再構成型アーキテクチャ特論 （10）

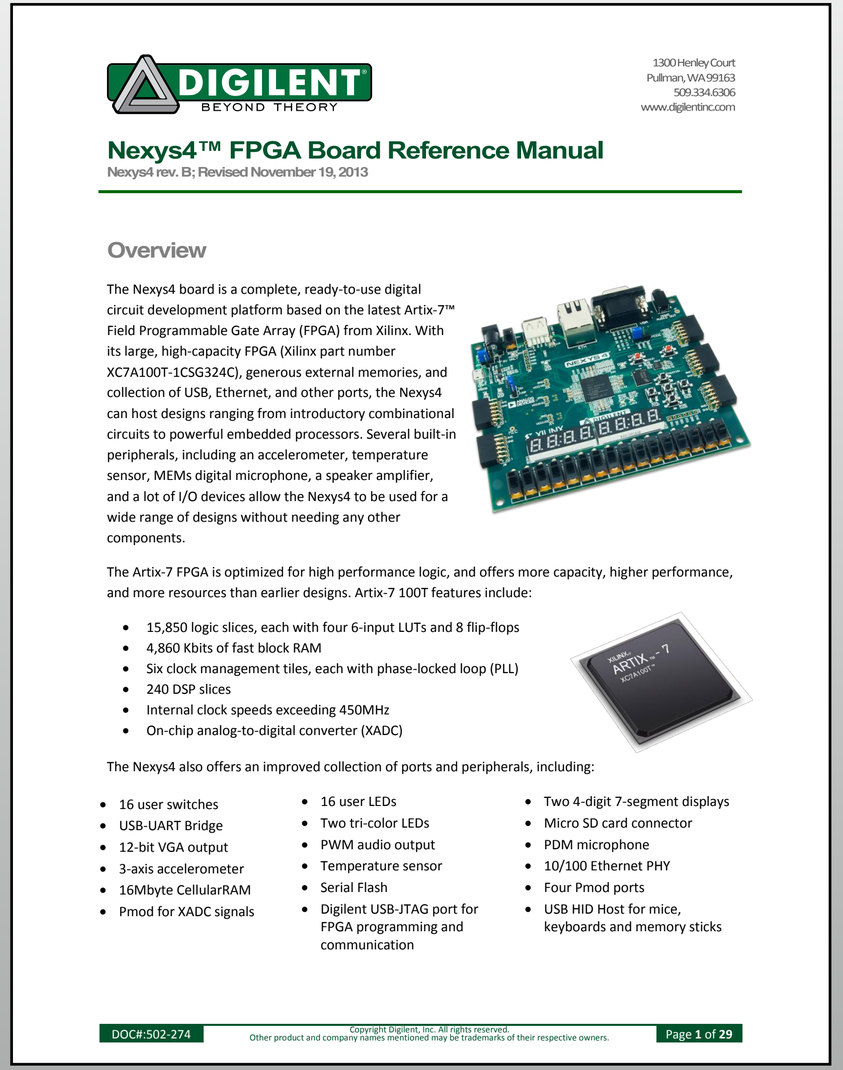
osana@eee.u-ryukyu.ac.jp

今週の内容

- 周辺デバイスを動かしましょう
  - PWM で LED を点灯させる
  - I²C (Inter-Integrated Circuit) と温度センサ
  - SPI (Serial Peripheral Interface) と加速度センサ
- 実機でのオンチップデバッグ

資料編 (1): ボードのマニュアル

- digilentinc.com → Products → FPGA Boards → Nexys4
  - まずはリファレンスマニュアル
  - 載っているデバイスと簡単な説明



DIGILENT
BEYOND THEORY

1300 Henley Court
Pullman, WA 99163
509.334.6306
www.digilentinc.com

Nexys4™ FPGA Board Reference Manual

Nexys4 rev. B; Revised November 19, 2013

Overview

The Nexys4 board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx. With its large, high-capacity FPGA (Xilinx part number XC7A100T-1CSG324C), generous external memories, and collection of USB, Ethernet, and other ports, the Nexys4 can host designs ranging from introductory combinational circuits to powerful embedded processors. Several built-in peripherals, including an accelerometer, temperature sensor, MEMs digital microphone, a speaker amplifier, and a lot of I/O devices allow the Nexys4 to be used for a wide range of designs without needing any other components.

The Artix-7 FPGA is optimized for high performance logic, and offers more capacity, higher performance, and more resources than earlier designs. Artix-7 100T features include:

- 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops
- 4,860 Kbits of fast block RAM
- Six clock management tiles, each with phase-locked loop (PLL)
- 240 DSP slices
- Internal clock speeds exceeding 450MHz
- On-chip analog-to-digital converter (XADC)

The Nexys4 also offers an improved collection of ports and peripherals, including:

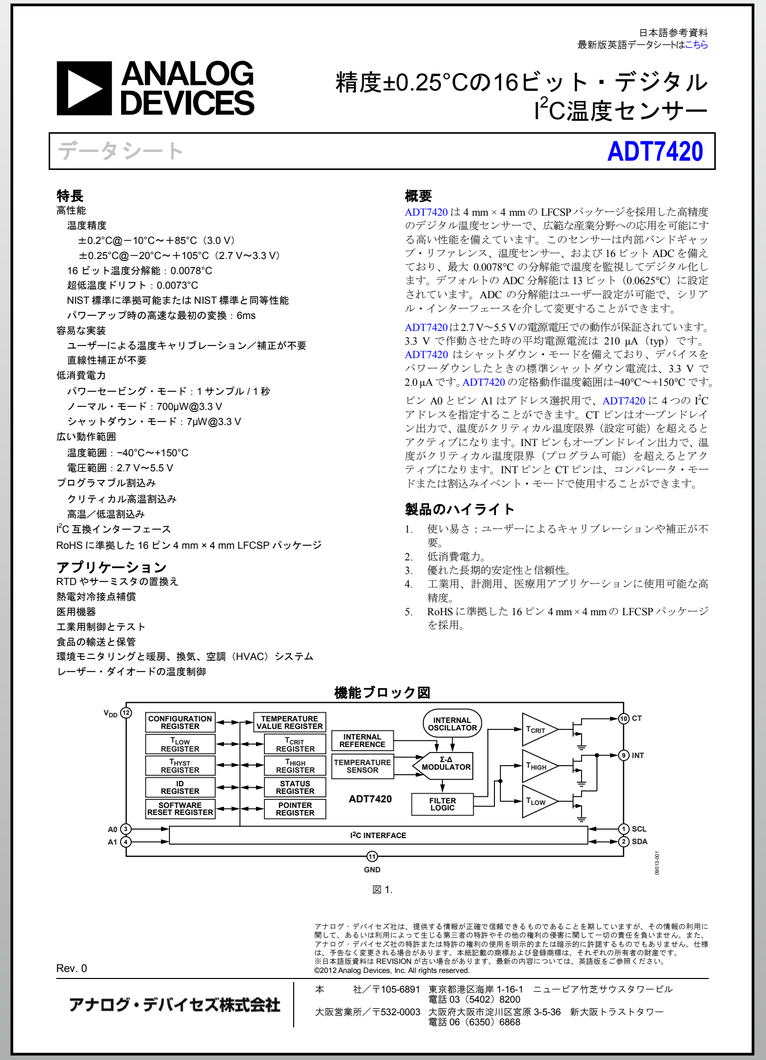
- 16 user switches
- USB-UART Bridge
- 12-bit VGA output
- 3-axis accelerometer
- 16Mbyte CellularRAM
- Pmod for XADC signals
- 16 user LEDs
- Two tri-color LEDs
- PWM audio output
- Temperature sensor
- Serial Flash
- Digilent USB-JTAG port for FPGA programming and communication
- Two 4-digit 7-segment displays
- Micro SD card connector
- PDM microphone
- 10/100 Ethernet PHY
- Four Pmod ports
- USB HID Host for mice, keyboards and memory sticks

資料編 (2): Master XDC

- ボード上のデバイスに接続されているピンの名前と IOSTANDARD
  - ピンの名前はメーカーが付けたもの
  - もちろん自分で付け替えてもよい
- あとファイルが案外見づらいです、が、間違えを避けるためには便利
- マニュアルと同じところからダウンロードできる

資料編 (3): デバイスのデータシート

- デバイスの型番はボードのマニュアルに掲載
  - 詳しい使い方は各デバイスのデータシートを
  - デバイスのベンダの web サイトから入手
  - けっこう日本語になっています



日本語参考資料
最新版英語データシートはこちら

ANALOG DEVICES

精度±0.25°Cの16ビット・デジタル I²C温度センサー

データシート **ADT7420**

特長

高性能
- 温度精度
  - ±0.2°C@−10°C～+85°C（3.0 V）
  - ±0.25°C@−20°C～+105°C（2.7 V～3.3 V）
- 16 ビット温度分解能：0.0078°C
- 超低温度ドリフト：0.0073°C
- NIST 標準に準拠可能または NIST 標準と同等性能
- パワーアップ時の高速な最初の変換：6ms

容易な実装
- ユーザーによる温度キャリブレーション／補正が不要
- 直線性補正が不要

低消費電力
- パワーセービング・モード：1 サンプル / 1 秒
- ノーマル・モード：700μW@3.3 V
- シャットダウン・モード：7μW@3.3 V

広い動作範囲
- 温度範囲：−40°C～+150°C
- 電圧範囲：2.7 V～5.5 V

プログラマブル割込み
- クリティカル高温割込み
- 高温／低温割込み

I²C 互換インターフェース

RoHS に準拠した 16 ピン 4 mm × 4 mm LFCSP パッケージ

アプリケーション

RTD やサーミスタの置換え
熱電対冷接点補償
医用機器
工業用制御とテスト
食品の輸送と保管
環境モニタリングと暖房、換気、空調（HVAC）システム
レーザー・ダイオードの温度制御

概要

ADT7420 は 4 mm × 4 mm の LFCSP パッケージを採用した高精度のデジタル温度センサーで、広範な産業分野への応用を可能にする高い性能を備えています。このセンサーは内部バンドギャップ・リファレンス、温度センサー、および 16 ビット ADC を備えており、最大 0.0078°C の分解能で温度を監視してデジタル化します。デフォルトの ADC 分解能は 13 ビット（0.0625°C）に設定されています。ADC の分解能はユーザー設定が可能で、シリアル・インターフェースを介して変更することができます。

ADT7420は2.7 V～5.5 Vの電源電圧での動作が保証されています。3.3 V で作動させた時の平均電源電流は 210 μA（typ）です。ADT7420 はシャットダウン・モードを備えており、デバイスをパワーダウンしたときの標準シャットダウン電流は、3.3 V で 2.0 μA です。ADT7420 の定格動作温度範囲は−40°C～+150°C です。

ピン A0 とピン A1 はアドレス選択用で、ADT7420 に 4 つの I²C アドレスを指定することができます。CT ピンはオープンドレイン出力で、温度がクリティカル温度限界（設定可能）を超えるとアクティブになります。INT ピンもオープンドレイン出力で、温度がクリティカル温度限界（プログラム可能）を超えるとアクティブになります。INT ピンと CT ピンは、コンパレータ・モードまたは割込みイベント・モードで使用することができます。

製品のハイライト

1. 使い易さ：ユーザーによるキャリブレーションや補正が不要。
2. 低消費電力。
3. 優れた長期的安定性と信頼性。
4. 工業用、計測用、医療用アプリケーションに使用可能な高精度。
5. RoHS に準拠した 16 ピン 4 mm × 4 mm の LFCSP パッケージを採用。

機能ブロック図

図 1.

