

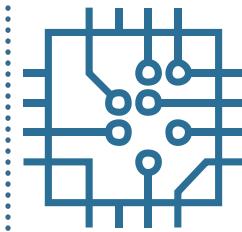
# Načrtovanje strojne opreme in vgradnih sistemov z vezji FPGA

Spletna delavnica  
20. 10. 2021, 12:30 – 16:00



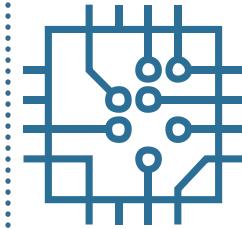
REPUBLIKA SLOVENIJA  
MINISTRSTVO ZA GOSPODARSKI  
RAZVOJ IN TEHNOLOGIJO

# Outline



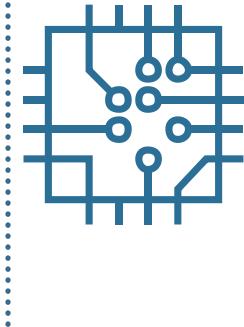
- Part 1
  - FPGA structure and design platform
  - VHDL hardware design in FPGA
  - Embedded system design on FPGA
- Part 2
  - Embedded software design FPGA
  - IP core development and integration
- Part 3
  - Software and hardware debugging

# Introduction



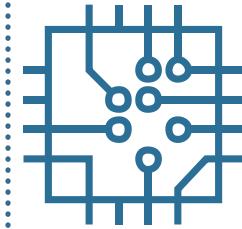
- Field Programmable Gate Array
  - Integrated circuit that can be programmed after manufacturing (at field)
- First FPGA in 1984 (Altera EP300)
- Initially primarily used in telecommunication and networking
- Spread to consumer, automotive, and industrial applications
- Used for acceleration (Bing search engine), in data centers (Amazon AWS)

# Introduction



- Producers
  - AMD (just acquired Xilinx),
  - Intel (acquired Altera),
  - Microsemi (Actel),
  - Lattice, ...
- Digital circuits
  - synchronous design,
  - asynchronous design
- Analog and mixed signal blocks (ADC, PLL)

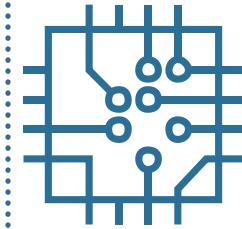
# Introduction



## Comparison with other technologies

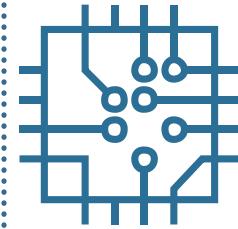
	ASIC	FPGA	CPU
Speed	Very fast (GHz)	Fast (100 MHz)	Medium
Power consumption	Low	Medium	High
Development cost	Very high	Medium	Low
Reconfiguration	Limited	Field upgrades	High

# Outline

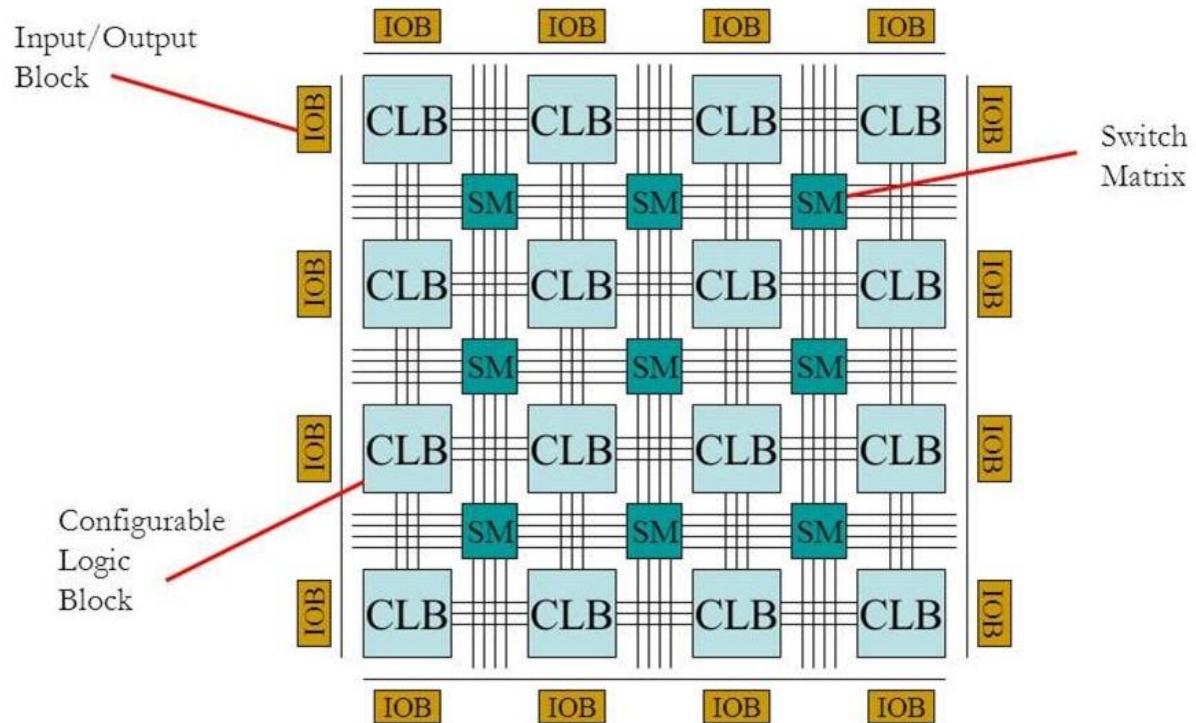


- Part 1
  - **FPGA structure and design platform**
  - VHDL hardware design in FPGA
  - Embedded system design on FPGA
- Part 2
  - IP core development and integration
  - Embedded software design FPGA
- Part 3
  - Software and hardware debugging

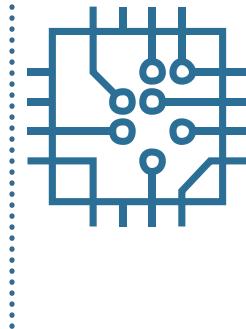
# General FPGA structure



## FPGA Structure

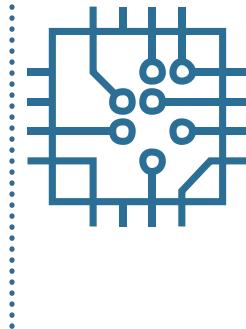


# Configuration memory



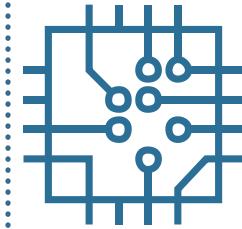
- 2D array of frames
- Technology
  - Volatile memory: SRAM (Xilinx, Intel)
  - Non-volatile memory
    - Flash (Microsemi, Lattice)
    - EEPROM, EEPROM (obsolete)
    - fuses
- Vulnerable to radiation induced faults (SEU)
  - Error correction techniques

# FPGA hardware resources

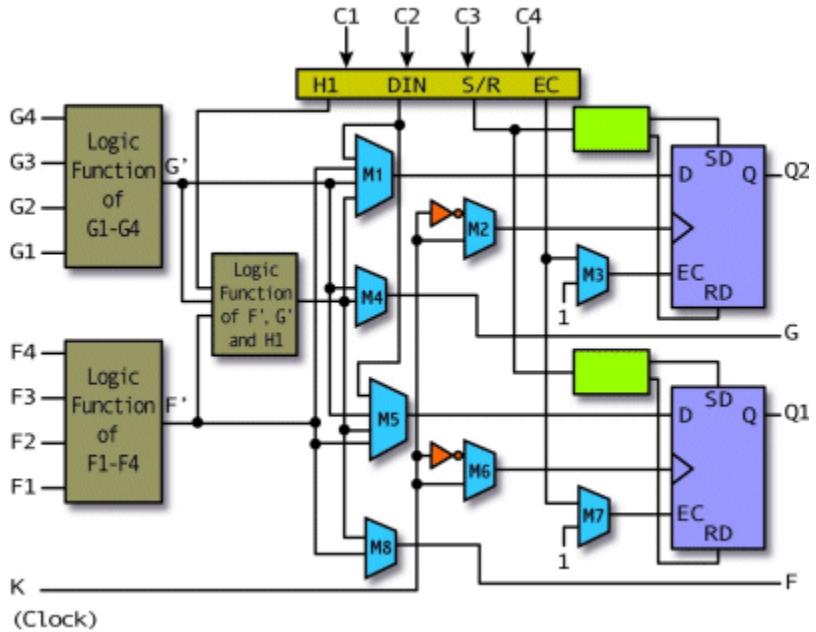


- Logic blocks (CLB or slice)
- IOB
- Memory
- DSP blocks (computational blocks)
- Switching matrix
- Clock sources
- Special blocks (e.g. ICAP)
- Analog or mixed signal blocks

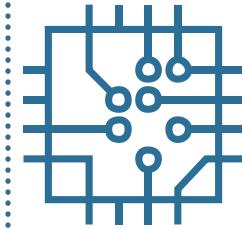
# Logic block



- Look-Up Table
  - 6 or 5 input
- Flip-flop
- Carry-add logic
- Multiplexer
- Shift register
- Distributed RAM

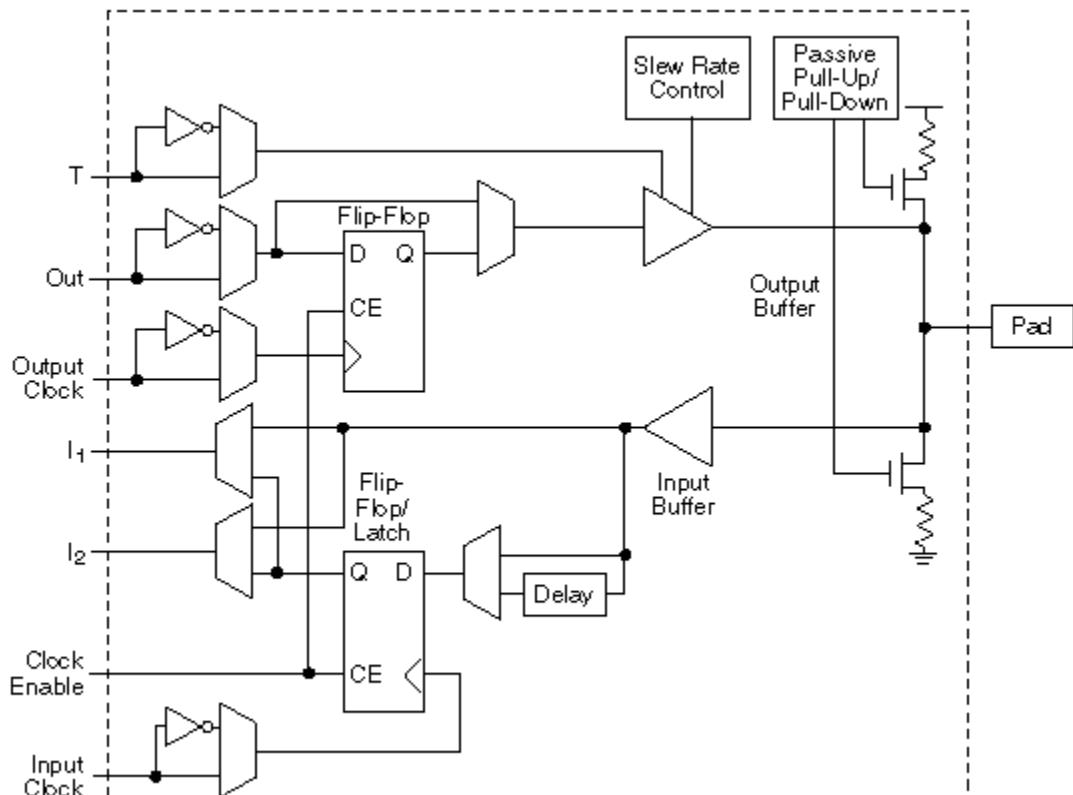
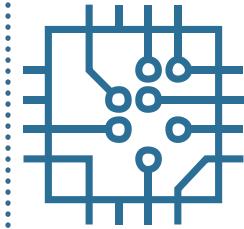


# Input / Output Block



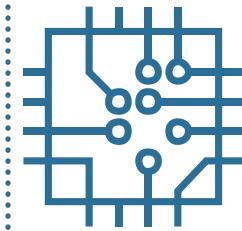
- Interface to chip pins
- Support voltage levels for different semiconductor technologies
  - TTL, CMOS, ...
- Different speed:
  - General purpose interface (majority of IO pins)
  - High performance interface (DDR, PCI, Ethernet)
- Latching flip-flops
- Buffers on the IO pins

# Input / Output Block



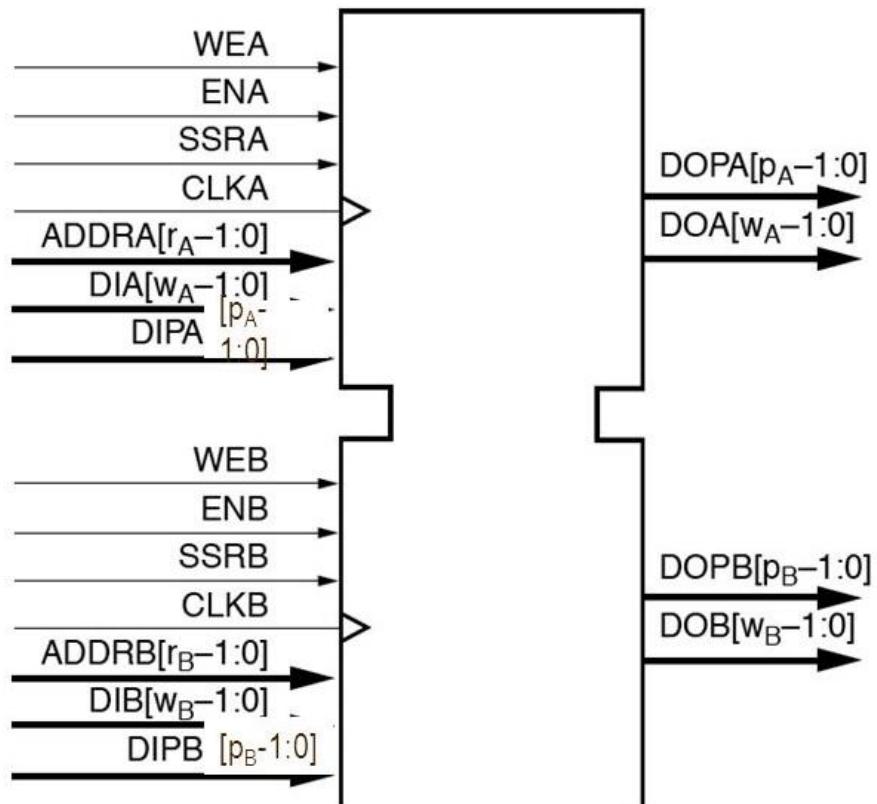
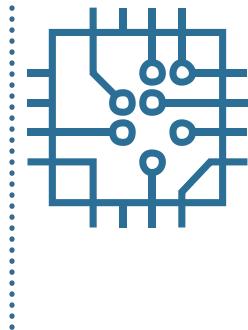
X6704

# Memory

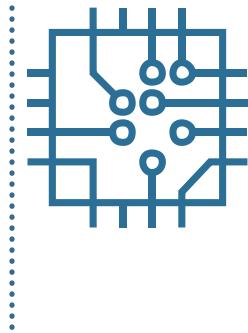


- Distributed memory consumes logic blocks
- Block RAM (memory cores)
- Synchronous memory
- 2 x 18kb or 36kb RAM blocks
- Single/Dual port RAM
- Used for
  - Buffers
  - FIFOs
- Integrated error correction

# Memory

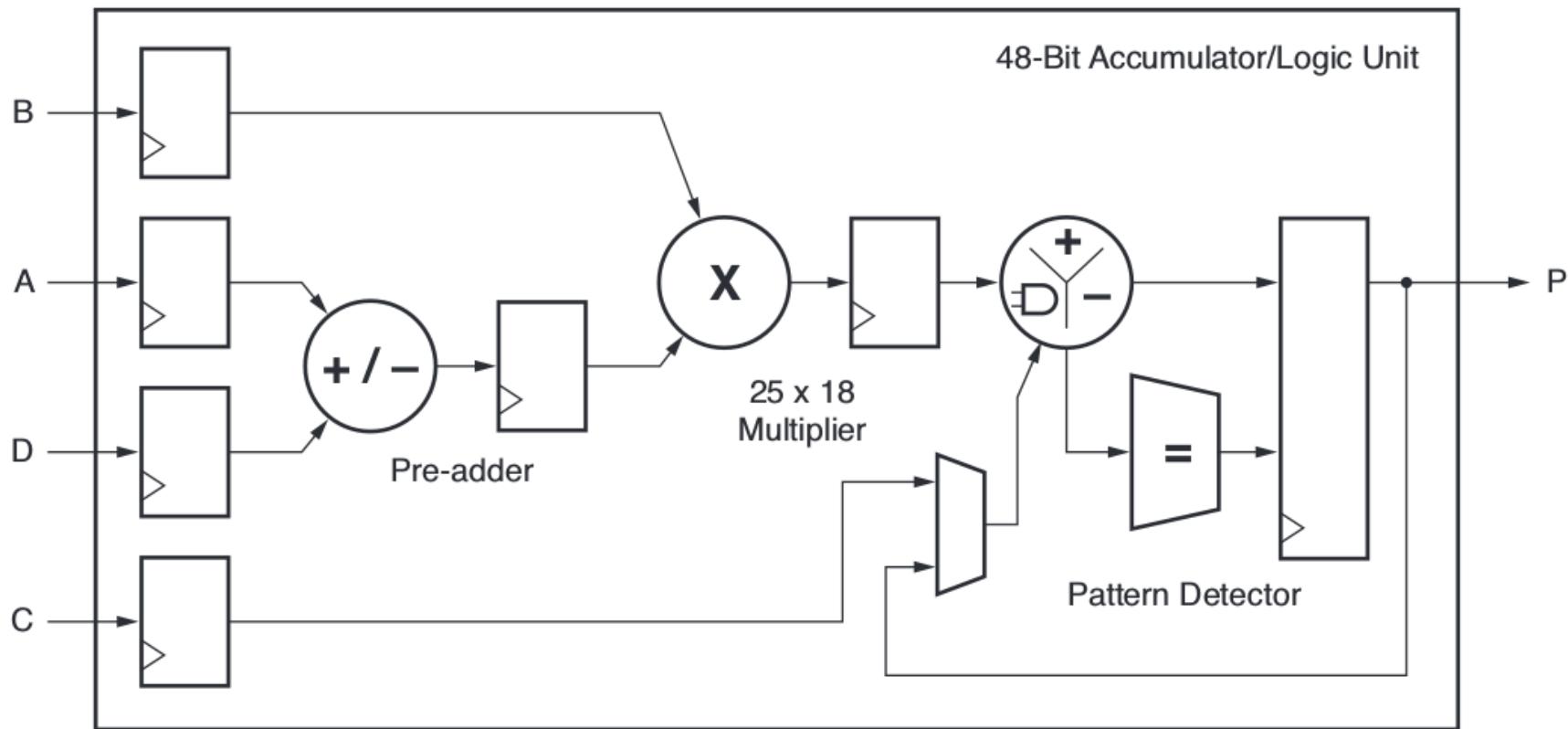
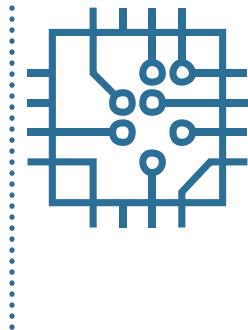


# DSP block



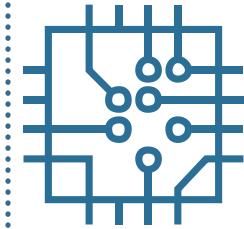
- 25 x 18 multiplier
- 48 bit adder
- 48 bit logic operators
- Pipeline registers
- Pattern detectors
- SIMD operations
- Pre-adder

# DSP block

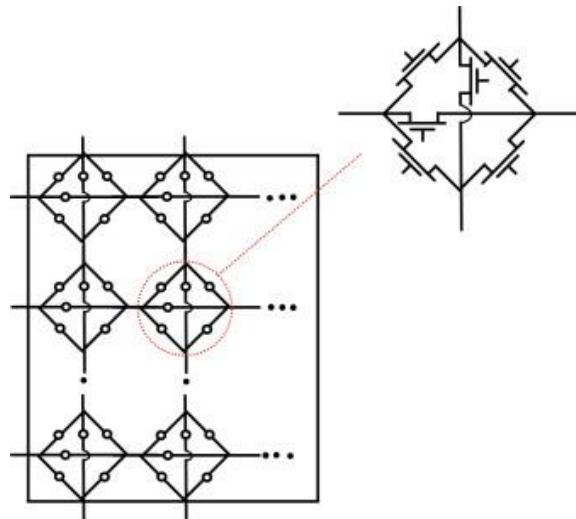


UG479\_c1\_21\_032111

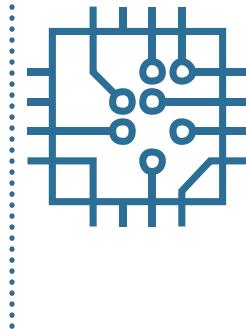
# Switching matrix



- Connects horizontal and vertical signals
- Route signals between IOBs, CLB
- Controlled by configuration memory

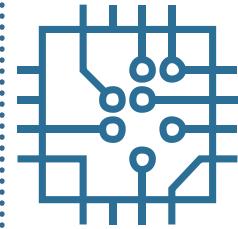


# Clock sources



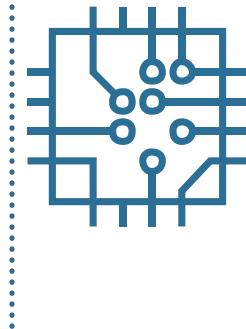
- Global clock buffer
  - High fan-out clock distribution buffer
- Low-skew clock distribution
- Clock regions
- Clock management tile (CMT)
  - One Mixed-Mode Clock Manager (MMCM) and one Phase-Locked Loop (PLL) per clock
  - Frequency synthesis, clock de-skew, and jitter filter

# Special blocks



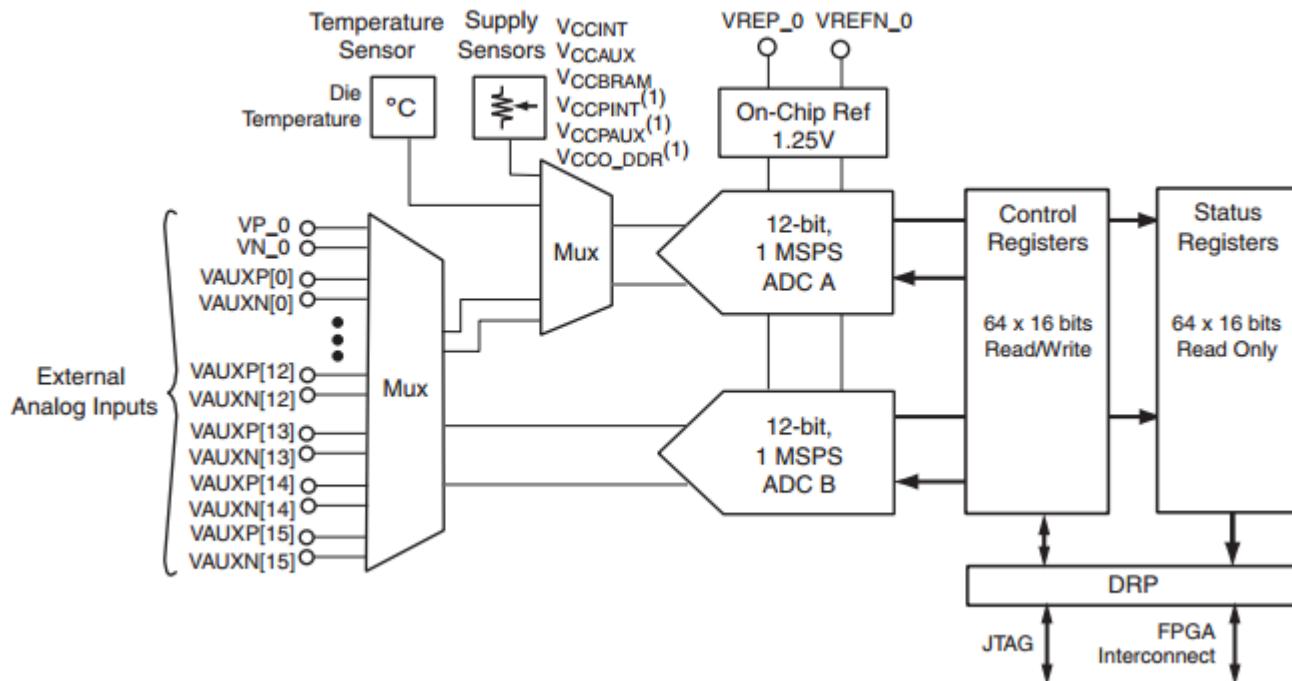
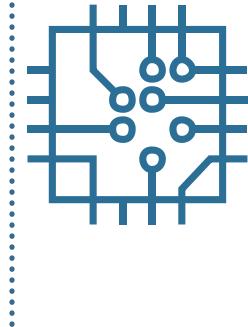
- ICAP interface
  - Dynamic partial reconfiguration
- SEM controller
  - Soft Error Mitigation controller
- Frame ECC interface
- Status interface
- Error Injection interface
- Monitor interface, ...

# Analog or mixed signal blocks

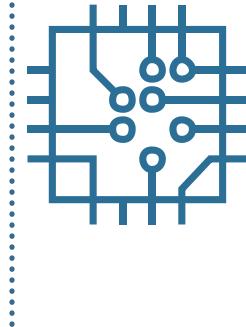


- XADC block introduced in latest Xilinx 7-series and Zynq family
  - Dual 12-bit 1Msps ADC
  - 17 analog inputs
  - 1V input range
  - 16 bit resolution
  - Built in digital gain and offset calibration

# Analog or mixed signal blocks



# Xilinx FPGA device families

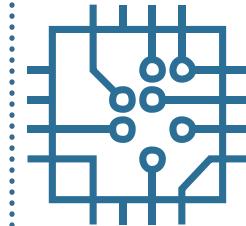


- XC series (from 1984)
- Spartan series
- Virtex – high performance family

With 7-series Xilinx introduced new families:

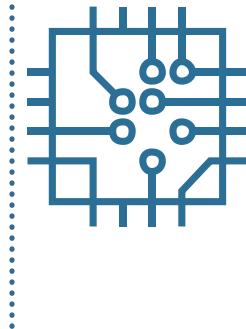
- Artix – low power and cost
- Kintex – best price/performance ratio
- Zynq – SoC with hard ARM processor core

# Xilinx 7-series families



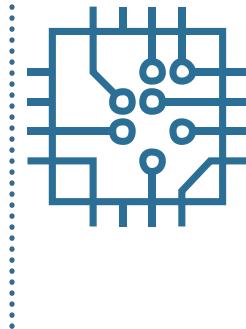
	Artix	Kintex	Virtex	Zynq
Logic cells [k]	33-215	66-478	583-1955	26-444
BRAM [Mb]	2-13	4-34	26-68	2-27
DSP	90-740	240-1920	1260-3600	80-2020
DSP perf (GMAC)	929	2845	5335	2622
Transcievers	Up to 16	Up to 32	Up to 96	Up to 16
Trans. Perf (Gbps)	6.6	12.5	28	12.5
Mem. Perf. (Mbps)	1066	1866	1866	1333
I/O pins	106-500	285-500	350-1100	54-400

# Xilinx Vivado



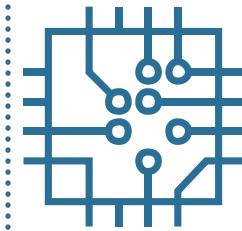
- WebPack:
  - Limited to smaller FPGA devices
  - Synthesis
  - HDL Simulation
  - Logic analyzer
  - High-level synthesis
- Design edition:
  - All 7-series devices and later devices
- System edition:
  - System Generator for DSP

# Xilinx Vivado



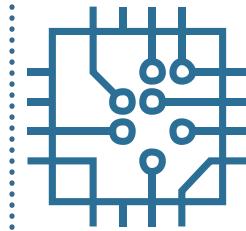
- Hardware representation (Vivado HLx):
  - HDL languages:
    - VHDL language
    - Verilog
    - Not all code is synthesizable
  - Schematic capture is not supported
  - C / C++/ SystemC (Vivado HLS)
  - Constraint specification
    - Timing constraints, clock definitions
    - I/O pin definitions

# Outline



- Part 1
  - FPGA structure and design platform
  - **VHDL hardware design in FPGA**
  - Embedded system design on FPGA
- Part 2
  - IP core development and integration
  - Embedded software design FPGA
- Part 3
  - Software and hardware debugging

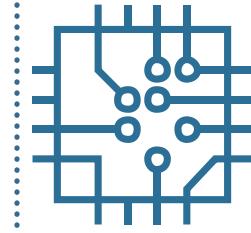
# Vivado: Getting started



- Quick start
  - Create new projects
  - Open existing projects
  - Open example projects
- List of recent projects
- Link to various tasks
  - Manage IP
- Learning center
  - User guides
  - Tutorials



# Vivado: Create new project



- Select project folder
- Different types
  - RTL
  - Post-synthesis
  - I/O planning
  - Import from other tools

**Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:  ✖

Project location:  ✖ ...

Create a new Vivado project

**Project Type**  
Specify the type of project to create.

RTL Project  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
 Do not specify sources at this time

Post-synthesis Project  
You will be able to add sources, view device resources, run design analysis, planning and implementation.  
 Do not specify sources at this time

I/O Planning Project  
Do not specify design sources. You will be able to view part/package resources.

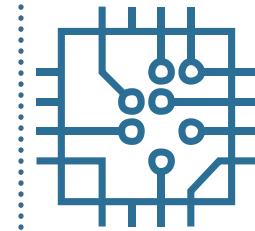
Imported Project  
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project  
Create a new Vivado project from a predefined template.

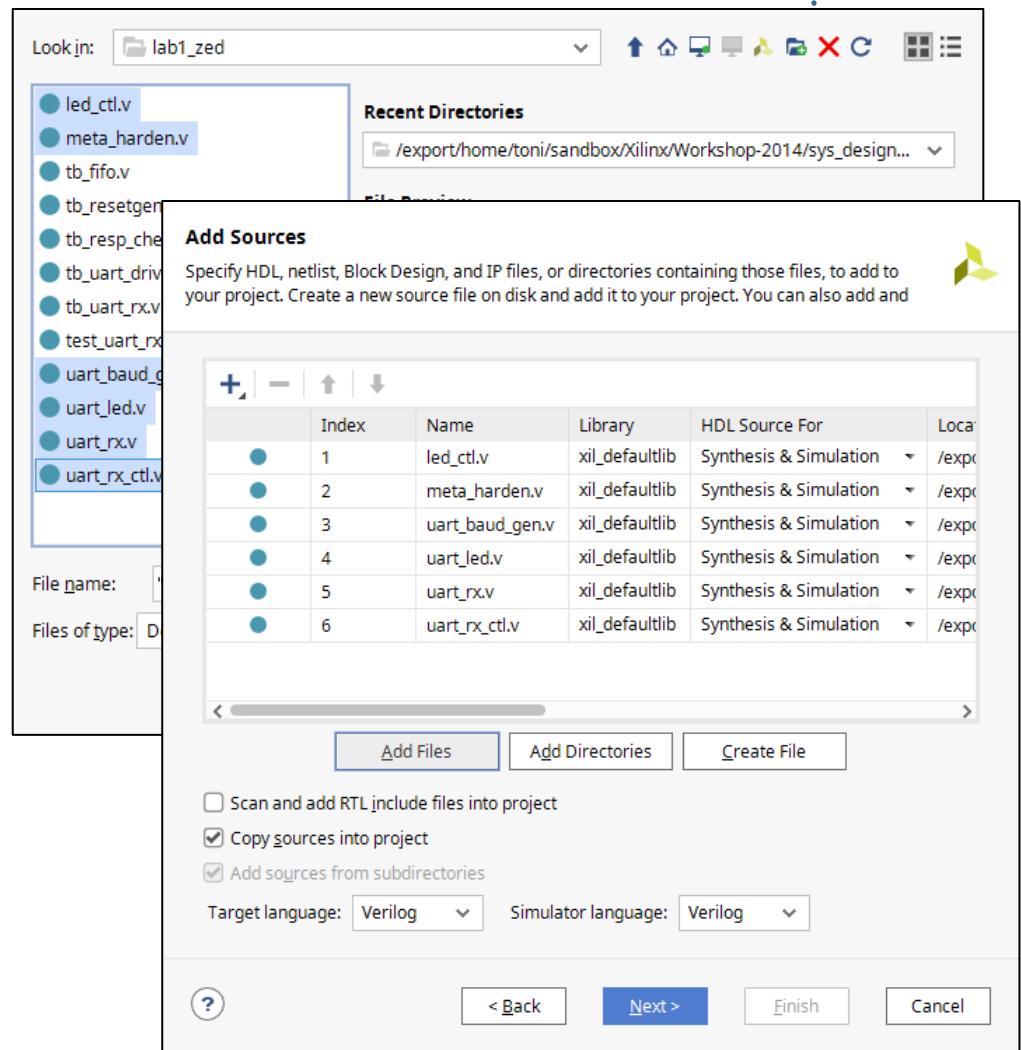
?

