Parallel and Reconfigurable VLSI Computing (3)

Walk Through FPGA Design

Hiroki Nakahara

Tokyo Institute of Technology

Outline

- Design flow, design tool, development environment for implementation of target system on FPGA
- Target board: Digilent Zybo-Z7
- Design tool: Xilinx Vivado 2017.4, XSDK 2017.4

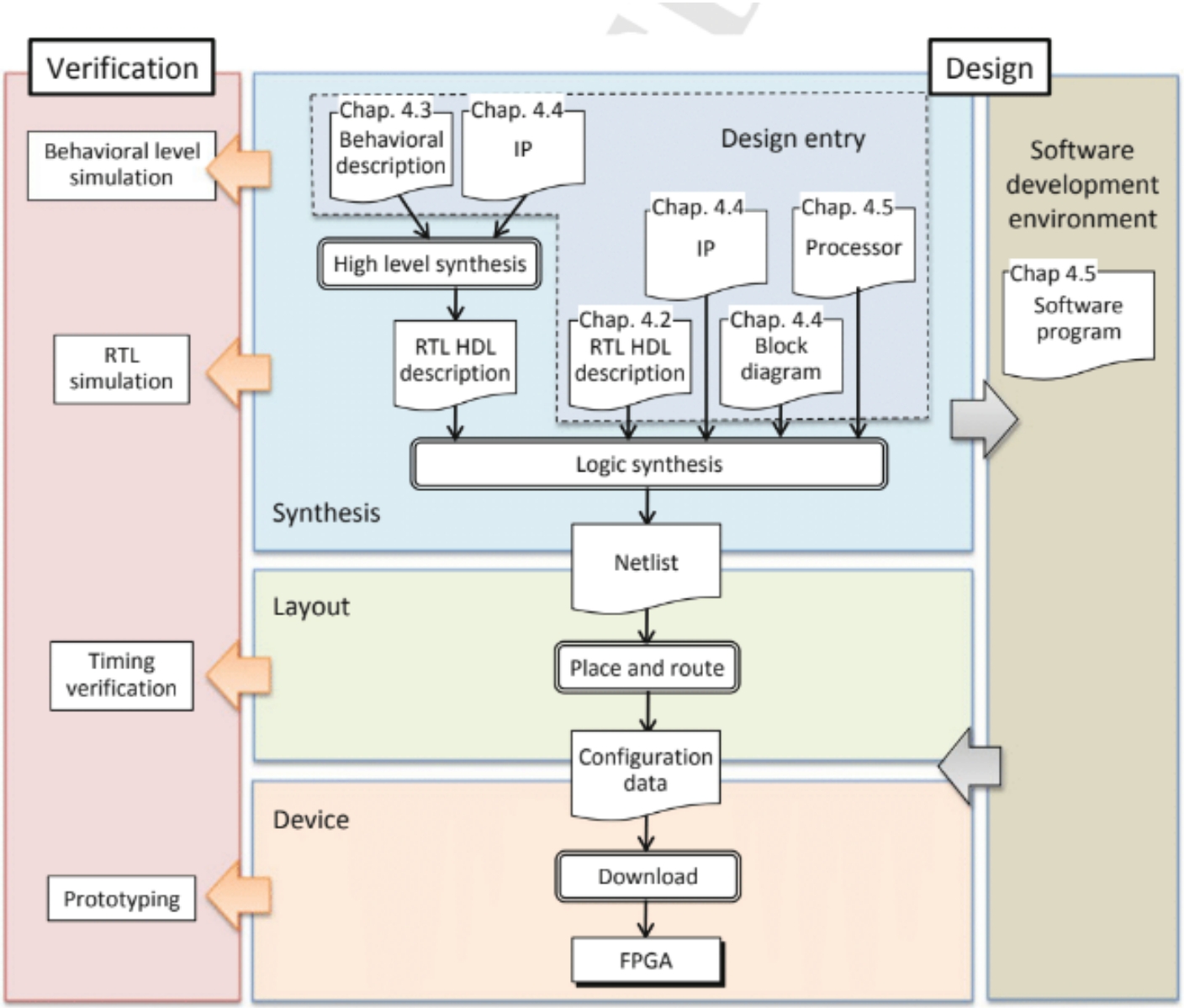
1 Design flow overview

2 HDL design flow with logic simulation

3 Processor design flow

Conventional FPGA Design Flow

FPGA Design Flow



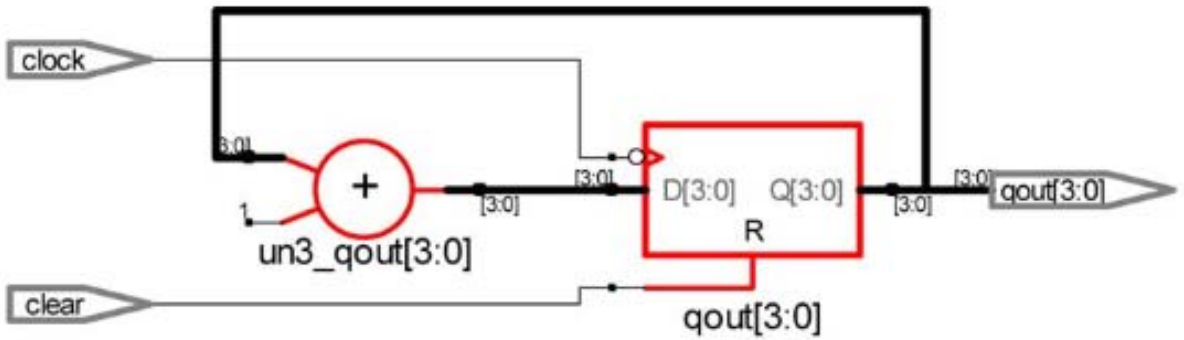
Register-Transfer Level (RTL) Design

- Combinational Logic
  - Un-timed behavior (equation or truth table)
- Sequential Logic
  - Timed one (Finite state machine (FSM))
  - Register + Combinational logic
- RTL design
  - Design behavior using high-level state machine (to be explained)
  - Remaining: Convert to sequential circuit

Hardware Description Language (HDL)

- Represents hardware structure and behavior
  - Can be processed by computers (as a software)
- VHDL, Verilog-HDL, AHDL, myHDL, MODAL
- Used for simulation/synthesis

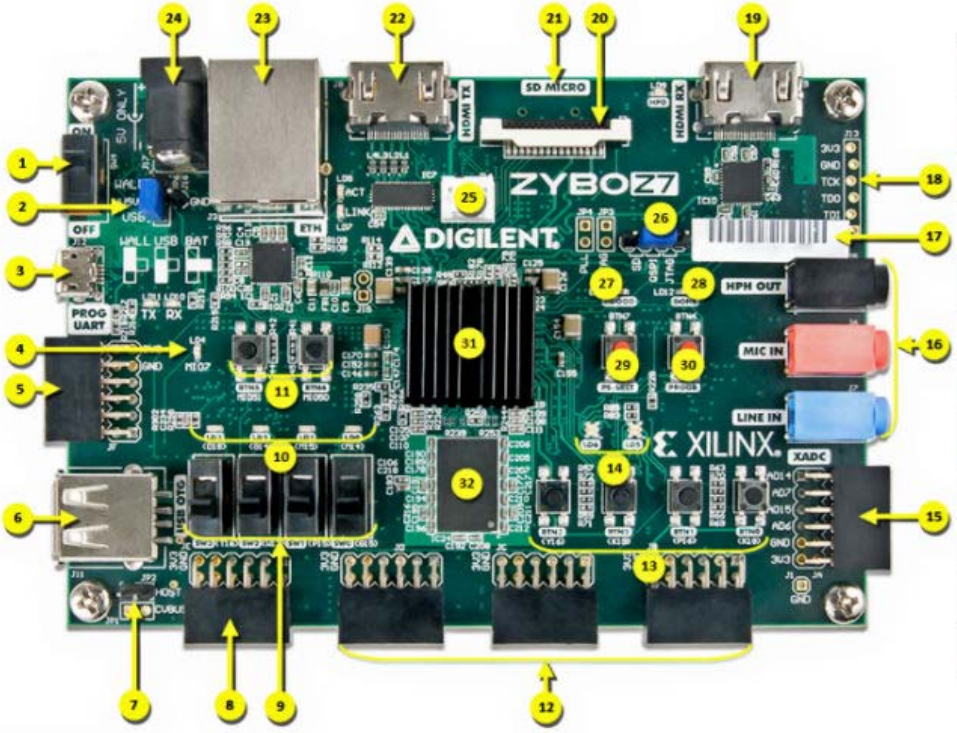
```
// the body of the 4-bit counter.
always @(negedge clock or posedge clear)
    if (clear)
        qout <= 4'd0;
    else
        qout <= (qout + 1) ;   // qout = (qout + 1) % 16;
```



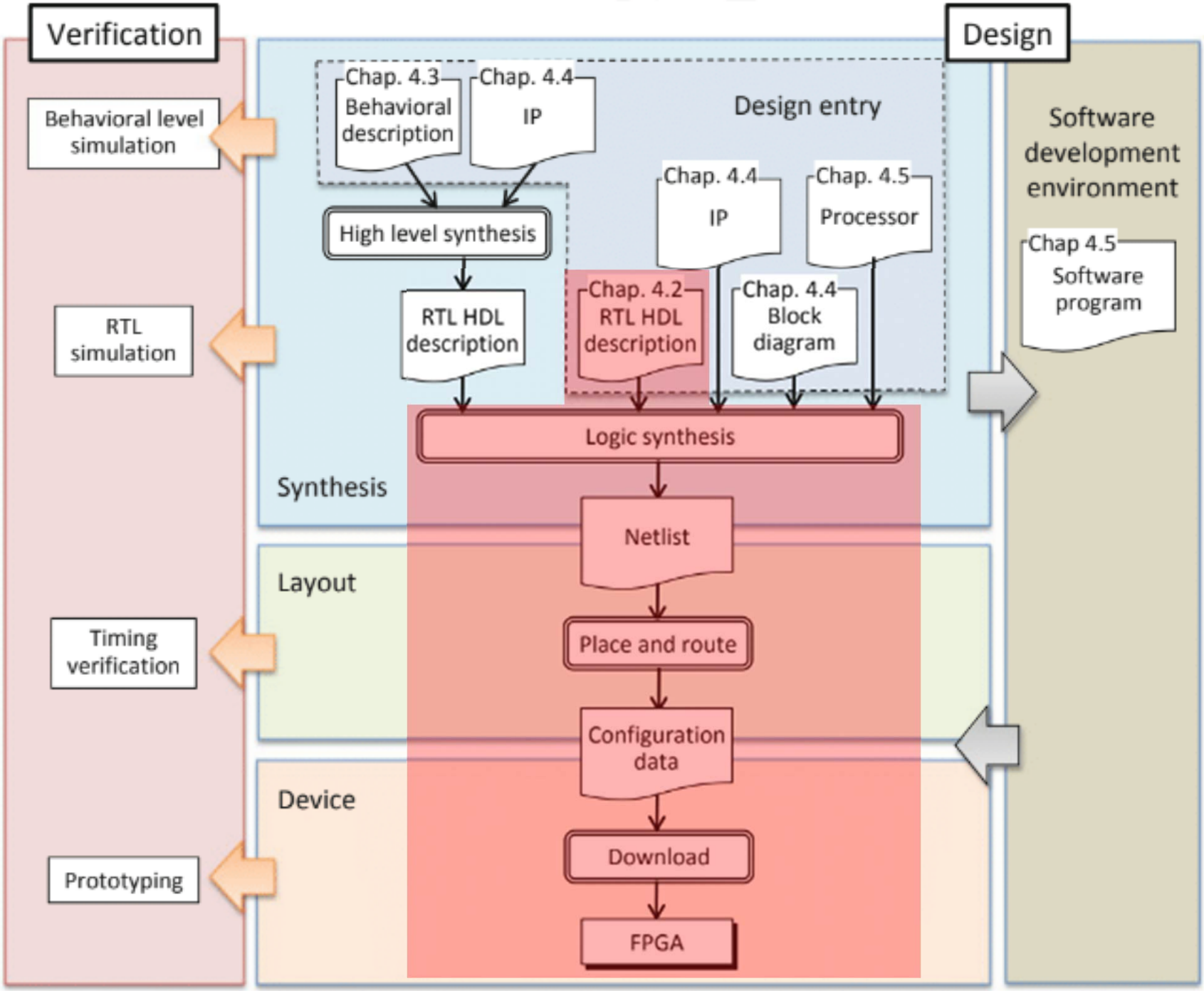
HDL Design Flow with Simulation

Target FPGA Board

- Digilent Zybo Z7-10/Z7-20
- Resource: https://reference.digilentinc.com/reference/programmable-logic/zybo-z7/start
- Manual: https://reference.digilentinc.com/_media/reference/programmable-logic/zybo-z7/zybo-z7_rm.pdf



Call out	Description	Call out	Description	Call out	Description
1	Power Switch	12	High-speed Pmod ports *	23	Ethernet port
2	Power select jumper	13	User buttons	24	External power supply connector
3	USB JTAG/UART port	14	User RGB LEDs *	25	Fan connector (5V, three-wire) *
4	MIO User LED	15	XADC Pmod port	26	Programming mode select jumper
5	MIO Pmod port	16	Audio codec ports	27	Power supply good LED
6	USB 2.0 Host/OTG port	17	Unique MAC address label	28	FPGA programming done LED
7	USB Host power enable jumper	18	External JTAG port	29	Processor reset button
8	Standard Pmod port	19	HDMI input port	30	FPGA clear configuration button
9	User switches	20	Pcam MIPI CSI-2 port	31	Zynq-7000
10	User LEDs	21	microSD connector (other side)	32	DDR3L Memory
11	MIO User buttons	22	HDMI output port	* denotes difference between Z7-10 and Z7-20	


Verification
Design
Behavioral level simulation
RTL simulation
Timing verification
Prototyping
Chap. 4.3
Behavioral description
Chap. 4.4
IP
Design entry
High level synthesis
RTL HDL description
Chap. 4.4
IP
Chap. 4.5
Processor
Chap. 4.2
RTL HDL description
Chap. 4.4
Block diagram
Logic synthesis
Synthesis
Netlist
Layout
Place and route
Configuration data
Device
Download
FPGA
Software development environment
Chap 4.5
Software program

Run Vivado (Not HLS!!)

source /opt/Xilinx/Vivado/2017.4/settings64.sh

vivado &

File -> New Project

In "Create a New Vivado Project" window, Click "Next"

In "Project Name" window, set followings:

Project name: hello_hw_1

Project location: C:/FPGA/lec2 (Windows)

/root/FPGA/lec2 (Unix)

Then, project will be created at: C:/FPGA/lec2/hello_hw_1

(/root/FPGA/lec2/hello_hw_1)

Next, Click "Next"

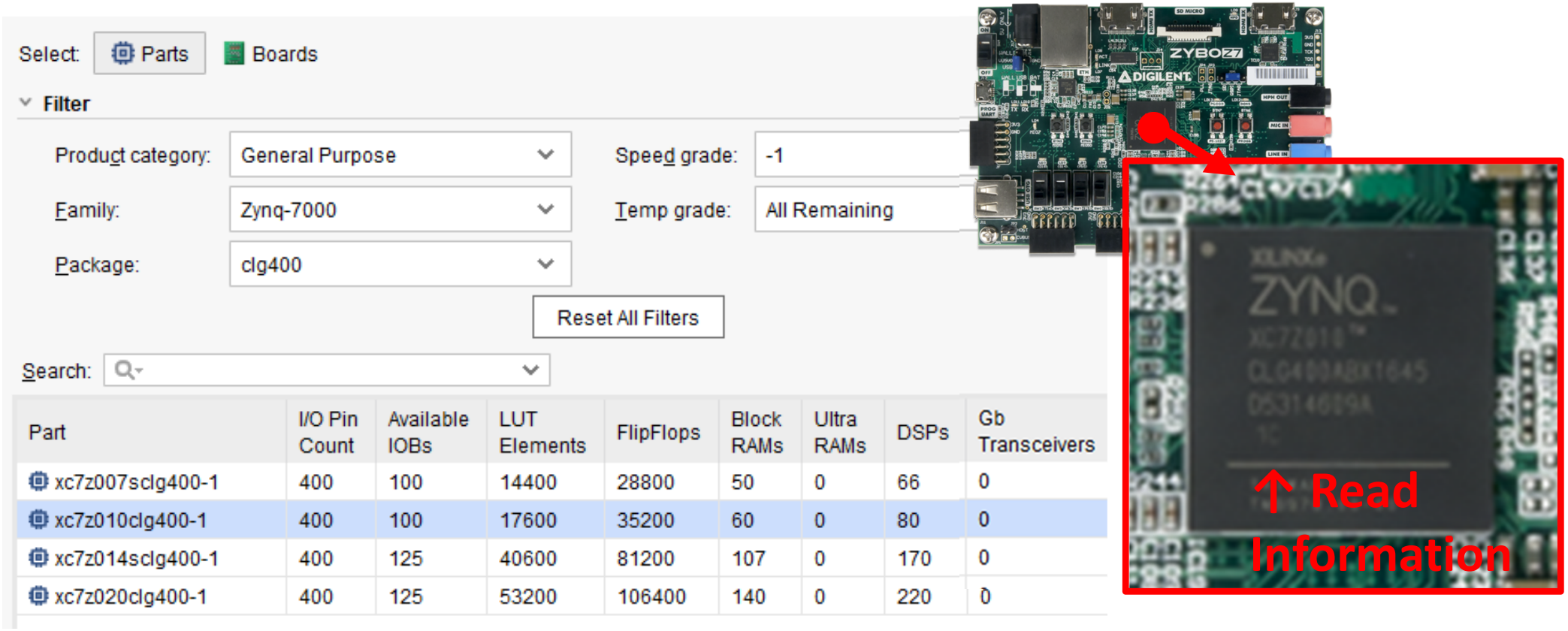
Settings (Cont'd)

In "Project Type", check "RTL Project", then "Next"

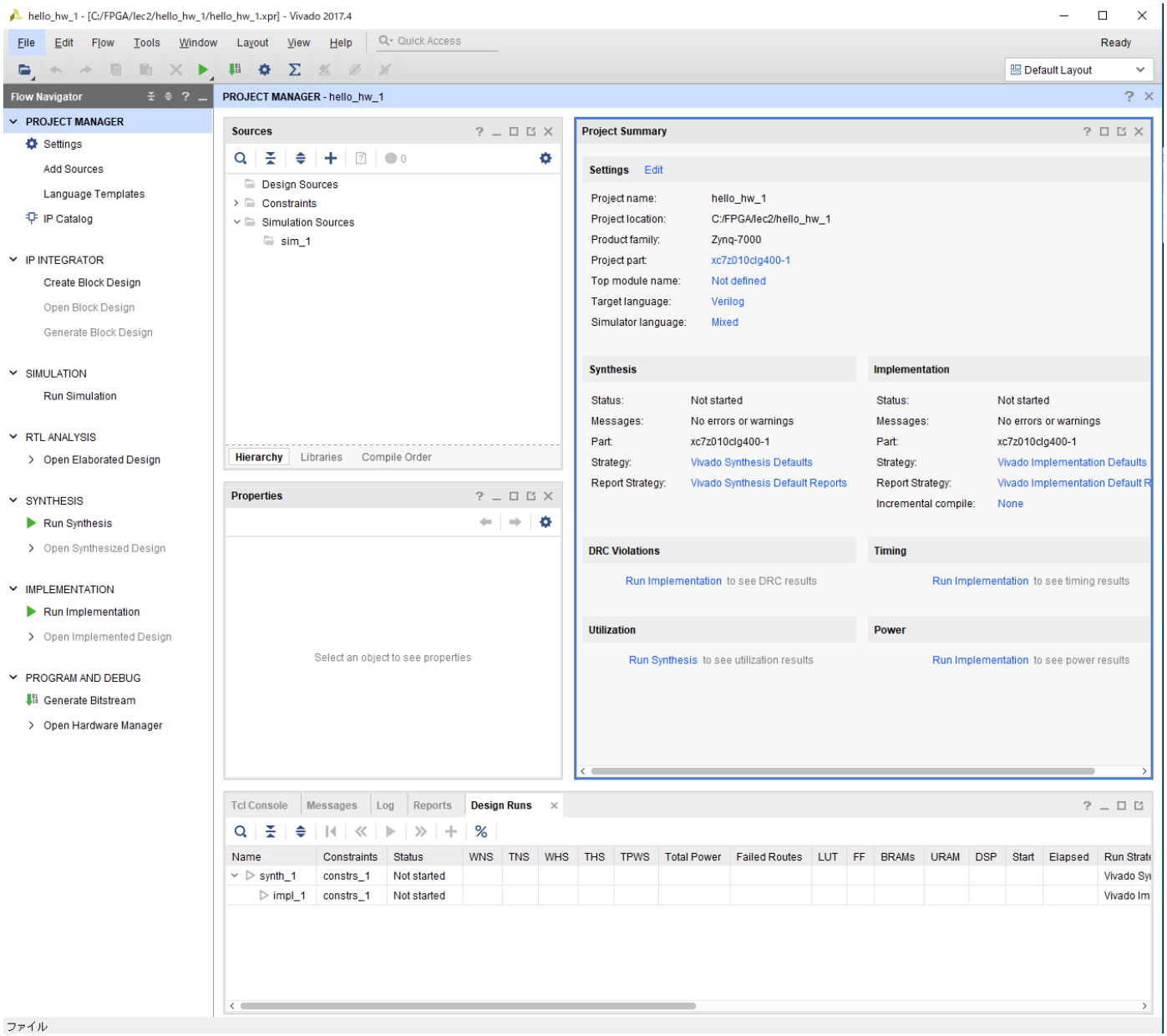
In "Add sources", just click "Next", and in "add constraints", click "Next"

In "Default Part", carefully choose your FPGA!!, then "Next"

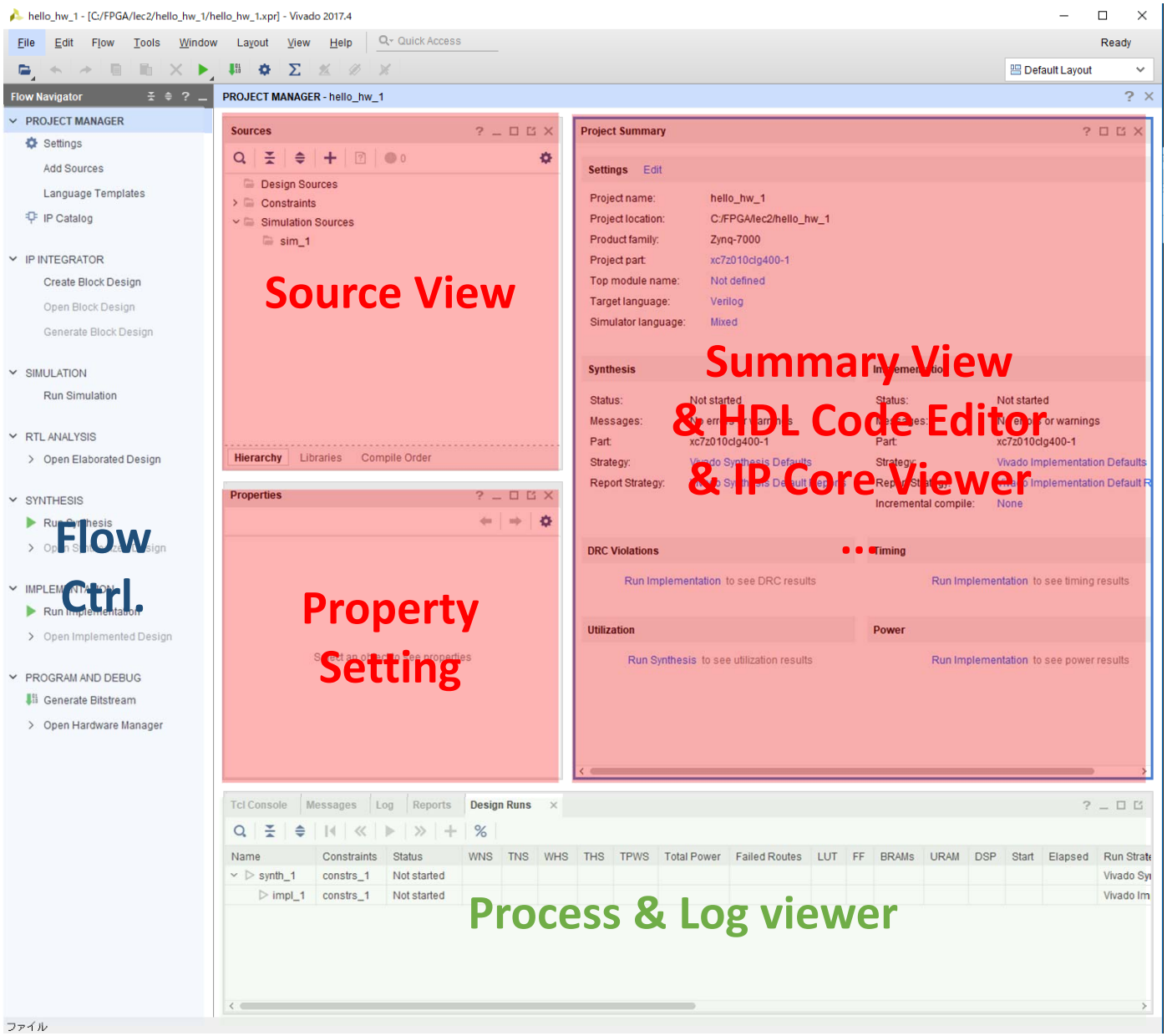
Finally, in "New Project Summary", then click "Finish"

