セキュリティキャンプ2022 Y4
RISC-V CPU自作ゼミ
事前学習

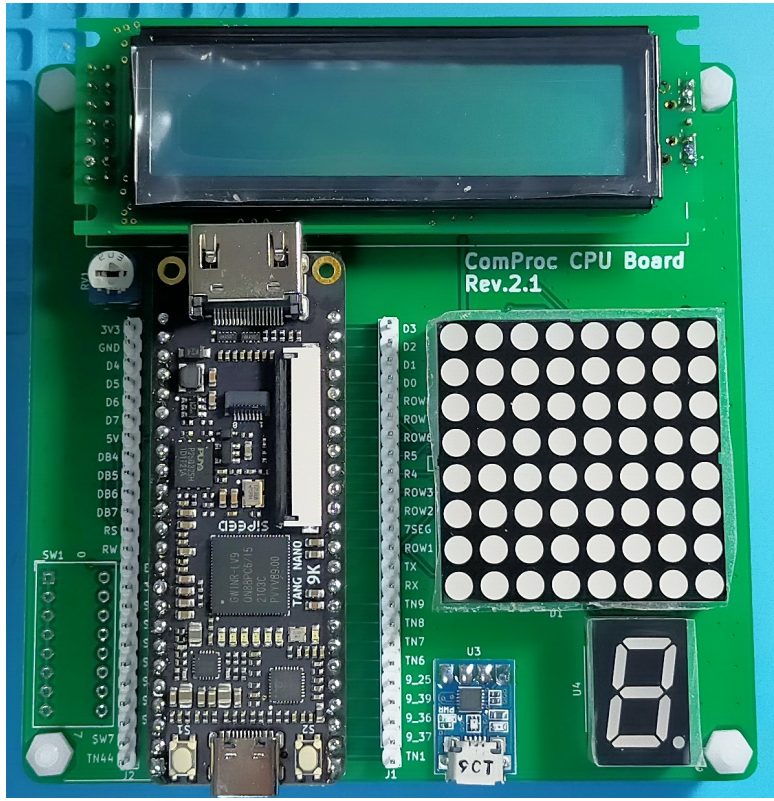
目次

事前学習内容について

- 書籍 「RISC-VとChiselで学ぶ初めてのCPU自作」 第3部まで
  - パイプラインの実装の章
  - 講義中にFPGA上に実装するCPUは第3部までで作成したデザインを使用します
- 機材「ComProc CPU Board Rev.2.1」の動作確認
  - 送付時に書き込み済みのファームウェアの動作確認
  - 簡単なFPGAの論理回路の合成と書き込み