? < Back Next > Finish Cancel

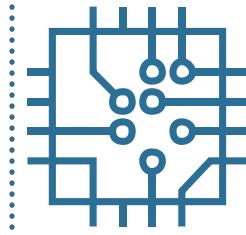
# Vivado: Create new project



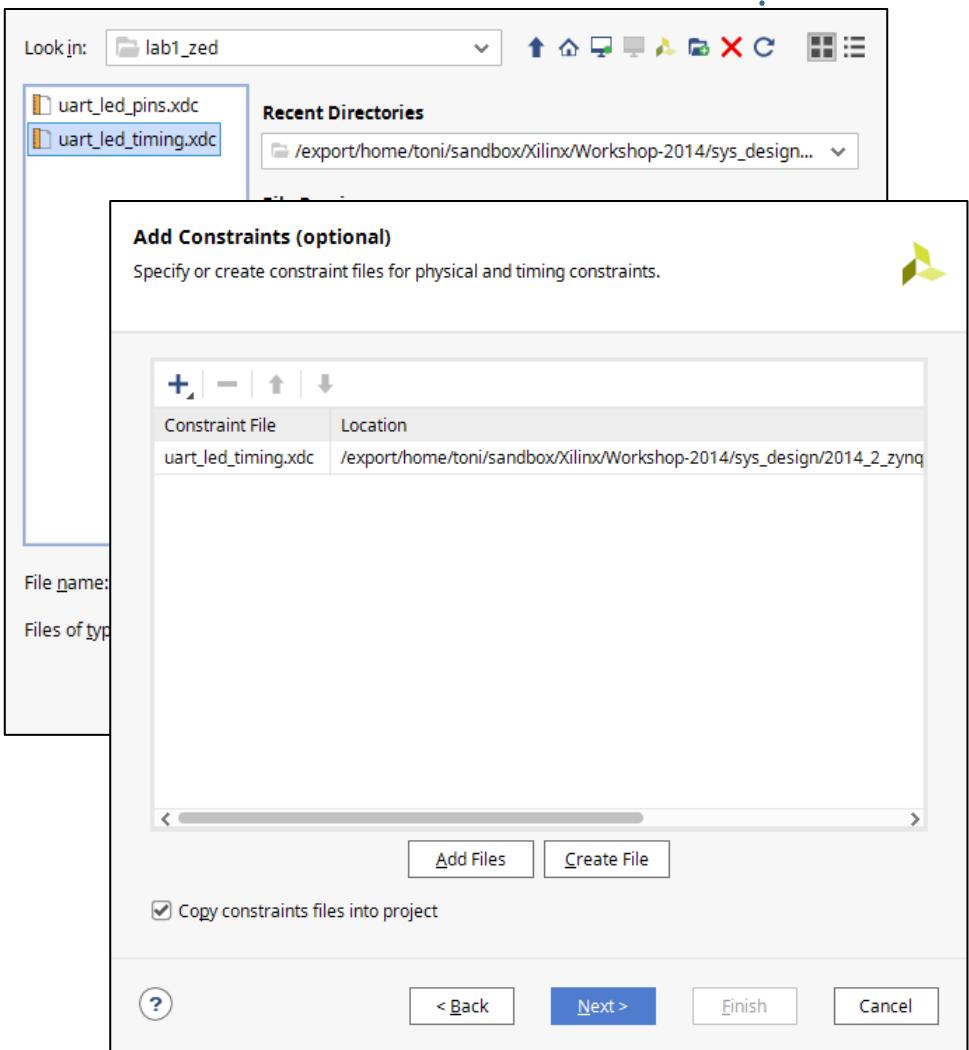
- Add HDL sources
  - Select HDL sources
  - Copy sources into project
  - tb\_XXX sources are simulation definitions



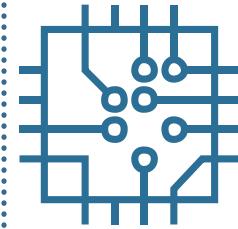
# Vivado: Create new project



- Add Constraints
  - Select constraint definitions XDC
  - Copy constraints into project
  - Multiple constraint sets of same type
  - One is active



# Vivado: Create new project



- FPGA device / board
  - Select correct FPGA board / device
  - Speed information
  - I/O pin assignment
  - Predefined constraints

**Default Part**  
Choose a default Xilinx part or board for your project.

Parts | Boards Update Board Repositories

Reset All Filters

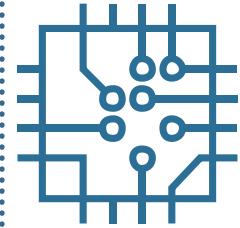
Vendor: All Name: All

Search:  Q-

Display Name	Preview	Vendor	File
Kintex-Ultrascale Alphadata board		alpha-data.com	1
ZedBoard Zynq Evaluation and Development Kit <a href="#">Add Daughter Card</a> <a href="#">Connections</a>		em.avnet.com	1
Artix-7 AC701 Evaluation Platform <a href="#">Add Daughter Card</a> <a href="#">Connections</a>		xilinx.com	1
Alveo U200 Data Center Accelerator Card		xilinx.com	1
Alveo U250 Data Center Accelerator Card		xilinx.com	1

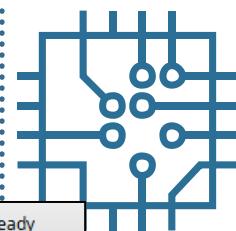
? < Back Next > Finish Cancel

# Vivado: Project manager



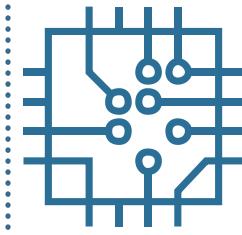
- Manage source, customize IP, and view project details in the Project Summary window
- Flow navigator
- Source view:
  - Hierarchical display of sources
  - IP sources and library view
  - Access to constraint sources
- Project summary
  - Report device utilization, timing, ...
- Consoles: tcl, messages, reports,...

# Vivado: Project manager

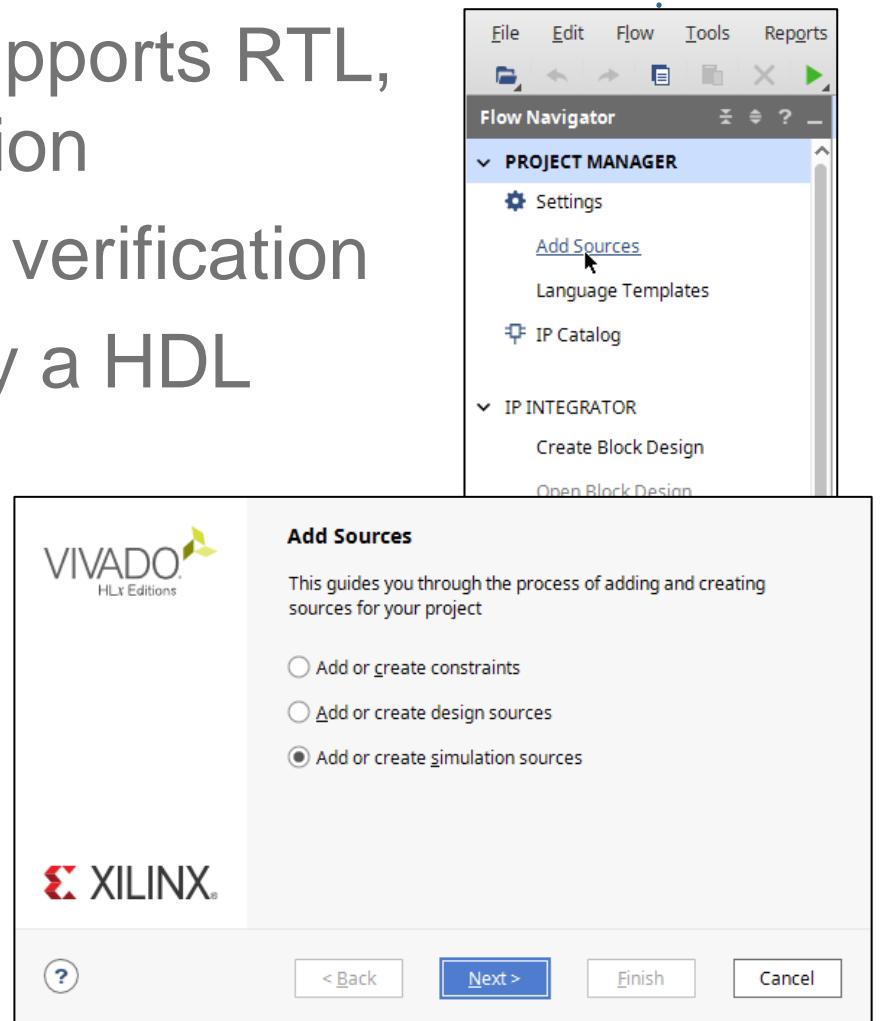


The screenshot shows the Xilinx Vivado 2019.1 software interface. The left sidebar contains a tree view of project management, IP integrator, simulation, RTL analysis, synthesis, implementation, and program/debug options. The main area displays the 'PROJECT MANAGER - UART' window. This window includes a 'Sources' panel listing design sources like 'uart\_led.v', constraints, simulation sources, and utility sources. Below it is a 'Properties' panel with a message to 'Select an object to see properties'. At the bottom is a 'Design Runs' panel showing two entries: 'synth\_1' and 'impl\_1', both in the 'Not started' state. On the right, a 'Project Summary' panel provides an overview of the project settings, including project name (UART), location (/export/home/toni/sandbox/Xilinx/Vivado/UART), product family (Zynq-7000), and board part (ZedBoard Zynq Evaluation and Development Kit). It also lists top module (uart\_led), target language (Verilog), and simulator language (Verilog). A 'Board Part' section details the board's display name, part name, revision, connectors, repository path, and URL.

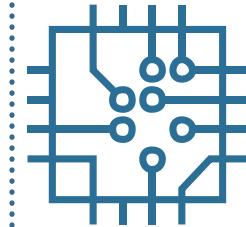
# Vivado: HDL simulation



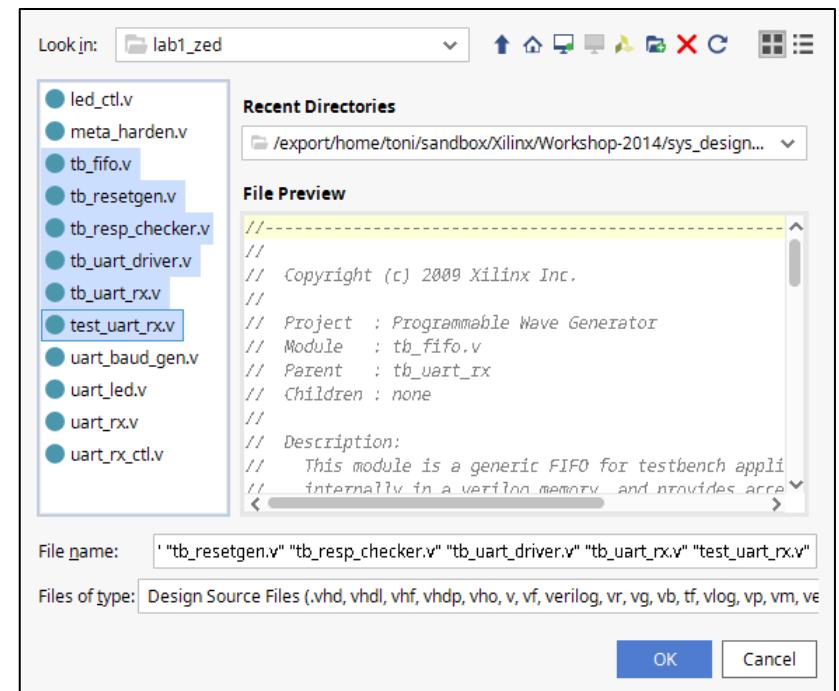
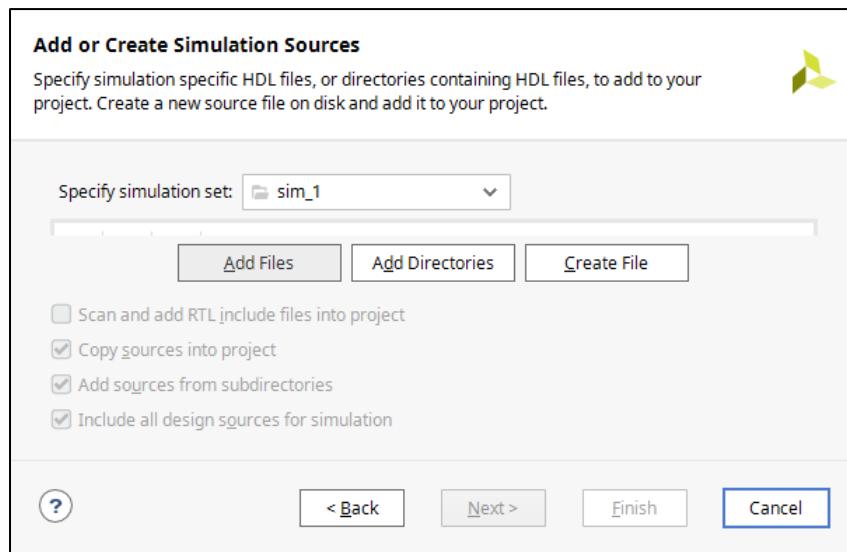
- Vivado XSIM simulator supports RTL, netlist, and timing simulation
- First stage of HDL design verification
- Simulation is controlled by a HDL wrapper called testbench
- Add testbench sources



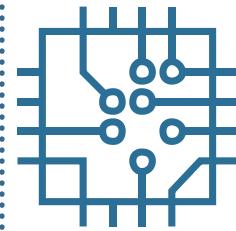
# Vivado: HDL simulation



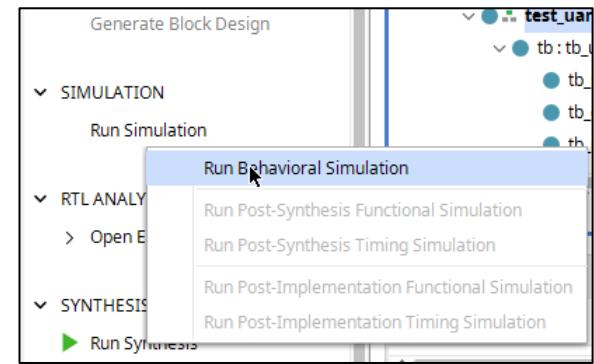
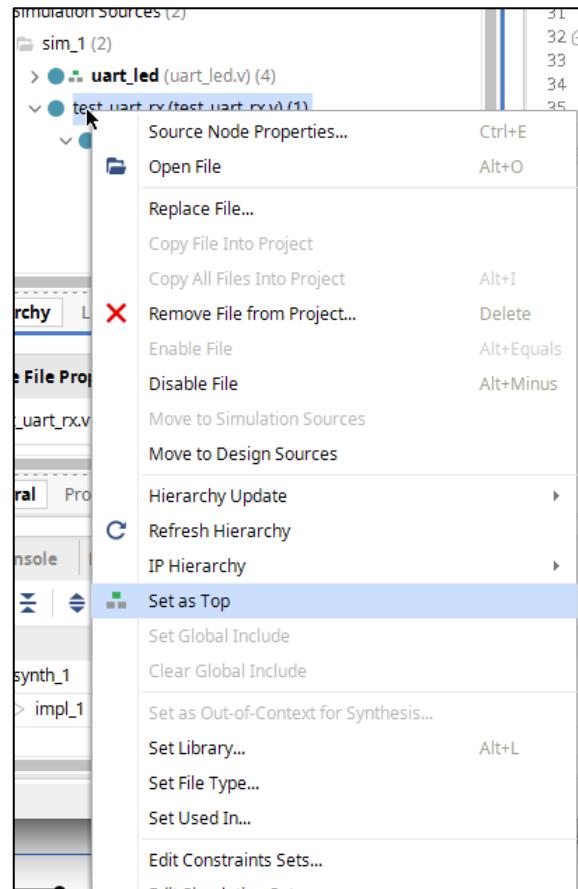
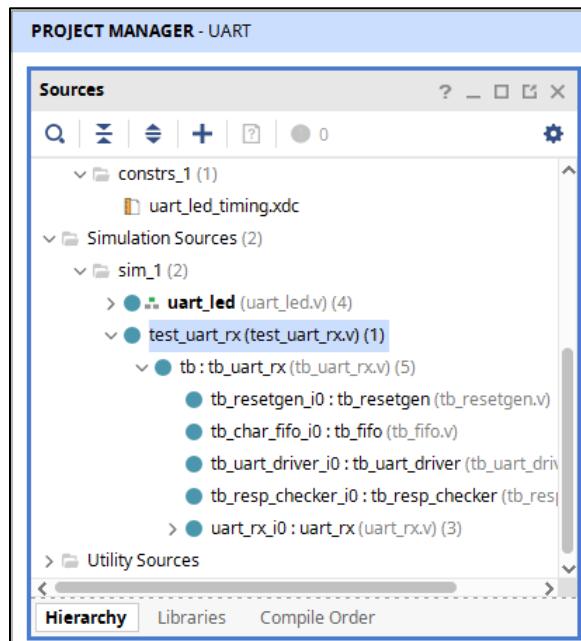
- Create separate simulation set
- Simulation set is compiled to a program
- New set requires recompilation – time consuming



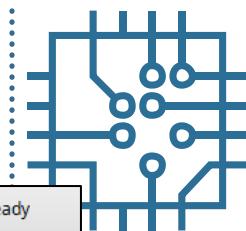
# Vivado: HDL simulation



- Set new simulation set as top module
- Start behavioral simulation



# Vivado: HDL simulation



SIMULATION - Behavioral Simulation - Functional - sim\_1 - test\_uart\_rx

uart\_led.v | uart\_rx.v | Untitled 1

Scope Sources Ob Prot ? □ □

Scope Sources Ob Prot ? □ □

Scope Sources Ob Prot ? □ □

File Edit Flow Tools Reports Window Layout View Run Help Quick Access Ready Default Layout

Flow Navigator PROJECT MANAGER IP Catalog IP INTEGRATOR SIMULATION RTL ANALYSIS SYNTHESIS IMPLEMENTATION PROGRAM AND DEBUG

Run Simulation Open Elaborated Design Run Synthesis Open Synthesized Design Run Implementation Open Implemented Design Generate Bitstream Open Hardware Manager

10 us

1,000,000 ns

0 ns 500 ns 1,000 ns

Name Value

Name	Value
i[31:0]	X
j[31:0]	X
char_to_xx	XX
string[0:327]	57656c636f6d652074f2058696c696...
NUM_CHAR[31:0]	00000029

Tcl Console Messages Log

INFO: [USF-XSim-96] XSim completed. Design snapshot 'test\_uart\_rx\_behav' loaded.

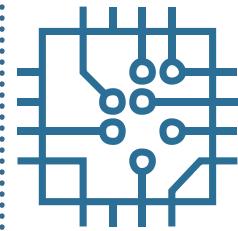
INFO: [USF-XSim-97] XSim simulation ran for 1000ns

launch\_simulation: Time (s): cpu = 00:00:08 ; elapsed = 00:00:15 . Memory (MB): peak = 7326.012 ; gain = 549.035 ; free

Type a Tcl command here

Sim Time: 1 us

# Vivado: HDL simulation



- Select modules and add signals to monitor
- Change signal radix for easier inspection

Name	Design Unit	Blk^
test_uart_rx	test_uart_rx	Verilog
tb	tb_uart_rx	Verilog
tb_rese	tb_resetgen	Verilog
tb_char	tb_fifo	Verilog
tb_uart	tb_uart_drive	Verilog
tb_resp	tb_resp_check	Verilog
uart_rx	uart_rx	Verilog
glbl	glbl	Verilog

Name	Value	Data Type
clk_rx	0	Logic
rst_clk_rx	0	Logic
rxd_i	1	Logic
rxd_clk_rx	1	Logic
rx_data[7:0]	00	Array
rx_data_rdy	0	Logic
frm_err	0	Logic
baud_x16_en	0	Logic
BAUD_RATE[115200]	115200	Array
CLOCK_RATE	200000	Array

clk\_rx

rst\_clk\_rx

rxd\_i

rxd\_clk\_rx

rx\_data[7:0]

rx\_data\_rdy

frm\_err

baud\_x16\_en

BAUD\_RATE[115200]

CLOCK\_RATE

Add to Wave Window

Log to Wave Database

Show in Wave Window

Go to Source Code

Go to Actual

Radix

Show as Enumeration

Report Drivers

Force Constant...

Force Clock...

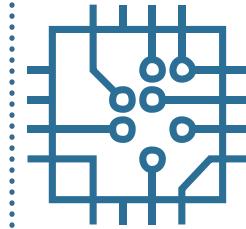
Remove Force

Default Radix

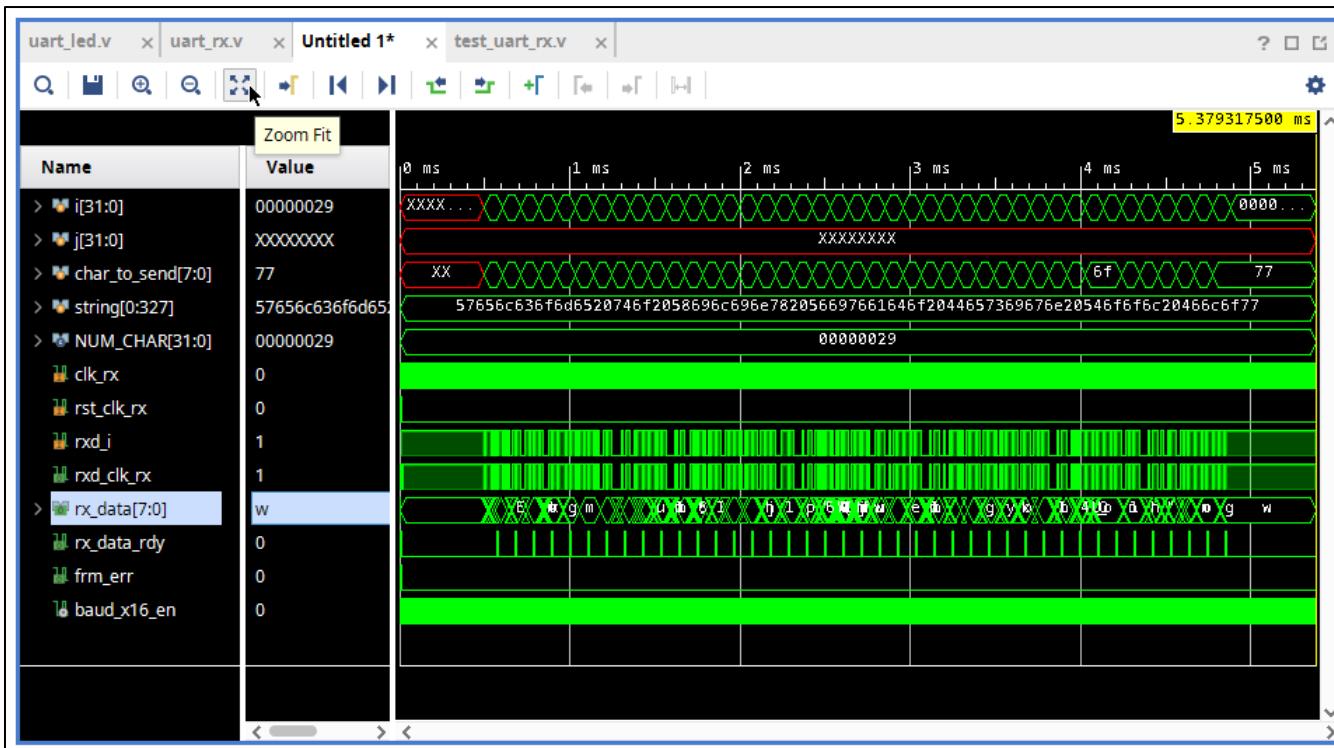
  

Name	Value
i[31:0]	xxxxxxxx
j[31:0]	xxxxxxxx
char_to_send[7:0]	XX
string[0:327]	57656c636f6d65
NUM_CHAR[31:0]	00000029

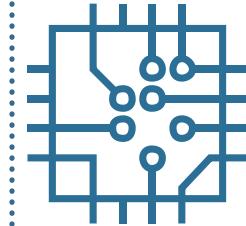
# Vivado: HDL simulation



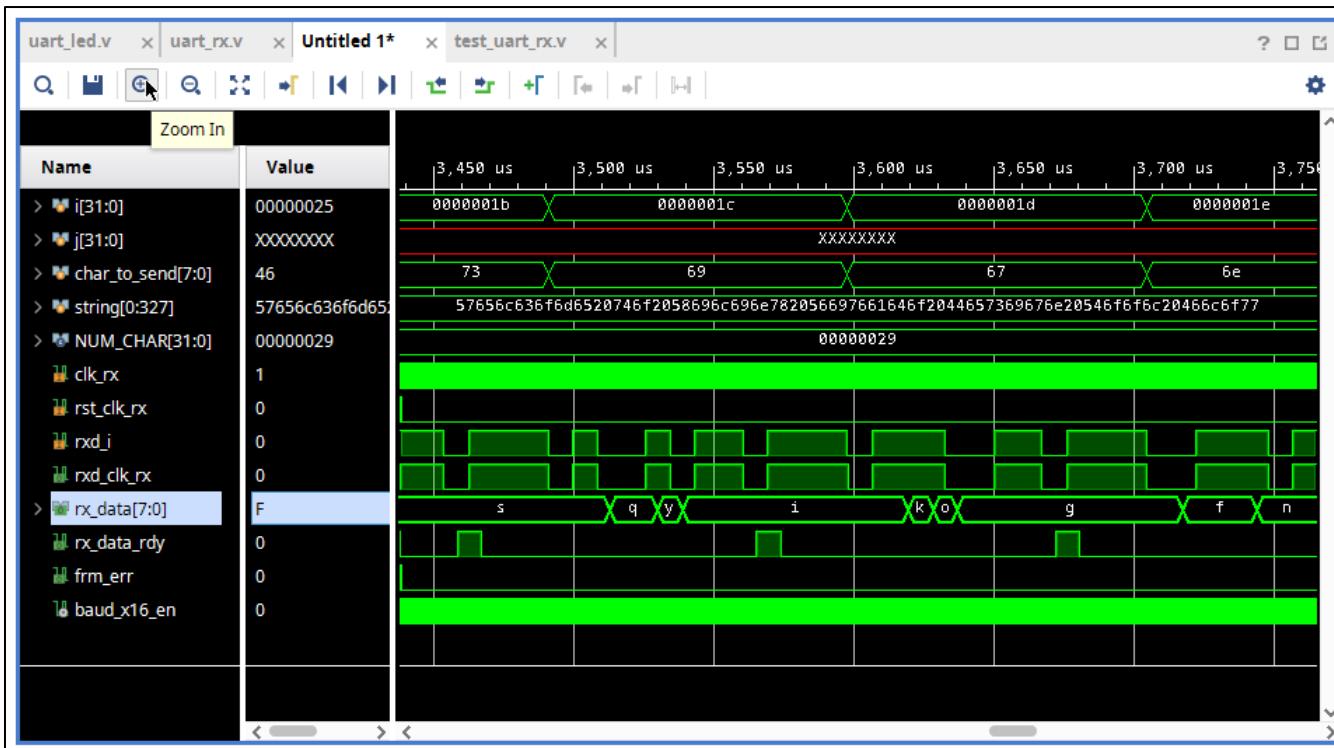
- Restart the simulation with **Restart** button
- Re-run simulation with **Run All** button



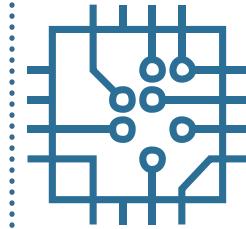
# Vivado: HDL simulation



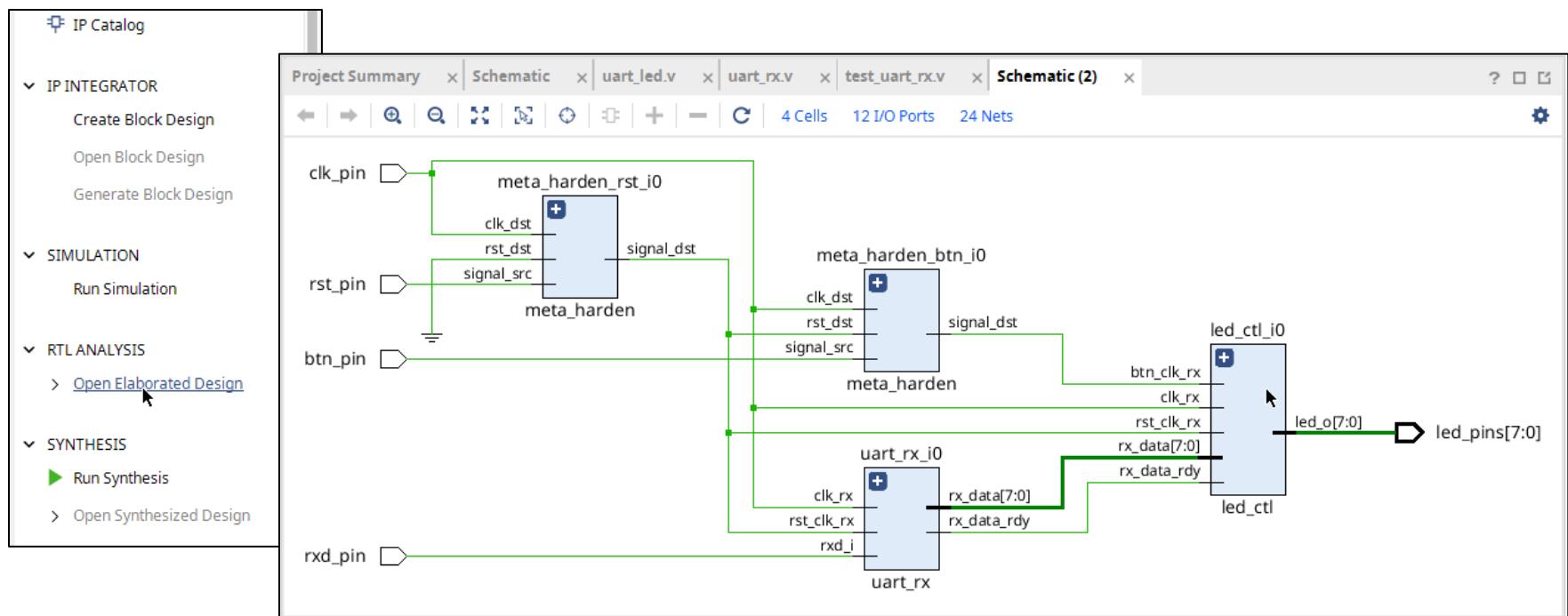
- Use markers, delay measurement, and zoom
- Buses can be expanded to individual signals



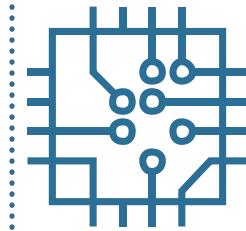
# Vivado: Elaborate RTL design



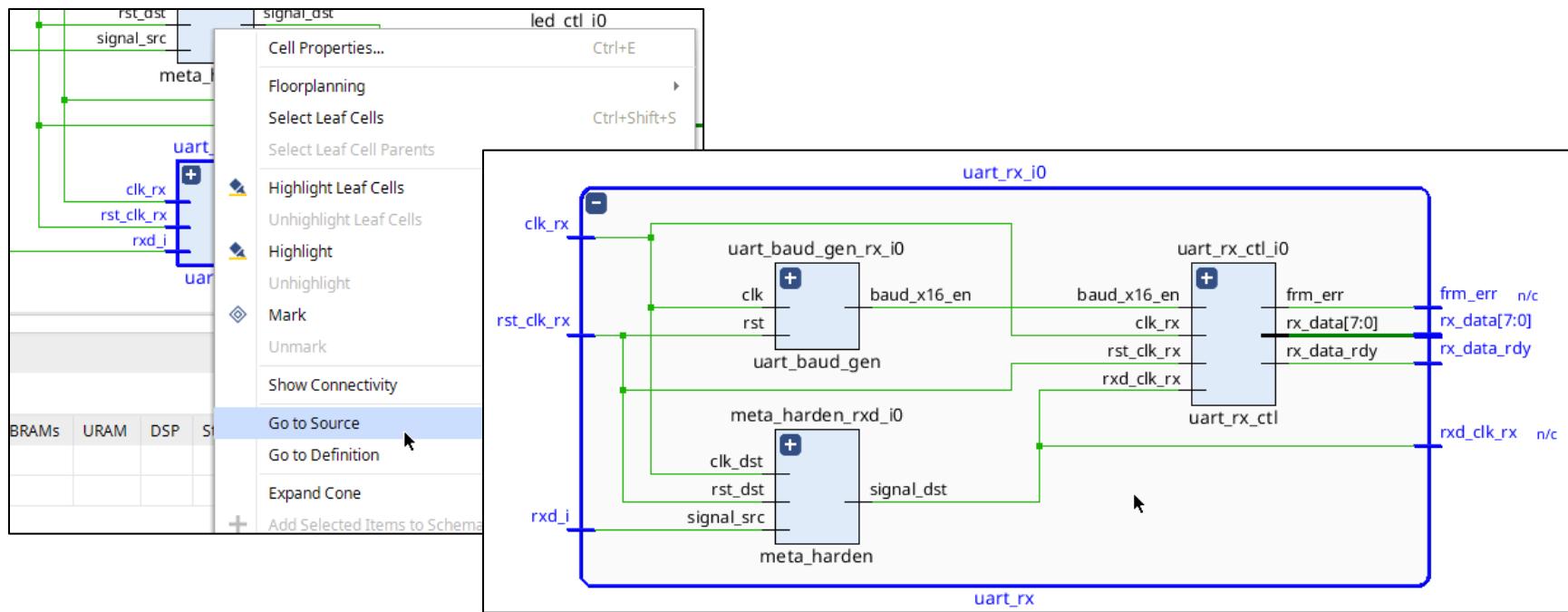
- Perform RTL analysis on the source files
- In Flow Navigator select Open Elaborated Design
- Structure of top level module is visualized



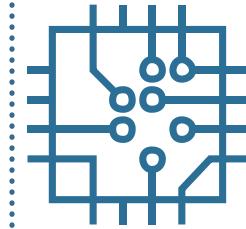
# Vivado: Elaborate RTL design



- HDL code for each module can be searched
- Structure of modules can be examined
- Unconnected signals can be identified



# Vivado: Elaborate RTL design



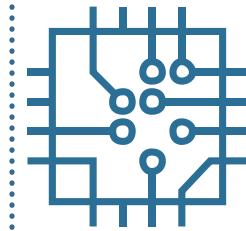
- From Flow Navigator generate noise report
- Several warnings since I/O pins are not defined

The screenshot shows the Vivado IDE interface with the following details:

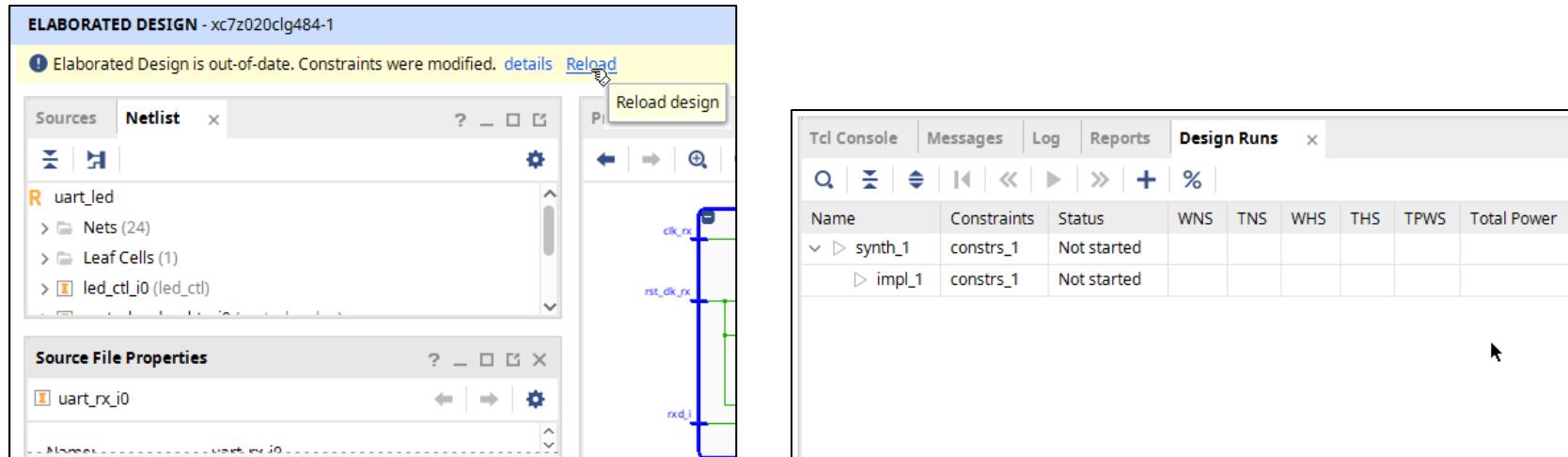
- Left Sidebar (RTL ANALYSIS):**
  - Open Elaborated Design:**
    - Report Methodology
    - [Report DRC](#)
    - [Report Noise](#) (highlighted with a mouse cursor)
  - Schematic:**
    - [Run Synthesis](#)
- Main Window:**
  - Tabs:** Tcl Console, Messages, Log, Reports, Design Runs, **Noise** (selected).
  - Summary:** Messages (2), I/O Bank Details (highlighted).
  - I/O Bank Details Table:**

Name	Port	I/O Std	Vcco	Slew	Dr...	Off-Chip Te...	Rem...	Notes
I/O Bank 33 (0)								
I/O Bank 34 (0)								
I/O Bank 35 (0)								
Unplaced Ports (8)								
led_pins[0]	led_pins[0]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[1]	led_pins[1]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[2]	led_pins[2]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[3]	led_pins[3]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[4]	led_pins[4]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[5]	led_pins[5]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[6]	led_pins[6]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port
led_pins[7]	led_pins[7]	LVCMS18	1.80	SLOW	12	FP_VTT_50		CRITICAL WARNING - Unplaced port

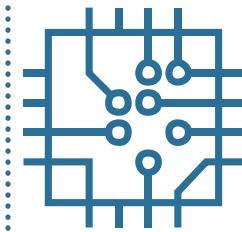
# Vivado: Elaborate RTL design



- Add I/O pin constraints
- Reload the design
- Regenerate noise report

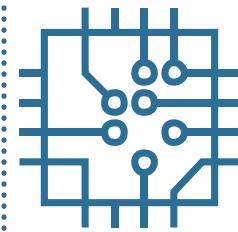


# Outline



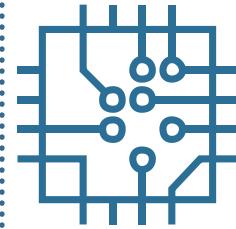
- Part 1
  - FPGA structure and design platform
  - VHDL hardware design in FPGA
  - **Embedded system design on FPGA**
- Part 2
  - IP core development and integration
  - Embedded software design FPGA
- Part 3
  - Software and hardware debugging

# System-on-Chip



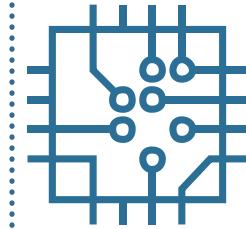
- Complete system implemented in an ASIC
- System-on-Chip (SoC) can include:
  - Processor,
  - Memory (RAM and storage),
  - Other digital devices (timers, interfaces,...),
  - Radio frequency components,
  - Mixed signal devices (ADC, DAC, ...)