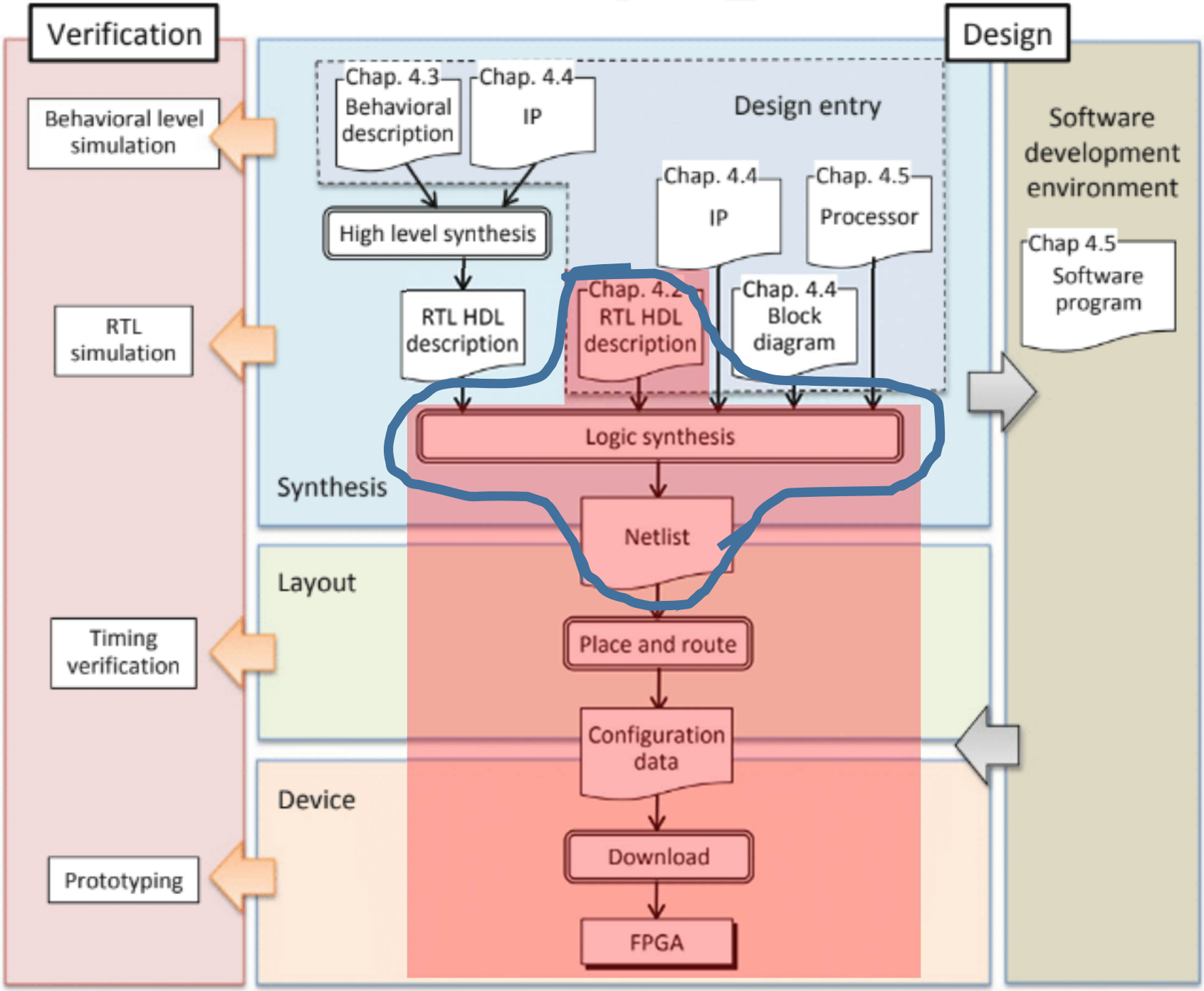


Vivado 2017.4 GUI



Vivado 2017.4 IDE




Verification
Design
Behavioral level simulation
RTL simulation
Timing verification
Prototyping
Chap. 4.3
Behavioral description
Chap. 4.4
IP
Design entry
High level synthesis
RTL HDL description
Chap. 4.2
RTL HDL description
Chap. 4.4
IP
Chap. 4.5
Processor
Chap. 4.4
Block diagram
Logic synthesis
Synthesis
Netlist
Layout
Place and route
Configuration data
Device
Download
FPGA
Software development environment
Chap 4.5
Software program

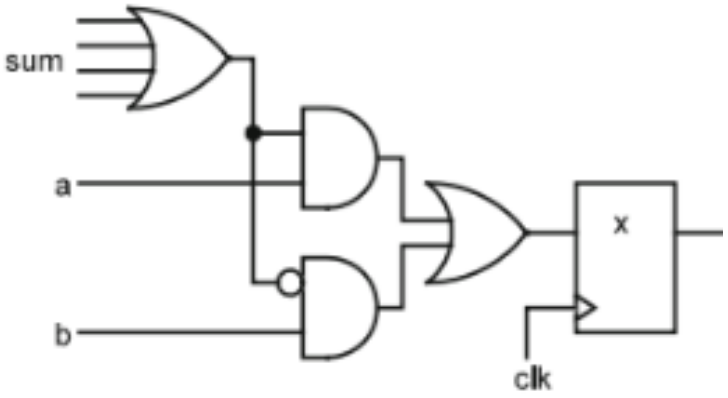
Logic Synthesis

- Synthesis of logic circuit (Mainly, sequential circuit) from RTL description
- Output: Netlist
- Netlist: Represents a set of logic elements such as primitive gates and flip-flops and their connections

```
always @(posedge clk) begin
      if (sum > 0)
      x <= a;
      else
      x <= b;
end
```

(a) RTL description

Logic synthesis



(b) Gate level netlist

Create HDL Source File

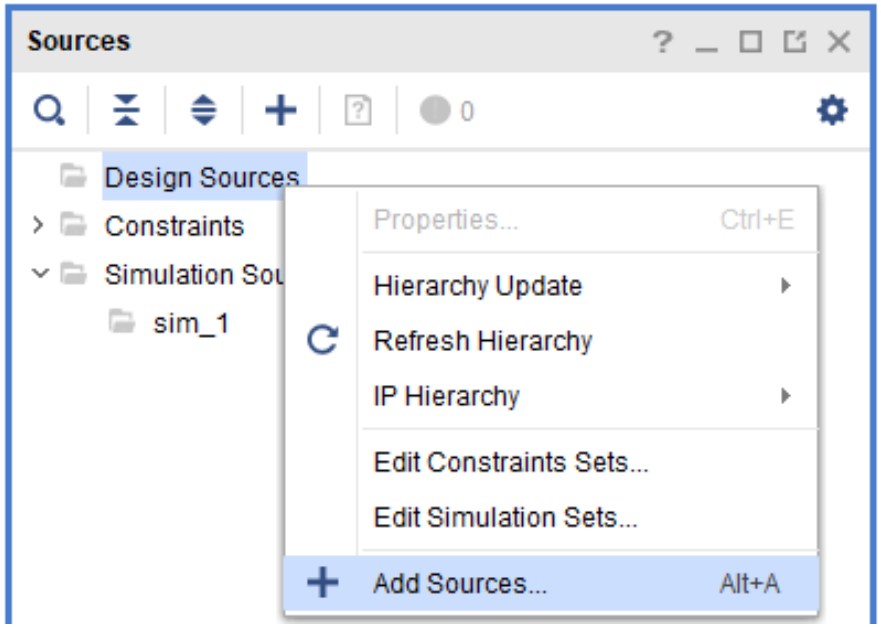
In "Sources", right clock "Design Sources", then click "Add Sources..."

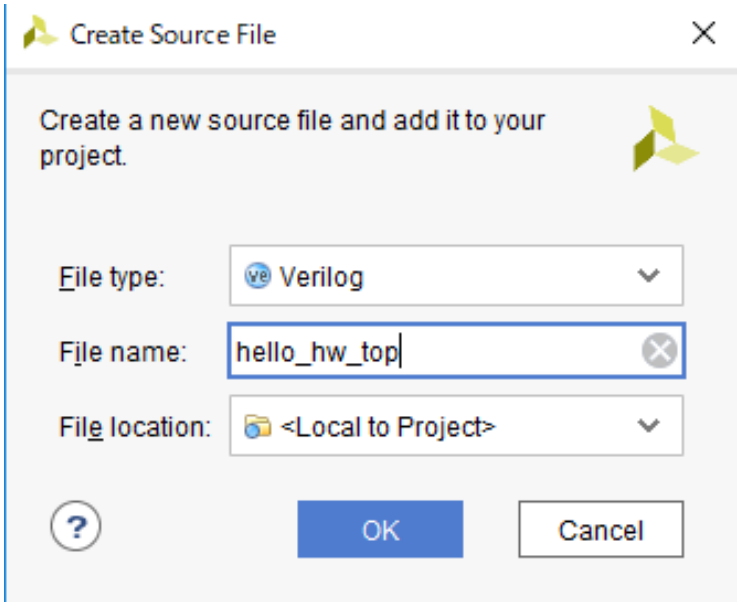
In "Add Sources", confirm "Add or create design sources", then "Next"

In "Add or Create Design Sources", click "Create File"

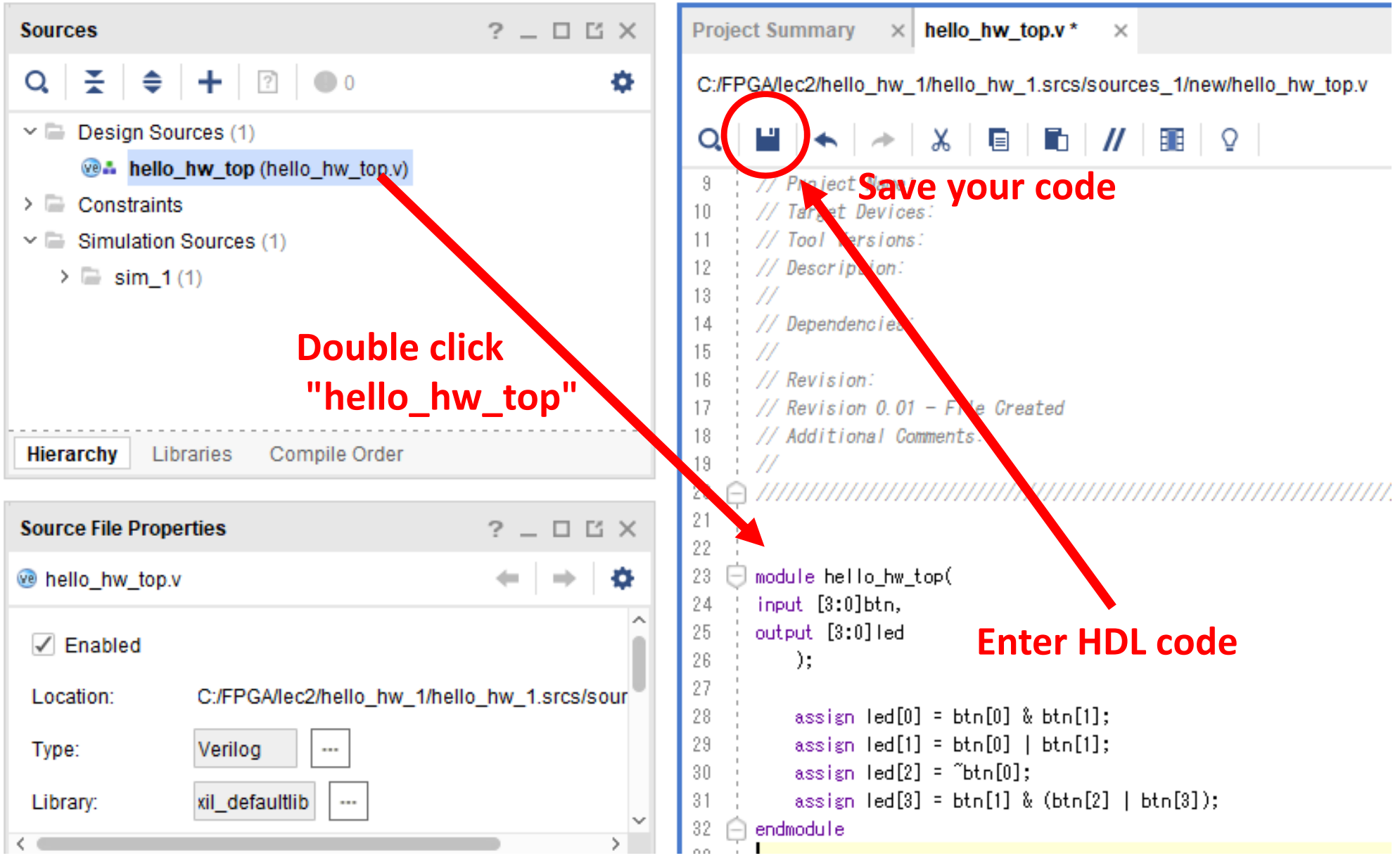
In "Create Source File", enter "hello_hw_top" to "File name:", then "OK"

Back to "Add or Create Design Sources", then "Finish", "OK", and "Yes"





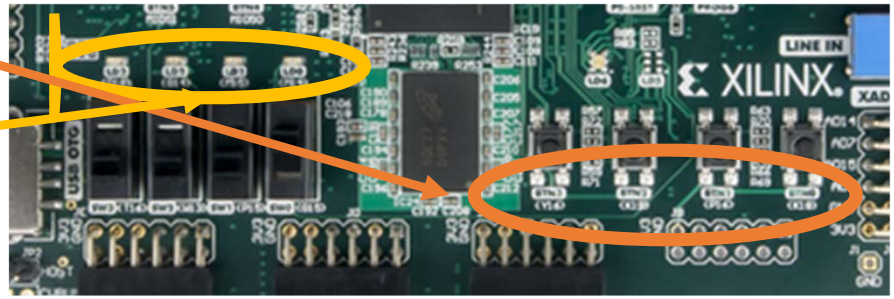
Write Your First HDL



First Hardware

hardware must be specified by these words

```
module hello_hw_top(
input [3:0]btn,
output [3:0]led
    );
```



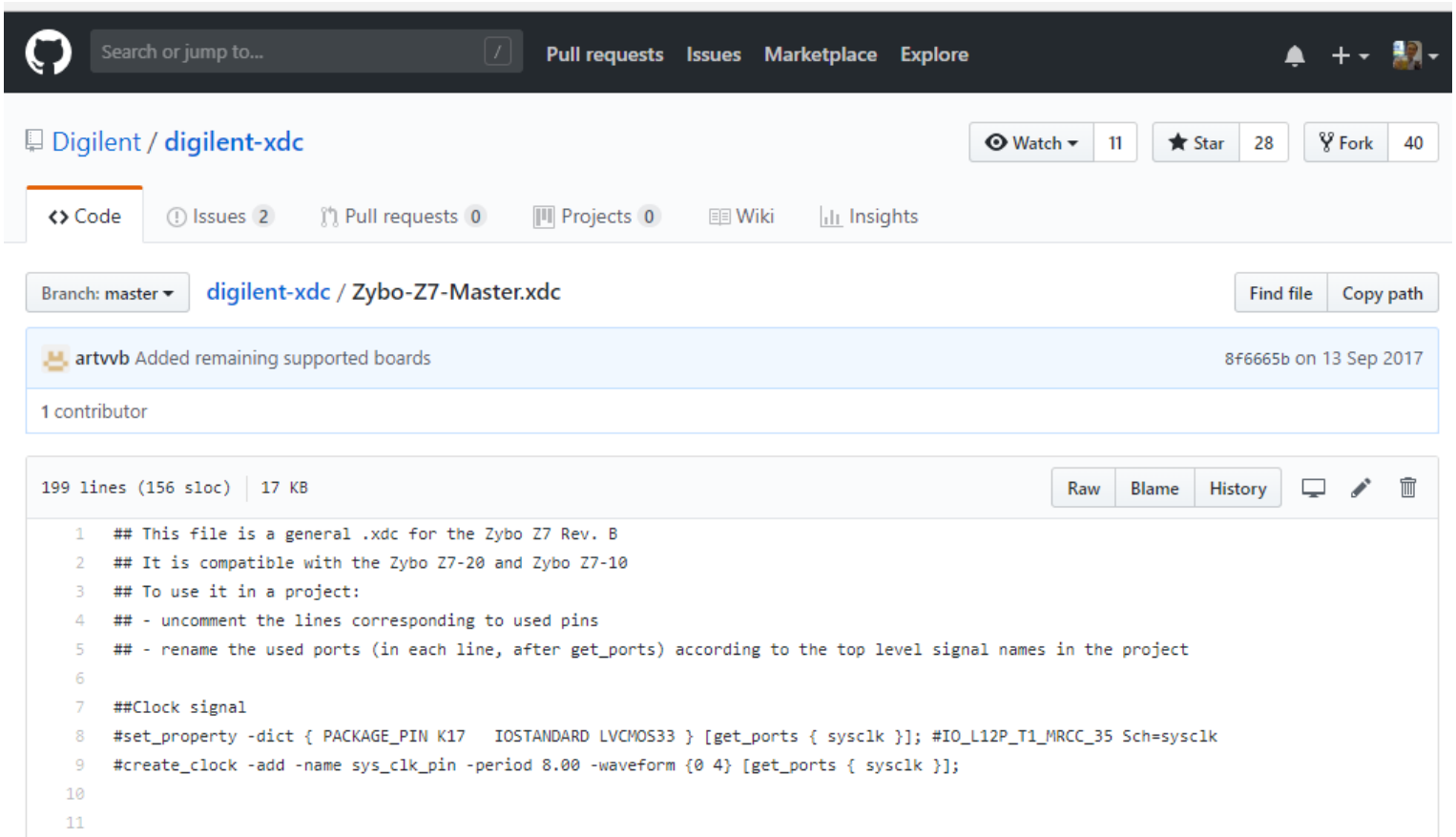
Assignment of a combinational logic

```
    assign led[0] = btn[0] & btn[1];
    assign led[1] = btn[0] | btn[1];
    assign led[2] = ~btn[0];
    assign led[3] = btn[1] & (btn[2] | btn[3]);
endmodule
```

Specify Constraint

- Pins (Location), and its direction(Input/Output), scandalization (LVDS, LVCMOS33,...)
- Download Digilent "Zybo-Z7-Master.xdc" from GitHub

https://github.com/Digilent/digilent-xdc/blob/master/Zybo-Z7-Master.xdc

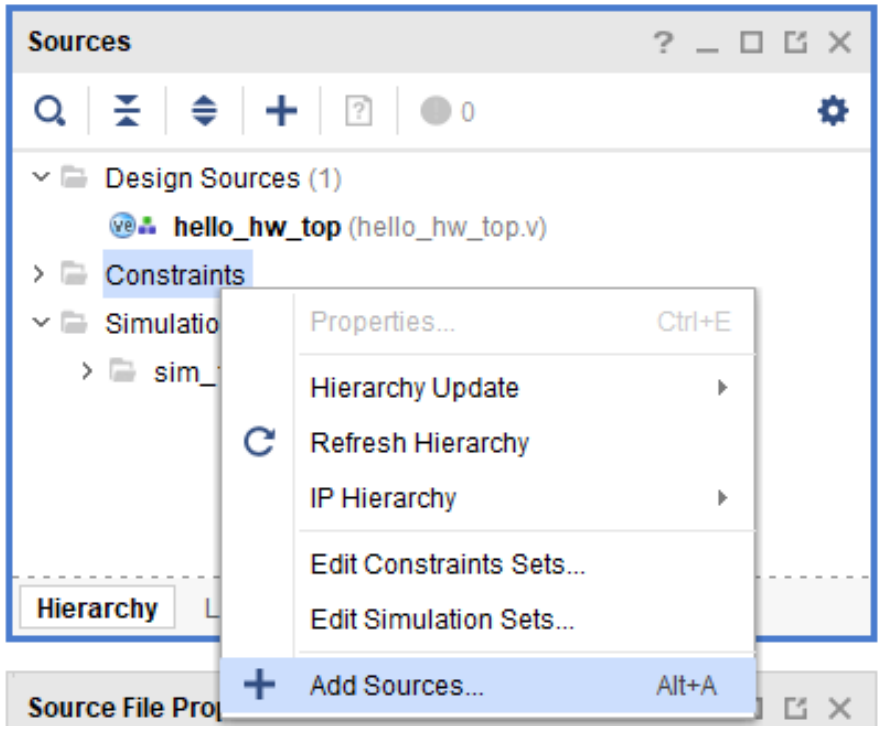


Add Constraint File

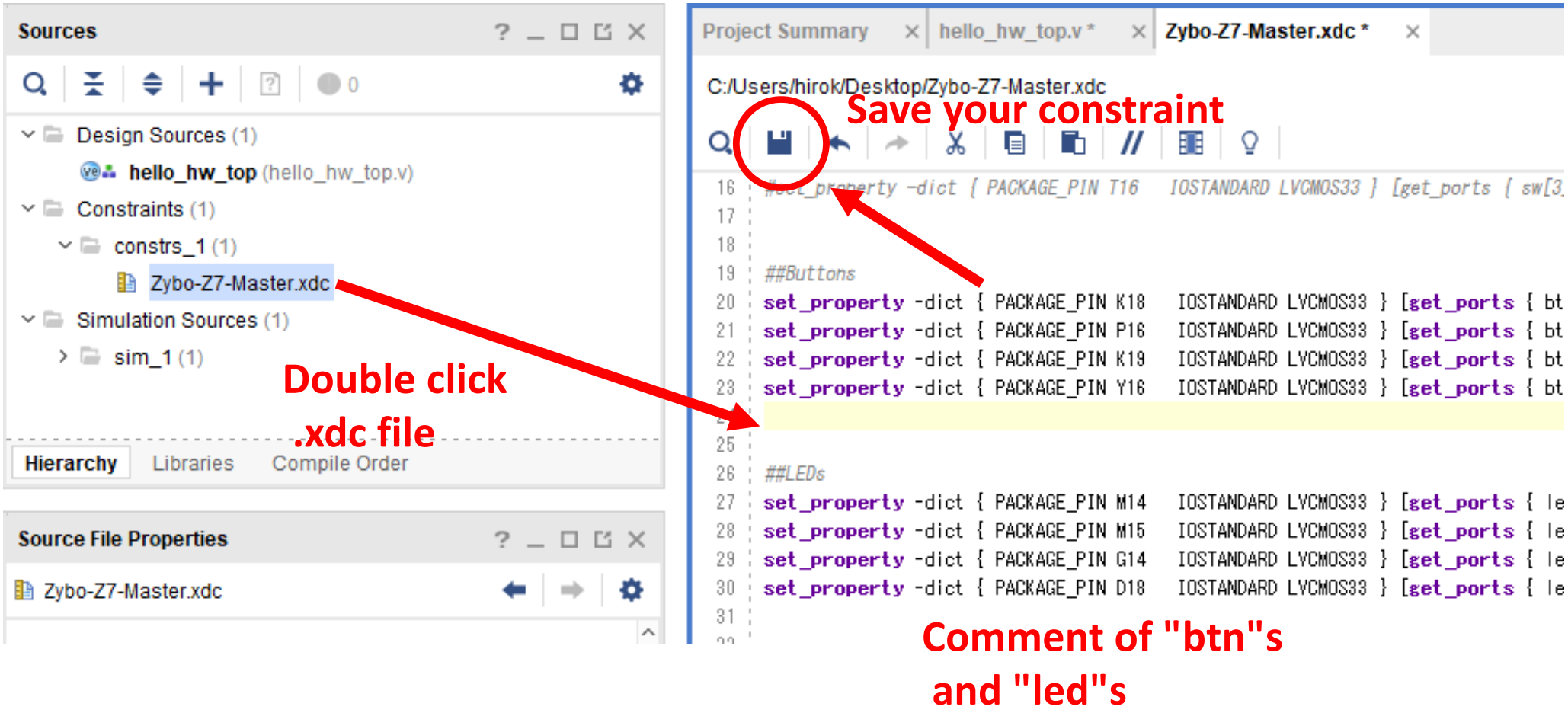
Right click on "Constraints", and select "Add Sources..."