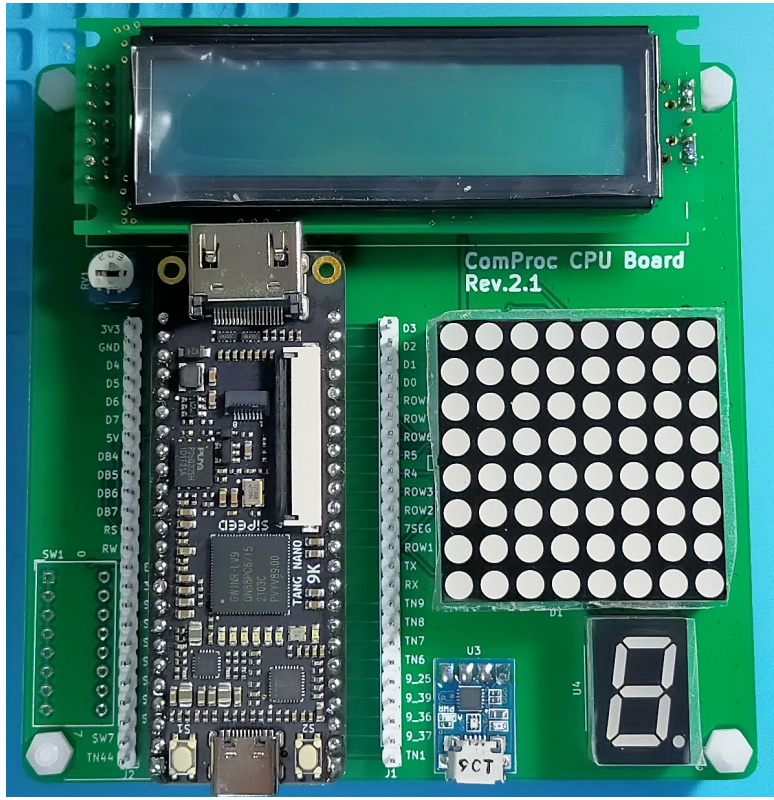
ComProc CPU Board

- @uchan_nos さんが自作CPU実装用に開発した基板
- Sipeed社のTang Nano 4K/9K 向けの拡張基板



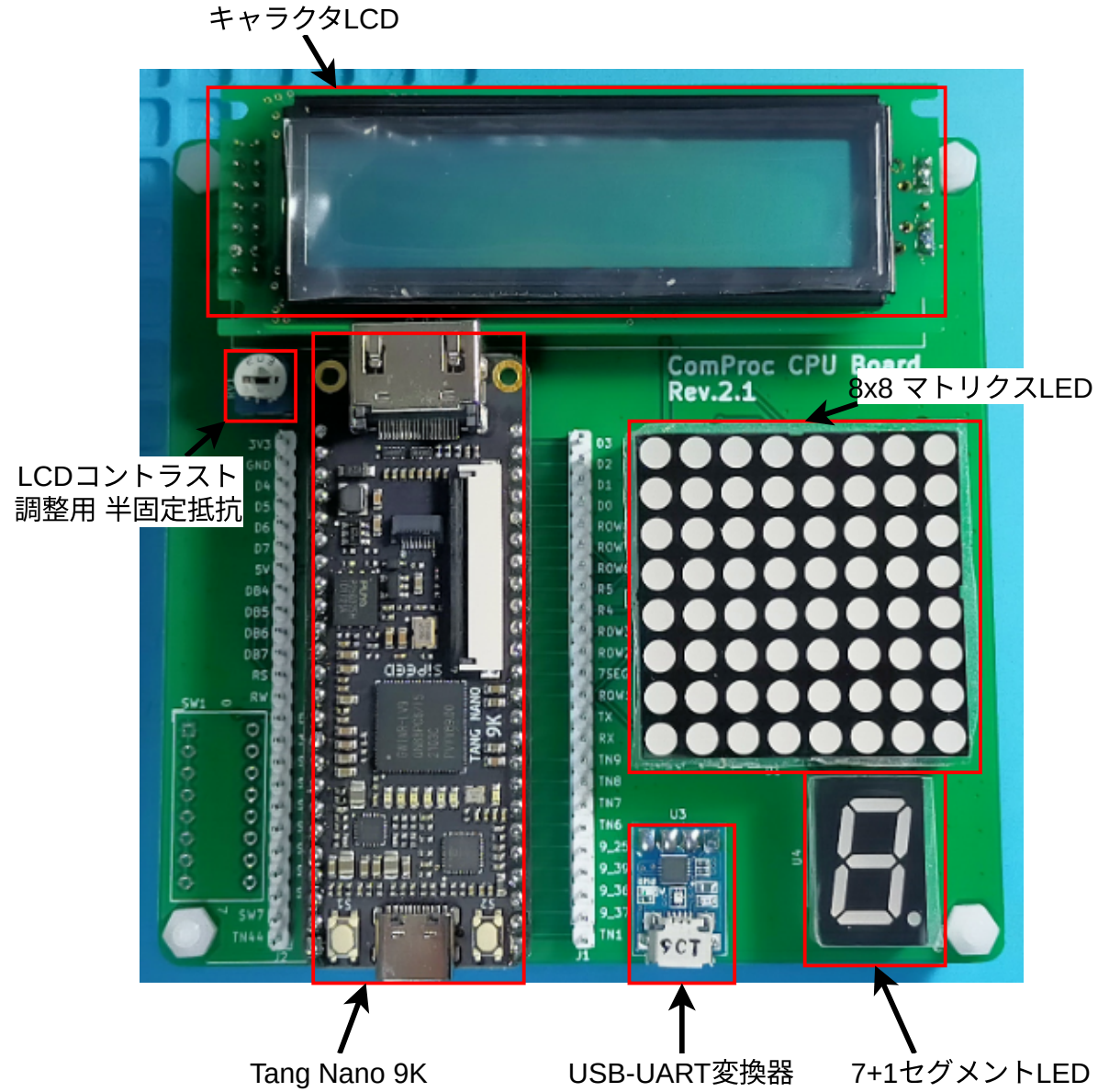
ComProc CPU Board

- 本講義ではオリジナルの Com Proc CPU Board Rev.2 をTang Nano 9K専用にして簡略化したものを使用
  - 便宜上 Rev 2.1と呼びます
- 以降、講義中に Com Proc CPU Board と言った場合はRev 2.1のことを指します



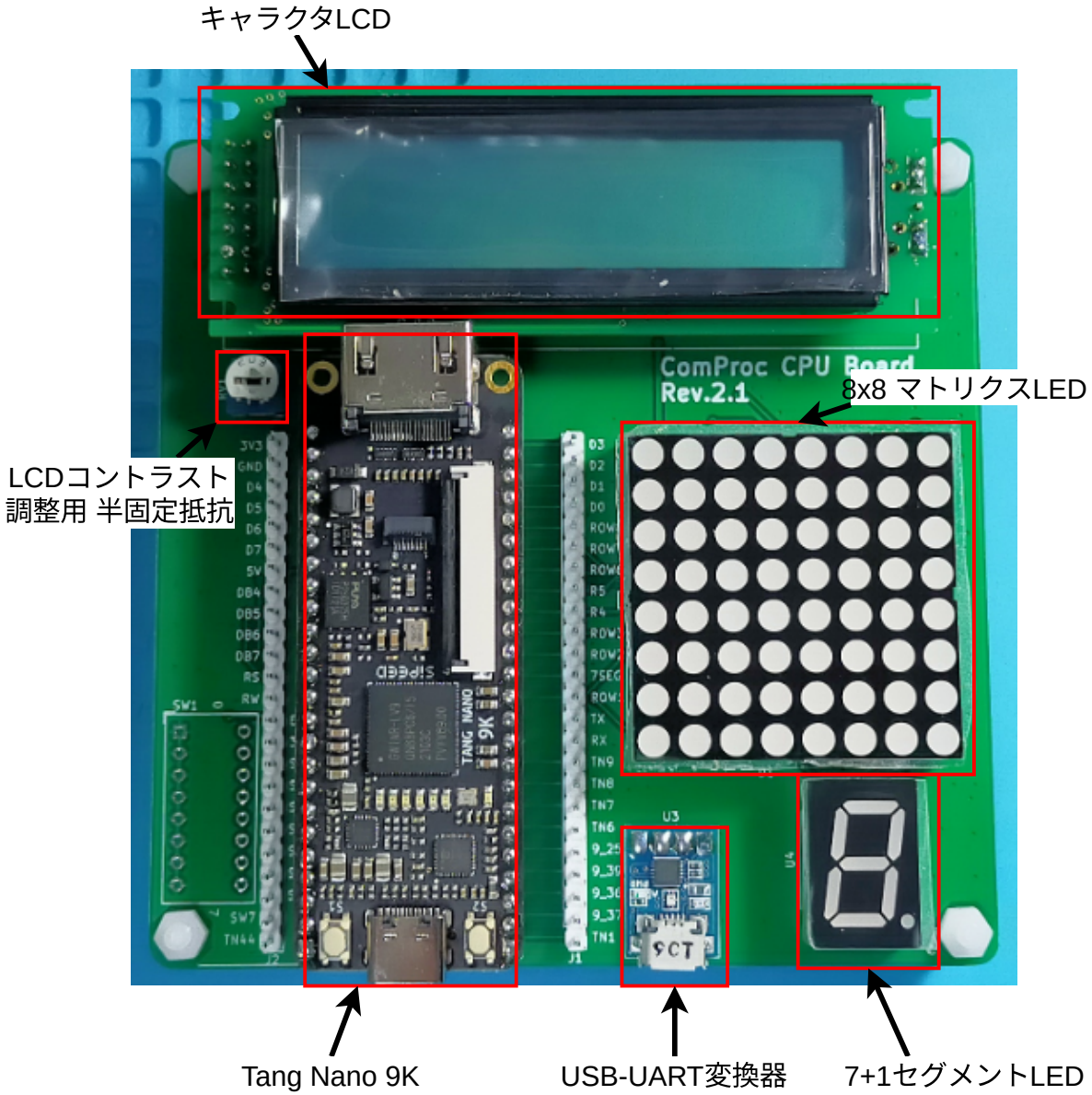
ComProc CPU Boardの機能

- FPGAボードとしてTang Nano 9K (Sipeed) を使用
- 搭載FPGA: GOWIN GW1NR-LV9QN88PC6/I5
  - 2022/07 時点で2000円強程度と安価