# System-on-Chip on FPGA

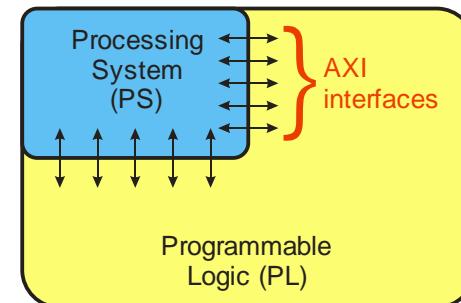
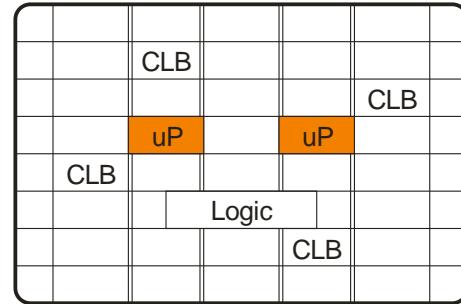


- All-programmable System-on-Chip (APSoC)
- Hardware devices can be modified
- FPGA implementations
  - Soft processor: microBlaze, Nios, Leon, or1200, ...
  - Hard processor: PowerPC, ARM
- Dedicated IP cores
- On field upgrades

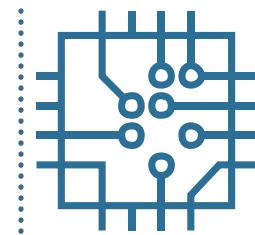
# Hard processor based FPGA



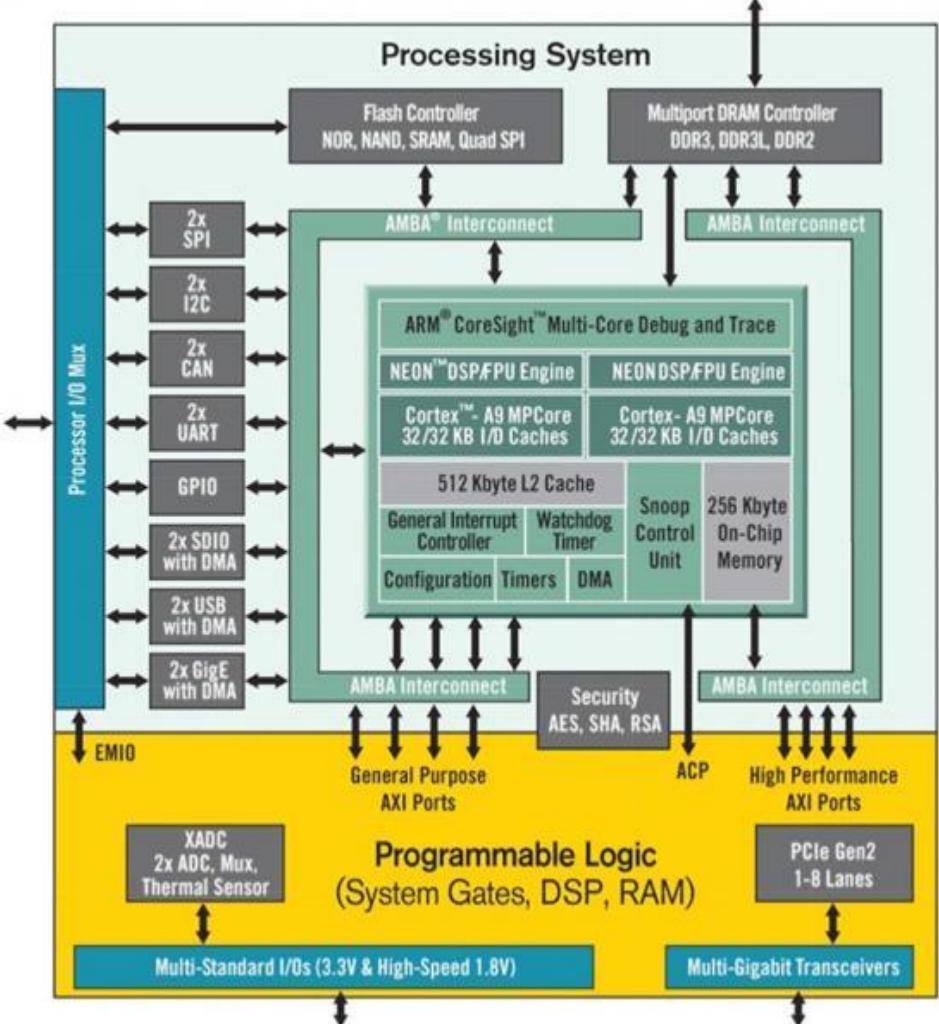
- PowerPC-based system
  - CPU in the middle of FPGA
  - FPGA fabric between CPU and device IO
- ARM-based system
  - CPU boots independently
  - Dual ARM Cortex-A9
  - I/O peripherals



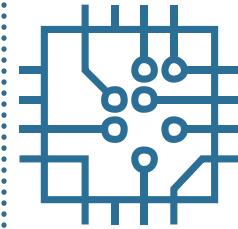
# Zynq-7000 family



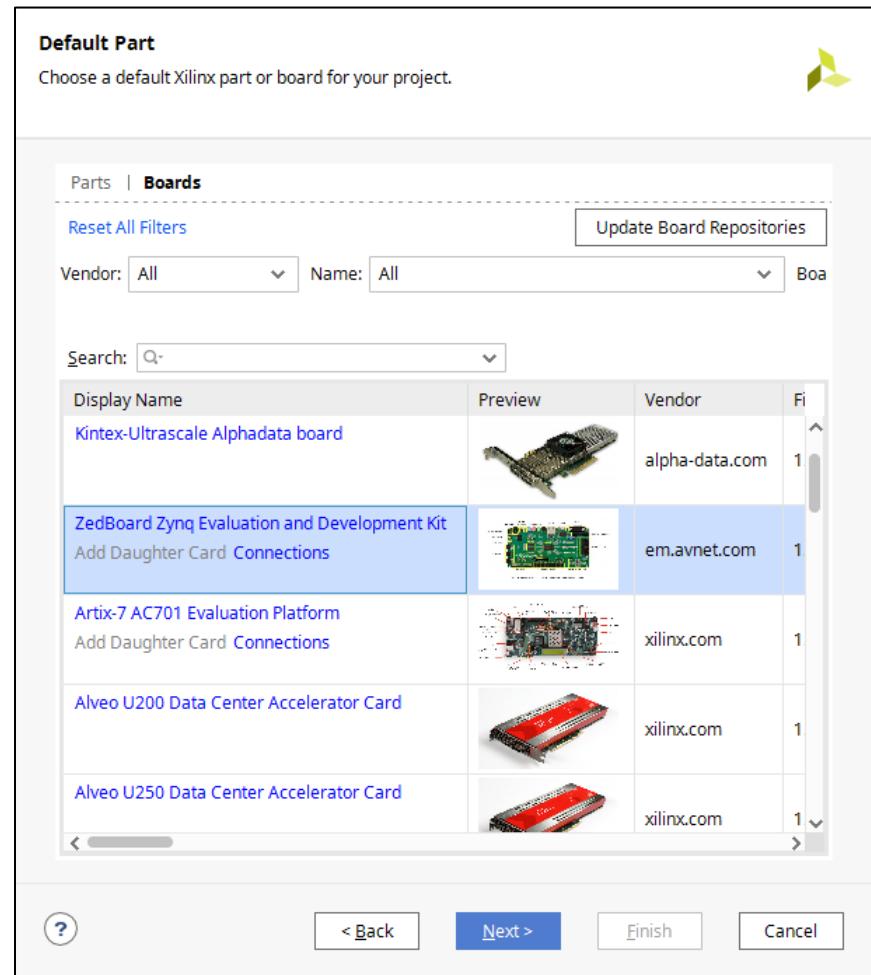
- ARM-based system
  - Dual ARM Cortex-A9
  - Integrated memory controllers
  - I/O peripherals
- Tightly integrated programmable logic
  - Extend the processing system
- Flexible array IO
  - Wide range of external multi-standard I/O
  - ADC inputs



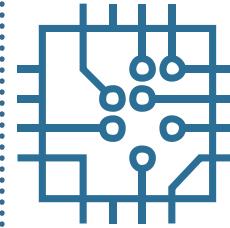
# Vivado: Create embedded system



- Start with empty project
  - No HDL sources
- FPGA device / board
  - Select correct FPGA board / device
  - Speed information
  - I/O pin assignment
  - Predefined constraints
- IP integrator
  - Create block design



# Simple embedded system



- Start with empty system
- First system consist of CPU and UART

The screenshot shows the Vivado HLx Editions software interface. On the left, a sidebar titled "Quick Start" offers options like "Create Project >" (with a red arrow pointing to it), "Open Project >", and "Open Example Project >". Below that is a "Tasks" section with links to "Manage IP >", "Open Hardware Manager >", and "XHub Stores >". The main area is titled "VIVADO" and shows a "New Project <@martin>". It has two tabs: "Project Name" and "Project Type". In the "Project Name" tab, the "Project name:" field contains "project\_1" and the "Project location:" field contains "/export/home/toni/work/TetraMax/System.Design/seminar". A checked checkbox "Create project subdirectory" is present. The "Project will be created at: .../System.Design/seminar/project\_1" path is shown below. In the "Project Type" tab, a radio button "RTL Project" is selected. Its description states: "You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis." A checked checkbox "Do not specify sources at this time" is also present. Other project type options include "Post-synthesis Project", "I/O Planning Project", "Imported Project", and "Example Project", each with its own description and checkbox.

File Flow Tools Window Help Q+ Quick Access

VIVADO HLx Editions

New Project <@martin>

Project Name

Enter a name for your project and specify a directory where the project data

Project name: project\_1

Project location: /export/home/toni/work/TetraMax/System.Design/seminar

Create project subdirectory

Project will be created at: .../System.Design/seminar/project\_1

Project Type

Specify the type of project to create.

RTL Project  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
 Do not specify sources at this time

Post-synthesis Project  
You will be able to add sources, view device resources, run design analysis, planning and implementation.  
 Do not specify sources at this time

I/O Planning Project  
Do not specify design sources. You will be able to view part/package resources.

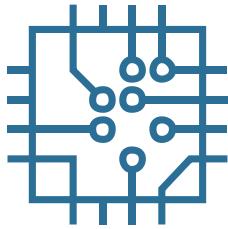
Imported Project  
Create a Vivado project from a Synplify, XST or ISE Project File.

Example Project  
Create a new Vivado project from a predefined template.

Tcl Console

< Back Next > ? ? < Back Next > ? ? Finish Cancel

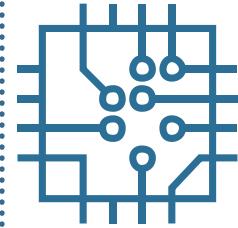
# Simple embedded system



- Select hardware platform (board)

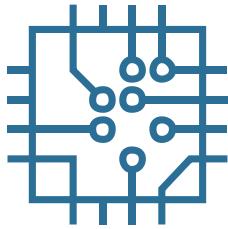
The screenshot shows the Xilinx Vivado software interface. The title bar includes 'File', 'Flow', 'Tools', 'Window', 'Help', and a 'Quick Access' search bar. The main window is titled 'New Project <@martin>' and features the 'VIVADO' and 'XILINX' logos. A central panel is titled 'Default Part' with the sub-instruction 'Choose a default Xilinx part or board for your project.' Below this, there are tabs for 'Parts' and 'Boards'. The 'Boards' tab is selected, showing a search bar and filters for 'Vendor' (set to 'em.avnet.com'), 'Name' (set to 'ZedBoard Zynq Evaluation and Development Kit'), and 'Board Rev' (set to 'Latest'). A 'Reset All Filters' button and an 'Install/Update Boards' button are also present. A table lists the selected board details: Display Name 'ZedBoard Zynq Evaluation and Development Kit', Preview image of the board, Vendor 'em.avnet.com', File Version '1.4', and Part 'xc7z020clg484-1'. Below the table, there are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

# Simple embedded system

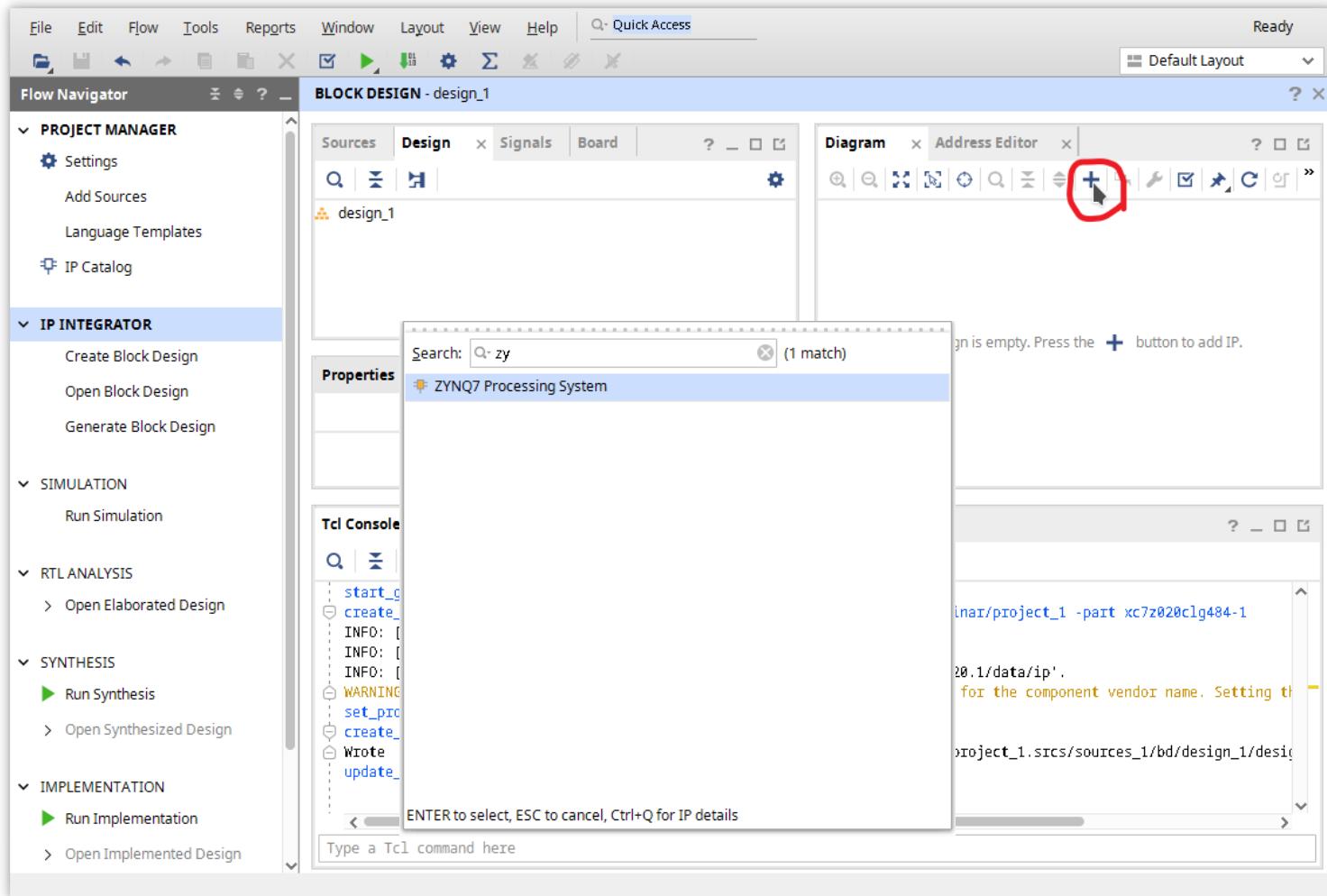


- Project manager (Create Block Design)

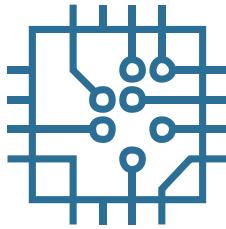
# Simple embedded system



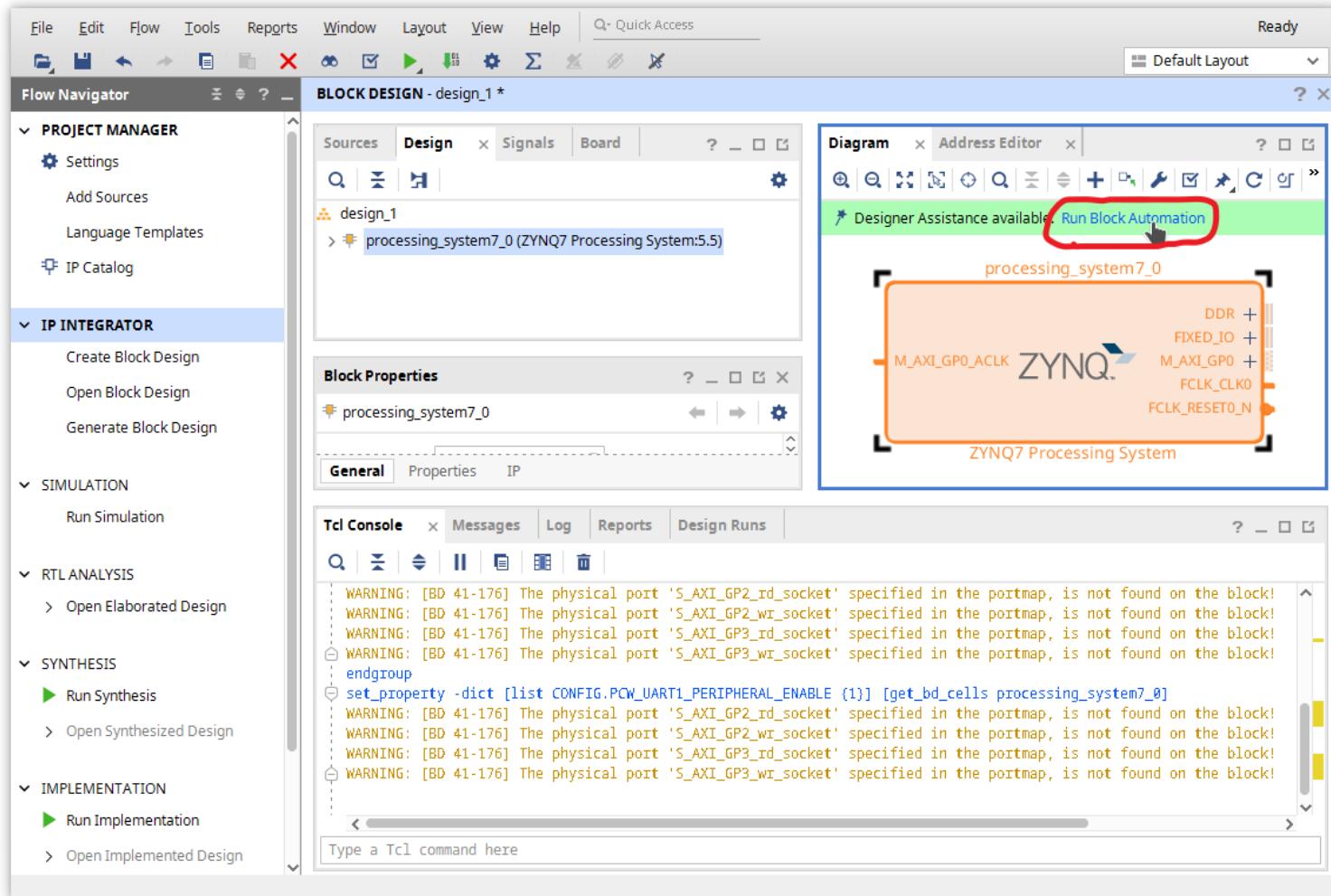
- Add Processor core (ZYNQ)



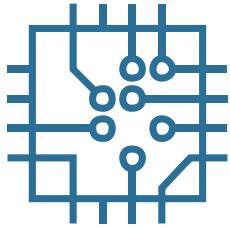
# Simple embedded system



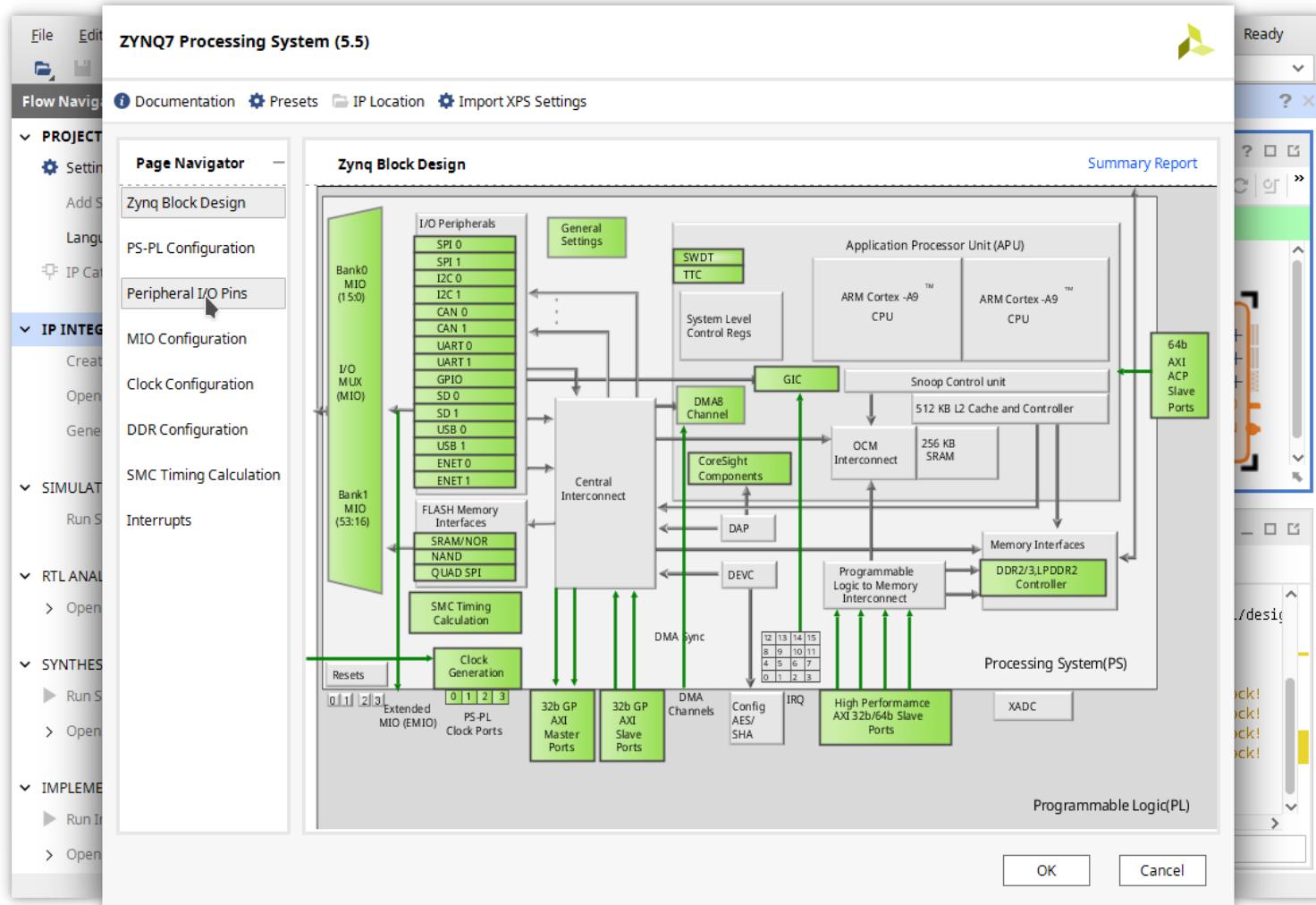
- Customize ZYNQ processor



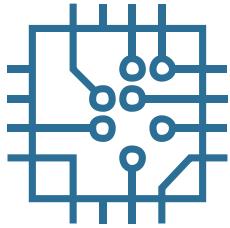
# Simple embedded system



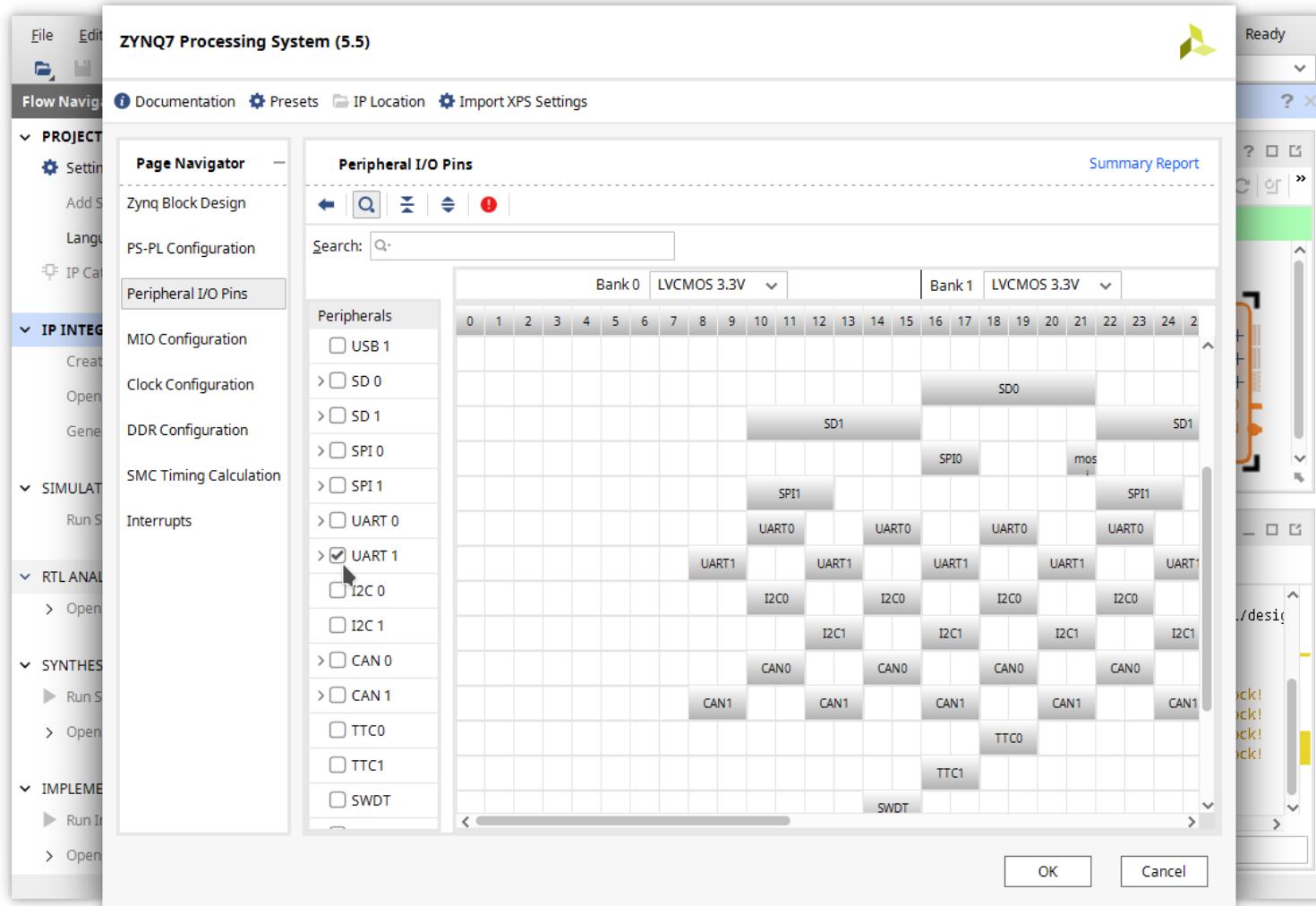
- Customize ZYNQ processor



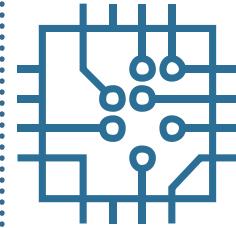
# Simple embedded system



- Customize ZYNQ processor



# Simple embedded system



- Connect signals (Block Automation)

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.



Q | F | ▾

✓ All Automation (1 out of 1 selected)  
✓ processing\_system7\_0

**Description**

This option sets the board preset on the Processing System. All current properties will be overwritten by the board preset. This action cannot be undone. Zynq7 block automation applies current board preset and generates external connections for FIXED\_IO, Trigger and DDR interfaces.

NOTE: Apply Board Preset will discard existing IP configuration - please uncheck this box, if you wish to retain previous configuration.

Instance: /processing\_system7\_0

**Options**

Make Interface External: FIXED\_IO, DDR

Apply Board Preset:

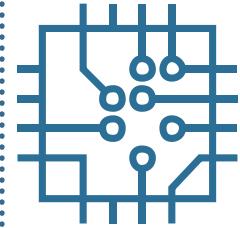
Cross Trigger In:

Cross Trigger Out:

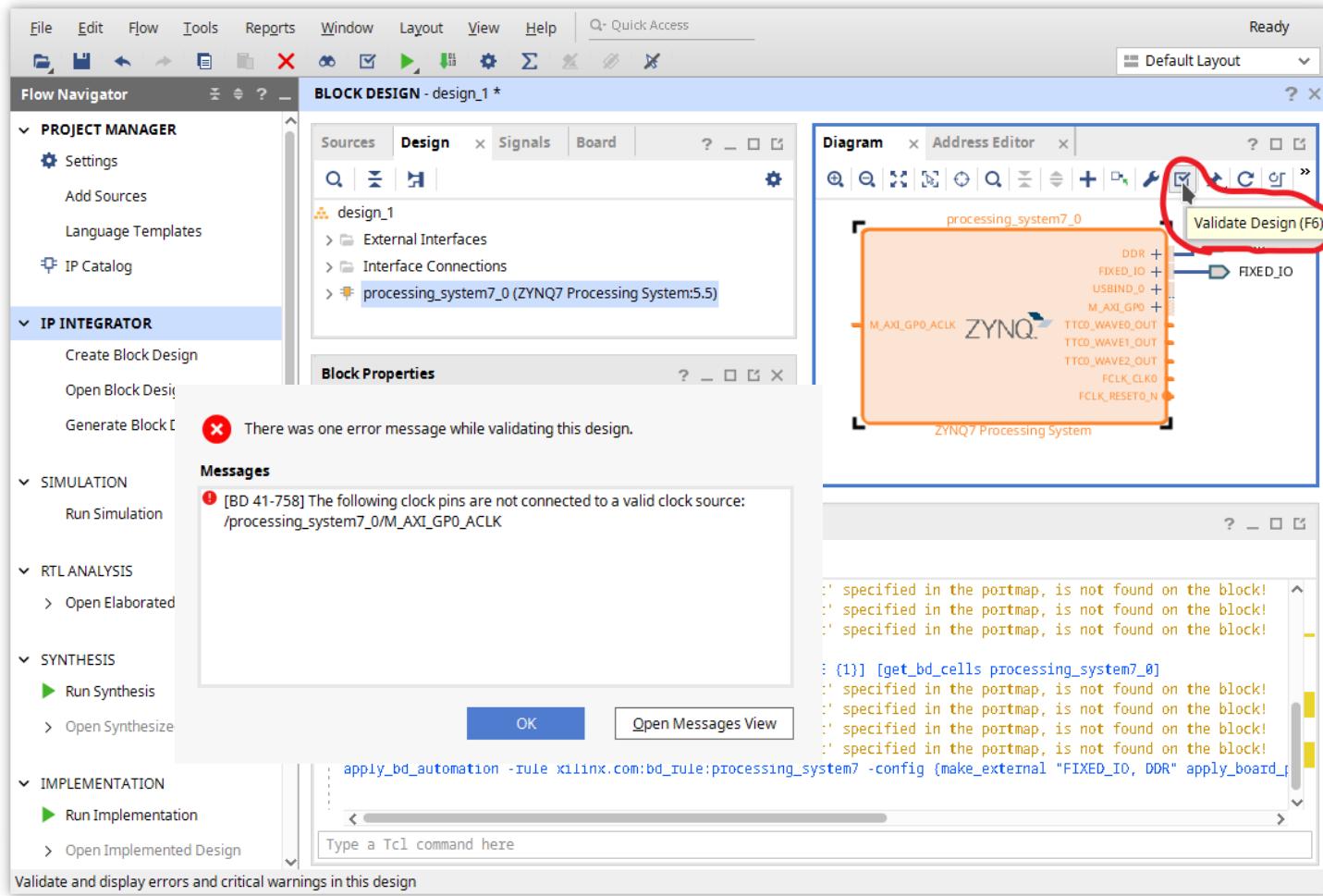
?