Make sure "Add or create constraints", then "Next"

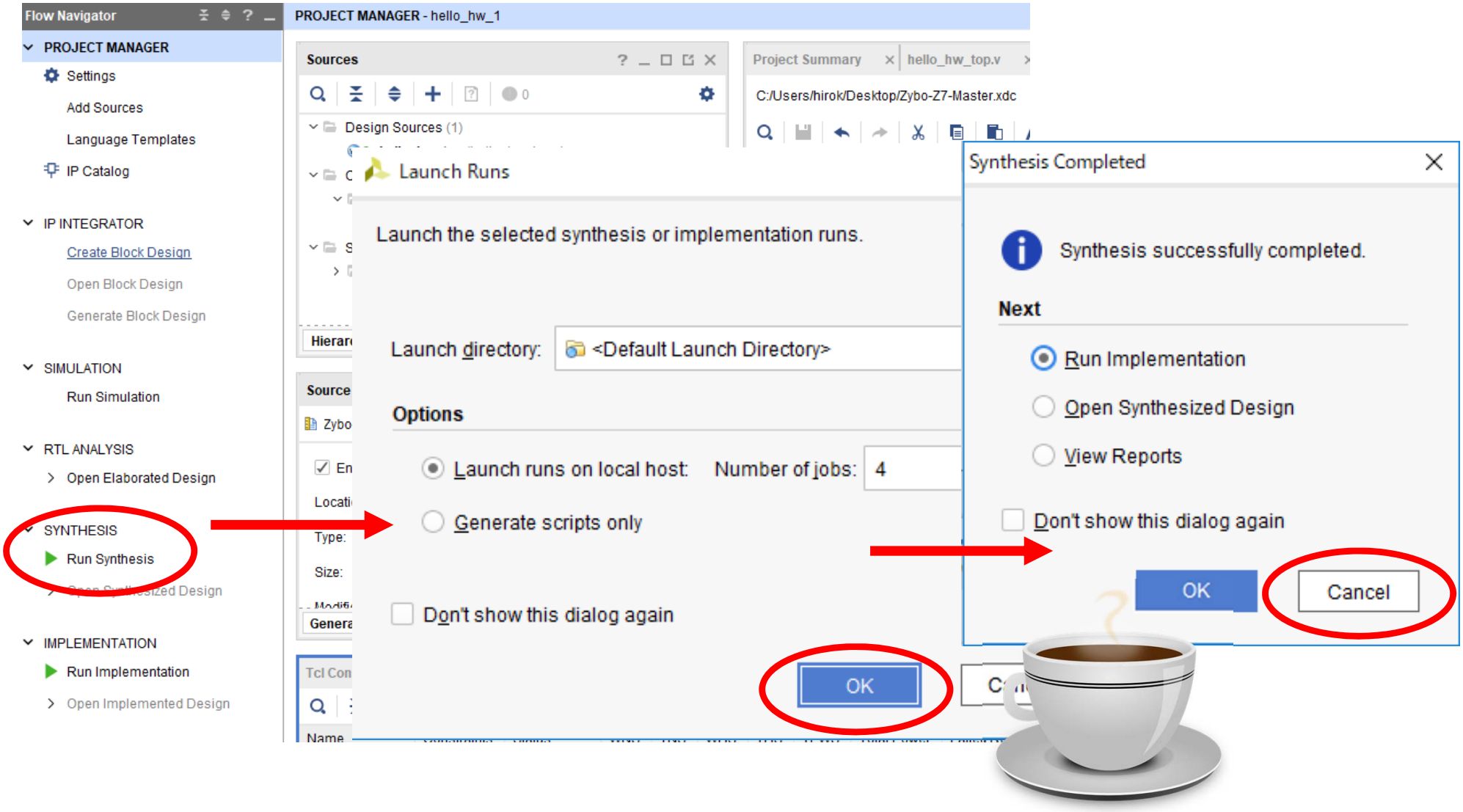
In "Add or Create Constraints", click "Add Files", then load "Zybo-Z7-Master.xdc" from Digilent's GitHub repository, and "Finish"



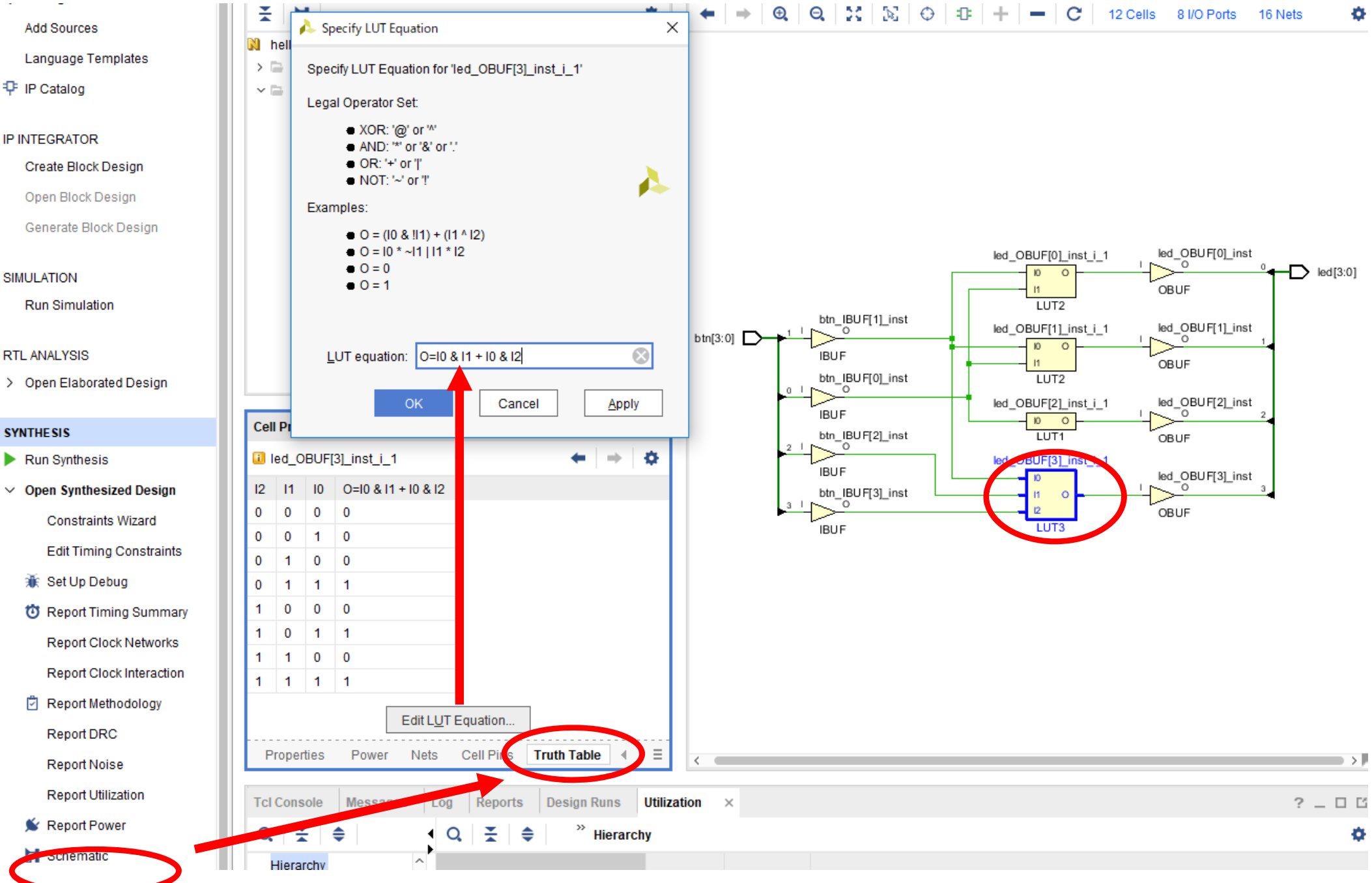
Modify Constraint File

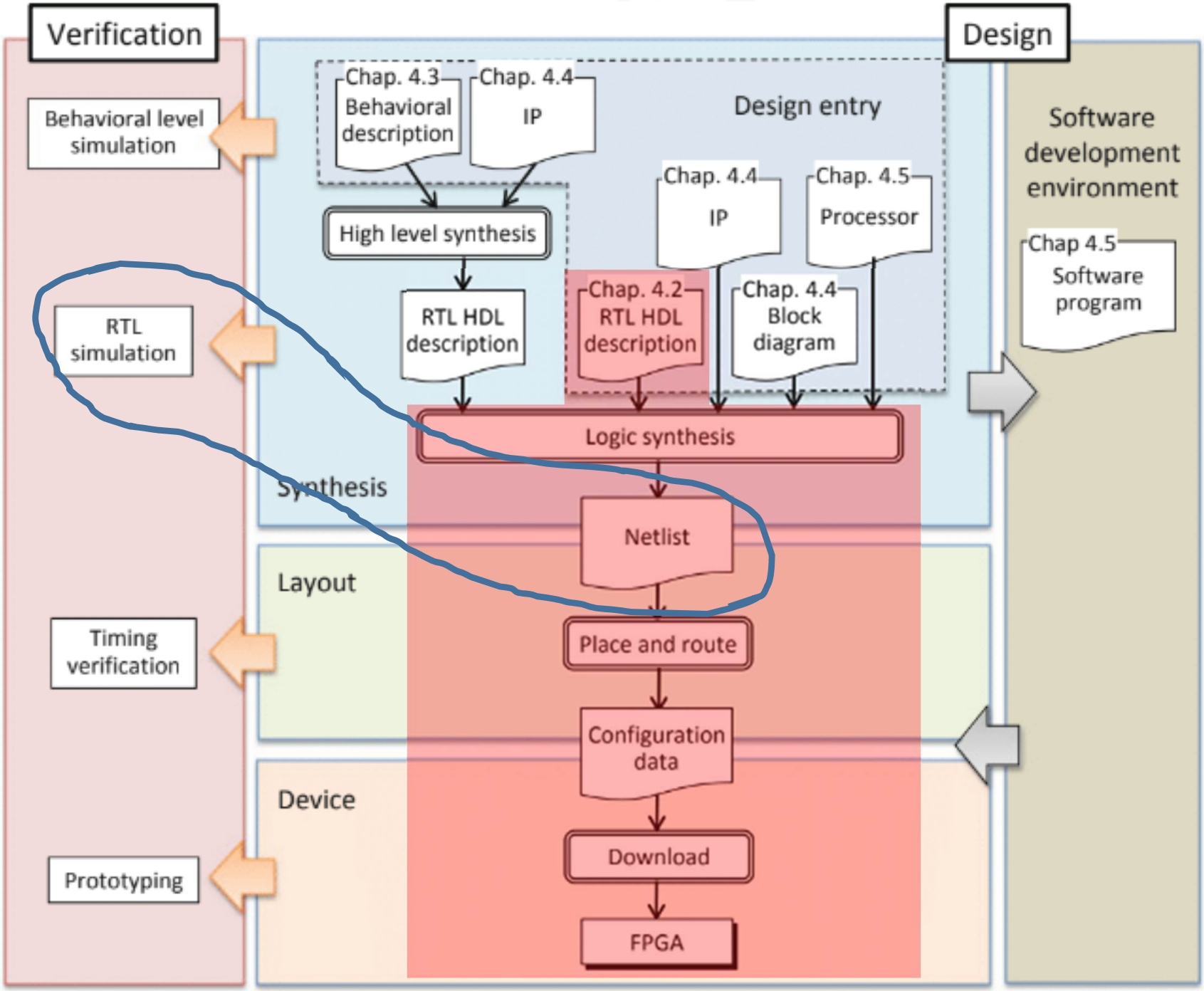


Do Logic Synthesis



Schematic View




Verification
Design
Behavioral level simulation
RTL simulation
Timing verification
Prototyping
Chap. 4.3
Behavioral description
Chap. 4.4
IP
Design entry
High level synthesis
RTL HDL description
Chap. 4.4
IP
Chap. 4.5
Processor
Chap. 4.2
RTL HDL description
Chap. 4.4
Block diagram
Logic synthesis
Synthesis
Netlist
Layout
Place and route
Configuration data
Device
Download
FPGA
Software development environment
Chap 4.5
Software program

RTL Simulation

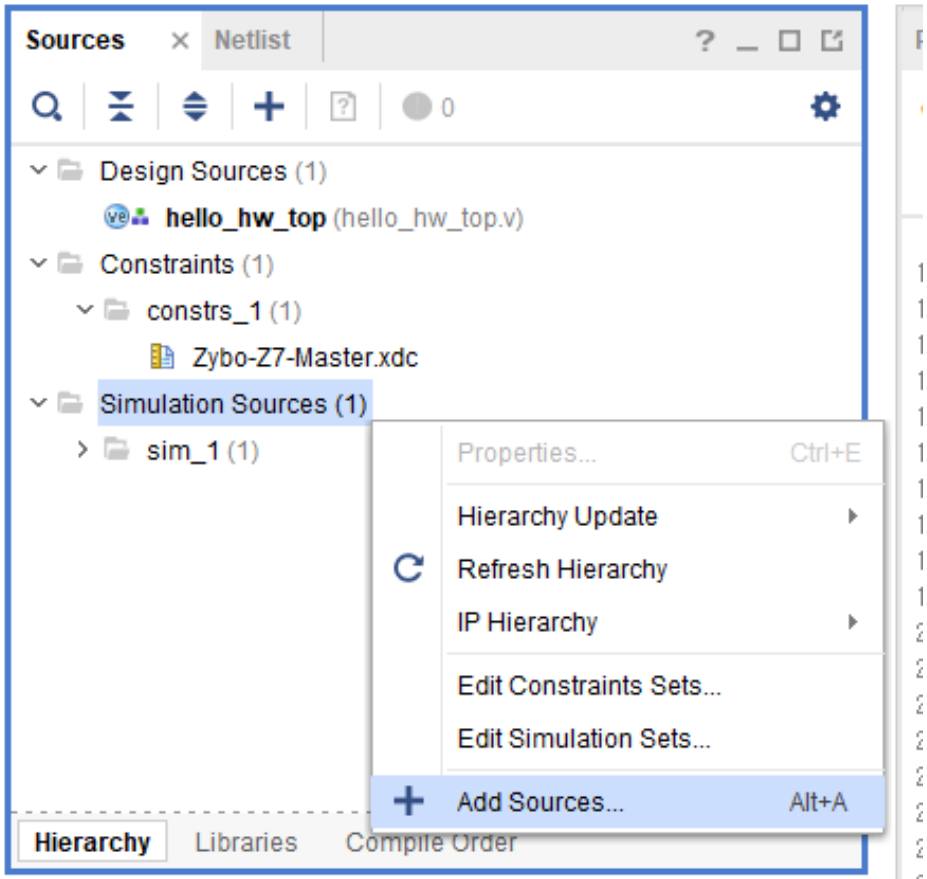
- Confirm the designed circuit meets the required specifications (or not)
- Simulation model
  - Behavior of RTL description
    - Verify correctness of functions and operations
  - Synthesized netlist
    - Timing and delay of signal change based on the delay time of the assigned logic/memory element, analyze the state of signal transition with propagation delay
    - Allows analysis of power consumption
  - Post placement and routing netlist
    - Estimate wiring delay time and enable the most detailed timing /power analysis

Set Simulation Source

Make sure "Sources tab" is selected

Right click on "Simulation Sources", then select "Add Sources..."

In "Add Sources", confirm "Add or create simulation sources", then "Next"

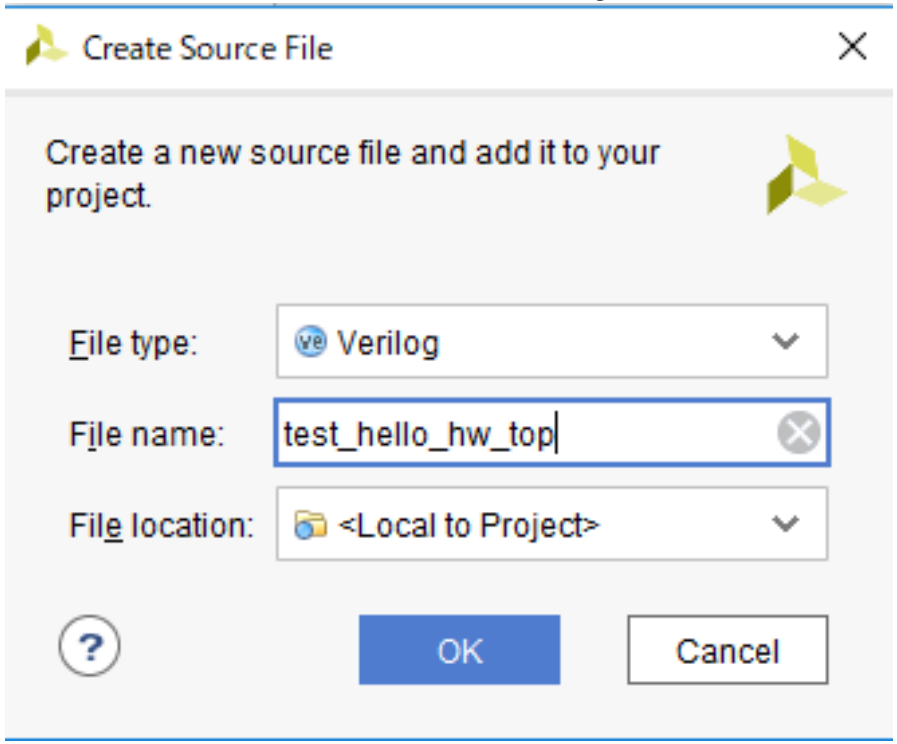


Cont'd

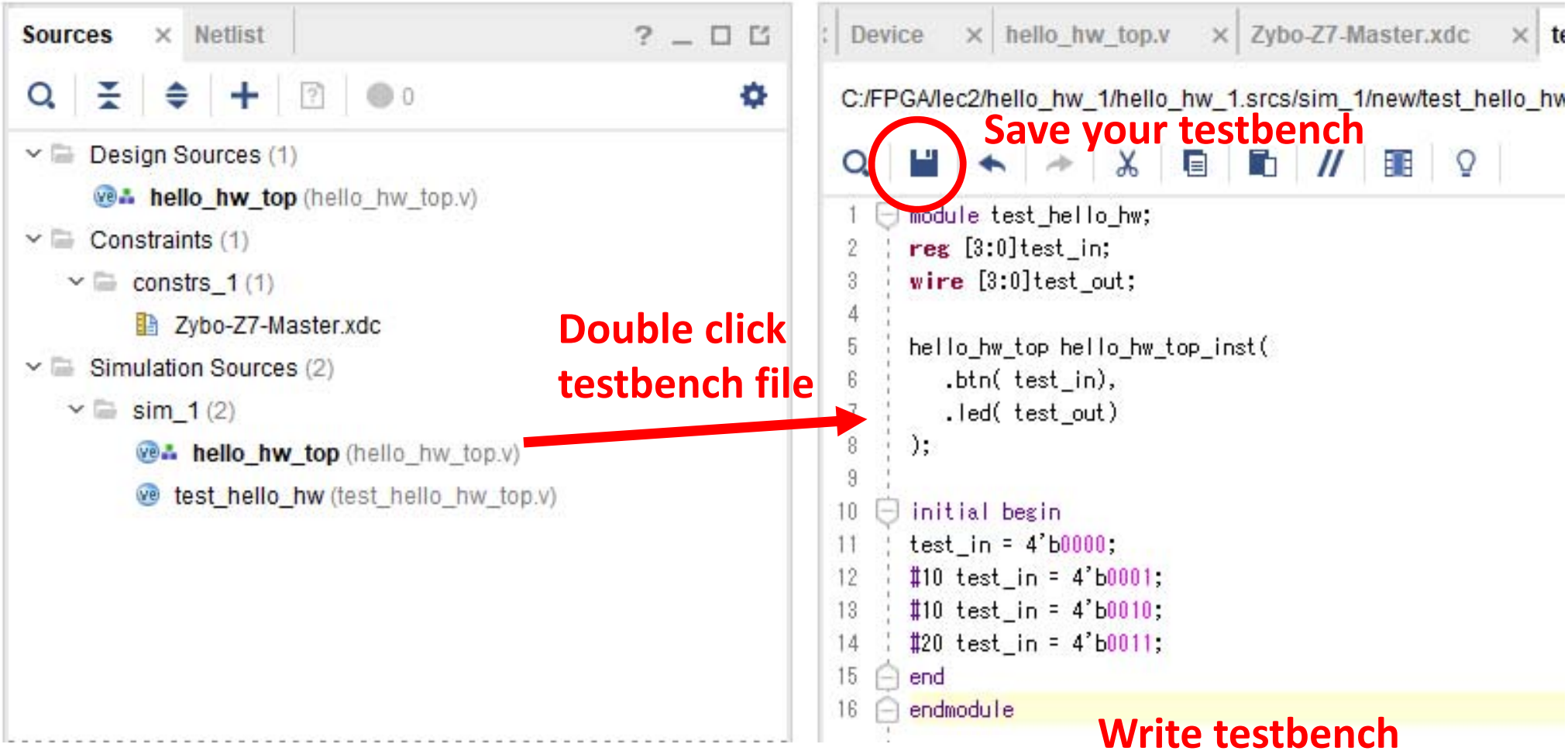
In "Add or Create Simulation Sources", click "Create File"

In "Create Source File", enter "test_hello_hw_top" in "File name", then "OK", and "Finish"

