Reconfigurable Architecture (10)

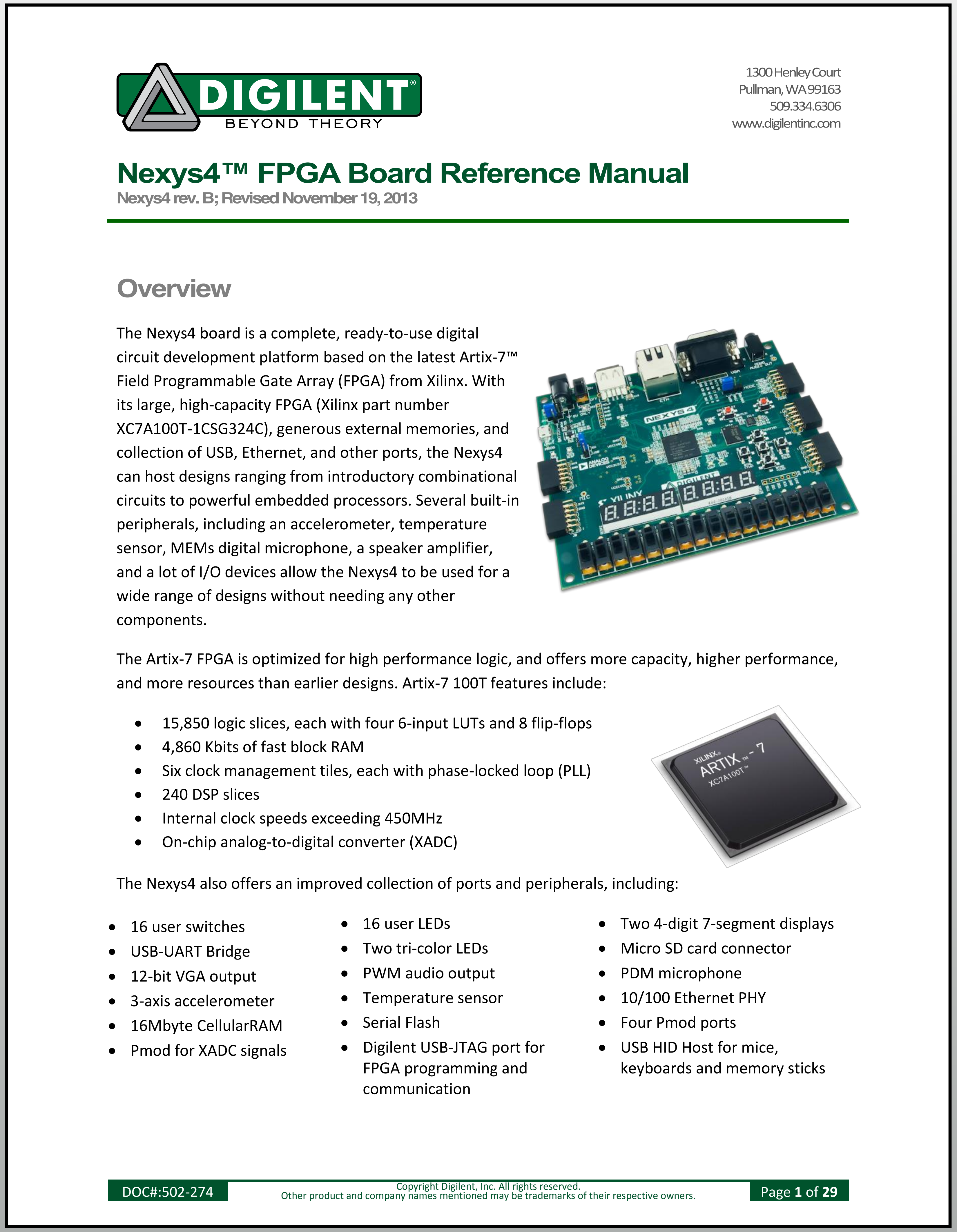
Nexys4 peripherals & Using the Internal logic analyzer

Agenda of this week

* Working with Nexys4 peripheral devices
    * PWM to illuminate LEDs
    * I^2C (Inter-Integrated Circuit) to temperature sensor
    * SPI (Serial Peripheral Interface) to accelerometer
* On-chip debug on a working FPGA

Documents (1): Board Ref. Manual

* digilentinc.com → Products → FPGA Boards → Nexys4
    * First of all, see the reference manual
    * Short descriptions about on-board devices



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

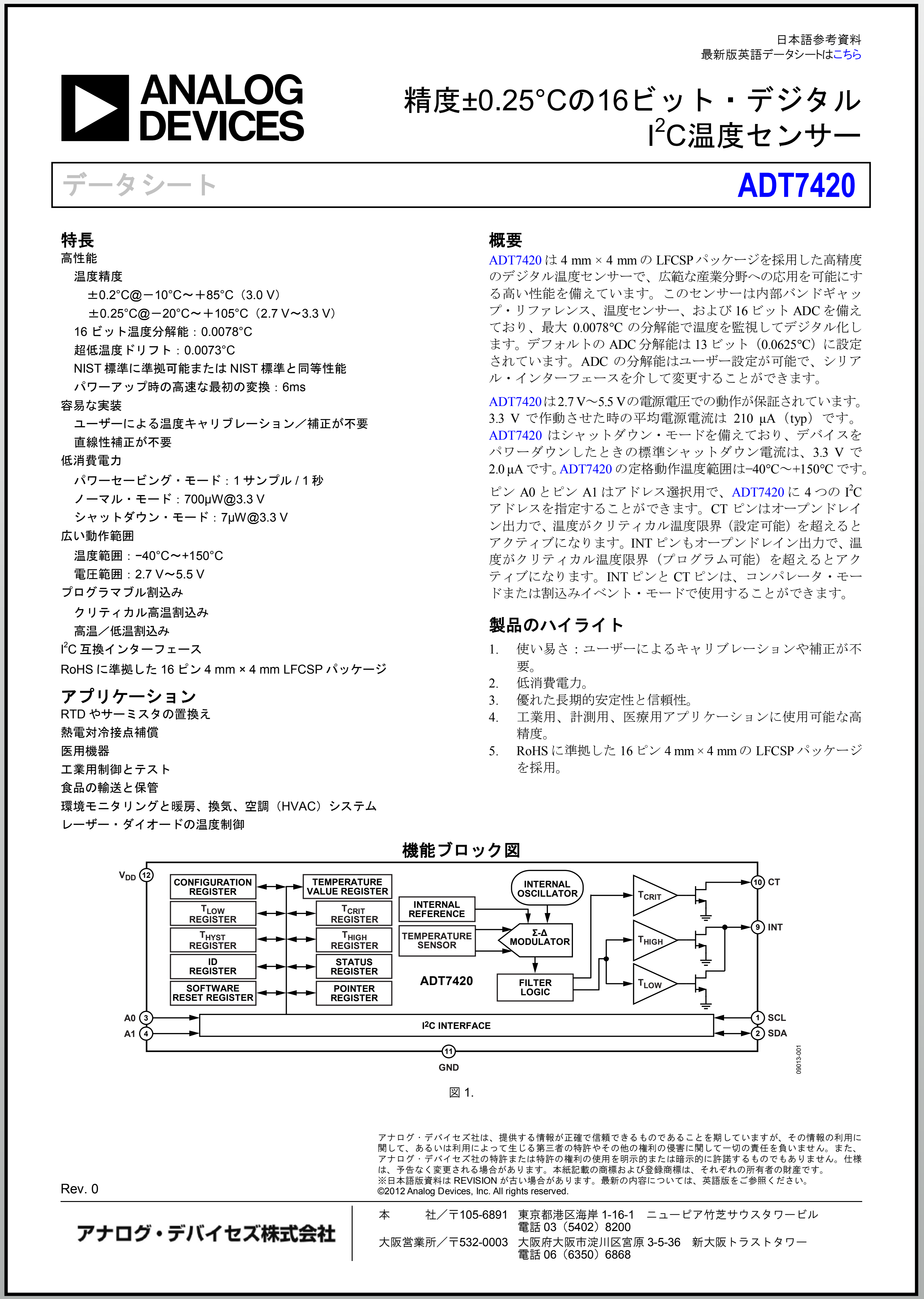
DOC#:502-274 Copyright Digilent, Inc. All rights reserved. Other product and company names mentioned may be trademarks of their respective owners. Page 1 of 29

Documents (2): Master XDC

* Available from the same webpage with the reference manual
* All FPGA pins' signal name + IOSTANDARD
  * Signals named by Digilent, along their role
  * Of course you can change the names
* Good to avoid misconfigurations from typo

Documents (3): Device datasheets

* Device part numbers on the reference manuals
    * Details on the Devices' data sheets
    * Available from the vendor websites
        * Japanese translation available (sometimes)



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
I^2C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

I^2C 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 I^2C 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

V_{DD} 12
CONFIGURATION REGISTER
TEMPERATURE VALUE REGISTER
T_{LOW} REGISTER
T_{CRIT} REGISTER
T_{HYST} REGISTER
T_{HIGH} REGISTER
ID REGISTER
STATUS REGISTER
SOFTWARE RESET REGISTER
POINTER REGISTER
INTERNAL REFERENCE
TEMPERATURE SENSOR
INTERNAL OSCILLATOR
Σ-Δ MODULATOR
FILTER LOGIC
ADT7420
T_{CRIT}
T_{HIGH}
T_{LOW}
10 CT
9 INT
A0 3
A1 4
I^2C INTERFACE
1 SCL
2 SDA
11 GND
09013-001

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 Peripherals

* Switches, 7 segment LEDs, Discrete LEDs, Tri-Color LEDs
* Temperature sensor, 3-axis accelerometer, PWM audio output, PDM mic
* 128Mb Pseudo-SRAM (CellularRAM: Verilog model available)
* VGA (display), USB-PS/2 (Keyboard & Mouse)
* USB-UART (Serial), 10/100M Ethernet (LAN)

Simple on-off control

* **Switches**, 7 segment LEDs, **Discrete LEDs**, Tri-Color LEDs
* Temperature sensor, 3-axis accelerometer, PWM audio output, PDM mic
* 128Mb Pseudo-SRAM (CellularRAM: Verilog model available)
* VGA (display), USB-PS/2 (Keyboard & Mouse)
* USB-UART (Serial), 10/100M Ethernet (LAN)

Dynamic drive and PWM

* Switches, **7 segment LEDs**, Discrete LEDs, **Tri-Color LEDs**
* Temperature sensor, 3-axis accelerometer, PWM audio output, PDM mic
* 128Mb Pseudo-SRAM (CellularRAM: Verilog model available)
* VGA (display), USB-PS/2 (Keyboard & Mouse)
* USB-UART (Serial), 10/100M Ethernet (LAN)

Standard serial interfaces (SPI/I2C)

* Switches, 7 segment LEDs, Discrete LEDs, Tri-Color LEDs
* **Temperature sensor**, **3-axis accelerometer**, PWM audio output, PDM mic
* 128Mb Pseudo-SRAM (CellularRAM: Verilog model available)
* VGA (display), USB-PS/2 (Keyboard & Mouse)
* USB-UART (Serial), 10/100M Ethernet (LAN)