OK Cancel

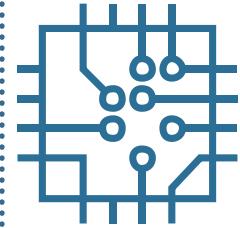
# Simple embedded system



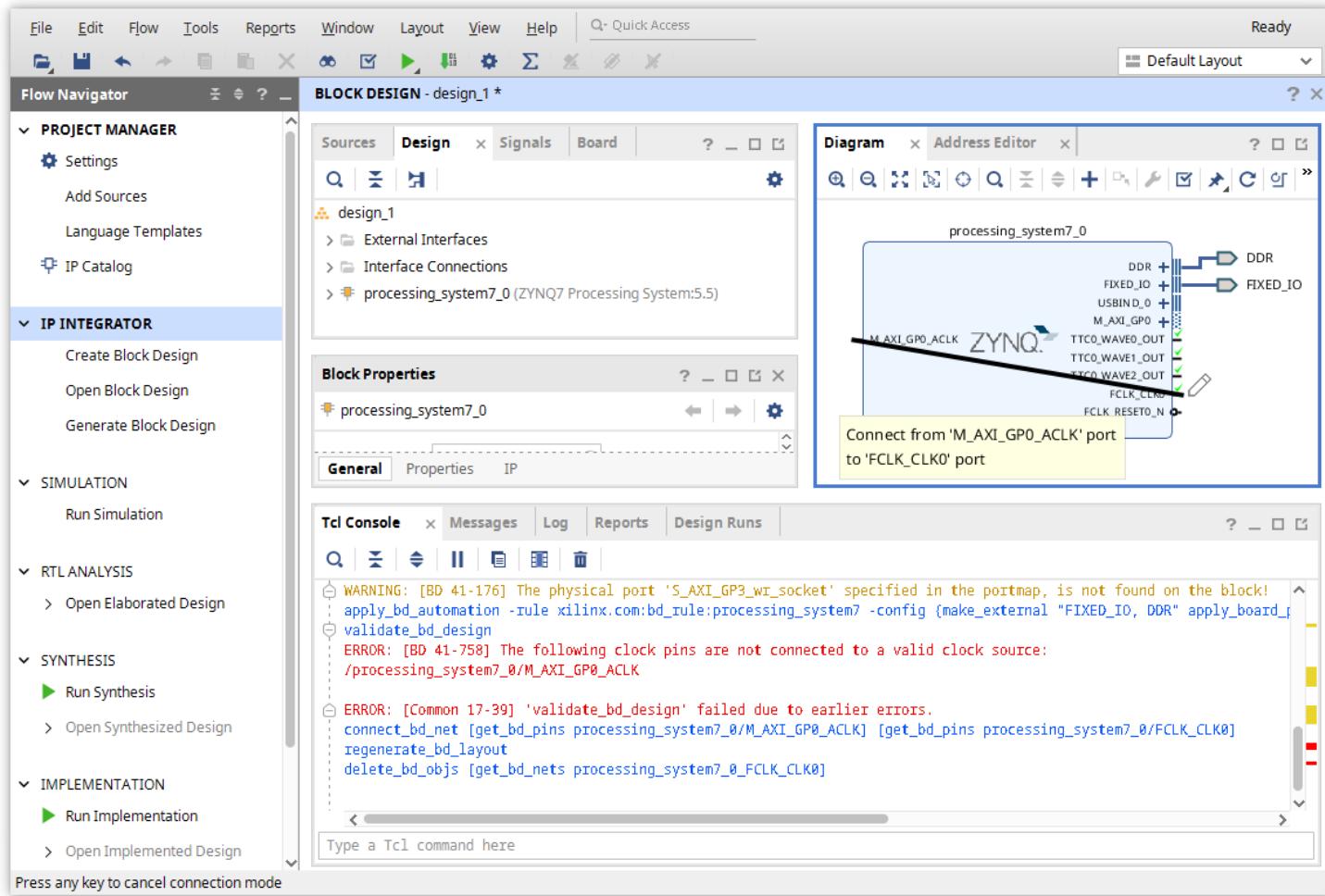
- Design validation



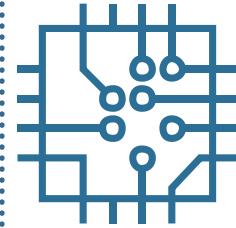
# Simple embedded system



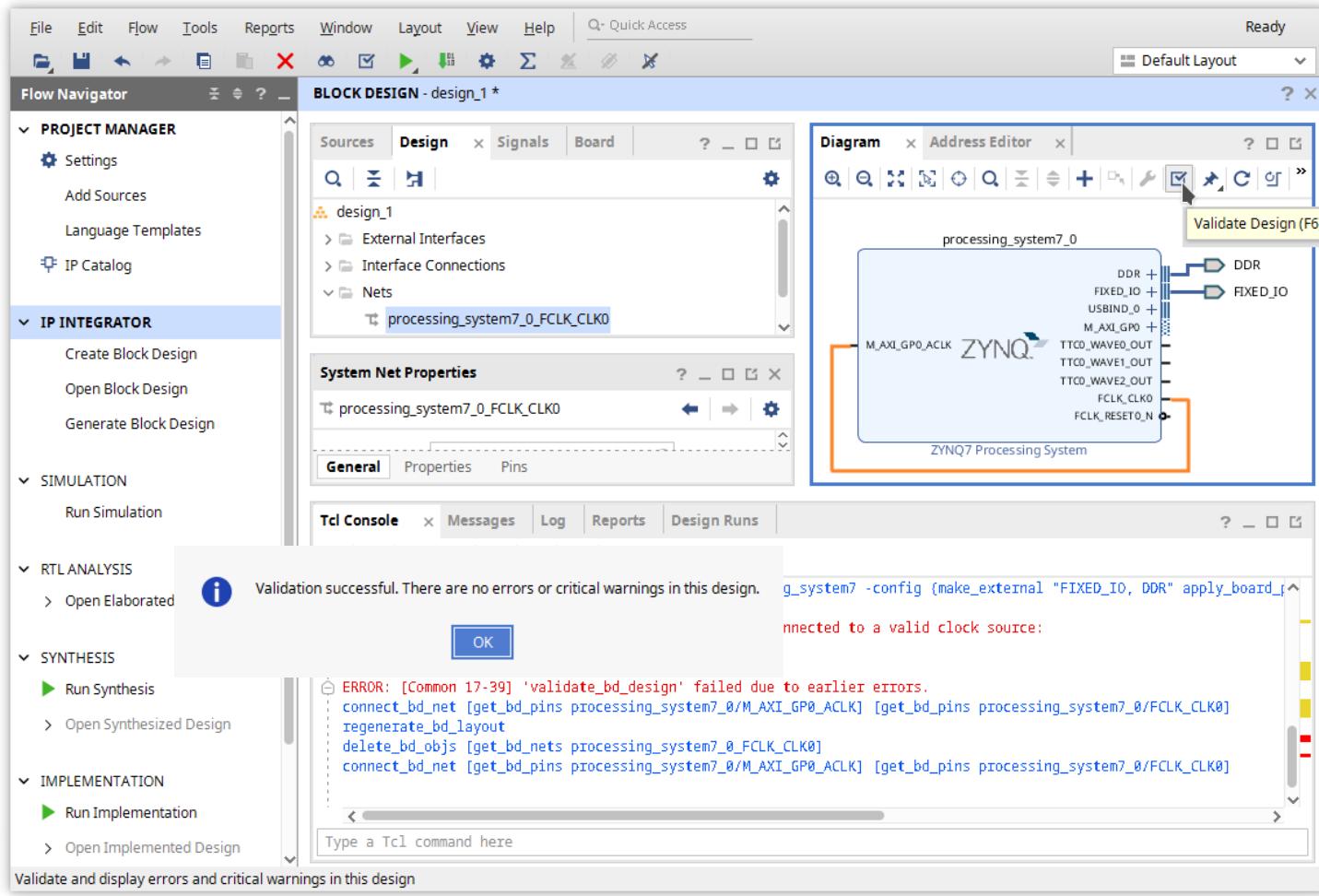
- Manually connect clock



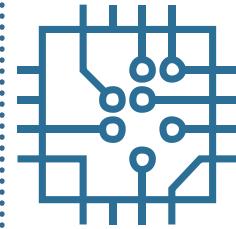
# Simple embedded system



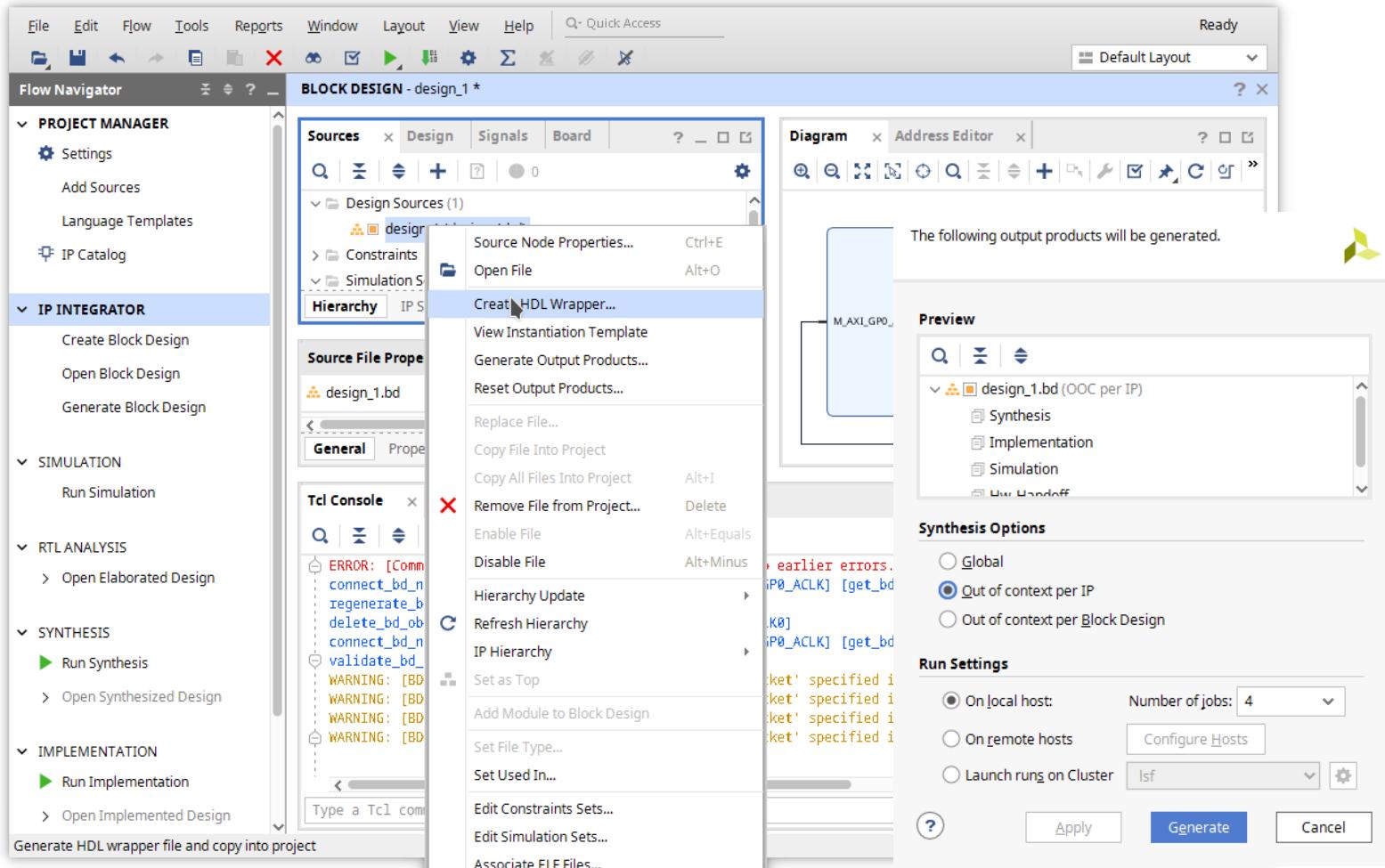
- Design validation



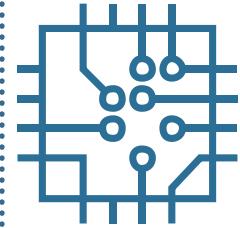
# Simple embedded system



- Create HDL wrapper



# Simple embedded system



- Synthesis, implementation, generation

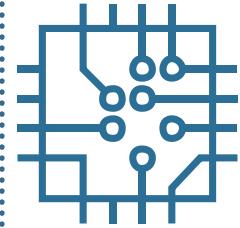
The screenshot shows the Xilinx Vivado IDE interface with three main panels:

- Flow Navigator:** On the left, it lists various design flows. The "SYNTHESIS" section is circled in red and labeled 1. It contains "Run Synthesis".
- BLOCK DESIGN - design\_1:** The central panel displays the "BLOCK DESIGN" window for "design\_1". It shows a "Sources" tab listing "design\_1\_wrapper" and "design\_1\_i: design\_1". The "Diagram" tab shows a ZYNQ7 Processing System block with various pins like DDR, FIXED\_IO, USBIN\_D\_0, M\_AXI\_GPO, TTCO\_WAVE0\_OUT, TTCO\_WAVE1\_OUT, TTCO\_WAVE2\_OUT, FCLK\_CLK0, and FCLK\_RESETO\_N.
- Tcl Console:** The bottom panel shows the "Tcl Console" window with a log of synthesis commands and their execution times. The log includes:

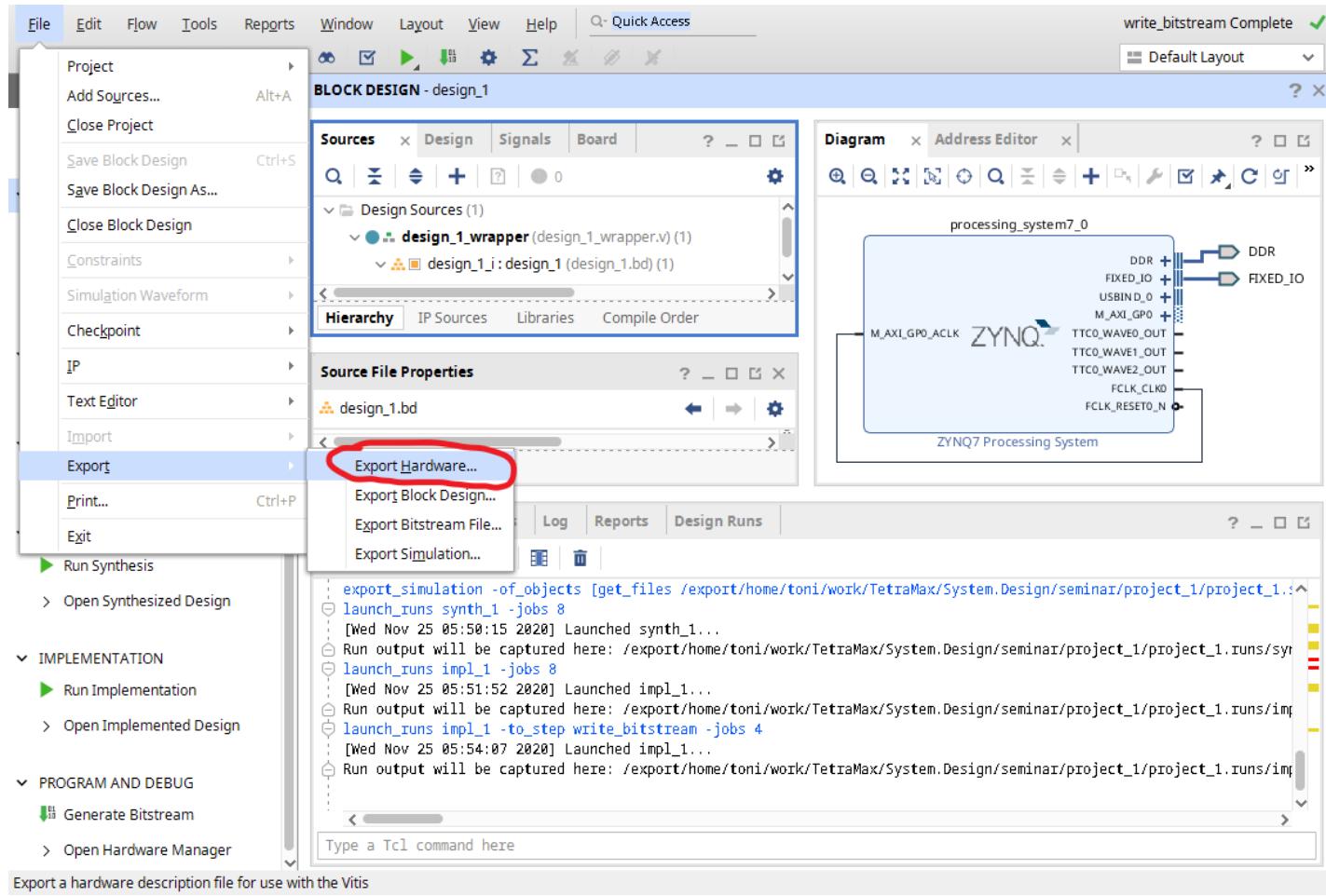
```
catch { config_ip_cache -export [get_ips -all design_1_processing_system7_0_0] }
export_ip_user_files -of_objects [get_files /export/home/toni/work/TetraMax/System.Design/seminar/project_1/project_1_processing_system7_0_0_processing_system7_0_0_wrapper.v}
create_ip_run [get_files -of_objects [get_fileset sources_1]] /export/home/toni/work/TetraMax/System.Design/seminar/project_1/project_1_processing_system7_0_0_processing_system7_0_0_wrapper
launch_runs design_1_processing_system7_0_0_synth_1 -jobs 4
[Wed Nov 25 05:45:00 2020] Launched design_1_processing_system7_0_0_synth_1...
Run output will be captured here: /export/home/toni/work/TetraMax/System.Design/seminar/project_1/project_1.runs/design_1_processing_system7_0_0_synth_1
export_simulation -of_objects [get_files /export/home/toni/work/TetraMax/System.Design/seminar/project_1/project_1_processing_system7_0_0_processing_system7_0_0_wrapper]
launch_runs synth_1 -jobs 8
[Wed Nov 25 05:50:15 2020] Launched synth_1...
Run output will be captured here: /export/home/toni/work/TetraMax/System.Design/seminar/project_1/project_1.runs/synth_1
```

Red numbers 1, 2, and 3 are overlaid on the Flow Navigator to indicate the sequence of steps: 1. Run Synthesis, 2. Run Implementation, 3. Generate Bitstream.

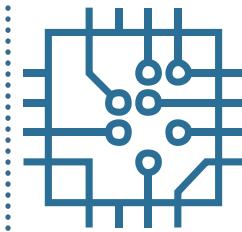
# Simple embedded system



- Export hardware description to Vitis

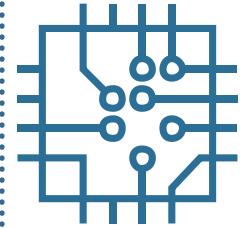


# Outline



- Part 1
  - FPGA structure and design platform
  - VHDL hardware design in FPGA
  - Embedded system design on FPGA
- Part 2
  - Embedded software design FPGA
  - IP core development and integration
- Part 3
  - Software and hardware debugging

# Simple embedded system



- Export hardware description to Vitis

**VIVADO**  
HLx Editions

**Export Hardware Platform**

This wizard will guide you through the export of a hardware platform for use in the Vitis or PetaLinux software tools.

To export a hardware platform, you will need to specify the platform properties.

**Platform type**

- Fixed  
A platform supporting embedded software tools.
- Expandable  
A platform supporting accelerators.

**Output**

Set the platform properties to inform downstream tools.

**Files**

Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

XSA file name:  X

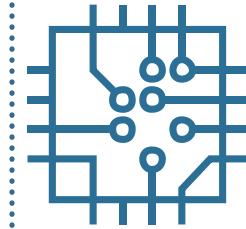
Export to:  X ...

The XSA will be written to: /export/home/toni/work/TetraMax/System.Design/seminar/project\_1/design\_1\_wrapper.xsa

**XILINX**

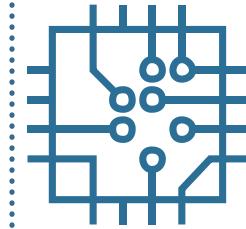
< Back Next > Finish Cancel

# Vitis: Software Development Kit



- Unified software development environment
- Based on eclipse
- Requires hardware definition files (Vivado)
- Supports different types of applications:
  - Standalone
  - freeRTOS
  - Petalinux
- Programming tools (over JTAG)
- Debugging tools

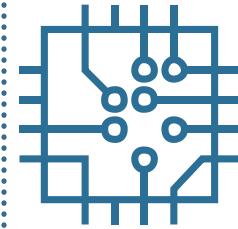
# Vitis: create platform project



- From hardware definition file build platform description

The screenshot shows the Vitis IDE interface. On the left, the main window has a green border and displays the Xilinx Vitis logo. The central area features the text "VITIS IDE". Below this, there are three main sections: "PROJECT", "PLATFORM", and "RESOURCES". Under "PROJECT", there are links for "Create Application Project", "Create Platform Project" (which is highlighted with a red arrow), and "Create Library Project". Under "PLATFORM", there are links for "Add Custom Platform" and "Import Project". Under "RESOURCES", there are links for "Vitis Documentation" and "Xilinx Developer". At the top of the screen, the menu bar includes "File", "Edit", "Search", "Xilinx", "Project", "Window", and "Help". A "Welcome" tab is open in the top-left corner. A modal dialog box titled "Select a directory as workspace" is displayed in the center-right. It contains a "Workspace:" field with the value "/toni/work/TetraMax/System.Design/seminar/workspace", a "Browse..." button, a checkbox for "Use this as the default and do not ask again", and a "Restore other Workspace" link. At the bottom right of the dialog are "Cancel" and "Launch" buttons, with "Launch" being the one currently being clicked, as indicated by a red arrow.

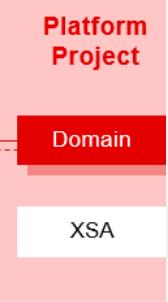
# Vitis: create platform project



- Importing hardware definition file

**Create new platform project**  
Enter a name for your platform project

Project name:   Use default location  
Location:



**Platform Project**

**Platform Project Specification**  
Provide the hardware and software specification for the new platform project

**Hardware Specification**

XSA file:

**Create from hardware specification**  
Create a new platform project from the output to specify options for the kernels, BSPs, as for embedded software developers.

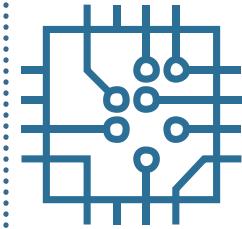
**Create from existing platform**  
Load the platform definition from an existing platform to quickly start your platform project.

Operating system: standalone   
Processor: ps7\_cortexa9\_0

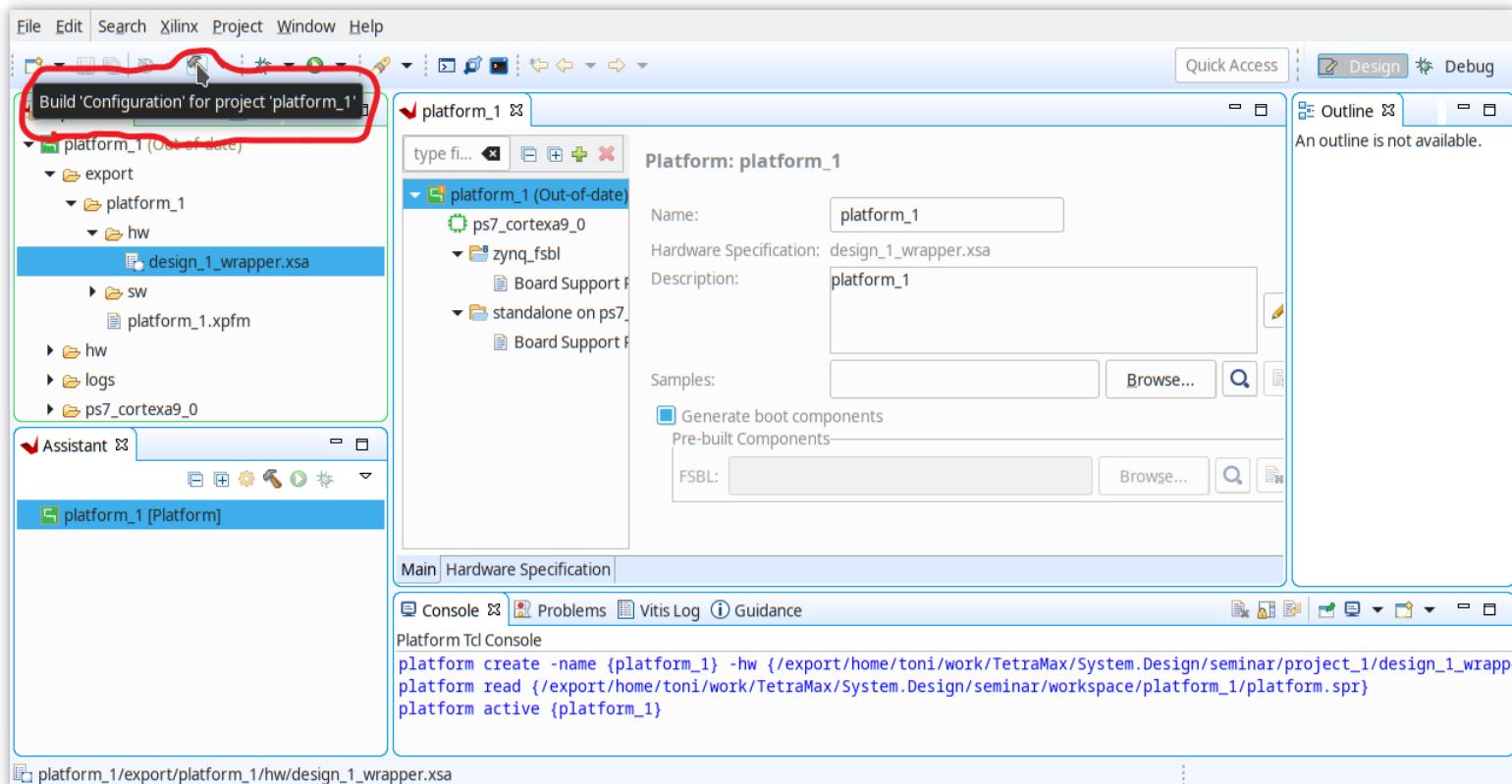
*i* Note: A domain with selected operating system and processor will be added to the platform. The platform project can be modified later to add new domains or change settings.

Generate boot components

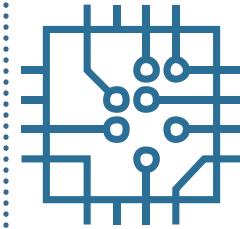
# Vitis: create platform project



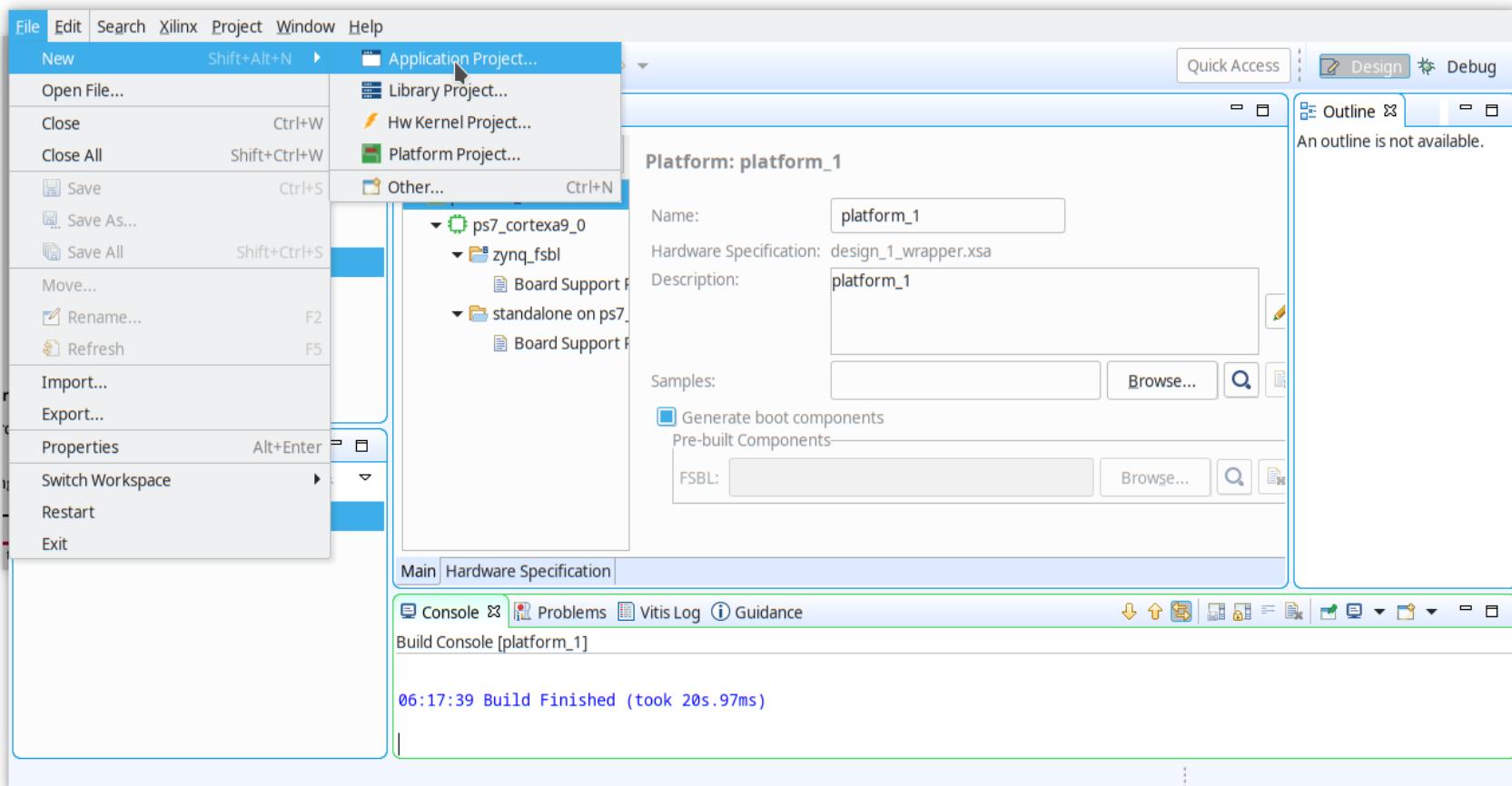
- Build platform configuration



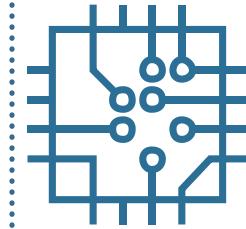
# Vitis: create application project



- Using hardware platform data (libraries)



# Vitis: create application project



- Using hardware platform data (libraries)

**Create a New Application Project**

This wizard will guide you through the 4 steps of creating new a

1. Choose a **platform** or create a **platform project** from Vivado
2. Put application project in a **system project**, associate it with
3. Prepare the application runtime – **domain**
4. Choose a template for application to quick start development

**Platform Project**

**System Project**

**XSA**

**Processor**

**Domain**

**App**

**Platform**

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

**Select a platform from repository** **Create a new platform from hardware (XSA)**

**Find:**

**Add** **Manage**

Name	Board	Flow	Vendor	Path
platform_1 [custom]	zed	Embedded SW Dev	xilinx	/export/home/toni/work/TetraMax/System.Design/semi

**Platform Info**

**General Info**

Name: platform\_1  
Part: xc7z020clq484-1  
Family: zynq  
Description: platform\_1

**Acceleration Resources**

The selected platform does not have application acceleration capabilities

**Domain Details**

**Domains**

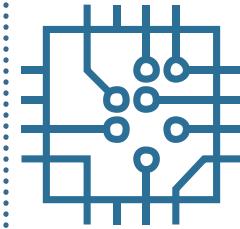
Domain name	Details
standalone on ps7_cortexa...	CPU: ps7_cortexa9_ OS: standalone

Skip welcome page

**?** **?**

< Back Next > Cancel Finish

# Vitis: create application project



- Define name, type, and domain)

## Application Project Details

Specify the application project name and its system project properties



Application project name:

## Domain

Select a domain for your project or create a new domain

## System Project

Create a new system project for the application or select an existing one from the w

Select a system project

+ Create new...

## System project details

System project name:

## Target processor

Select target processor for the.

