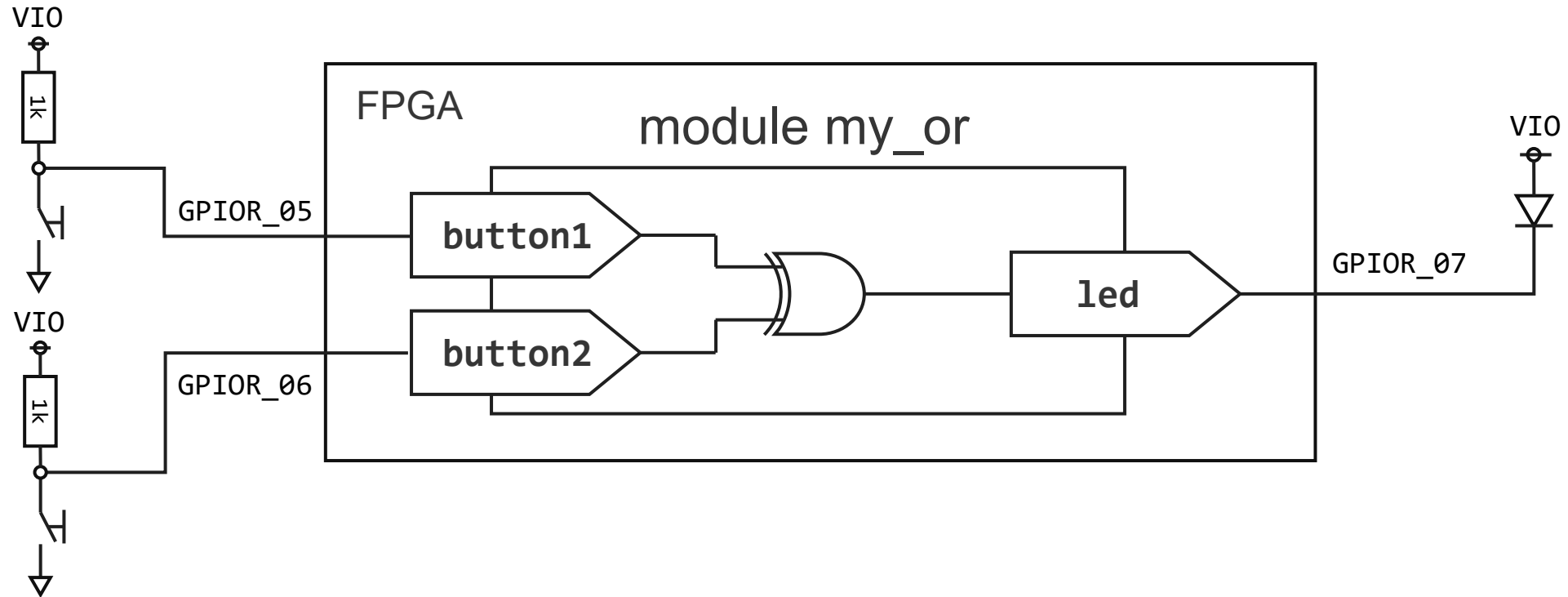
ORゲートは|を用いて表す。

GPIO : Instance View

Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
button1	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
button2	C6	GPIOR_06	2A	None	None	R1	GPIOR_06
led	C7	GPIOR_07	2A	None	None	R1	GPIOR_07

ExORゲートの確認回路

2入力ExORゲートとして動作する回路をFPGAに実装する。
入力として2つのボタンと1つのLEDを有する。



2個のボタンはGPIOR_05とGPIOR_06、LEDはGPIOR_07に接続する。
配線はANDゲートの実験と同じものを使用する。

Verilogコードとピンアサイン

※Top Module/Entityをmy_xorに変更するのを忘れずに

Code Editor

```

1 module my_xor(
2   input wire button1,
3   input wire button2,
4   output wire led);
5
6   assign led = button1 ^ button2;
7
8   endmodule
9