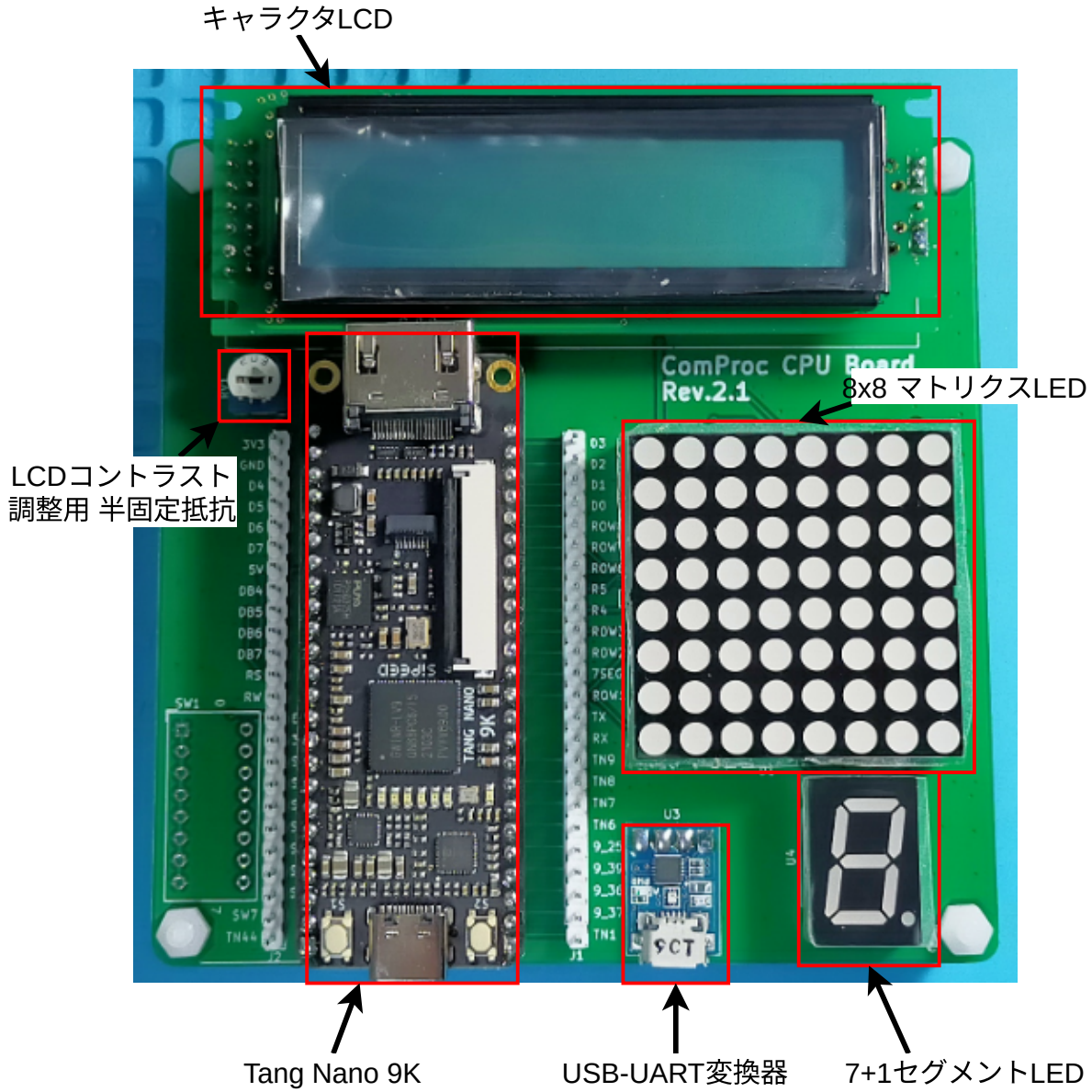
ComProc CPU Boardの機能

- 8x8 赤色マトリクスLED
- 7+1セグメントLED
- FPGAからダイナミック点灯により制御可能



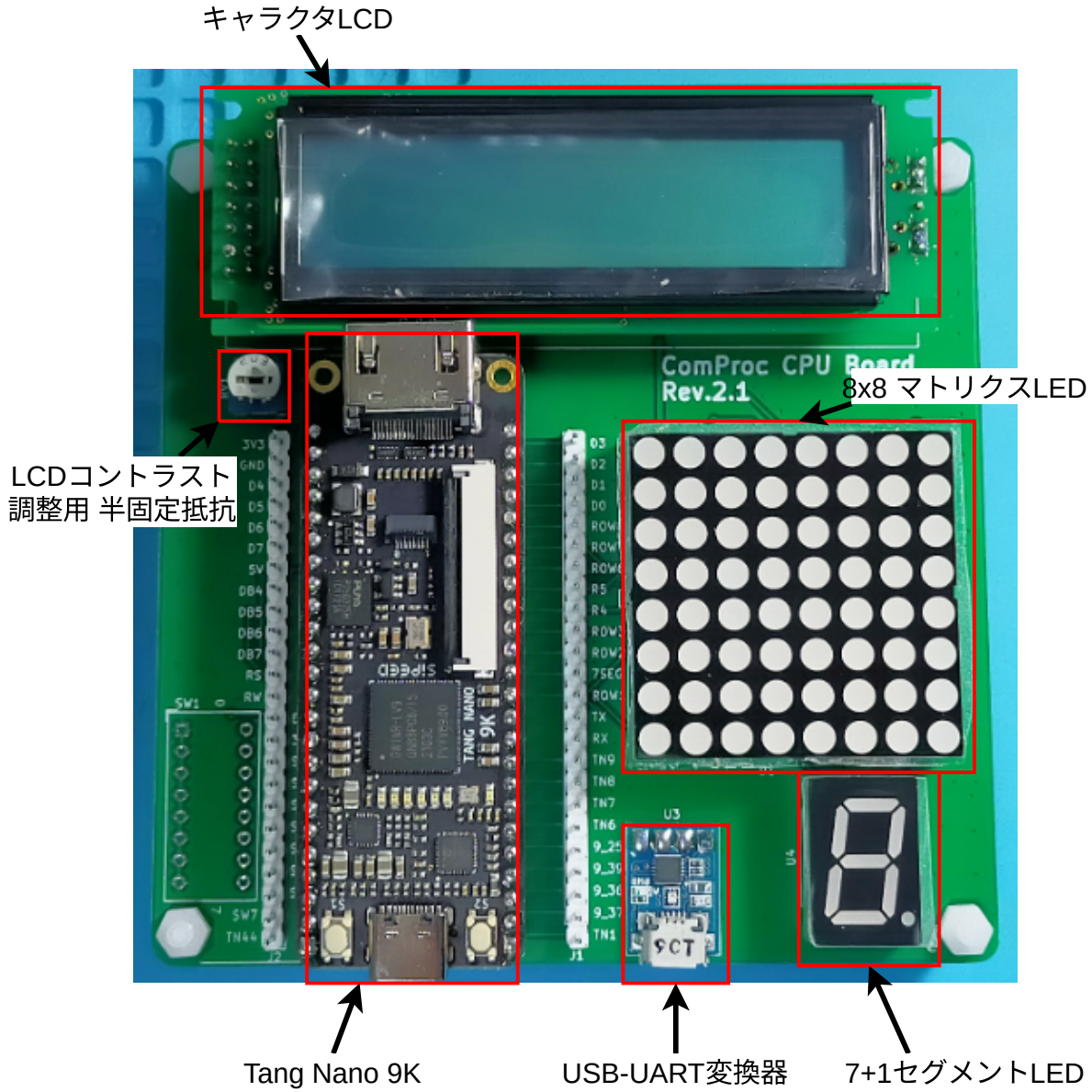
ComProc CPU Boardの機能

- キャラクタLCD
- HD44780互換のLCDコントローラ
- FPGAとは4bitモードで接続



ComProc CPU Boardの機能

- USB-UART変換器
- FPGAとホスト間でUART通信が可能



開発環境(1/2)

- OS
  - Windows: Ubuntu 20.04 on WSL2
  - Linux : Ubuntu 20.04
  - Mac : Ubuntu 20.04 on VirtualBox

開発環境(2/2)

- GOWIN EDA Education版 1.9.8.03 or later
  - Standard版ライセンスがあるならそれでも可
- openFPGALoader
  - FPGAへの回路情報書き込みツール
- Icarus Verilog (RTLシミュレータ)

ComProc CPU Boardの動作確認

- Tang Nano 9KのUSB Type-Cコネクタを PCや電源に接続
- 下記のリンク先の動画のように動作すること
  - https://twitter.com/i/status/1543679412818034689
- 8x8マトリクスLEDが左上から順に点灯→消灯を繰り返す
- 7+1セグメントLEDの各セグメントが順に点灯→消灯を繰り返す
- キャラクタLCDに A~Z の文字が順に表示される
- Tang Nano 9K上のLEDが順に点灯→消灯を繰り返す

FPGAのデザイン合成確認

- 出荷時のファームウェアとことなるデザインを書き込んで
  FPGAのデザインの論理合成・書き込みを行えることを確認する
- 論理合成とは
  - Verilog HDLなどで記述された論理回路をFPGAに書き込み可能な形式 (**ビットストリーム**) に変換することを指す
  - 厳密には **論理合成 (Logic Synthesis)** はHDLからFPGAに書き込み可能なビットストリームを生成するプロセスの一部をさすが、ここでは説明を割愛する
- GUIで行う方法、CUIで行う方法の2つがある