Processor

Show all processors in the hardw

## Select a domain

standalone on ps7\_cortexa9\_0

+ Create new...

## Domain details

Name:

standalone\_domain

Display Name:

standalone on ps7\_cortexa9\_0

Operating System:

standalone

Processor:

ps7\_cortexa9\_0

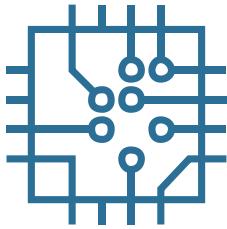


< Back

Next >

Cancel

Finish



# Vitis: create application project

- Select template to ease programming
- Application is “Hello World” demo

Templates

Select a template to create your project.

Available Templates:

Find:  [x] [...] [+]

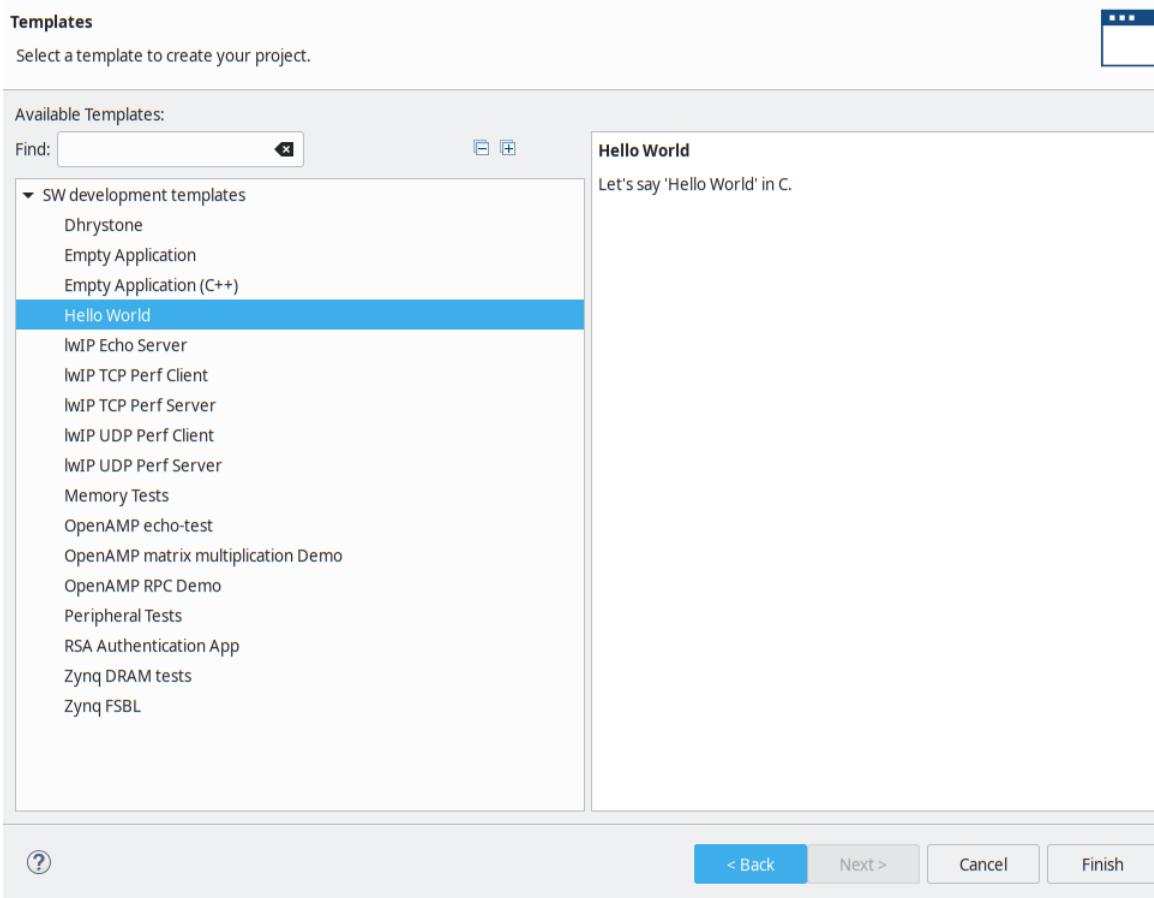
▼ SW development templates

- Dhrystone
- Empty Application
- Empty Application (C++)
- Hello World** (selected)
- IwIP Echo Server
- IwIP TCP Perf Client
- IwIP TCP Perf Server
- IwIP UDP Perf Client
- IwIP UDP Perf Server
- Memory Tests
- OpenAMP echo-test
- OpenAMP matrix multiplication Demo
- OpenAMP RPC Demo
- Peripheral Tests
- RSA Authentication App
- Zynq DRAM tests
- Zynq FSBL

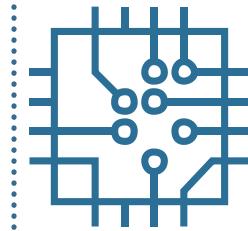
**Hello World**

Let's say 'Hello World' in C.

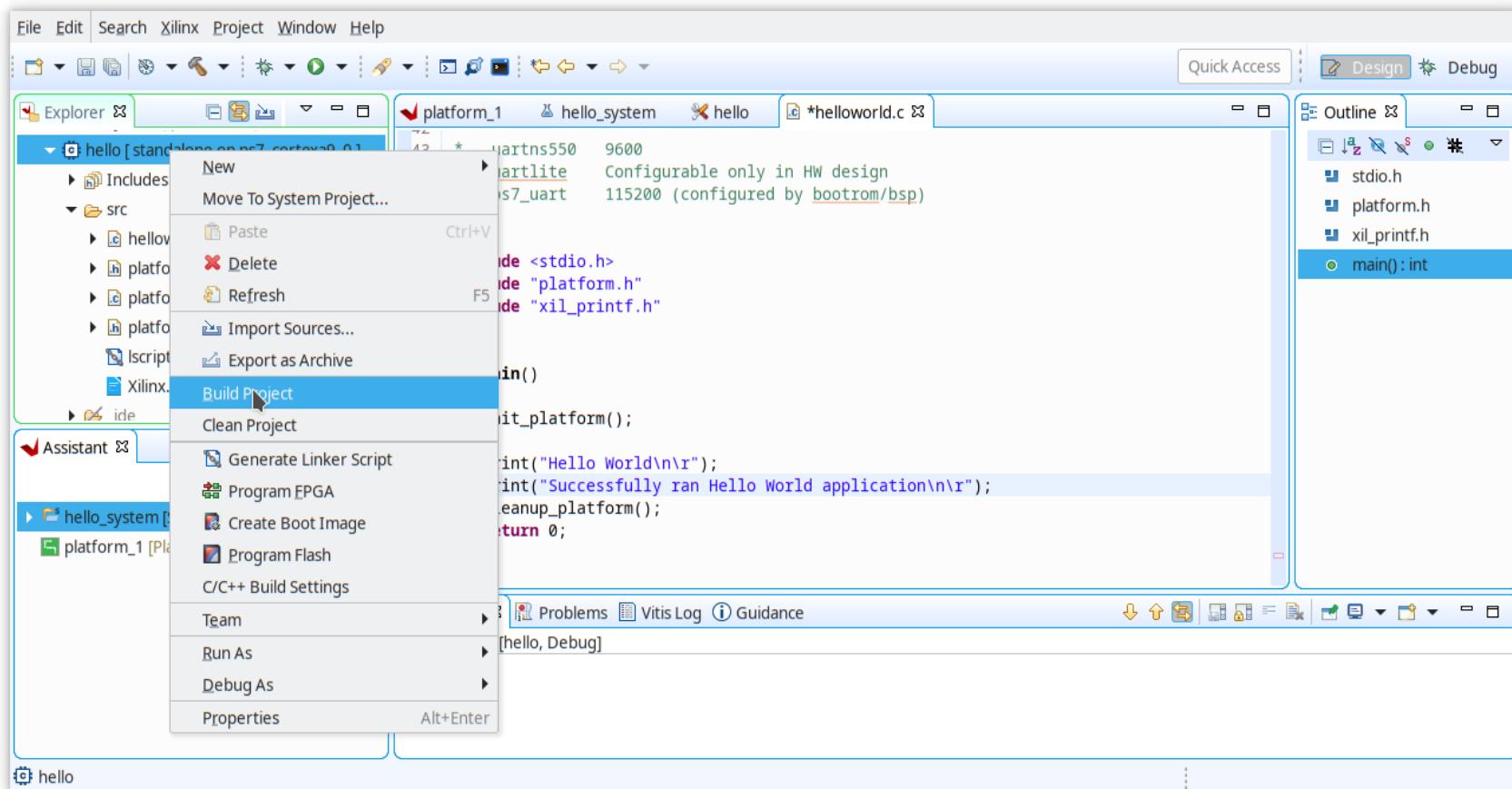
< Back Next > Cancel Finish

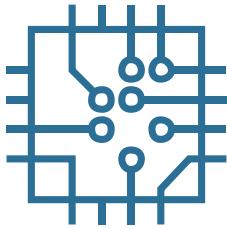


# Vitis: create application project



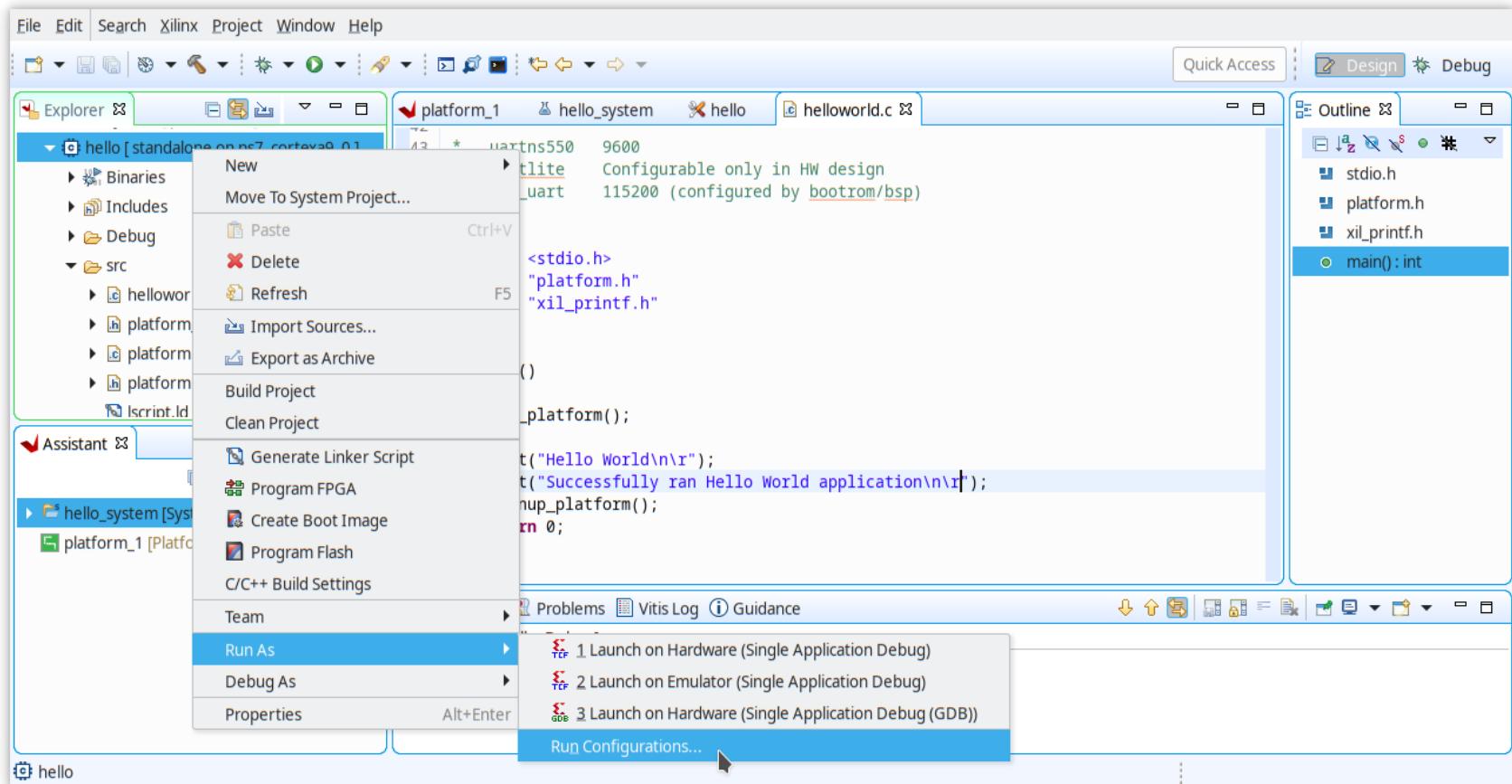
- Build “Hello World” application



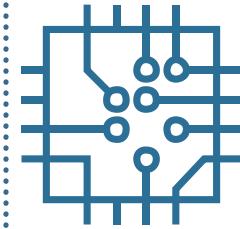


# Vitis: create application project

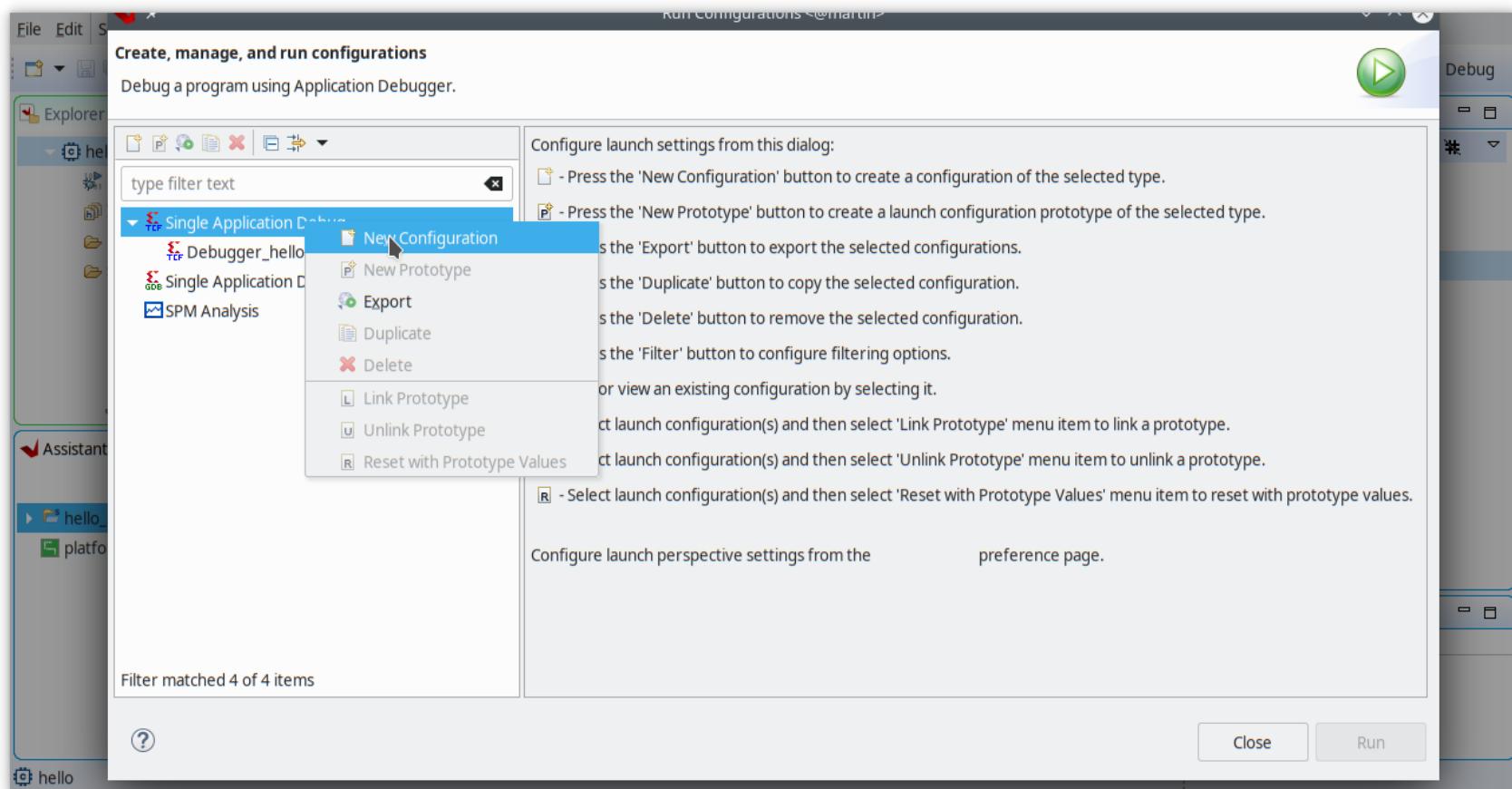
- Run “Hello World” application



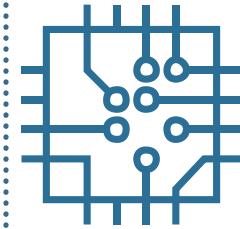
# Vitis: create application project



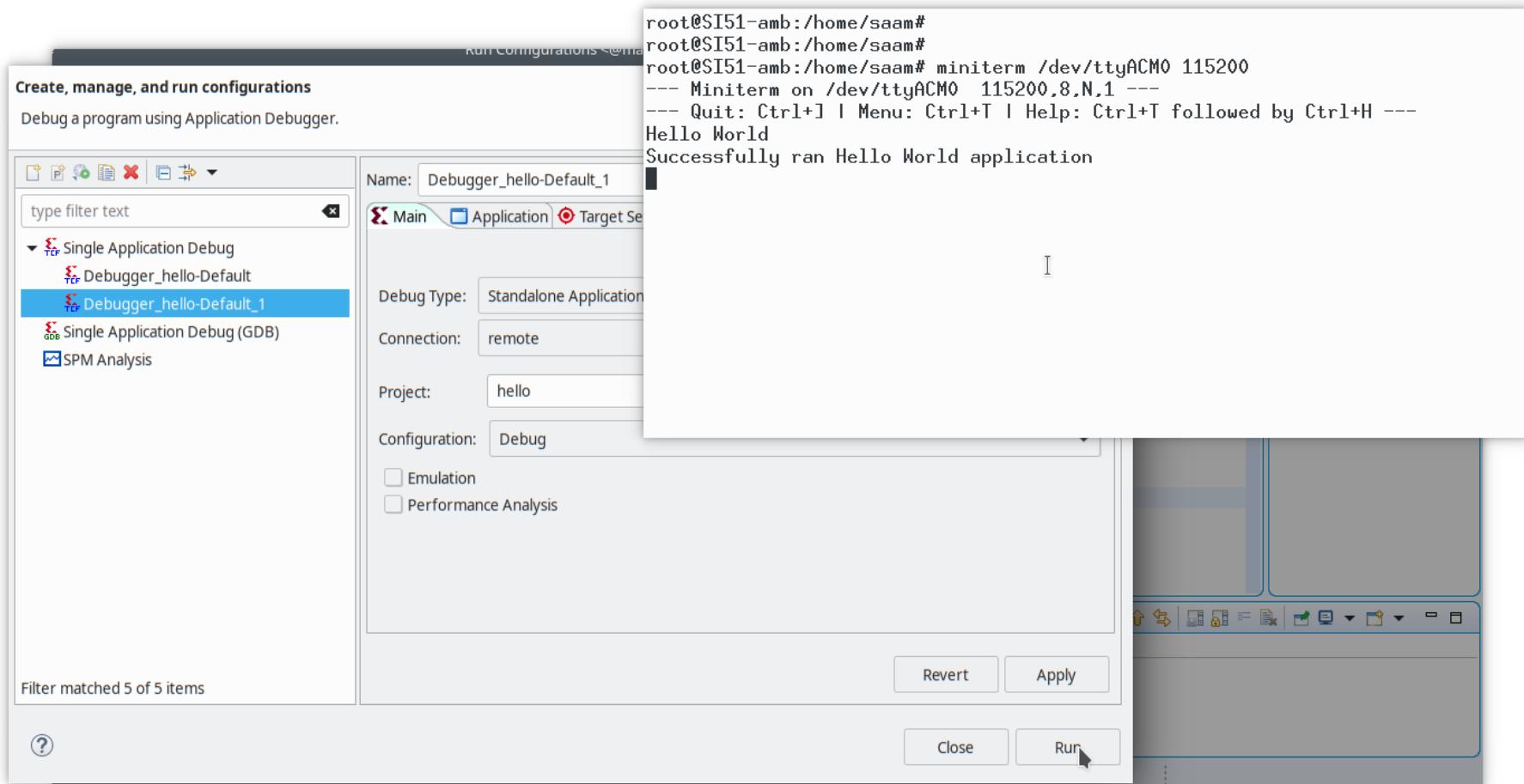
- Select run / debug environment (JTAG)



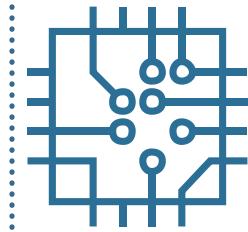
# Vitis: create application project



- Execution “Hello World” application

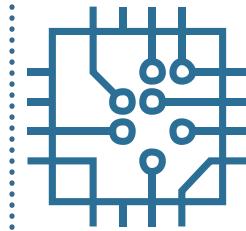


# Embedded system with PL part

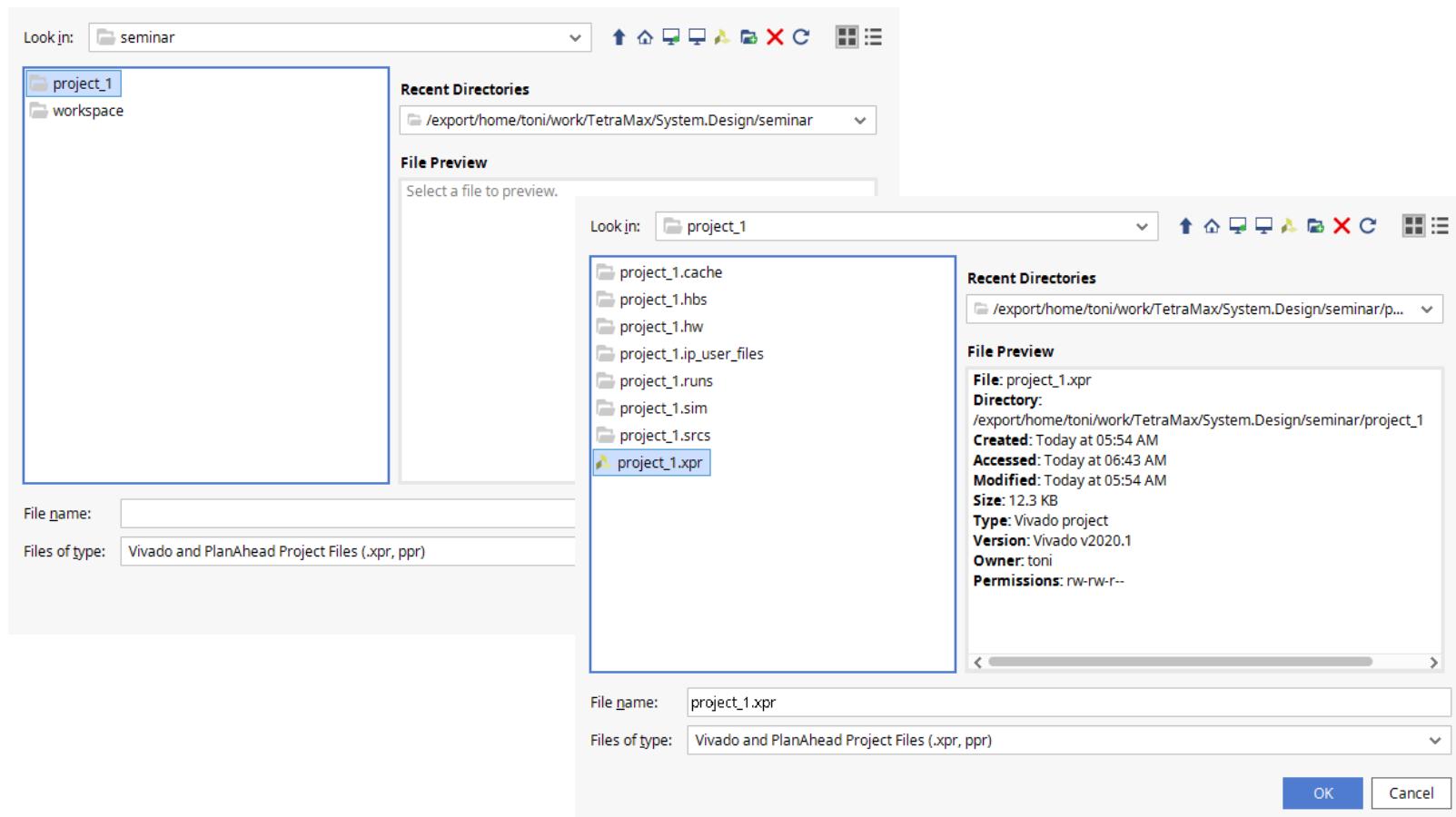


- Start from previously build system
- Use Zedboard prototyping board
- Define two GPIO interfaces in PL
  - One attached to five push buttons
  - Other attached to eight slide buttons
- Use available AXI-Lite cores

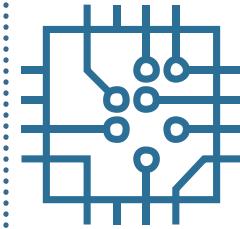
# Embedded system with PL part



- Open simple project



# Embedded system with PL part



- Save to new project

The screenshot shows the Xilinx Vivado IDE interface. The 'File' menu is open, and the 'Save As...' option is highlighted. A 'Save As...' dialog box is overlaid on the interface, prompting the user to enter a new project name ('project\_2') and specify a project location ('/export/home/toni/work/TetraMax/System.Design/seminar'). The dialog also includes checkboxes for 'Create project subdirectory' and 'Include run results'. In the background, the 'Project' panel shows the current project structure, and the 'Properties' panel is visible. The bottom status bar displays the message 'Save the current project under a new name'.

File Edit Flow Tools Reports Window Layout View Help

Project

- Add Sources... Alt+A
- Close Project
- Constraints
- Simulation Waveform
- Checkpoint
- IP
- Text Editor
- Import
- Export
- Print... Ctrl+P
- Exit

SIMULATION

- Run Simulation

RTL ANALYSIS

- Open Elaborated Design

SYNTHESIS

- Run Synthesis
- Open Synthesized Design

IMPLEMENTATION

- Run Implementation
- Open Implemented Design

New... Open... Select\_1

Open Recent

Open Example...

Save As...

Open Log File

Open Journal File

Write Tcl...

Archive...

Project Summary

Overview | Dashboard

Settings Edit

Project name: project\_1

Project location: /export/home/toni/work/TetraMax/System.Design/seminar

Product family: Zynq-7000

Project part: ZedBoard Zynq Evaluation Board

Top module name: design\_1\_wrapper

Target language: Verilog

Simulator language: Mixed

Select an object to see properties

Save this project to a new name and location.

Tcl Console Messages Log Reports Design Runs

Name Constraints Status

synth\_1 (active) constrs\_1 synth\_design Complete!

impl\_1 constrs\_1 write\_bitstream Complete!

Out-of-Context Module Runs

design\_1 Submodule Runs Complete

Project name: project\_2

Project location: /export/home/toni/work/TetraMax/System.Design/seminar

Create project subdirectory

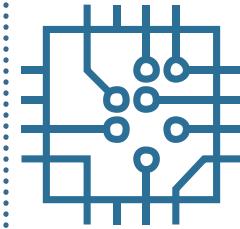
Project will be created at: .../System.Design/seminar/project\_2

Include run results

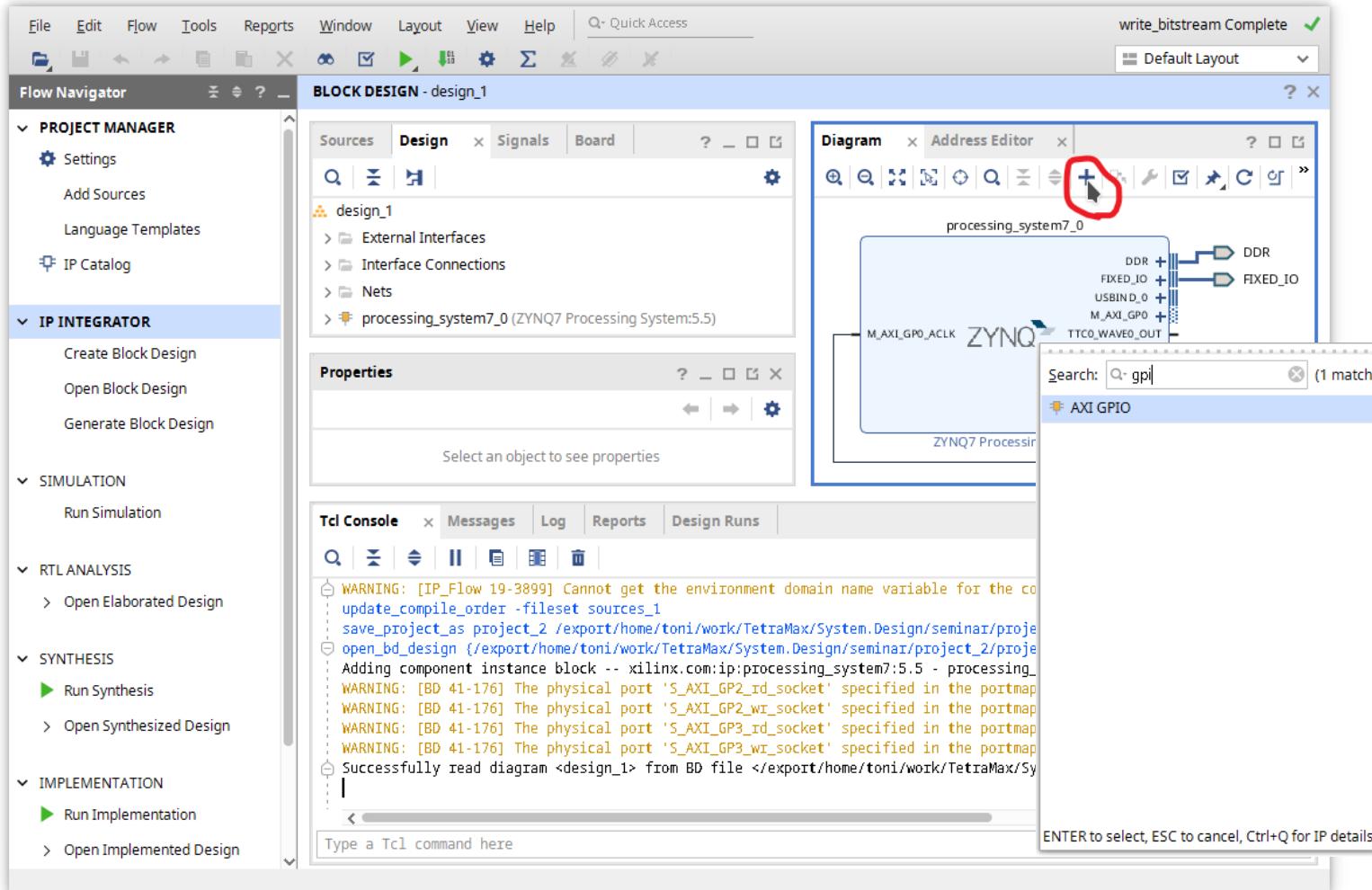
OK Cancel

Save the current project under a new name

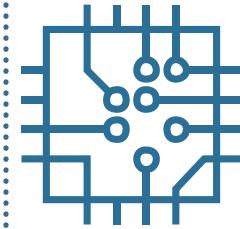
# Embedded system with PL part



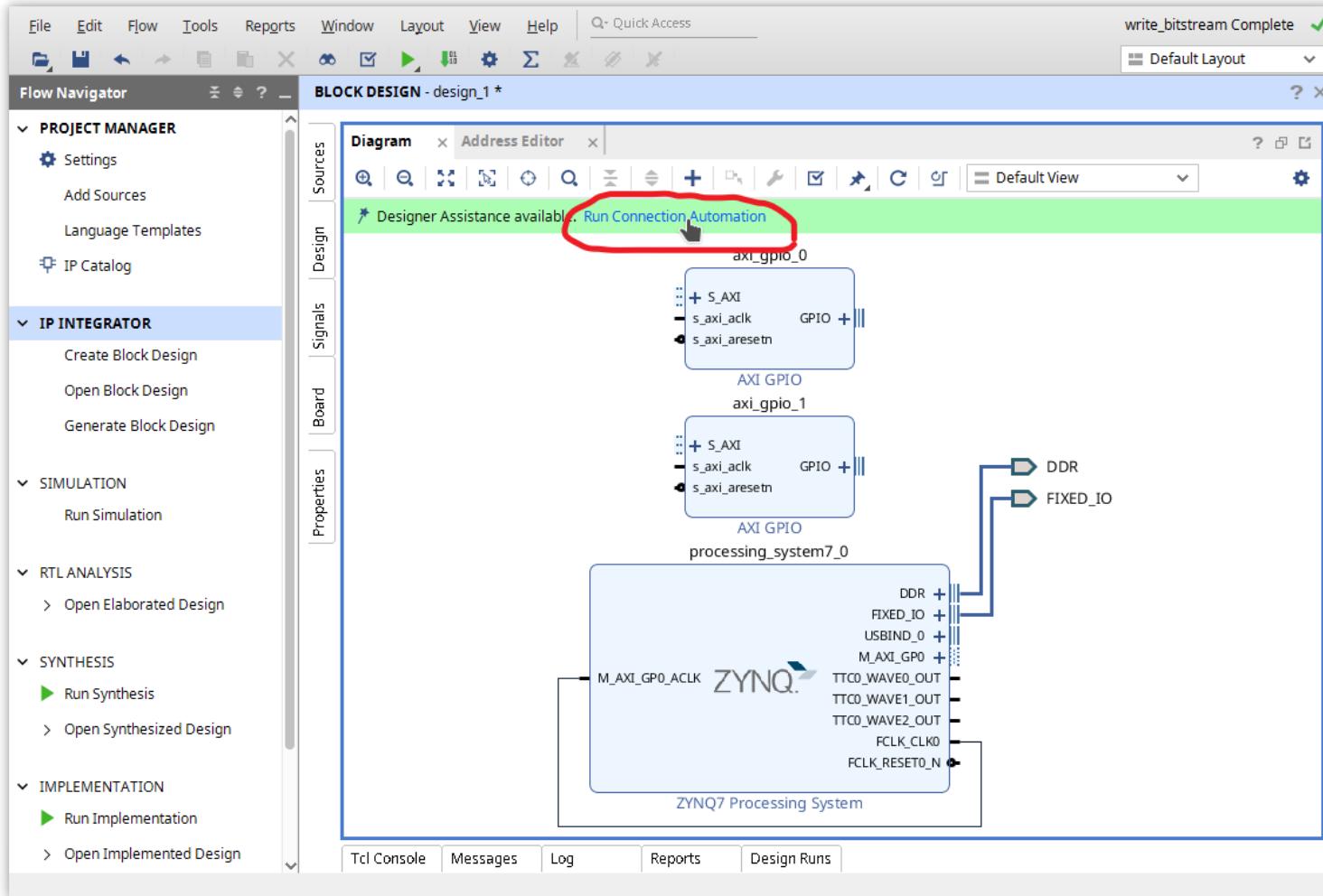
- Open Block Design and add new IP cores



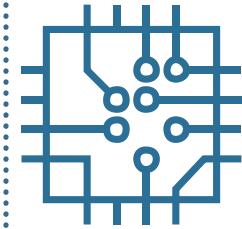
# Embedded system with PL part



- Connecting new cores



# Embedded system with PL part

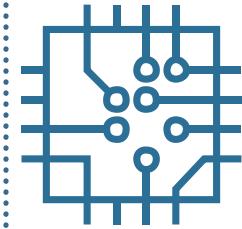


- Connecting new cores to the AXI bus

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.