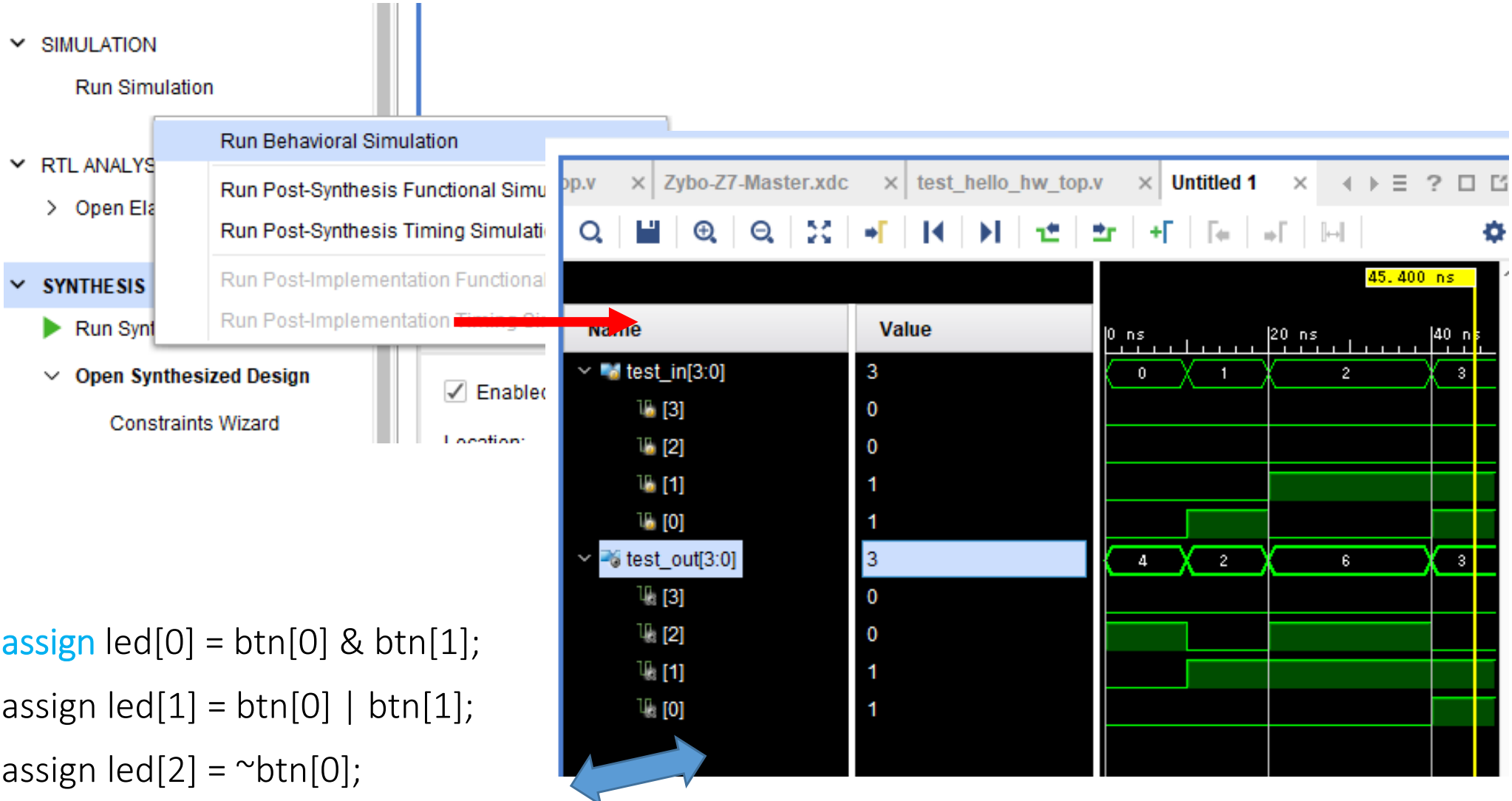
In Define module, just "OK"



Write TestBench



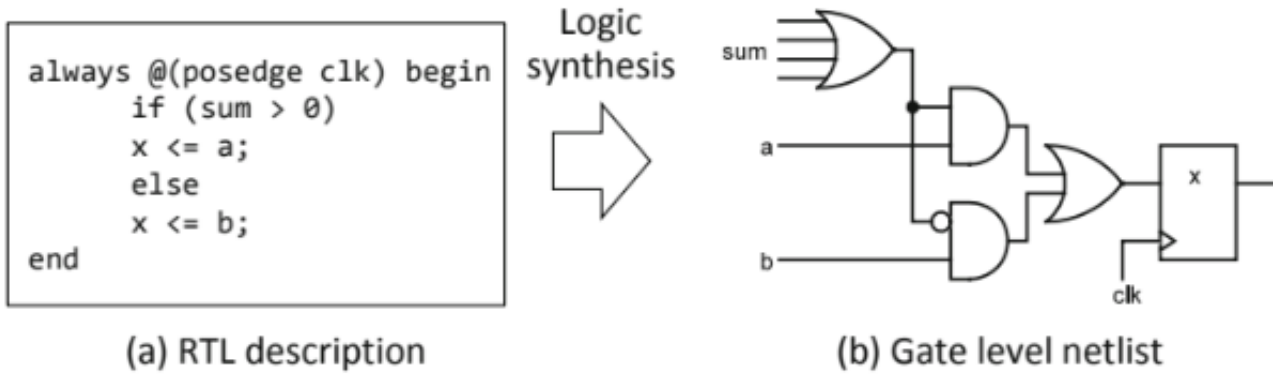
Run Behavioral Simulation



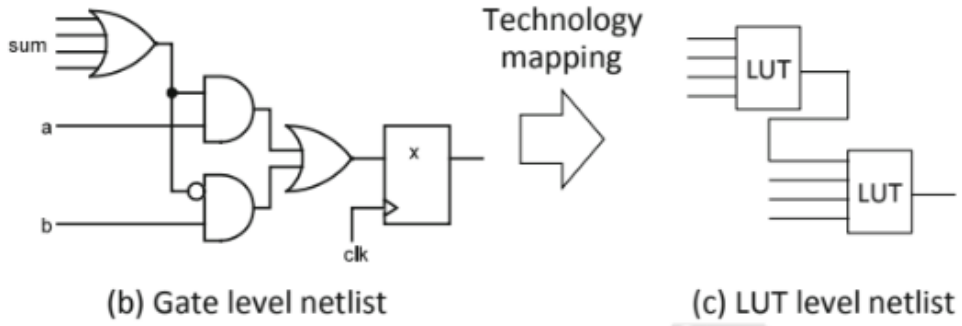

```
assign led[0] = btn[0] & btn[1];
assign led[1] = btn[0] | btn[1];
assign led[2] = ~btn[0];
assign led[3] = btn[1] & (btn[2] | btn[3]);
```

Verify your description with simulation result

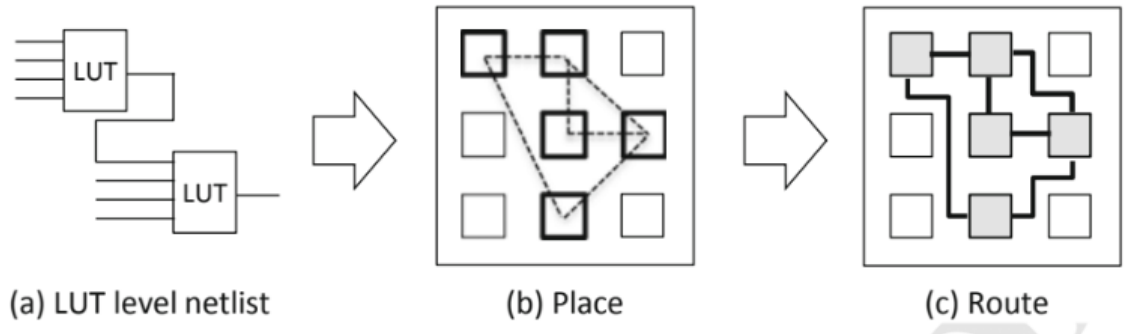
Simulation Level



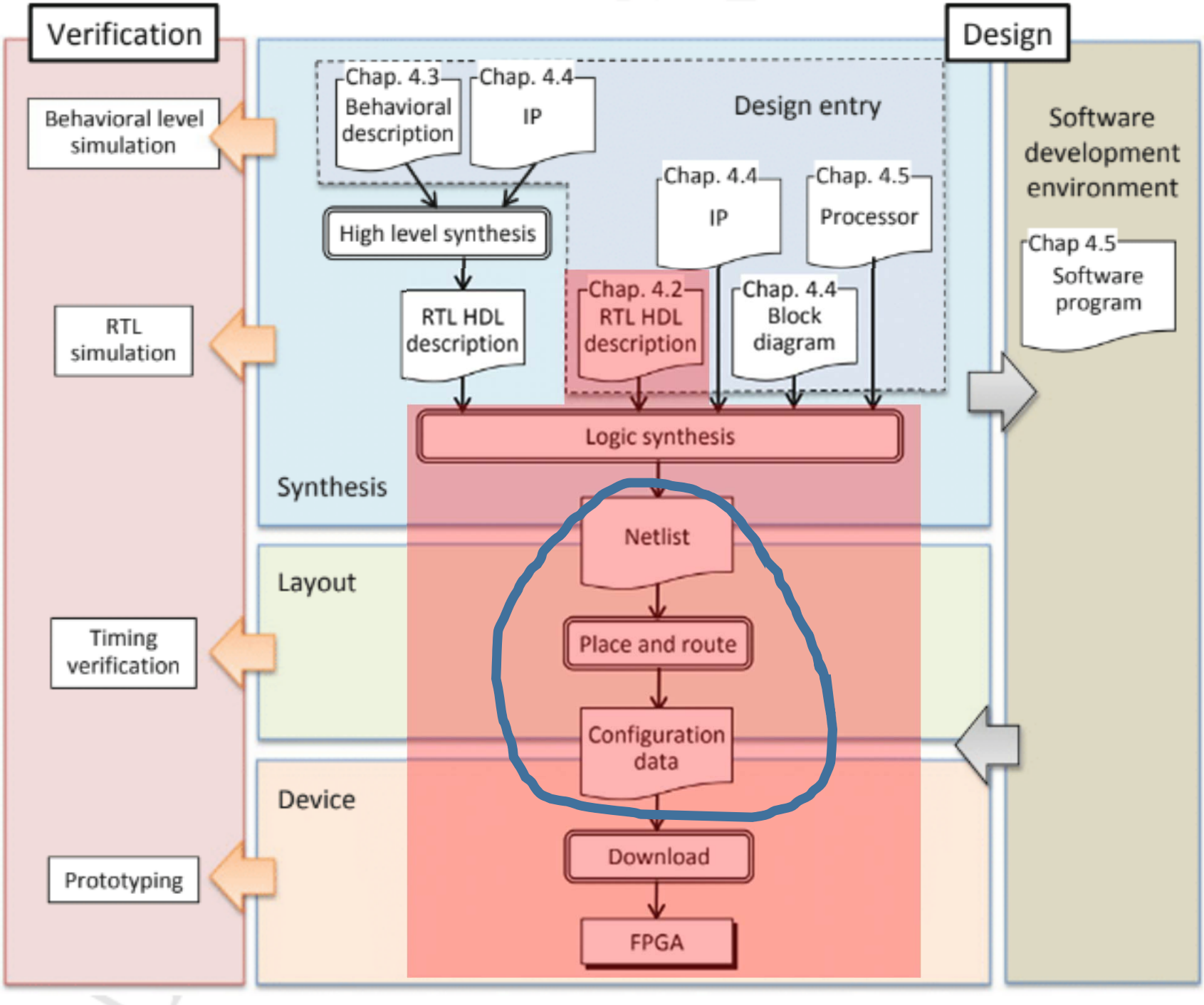
Behavioral Simulation:
Sim. Speed is faster,
but, low accurate



Post-Synthesis:
Sim. Speed is middle,
but, relatively accurate

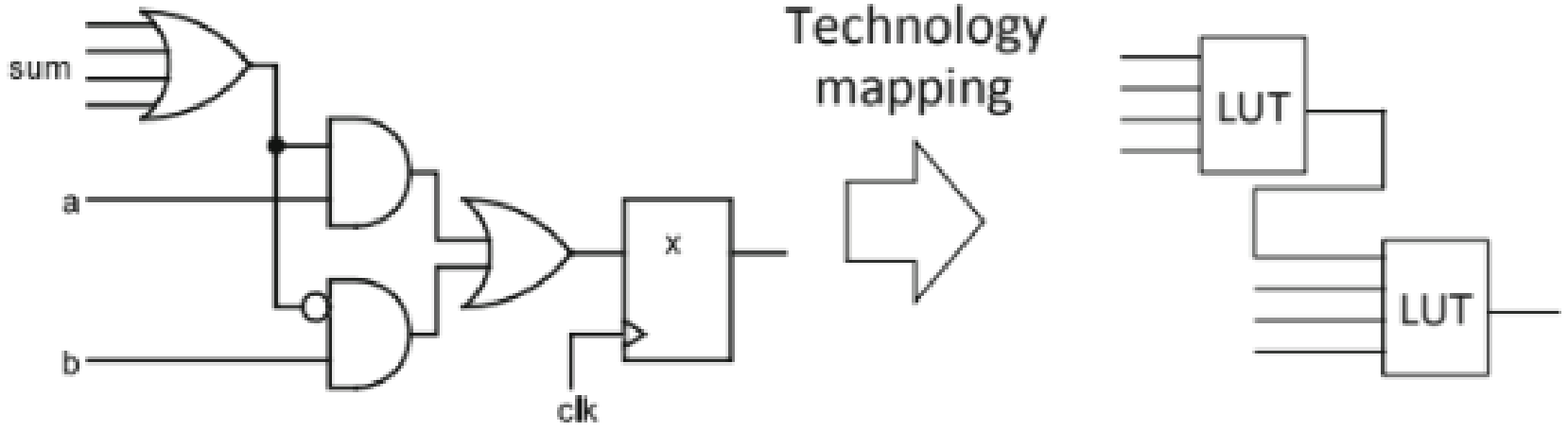


Post-Implementation:
Sim. Speed is slower,
but, more accurate


Verification
Design
Behavioral level simulation
RTL simulation
Timing verification
Prototyping
Chap. 4.3
Behavioral description
Chap. 4.4
IP
Design entry
High level synthesis
RTL HDL description
Chap. 4.4
IP
Chap. 4.5
Processor
Chap. 4.2
RTL HDL description
Chap. 4.4
Block diagram
Logic synthesis
Synthesis
Netlist
Layout
Place and route
Configuration data
Device
Download
FPGA
Software development environment
Chap 4.5
Software program

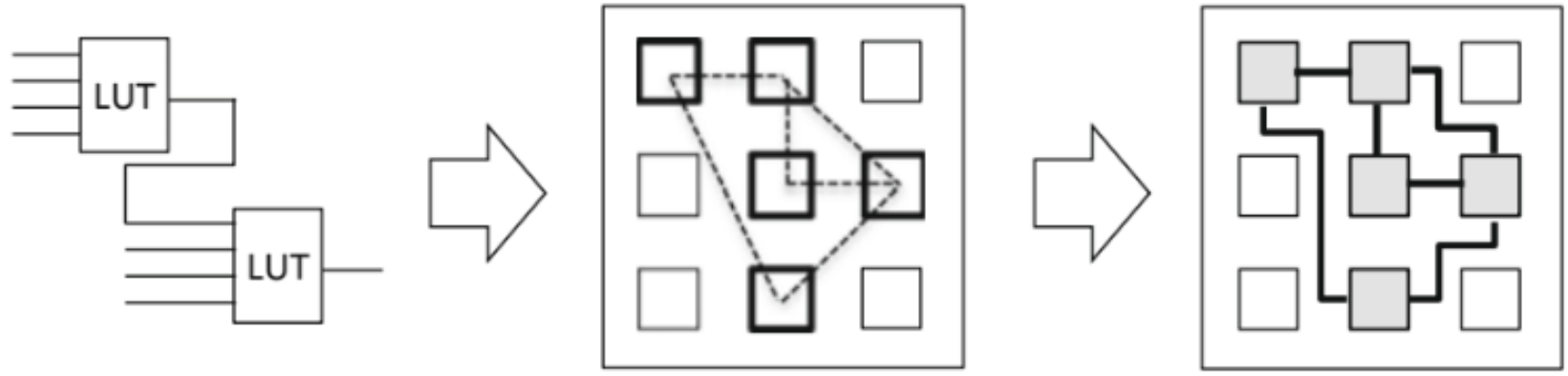
Technology Mapping

- Assign the synthesized netlist to the logic elements of the FPGA
- Rewritable logic elements called LUT(Look-Up Table) are used

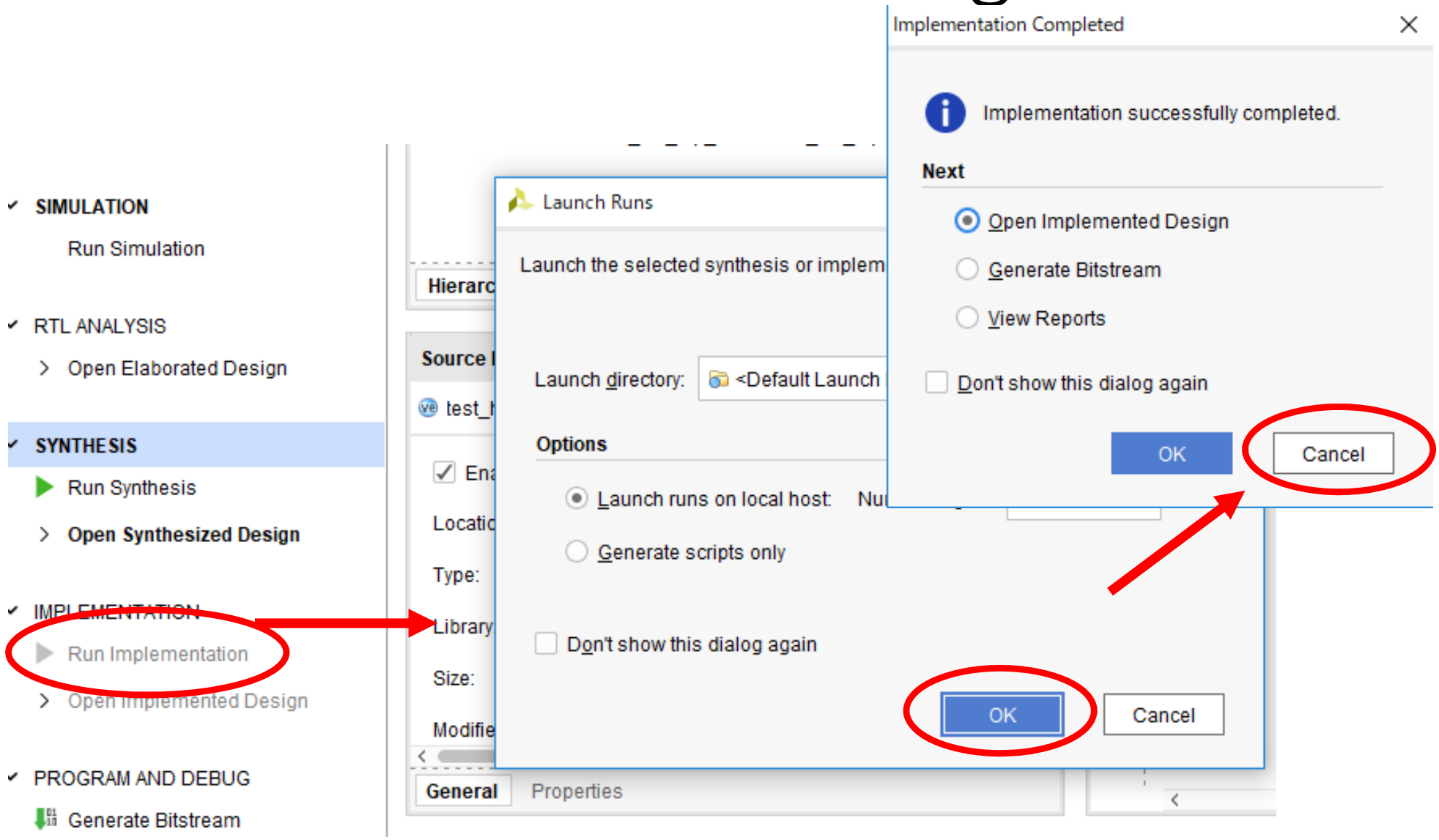


Place-and-Routing

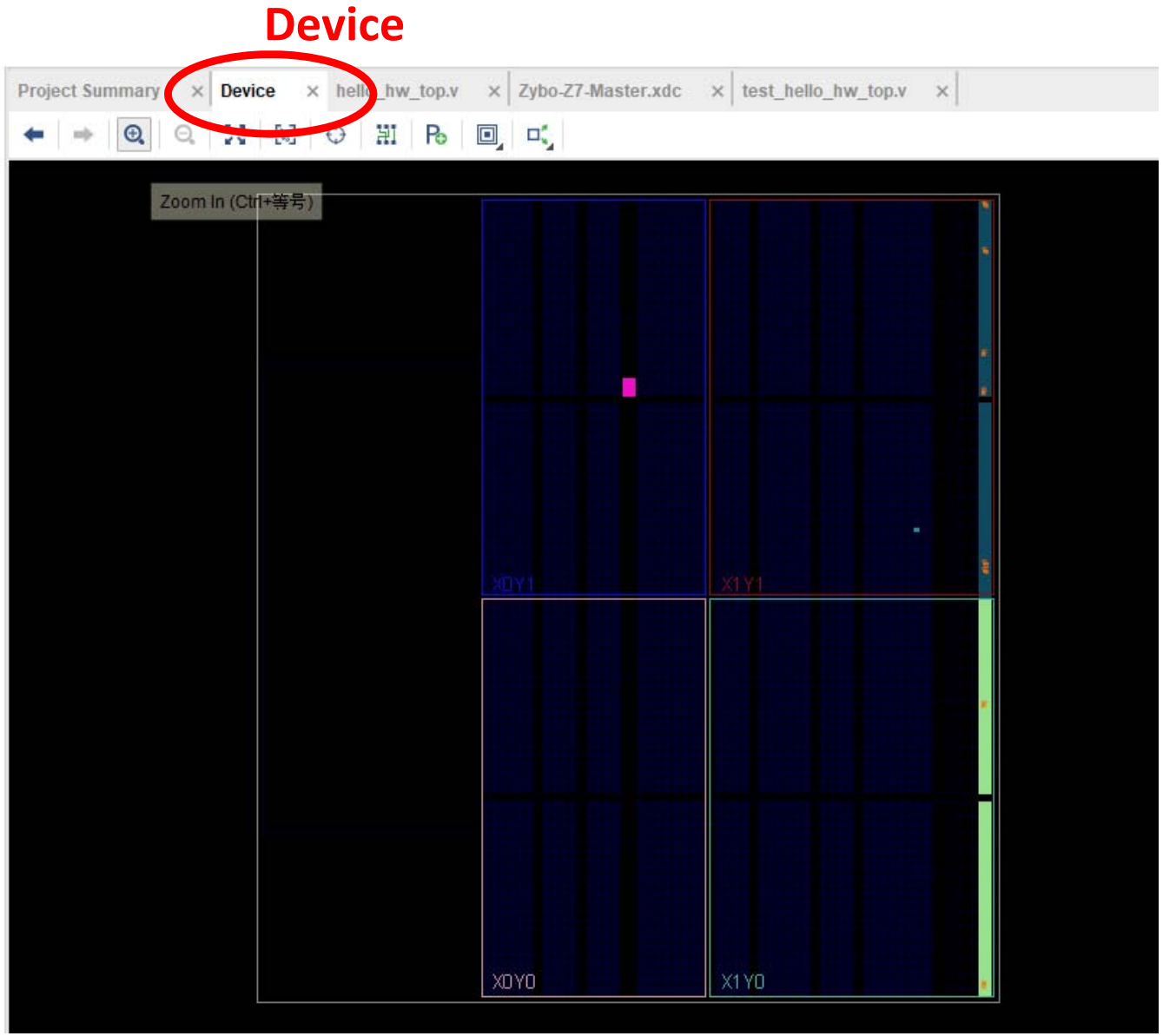
- Assign LUT-based netlist to logical resources and routing resources on FPGA
- Generaly, after allocate logical resources and then perform routing