Rev. 0

アナログ・デバイセズ株式会社

本　　社／〒105-6891　東京都港区海岸 1-16-1　ニューピア竹芝サウスタワービル
電話 03（5402）8200

大阪営業所／〒532-0003　大阪府大阪市淀川区宮原 3-5-36　新大阪トラストタワー
電話 06（6350）6868

Nexys4の周辺デバイス

* スイッチ・7セグメントLED・単色LED・Tri-Color LED
* 温度センサ・3軸加速度センサ・PWM オーディオ出力・PDM マイク
* 128Mb 疑似SRAM (CellularRAM: Verilog モデルあり)
* VGA (ディスプレイ)・USB-PS/2 (キーボード・マウス)
* USB-UART (シリアル)・10/100M Ethernet (LAN)

On/OFF 制御で簡単

- **スイッチ**・7セグメントLED・**単色LED**・Tri-Color LED
- 温度センサ・3軸加速度センサ・PWM オーディオ出力・PDM マイク
- 128Mb 疑似SRAM (CellularRAM: Verilog モデルあり)
- VGA (ディスプレイ)・USB-PS/2 (キーボード・マウス)
- USB-UART (シリアル)・10/100M Ethernet (LAN)

ダイナミック駆動とか PWM とか

* スイッチ・**7セグメントLED**・単色LED・**Tri-Color LED**
* 温度センサ・3軸加速度センサ・PWM オーディオ出力・PDM マイク
* 128Mb 疑似SRAM (CellularRAM: Verilog モデルあり)
* VGA (ディスプレイ)・USB-PS/2 (キーボード・マウス)
* USB-UART (シリアル)・10/100M Ethernet (LAN)

標準シリアル・インタフェイス (SPI/I2C)

* スイッチ・7セグメントLED・単色LED・Tri-Color LED
* **温度センサ・3軸加速度センサ**・PWM オーディオ出力・PDM マイク
* 128Mb 疑似SRAM (CellularRAM: Verilog モデルあり)
* VGA (ディスプレイ)・USB-PS/2 (キーボード・マウス)
* USB-UART (シリアル)・10/100M Ethernet (LAN)

標準シリアル・インタフェイス

- スイッチ・7セグメントLED・単色LED・Tri-Color LED
- 温度センサ・3軸加速度センサ・PWM オーディオ出力・PDM マイク
- 128Mb 疑似SRAM (CellularRAM: Verilog モデルあり)
- VGA (ディスプレイ)・**USB-PS/2 (キーボード・マウス)**
- **USB-UART (シリアル)**・10/100M Ethernet (LAN)

わりと簡単に作れるもの

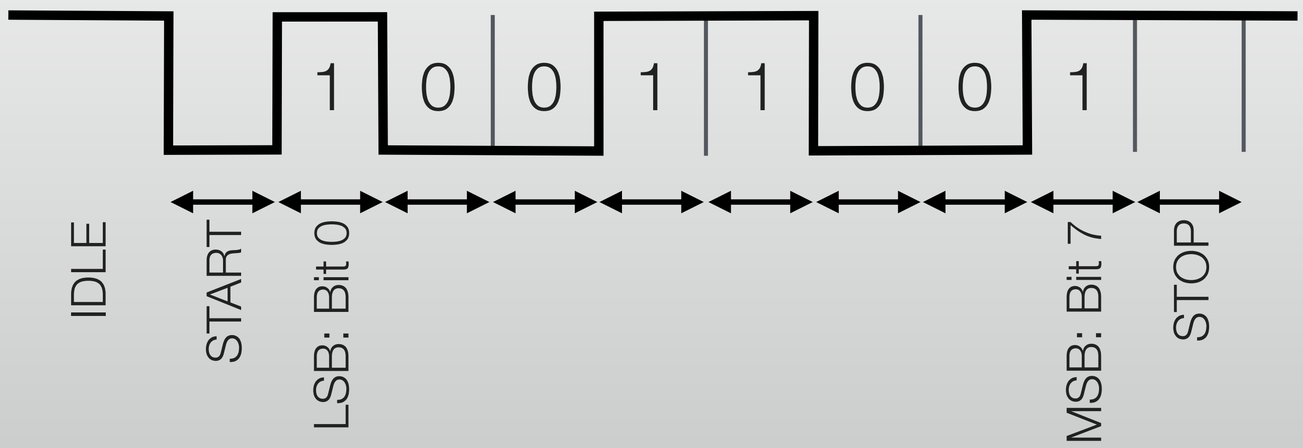
- スイッチ・7セグメントLED・単色LED・Tri-Color LED
- 温度センサ・3軸加速度センサ・**PWM オーディオ出力・PDM マイク**
- 128Mb 疑似SRAM (CellularRAM: Verilog モデルあり)
- **VGA (ディスプレイ)**・USB-PS/2 (キーボード・マウス)
- USB-UART (RS-232C シリアル)・10/100M Ethernet (LAN)

シリアル・インタフェイス

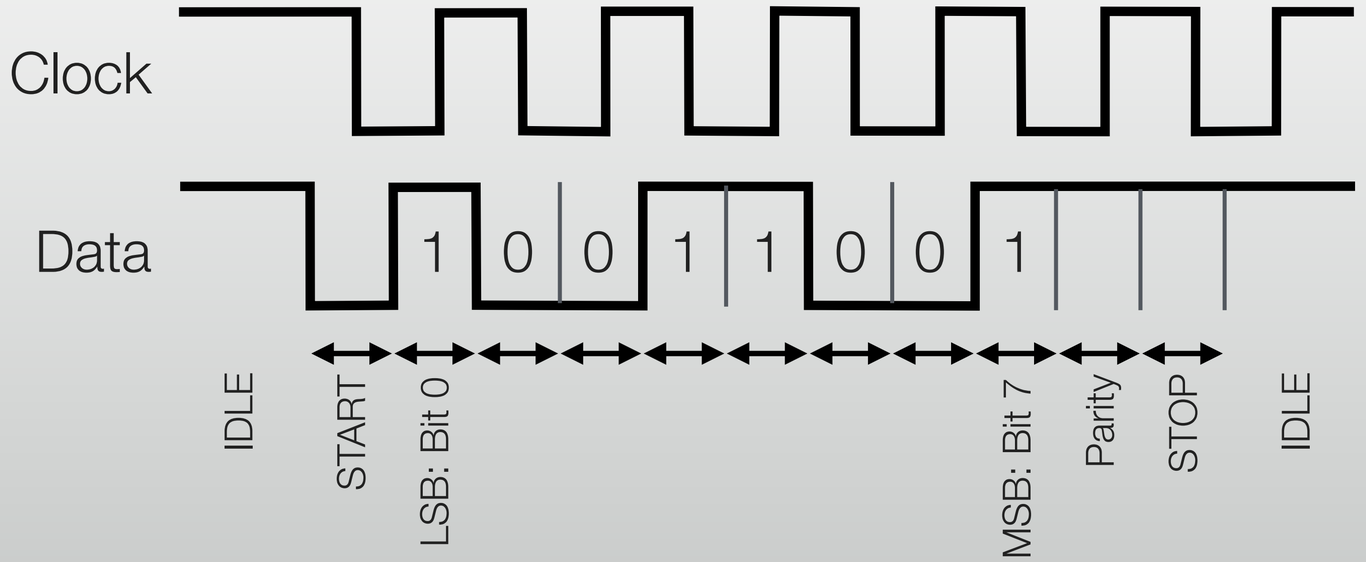
- 1本の信号線でデータを送受信
  - 独立したクロック信号線を使うもの / 使わないもの
  - 送信と受信で別の信号線を使うもの / 使わないもの

RS-232C

- 送受信各1本の信号線
- 1ビットの長さを決めておく
- アイドル時は High
  - Low になったらスタートで、これで各ビット受信のタイミングが決定
  - たとえば 9600bps なら 1bit = 1/9600sec = 104μs

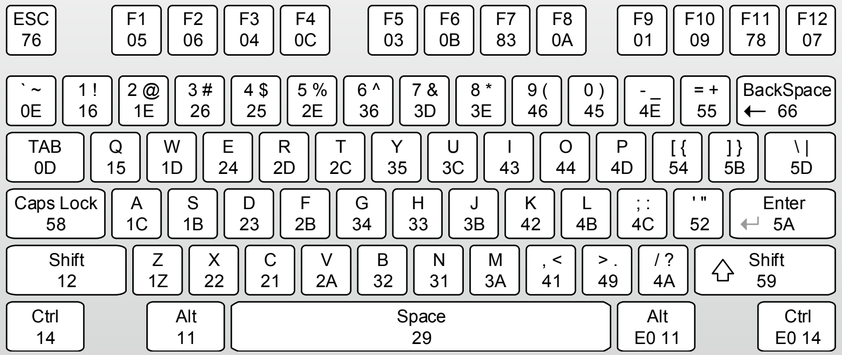


PS/2



- クロックとデータ各1本
  - 双方向通信が可能
  - クロックは普段は停止
  - 受信するだけならパリティは無視すれば OK
- クロックの周期は 60-100μsと長く、これに合わせて送受信

PS/2 Keyboard, Mouse



- キーにはスキャンコードが割り当てられている
  - 押されたらスキャンコードが送信される
  - 離されたら F0 (Key-up) + スキャンコードが送信される
- マウスは移動したりボタンが押されたらデータが送信される

I²C と SPI: 低速周辺機器用 I/O

- I2C: 信号線は2本: SCL (クロック) と SDA (データ・アドレス)
  - アドレスでデバイスを選択できる。400kbps くらいまで
- SPI: 信号線は4本: SCK (クロック)、MISO・MOSI (信号)、SS (Slave Select)
  - デバイスの数だけ SS をもつ。I2C より速い (5Mbpsとか)
  - 並列バージョンもある (Dual SPI / Quad SPI)