PC standard serial interface

* Switches, 7 segment LEDs, Discrete LEDs, Tri-Color LEDs
* Temperature sensor, 3-axis accelerometer, PWM audio output, PDM mic
* 128Mb Pseudo-SRAM (CellularRAM: Verilog model available)
* VGA (display), **USB-PS/2 (Keyboard & Mouse)**
* **USB-UART (Serial)**, 10/100M Ethernet (LAN)

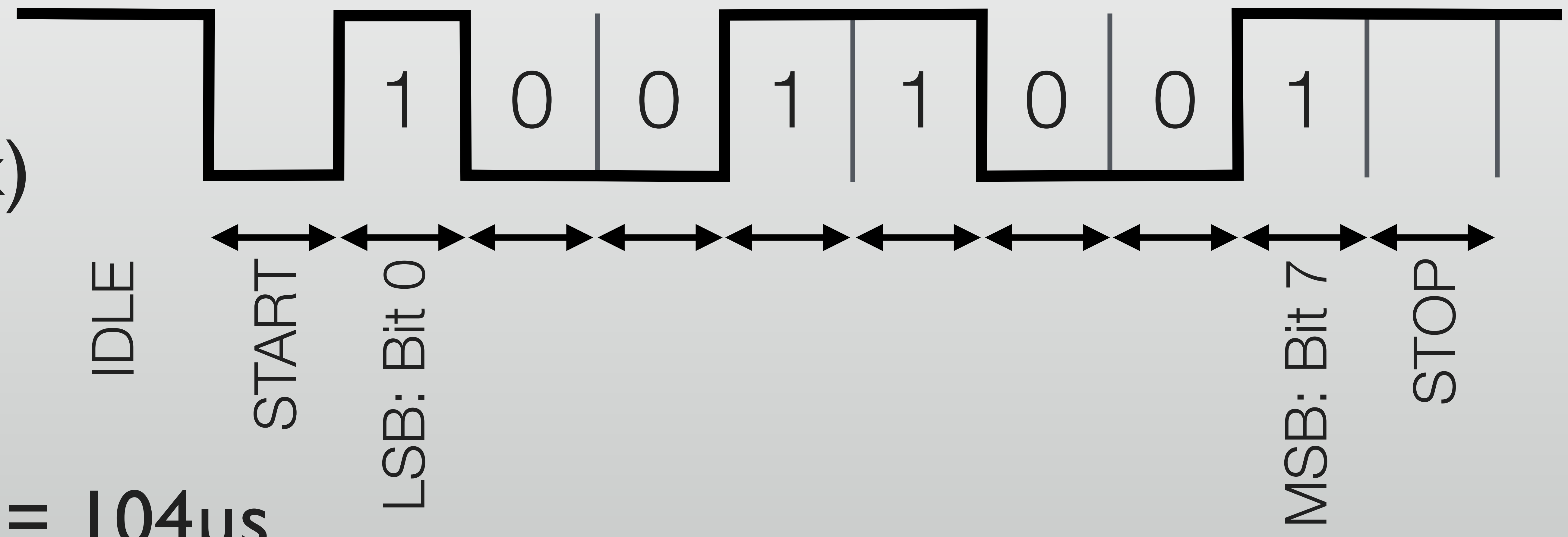
Easy-to-make interfaces

* Switches, 7 segment LEDs, Discrete LEDs, Tri-Color LEDs
* Temperature sensor, 3-axis accelerometer, **PWM audio output, PDM mic**
* 128Mb Pseudo-SRAM (CellularRAM: Verilog model available)
* **VGA (display)**, USB-PS/2 (Keyboard & Mouse)
* USB-UART (Serial), 10/100M Ethernet (LAN)

Serial interface

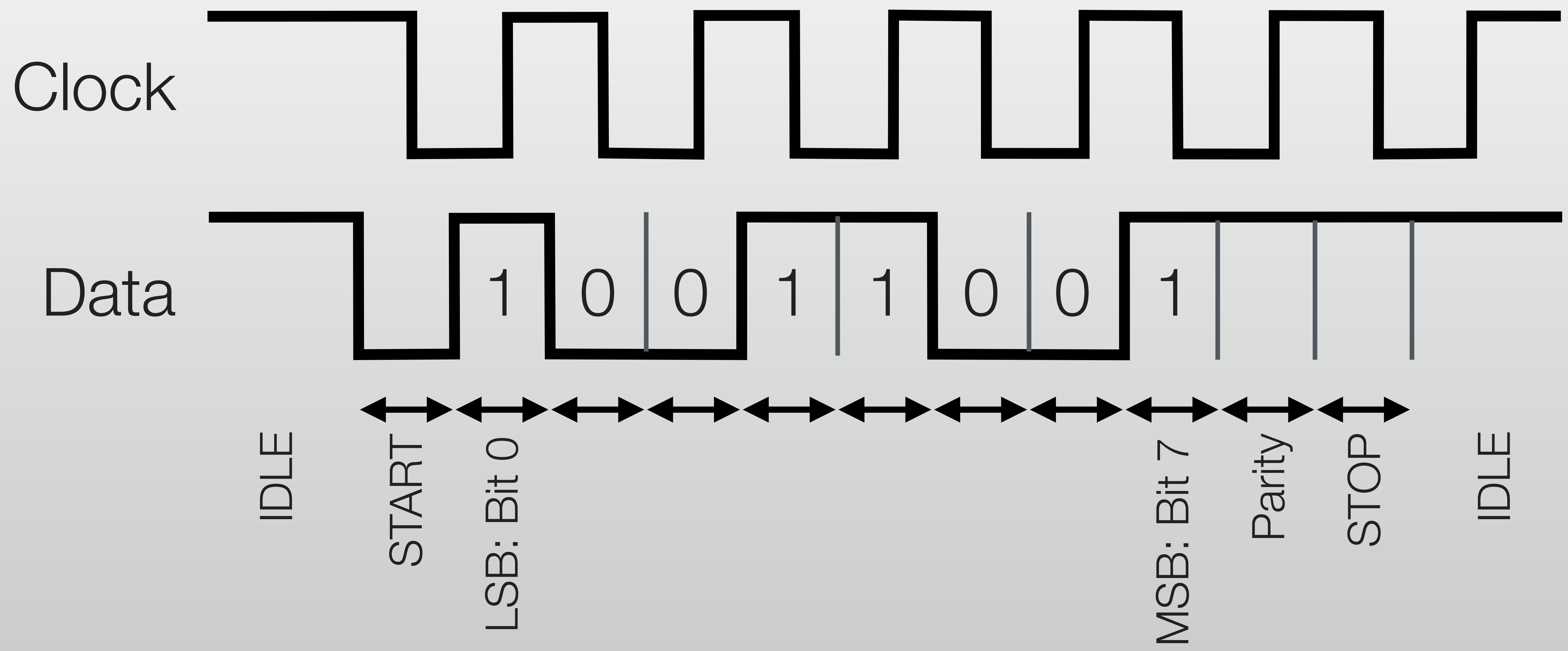
- Send or receive data bits on single wire
  - With or without separate clock signal line
  - With or without separate send/receive line

RS-232C



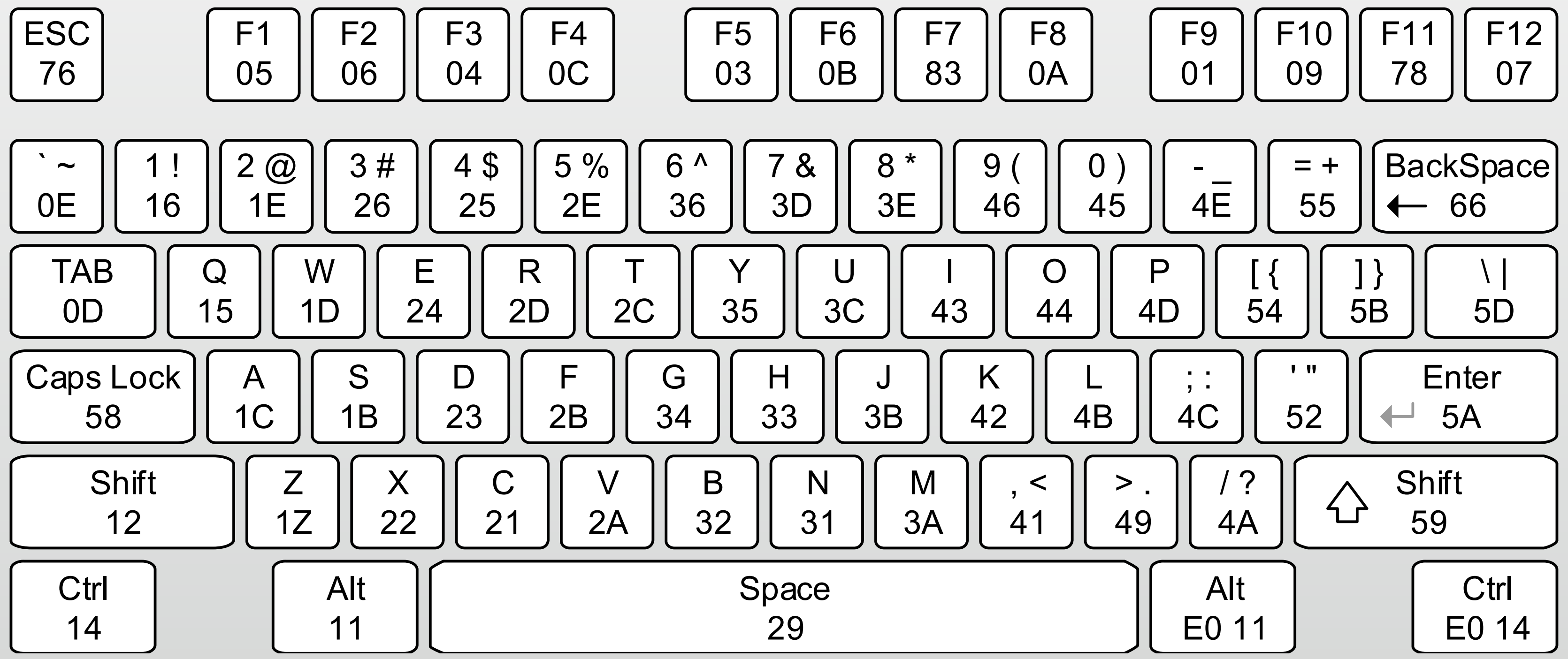
* 2 wires to send (Tx) and receive (Rx)
* Pre-defined period for 1 bit
  * ex) for 9600bps, 1bit = 1/9600sec = 104μs
* High on idle
  * Start on a "0", sample timing of data bits are relative to this transition to low

PS/2



- 2 wires for clock and data
  - Clock remains high while idle
  - Data signal is bidirectional
    - Just ignore parity for receive-only applications
- Clock period is as long as 60-100μs

PS/2 Keyboard, Mouse



- Each key has its own scan code
  - On press down, scan code is transmitted
  - On release, F0 (Key-up) + scan code are transmitted
- Mice emits multiple bytes on move and button press/release

I²C and SPI: Low-speed peripheral I/O

- I2C: 2 signal lines: SCL (Clock), SDA (Data + Address)
  - Address to select device, up to about 400kbps
- SPI: 4 signal lines: SCK (Clock), MISO+MOSI (Data), SS (Slave Select)
  - An SS line corresponds to a device, faster than I2C as 5Mbps
  - Parallel version with multiple MISO/MOSI signals (Dual SPI / Quad SPI)