Launch Place-and-Routing

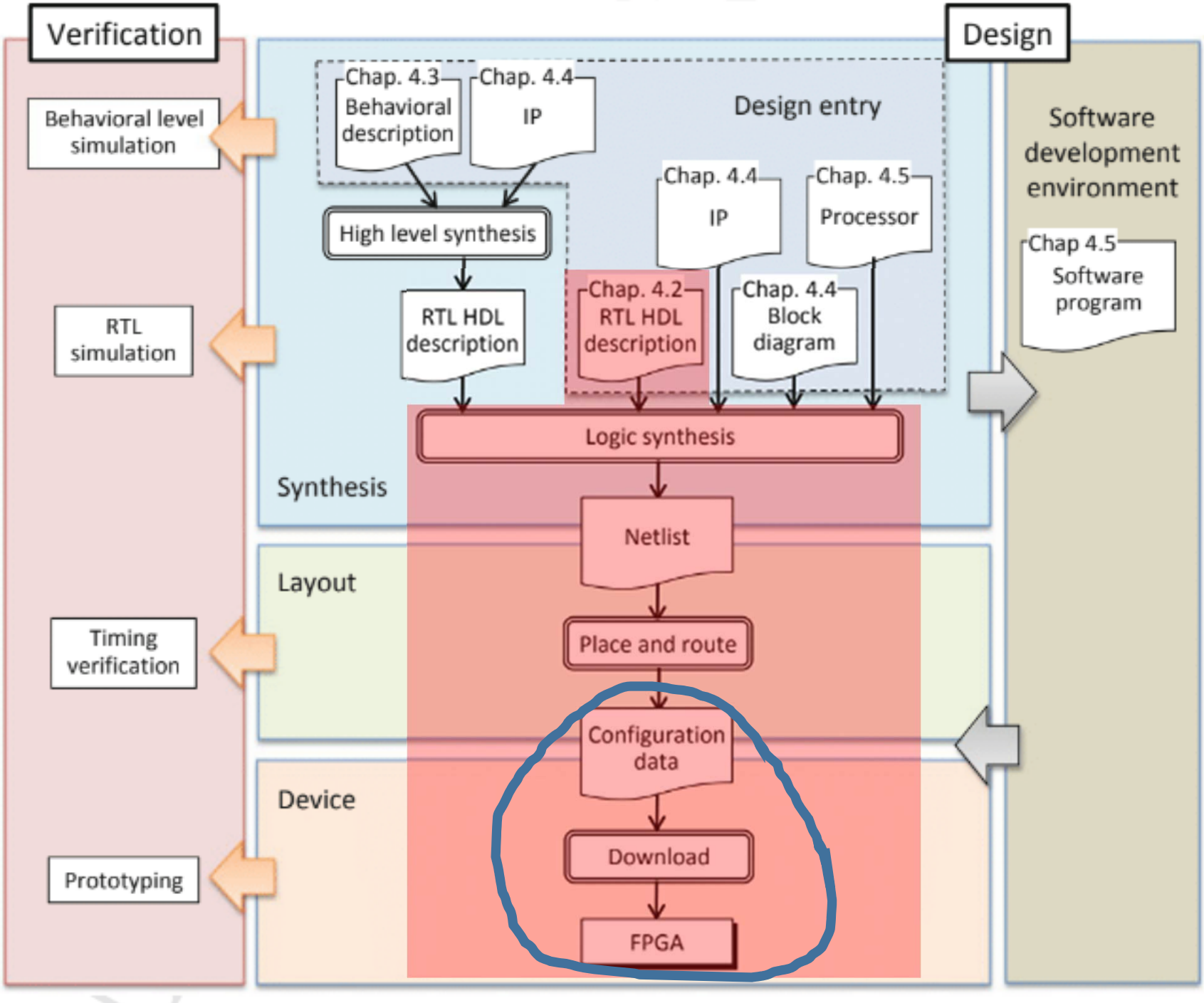


Investigate FPGA Architecture



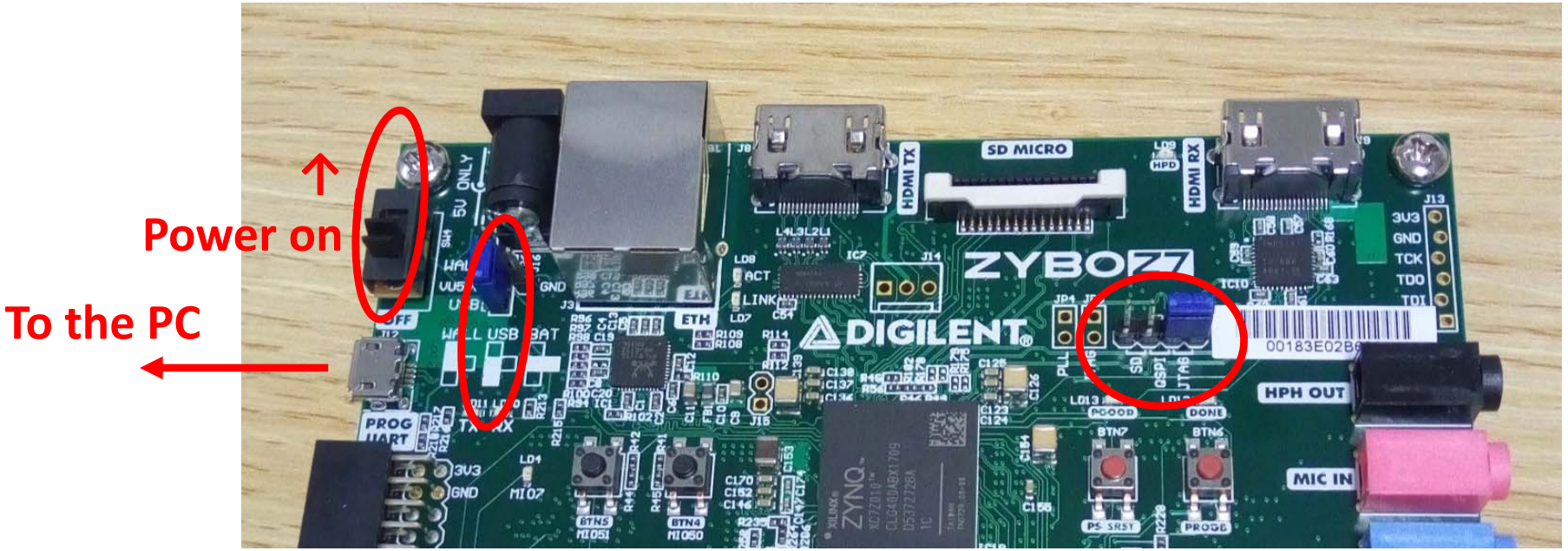
Programming

- The completed circuit is converted as bitstream (configuration data or program file) for programming the logical and wiring resources in the FPGA
- Send bitstream to the FPGA using the programmer
  - Direct writing by Joint Test Action Group (JTAG) to program nonvolatile memory (Flash, EEPROM)
  - Circuit configuration is erased due to power off or reset of FPGA
- After writing, the board can be started as a bootable unit

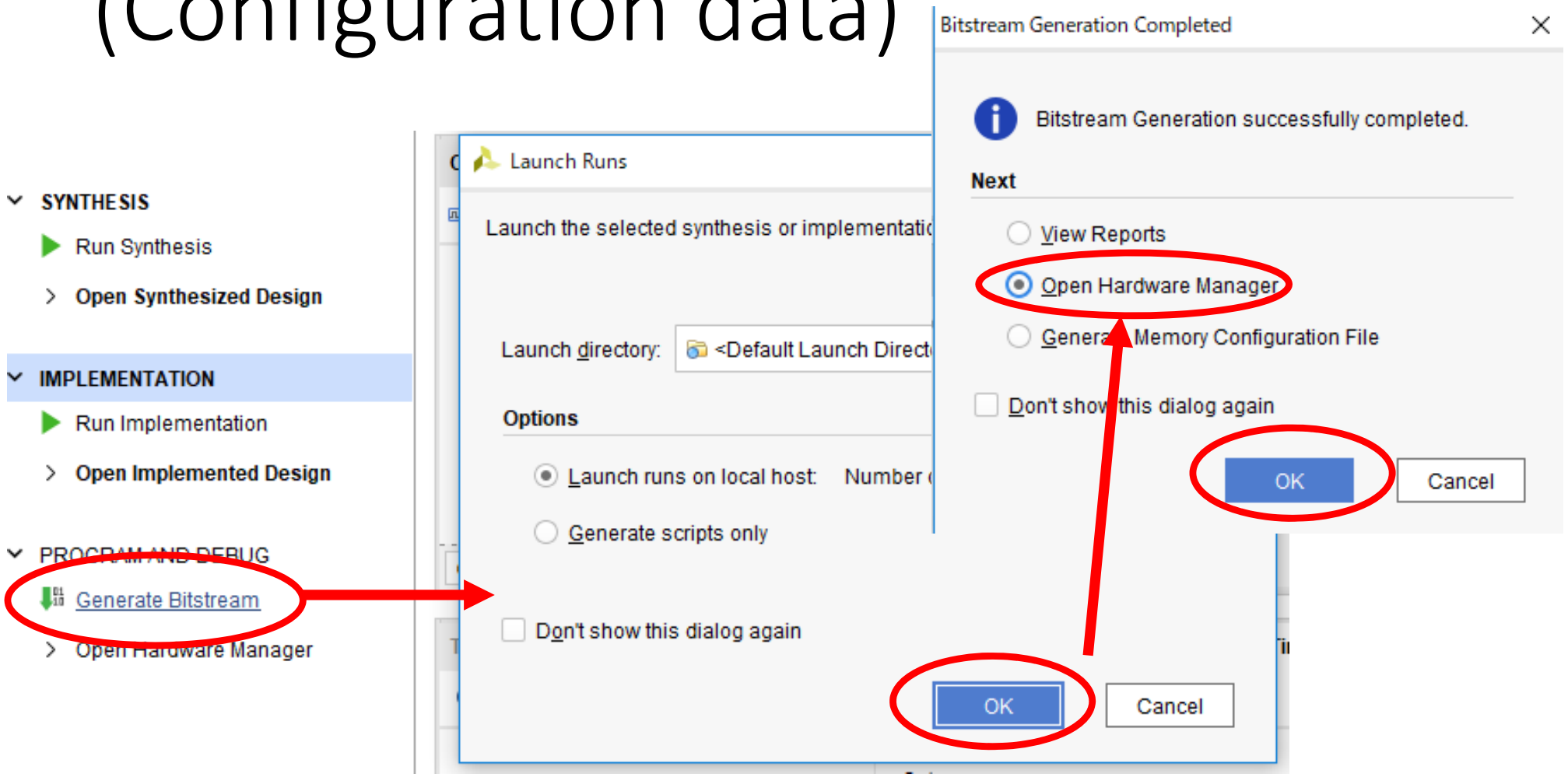

Verification
Design
Behavioral level simulation
RTL simulation
Timing verification
Prototyping
Chap. 4.3
Behavioral description
Chap. 4.4
IP
Design entry
High level synthesis
RTL HDL description
Chap. 4.4
IP
Chap. 4.5
Processor
Chap. 4.2
RTL HDL description
Chap. 4.4
Block diagram
Logic synthesis
Synthesis
Netlist
Layout
Place and route
Configuration data
Device
Download
FPGA
Software development environment
Chap 4.5
Software program

Setup Zybo-Z7 Board

- Make sure the FPGA board is not connect to your PC
- Set jumper pins as shown in Photograph
  - USB-power and JTAG mode
- Connect to the PC via mini-USB cable
- Turn on your FPGA board

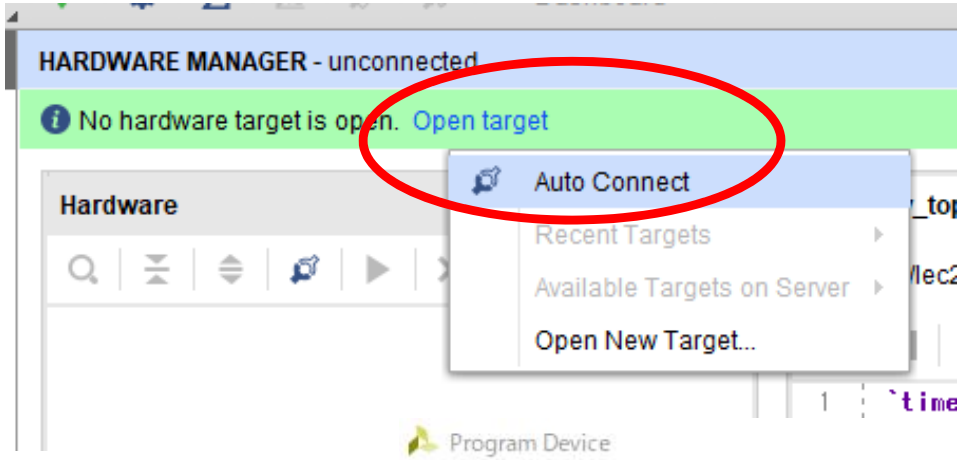


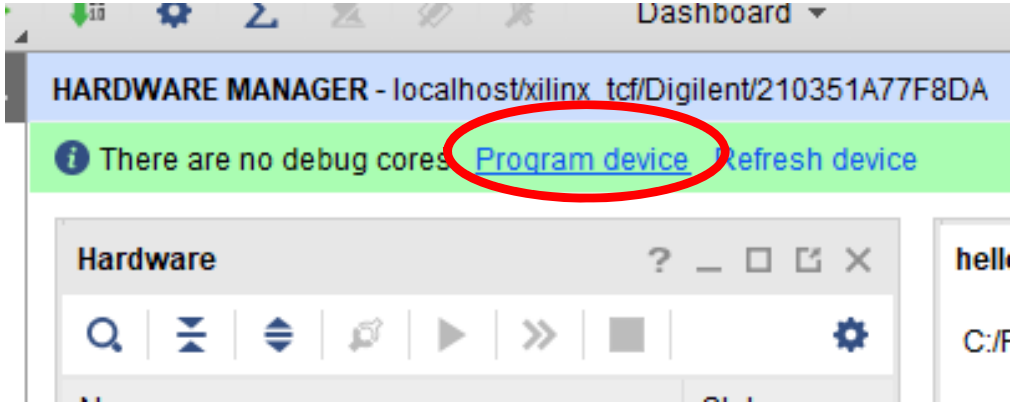
Generate Bitstream (Configuration data)

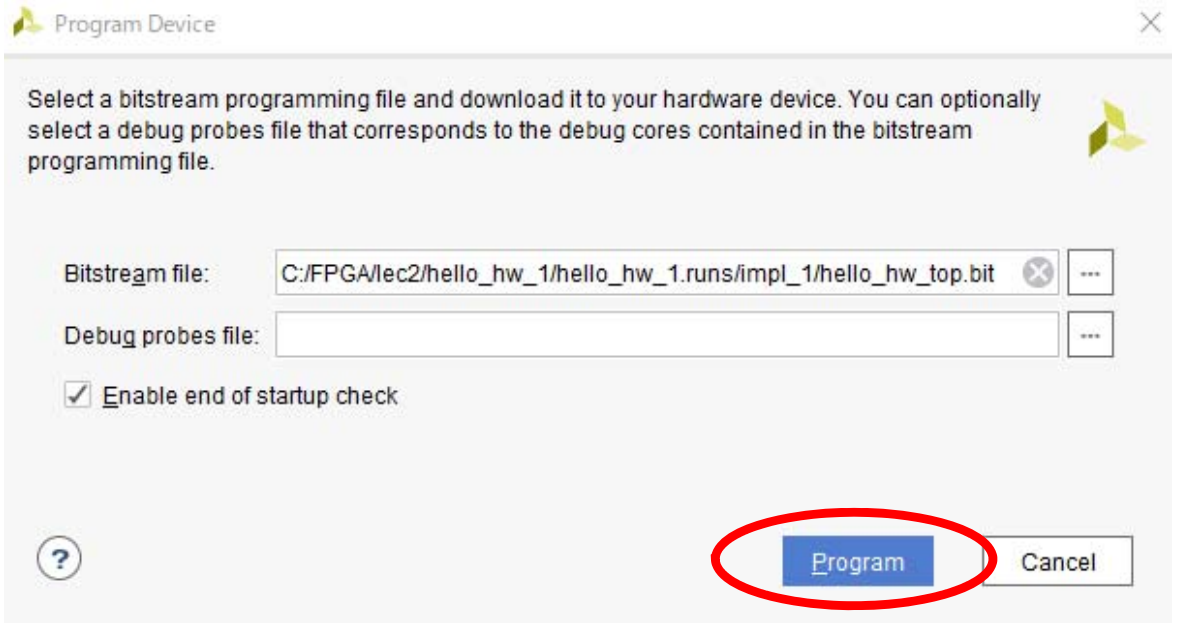


Programming

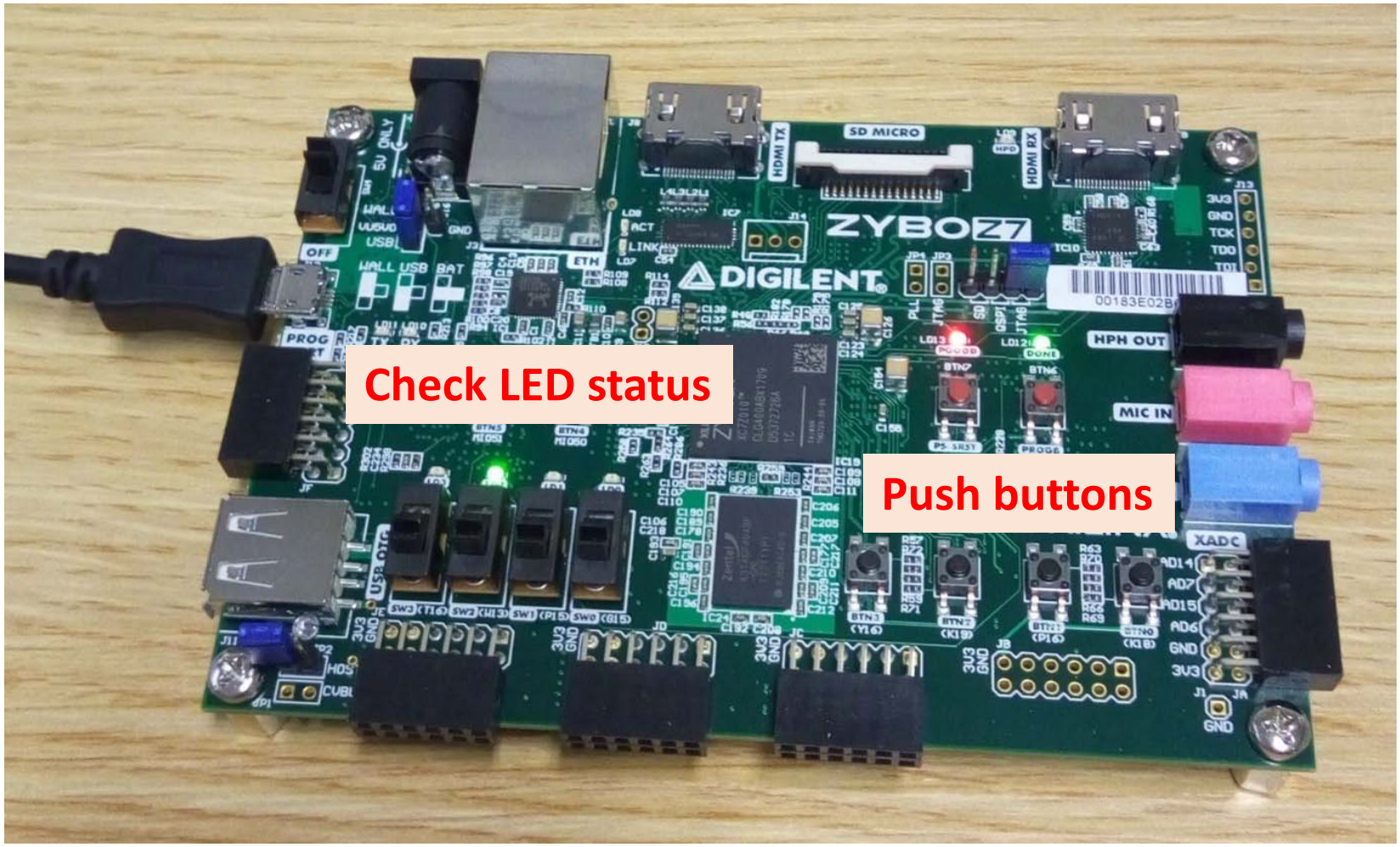
- Click "Open target", then "Auto Connect"
- Next, click "Program device", then "Program"