GUIを使った作業手順

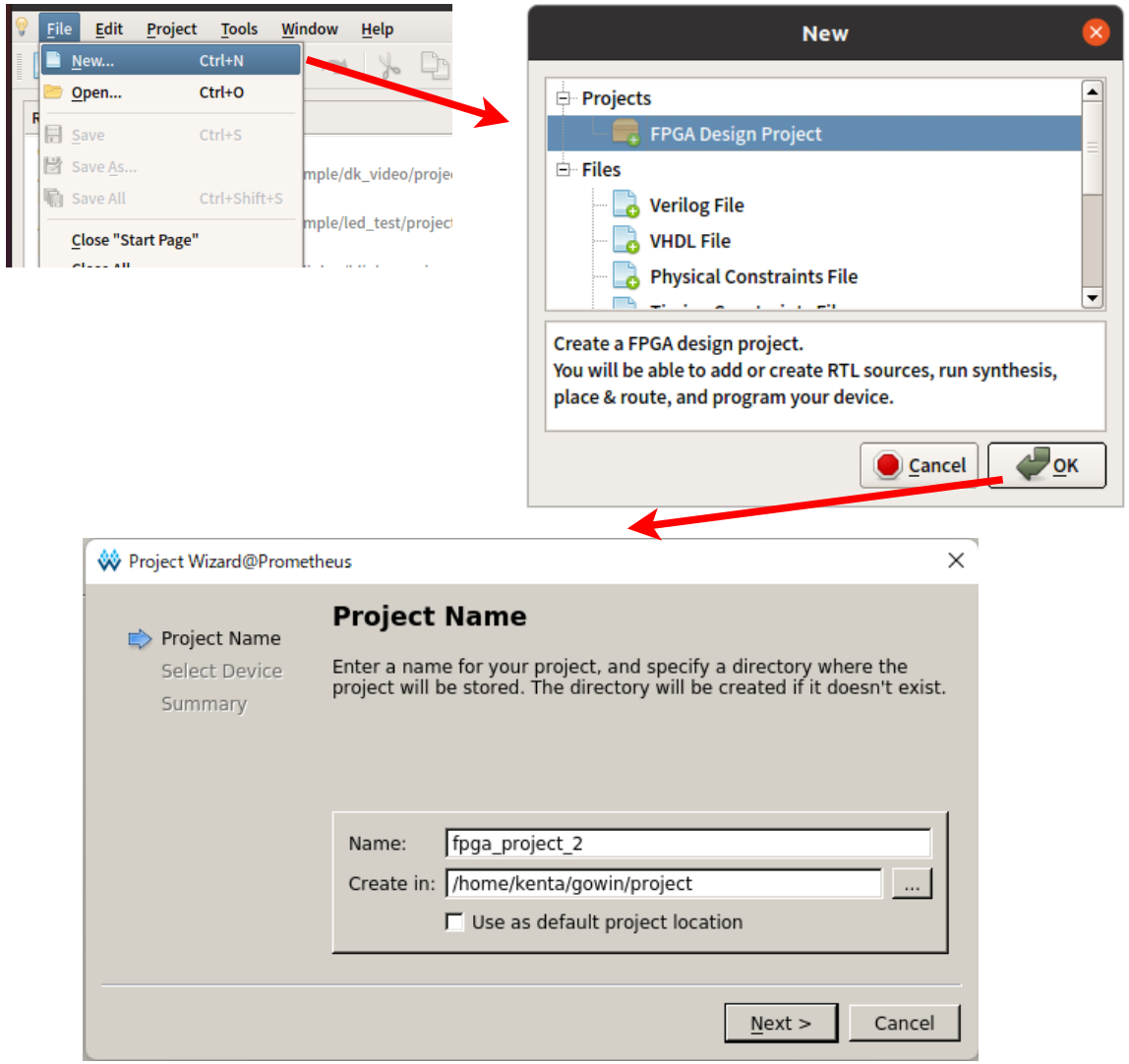
- GOWIN EDAのGUIを使ったデザインの合成手順
- GOWIN EDAのインストールパスを `$GOWIN_HOME` とする
  - e.g. `~/gowin/1.9.8.03_edu`
- まずは GOWIN EDAを起動する

```
$ $GOWIN_HOME/IDE/bin/gw_ide&
```

- Windowsの場合はスタートメニューからGowin IDEを起動

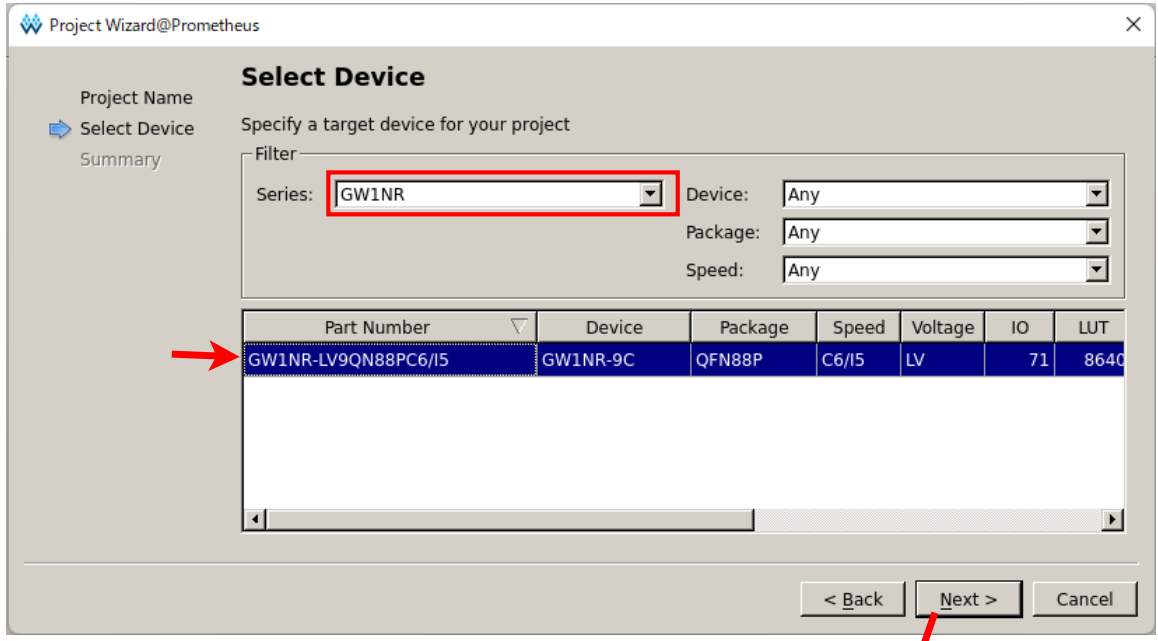
プロジェクトの作成(1/2)

