Знакомство с Vitis

СОЗДАЕМ СВОЙ ПРОЕКТ

Проектирование цифровой техники с применением ПЛИС и аппаратного языка разработки System Verilog

Н. Г. Зайцев

Кандидат технических наук, преподаватель кафедры КИПР ТУСУР, начальник сектора цифровой электроники ООО «ЛЭМЗ-Т»

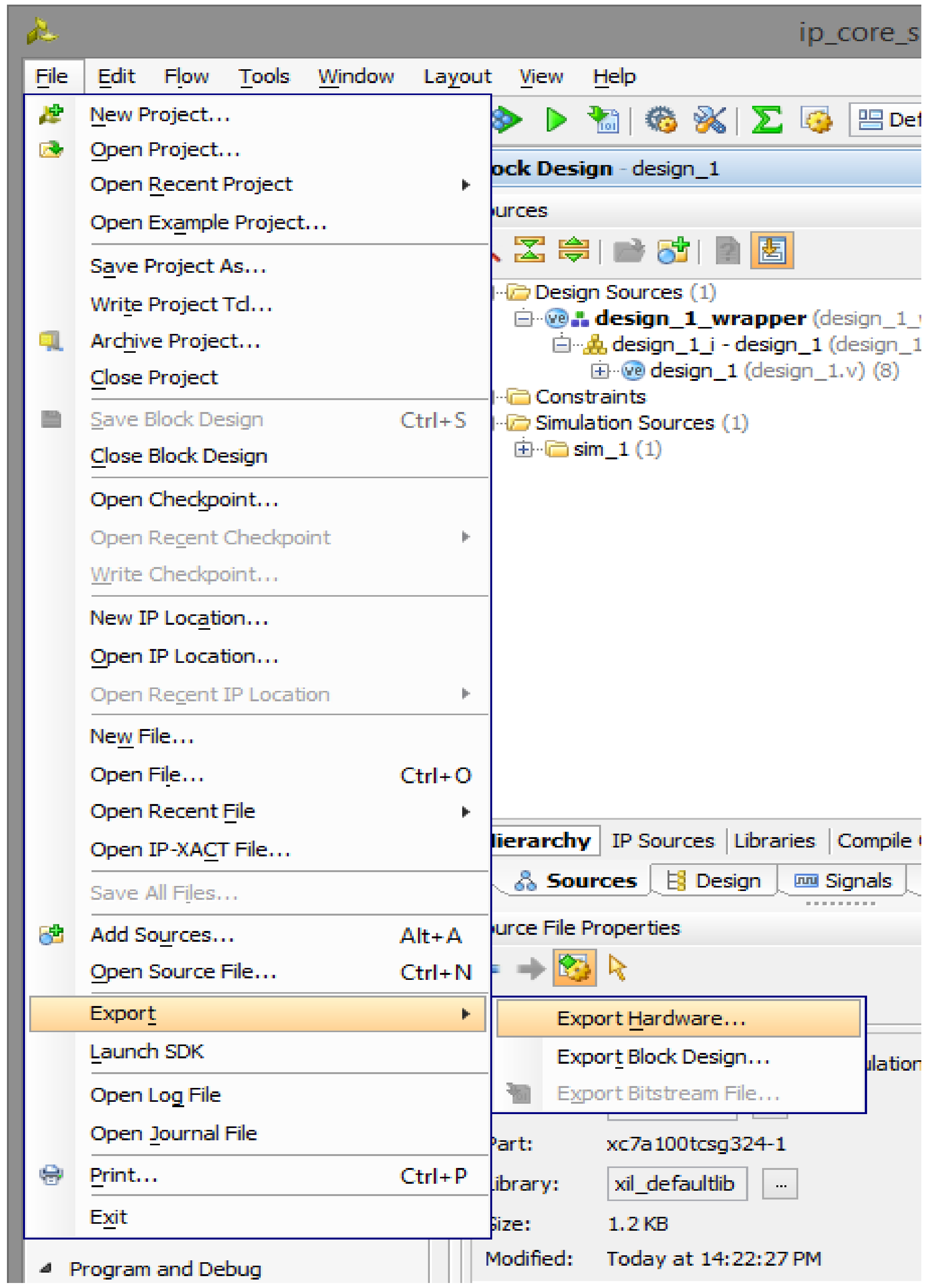
М. В. Кулешов

Ведущий инженер-электроник ООО «ЛЭМЗ-Т»