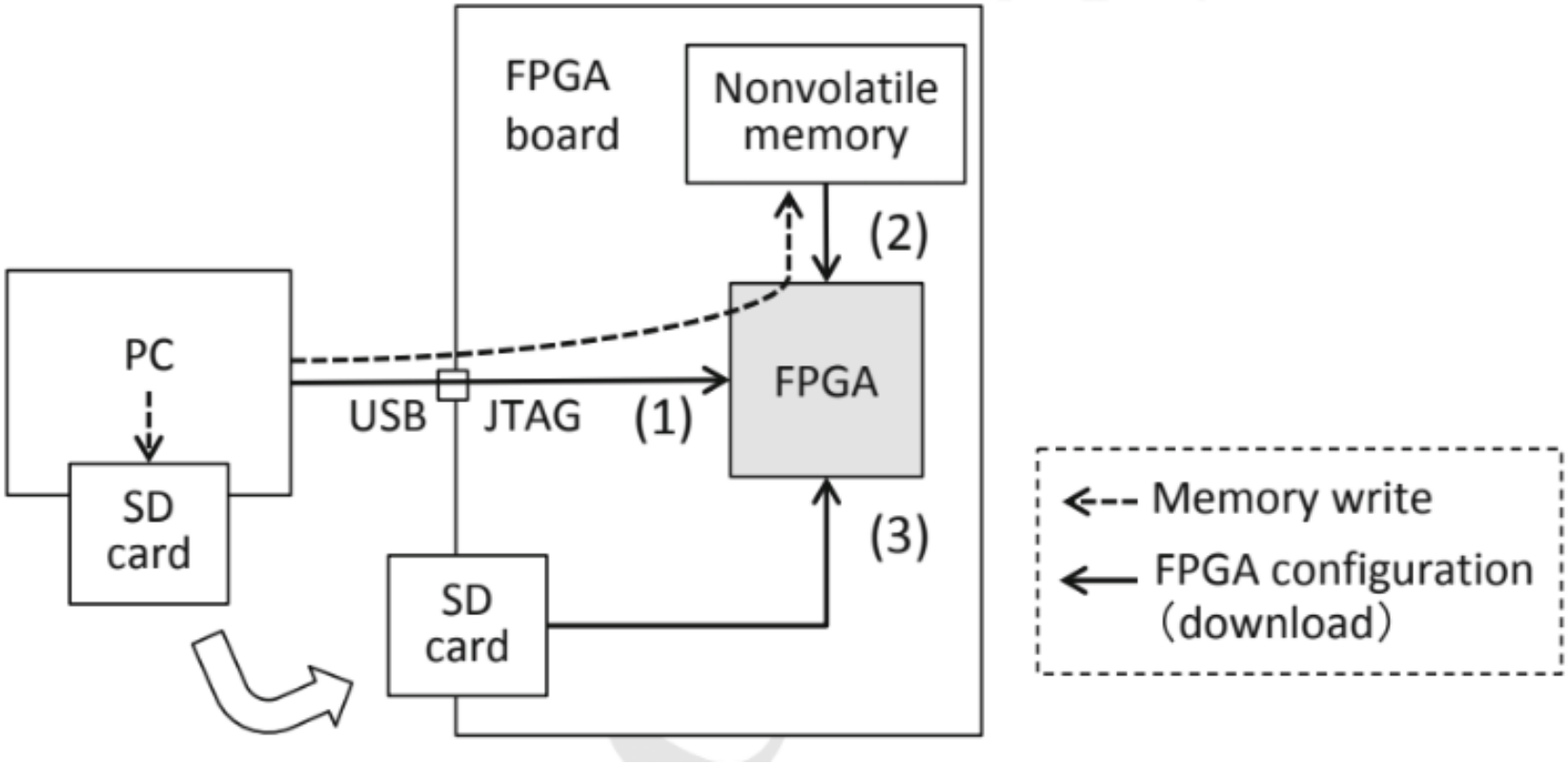




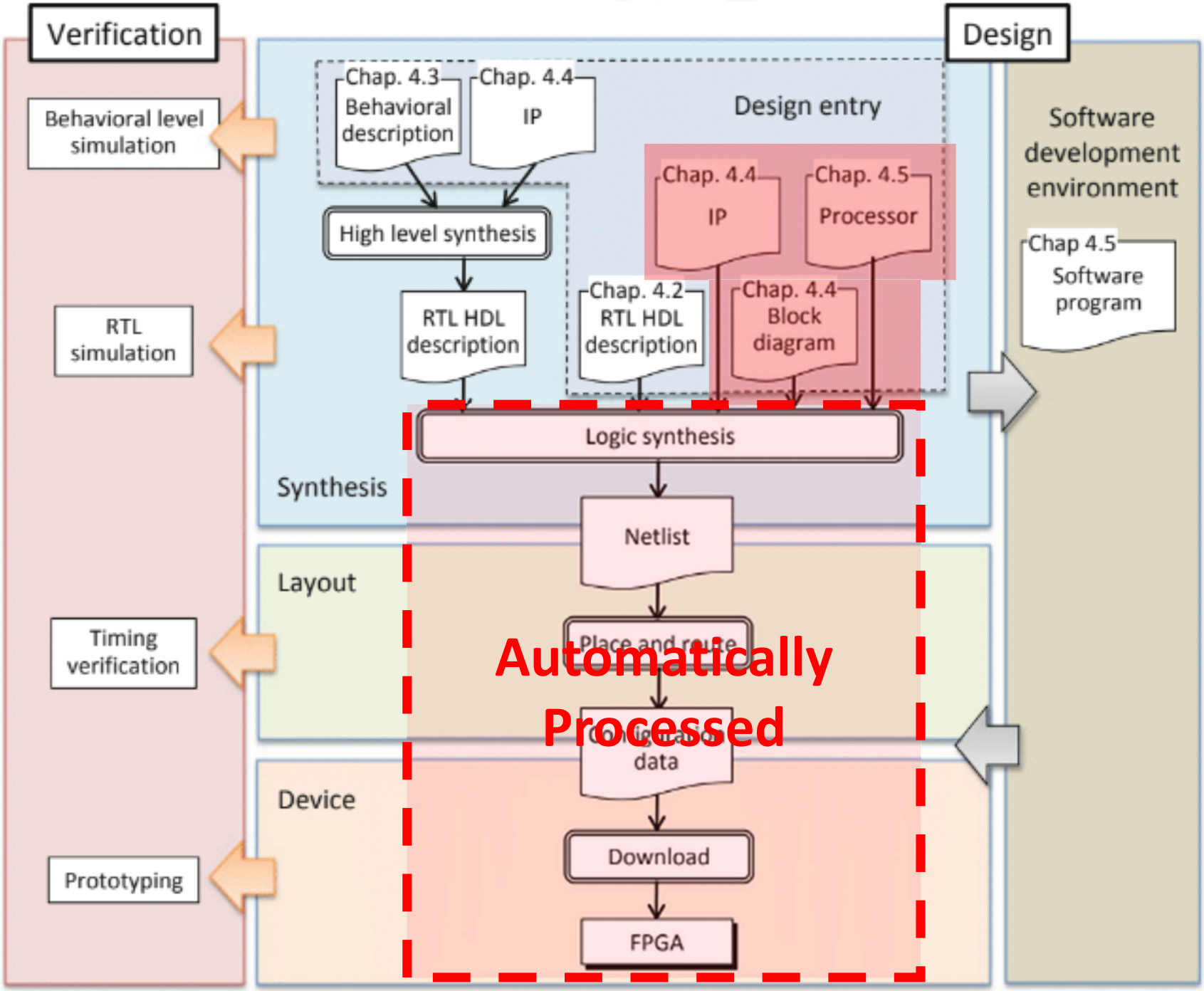
Welcome to HW world!



FPGA Configuration Methods



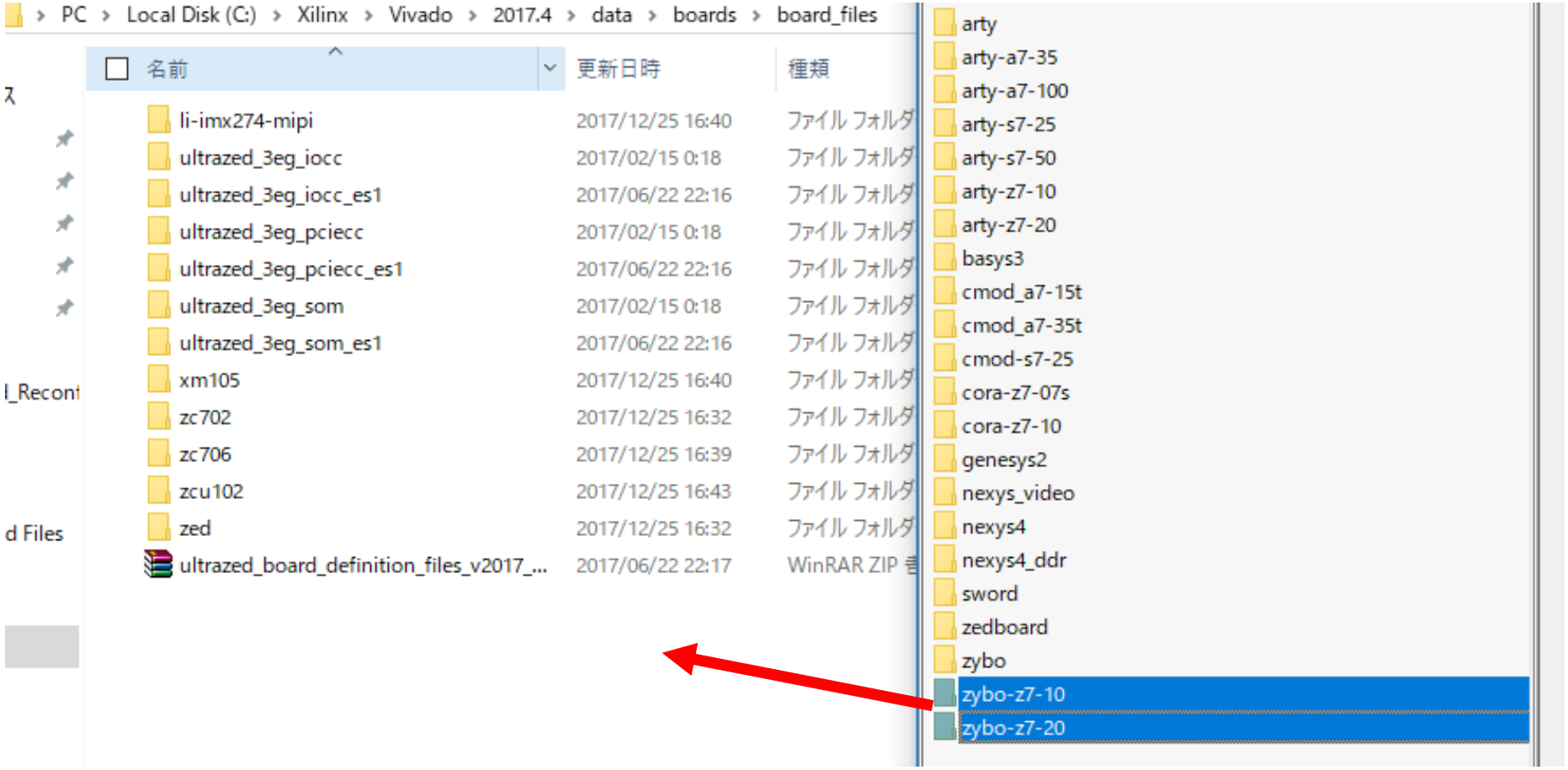
Processor Design Flow


Verification
Design
Behavioral level simulation
RTL simulation
Timing verification
Prototyping
Chap. 4.3
Behavioral description
Chap. 4.4
IP
Design entry
High level synthesis
RTL HDL description
Chap. 4.4
IP
Chap. 4.5
Processor
Chap. 4.2
RTL HDL description
Chap. 4.4
Block diagram
Logic synthesis
Synthesis
Netlist
Layout
Place and route
Configuration data
Device
Download
FPGA
Software development environment
Chap 4.5
Software program
Automatically Processed

Board File Definition File

- Constraint File & IP cores for specified board
- Zybo-Z7: https://github.com/Digilent/vivado-boards/archive/master.zip
- Unzip and store to C:/Xilinx/Vivado/2017.4/data/boards/board_files

(For Unix, it is located in "/opt/Xilinx/Vivado/2017.4/data/boards/board_files")



Hello World on Zybo-Z7

- Output of "Hello World" with UART of PS section and software on CPU
- Main part is PS section, but first we will make hardware at Vivado
- After that, we will write Hello World software on SDK

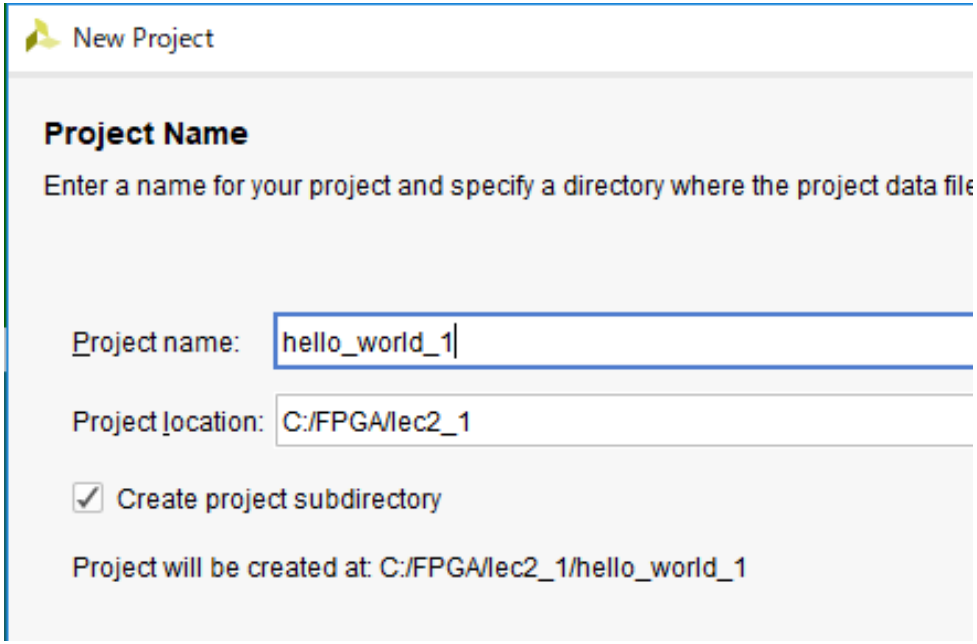
Create Project

Run Vivado 2017.4

File->New Project, then "Next"

Specify project name "hello_world_1" and its location is "C:/FPGA/lec2_1" (For Unix, it is "/root/FPGA/lec2_1")

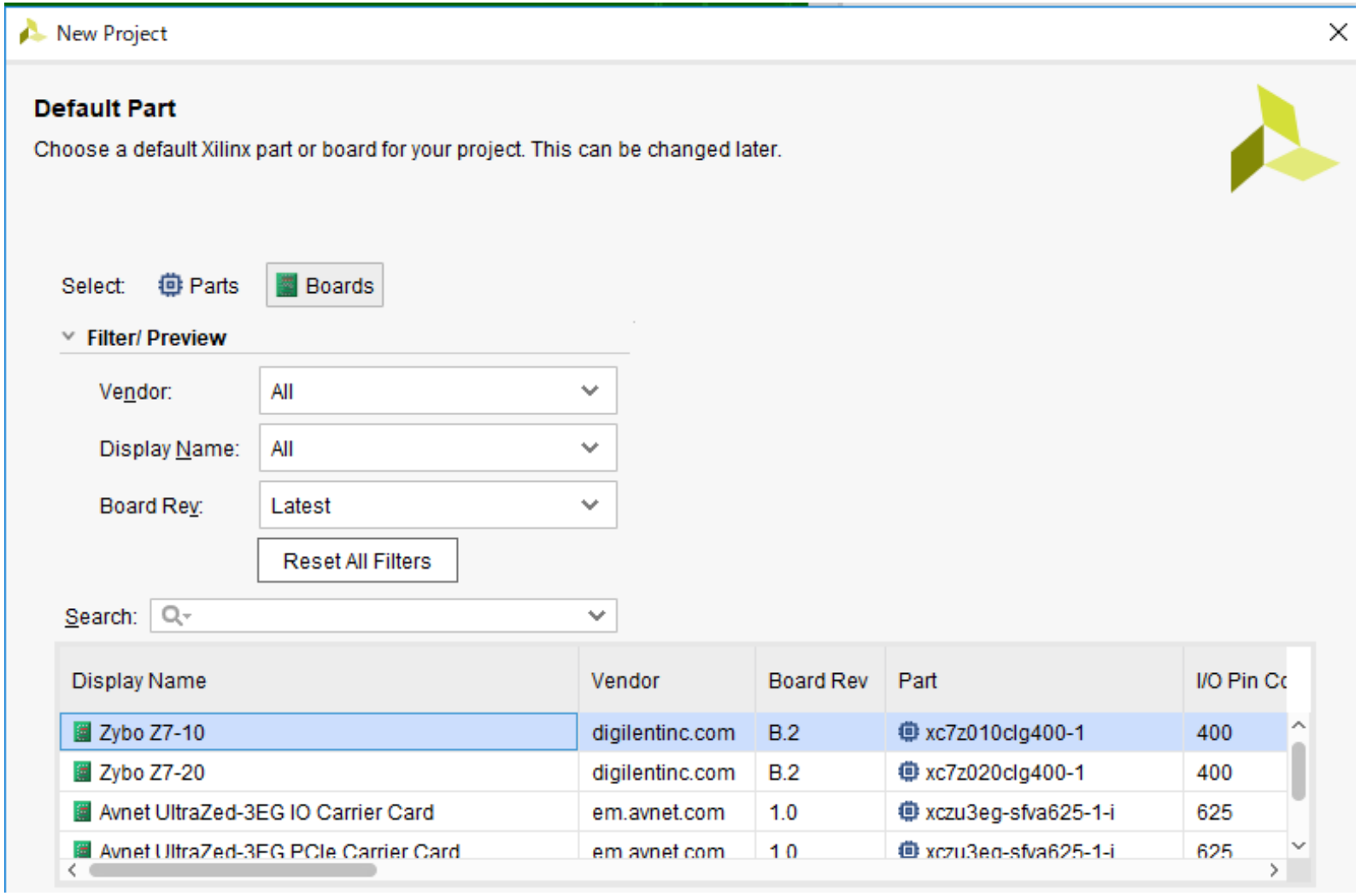
Select "RTL Project" with no sources and no



Select Zybo-Z7

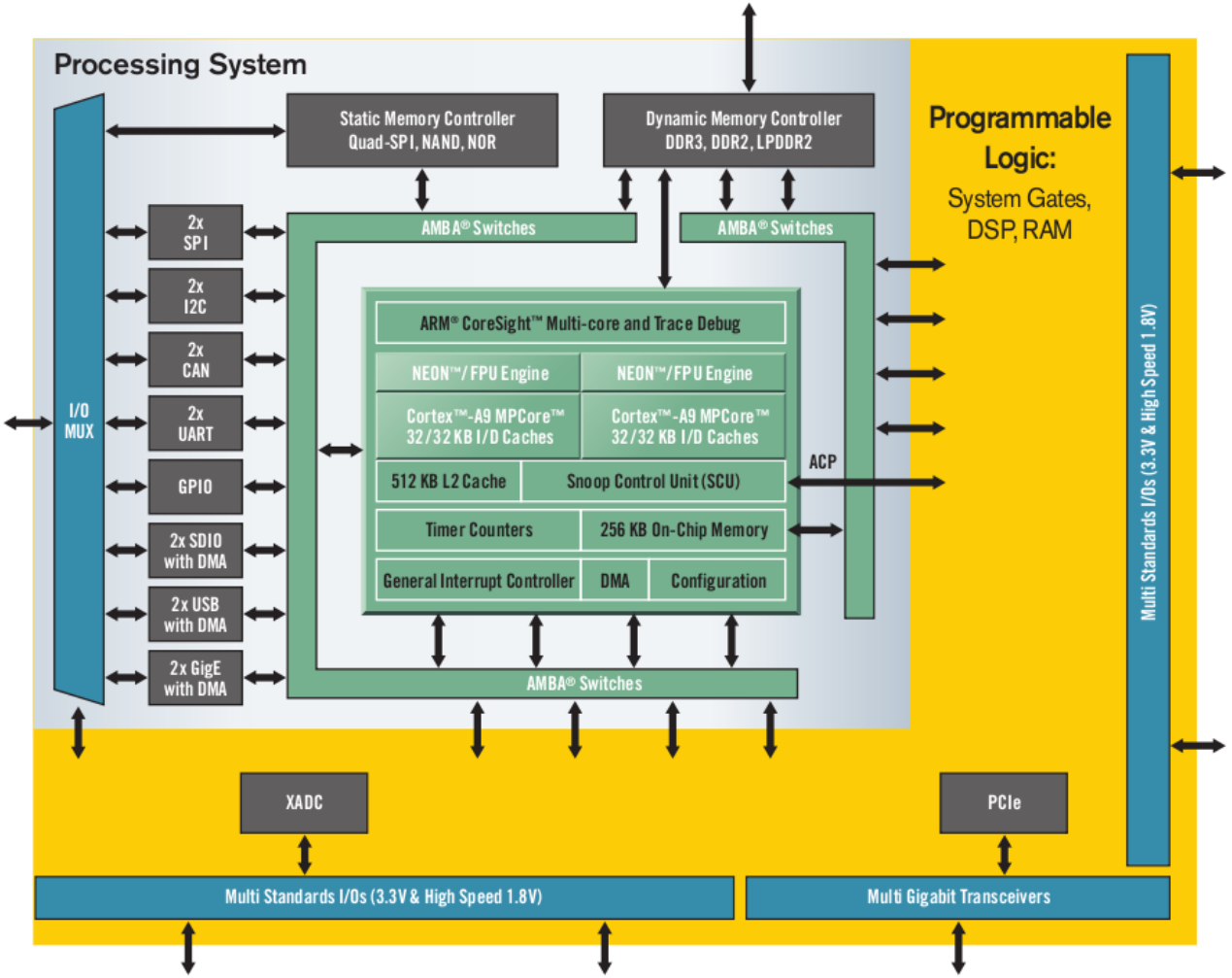
In "Default Part", select "Boards" tab, and select your Zybo! (Z7-10 or Z7-20)

Then, "Next" and "Finish"



Modern FPGA

- Processor System (PS) + Programmable Logic (PL)
- Hard macro IPs with dedicated IP (on PL)→ Short-time design



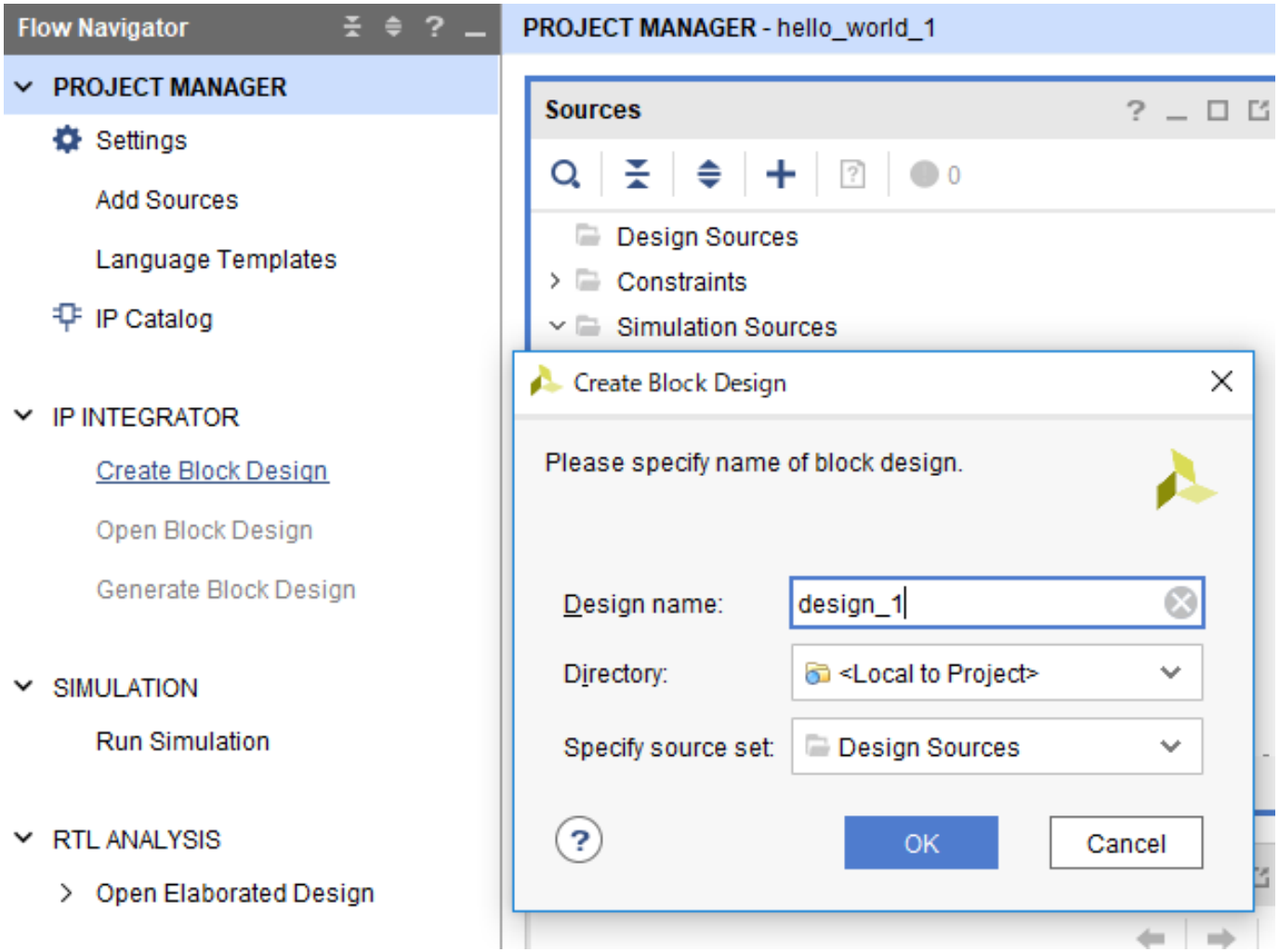
Source: Xilinx.com

IP Design

- The scale of the digital system increases day by day, the design-time is prolonged and the development cost is also increased
- Modules such as interface, control of peripheral devices, communication, encryption, compression, signal and image processing are common in many cases
  - Possible to reduce development time and cost problems by reusing them
- Commonable and reusable hardware library