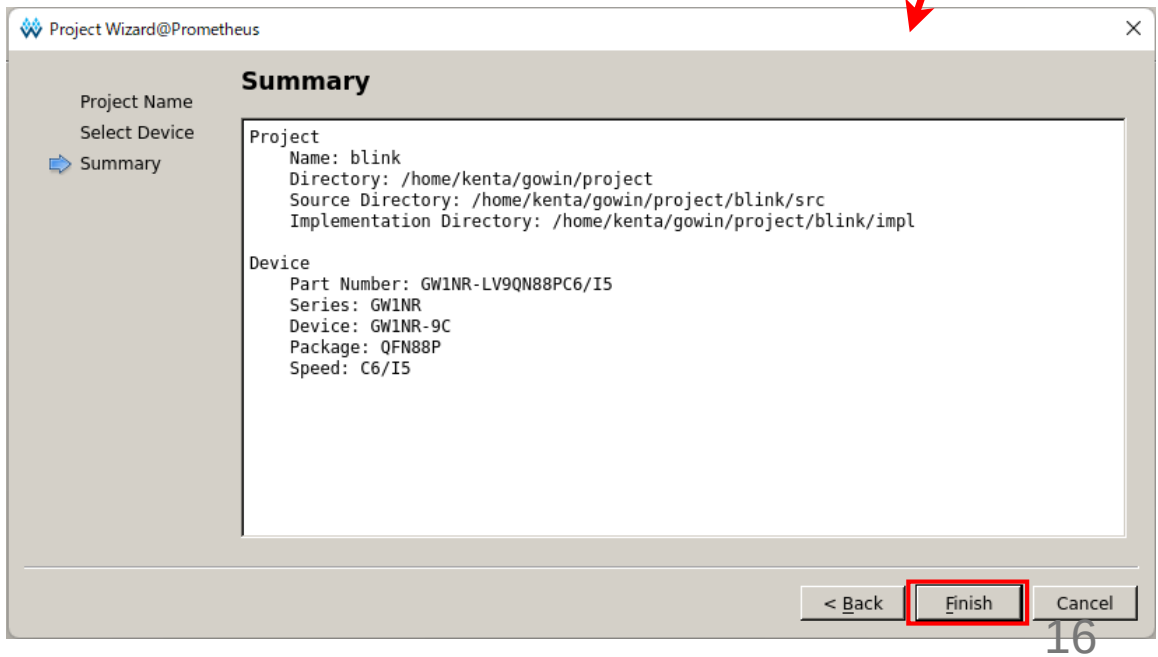
1. **File -> New** メニューを選んで、**New** ダイアログで **FPGA Design Project** を選んで **OK** を押す
2. **Project Name** 画面で **Name** に `blink` 、 **Create in** に適当なホームディレクトリ以下のディレクトリを指定して **Next** を押す



プロジェクトの作成(2/2)

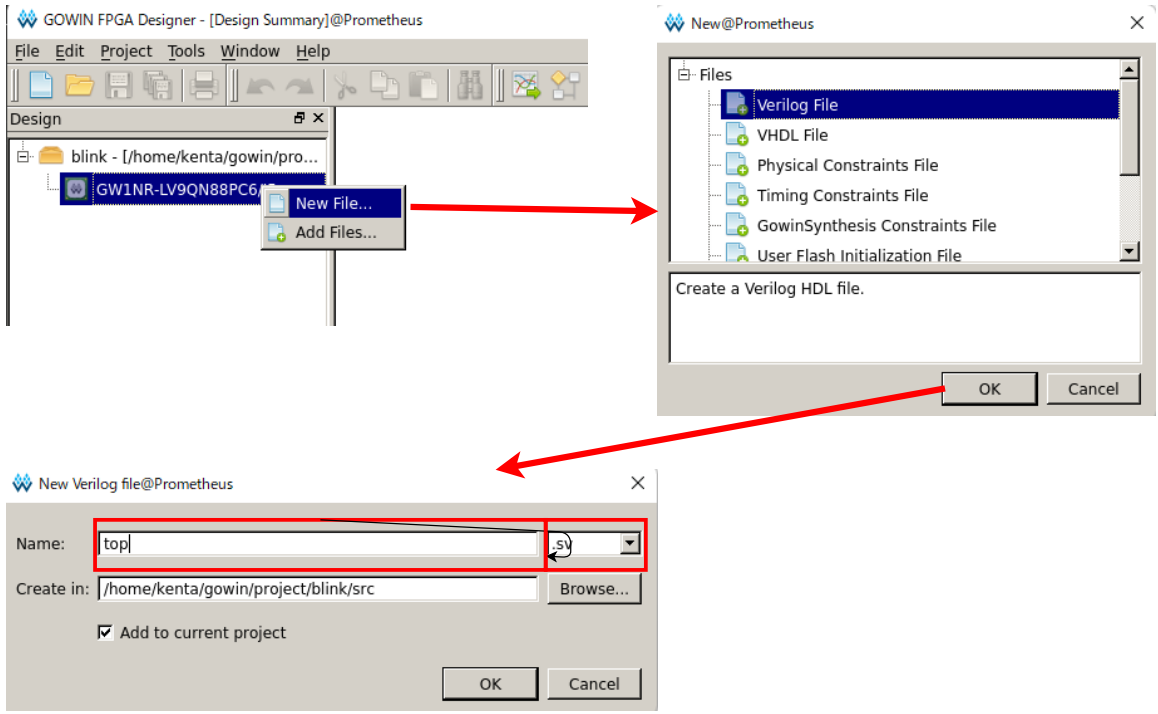
3. **Series** で **GW1NR** を選択し、 **Part Number** で **GW1NR-LV9QN88PC6/I5** を選択し、 **Next** を押す
4. **Summary** 画面で **Finish** を押す





HDLファイルの追加(1/2)

1. **Design** 画面を右クリックして **New File** を選択する
2. **Verilog File** を選択して **OK** を押す
3. **Name** に **top** と入力し、拡張子リストから **.sv** を選択して **OK** を押す



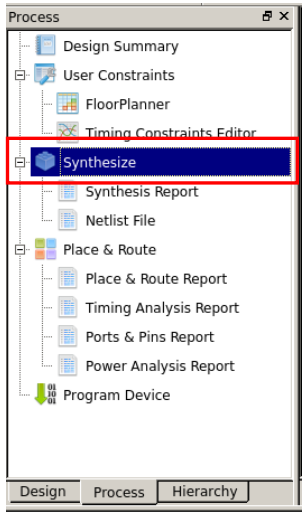
HDLファイルの追加(2/2)

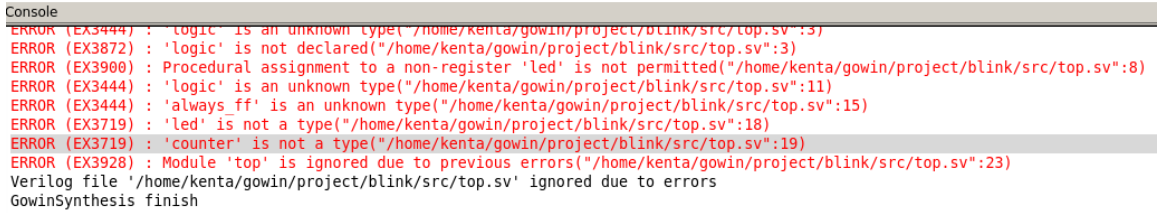
- 以下の内容を **top.sv** に追加

```
module top(
    input wire          clock,
    output logic [5:0]  led
);


initial begin
    led <= 0;
end

logic [24:0] counter = 0;

always_ff @(posedge clock) begin
    if( counter < 25'd27_000_000 ) begin
        counter++;
    end
    else begin
        led <= ~led;
        counter <= '0;
    end
end

endmodule
```

論理合成(1/4)

1. **Process** タブの **Synthesis** をダブルクリックする
2. **Console** ウィンドウにエラーメッセージがたくさん出力される。