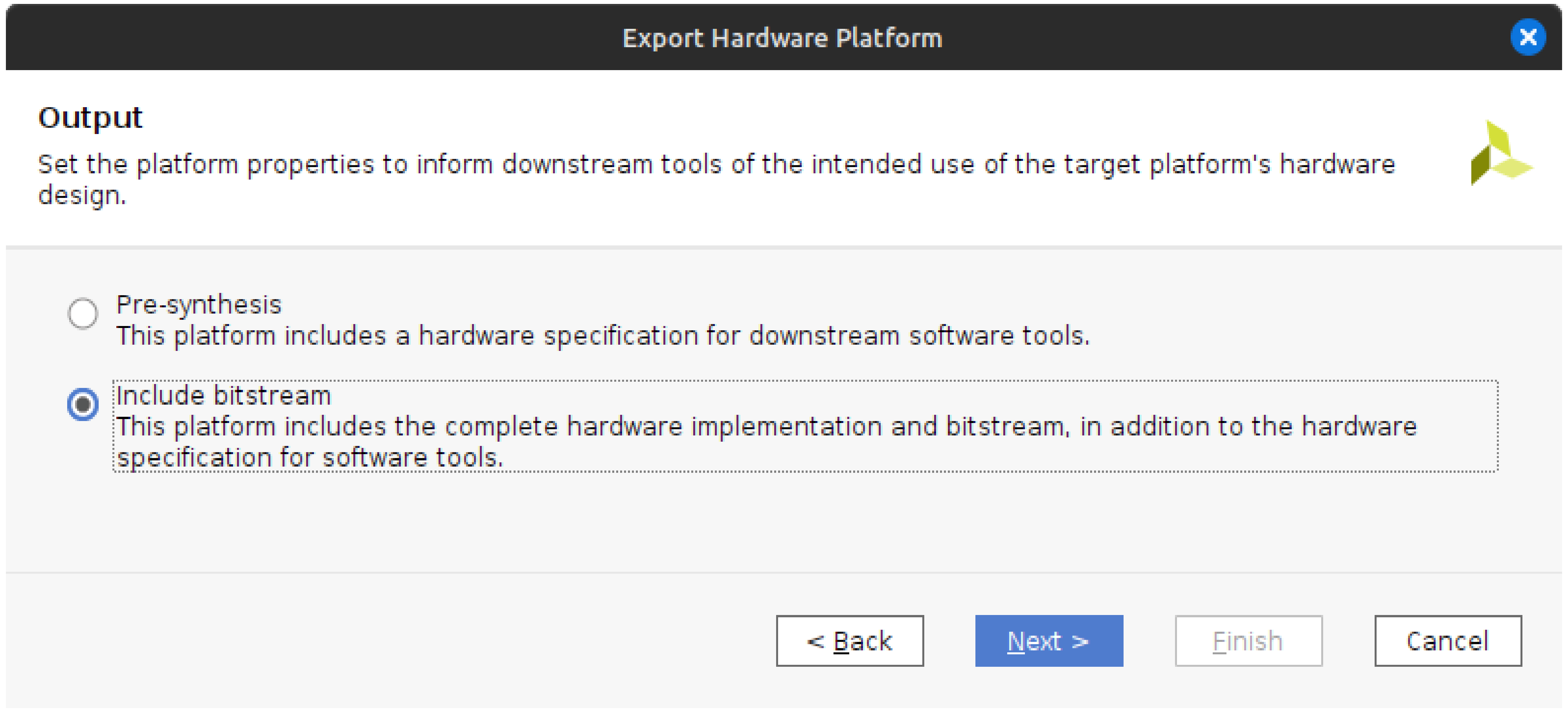
Я. В. Непомнящих

Ведущий инженер-программист ООО «ЛЭМЗ-Т»