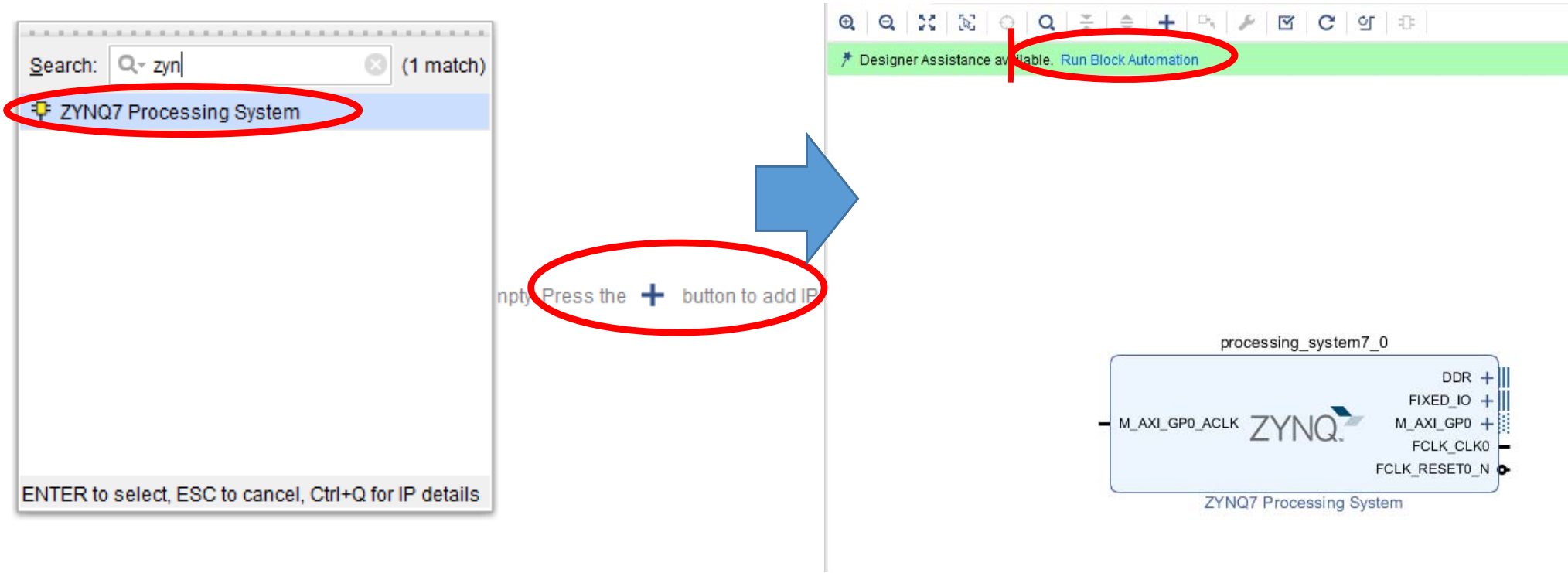
  → IP (Intellectual Property)

Launch IP Designer



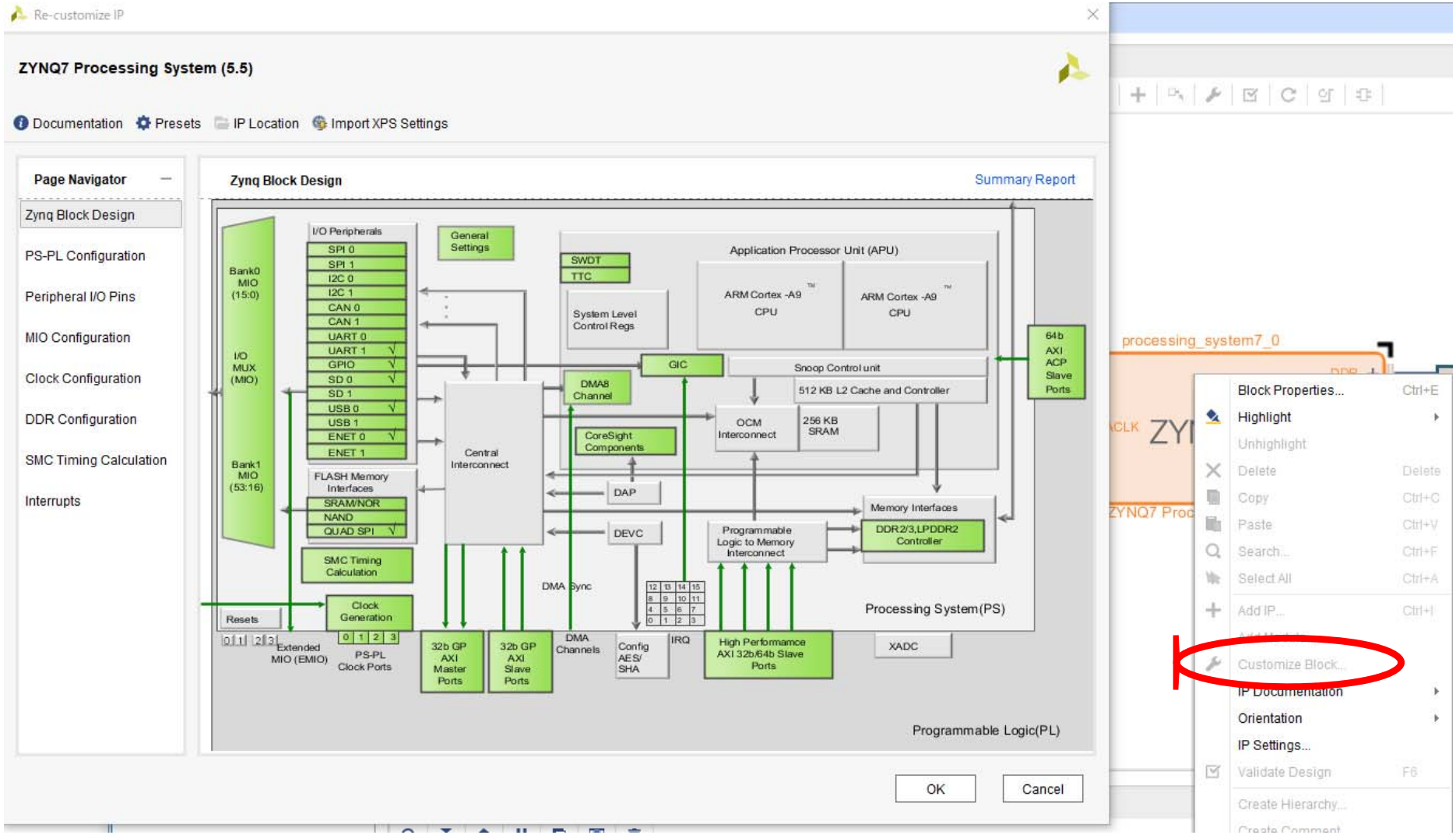
Select Zynq PS

- Click "+" button and select "ZYNQ7", then ZYNQ IP core is placed to BLOCK DESIGN Editor
- Click "Run Block Automation", and make sure "Apply Board Preset" is checked, then "OK"



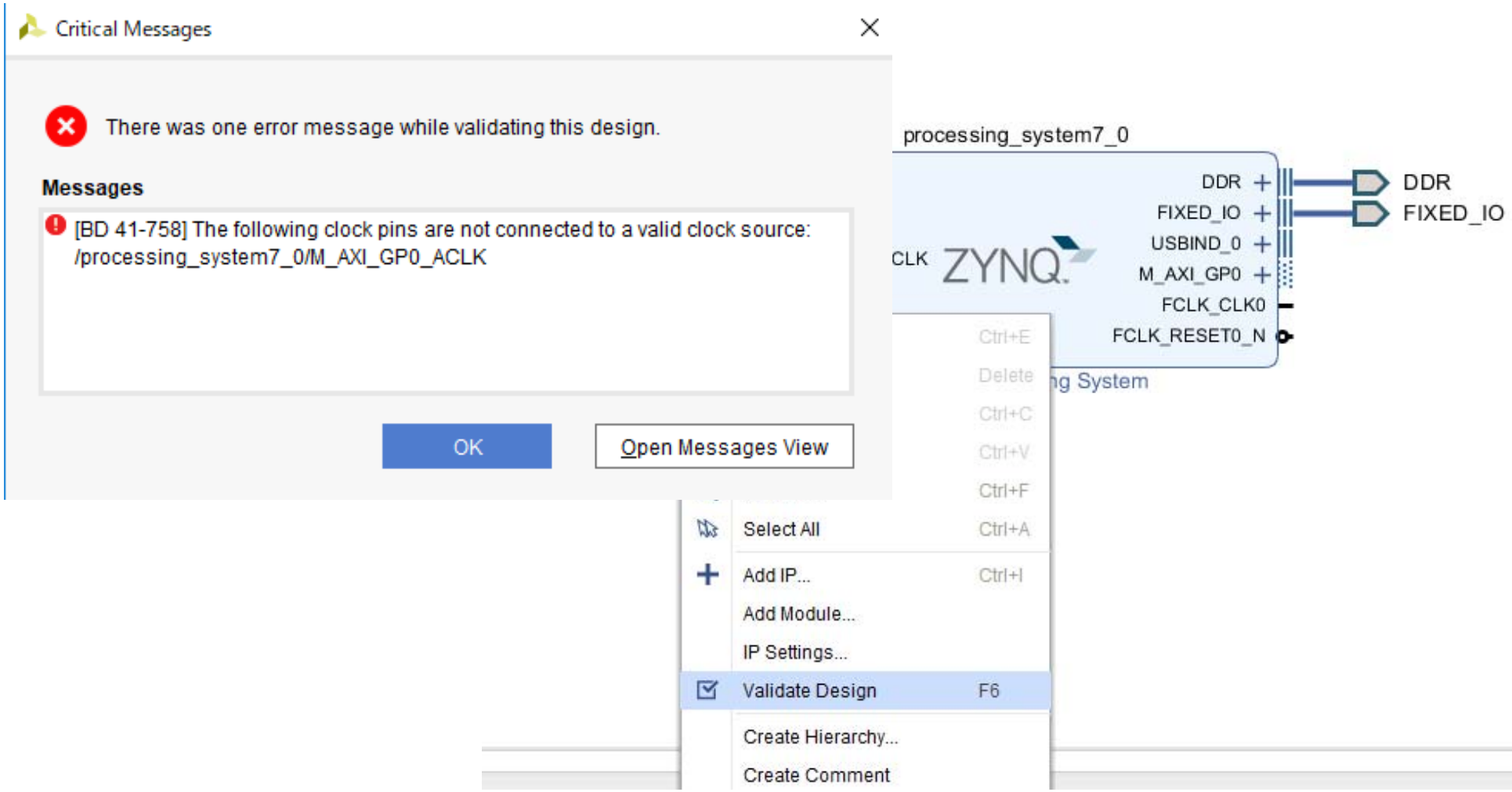
View Re-customize IP

Right click on "ZYNQ" block and select "Customize Block ...", then we can modify the settings (However, these have already been specified by BDF)



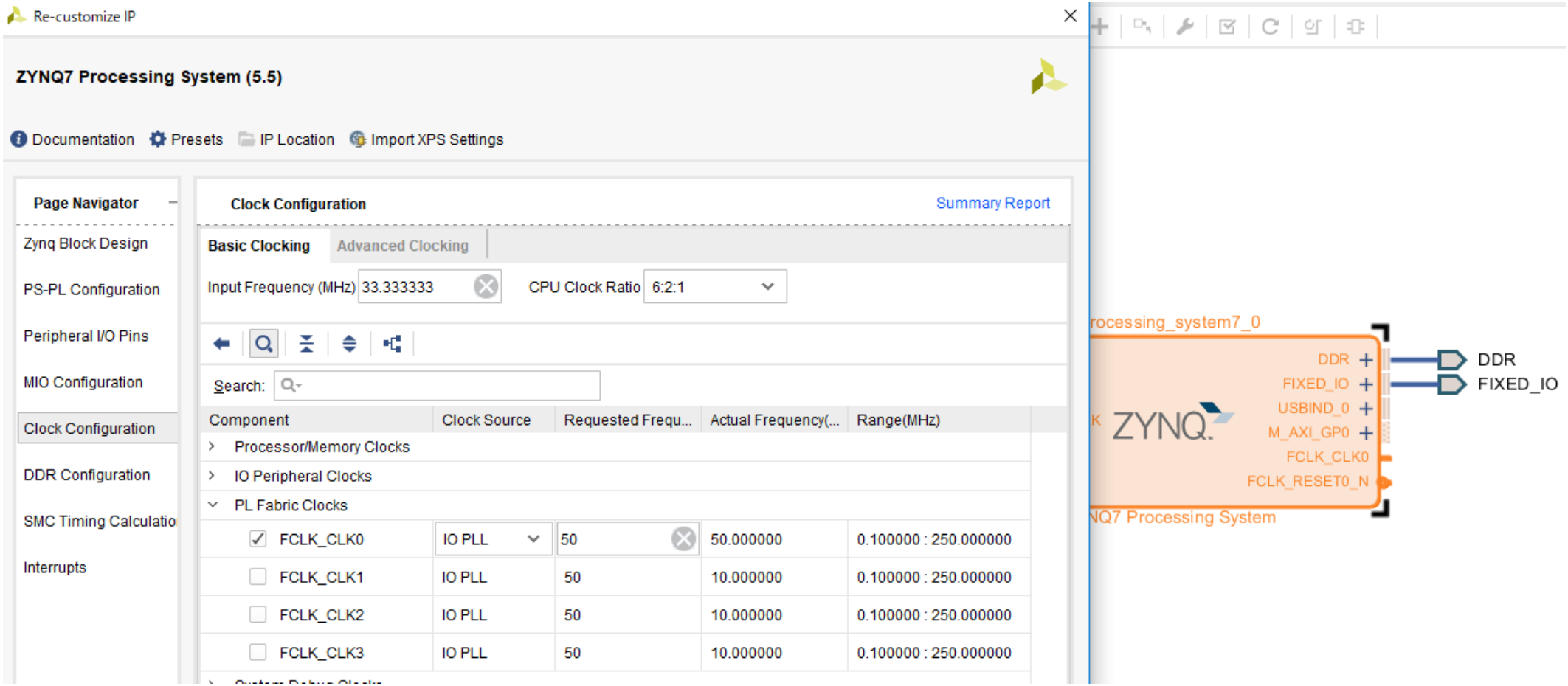
Valid Design

- Right click on empty space, and select "Validate Design"
- This tutorial meets critical error



Re-use PL Fabric Clocks

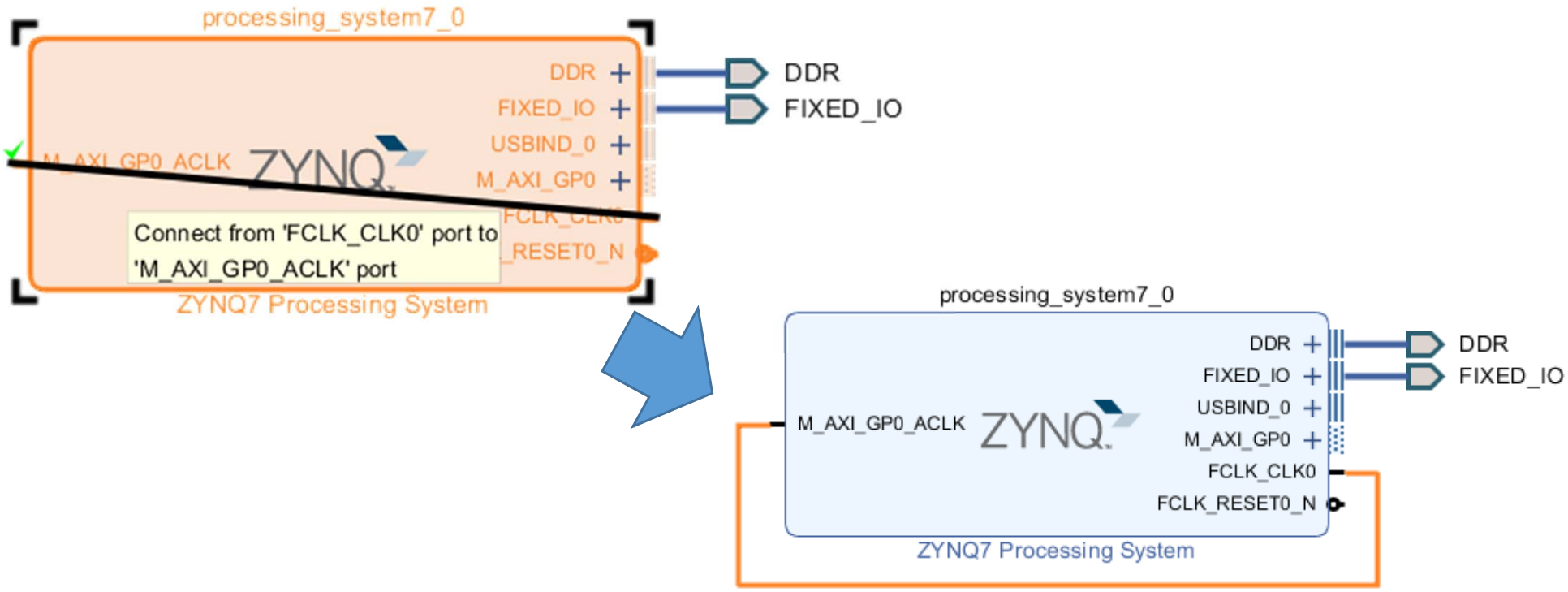
- PS generates 50MHz clock signal to control PL



Connect Clock Sources

Drag "FCLK_CLK0" and drop to "M_AXI_GP0_ACLK", then it automatically connect them

Validate design again, then there are no errors

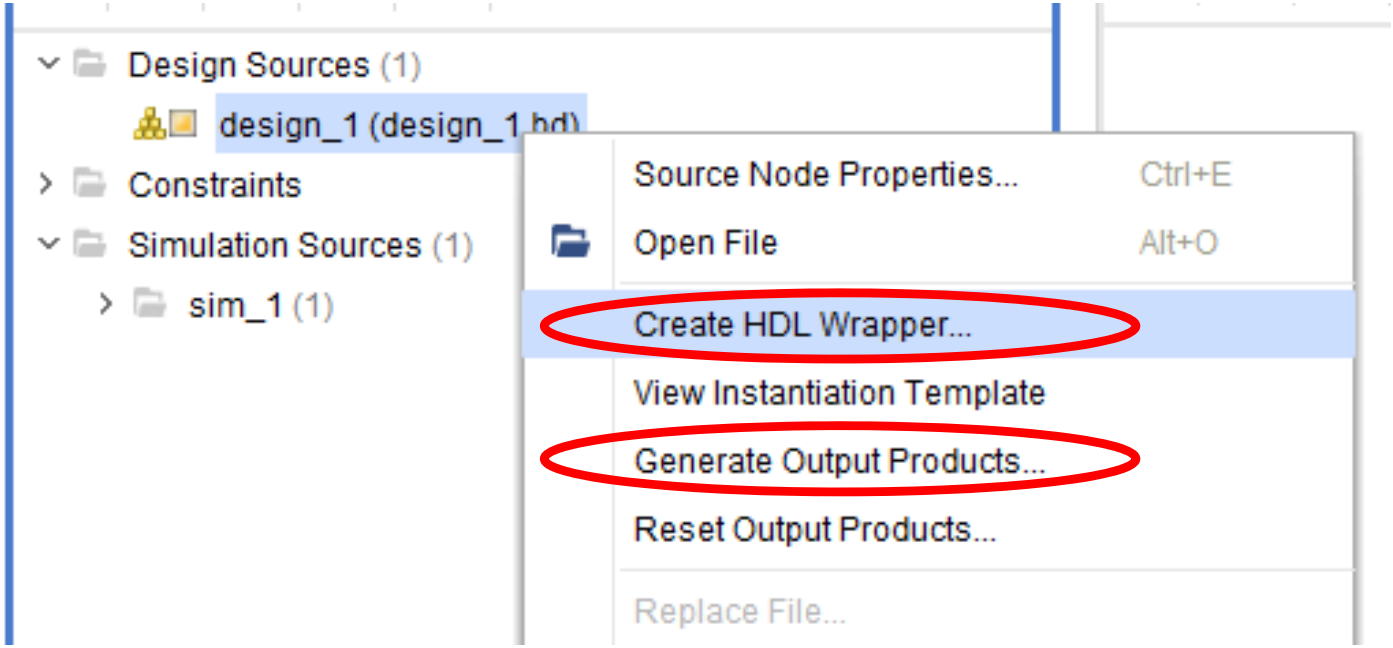


Generate HDL Files

To re-use HDL design flow, do following steps:

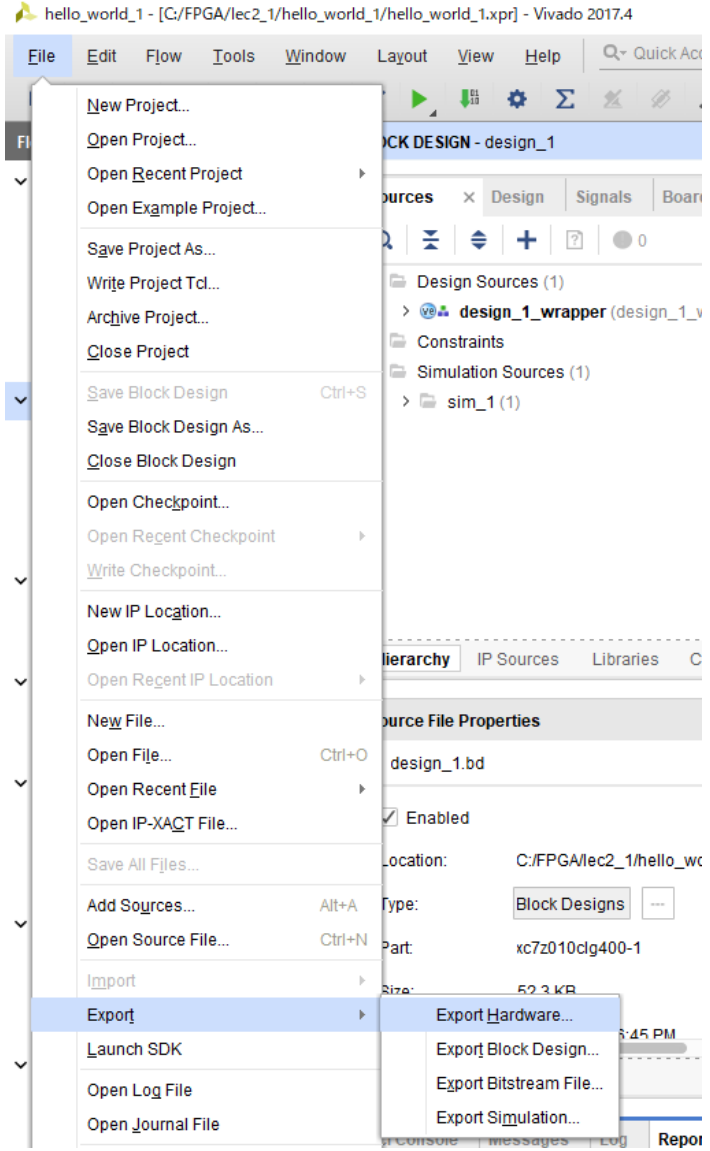
1. Right-click on "design_1", and select "Generate Output Products...", then "Generate"