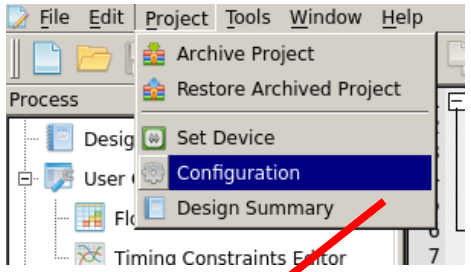


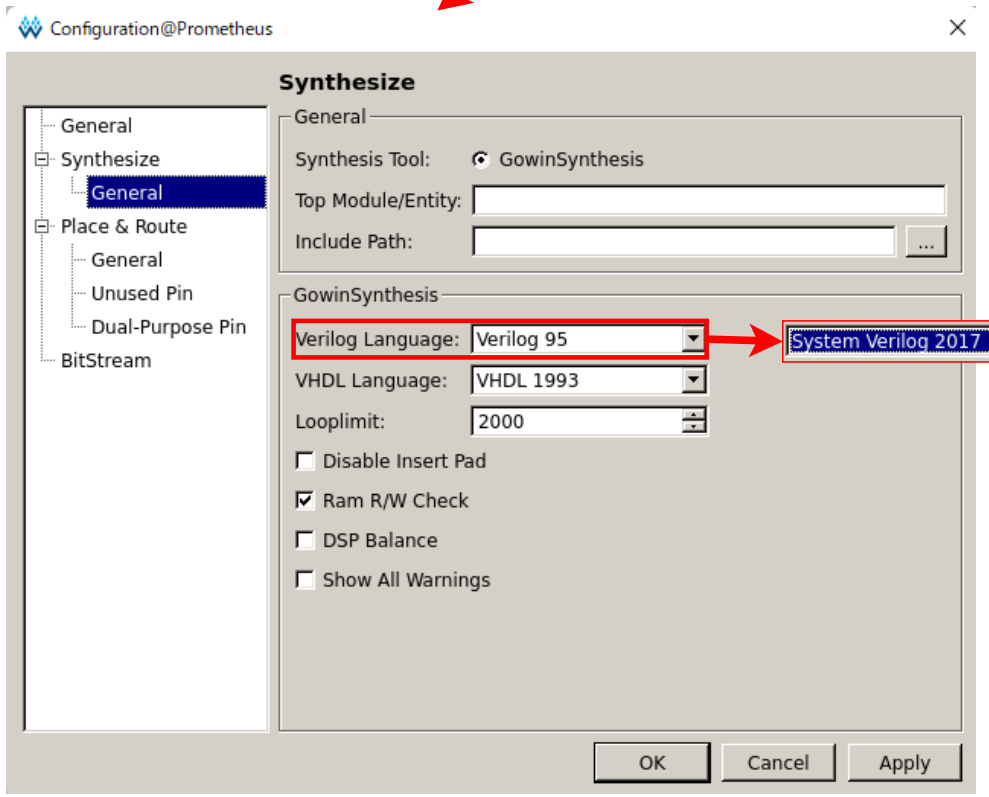
論理合成(2/4)

- プロジェクトのVerilog言語バージョンの設定がデフォルトでVerilog 95なのが問題
- 拡張子 **.sv** でもSystemVerilog扱いにならない

論理合成(3/4)

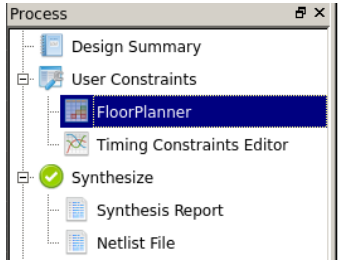
1. **Project -> Configuration** メニューを選択する
2. **Synthesize - General** を選択する
3. **Verilog Language** を **System Verilog 2017** に設定する。

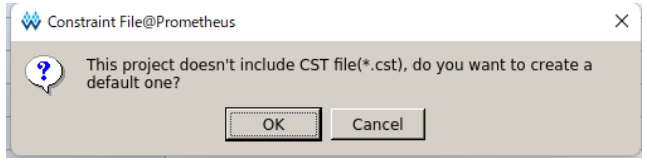




フロアプラン - IO制約

- デザインのポートとFPGAのピンの対応付けを行う
- ピンの情報はTang Nano 9Kの回路図を確認する





I/O Constraints

	Port	Direction	Diff Pair	Location	Bank	Exclusive	IO Type	Drive	Pull Mode
1	clock	input		drag or type ...		False	LVCMOS18	N/A	UP
2	led[0]	output		drag or type ...		False	LVCMOS18	8	UP
3	led[1]	output		drag or type ...		False	LVCMOS18	8	UP
4	led[2]	output		drag or type ...		False	LVCMOS18	8	UP
5	led[3]	output		drag or type ...		False	LVCMOS18	8	UP
6	led[4]	output		drag or type ...		False	LVCMOS18	8	UP
7	led[5]	output		drag or type ...		False	LVCMOS18	8	UP

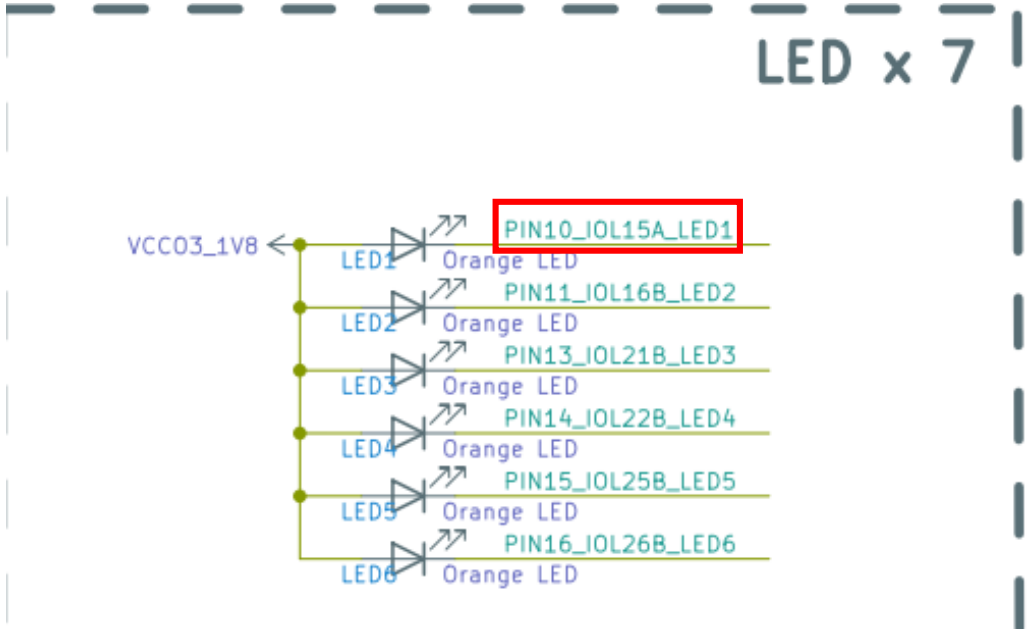
Message | I/O Constraints | Primitive Constraints | Group Constraints | Resource Reservation | Clock Assignment | Quadrant Constraints

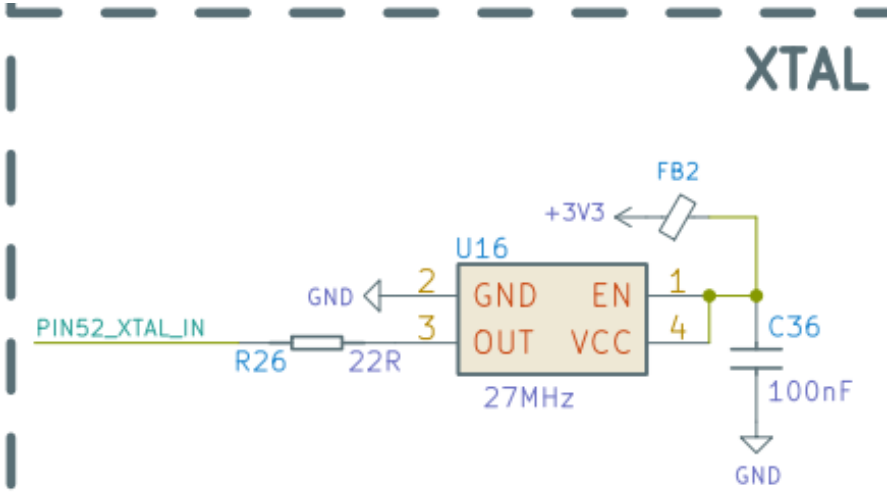
Tang Nano 9Kの回路図

- Tang Nano 9Kの情報はSipeedのサイトにある
- https://dl.sipeed.com/shareURL/TANG/Nano 9K/2_Schematic