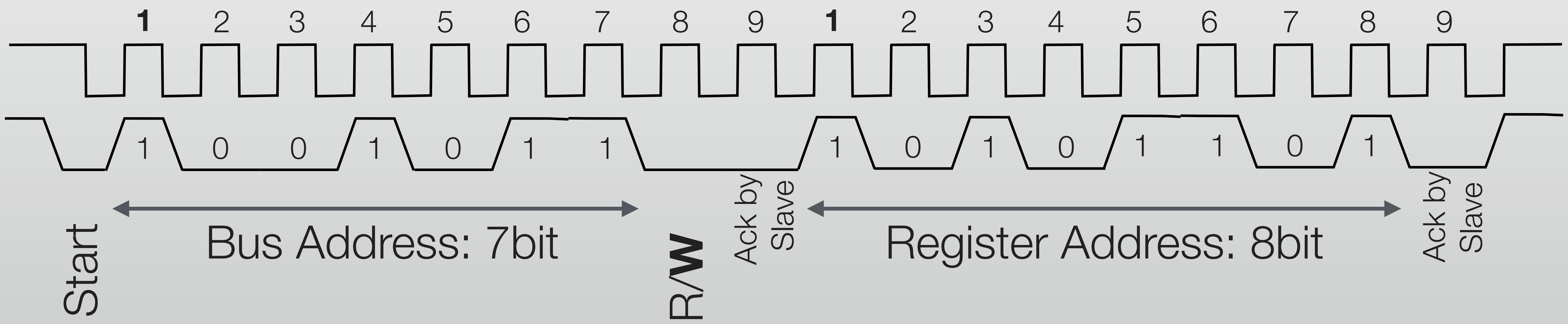
I^2C and SPI

* Each device has multiple devices
  * Registers have their own address and specific features
  * Write sensor settings, or read sensor values
* In I^2C, device has address to select; use SS signal instead in SPI

I²C Protocol (Read)

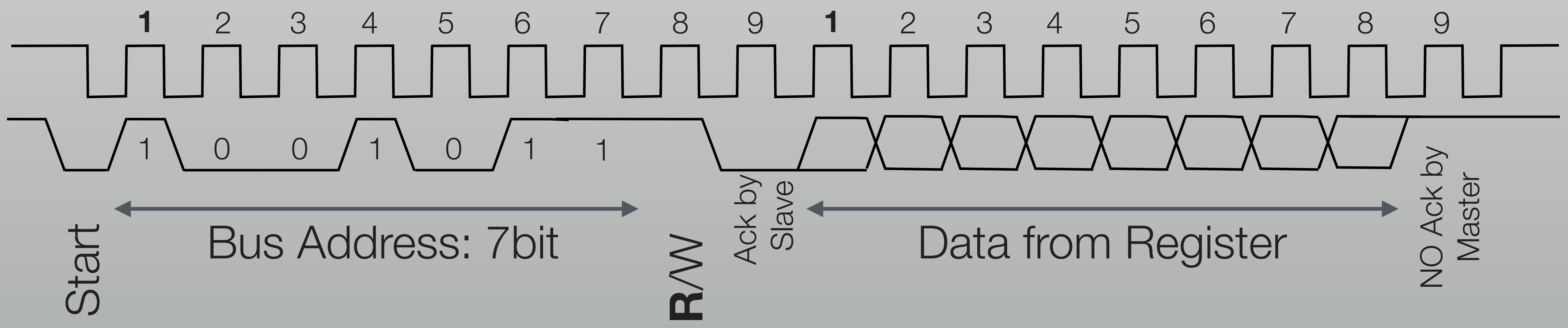
Step 1:

Write register address



Step 2:

Read data



I^2C Protocol

* Bus (device) address must be given every time
* To write registers, give data after Step 1
* To read the same register again, repeat Step 2

I^2C example: ADT7420

* 16bit temperature sensor
  * Positive celcius: (16bit value) / 128
  * Negative celcius: (16bit value - 65536) / 128
* Default register address is the temperature register
  * No need to set the register address (Step 2 is sufficient)
  * The register is 16bit, so receive twice after transmitting bus address

SPI protocol

- 4 wires
  - SS (CS_), SCLK, MOSI (from FPGA), MISO (to FPGA)
  - Continuous CS_ and SCLK without Ack will be continuous access
- SPI example: ADXL362
  - Please refer to the datasheet …

Final Assignment

* Make something fun with Nexys4 peripherals
* Following sample code is available on the web, see the top.v for instruction
    * Controllers for PS/2, Temperature sensor, Accelerometer and Microphone
    * PWM controller for LED

Post synthesis/implement simulation

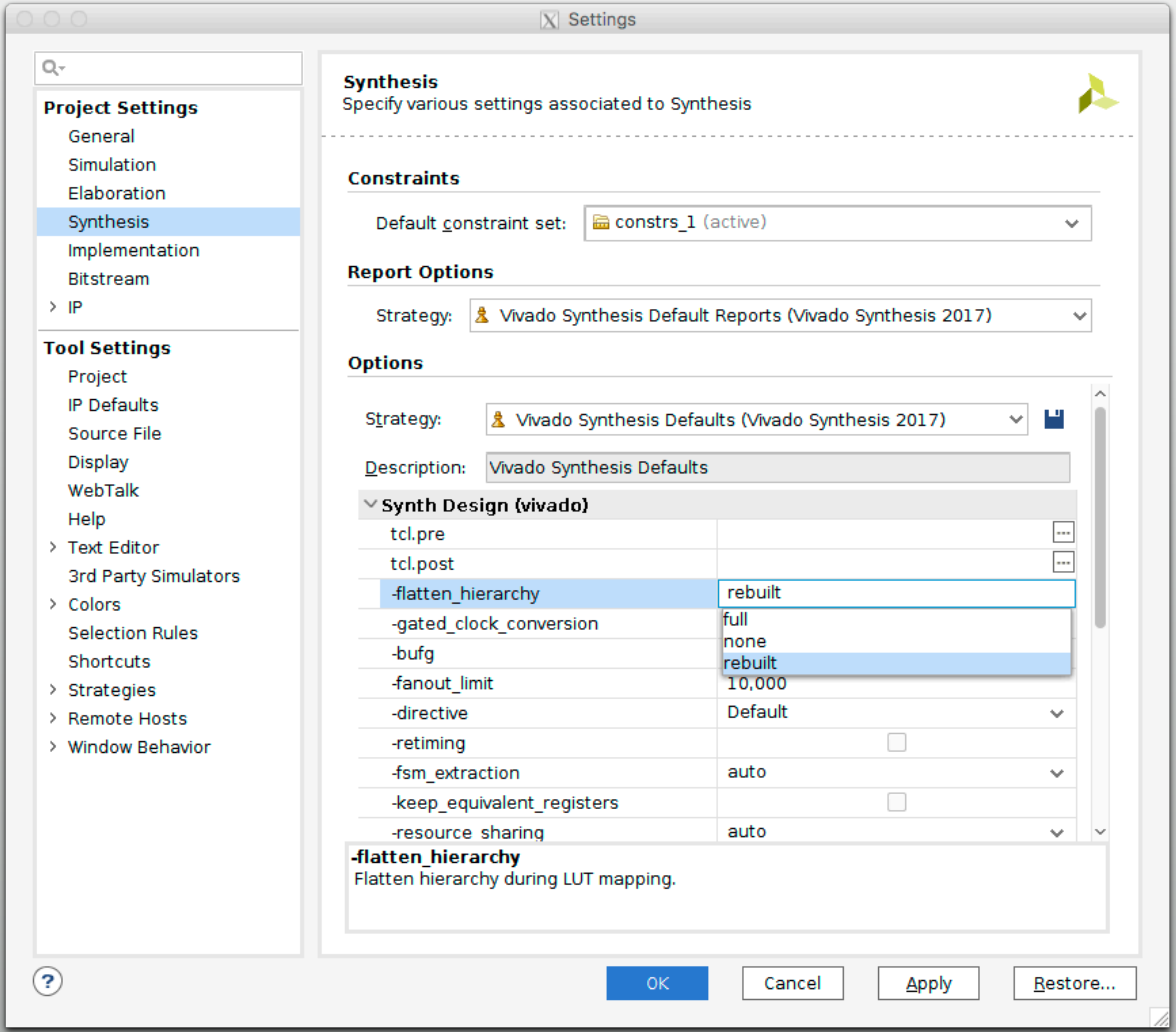
* Generate HDL model from synthesized/implemented netlist
    * Functional simulation: without delay model to check the correctness
    * Timing simulation: with delay model to check the timing
* Using RTL testbench is possible and sufficient for most cases

Synthesis and module hierarchy

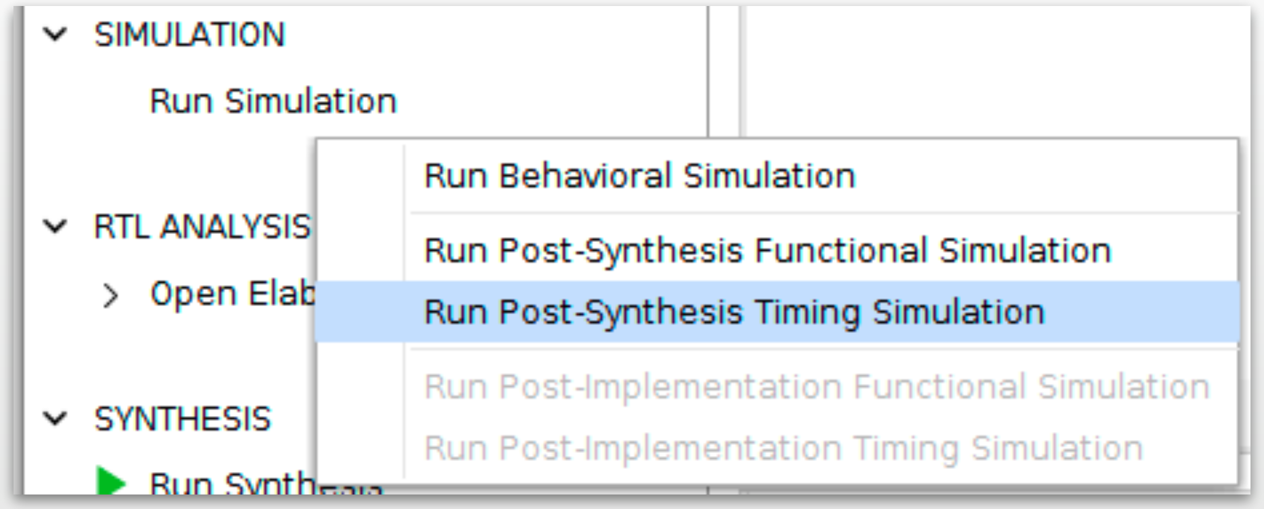
* RTL has module hierarchy
* In synthesis, it's flattened for optimization
  * Optimization beyond the module border gives better result
  * But hard to read synthesized gate-level HDL

Rebuilding Hierarchy

- Project Settings → Synthesis → Flatten Hierarchy
  - Full: Fully flatten, don't rebuild
  - None: No flattening
  - Rebuilt: Flatten then rebuild
    - Same performance with "full"