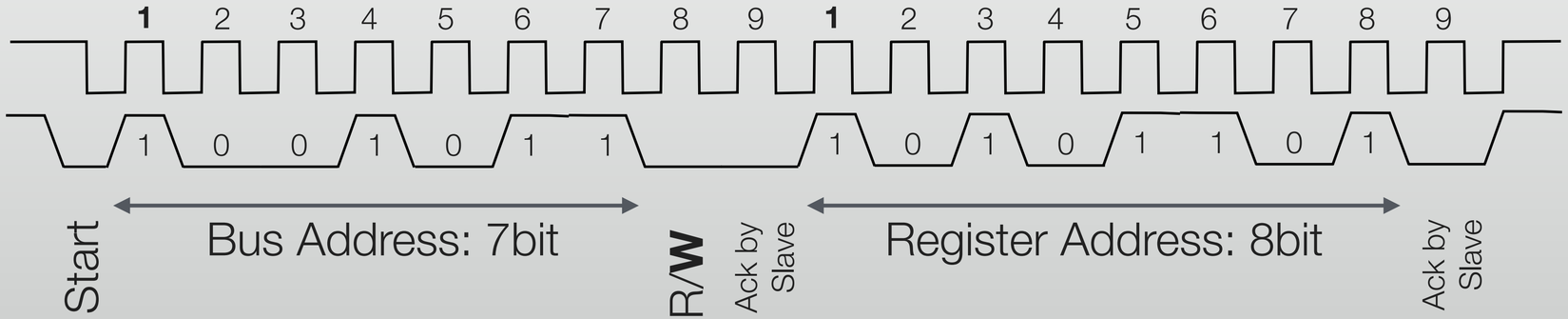
I²C と SPI

- 各デバイスは複数のレジスタを持つ
  - レジスタにはアドレスがついていて、それぞれ役割がある
  - 測定モードの設定を書き込んだり、測定値を読み出したり
- I²C はデバイスも固有のアドレスを持ち、これで選択する
  - SPI の場合には SS 信号線を使ってデバイスを選択する

I²C のプロトコル (Read)

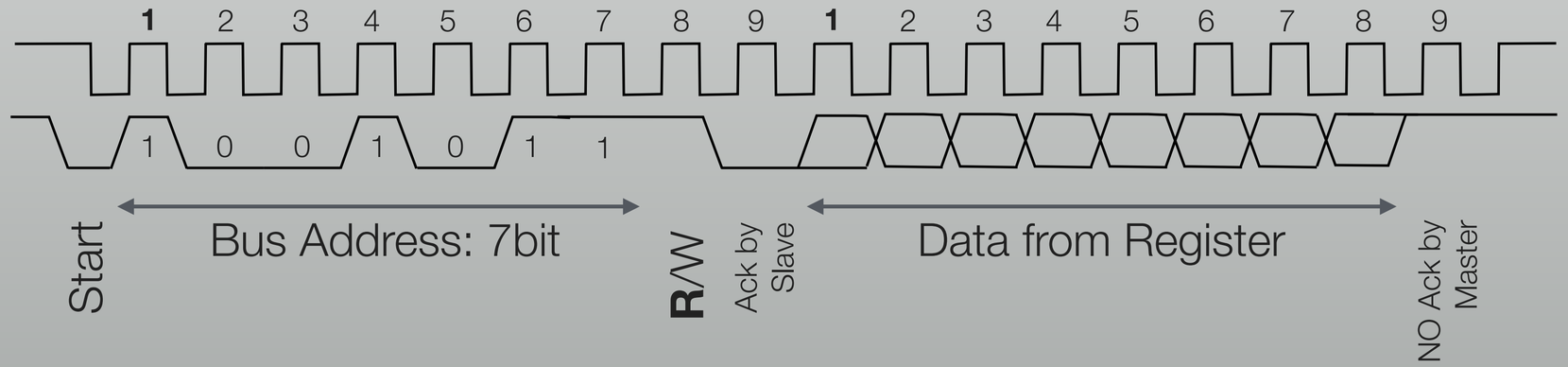
Step 1:

レジスタアドレス
書き込み



Step 2:

データ読みだし



I²C のプロトコル

* Bus address は毎回与えなければならない
* Step 1 のあとに連続してデータを与えればレジスタに書き込みできる
* 同じレジスタを繰り返し読み出すときは Step 2 だけ繰り返せばよい

I²Cの例: ADT7420

- 16bit 温度センサ
  - 正の温度: (16bit の値) / 128
  - 負の温度: (16bit の値-65536) / 128
- デフォルトで温度のレジスタが読み出される
  - 16bit なので、バスアドレスを指定したら8bit を2回うけとる

SPI のプロトコル

- 4線式
  - SS (CS_), SCLK, MOSI (FPGAから出力), MISO (センサから出力)
  - Ack を使わずに CS_ と SCLK が続けば連続アクセスになる
- SPI の例: ADXL362
  - すみませんスライド全然間に合いません (データシートみて…)

課題

- Nexys4 の周辺デバイスを使って何か面白いのを作りましょう
- 以下のサンプルコードは提供します
  - PS/2 ・温度センサ・加速度センサ受信回路・PDM Microphone
  - LED 用の PWM

合成・配線後のシミュレーション

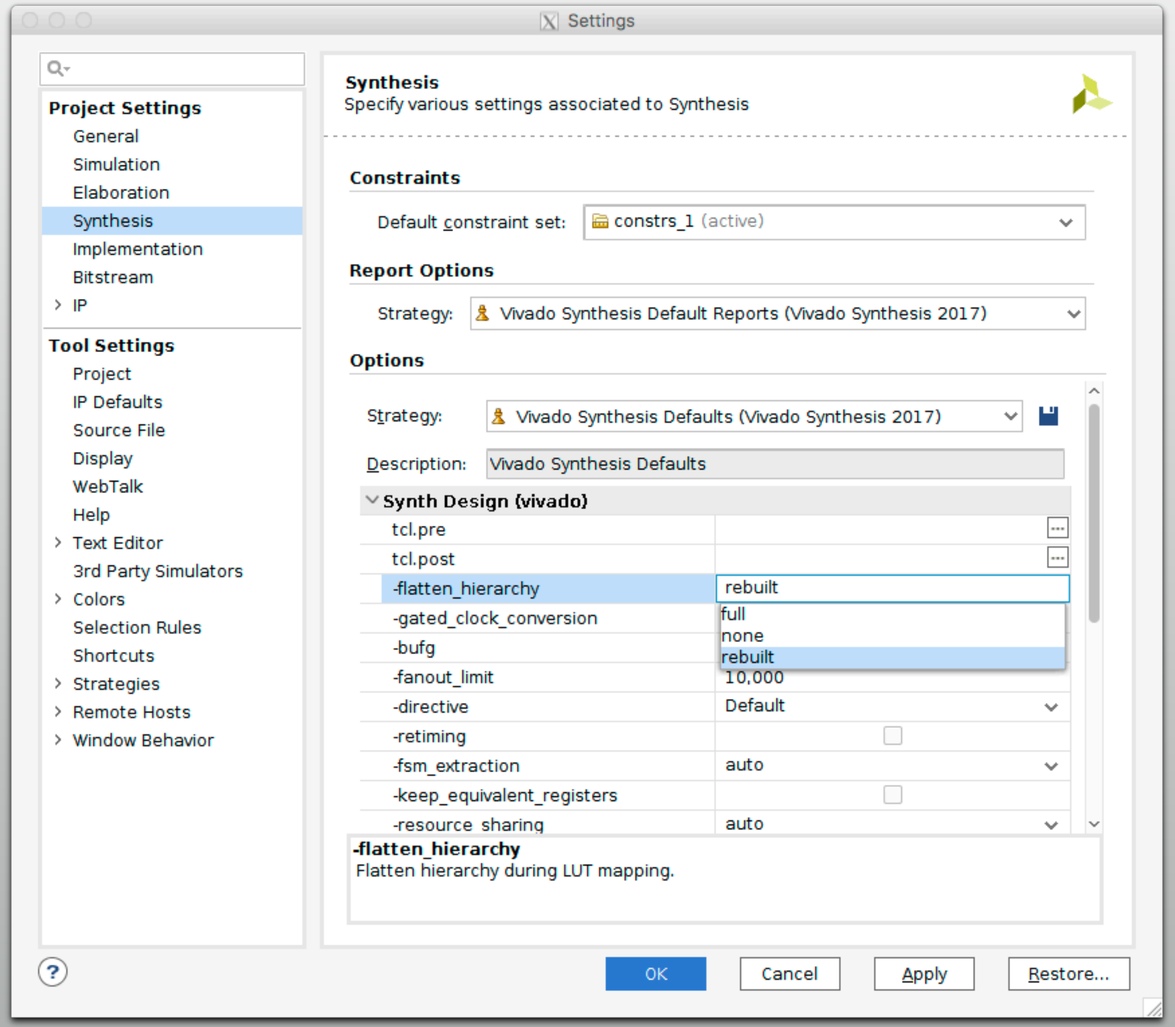
- 論理合成後・配置配線後のネットリストからHDLを生成
  - 遅延なし: 正しく論理合成できているかを確認できる (まあまあ速い)
  - 遅延あり: 配線までの遅延を含めてかなり正確にシミュレーション
- テストベンチはそのまま使える (ちゃんと書いてあれば、ですが)

論理合成とモジュール階層

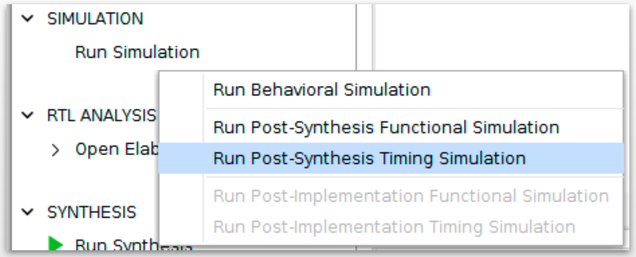
- RTL にはモジュール階層がある
- 論理合成では最適化のためにこれをフラットにする
  - モジュールを超えて回路を最適化したほうがよい結果になるため
  - 合成の結果を HDL にして読む場合にわかりづらく(わからなく)なる

階層の復元

- Project Settings → Synthesis → Flatten Hierarchy
  - Full: 完全にひとつにする
  - None: 階層を完全に維持
  - Rebuilt: Flatten したあとに復元
    - Rebuilt は最適化に影響しない