```

モジュール名はmy_xorに

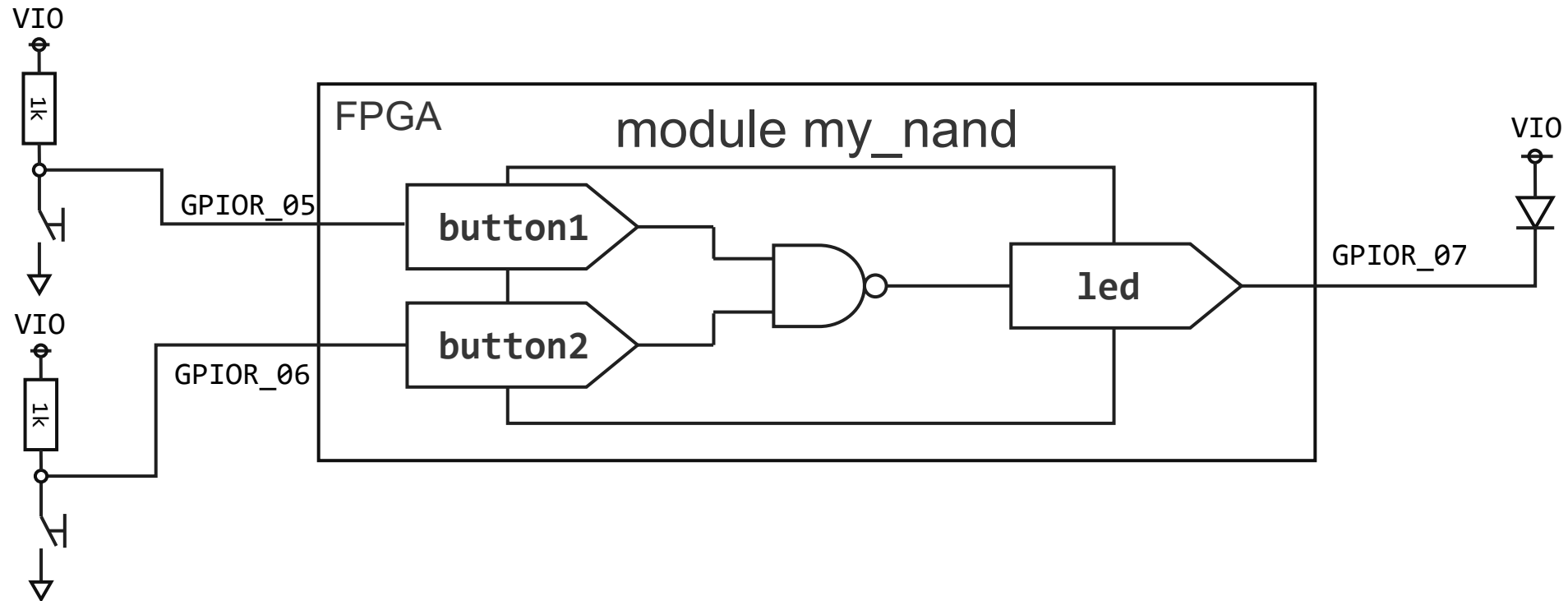
ORゲートは^(ハット)を用いて表す。

GPIO : Instance View

Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
button1	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
button2	C6	GPIOR_06	2A	None	None	R1	GPIOR_06
led	C7	GPIOR_07	2A	None	None	R1	GPIOR_07

NANDゲートの確認回路

2入力NANDゲートとして動作する回路をFPGAに実装する。
入力として2つのボタンと1つのLEDを有する。



2個のボタンはGPIOR_05とGPIOR_06、LEDはGPIOR_07に接続する。
配線はANDゲートの実験と同じものを使用する。

Verilogコードとピンアサイン

※Top Module/Entityをmy_nandに変更するのを忘れずに

Code Editor

my_or.v x my_nand.v x

```

1 module my_nand(
2   input wire button1,
3   input wire button2,
4   output wire led);
5
6   assign led = ~(button1 & button2);
7
8   endmodule
9