# Embedded system with PL part

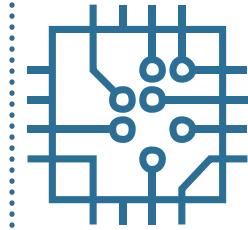


- Connecting new cores to the AXI bus

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.



# Embedded system with PL part



- Connecting new cores to the I/O pins

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.



Q | X | ▲

✓ All Automation (2 out of 2 selected)

✓ axi\_gpio\_0

✓ GPIO

✓ axi\_gpio\_1

✓ GPIO

Description

Connect Board Part Interface to IP interface.

Interface: /axi\_gpio\_0/GPIO

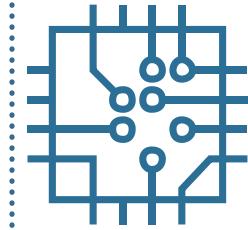
Options

Select Board Part Interface sws\_8bits ( DIP switches ) ▾

?

OK Cancel

# Embedded system with PL part



- Connecting new cores to the I/O pins

Automatically make connections in your design by checking the boxes of the interfaces to connect. Select an interface on the left to display its configuration options on the right.



Q | X | ▲

✓ All Automation (2 out of 2 selected)

✓ axi\_gpio\_0

✓ GPIO

✓ axi\_gpio\_1

✓ GPIO

Description

Connect Board Part Interface to IP interface.

Interface: /axi\_gpio\_1/GPIO

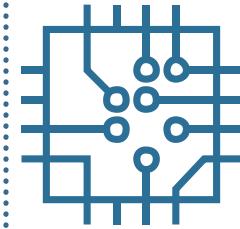
Options

Select Board Part Interface: btns\_5bits ( Push buttons )

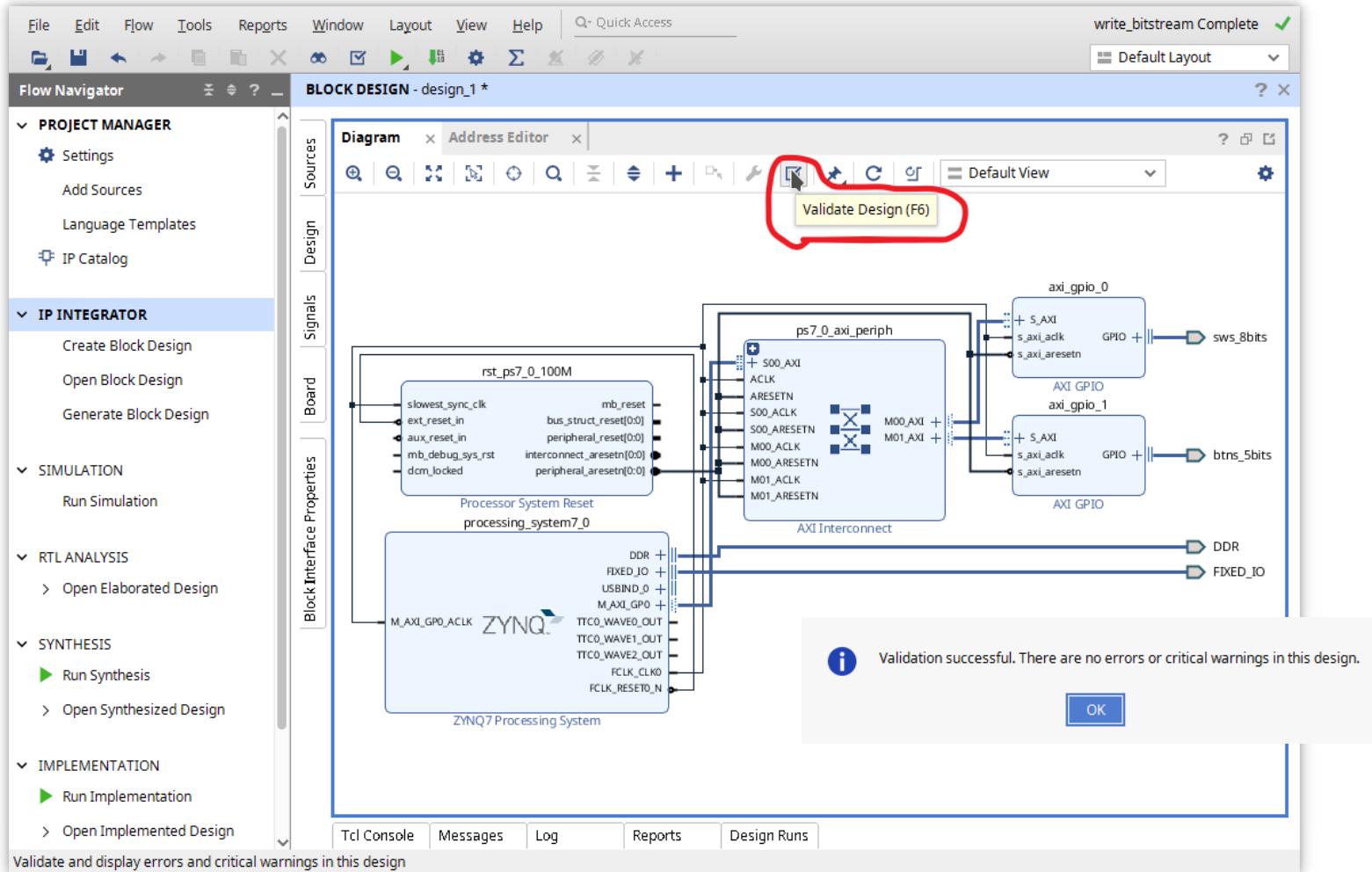
?

OK Cancel

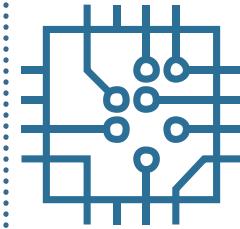
# Embedded system with PL part



- Validate hardware design



# Embedded system with PL part

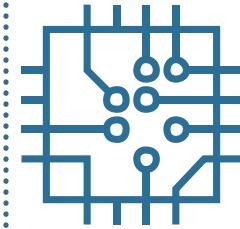


- Verify memory addresses of GPIO

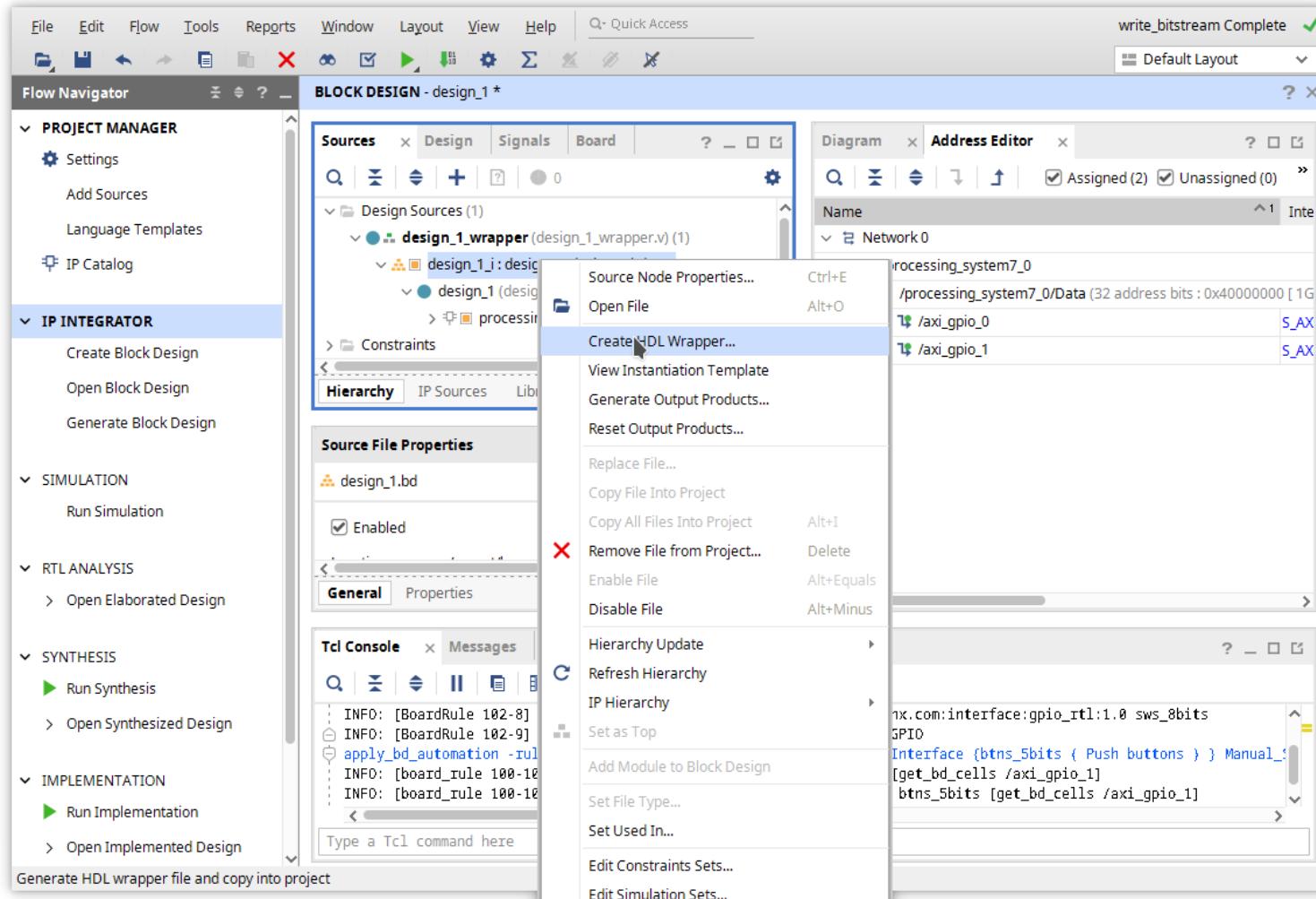
The screenshot shows the Xilinx Vivado Block Design Editor interface. The left sidebar contains a tree view of project components under 'IP INTEGRATOR'. The main area is titled 'BLOCK DESIGN - design\_1 \*' and shows the 'Address Editor' tab selected. The table lists assigned memory addresses:

Name	Type	Reg	Master Base Address	Range	Slave Segment	Interface
/processing_system7_0/Data /axi_gpio_0	S_AXI	Reg	0x4120_0000	64K	0x4120	
/processing_system7_0/Data /axi_gpio_1	S_AXI	Reg	0x4121_0000	64K	0x4121	

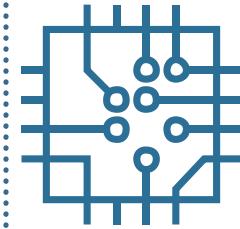
# Embedded system with PL part



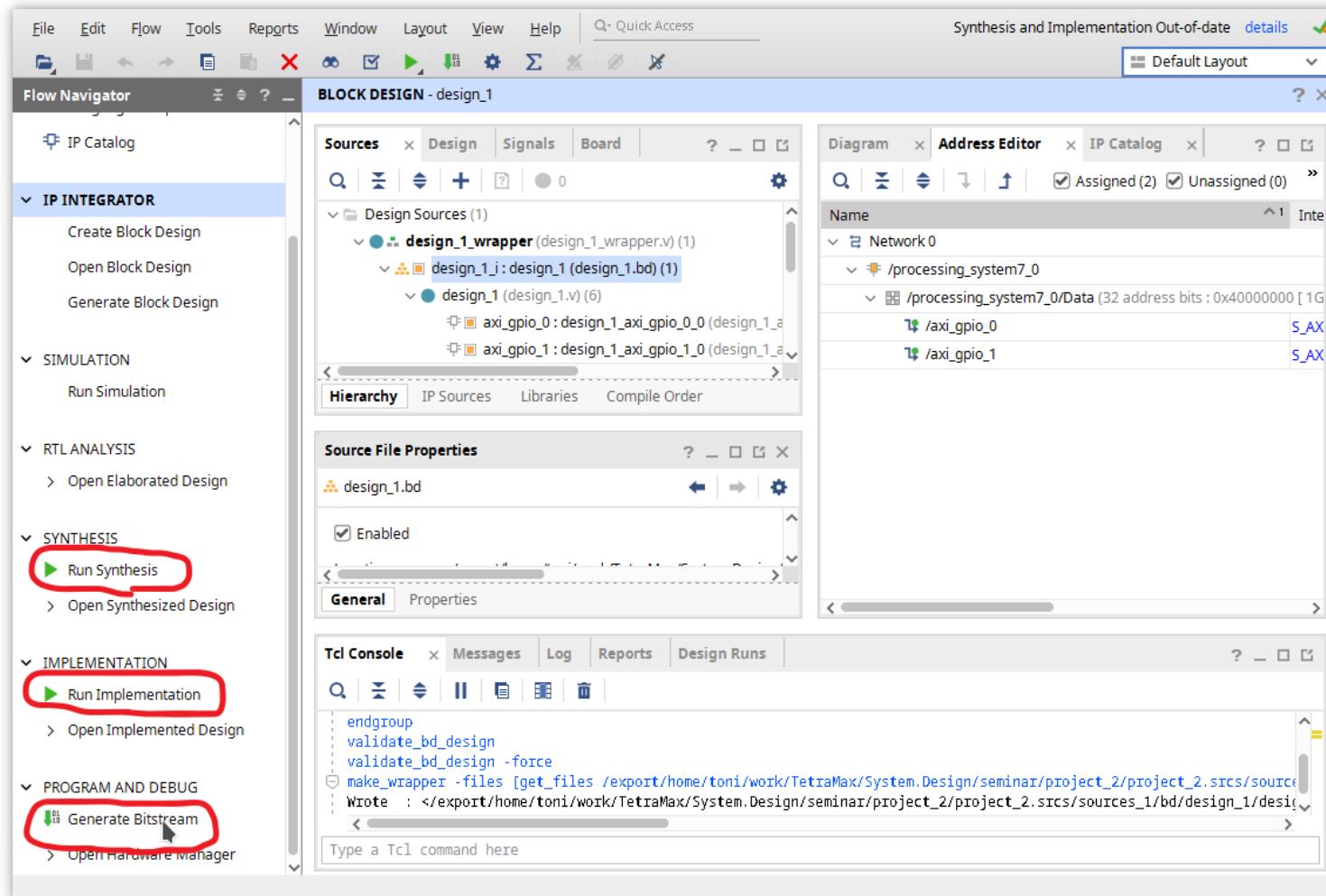
- Create new HDL wrapper



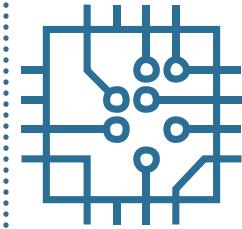
# Embedded system with PL part



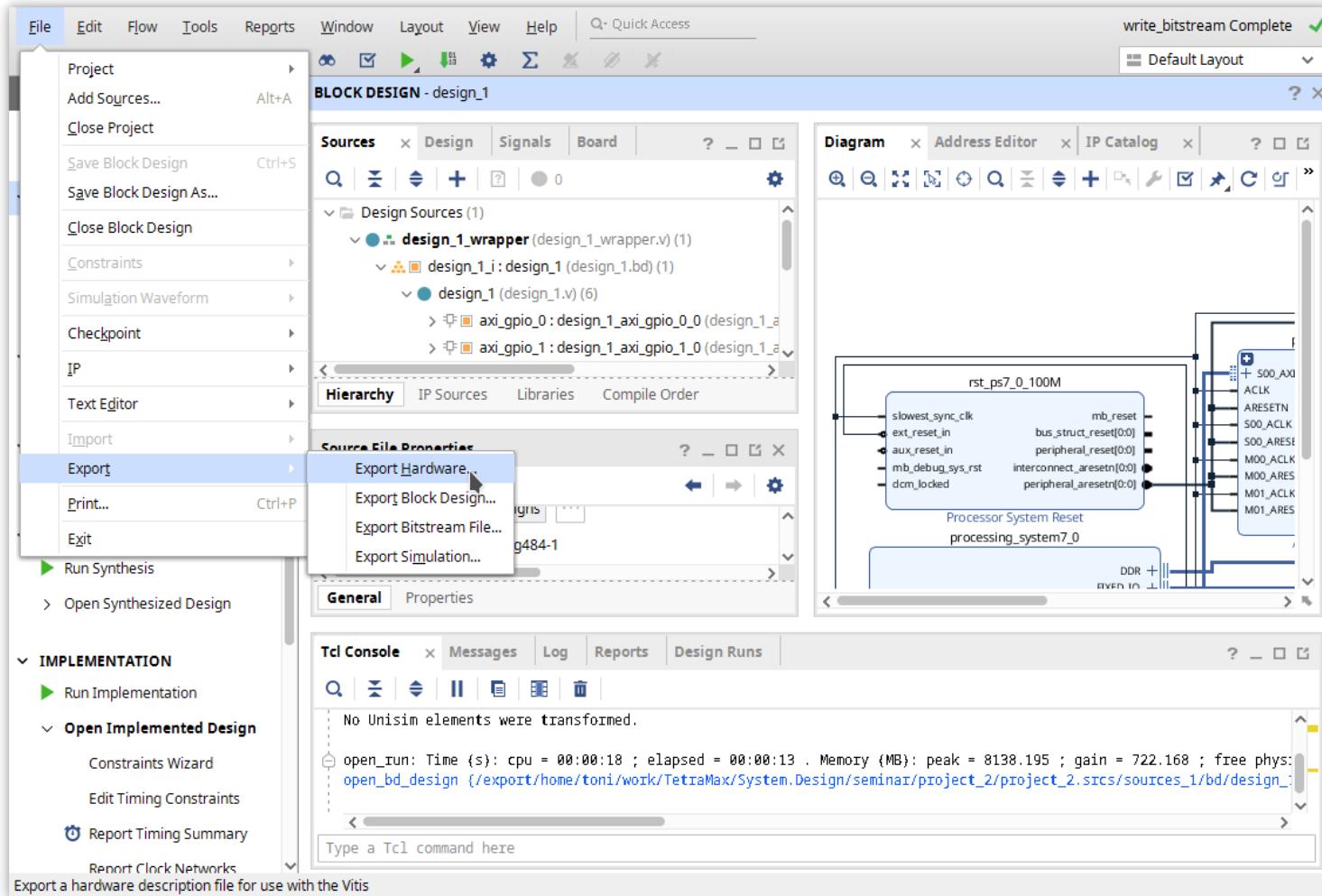
- Synthesis, implementation, generation



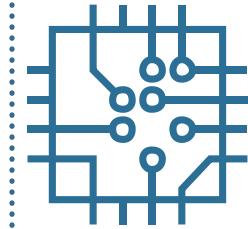
# Embedded system with PL part



- Export hardware description



# Embedded system with PL part



- Export hardware description



**Export Hardware Platform**

This wizard will guide you through the export of a hardware platform for use in the Vitis or PetaLinux software tools.

To export a hardware platform, you will need to specify the platform properties.

**Platform type**

Fixed  
A platform supporting embedded software.

Expandable  
A platform supporting acceleration.

**Output**

Set the platform properties to inform downstream tools of the intended use of the target platform's hardware design.

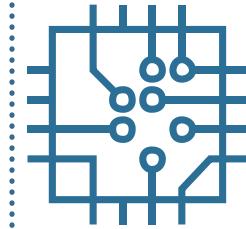
Pre-synthesis  
This platform includes a hardware specification for downstream software tools.

Include bitstream  
This platform includes the complete hardware implementation and bitstream, in addition to the hardware specification for downstream software tools.

< Back

< Back Next > Finish Cancel

# Vitis: create platform project



- New platform description must be built

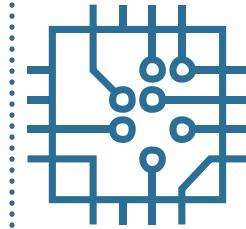
The screenshot shows the Vitis IDE interface. The left sidebar has a 'File' menu open, with 'Platform Project...' highlighted. The main workspace shows a C file named 'helloworld.c' with the following code:

```
ps7_uart 115200 (configured by bootrom/bsp)

45 * ps7_uart 115200 (configured by bootrom/bsp)
46 */
47
48 #include <stdio.h>
49 #include "platform.h"
50 #include "xil_printf.h"
51
52
53 int main()
54 {
55     init_platform();
56
57     print("Hello World\n\r");
58     print("Successfully ran Hello World application\n\r");
59     cleanup_platform();
60     return 0;
61 }
```

The right side of the interface includes an 'Outline' view showing files like stdio.h, platform.h, xil\_printf.h, and main(). At the bottom, there are tabs for 'Console', 'Problems', 'Vitis Log', and 'Guidance', along with a 'Build Console' area.

# Vitis: create platform project



- Importing hardware definition file

**Create new platform project**

Enter a name for your platform project

This wizard will guide you through creation of a platform project. This platform will enable you to specify options for the kernels, BSPs, supported for embedded software developers.

Platform project name: platform\_2

A new platform project can be created from one of the two inputs:

**From hardware specification (XSA)**  
Create a new platform project from a hardware specification customized later from the platform project editor.

**From existing platform**  
Load the platform definition from an existing platform. You project.

**Platform**

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

Create a new platform from hardware (XSA)  Select a platform from repository

XSA File: /export/home/toni/work/TetraMax/System.Design/seminar/project\_2/design\_1\_wrapper.xsa

**Hardware Specification**

Specify the details for the initial domain to be added to the platform. More domains can be after the platform is created by double clicking the platform.spr file

Operating system: standalone

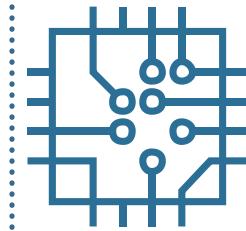
Processor: ps7\_cortexa9\_0

*Note: A domain with selected operating system and processor will be added to the platform. The platform project can be modified later to add new domains or change settings.*

Generate boot components

< Back Next > Cancel Finish

# Vitis: create platform project



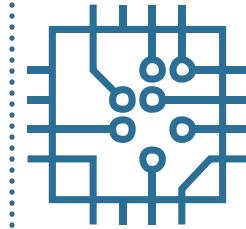
- Build platform configuration

The screenshot shows the Vitis IDE interface with the following details:

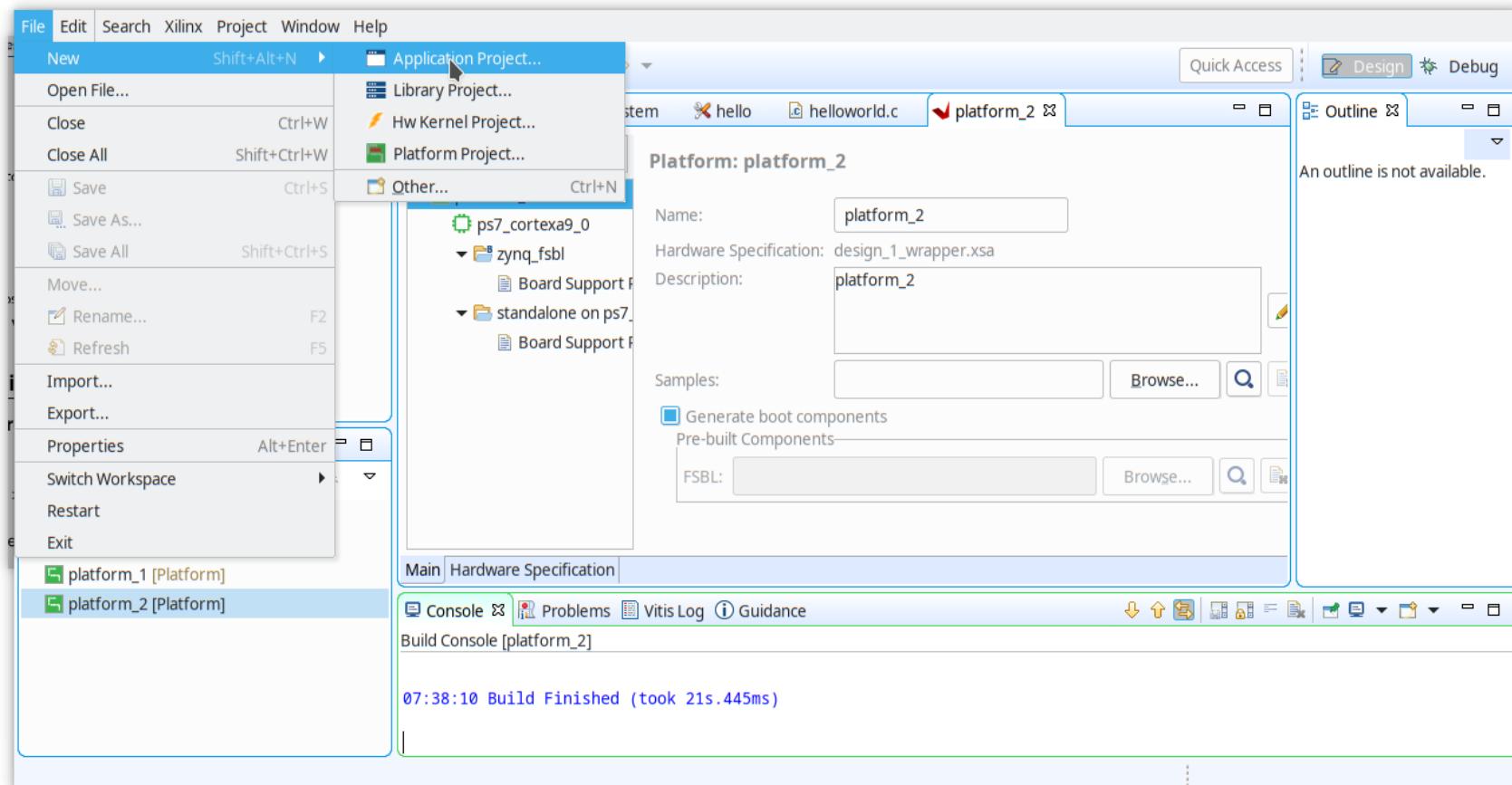
- File Bar:** File, Edit, Search, Xilinx, Project, Window, Help.
- Toolbar:** Includes icons for New, Open, Save, Import, Export, Run, Stop, and others.
- Project Explorer:** Shows two projects: "platform\_1" and "platform\_2 (Out-of-date)". Under "platform\_2", there are sub-folders: "bitstream", "export", "hw", "logs", "ps7\_cortexa9\_0", "resources", "zyng\_fsbl", and "platform.spr".
- Platform Configuration:** The "platform\_2" tab is active. The "Main" tab shows the configuration:
  - Name: platform\_2
  - Hardware Specification: design\_1\_wrapper.xsa
  - Description: platform\_2
  - Samples: (empty)
  - Generate boot components: (unchecked)
  - Pre-built Components:
    - FSBL: (empty)
- Console:** Shows Platform Tcl Console output:

```
platform create -name {platform_2} -hw {/export/home/toni/work/TetraMax/System.Design/seminar/project_2/design_1_wrapper.xsa}
platform read {/export/home/toni/work/TetraMax/System.Design/seminar/workspace/platform_2/platform.spr}
platform active {platform_2}
```
- Outline:** Shows the message "An outline is not available."

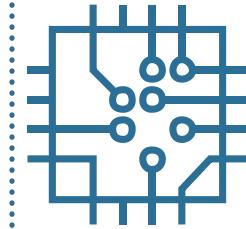
# Vitis: create application project



- Using hardware platform data (libraries)



# Vitis: create application project



- Using hardware platform data (libraries)

Create a New Application Project

Platform

Choose a platform for your project. You can also create an application from XSA through the 'Create a new platform from hardware (XSA)' tab.

This wizard will guide you through the 4 steps of creating r

1. Choose a **platform** or create a **platform project** from
2. Put application project in a **system project**, associate it
3. Prepare the application runtime - **domain**
4. Choose a template for application to quick start develop

**Platform Project**

**Processor** —————— **Domain** —————— **XSA**

**Sy P**

**Select a platform from repository** **Create a new platform from hardware (XSA)**

**Find:**  X

**Add** **Manage**

Name	Board	Flow	Vendor	Path
platform_1 [custom]	zed	Embedded SW Dev	xilinx	/export/home/toni/work/TetraMax/System.Design/semi
platform_2 [custom]	zed	Embedded SW Dev	xilinx	/export/home/toni/work/TetraMax/System.Design/semi

**Platform Info**

**General Info**

Name:   
Part:   
Family:   
Description:

**Acceleration Resources**

The selected platform does not have application acceleration capabilities

**Domain Details**

**Domains**

Domain name	Details
standalone on ps7_corte...	CPU: ps7_cortexa9_ OS: standalone

Skip welcome p

?

?

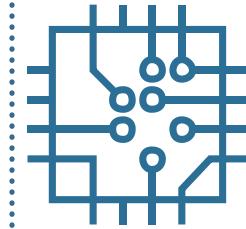
< Back

Next >

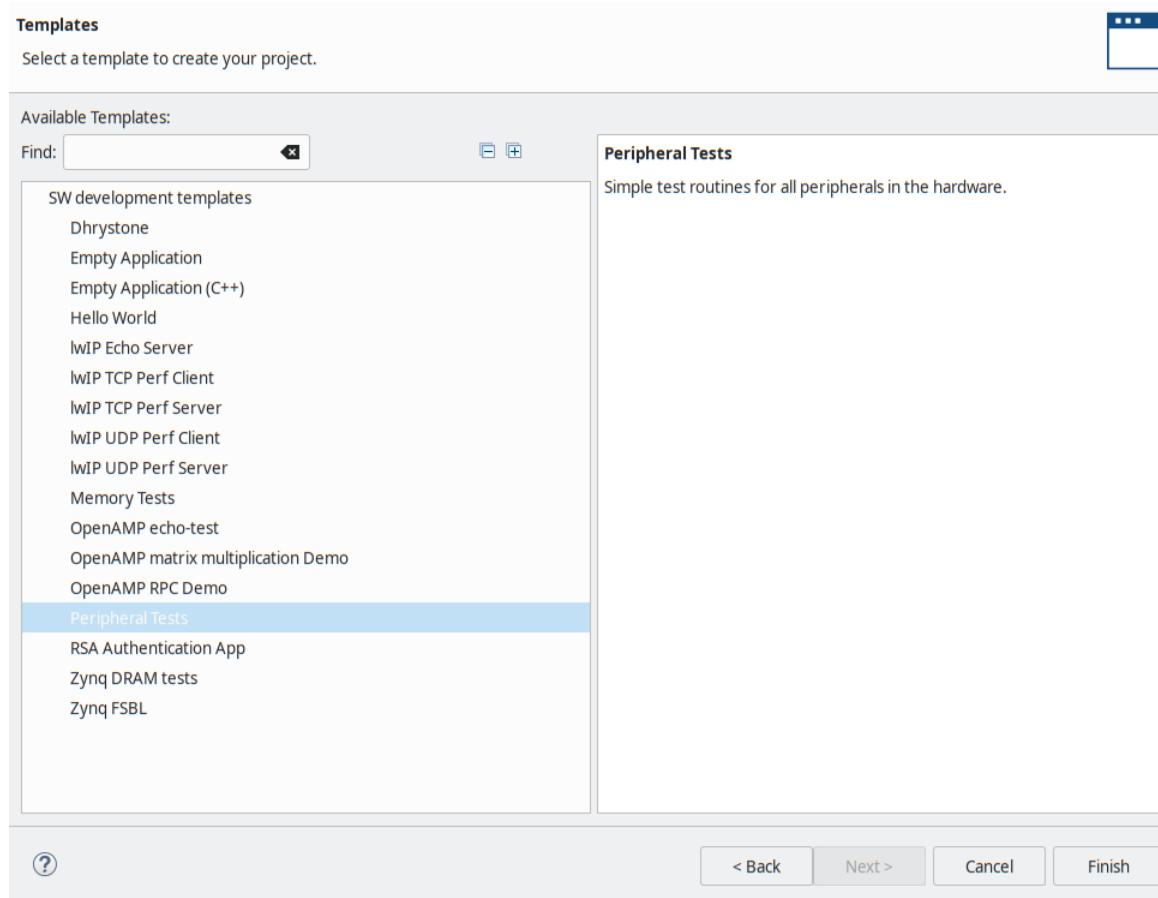
Cancel

Finish

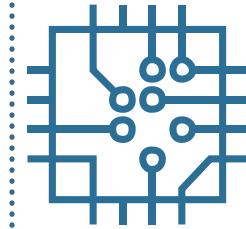
# Vitis: create application project



- Select template to ease programming
- Application read GPIO data end write to UART



# Vitis: create application project



- Build GPIO application

The screenshot shows the Vitis IDE interface with the following components:

- File Bar:** File, Edit, Search, Xilinx, Project, Window, Help.
- Toolbar:** Standard file operations (New, Open, Save, Print, Find, Replace, Copy, Paste, Cut, Undo, Redo), Build (Build, Clean, Rebuild, Run), and Design/Debug tools.
- Project Explorer:** Shows the project structure under "Exp".
  - primer\_system [platform\_2]**:
    - primer [standalone on ps7\_cortexa9\_0]**: Includes devcfg\_header.h, dmaps\_header.h, gpio\_header.h, qspips\_header.h, scugic\_header.h, scutimer\_header.h, scuwdt\_header.h, testperiph.c, ttcps\_header.h.
  - hello\_system [System]**: hello [Application], platform\_1 [Platform], platform\_2 [Platform].
  - primer\_system [System]**.
- Code Editor:** Displays the content of **testperiph.c**. The code performs self-tests for the SCuGic module and sets up its interrupt system.

```
int Status;

print("\r\n Running ScuGicSelfTestExample() for ps7_scugic_0...\r\n");

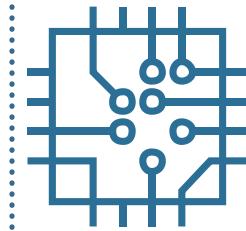
Status = ScuGicSelfTestExample(XPAR_PS7_SCUGIC_0_DEVICE_ID);

if (Status == 0) {
    print("ScuGicSelfTestExample PASSED\r\n");
}
else {
    print("ScuGicSelfTestExample FAILED\r\n");
}

{
    int Status;

    Status = ScuGicInterruptSetup(&intc, XPAR_PS7_SCUGIC_0_DEVICE_ID);
    if (Status == 0) {
        print("ScuGic Interrupt Setup PASSED\r\n");
    }
}
```
- Outline View:** Lists all header files included in the project: stdio.h, xparameters.h, xil\_cache.h, xscugic.h, xil\_exception.h, scugic\_header.h, xgpio.h, gpio\_header.h, xdevcfg.h, devcfg\_header.h, xdmmaps.h, dmaps\_header.h, xqspips.h, qspips\_header.h.
- Console:** Shows the build console output for "primer, Debug".
- Status Bar:** Writable, Smart Insert, 1 : 1.