Post-synthesis simulation

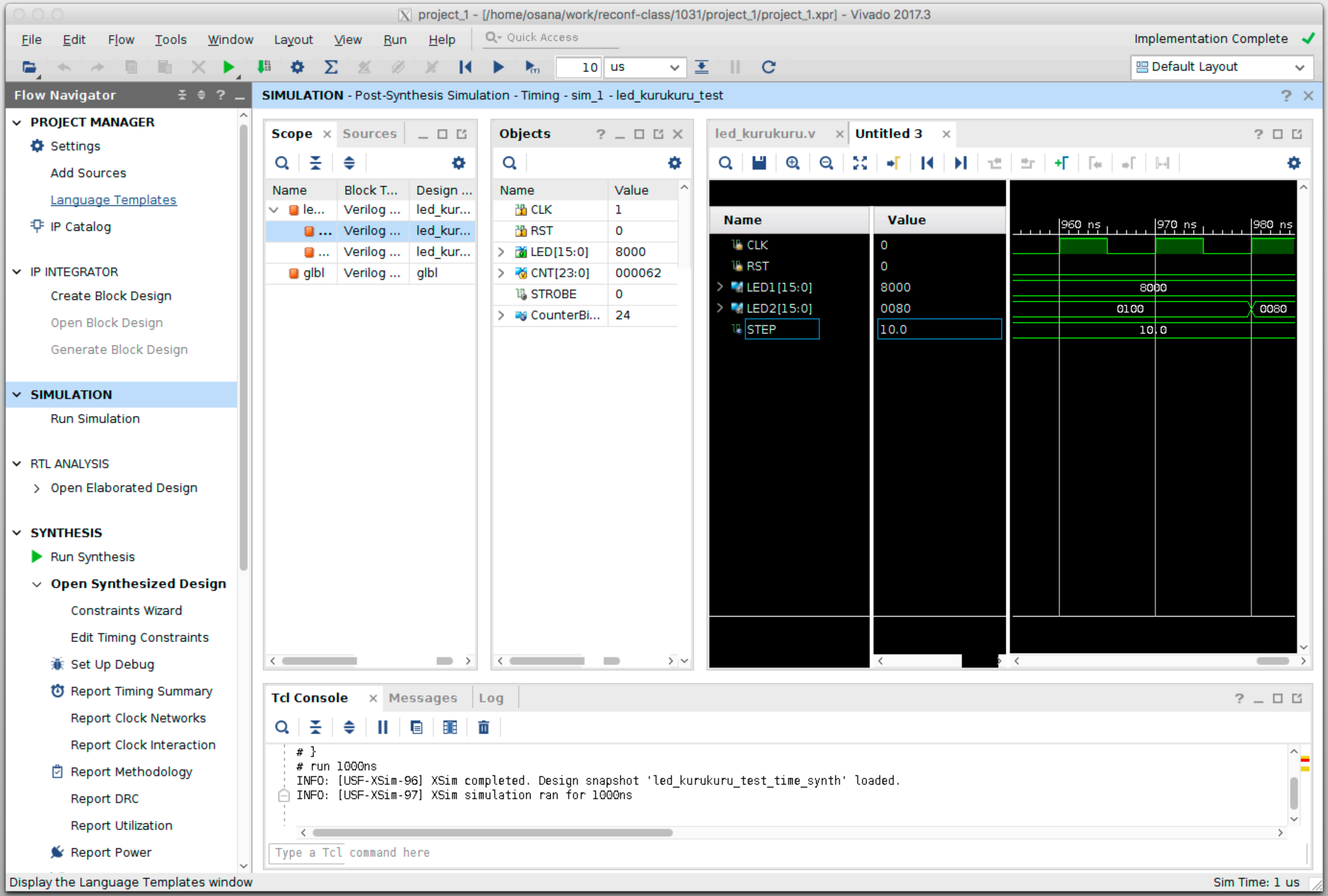


* Try with this source: LED flashing

```
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
```

Post-synthesis simulation

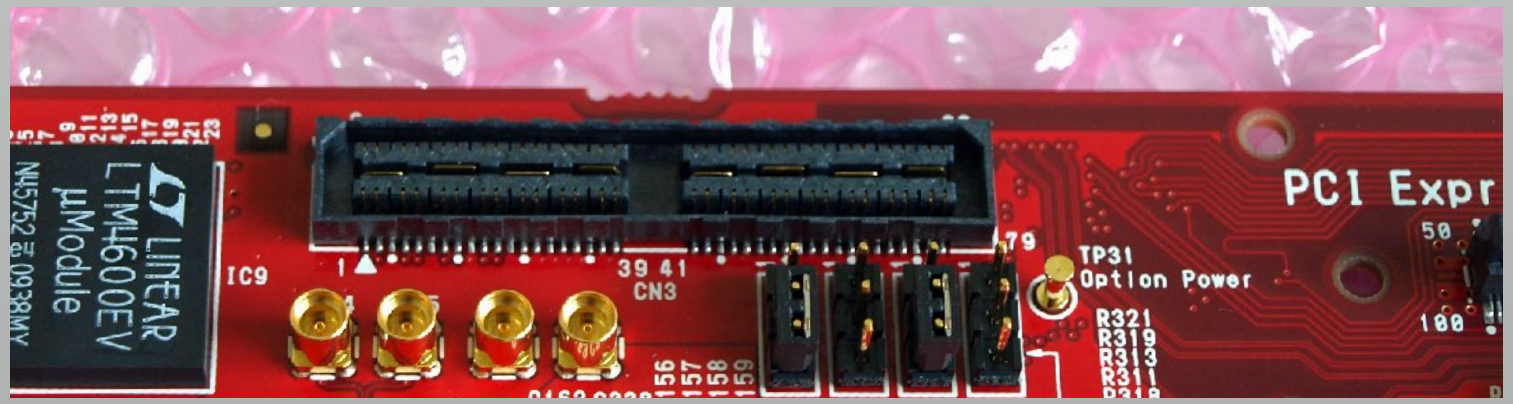
* Same to RTL sim at first sight
    * Some signal names may be changed
    * Signals come with delay (in timing simulation)



Off-chip signal probing

* Observing Off-chip signals from FPGA
  * Unused I/O pins may be used for debug
  * Standard 2.54mm pin headers, or some special high-density debug connectors

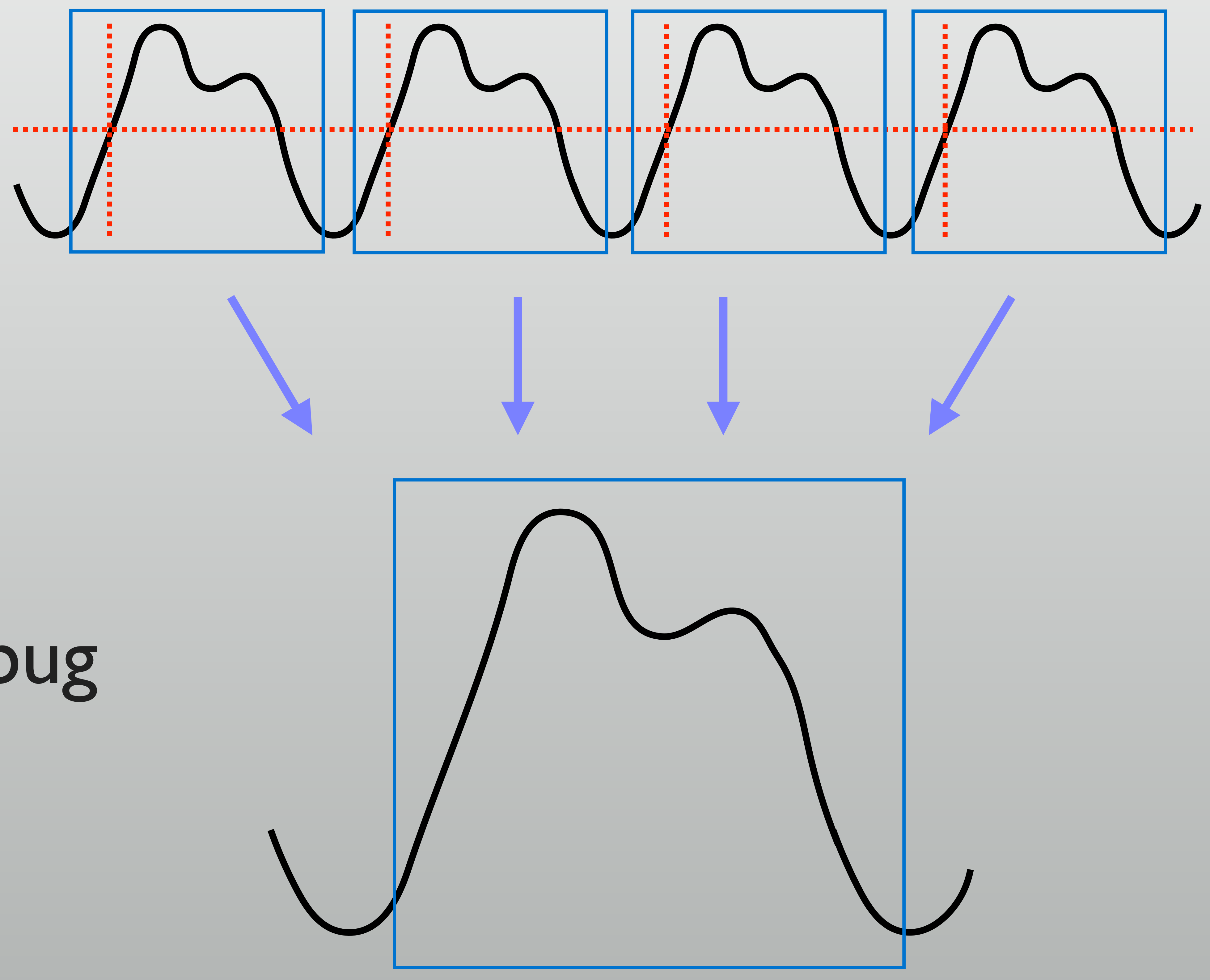
2.54mm square pin headers



Mictor connector

Using oscilloscopes?