```

モジュール名はmy_nandに

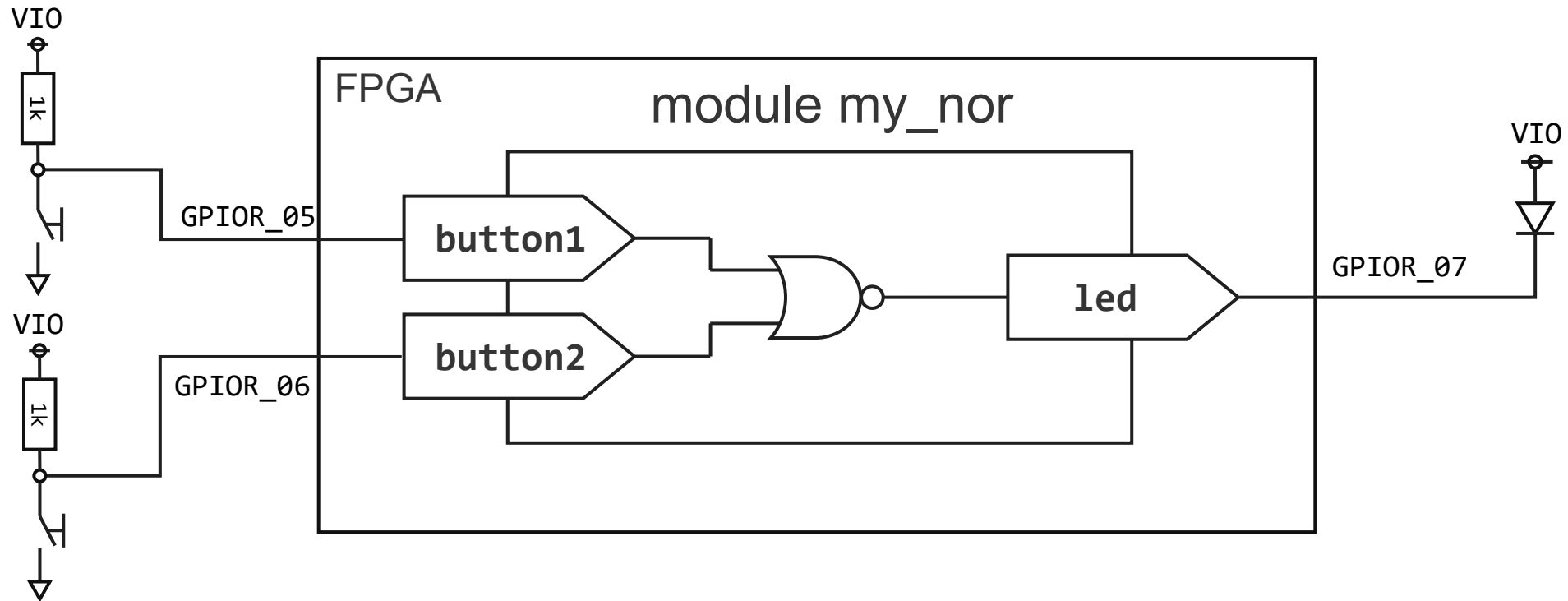
NANDゲートは()で囲った&を論理反転することで表す。

GPIO : Instance View

Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
button1	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
button2	C6	GPIOR_06	2A	None	None	R1	GPIOR_06
led	C7	GPIOR_07	2A	None	None	R1	GPIOR_07

NORゲートの確認回路

2入力ORゲートとして動作する回路をFPGAに実装する。
入力として2つのボタンと1つのLEDを有する。



2個のボタンはGPIOR_05とGPIOR_06、LEDはGPIOR_07に接続する。
配線はANDゲートの実験と同じものを使用する。

Verilogコードとピンアサイン

※Top Module/Entityをmy_norに変更するのを忘れずに

Code Editor

my_nor.v

```

1 module my_nor(
2   input wire button1,
3   input wire button2,
4   output wire led);
5
6   assign led = ~(button1 | button2);
7
8 endmodule

```

モジュール名はmy_norに

NORゲートは()で囲った|を論理反転することで表す。

GPIO : Instance View

Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
button1	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
button2	C6	GPIOR_06	2A	None	None	R1	GPIOR_06
led	C7	GPIOR_07	2A	None	None	R1	GPIOR_07