合成後シミュレーション

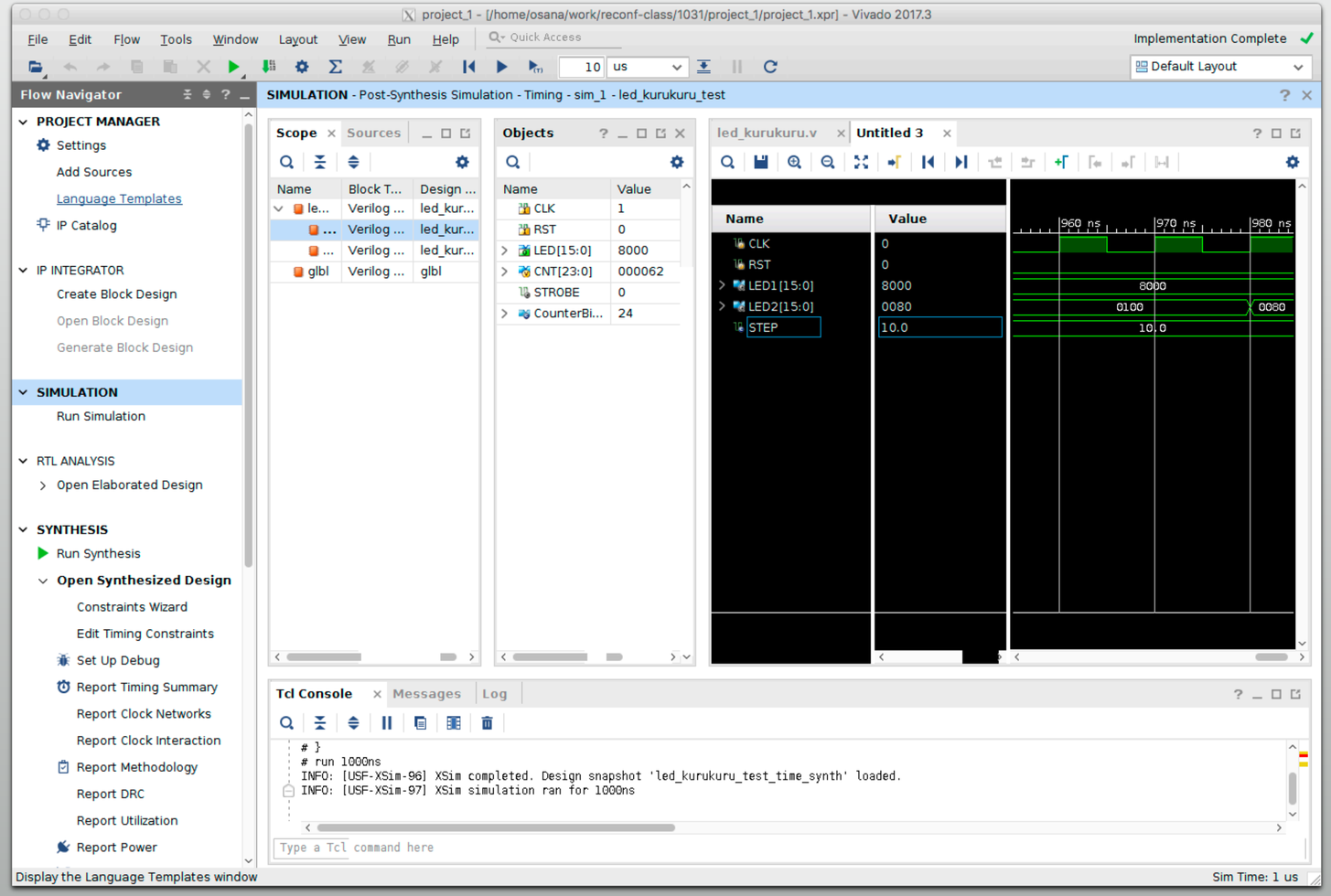


- たとえばこれで試してみる
  - LED くるくる

```
module led_kurukuru
  ( input CLK, RST, output reg [7:0] LED );
   parameter CounterBits = 24;

   reg [(CounterBits-1):0] CNT;
       wire                STROBE = &CNT;

   always @ (posedge CLK) begin
      if (RST) begin
         CNT <= 0;
         LED <= 8'b1000_0000;
      end else begin
         CNT <= CNT+1;
         if (STROBE)
           LED <= {LED[0], LED[7:1]};
      end
   end
endmodule // led_kurukuru
```

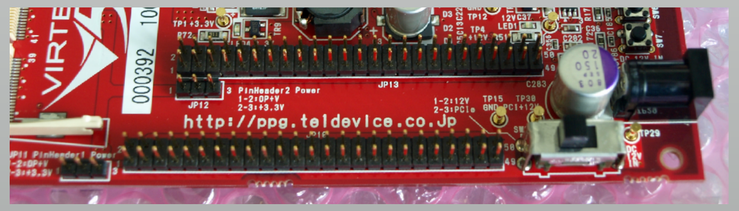
合成後シミュレーション

- ぱっと見、合成前と一緒
  - 信号名が変わったりする場合もある

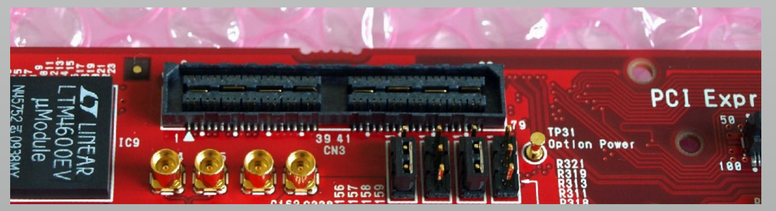


チップの外で波形を観測する

- FPGA から外に出ている信号は計測器で見ることができる
  - 余っている I/O ピンを引き出しておけばデバッグ用に使える
  - 普通の 2.54mm ピンヘッダだけでなく、測定用専用コネクタもある
    - 小さい面積で多数の信号線を接続できる