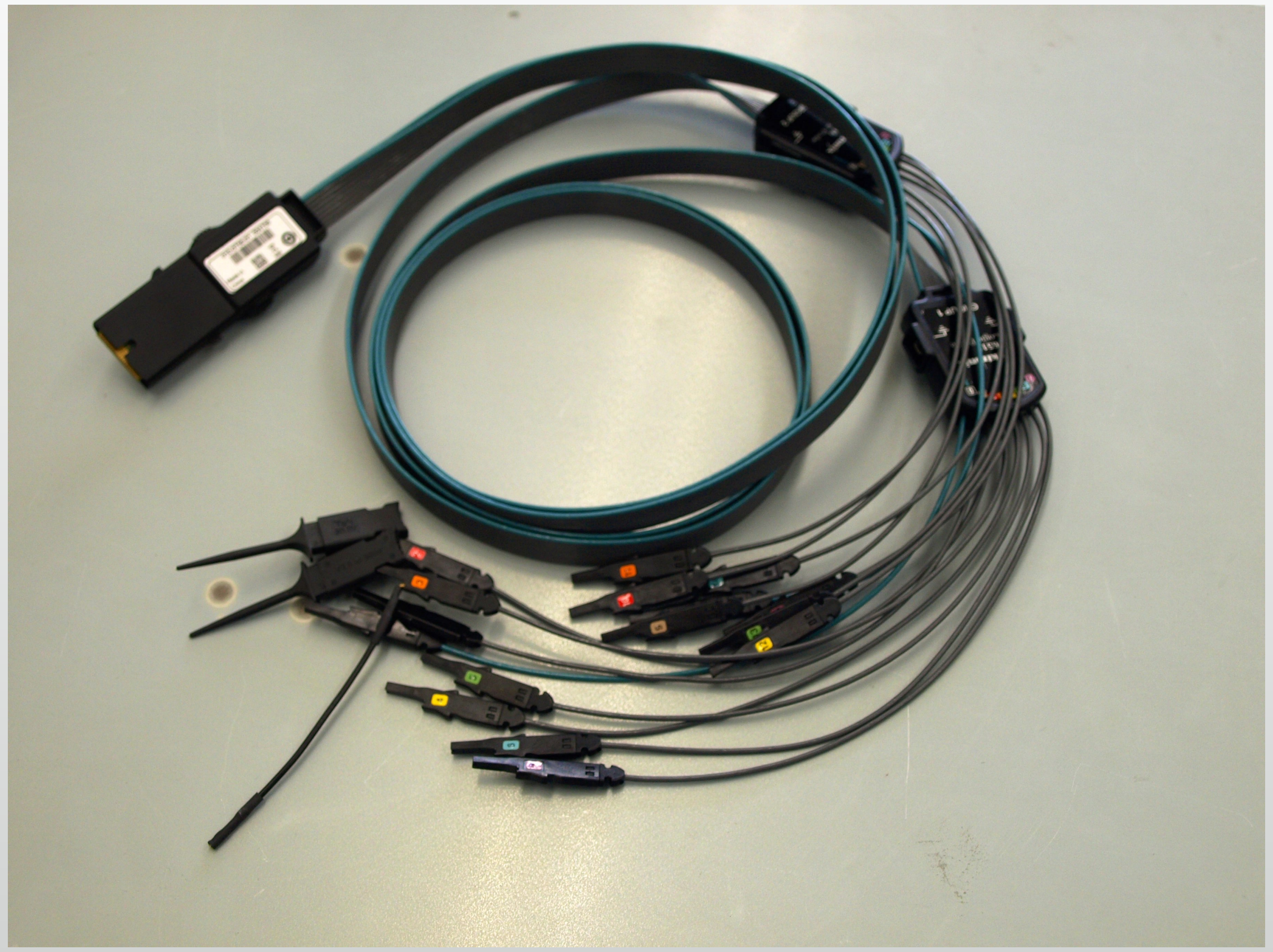
- Basically for periodic waveforms
    - Edge triggered
    - Small # of channels, continuous voltage levels
- But large # of channels are required in digital debug
    - No requirement for continuous voltage levels

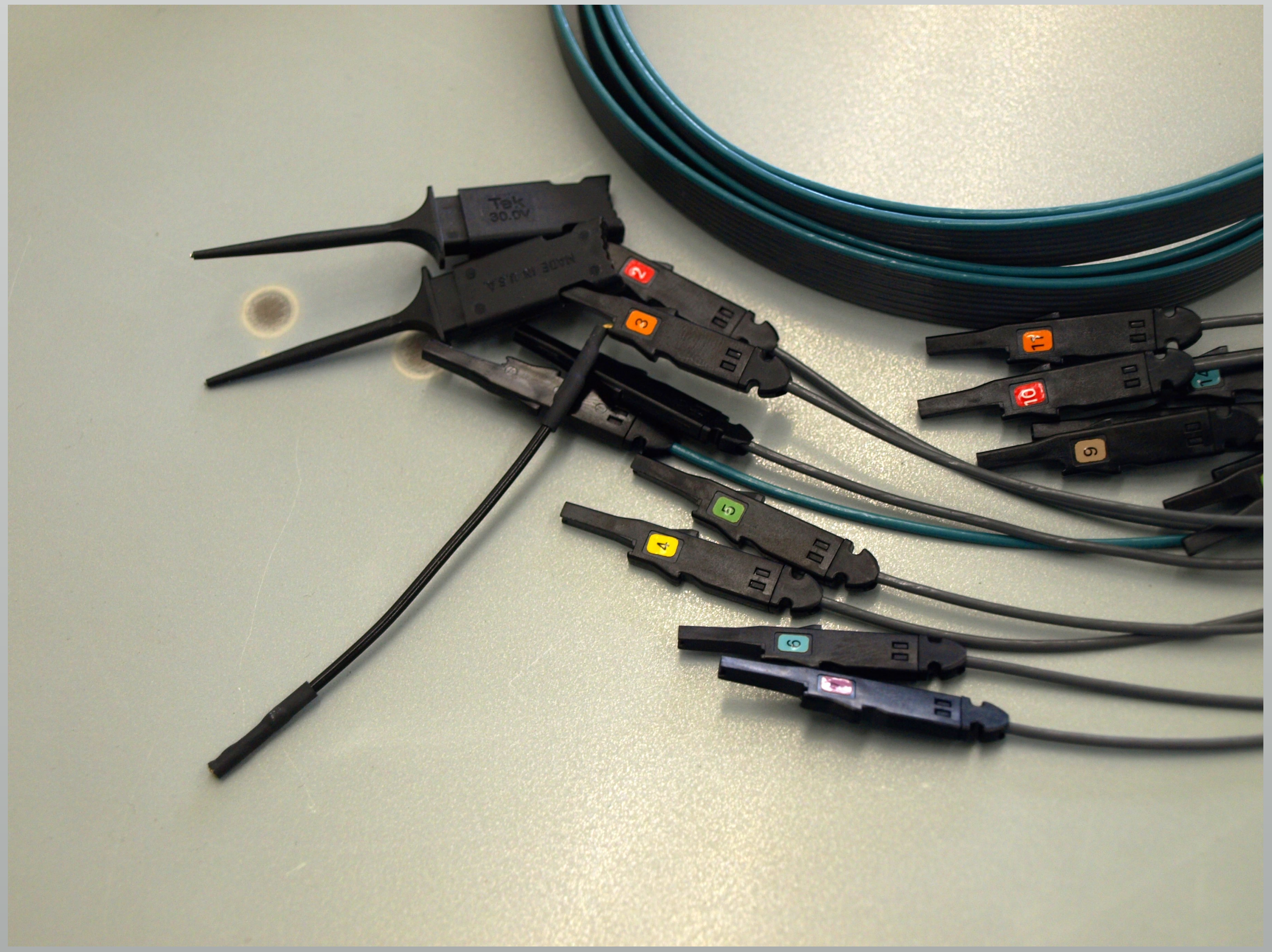


Logic analyzer

* Digital signal acquisition for many channels
  * Trigger once on specified condition
  * Observe waveform before and after the trigger
* Stand-alone logic analyzer, Integrated logic analyzer with oscilloscope, or on-chip ILA

Logic probes





* Much simpler than oscilloscopes'
  * with various IC clips and attachments
  * Pin-header attachments, Mictor attachments, etc.

Design considerations in off-chip debug

* Pulling out on-chip signals to outside by simply "assign" affects much on delay
    * Signal behavior may be changed
    * Placing an output register for off-chip signal can relax this problem
        * IOB has input/output register, used if no logic before/after input/output
* Especially for clock output, DDR register is useful (ask google for details…)

Plan B: Boundary Scan

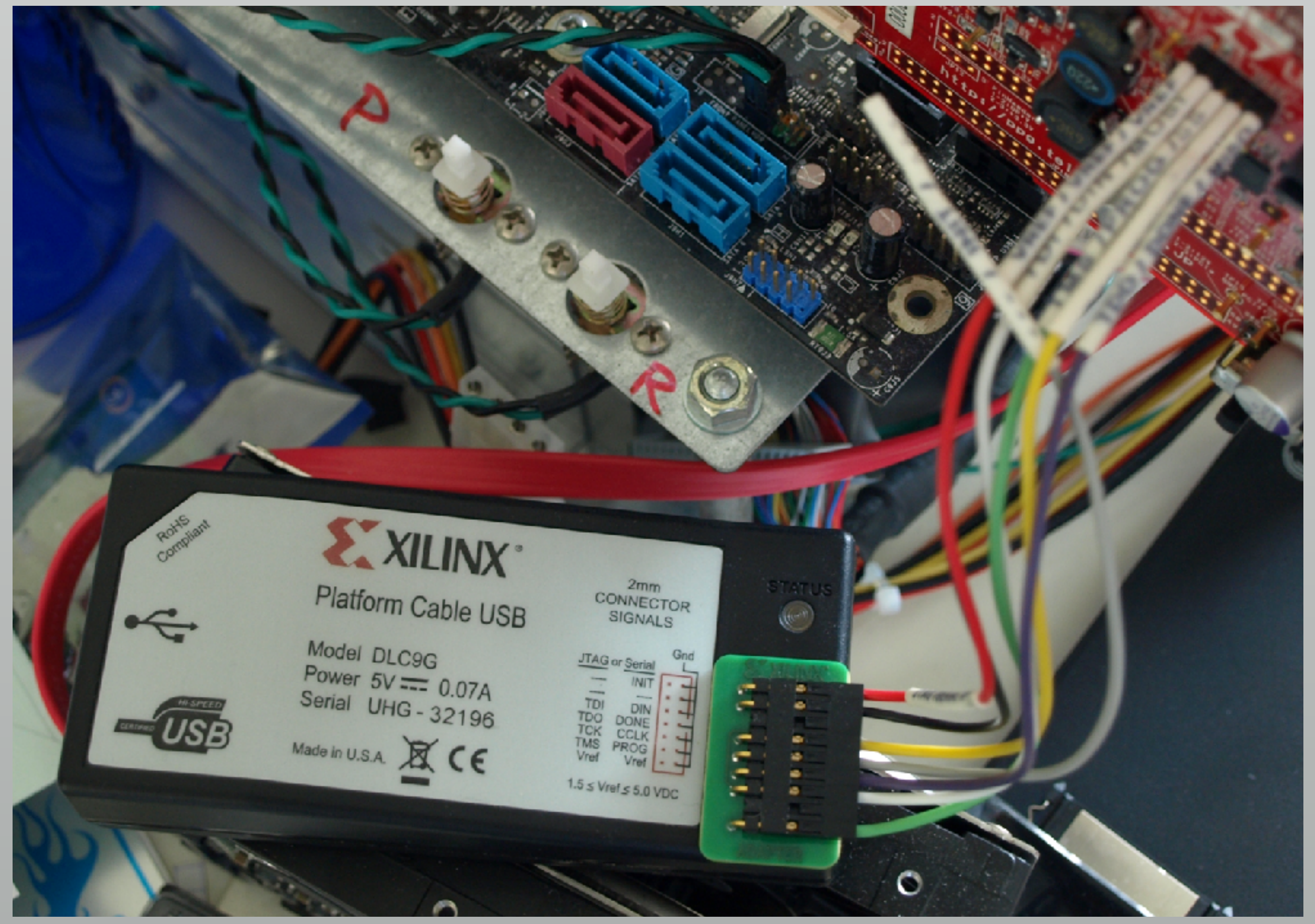
* Want to see all pins on all LSIs on a board
  * Not possible to probe all pins…
  * Make it possible with less cost (with some limitations)
* Place FFs on all pins of LSIs (and modules inside them), make a long long daisy-chain
  * Read bit-by-bit by shifting them on a clock signal