# Vitis: create application project

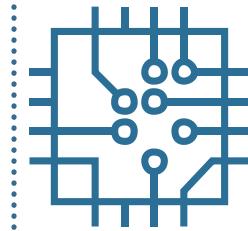


- Run GPIO application

The screenshot shows the Vitis IDE interface with the following details:

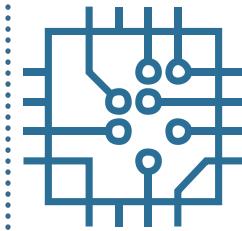
- File Menu:** File, Edit, Search, Xilinx, Project, Window, Help.
- Toolbars:** Standard toolbar with icons for file operations, search, and project management.
- Explorer View:** Shows the project structure with nodes like zynq\_fsbl, platform.spr, primer\_system [platform\_2], and primer [standalone].
- Code Editor:** Displays C++ code for a GPIO application, specifically a testperiph.c file. The code includes imports for stdio.h, xparameters.h, and various Xilinx header files. It contains logic for running a self-test example and setting up an interrupt.
- Outline View:** Shows a list of included header files: stdio.h, xparameters.h, xil\_cache.h, xscugic.h, xil\_exception.h, scugic\_header.h, xgpio.h, gpio\_header.h, xdevcfg.h, devcfg\_header.h, xdmmaps.h, dmaps\_header.h, xqspips.h, qspips\_header.h.
- Assistant View:** Shows options for generating a Linker Script, Program FPGA, Create Boot Image, and Program Flash.
- Run Configuration Menu:** A context menu for the project node "primer\_system [System]" is open, showing options like Run As, Debug As, and Properties. The "Run As" option is highlighted, and a submenu is displayed with three choices:
  - 1 Launch on Hardware (Single Application Debug)
  - 2 Launch on Emulator (Single Application Debug)
  - 3 Launch on Hardware (Single Application Debug (GDB))

# Vitis: create application project



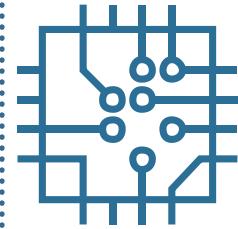
- Execution “Hello World” application

# Outline



- Part 1
  - FPGA structure and design platform
  - VHDL hardware design in FPGA
  - Embedded system design on FPGA
- Part 2
  - Embedded software design FPGA
  - **IP core development and integration**
- Part 3
  - Software and hardware debugging

# Manage IP



- Start with Manage IP
- Select New IP

The screenshot shows the Vivado HLx Editions software interface. The top menu bar includes File, Flow, Tools, Window, Help, and a Quick Access search bar. The XILINX logo is in the top right corner.

The main area has three sections: "Quick Start" (with Create Project, Open Project, and Open Example Project options), "Tasks" (with Manage IP, New IP Location Manager, and Open IP Location...), and "Learning Center" (with Documentation and Tutorials and Quick Take Videos).

In the "Tasks" section, the "New IP Location Manager" option is highlighted with a mouse cursor. Below the tasks, a message reads: "Start using Vivado with the IP Catalog as a starting point to more quickly customize and create IP."

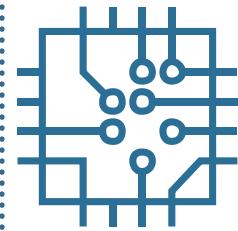
The right side of the interface features a "Recent Projects" list and a "Recent IP Locations" list. The "Recent Projects" list contains:

- ILA /export/home/toni/work/TetraMax/2021/test/ILA
- ILA /export/home/toni/work/TetraMax/2021/dbg/ILA
- check\_ILA /export/home/toni/work/TetraMax/2021/sandbox/check\_ILA
- UART\_LED\_IP /export/home/toni/work/TetraMax/2021/sandbox/UART\_LED\_IP
- Hello\_World /export/home/toni/work/TetraMax/2021/sandbox>Hello\_World
- UART\_LED /export/home/toni/work/TetraMax/2021/sandbox/UART\_LED
- UART\_LED\_ILA /export/home/toni/work/TetraMax/2021/sandbox/UART\_LED\_ILA
- counter /export/home/toni/work/TetraMax/2021/sandbox/counter
- uart\_1 /export/home/toni/work/TetraMax/2021/sandbox/uart\_1

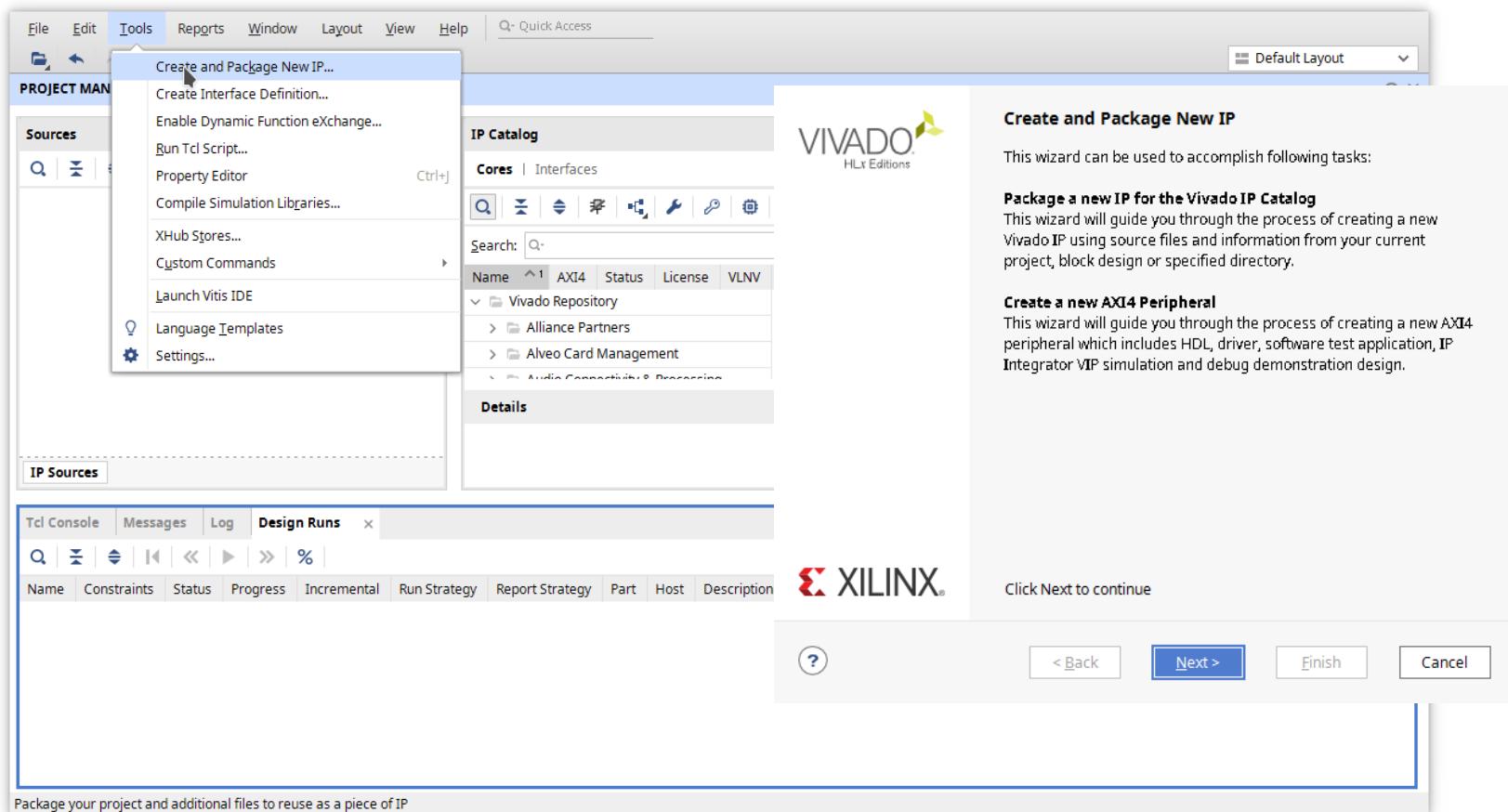
The "Recent IP Locations" list contains:

- led\_ip

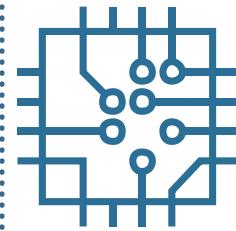
# IP Core Project



- Add custom core in VHDL



# IP Core Project



- Add custom core in VHDL

The screenshot shows the Xilinx IP Catalog interface. The top menu bar has 'Tools' selected, with a dropdown menu open showing 'Create and Package New IP...'. The main window is titled 'IP Catalog' with tabs for 'Cores' and 'Interfaces'. Below the tabs are search and filter tools. A central panel displays a tree structure of IP cores, with 'VLAN' selected. To the right, a 'Peripheral Details' panel is open, prompting the user to specify name, version, display name, description, and IP location for a new peripheral. The 'Name' field is set to 'led\_ip', 'Version' to '1.0', 'Display name' to 'led\_ip\_v1.0', 'Description' to 'My LED IP', and 'IP location' to '/export/home/toni/work/TetraMax/2021/sandbox/MY\_IP/p\_repo'. A checkbox for 'Overwrite existing' is present. At the bottom, there are navigation buttons: '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

Create Peripheral, Package IP or Package a Block Design  
Please select one of the following tasks.

**Peripheral Details**  
Specify name, version and description for the new peripheral

**Packaging Options**

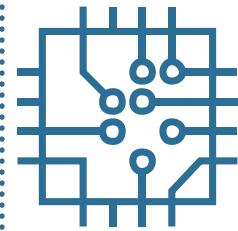
- Package your current project  
Use the project as the source for creating a new IP Definition.
- Package a block design from the current project  
Choose a block design as the source for creating a new IP Definition.
- Package a specified directory  
Choose a directory as the source for creating a new IP Definition.

**Create AXI4 Peripheral**

- Create a new AXI4 peripheral  
Create an AXI4 IP, driver, software test application, IP Integrator AXI4 VIP simulation and debug demonstration design.

? < Back Next > Finish Cancel

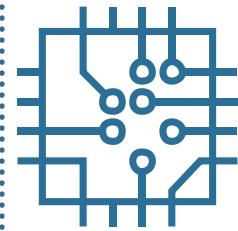
# IP Core Project



- Add custom core in VHDL

The screenshot shows the Xilinx Vivado IP Catalog interface. The top menu bar has 'Tools' selected, with a dropdown menu open showing 'Create and Package New IP...'. The main window displays the 'IP Catalog' with tabs for 'Cores' and 'Interfaces'. On the left, there's a 'PROJECT MANAGER' pane with 'Sources' and a search bar. Below it is a section titled 'Add Interfaces' with the sub-instruction 'Add AXI4 interfaces supported by your peripheral'. A tree view shows an 'led\_ip\_v1.0' project containing an 'S\_AXI' interface. To the right, there's a 'VIVADO HLx Editions' logo and a 'Create Peripheral' panel. This panel includes fields for 'Name' (set to 'S\_AXI'), 'Interface Type' (set to 'Lite'), 'Interface Mode' (set to 'Slave'), 'Data Width (Bits)' (set to '32'), 'Memory Size (Bytes)' (set to '64'), and 'Number of Registers' (set to '4 [4..512]'). The 'Create Peripheral' panel also lists three demonstration designs: 1. IP (user.org:user:led\_ip:1.0) with 1 interface(s), 2. AXI4 VIP Simulation demonstration design (with a 'more info' link), and 3. AXI4 Debug Hardware Simulation demonstration design (with a 'more info' link). It also indicates that the peripheral will be available in the catalog at /export/home/toni/work/TetraMax/2021/sandbox/MY\_IP/ip\_r. The bottom of the interface features 'Next >', 'Finish', and 'Cancel' buttons.

# IP Core Project



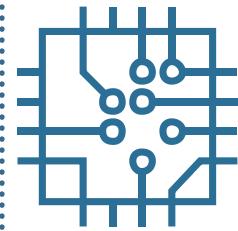
- Update HDL part of IP core template

hdl → MY\_IP/ip\_repo/led\_ip\_1.0/hdl

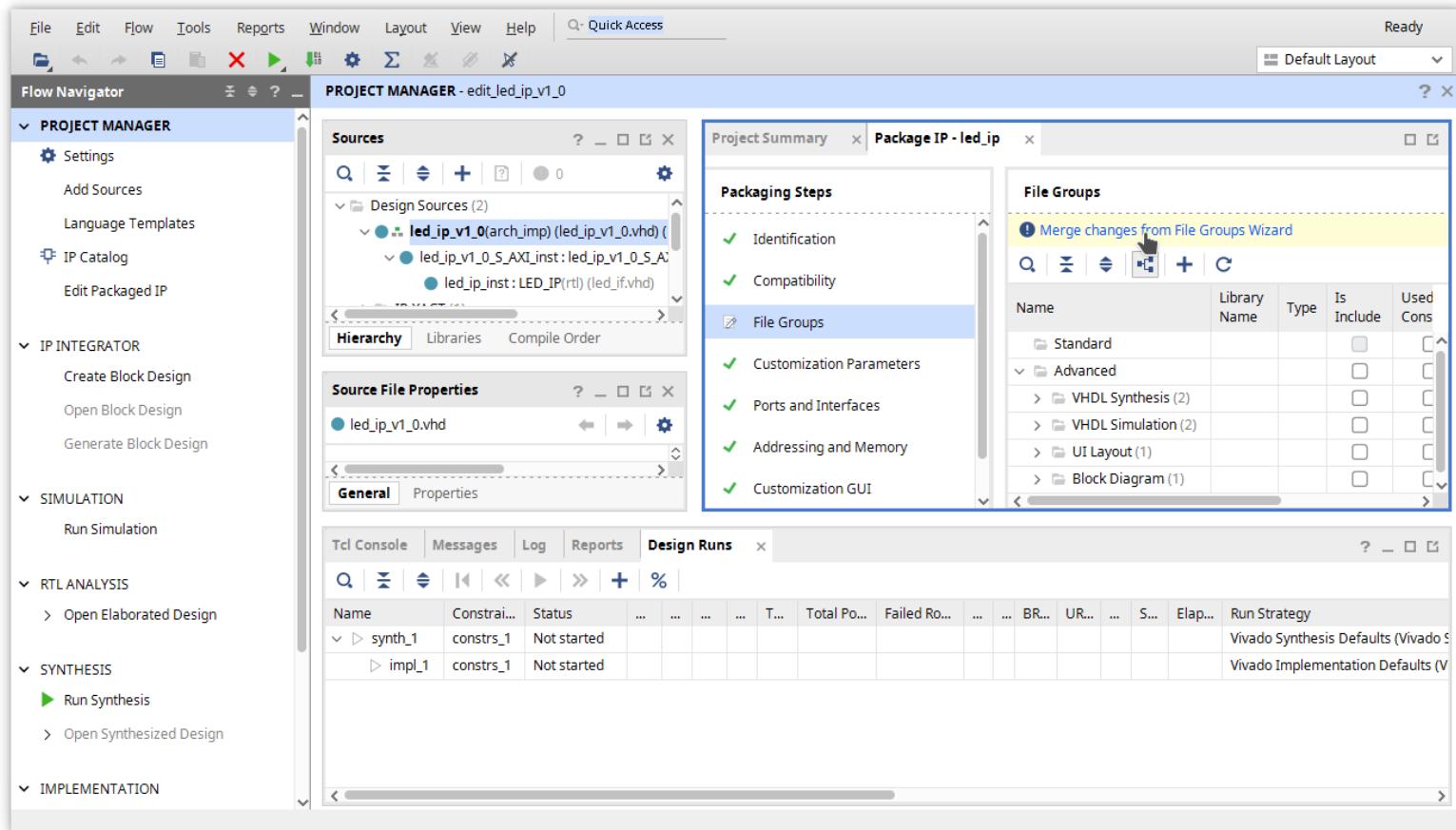
The screenshot shows the Xilinx Vivado IDE interface with the following details:

- Project Manager:** The main window displays the project structure under "PROJECT MANAGER".
- Sources:** The "Sources" tab lists three design sources:
  - "led\_ip\_v1\_0(arch\_imp)"
  - "led\_ip\_v1\_0.vhd" (selected)
  - "led\_ip\_v1\_0\_S\_AXI.lns"
- Context Menu:** A context menu is open over the "led\_ip\_v1\_0.vhd" file, with the "Hierarchy Update" option highlighted.
- Identification:** The "Identification" panel contains the following information:
  - Vendor: user.org
  - Library: user
  - Name: led\_ip
  - Version: 1.0
  - Display name: led\_ip\_v1.0
  - Description: My LED IP
  - Vendor display name:
  - Company url:
- Project Summary:** The "Project Summary" tab provides an overview of the project's build status.
- Package IP - led\_ip:** The "Package IP - led\_ip" tab shows the package configuration.
- Bottom Panel:** The "Tcl Console", "Messages", and "Log" panels are visible at the bottom.

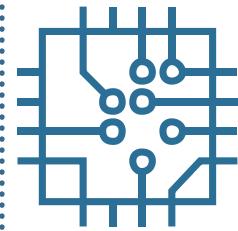
# IP Core Project



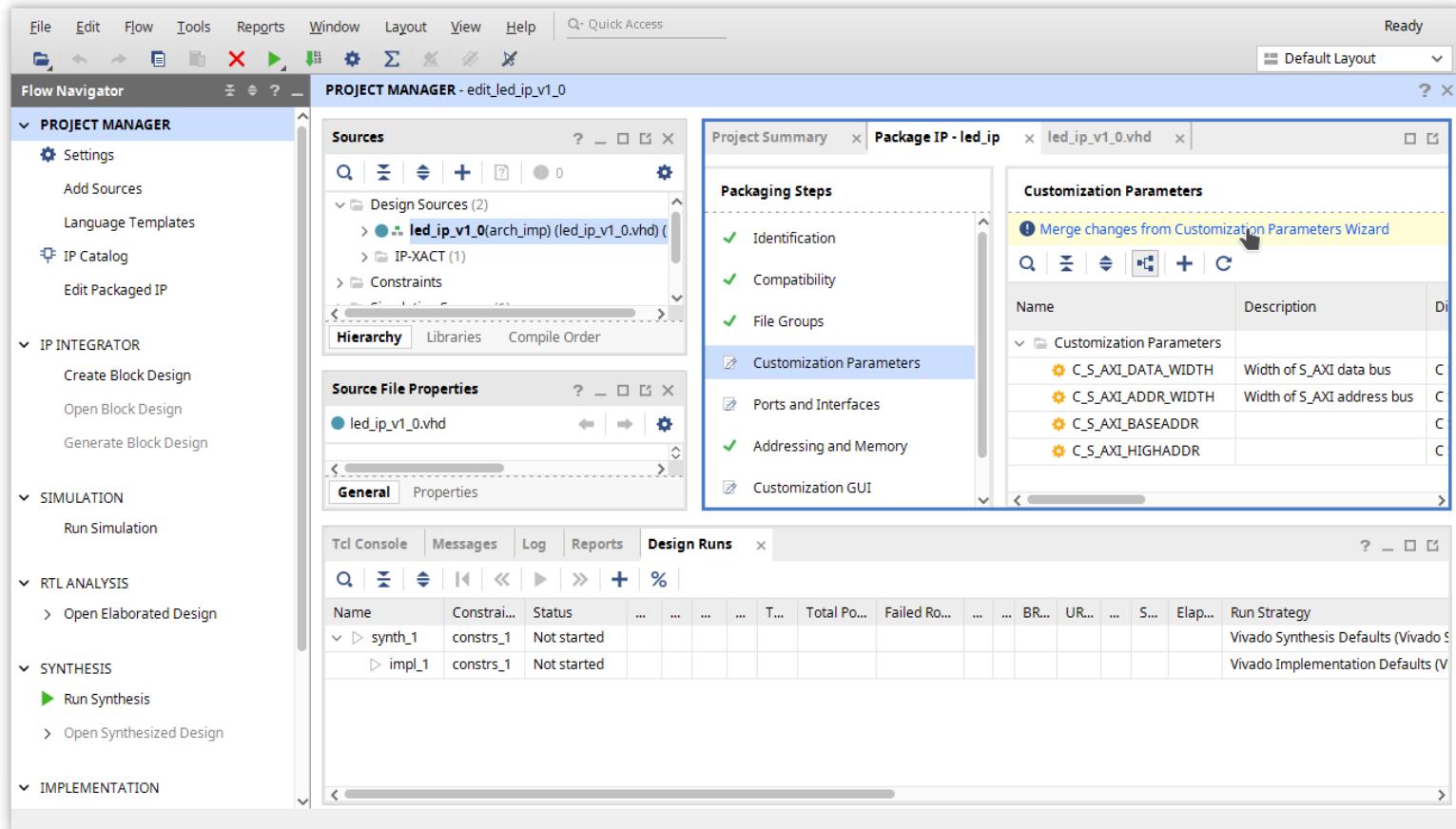
- Update IP core properties



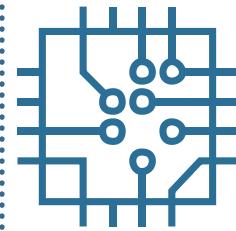
# IP Core Project



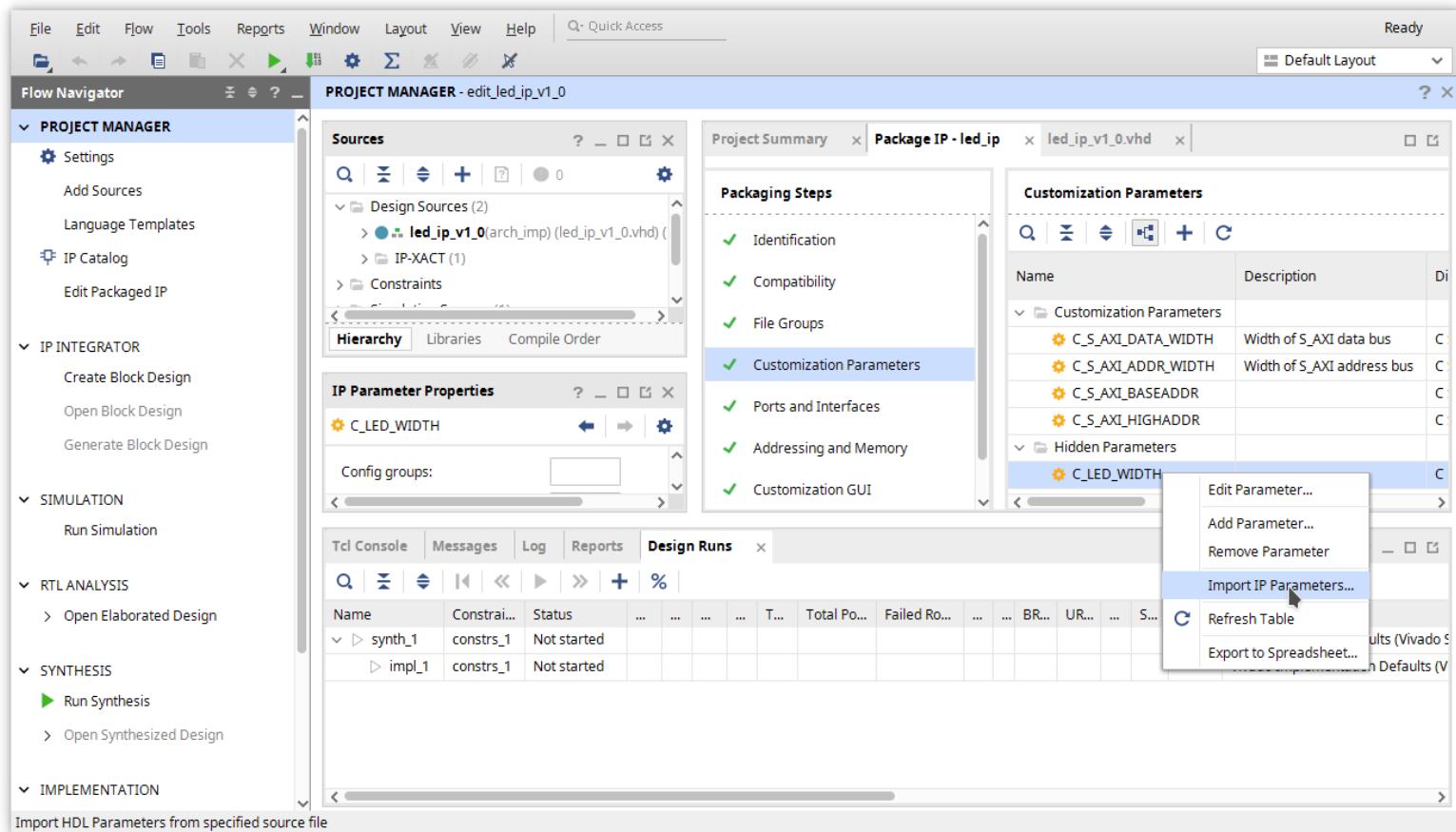
- Update IP core properties



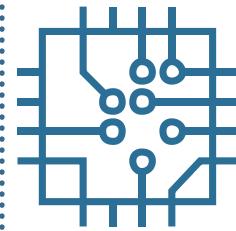
# IP Core Project



- Update IP core properties



# IP Core Project



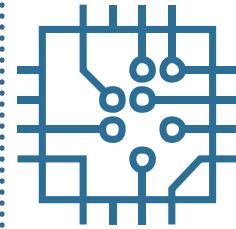
- Update IP core properties

The screenshot shows the Vivado Project Manager interface for a project named "edit\_led\_ip\_v1\_0". The left sidebar contains navigation links for Flow Navigator, Project Manager, IP Integrator, Simulation, RTL Analysis, Synthesis, and Implementation. The main area displays the "PROJECT MANAGER - edit\_led\_ip\_v1\_0" window. In the center, there is a "Sources" panel listing "Design Sources (2)" and "Constraints". Below it is a "Properties" panel with a message: "Select an object to see properties". To the right, the "Project Summary" tab is active, showing "Packaging Steps" (Compatibility, File Groups, Customization Parameters, Ports and Interfaces, Addressing and Memory, Customization GUI) and a "Customization Parameters" table. The "Customization Parameters" table lists the following entries:

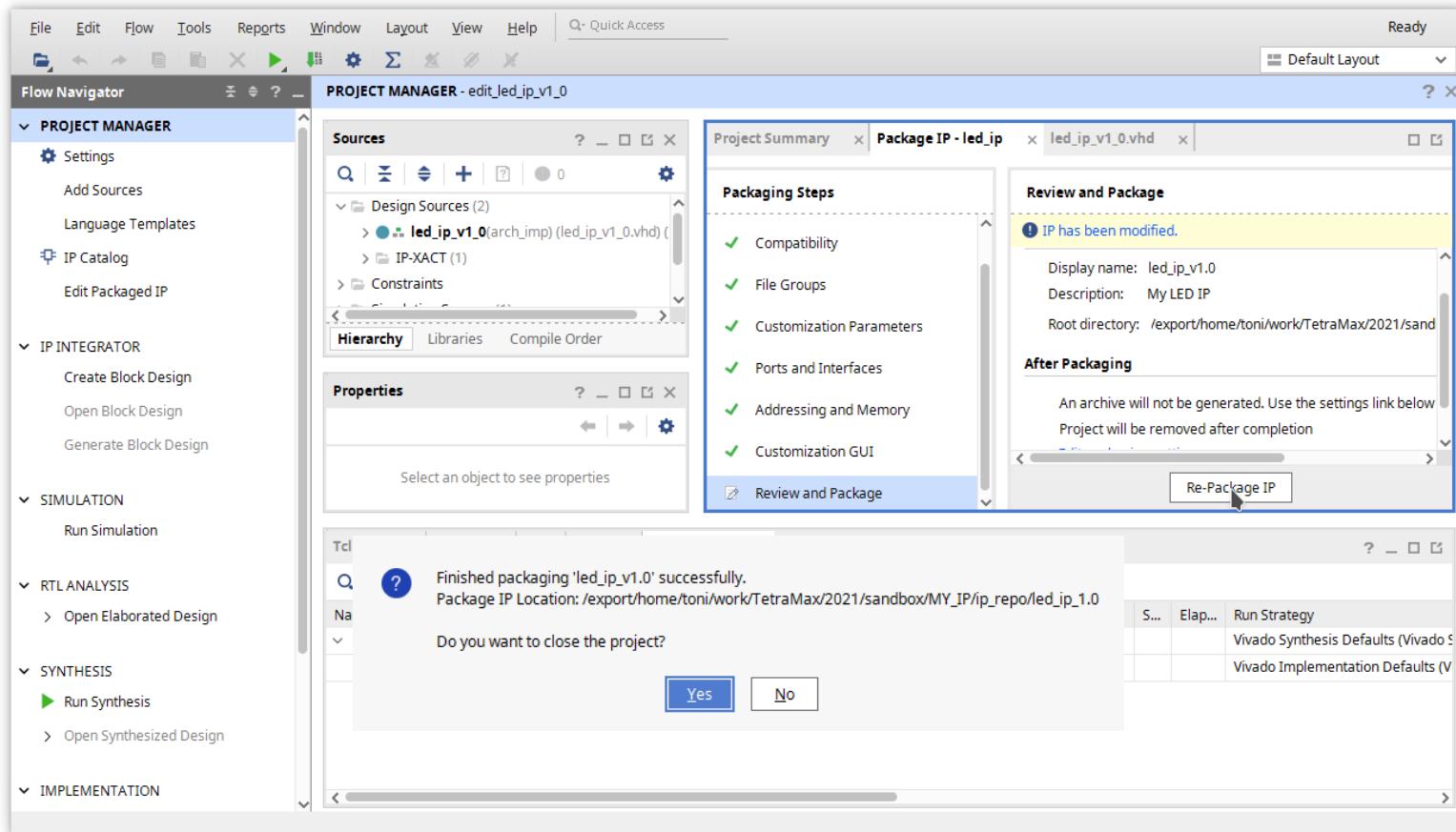
Name	Description	Display Name
C_S_AXI_BASEADDR		C S AXI BASEADDR
C_S_AXI_HIGHADDR		C S AXI HIGHADD
C_LED_WIDTH		C Led Width
C_S_AXI_DATA_WIDTH		C S Axi Data Width
C_S_AXI_ADDR_WIDTH		C S Axi Addr Width

At the bottom, the "Design Runs" tab is open, showing a table with two rows: "synth\_1" and "impl\_1", both in the "Not started" status.

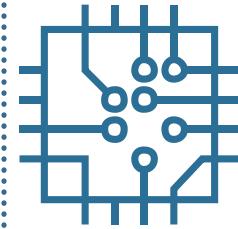
# IP Core Project



- Re-package and exit



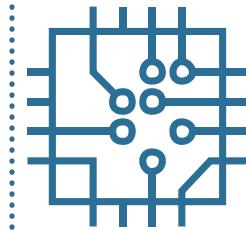
# IP Core Project



- Create new project

The screenshot shows the Vivado HLx Editions interface. At the top, there's a menu bar with File, Flow, Tools, Window, Help, and a Quick Access search bar. Below the menu is a toolbar with icons for Create Project, Open Project, and Open Example Project. The main area has three sections: 'Quick Start' (with Create Project, Open Project, and Open Example Project options), 'Tasks' (with Manage IP, Open Hardware Manager, and XHub Stores), and 'Learning Center' (with Documentation and Tutorials and Quick Take Videos). A 'Tcl Console' tab is at the bottom left. On the right, a 'Create a New Vivado Project' wizard window is open. It has a title 'Create a New Vivado Project' and a sub-instruction 'This wizard will guide you through the creation of a new project.' It explains the steps: providing a name and location for project files, specifying the type of flow, and choosing project sources and default part. The Xilinx logo is at the bottom of the wizard window. Navigation buttons at the bottom of the wizard include '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

# IP Core Project



- Create new project

**Project Name**  
Enter a name for your project and specify a directory where the project data files will be stored.

Project name:

Project location:

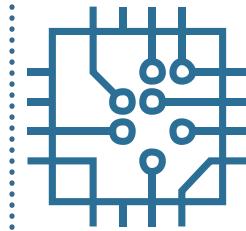
Create project subdirectory

Project will be created at: .../2021/sandbox/UART\_LED\_IP

**Project Type**  
Specify the type of project to create.

- RTL Project**  
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.  
 Do not specify sources at this time
- Post-synthesis Project**  
You will be able to add sources, view device resources, run design analysis, planning and implementation.  
 Do not specify sources at this time
- I/O Planning Project**  
Do not specify design sources. You will be able to view part/package resources.
- Imported Project**  
Create a Vivado project from a Synplify, XST or ISE Project File.
- Example Project**  
Create a new Vivado project from a predefined template.