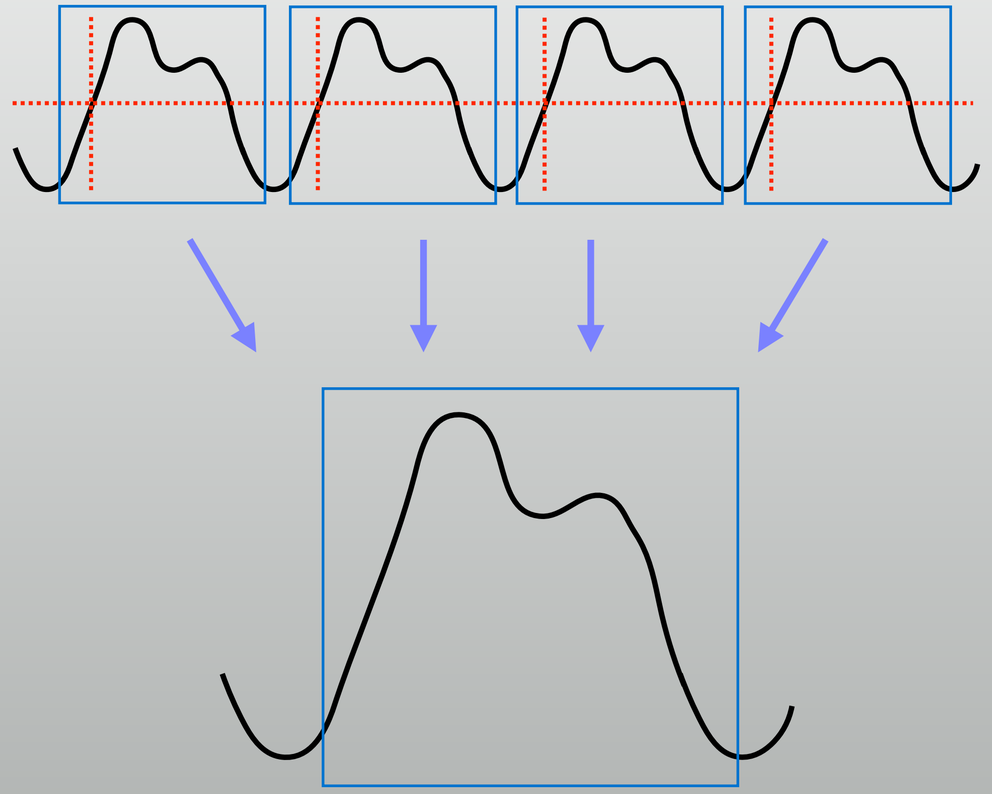
2.54mm 角ピンヘッダ



Mictor コネクタ

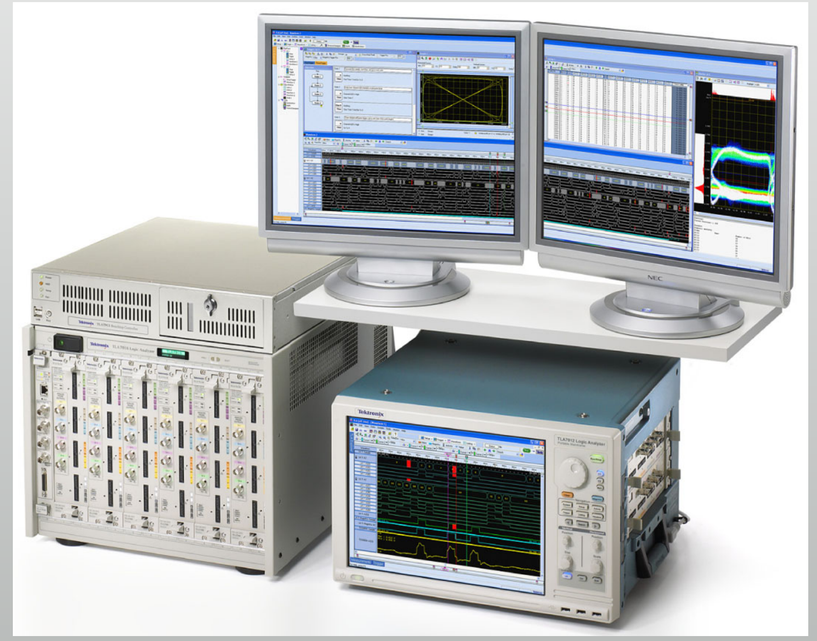
オシロスコープ？

- 基本的に繰返し波形を観測
  - エッジトリガが基本
  - 少ないチャネル数で、連続値を観測
- デジタルでは多チャネルの測定が必要



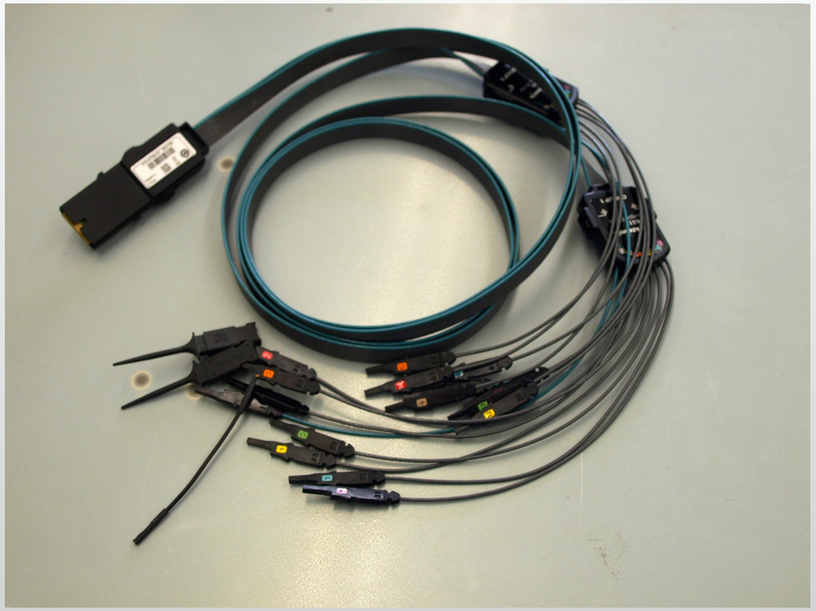
ロジックアナライザ

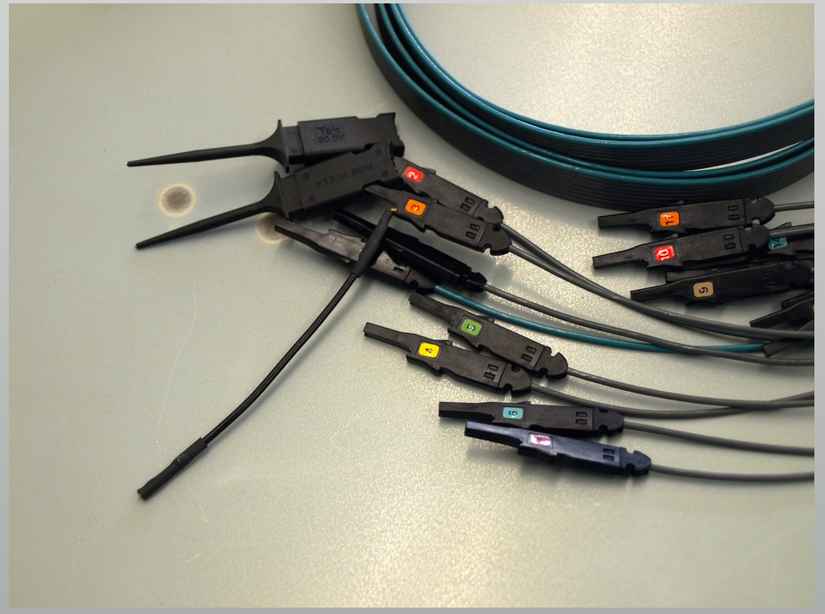
- 多数のチャネルで離散値 (H/L) を観測
  - 事前に設定した条件でトリガ
    - 基本的に1回だけ
  - その前後の波形を観測できる
- 専用機か、ロジアナ付きオシロか、ILA か



ロジアナ用プローブ

- オシロ用プローブよりずっと簡単な構造
  - 各種の IC クリップを接続して使う
- ピンヘッダに直接接続するアダプタや Mictor に直結するプローブなどもある





設計上の注意

- オンチップの信号を assign でそのまま外に出すと遅延が大きく変わる
  - もとの挙動とだいぶ変わってしまう可能性がある
  - 最低一つはレジスタをはさんでから出力しましょう
    - 出力の手前の最後にレジスタがあると IOB に配置される
- クロックを出力する場合には DDR レジスタで生成する (方法はググりましょう)

Boundary Scan

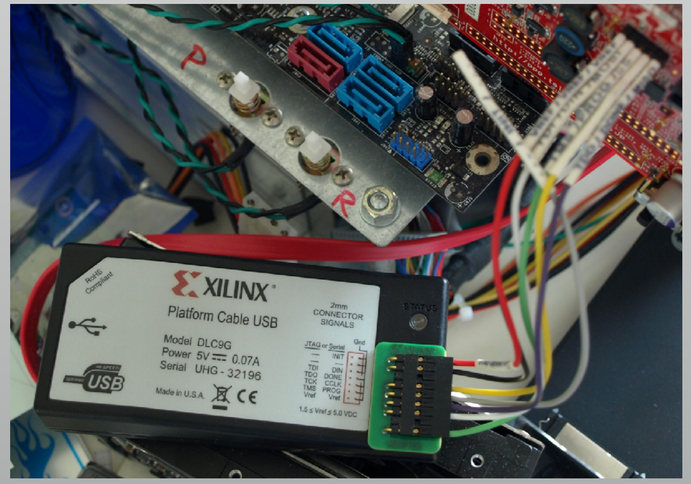
- ボード上の LSI のピンの信号の状態をみたい
  - 全部のピンにプローブをつなぐのは大変 (BGAだとムリ)
  - 少ないコストでピンの状態の観測を可能にしたい
- LSI (やその上のモジュールの) ピンに FF を入れて、数珠つなぎに接続
  - クロックに同期してシフトすることで、1ビットずつ取り出せる

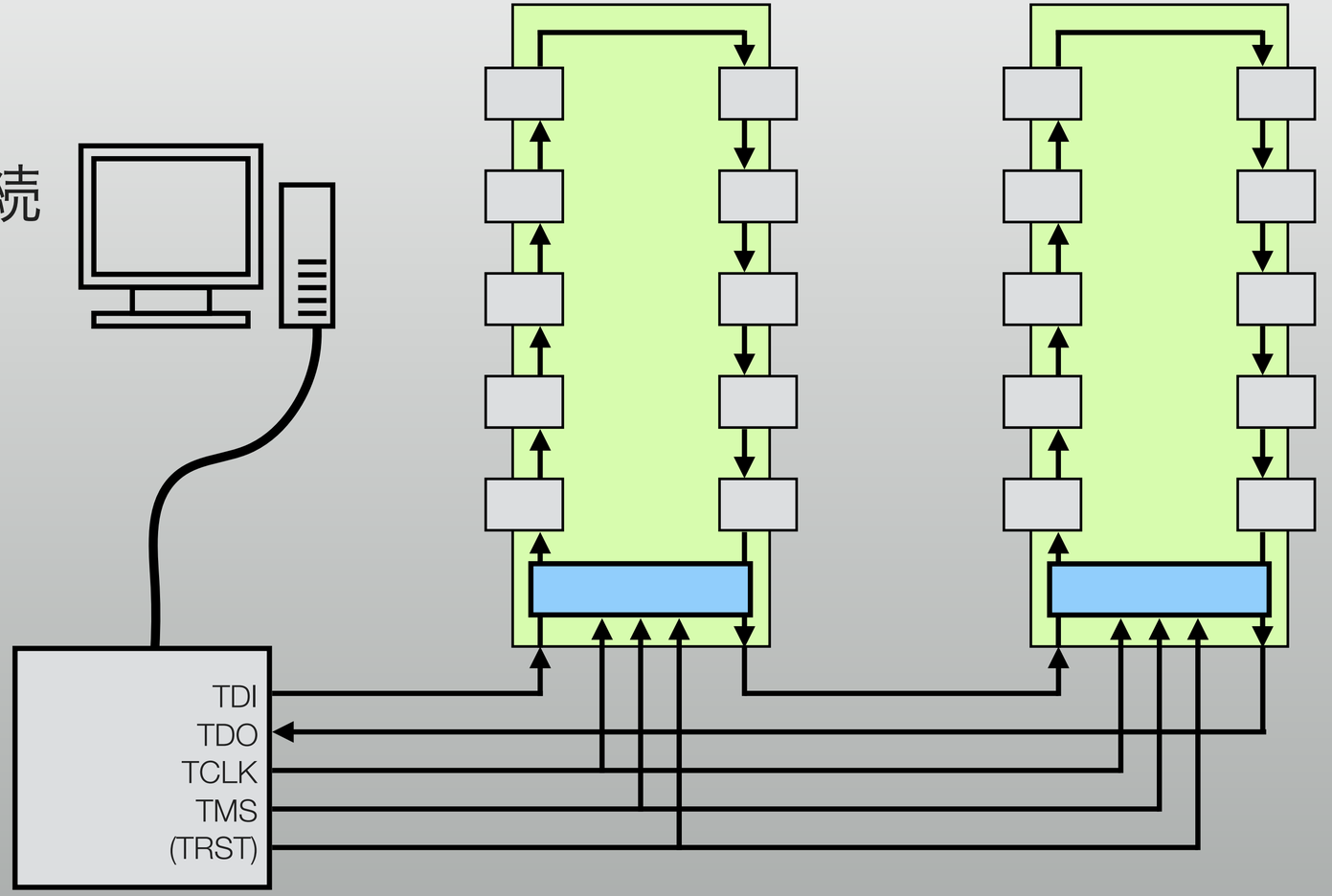
JTAG (Joint Test Action Group)

- IEEE 1149.1-1990 Standard Test Access Port and Boundary-Scan Architecture
  - 4本の信号線で基板上の LSI を数珠つなぎにして接続
  - ピンの状態をモニタしたり、制御することも可能
    - 最近では多くの LSI に JTAG インタフェイスが搭載されている
- JTAG は作業部会の名前だが、規格の名前のように使われている

JTAG インタフェイス

- コントローラはUSBなどで接続





Boundary Scan と FPGA

- FPGA のコンフィギュレーション
  - 全部の LUT や設定用の FF への書き込みが必要
  - 配線を減らすためにシリアルで1ビットずつ書きこむ
    - とはいえ、並列で速いモードもあるのが普通
  - JTAG によるテストインタフェイスが書き込みにも使える！

Internal Logic Analyzer

- FPGA 上に構成するオンチップデバッグツール
  - Trigger state machine を FPGA のロジックで構成
  - 未使用の Block RAM に波形を記録し、JTAG 経由で入出力
- 測定器不要、PC + JTAG ケーブルでデバッグができる
  - 実習はのちほど

直接測定では難しいことがいっぱい

- 測定器をつなぐにも ILA をつなぐにも回路の変更が必要
  - ただし、ILA の場合は再配置配線はいらない方法もある
- 測定できる時間は限られており、長い時間の波形がとれるわけではない
  - トリガ条件を工夫してもタイムスケールが長いとムリ
  - 特定の条件になったら増えるカウンタとかを作る