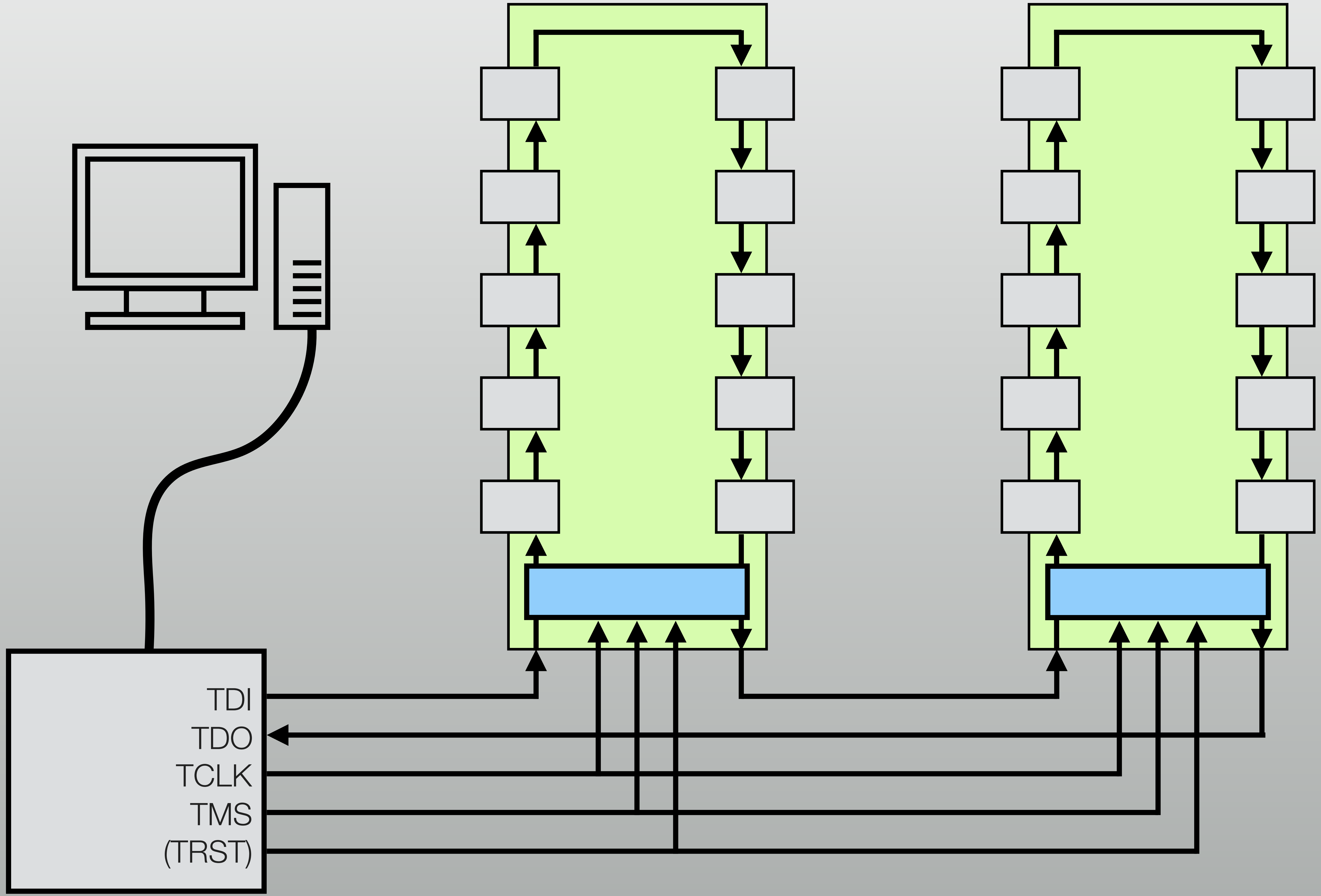
JTAG (Joint Test Action Group)

* IEEE 1149.1-1990 Standard Test Access Port and Boundary-Scan Architecture
    * 4-wire daisy chaining of LSIs
    * Monitor or control (!) pin status
        * Many recent LSIs have JTAG interfaces for test purpose
* JTAG is the name of working group, but used as the standard's name

JTAG interface

* Controllers are usually a USB device





Boundary Scan and FPGA

- Already using for FPGA configuration
  - Writing all LUTs and configuration FFs
  - Writing serially bit-by-bit requires less # of wires for configuration
    - But this is too slow, usually FPGAs come with parallel faster config modes
  - Good example of using JTAG interface for non-test purpose

Internal Logic Analyzer

* On-FPGA debug tool
  * Trigger state machine by FPGA logic cells
  * Block RAM as waveform memory, JTAG access from PC
* No instruments, only PC + JTAG cable is sufficient
  * Hands-on later

Direct probing is not a perfect solution

* RTL must be changed to use instruments or ILA
  * For ILA, some workaround to attach without RTL change is possible but at least the design must re-synthesized
* Always with limitations in waveform length
  * Observing in longer time scale is difficult
  * Additional signals (i.e., counters or some "OK" signals) are useful with ILAs

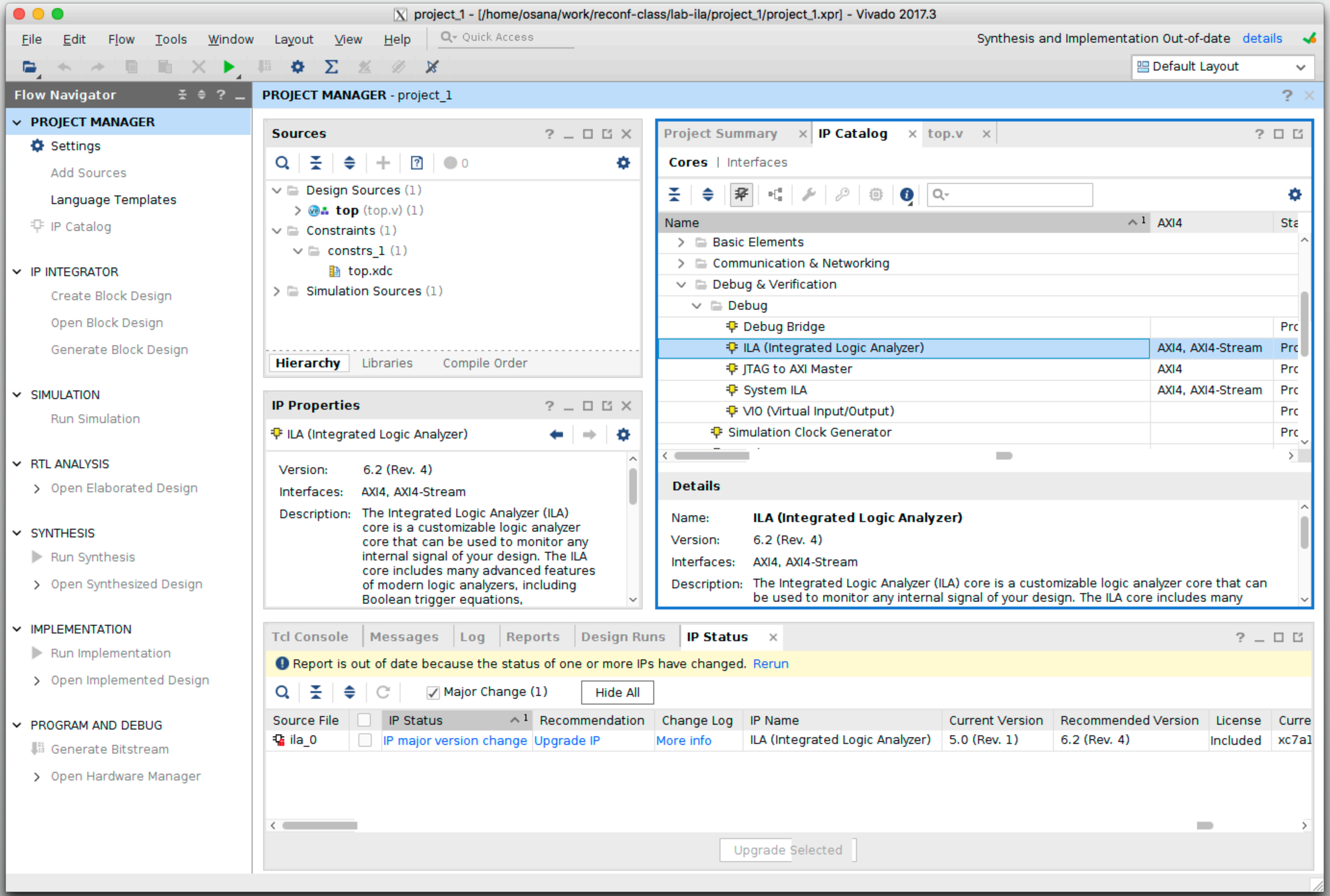
ILA hands-on

- Just a button and counter
  - See the counter with ILA

```
`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
```