Verilog解説その2

NOT

Verilogでは論理反転(NOT)は「~」(チルダ)を使用する。
論理反転したい信号ラベルの前に~を配置する。

NOT回路のVerilogコード

```
module my_not(  
    input wire a,  
    output wire b  
);  
    assign b = ~a;  
endmodule
```

aを論理反転する



AND

Verilogでは論理積(AND)は「&」を使用する。
論理積をとる2つの信号ラベルの間に&を配置する。

AND回路のVerilogコード

```
module my_and(  
  input wire a,  
  input wire b,  
  output wire c);  
  
  assign c = a & b;  
endmodule
```

入力を2つに増やすには、
inputを追加する。
ラベルはユニーク(一意)になるようにする

aとbを論理積を求める

OR

Verilogでは論理和(OR)は「|」を使用する。
論理和をとる2つの信号ラベルの間に|を配置する。

OR回路のVerilogコード

```
module my_or(  
    input wire a,  
    input wire b,  
    output wire c);  
  
    assign c = a | b;  
endmodule
```

aとbを論理和を求める



ExOR

Verilogでは排他的論理和は「^」を使用する。
排他的論理和をとる2つの信号ラベルの間に^を配置する。

ExOR回路のVerilogコード

```
module my_exor(  
  input wire a,  
  input wire b,  
  output wire c);  
  
  assign c = a^b;  
endmodule
```

aとbを排他的論理和を求める



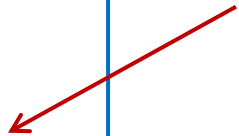
NAND

Verilogでは否定論理積(NAND)を直接計算する演算子はない。
そこで論理反転と論理積を組み合わせて実現する。

NAND回路のVerilogコード

```
module my_nand(  
    input wire a,  
    input wire b,  
    output wire c);  
  
    assign c = ~(a & b);  
endmodule
```

aとbの論理積を求めてから、
その結果を反転する。
カッコ内は優先して演算される。



ここまでの演算子まとめ

- 単項演算子: オペランド(対象)が1つのみの演算子
 - \sim : NOT
- 二項演算子: オペランド(対象)が2つの演算子
 - $\&$: AND
 - $|$: OR
 - \wedge : Exclusive OR

便利なVerilog文法

wire

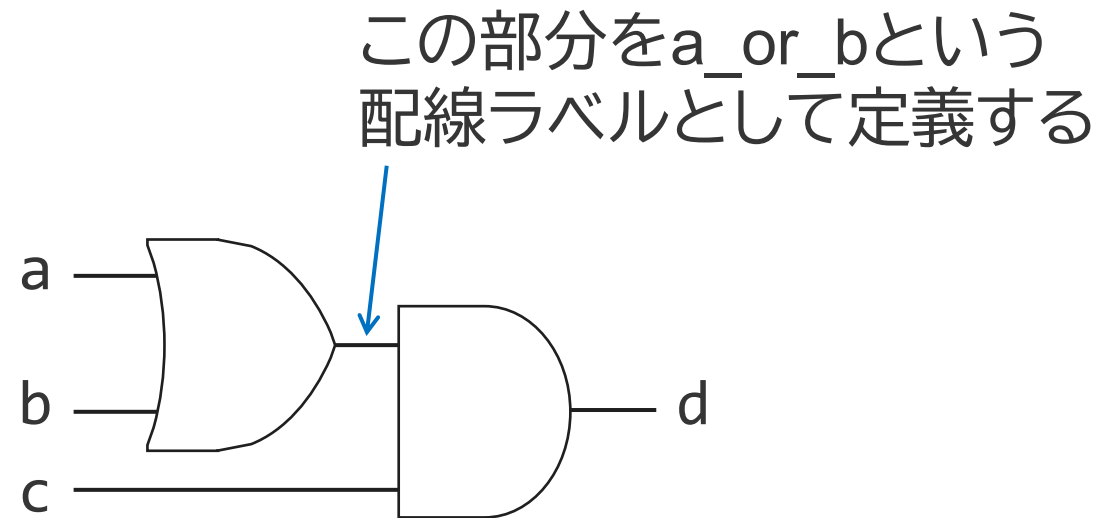
wire構文を用いることで新たに配線ラベルを定義できる。wire構文は、

```
wire wire_name;
```