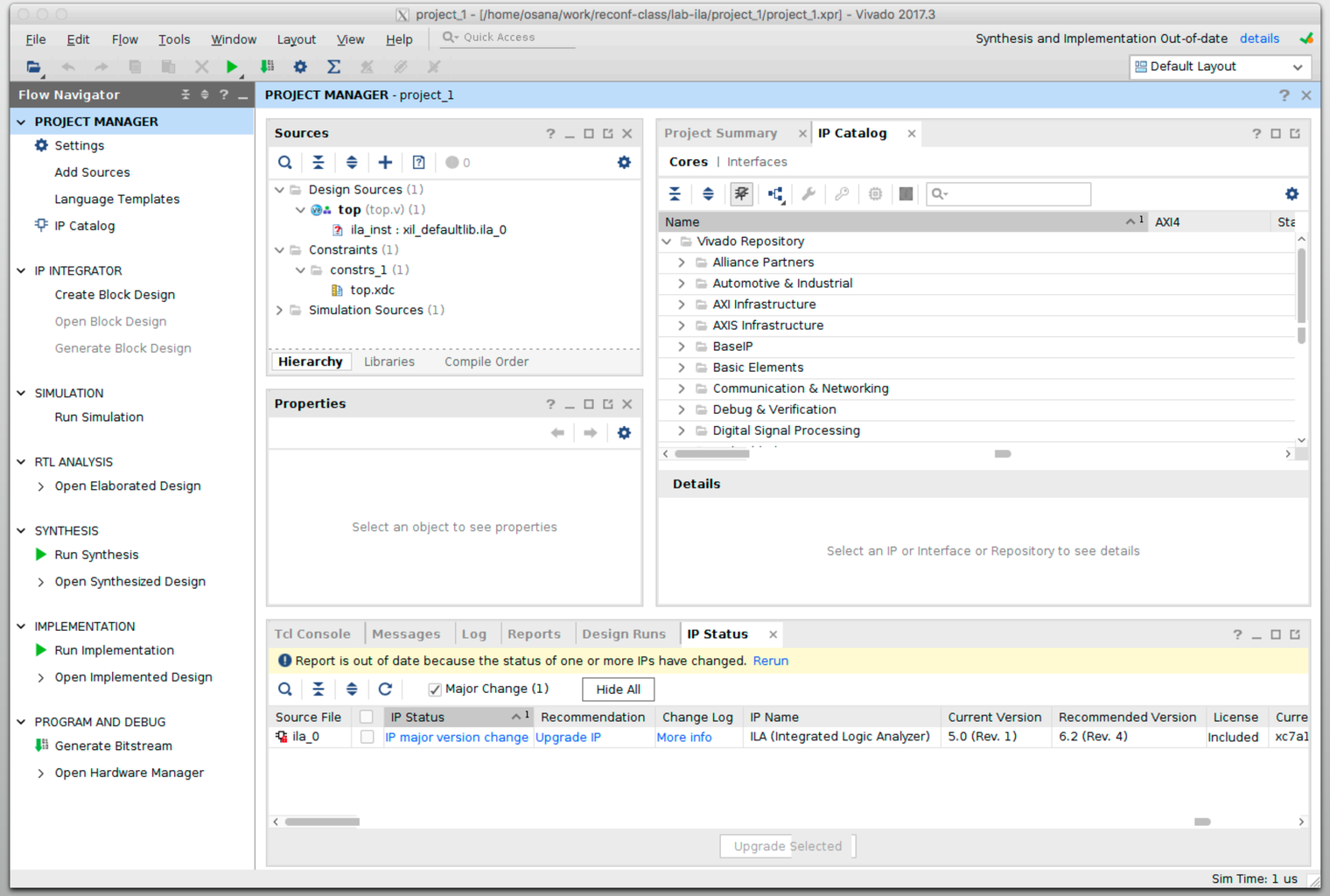
実習

- ボタンとカウンタだけの回路
  - ILA で動作をのぞいてみる

```
`timescale 1ns / 1ps

module top( input CLK, input RST, input BTN,
            output reg LED );

  reg [7:0] CNT;

  always @ (posedge CLK) begin
    LED <= BTN;
    if (RST)
      CNT <= 0;
    else
      CNT <= CNT+1;
  end

  ila_0 ila_inst
    (.clk(CLK), .probe0(CNT), .probe1(BTN));

endmodule
```

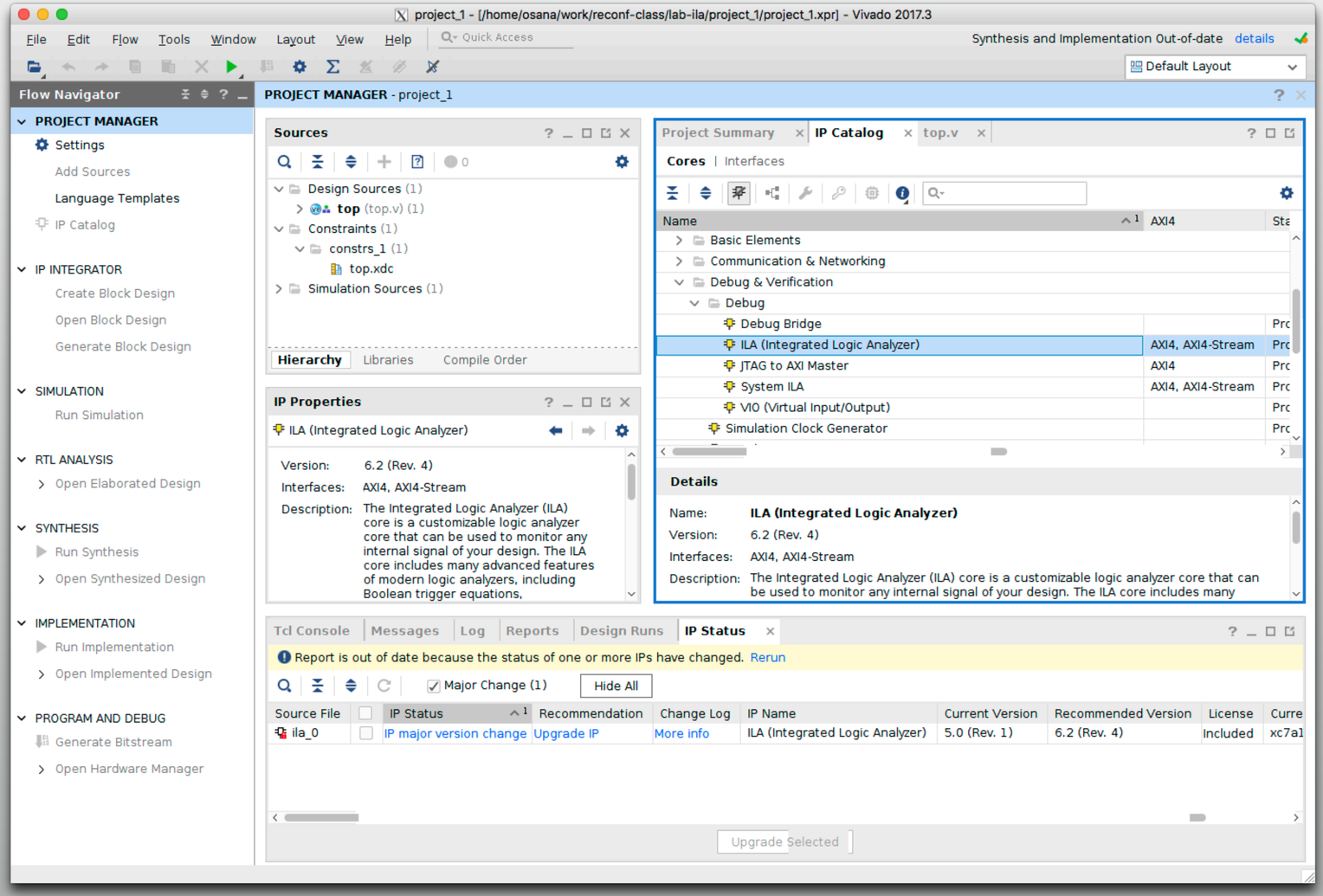
ILA 生成 (1)

- IP Catalog を開く



ILA 生成 (2)

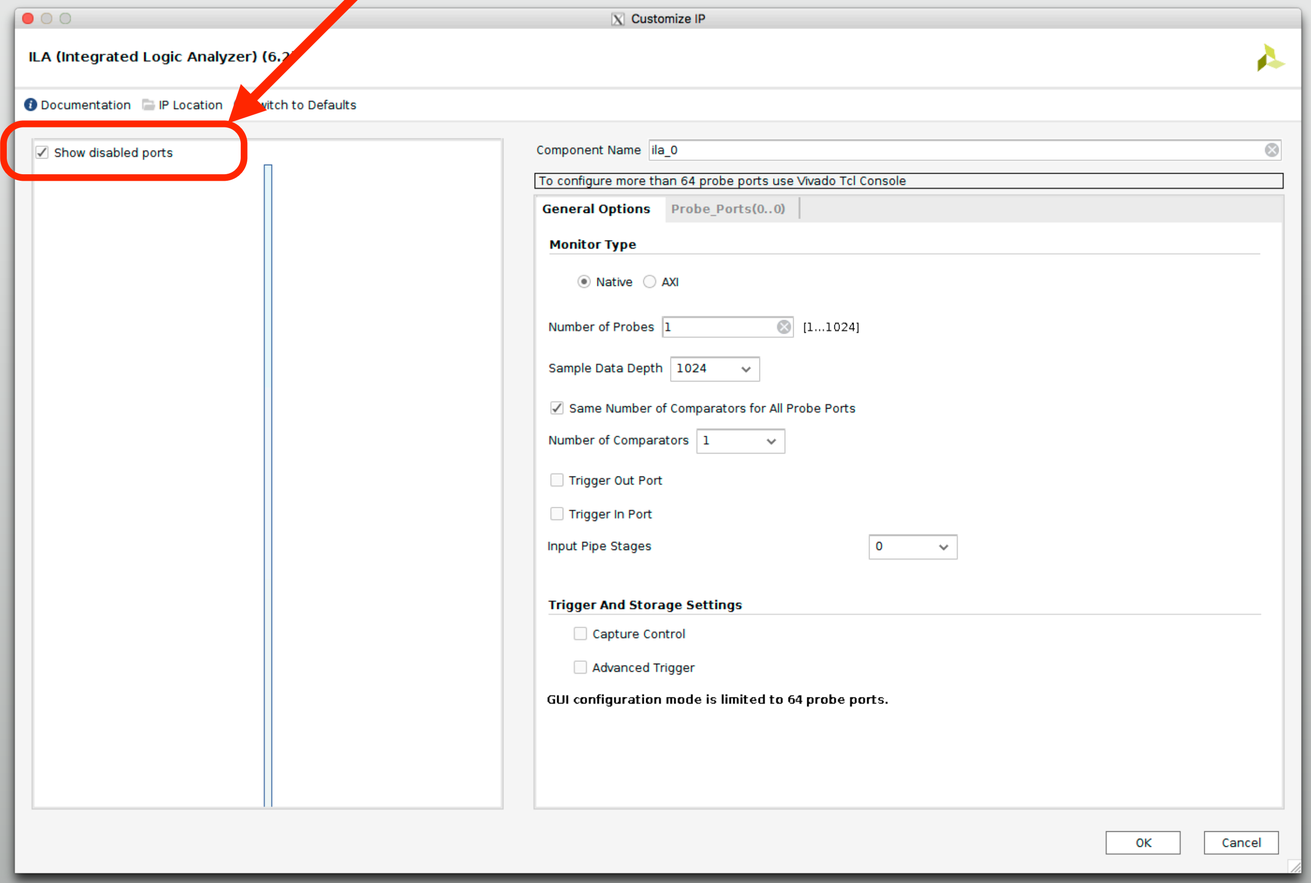
* Debug & Verification
  → Debug → ILA



ILA 設定 (1)