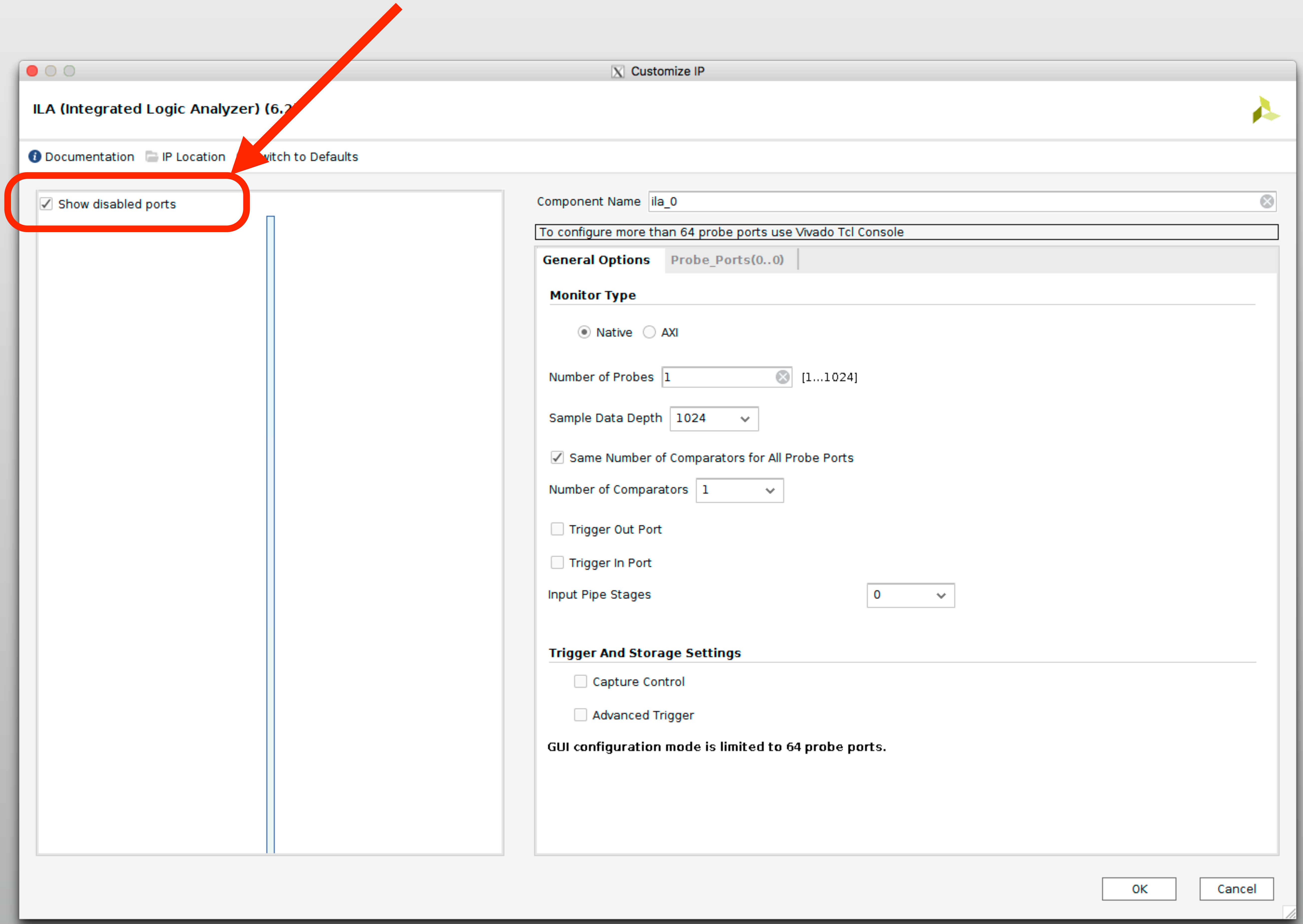
Generate ILA

* Open IP catalog
  * Debug & Verification → Debug → ILA



ILA configuration (1)

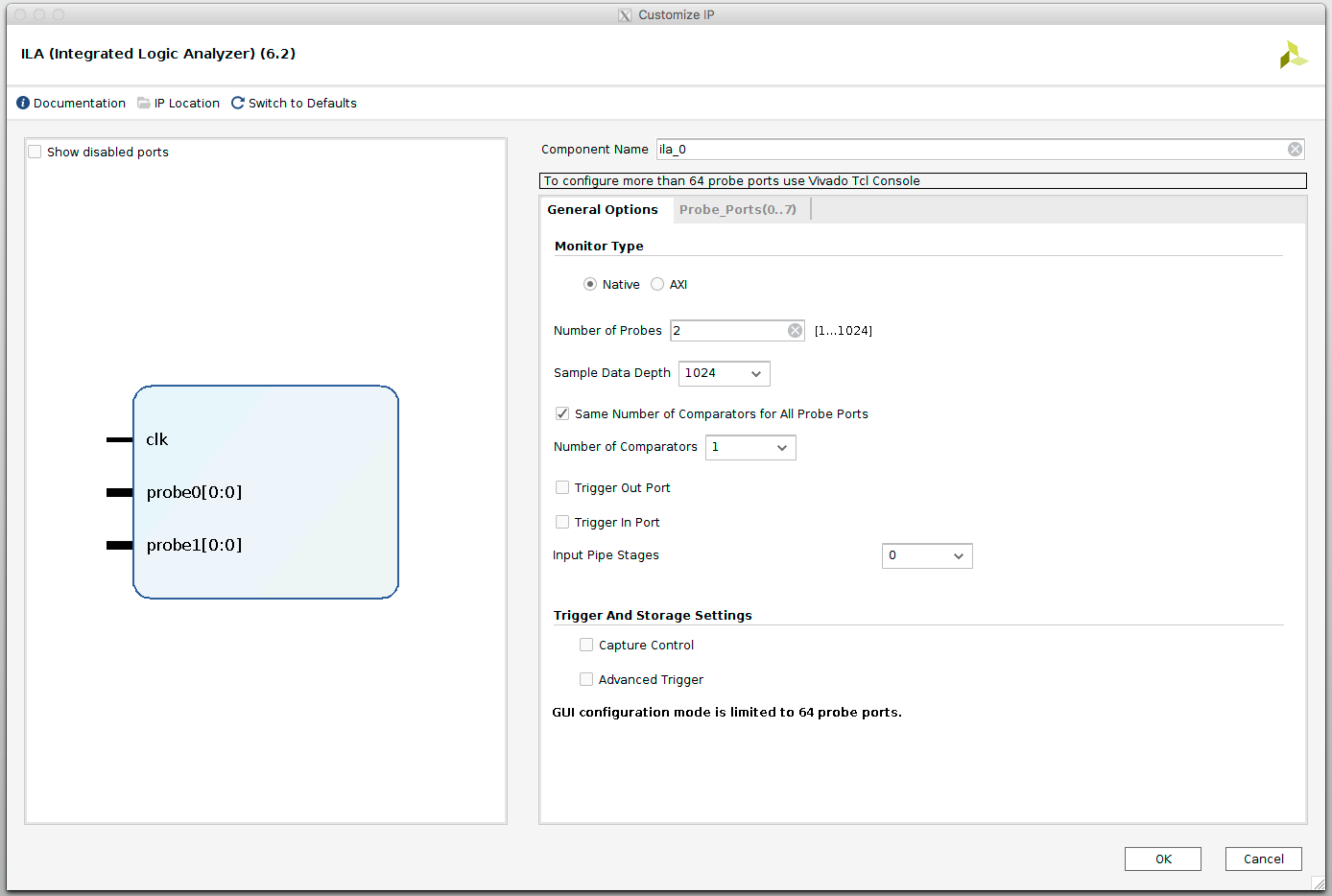
Keep this off (or very hard to see the diagram)



- Probe configurations
  - # of probes
  - Bit width of each probe
- Waveform record depth

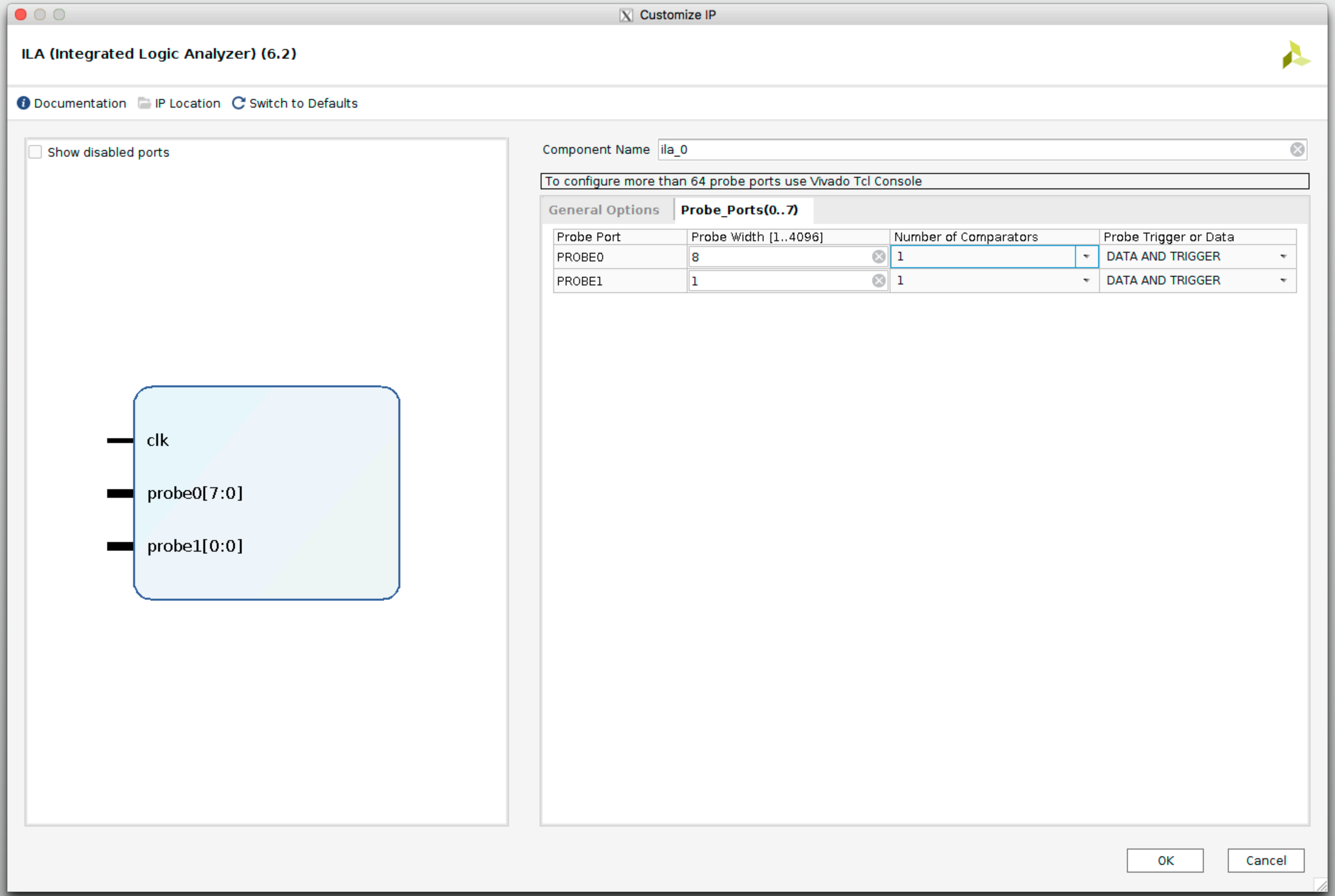
ILA configuration (2)