対象のピンを確認

- Tang Nano 9Kの回路図では、ネット名にFPGAのピン番号が記載されている
  - 他のボードだとそのかぎりではないので、回路図を追いかける必要がある



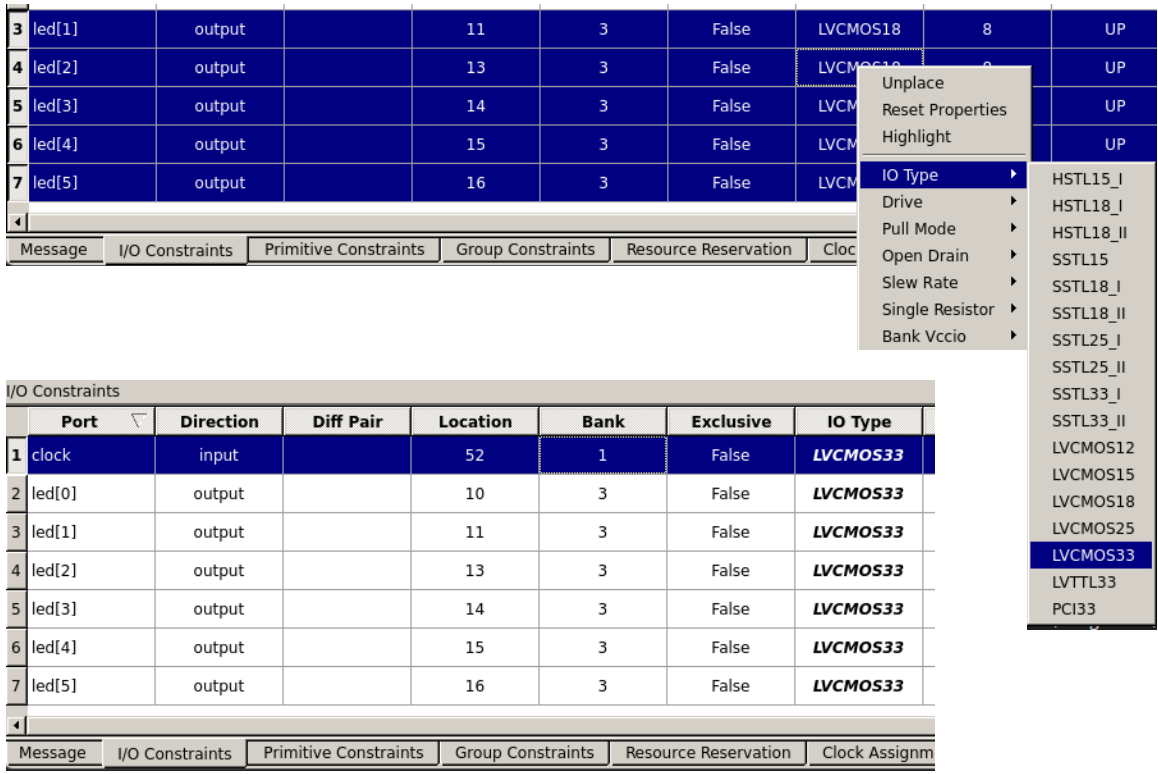


IO制約の設定

1. **IO Constraints** の全ての行を選択して右クリックし、 **IO Type -> LVCMOS33** を選択
2. 各ピンの **Location** を画像の通り設定

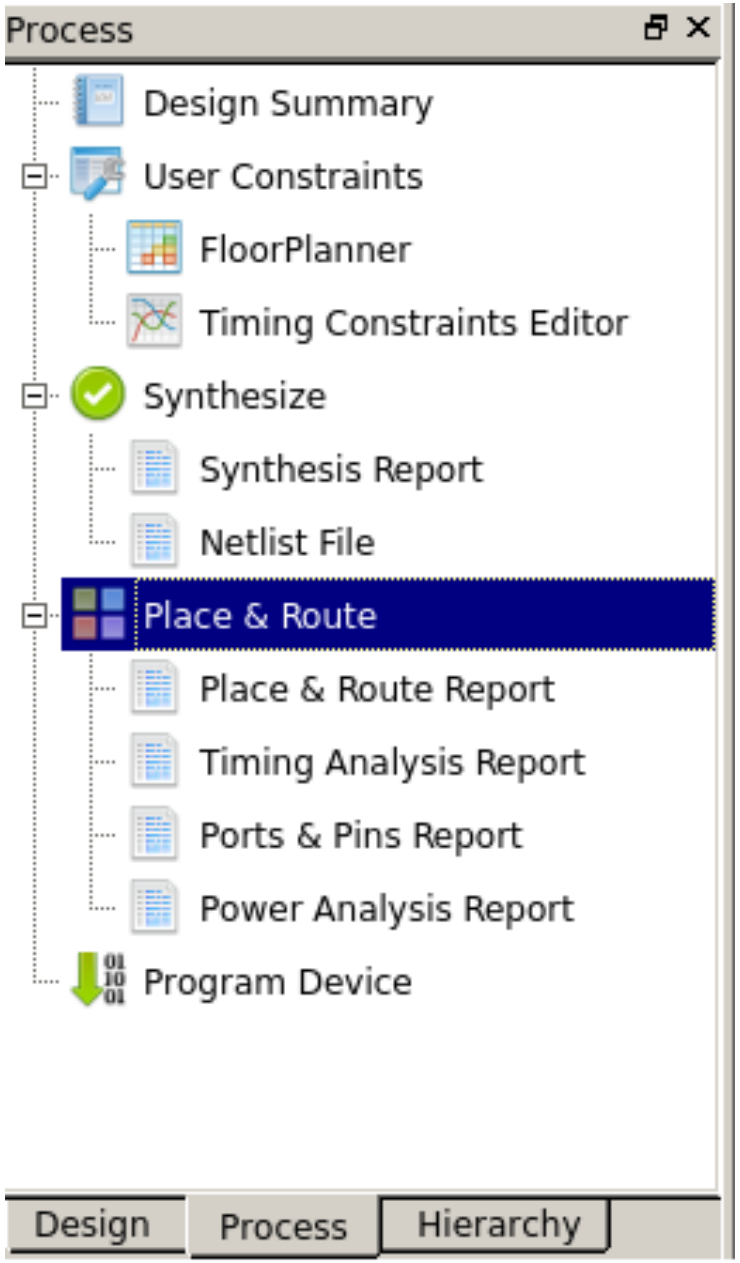
- 回路図と比較してみること

3. **Ctrl+S** を入力して保存したのち、FloorPlannerウィンドウを閉じる



配置配線

- IO制約を行ったら、配置配線を行いビットストリームを生成する
- **Place & Route** をダブルクリックする。配置配線とビットストリーム生成が実行される
- 成功すると `(プロジェクトのディレクトリ)/impl/pnr/blink.fs` が生成される。



ビットストリームの書き込み準備

- **openFPGALoader** を使ってビットストリームを書き込む
  - https://github.com/trabucayre/openFPGALoader
- まずは **openFPGALoader** をインストールする

openFPGALoaderのインストール (Linux)

- openFPGALoaderのリリースページからバイナリをダウンロードして展開
  - https://github.com/trabucayre/openFPGALoader/releases/tag/v0.8.0

```
$ mkdir -p ~/openFPGALoader && curl -L https://github.com/trabucayre/openFPGALoader/releases/download/v0.8.0/ubtuntu20.04-openFPGALoader.tgz | tar zx -C ~/openFPGALoader
$ export PATH=~/openFPGALoader/usr/local/bin:$PATH # パスを通しておく
```