# IP Core Project

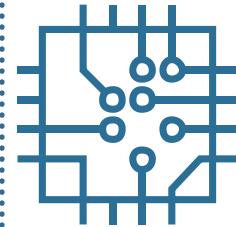


- Create new project

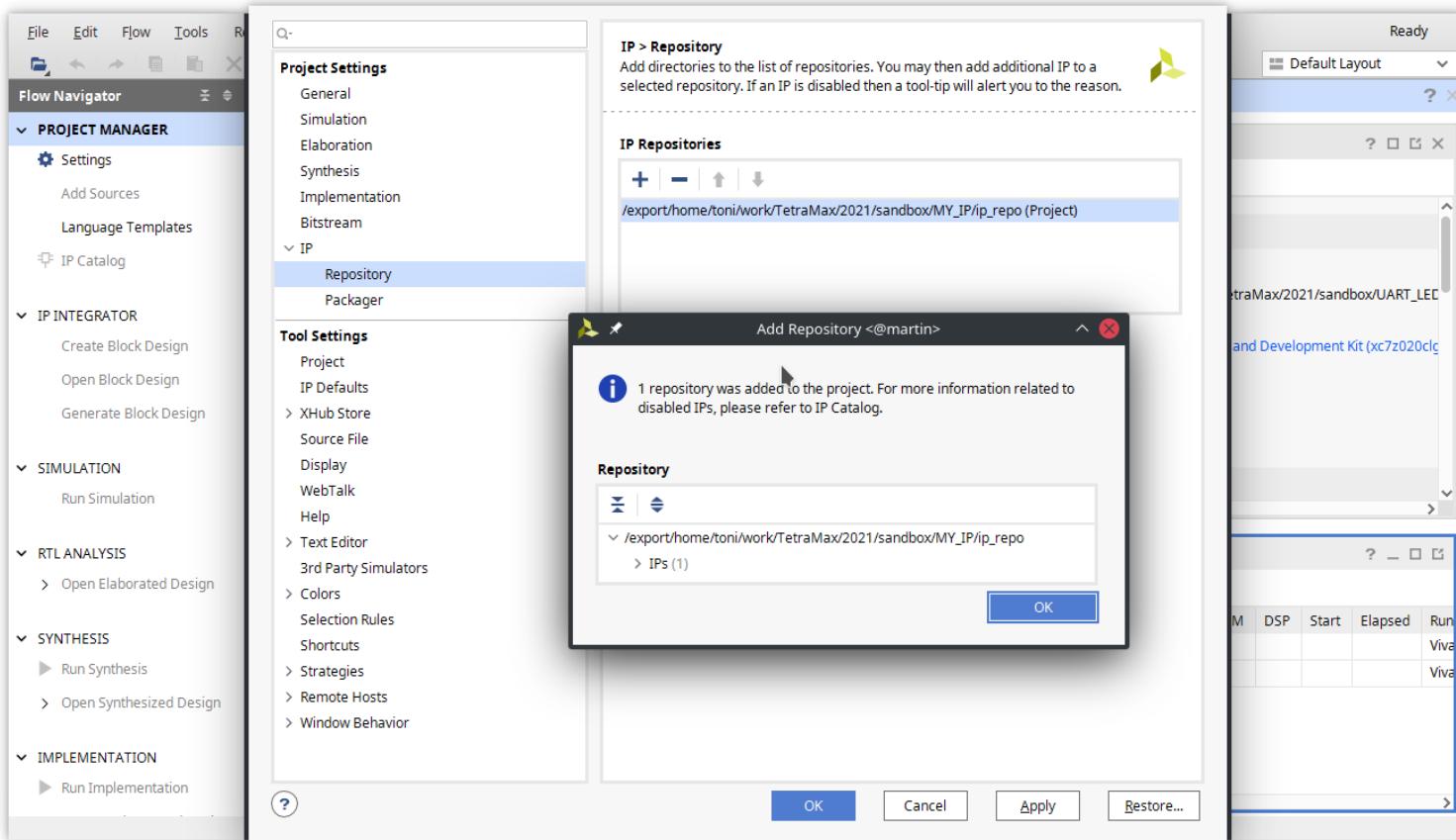
The screenshot shows the Vivado HLS Editions software interface. The top menu bar includes File, Flow, Tools, Window, Help, and Quick Access. The main window title is "VIVADO HLS Editions". A sub-header "Default Part" is displayed, with the instruction "Choose a default Xilinx part or board for your project." Below this, there are two tabs: "Parts" and "Boards". Under "Boards", there is a "Reset All Filters" button and dropdown menus for "Vendor" (set to "All") and "Name" (set to "All"). A search bar with placeholder text "Search: Q-" is also present. A large list of boards is shown, with the "ZedBoard Zynq Evaluation and Development Kit" selected and highlighted in blue. This board's details are displayed in a preview pane: "Display Name" (ZedBoard Zynq Evaluation and Development Kit), "Preview" (image of the ZedBoard), and "Version" (V1.0). Below this, another board entry is visible: "ZYNQ-7 TE0745-\*-81C11-A-35-1C (1GB DDR) \*SPRT PCB: REV02, REV01". At the bottom of the board list are navigation buttons: "< Back", "Next >", "Finish", and "Cancel".  

The screenshot shows the "New Project Summary" screen. The title "New Project Summary" is at the top. It contains two informational sections. The first section states: "A new RTL project named 'UART\_LED\_IP' will be created." The second section provides detailed information about the project: "The default part and product family for the new project: Default Board: ZedBoard Zynq Evaluation and Development Kit Default Part: xc7z020clg484-1 Product: Zynq-7000 Family: Zynq-7000 Package: clg484 Speed Grade: -1". At the bottom right, there is a message: "To create the project, click Finish". Navigation buttons at the bottom include "?", "< Back", "Next >", "Finish", and "Cancel".

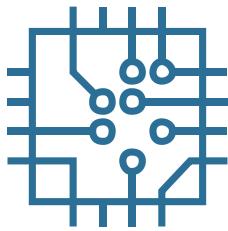
# IP Core Project



- Add new IP repository

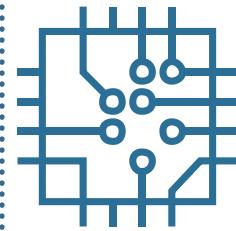


# IP Core Project

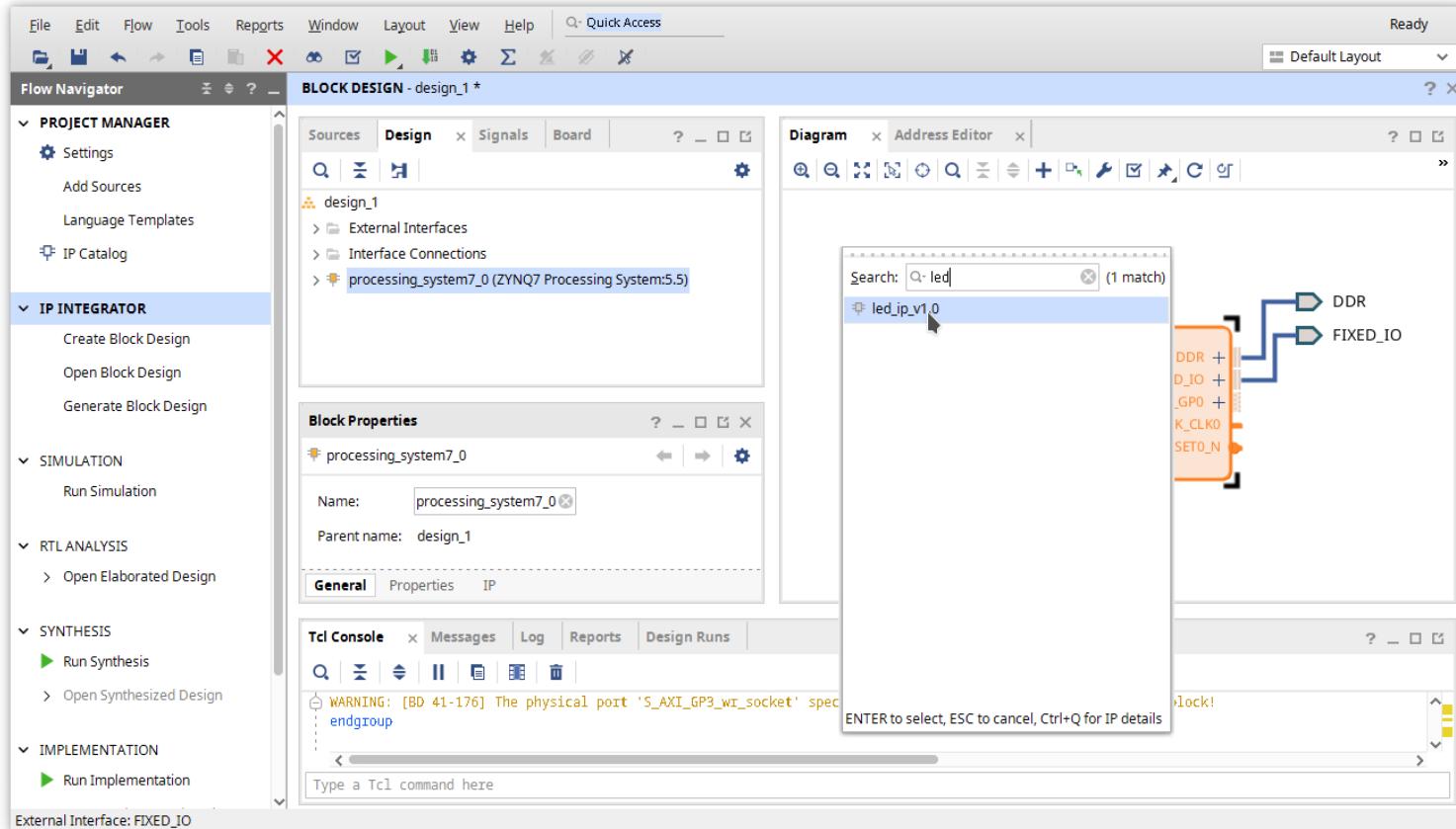


- Create block design

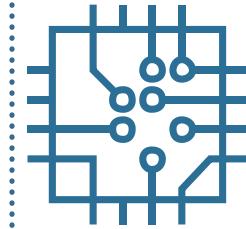
# IP Core Project



- Add and configure zynq



# IP Core Project



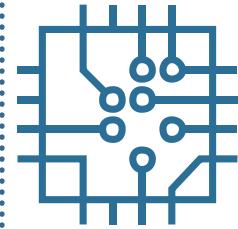
- Connect all cores

The screenshot shows the Xilinx Vivado interface for a Block Design project named "design\_1".

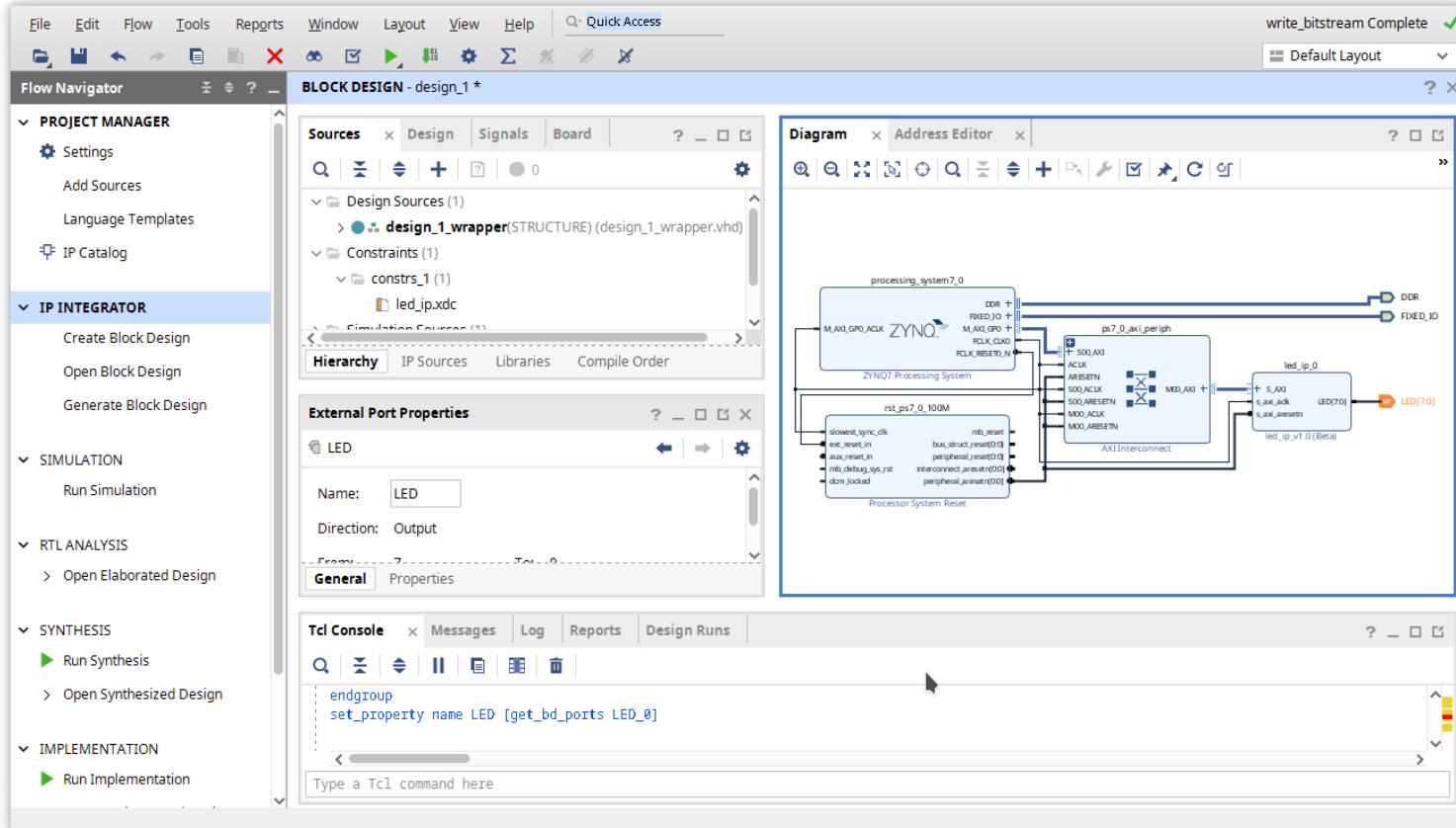
- Flow Navigator:** On the left, under the "IP INTEGRATOR" section, "Create Block Design" is selected.
- Diagram:** The central workspace displays a "processing\_system7\_0" block (ZYNQ) connected to an "led\_ip\_0" IP core. The "Diagram" tab is active, showing a message: "Designer Assistance available. Run Connection Automation".
- Tcl Console:** The bottom panel shows the command history:

```
Adding component instance block -- xilinx.com:ip:processing_system7:5.5 - processing_system7_0
WARNING: [BD 41-176] The physical port 'S_AXI_GP2_rd_socket' specified in the portmap, is not found on the block!
WARNING: [BD 41-176] The physical port 'S_AXI_GP2_wr_socket' specified in the portmap, is not found on the block!
WARNING: [BD 41-176] The physical port 'S_AXI_GP3_rd_socket' specified in the portmap, is not found on the block!
WARNING: [BD 41-176] The physical port 'S_AXI_GP3_wr_socket' specified in the portmap, is not found on the block!
SuccessFully read diagram <design_1> from BD file </export/home/toni/work/TetraMax/2021/sandbox/UART_LED_IP/UART_LED_IP.scs/sources_1/bd/design_1/ip
```

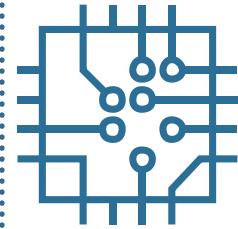
# IP Core Project



- Connect all cores (export LED pin)



# IP Core Project



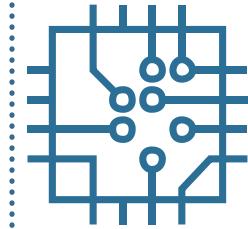
- Generate bitstream and export hardware

The screenshot shows the Vitis Block Design environment with the following details:

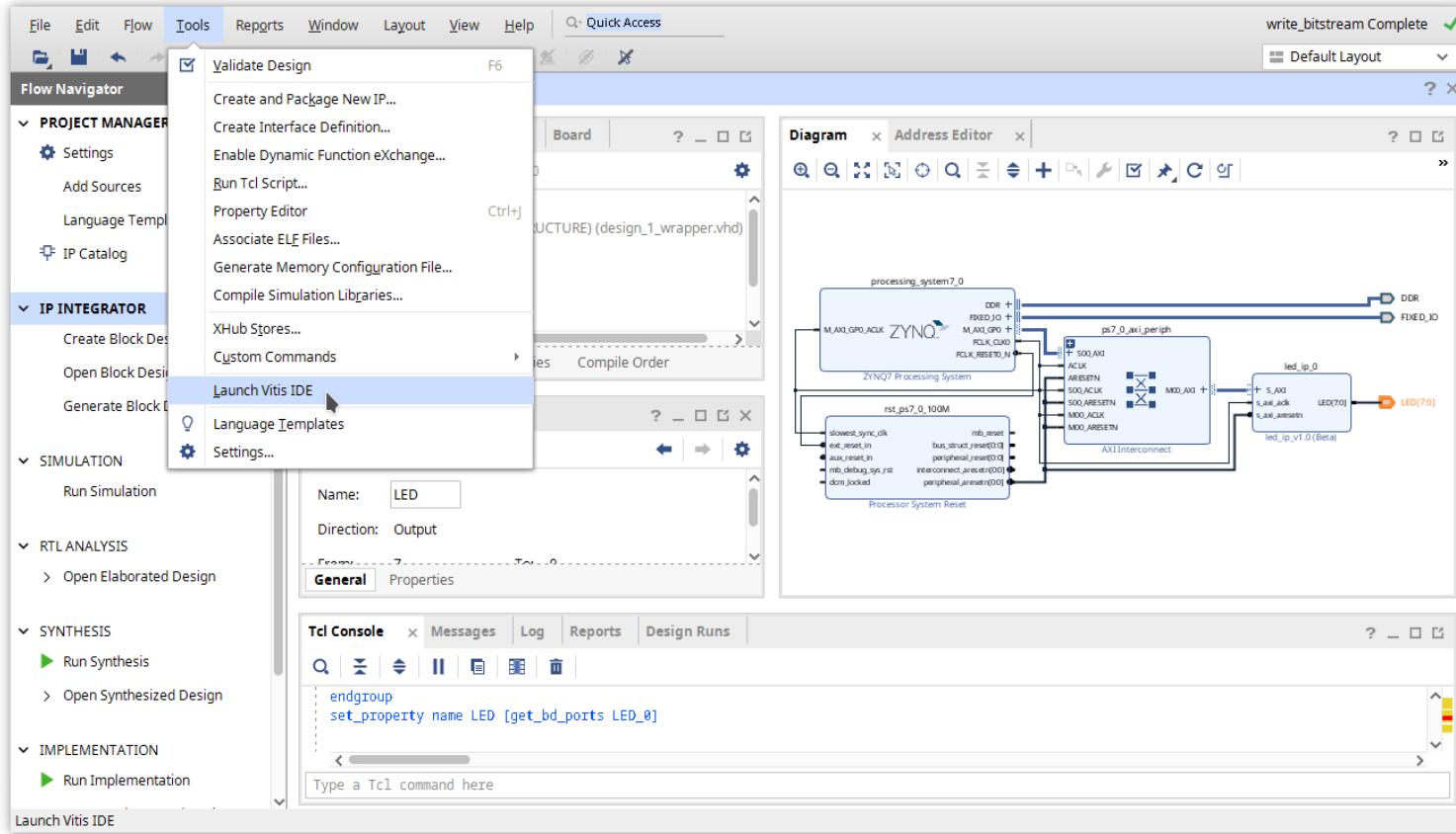
- File Menu:** Contains options like Project, Add Sources..., Save Block Design, Constraints, Simulation Waveform, Checkpoint, IP, Text Editor, Import, Export (highlighted), Print..., and Exit.
- Sources Tab:** Shows Design Sources (1) containing `design_1_wrapper(STRUCTURE)` (design\_1\_wrapper.vhd) and Constraints (1) containing `constrs_1 (1)` (led\_ip.xdc).
- Diagram Tab:** Displays a block diagram of the ZYNQ Processing System. It includes components like `processing_system7_0`, `rst_ps7_0_100M`, `ps7_0_axi_periph`, and `led_ip_0`. The `led_ip_0` component is connected to an LED labeled `LED[7:0]`.
- Tcl Console Tab:** Shows Tcl commands being run, including `endgroup` and `set_property name LED [get_bd_ports LED_0]`.
- Message Bar:** Shows "write\_bitstream Complete" with a green checkmark.
- Status Bar:** Shows "Default Layout".

A tooltip at the bottom left says: "Export a hardware description file for use with the Vitis."

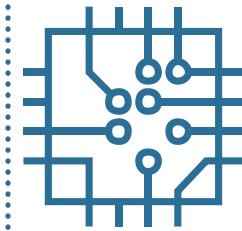
# IP Core Project



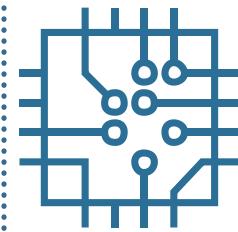
- Start Vitis SDK



# Outline



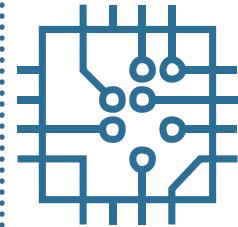
- Part 1
  - FPGA structure and design platform
  - VHDL hardware design in FPGA
  - Embedded system design on FPGA
- Part 2
  - Embedded software design FPGA
  - IP core development and integration
- Part 3
  - **Software and hardware debugging**



# Vitis: Software debugging

- Is part of the embedded toolkit
- Is supported over programming interface (JTAG)
- Can be performed remotely
- Applications must be build with enabled debugging
- Debugging is enabled by default

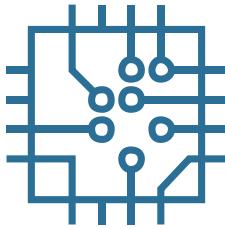
# Vitis: Software debugging



- Start debugging session

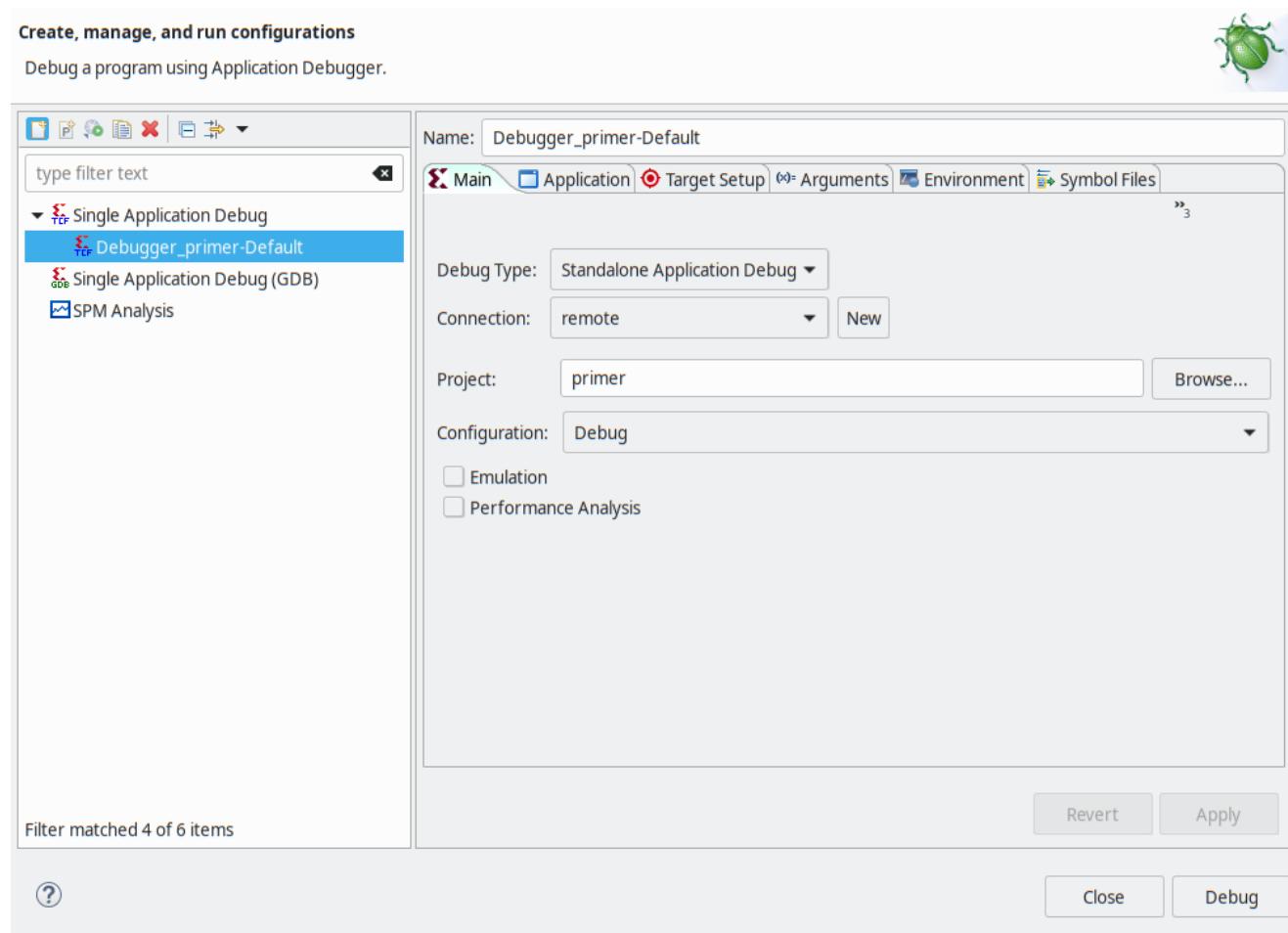
The screenshot shows the Vitis IDE interface with the following details:

- File Menu:** File, Edit, Run, Search, Xilinx, Project, Window, Help.
- Toolbars:** Standard toolbar with icons for file operations, search, and run.
- Explorer View:** Shows the project structure:
  - hw
  - logs
  - ps7\_cortexa9\_0
    - resources
  - zynq\_fsbl
  - platform.spr
  - primer\_system [platform\_2]
  - primer [standalone on ps7\_cortexa9\_0]
    - Binaries
    - Includes
      - /opt/Xilinx/Vitis/2020.2/include
      - /opt/Xilinx/Vitis/2020.2/include/fpga
      - /opt/Xilinx/Vitis/2020.2/include/fpga/ps7
      - platform\_2/export
    - Debug
    - src
      - devcfg\_header.h
      - dmaps\_header.h
      - gpio\_header.h
      - qspis\_header.h
      - scugic\_header.h
      - scutimer\_header.h
      - scuwdt\_header.h
      - testperiph.c
      - ttcps\_header.h
      - xdevcfg\_selftest\_example.c
      - xdmmaps\_example.c
      - yunio\_tapp\_example.c
- Code Editor:** Displays the file `testperiph.c` with code related to GPIO initialization and reading.
- Properties View:** Shows the properties for the selected file.
- Variables View:** A table showing variables with their types and values.
- Bottom Bar:** Includes tabs for Serial Terminal, Executables, Vitis Log, Problems, Debugger Console, and a status message "Serial - ARM Cortex-A9 MPCore #1".
- Bottom Dock:** Shows a list of recent configurations:
  - 1 Launch on Hardware (Single Application Debug)
  - 2 Launch on Emulator (Single Application Debug)
  - 3 Launch on Hardware (Single Application Debug (GDB))

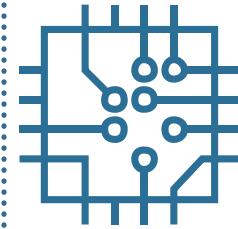


# Vitis: Software debugging

- Configuration is the same as in running



# Vitis: Software debugging

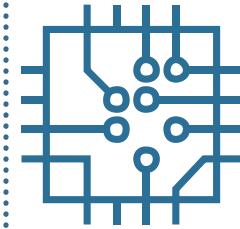


- Setting breakpoint

The screenshot shows the Vitis IDE interface with the following components:

- File Menu:** File, Edit, Run, Search, Xilinx, Project, Window, Help.
- Toolbars:** Standard toolbar with icons for file operations, search, and run.
- Left Sidebar:** Explorer pane showing project structure, Assistant pane, and a list of generated files like platform\_2/export/platform\_2/sw/pl/testperiph.c, devcfg\_header.h, etc.
- Code Editor:** The main window displays the source code for `testperiph.c`. A context menu is open over the first few lines of code, showing options such as "Toggle Breakpoint", "Add Breakpoint...", "Add Dynamic Printf...", "Disable Breakpoint", "Breakpoint Properties...", "Breakpoint Types", "Go to Annotation", "Add Bookmark...", "Add Task...", "Show Quick Diff", "Show Line Numbers", "Folding", and "Preferences...".
- Bottom Status Bar:** Shows "Build Console [primer, Debug]" and "08:51:18 Build Finished (took 340ms)".
- Right Side Panels:**
  - Memory:** A panel for monitoring memory contents.
  - XSCT Con... / Emulatio...:** Panels for system test and emulation configuration.
  - QEMU Process:** A terminal-like window for running QEMU processes.
- Bottom Navigation:** Writable, Smart Insert, and a line number indicator (35:1).

# Vitis: Software debugging



- Stepping into

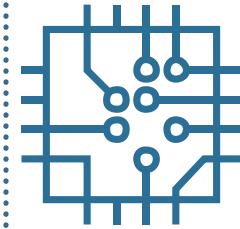
The screenshot shows the Vitis IDE interface with the following details:

- File Menu:** File, Edit, Run, Search, Xilinx, Project, Window, Help.
- Toolbar:** Includes icons for file operations, search, run, and debug.
- Step Into (F5) button:** A callout points to the "Step Into (F5)" button in the toolbar.
- Explorer View:** Shows the project structure under "platform\_2/export/platform\_2/sw/pl".
- Code Editor:** Displays the content of `testperiph.c`. The code initializes GPIO pins and performs a loop to read push button status.

```
26 * SDK application project when you run the "Generate Libraries" menu item
27 *
28 */
29
30 #include <stdio.h>
31 #include "xparameters.h"
32 #include "xgpio.h"
33 #include "gpio_header.h"
34 int main ()
35 {
36     print("---Entering main---\n\r");
37
38     XGpio dip, push;
39     int i, j, psb_check, dip_check;
40
41     print("\r\n-- Start of the Program --\r\n");
42
43     XGpio_Initialize(&dip, XPAR_GPIO_0_DEVICE_ID);
44     XGpio_SetDataDirection(&dip, 1, 0xffffffff);
45     XGpio_Initialize(&push, XPAR_GPIO_1_DEVICE_ID);
46     XGpio_SetDataDirection(&push, 1, 0xffffffff);
47
48     for (j=40; j> 0; j--) {
49         psb_check = XGpio_DiscreteRead(&push, 1);
50         xil_printf("Push Buttons Status %x\r\n", psb_check);
51         dip_check = XGpio_DiscreteRead(&dip, 1);
52     }
53 }
```

- Quick Access:** Includes icons for V, B, E, M, and R.
- Design Tab:** Shows a schematic view.
- Debug Tab:** Shows memory dump and QEMU process logs.
- Console Tab:** Shows build logs.
- Bottom Status Bar:** Writable, Smart Insert, 35 : 1.

# Vitis: Software debugging

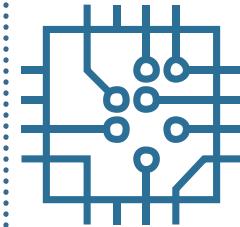


- Step over

The screenshot shows the Vitis IDE interface with the following components:

- File Bar:** File, Edit, Run, Search, Xilinx, Project, Window, Help.
- Toolbar:** Includes icons for file operations, search, and various tools.
- Step Over (F6) Button:** A button highlighted with a black box, indicating the current step-over operation.
- Explorer View:** Shows the project structure with files like platform\_2/export/platform\_2/sw/pl, src, and \_ide. The file `testperiph.c` is selected and highlighted with a blue bar.
- Code Editor:** Displays the C code for `testperiph.c`. The code initializes GPIO pins and prints push button status. The line `print("...Entering main---\n\r");` is currently being stepped over.
- Memory View:** A tab showing memory dump and visualization.
- Console View:** Shows build logs and terminal output. It displays "Build Finished (took 340ms)" and a QEMU process window.
- Design View:** Shows a schematic or board-level view.
- Debug View:** Shows simulation or debugger-related information.

# Vitis: Software debugging

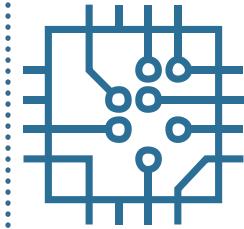


- Step return

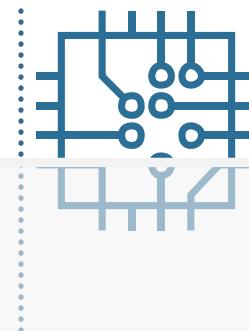
The screenshot shows the Vitis IDE interface with the following components:

- File Bar:** File, Edit, Run, Search, Xilinx, Project, Window, Help.
- Toolbar:** Includes icons for file operations, search, and various tools.
- Explorer View:** Shows the project structure under "platform\_2/export/platform\_2/sw/pl". Files listed include: Debug, src (devcfg\_header.h, dmaps\_header.h, gpio\_header.h, qspips\_header.h, scugic\_header.h, scutimer\_header.h, scuwdt\_header.h), testperiph.c, ttcps\_header.h, xdevcfg\_selftest\_example.c, xdmaps\_example\_w\_intr.c, xgpio\_tapp\_example.c, xqspips\_selftest\_example.c, xscugic\_tapp\_example.c, xscutimer\_intr\_example.c, xscutimer\_polled\_example.c, xscuwdt\_intr\_example.c, xttcps\_tapp\_example.c, lscript.ld, Xilinx.spec, \_ide (primer.pj), Debug, and primer\_system.sprj. "testperiph.c" is currently selected.
- Code Editor:** Displays the C code for "testperiph.c". The code initializes GPIO pins and prints their status in a loop. A tooltip "Step Return (F7)" points to the F7 key on the keyboard.
- Quick Access:** Buttons for Design and Debug.
- Design View:** A large empty area for circuit design.
- Memory View:** A window showing memory dump and analysis.
- Console View:** Shows build logs and terminal output. Log message: "08:51:18 Build Finished (took 340ms)".
- XSCT Console:** Shows QEMU Process output.

# Vivado: hardware debuggin



- ILA – embeded logic analyser
- Debug purely hardware solutions (PL)
- AXI\_ila core enabled hardware debugging of embedded hardware (custom cores)



# Hvala za vašo pozornost!



REPUBLIKA SLOVENIJA  
MINISTRSTVO ZA GOSPODARSKI  
RAZVOJ IN TEHNOLOGIJO