2. Again, Right-click on "design_1", and select "Create HDL Wrapper...", then "OK"



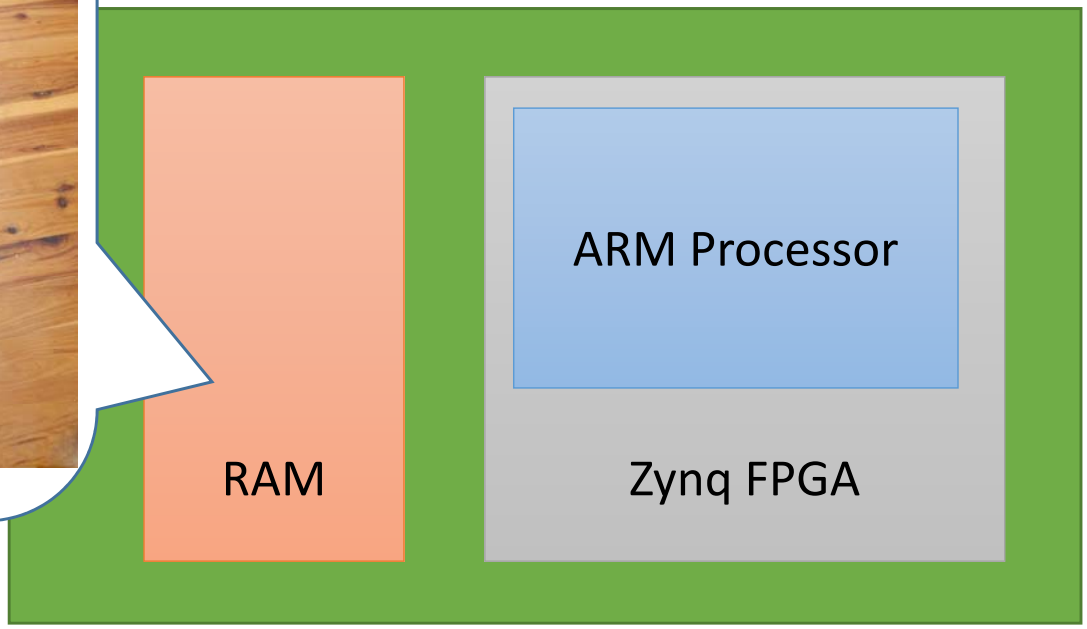
Synthesis Hardware and Export

- Click "Generate Bitstream", then "Yes" and "OK"
- Wait few minutes...
  - FPGA design tool do logic synthesis, place-and-routing, and generating bitstream
- After finish bitstream generation, then "Cancel"
- In "Menu", select "File" and "Export", then "Export Hardware"
- Check "Include bitstream", then "OK"



We are Here…

- Hardware (Processor) has already generated, however, there are no software on the RAM
  - HDF: Hardware definition file (Bitstream + Memory I/O map)
- Write software and compile it, then execute



Launch SDK

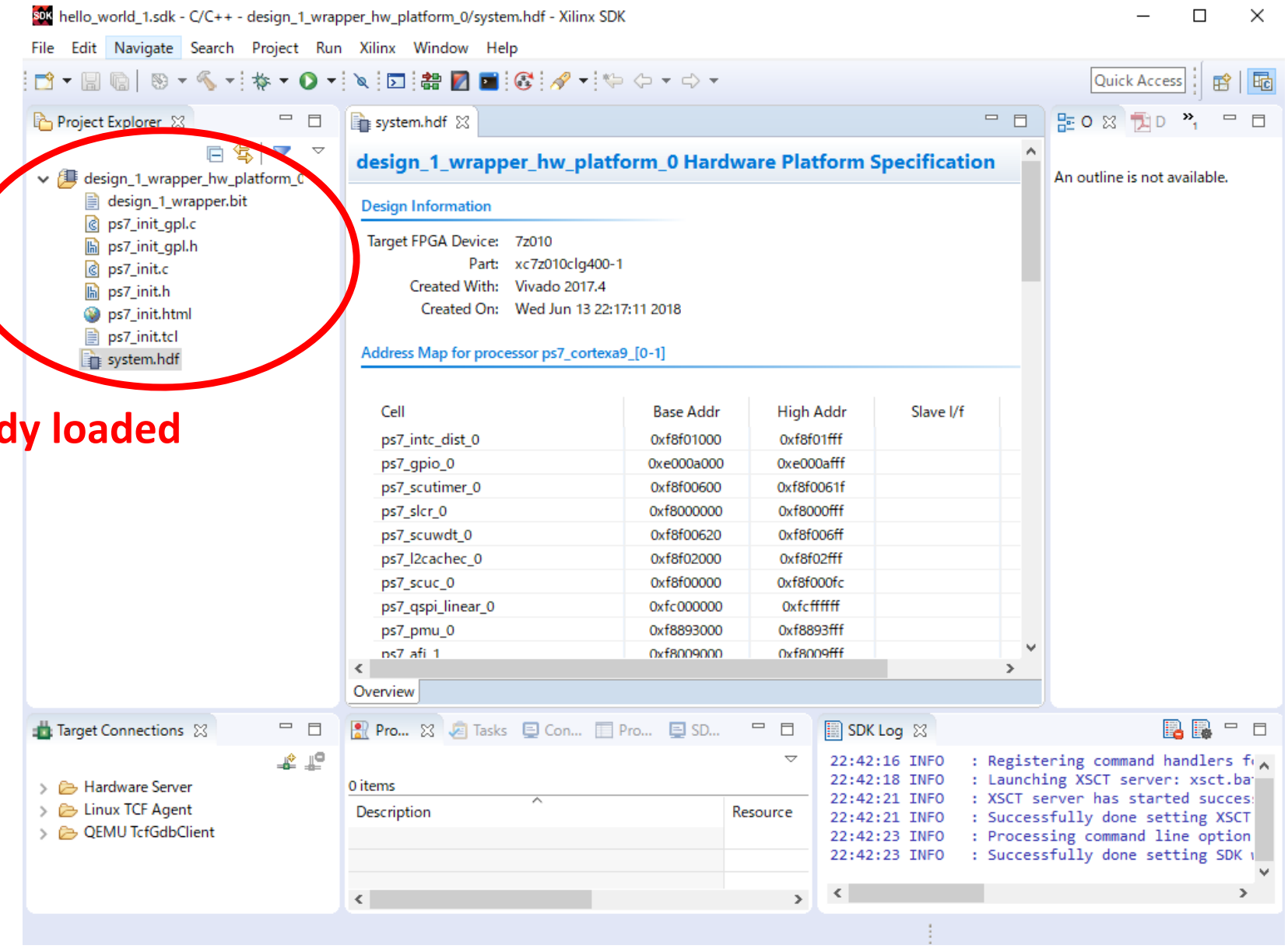
In Vivado, "Menu" -> "File" -> "Launch SDK" -> "OK"

Note: If you meet Visual C++ Runtime trouble, see https://forums.xilinx.com/t5/Installation-and-Licensing/Vivado-Xilinx-SDK-Error-Incorrect-Visual-C-Version/td-p/442628

-> Rename C:/Xilinx/SDK/2017.4/tips/win64/vcredist_x64.exe

(and xvcredist.ext both)

SDK Launched with designed HW



Make Project

In "Menu", "New" ->

"Application Project"

Set project name "HelloWorld"

OS: Standalone (default)

Hardware: (default)

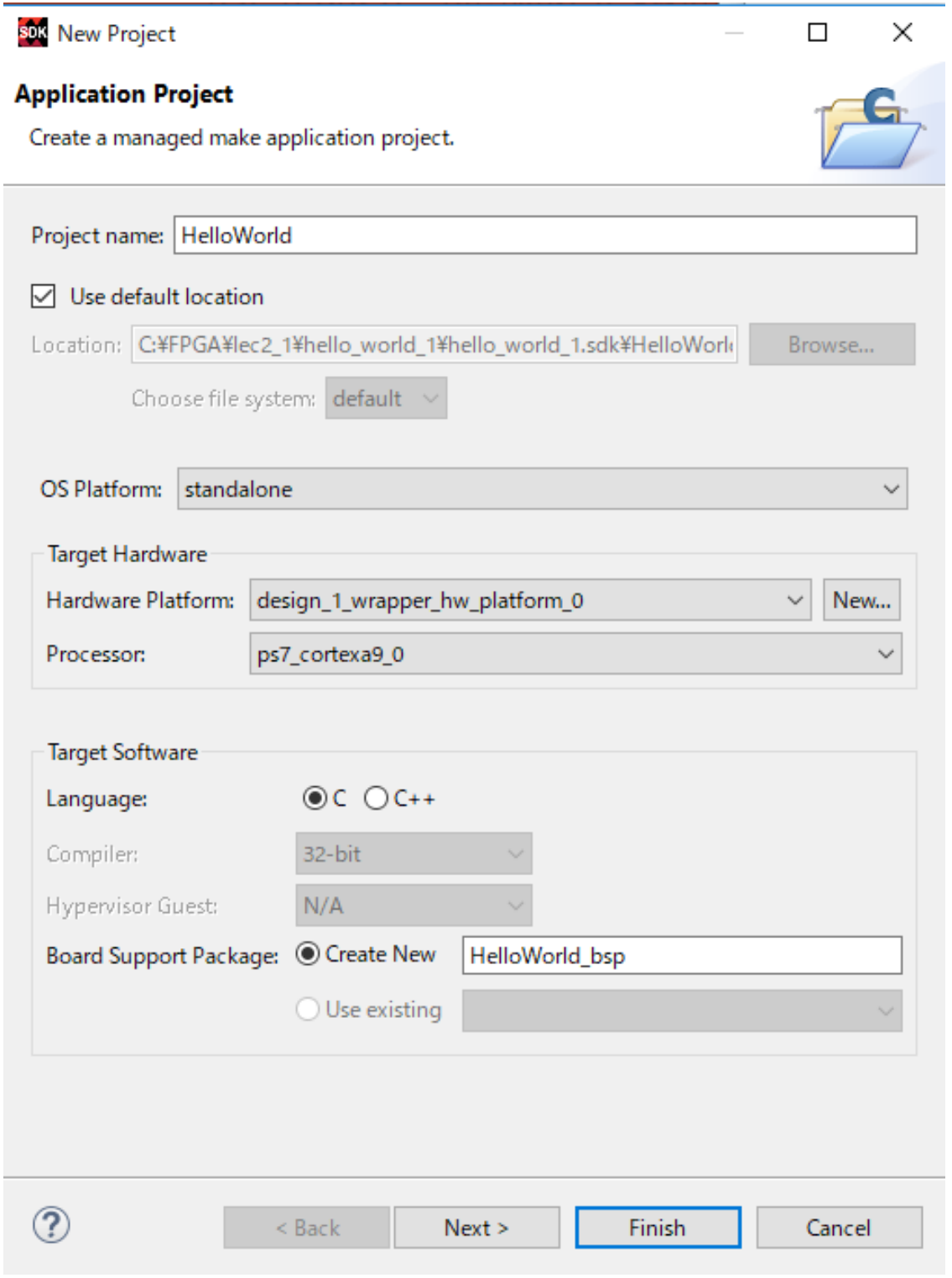
Processor: ps7_cortexa9_0

Language: C

Board Support Package (BSP):

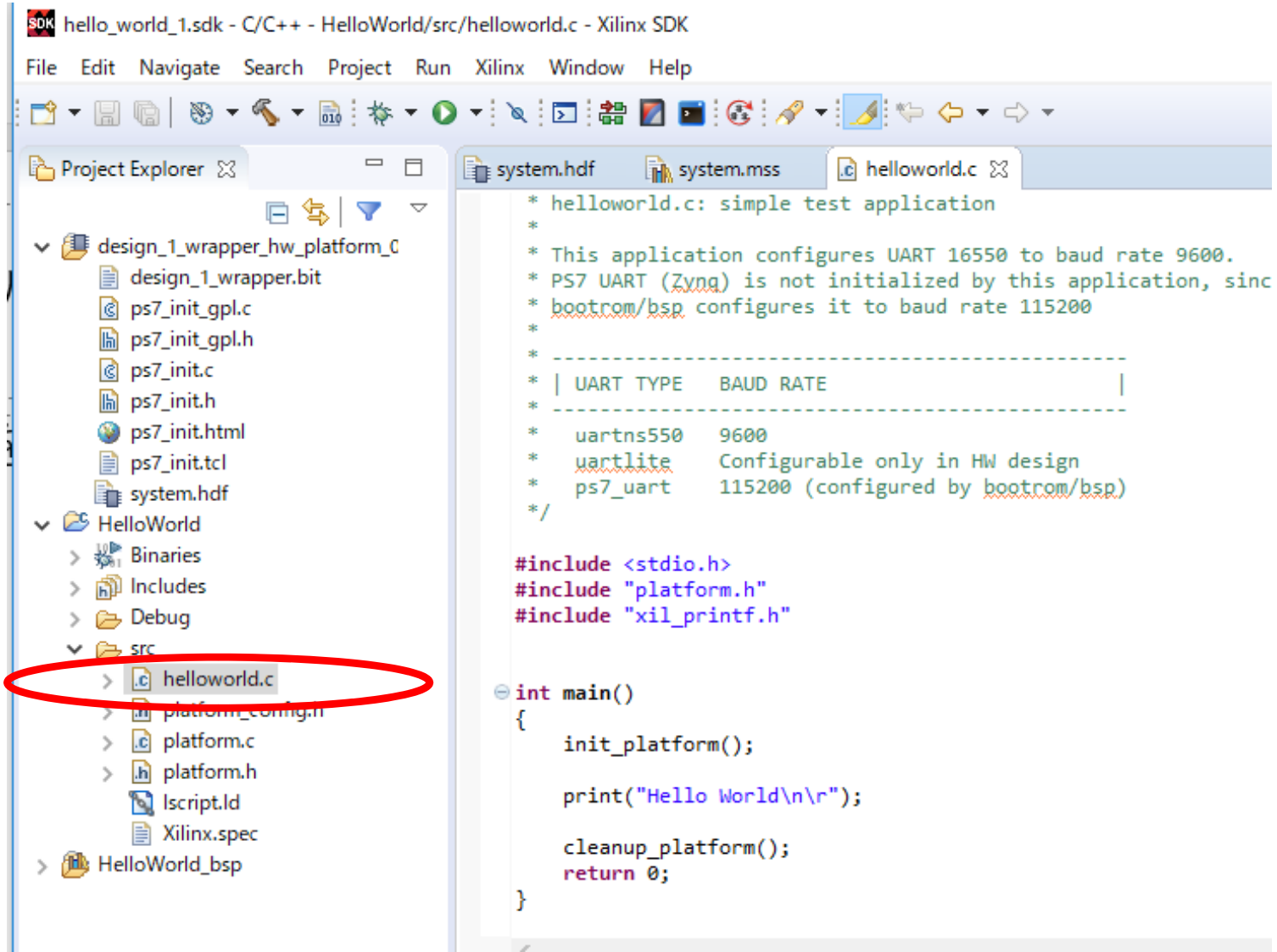
HelloWorld_bsp

(It is a library collection for your HW)



Check Source Code

"helloworld.c" has already been registered to a project



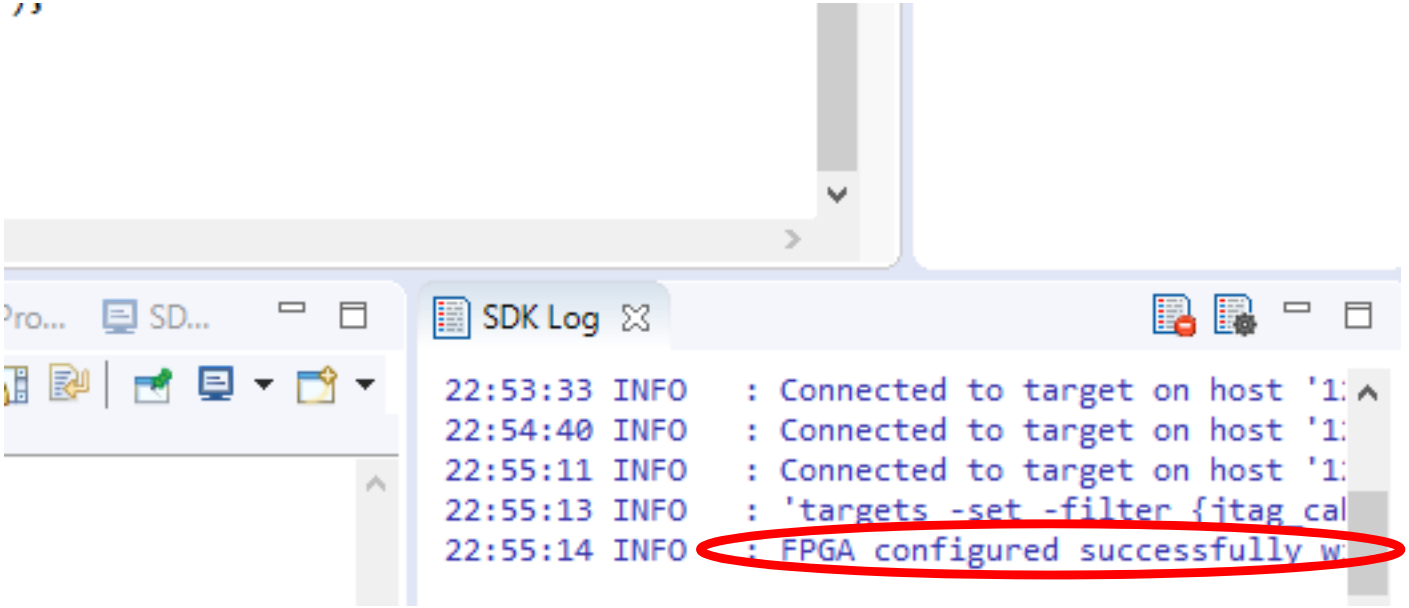
Environment Setup

Connect the Zybo to the PC, then turn-on power switch

In "Menu", "Xilinx" -> "Program FPGA", then "Program"

- Hardware is configured until you turn-off power

In SDK log, if "FPGA configured successfully..." is showed, then go to the next slide, otherwise, turn-off and turn-on power, then try to "Program" again



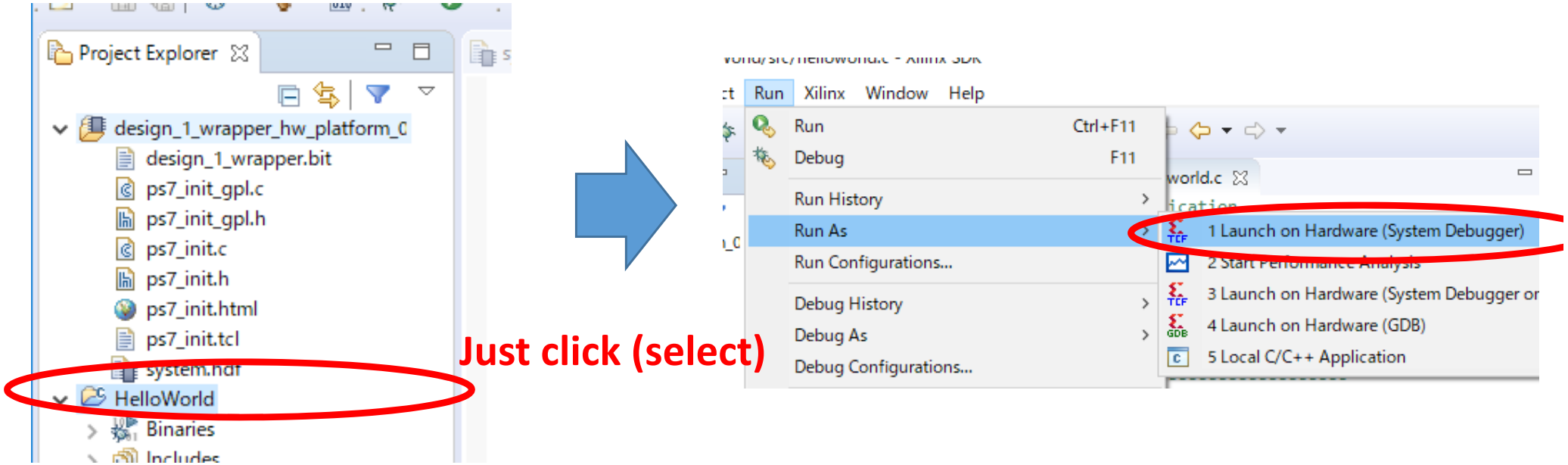
Environment Setup & Run

Connect the Zybo to the PC

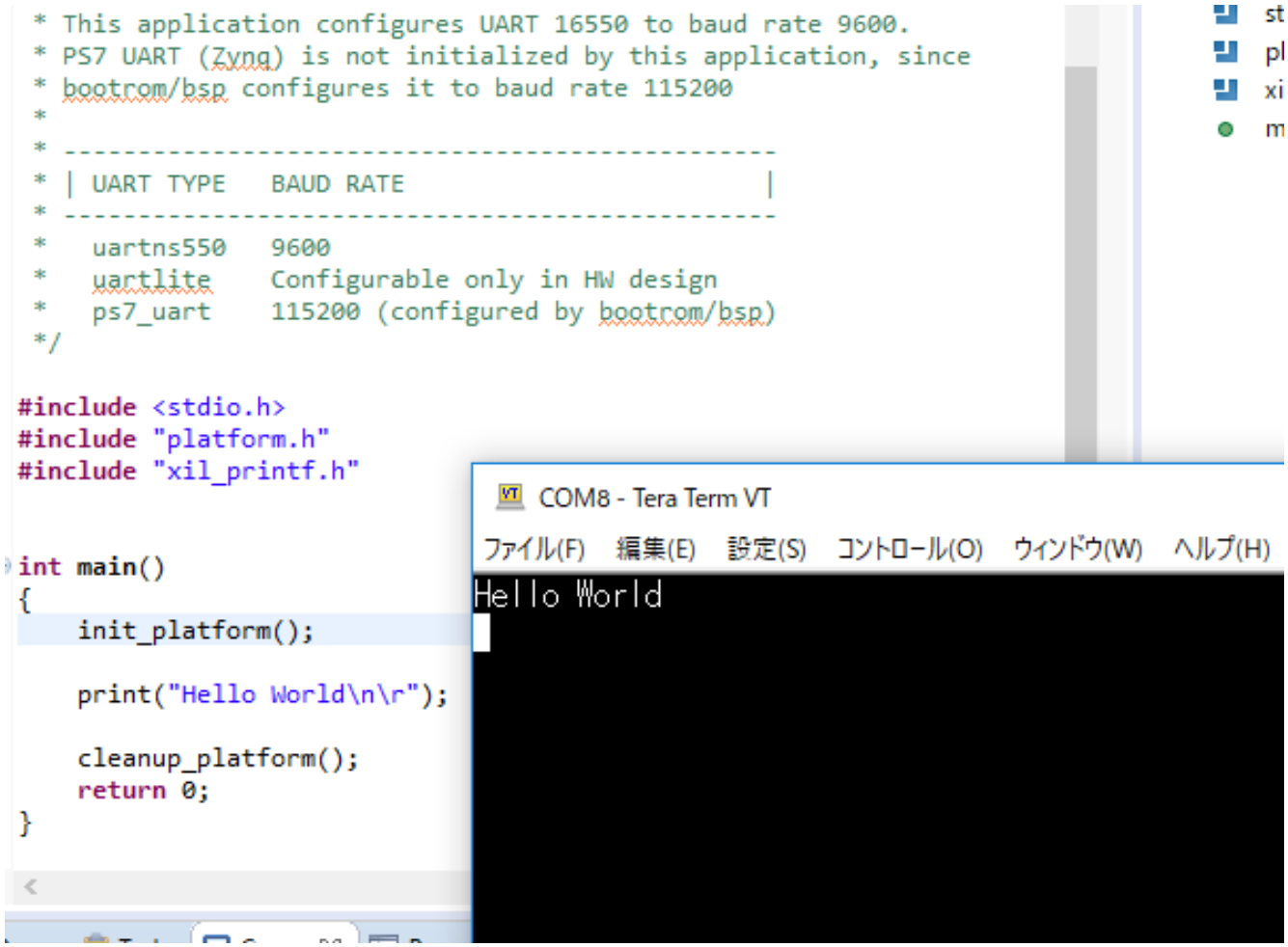
Run Terminal software (e.g. Tera Term for Windows, gtkterm for Unix)

Connect "USB Serial Port" with 115200 bps

Select "HelloWorld" project in the Project Explorer, then, in "Menu", "Run As" -> "Launch on Hardware (System Debugger)"



Welcome to SW World!



Exercise

1. (Mandatory) Execute the HDL design with following this tutorial, and "hello world" software tutorial. Then, send the source code and screen shot of your execution situation by a PDF.
2. If you meet any troubles, don't hesitate to contact me.

   nakahara@ict.e.titech.ac.jp

Deadline is 3rd, July, 2018 (At the beginning of the next lecture)