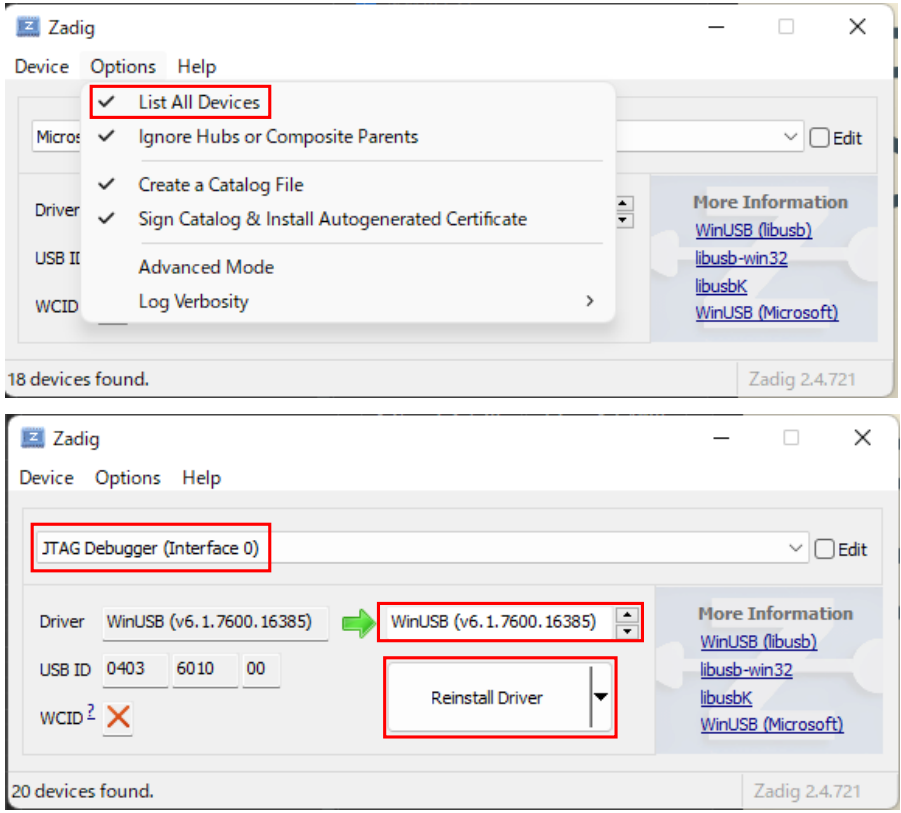
openFPGALoaderのインストール (Windows) (1/3)

- openFPGALoaderのリリースページのバイナリはMSYS用で面倒なので、Windows向けスタティックリンク版を用意
  - https://github.com/ciniml/seccamp_2022_riscv_cpu/tree/main/blob/openFPGALoader-0.8.0_win.zip

```
$ mkdir -p ~/openFPGALoader && curl -OL https://github.com/ciniml/seccamp_2022_riscv_cpu/raw/main/blob/openFPGALoader-0.8.0_win.zip
$ unzip -u openFPGALoader-0.8.0_win.zip -d ~/openFPGALoader
$ export PATH=~/openFPGALoader/openFPGALoader-win/bin:$PATH
```

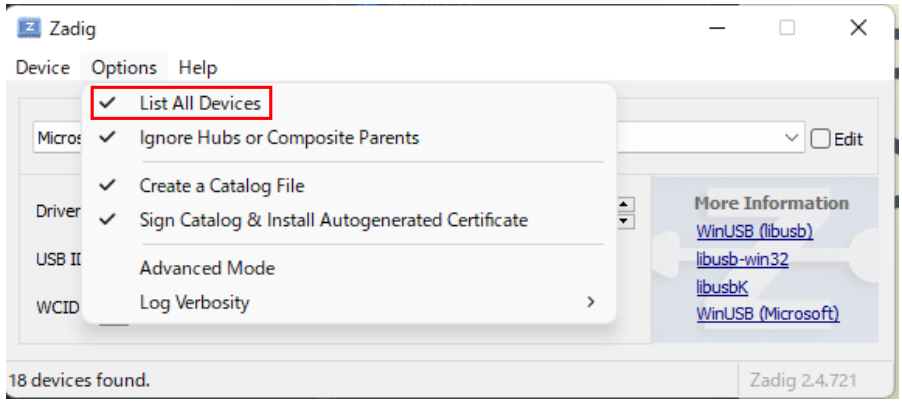
openFPGALoaderのインストール (Windows) (2/3)

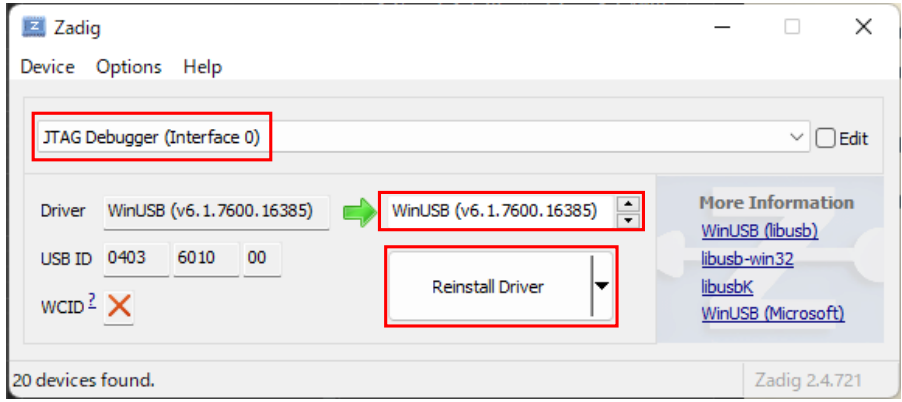
- openFPGALoaderから書き込み回路にアクセスするために **Zadig** でドライバを差し替える
  - https://github.com/pbatard/libwdi/releases/download/v1.4.1/zadig-2.7.exe



openFPGALoaderのインストール (Windows) (3/3)

1. **Options -> List All Devices** にチェックを入れる
2. PCにTang Nano 9Kを接続し、リストボックスから **JTAG Debugger (Interface 0)** を選択する
3. **Driver** の右側のリストから **WinUSB** を選択する
4. **Reinstall Driver** を押す





openFPGALoaderの動作確認

- Tang Nano 9Kを検出してみる
- 以降、Windowsでは、 `openFPGALoader` を `openFPGALoader.exe` に置き換えて実行すること

```
$ openFPGALoader --detect
Jtag frequency : requested 6.00MHz   -> real 6.00MHz
index 0:
        idcode 0x100481b
        manufacturer Gowin
        family GW1N
        model  GW1N(R)-9C
        irlength 8
```

ビットストリームの書き込み

- 先ほど生成したビットストリームを書き込む
- `path/to/blink.fs` は `(プロジェクトのディレクトリ)/impl/pnr/blink.fs` のパスに置き換えること

```
$ openFPGALoader --board tangnano9k --write-sram path/to/blink.fs
write to ram
Jtag frequency : requested 6.00MHz   -> real 6.00MHz
Parse file Parse blink.fs:
Done
DONE
Jtag frequency : requested 2.50MHz   -> real 2.00MHz
erase SRAM Done
Flash SRAM: [==================================================] 100.00%
Done
SRAM Flash: Success
```

まとめ

- FPGAの書き込み手順については各FPGAベンダーのツール毎にことなるが、流れは大体同じ
  - 論理合成、制約の設定、配置配線、ビットストリーム生成、書き込み
- ここまでの手順でComProc CPU Boardを使う準備が整った