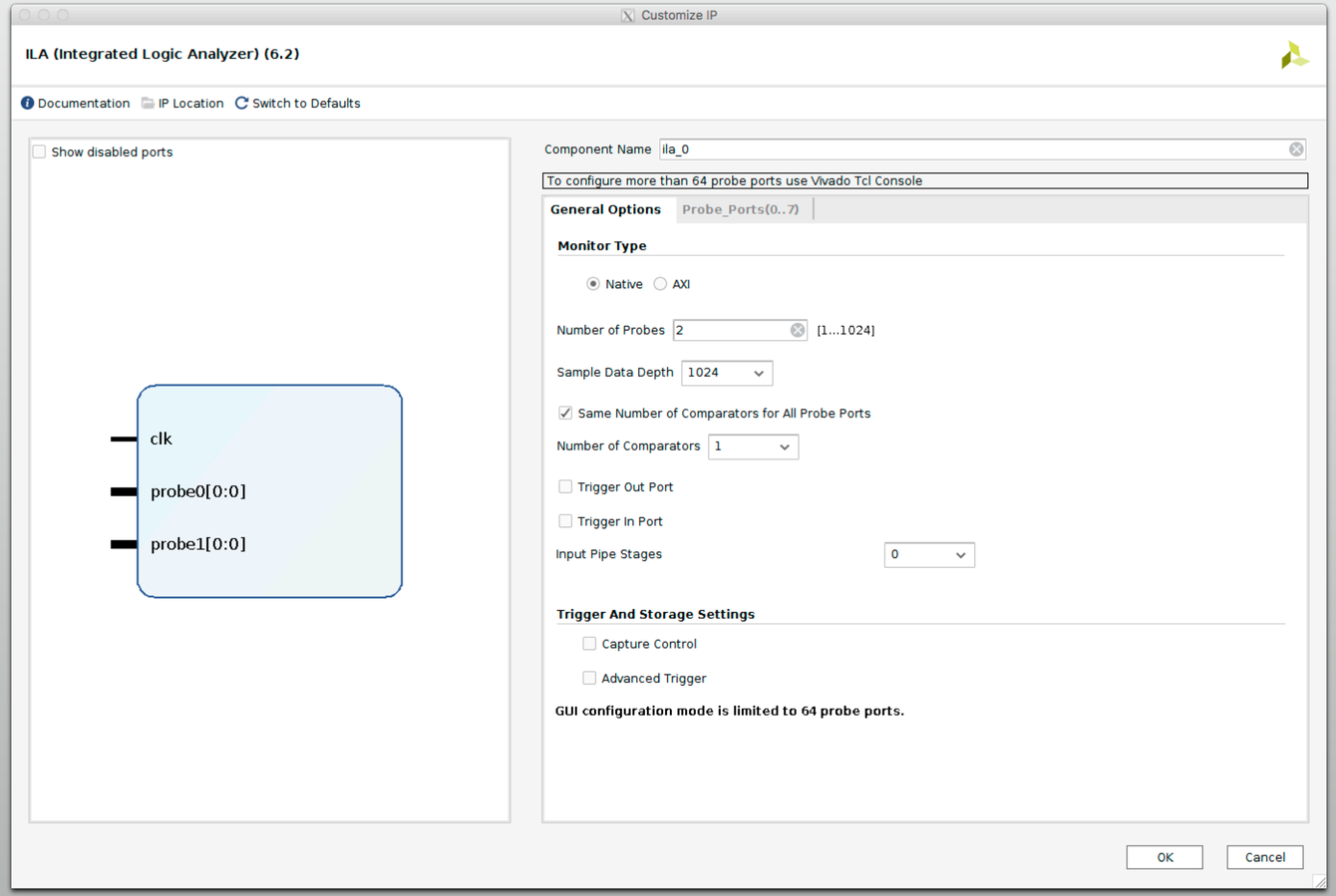
- プローブの設定
  - 本数
  - 各プローブのビット幅
- 記録する波形の長さ

見づらいのでとりあえずこれは off



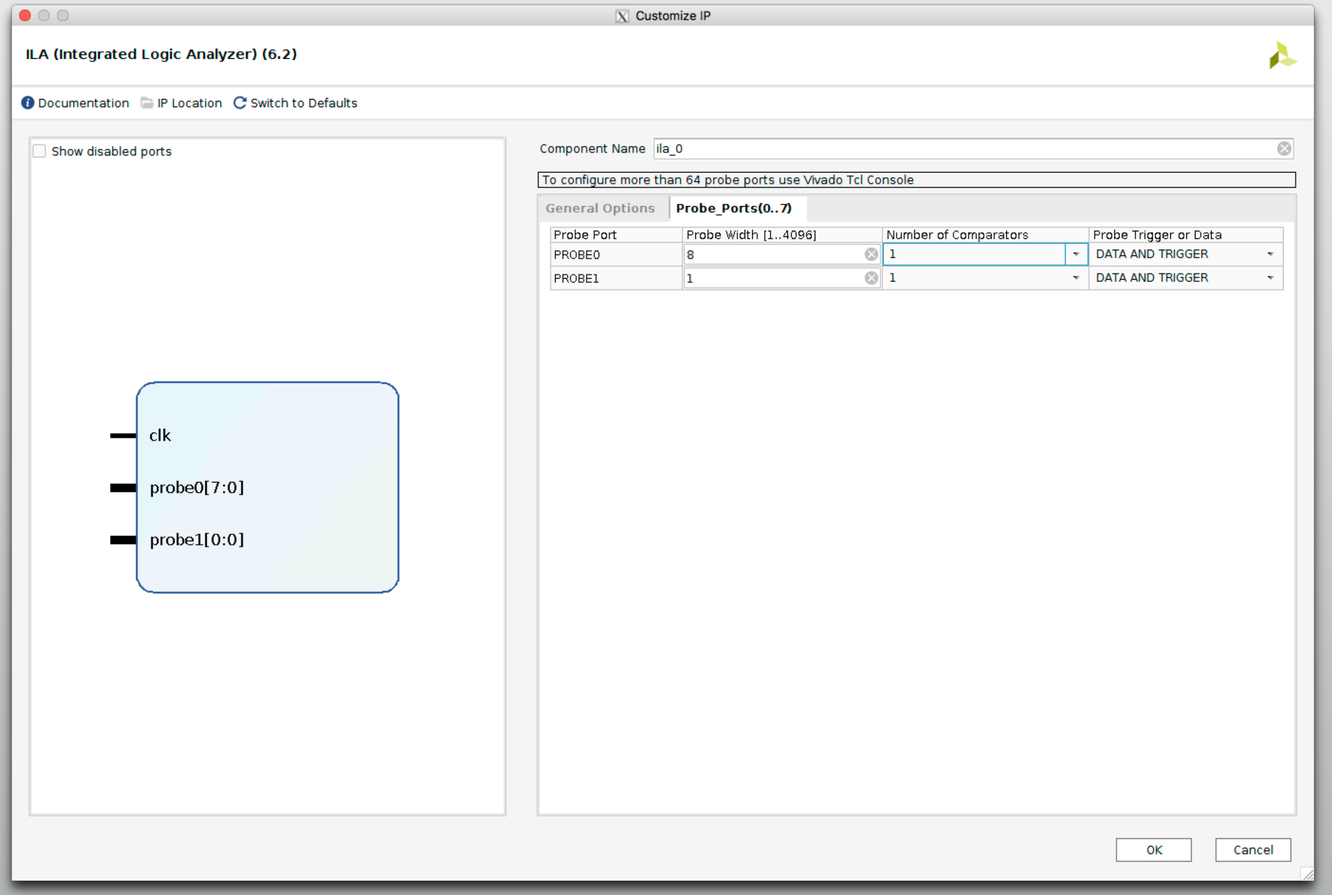
ILA 設定 (2)

- プローブは2本
  - 深さは 1024 のままで



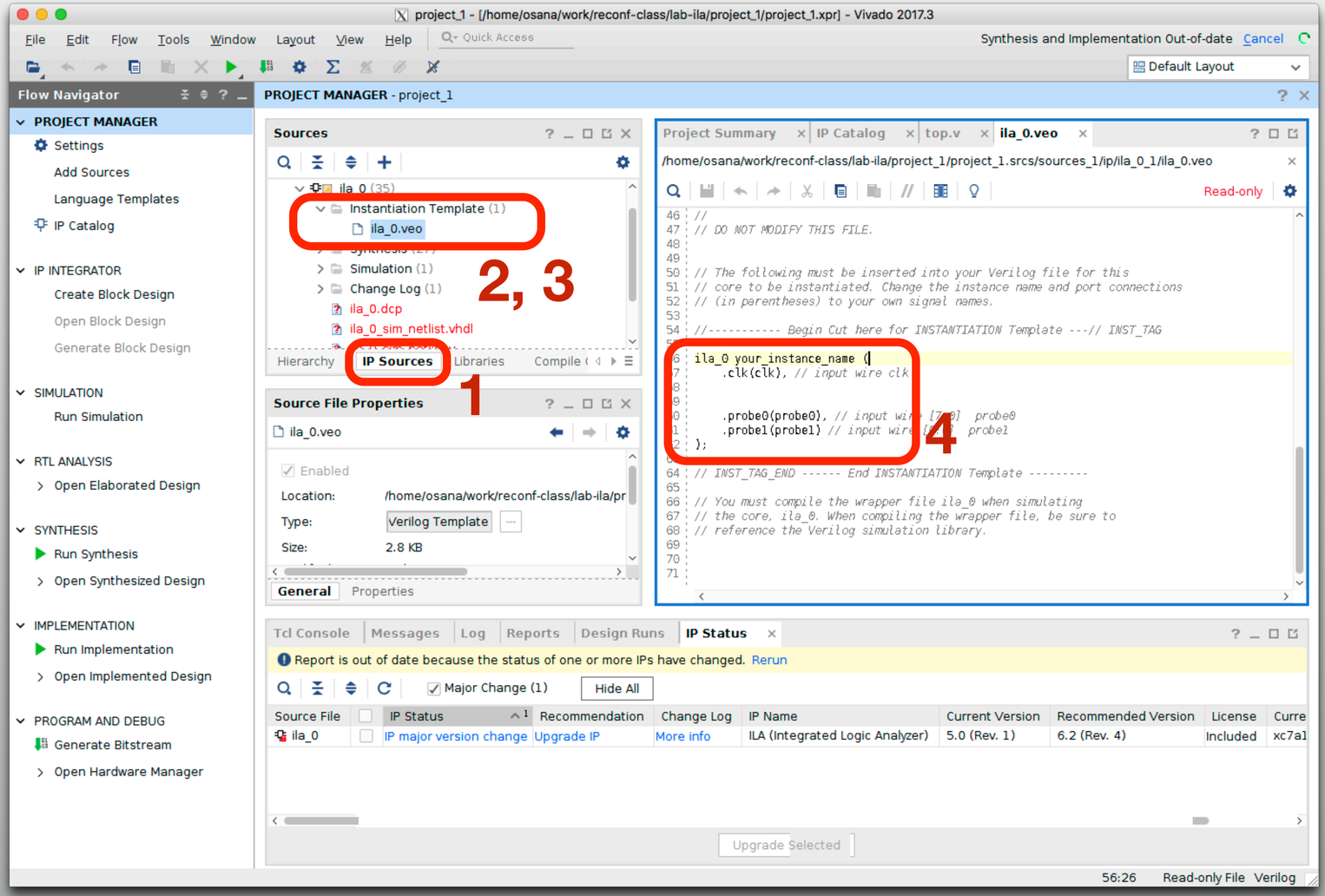
ILA 設定 (3)