のように記述する。wireの後ろには、配線名をつける。これで、wire_nameという配線が新たに定義された。

wireは複雑な組み合わせ回路で使用できる。

```
wire a_or_b;  
assign a_or_b = a | b;  
assign d = a_or_b & c;
```



組み合わせ回路

```
Code Editor
complex.v
1 module complex(
2   input wire a,
3   input wire b,
4   input wire c,
5   output wire d);
6
7   wire a_or_b;
8   assign a_or_b = a | b;
9   assign d = a_or_b & c;
10
11 endmodule
12
```

色がうまくつかないですが、
多分文法チェッカーがバグっています。

入力3個、出力1個となるように定義してください。
接続箇所はp.38を参照してください。

GPIO : Instance View

Instance	Package Pin	Resource	I/O Bank	Alt Conn	Features	Clock Region	Pad
a	B6	GPIOR_05	2A	None	None	R1	GPIOR_05
b	C6	GPIOR_06	2A	None	None	R1	GPIOR_06
c	C7	GPIOR_07	2A	None	None	R1	GPIOR_07
d	A8	GPIOR_08	2A	None	None	R1	GPIOR_08

今日の練習課題

3入力の論理積回路

3入力AND回路のVerilogコード

```
module three_and(  
    
```

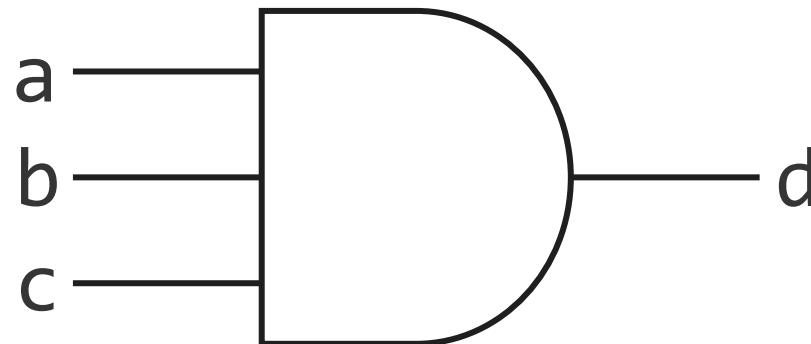
ここに自力でコードを書く

```
);
```

```
    assign d = 
```