* Set # of probes to 2
  * Keep the depth of 1024



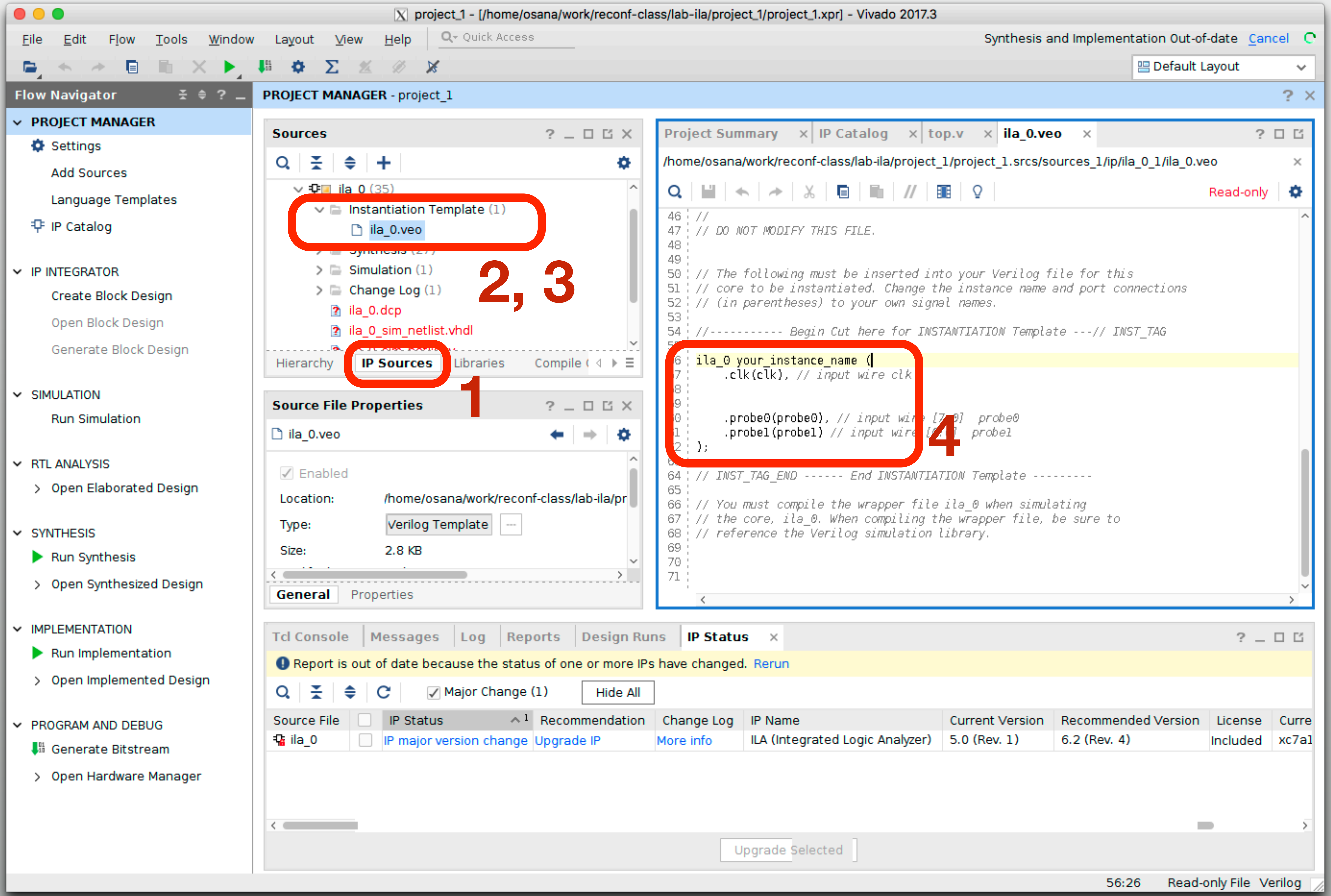
ILA configuration (3)

- Probe settings in “Probe ports”
  - Set Probe0 to 8 bits
  - OK to generate

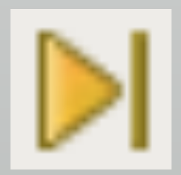


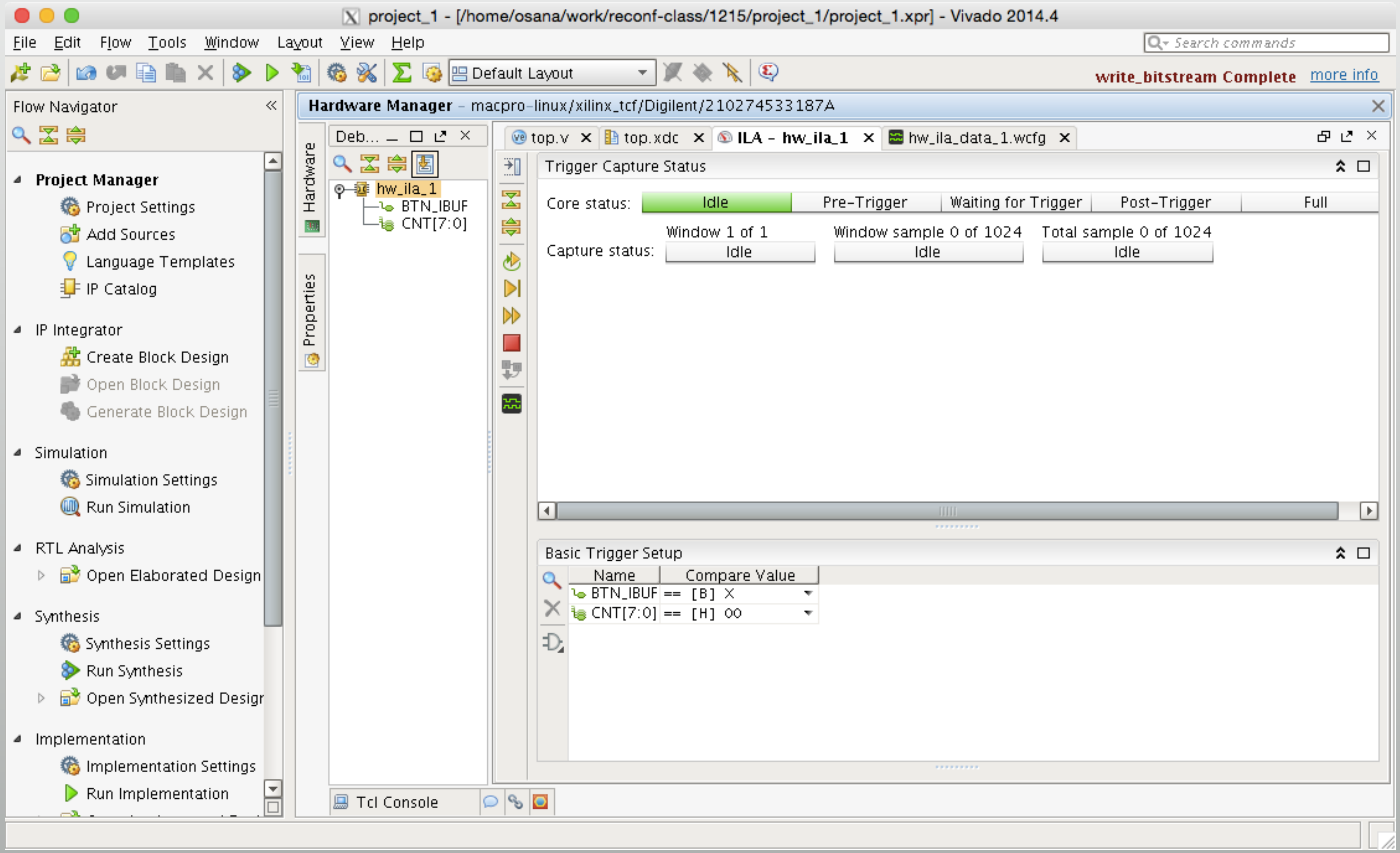
Check interface

* Sources → IP Sources
* Check the Instantiation template in the core



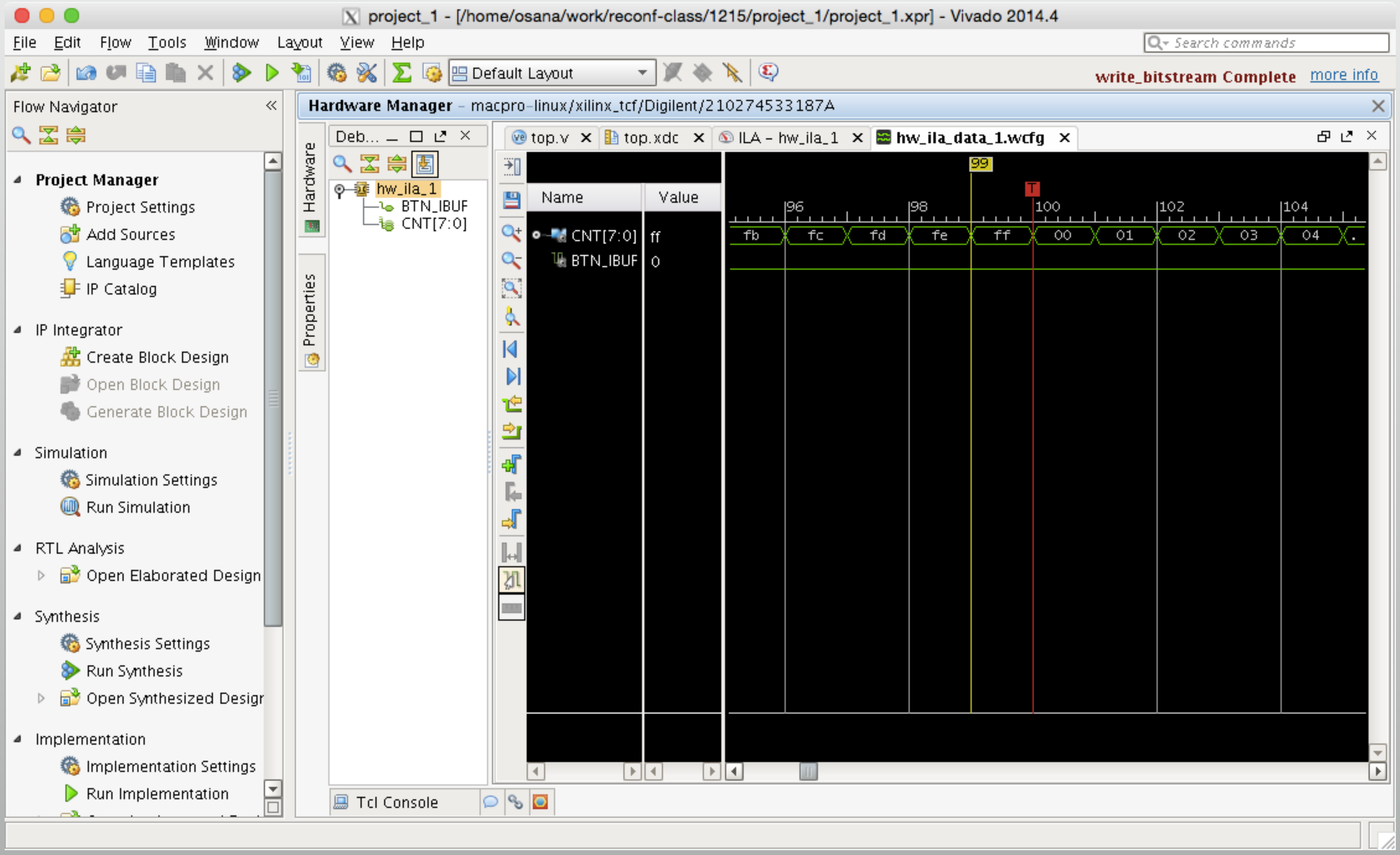
Hardware Manager

* ILA found automatically
  * Set trigger condition
  * Position is important
* Wait for trigger 
* Trigger now 



Observe waveform

* CNT = 0 at
  Position = 100



Things to do

- Wite RTL: on slide #39, Generate ILA core too
  - Without ILA core, CNT will be removed because it has no output load
  - Add "output [7:0] LED_O" and "assign LED_O = CNT;" (but this may not necessary)
- Write a testbench: CLK, RST, BTN are sufficient (Running CLK is essential; do as you like for other signals)
- Try RTL simulation, then synthesize the design and run Post-synthesis timing simulation
  - There will be a slight delay from clock edge to counter increment, while there's no delay in RTL simulation
- Create an XDC file, locate all signals: Clock on E3 pin, see the board for button and LEDs
- Run implementation, check all pin assignment and program the FPGA: ILA will appear in Vivado's hardware manager
  - Trigger Setup: set BTN_IBUF = 1, then arm the trigger. Pressing down the button will trigger the ILA.
  - Set Trigger position in Window to 900: then you'll see more clock cycles before the trigger