- 各プローブの設定は Probe ports
  - Probe0 を8ビットに
  - OK を押して生成



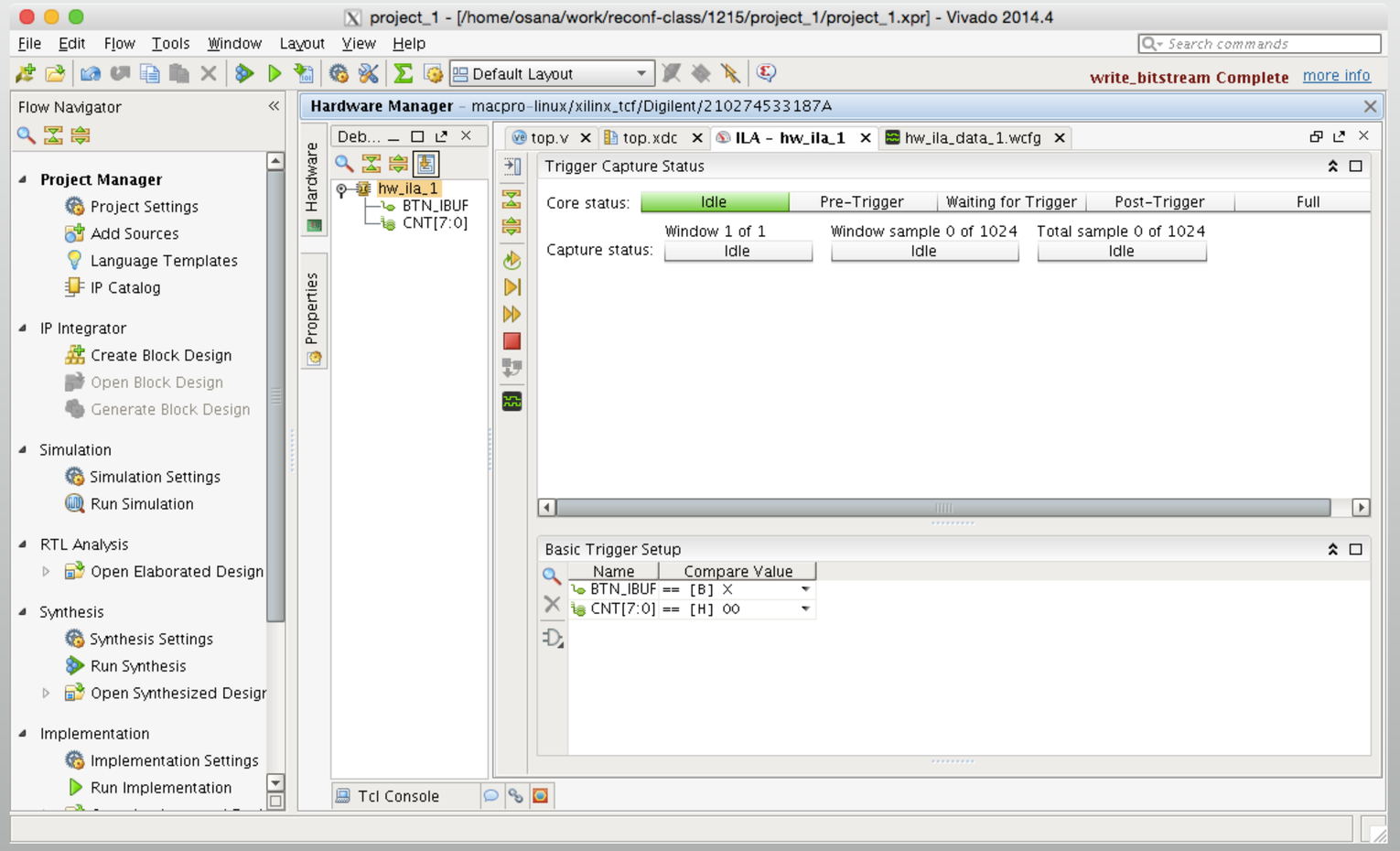
インタフェイスを確認

- Sources → IP Sources
- コアを開いて Instantiation template を確認



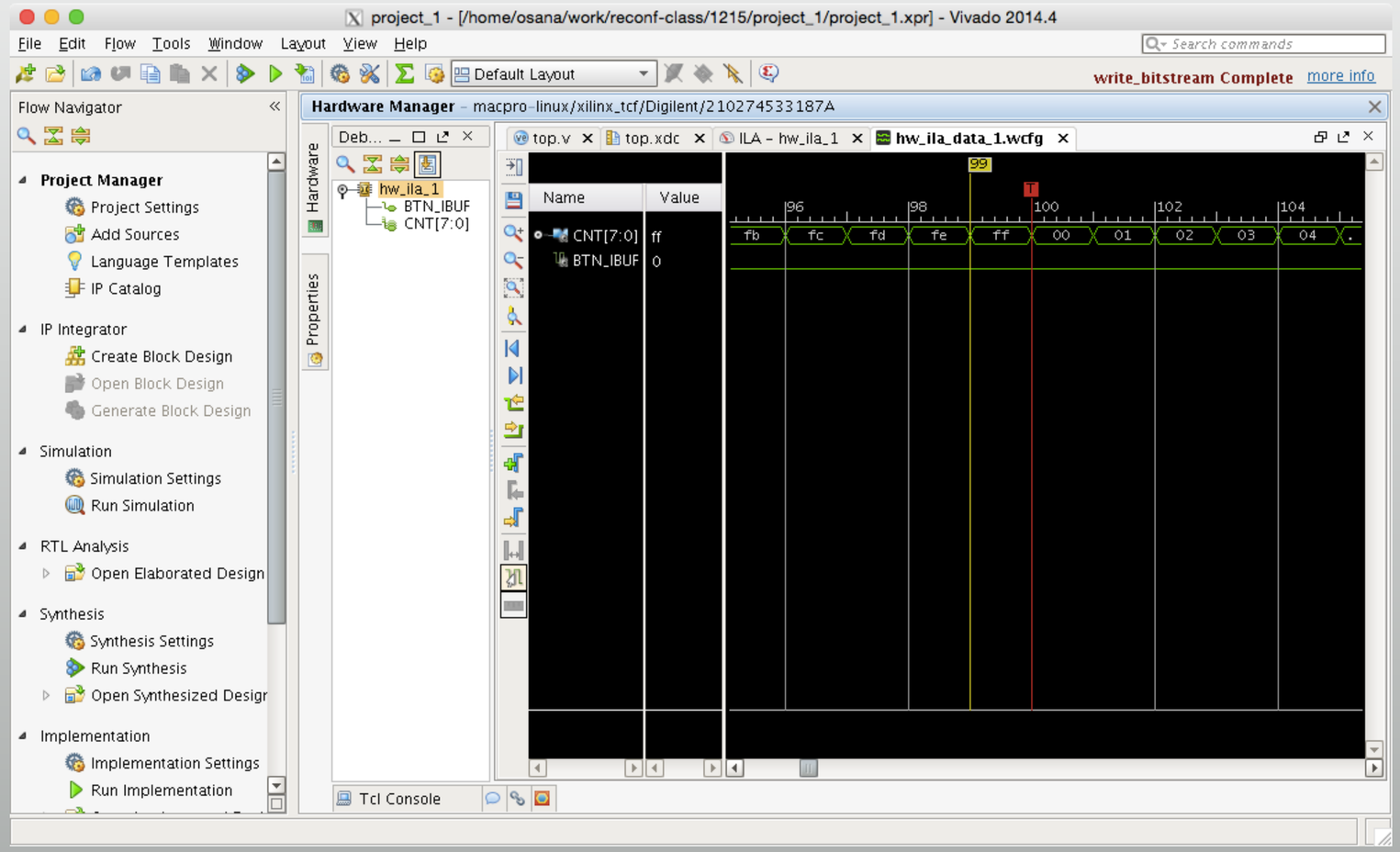
Hardware Manager

- ILA は自動で認識
  - トリガ条件を設定
  - Position も大事
- トリガ待ち 
- 強制トリガ 



波形観測

* CNT = 0 で、
  Position = 100



やること

- RTL を書く: スライド39枚目、ILA のコアも作りましょう
  - ILA のコアいれないと論理合成で CNT が消される (外に出ないので)
  - output [7:0] LED_O を追加して、assign LED_O = CNT; する **(これいらないかも)**
- テストベンチを書く: CLK, RST, BTN だけ書けば OK (クロックが動いていればあとの挙動は自由)
- RTL でシミュレーションできたら、論理合成して、Post-synthesis timing simulation とかしてみる
  - RTL と違って、クロックの立ち上がりからカウンタの値が変化するまでに遅延がはいることを確認
- XDC ファイルを作って、入出力のポートのピン位置を決める
  - スイッチとLEDのピン番号はボード参照、クロックのピンは E3
- 配置配線後、ピン位置が正しいことをもう一度確認したら FPGA に書き込み、ILA が PC から操作できることを確認する
  - Trigger Setup で BTN_IBUF = 1 とかすると、ボタンを押したら反応する
  - Trigger position in Window を 100 とか 200 にすると、ボタンが押される前の 100, 200 クロックの様子をみることができる