ここに自力でコードを書く

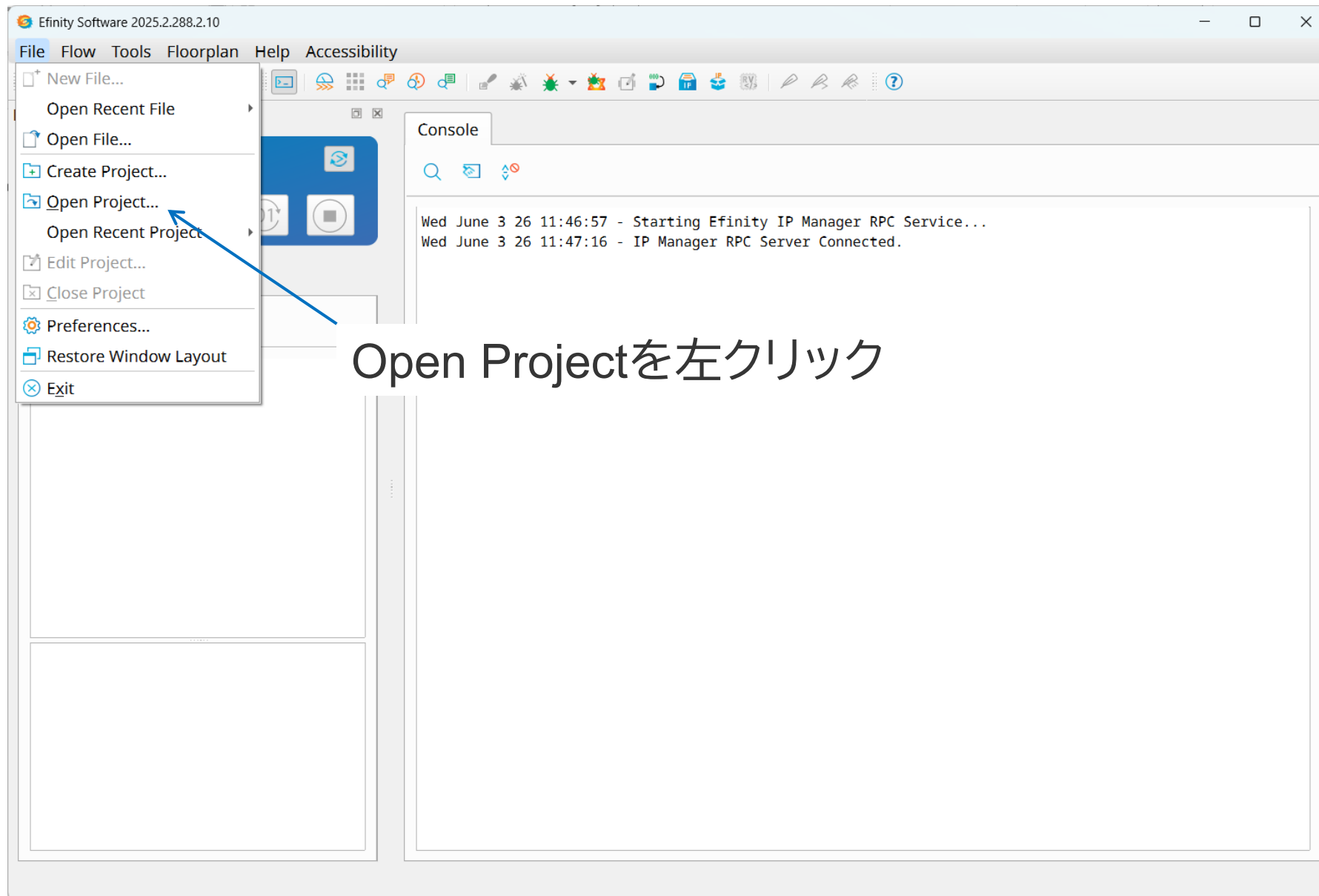
```
endmodule
```



まとめ

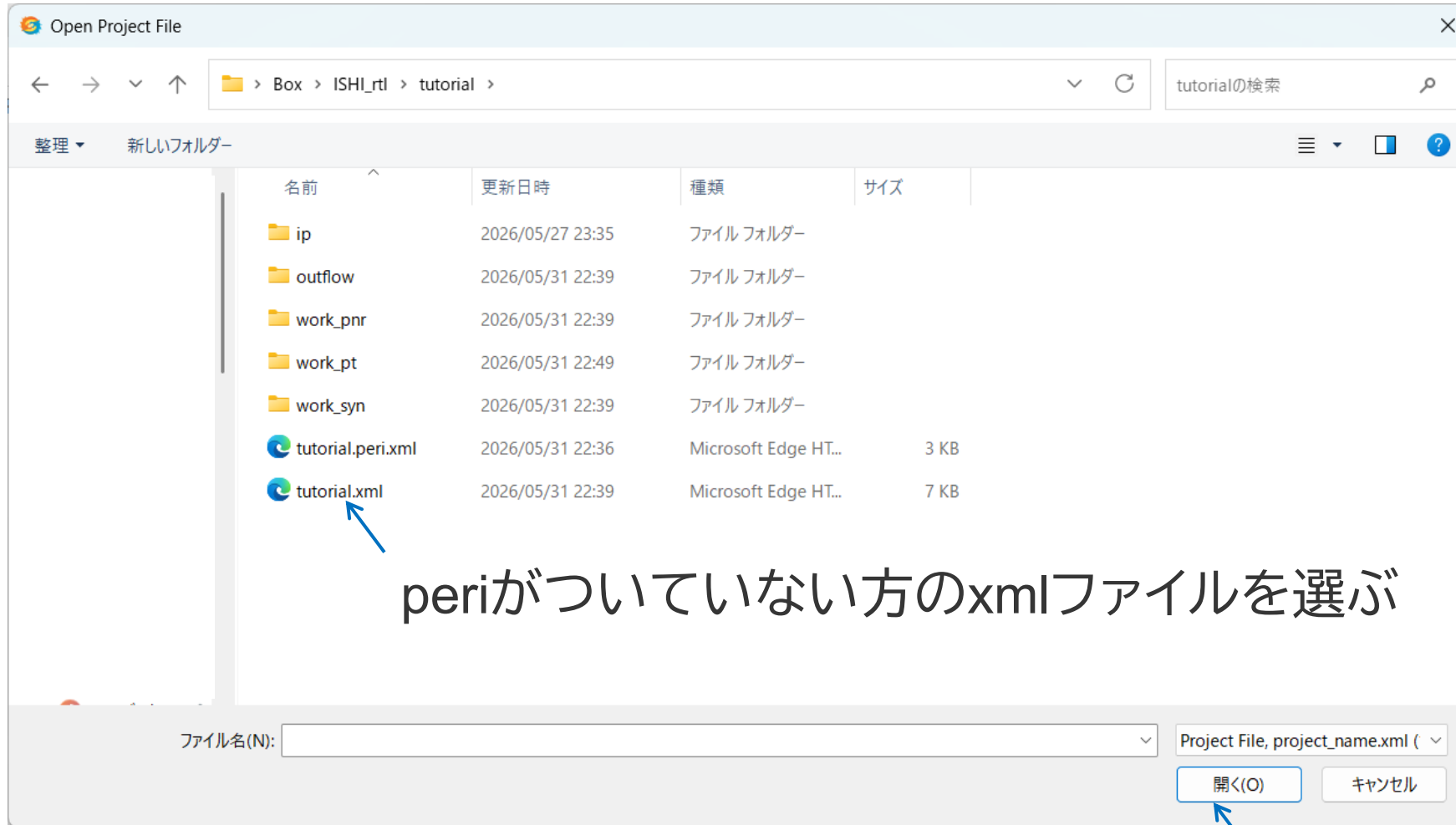
- EfinixのFPGA開発方法を説明しました。
- Verilogの基本的な文法を紹介しました。
 - 今日の文法
 - モジュール定義
 - 基本的な論理演算子
 - wire指定子の使い方

補足 | プロジェクトの開き方



補足 | プロジェクトの開き方

プロジェクトのフォルダに移動する。



選択したら開くをクリック

補足 | プロジェクトの開き方

The screenshot shows the Efinity Software 2025.2.288.2.10 interface. The main window displays a project named "tutorial". The console window shows the following logs:

```
Wed June 3 26 11:46:57 - Starting Efinity IP Manager RPC Service...
Wed June 3 26 11:47:16 - IP Manager RPC Server Connected.
INFO : Reading project database "C:/Users/pochi/Box/ISHI_rtl/tutorial/tutorial.xml"
[FlowMsg::ProjectDatabaseReading]
INFO : Maximum concurrency has been set to 8 [FlowMsg::MaxThreads]
Wed June 3 26 11:53:18 - Project loaded VM : 231.224 MB RSS : 342.688 MB
```

The Project panel shows the following structure:

- tutorial
 - Design
 - File : top_led.v (default)
 - File : my_inv.v (default)
 - File : my_and.v (default)
 - File : my_or.v (default)
 - File : my_nand.v (default)
 - File : my_nor.v (default)
 - File : my_xor.v (default)
 - File : complex.v (default)
 - Constraint
 - Simulation

The Property table shows the following information:

Property	Value
Top Module	complex
Top VHDL Arch	
Device	T8F81
Timing Model	C2
Family	Trion
Software Version	2025.2.288.2.10
Location	C:/Users/pochi/Box/IS

A blue arrow points from the text "プロジェクトが開くとこの辺の情報が更新される" to the Property table.