Знакомство с Vitis



Знакомство с Vitis



Export Hardware Platform

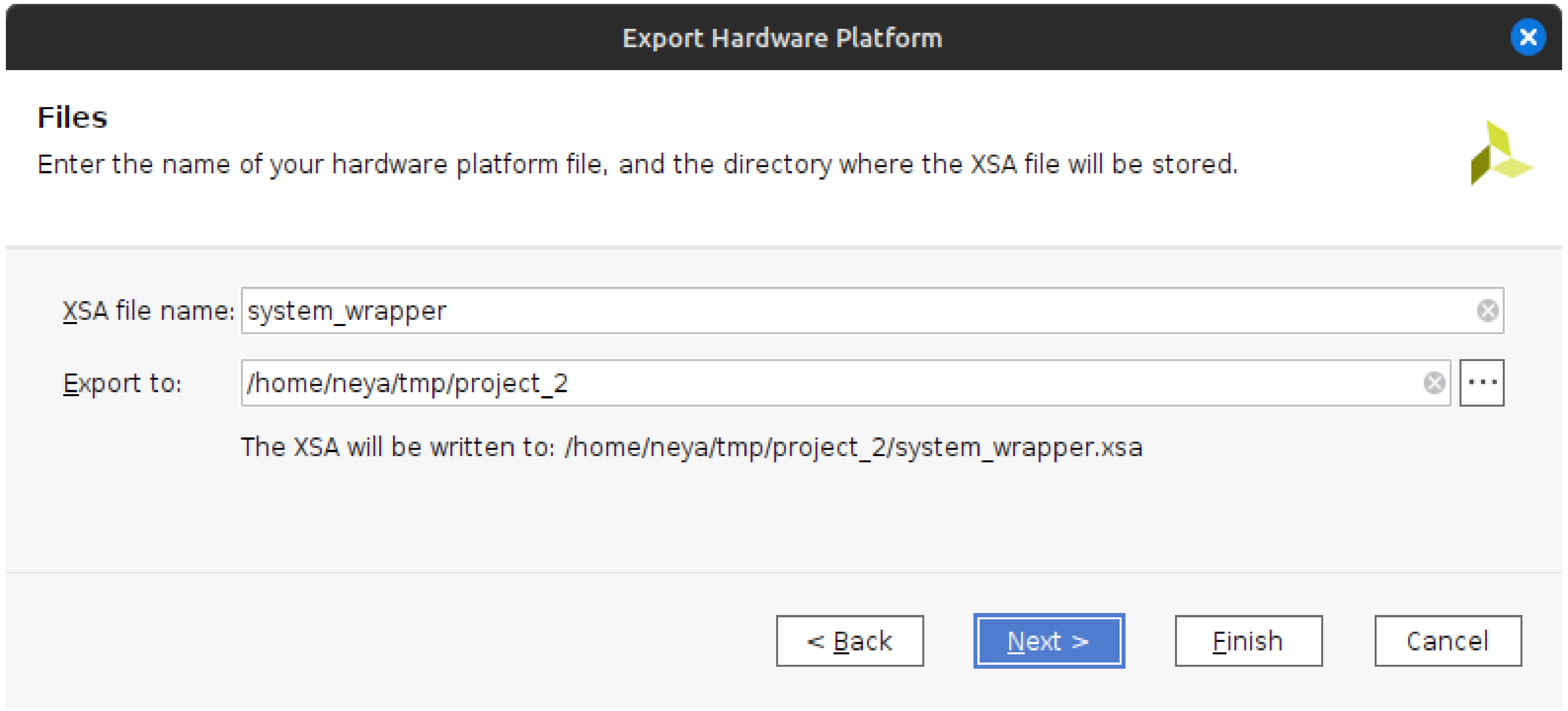
Output

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 software tools.

Знакомство с Vitis



Files

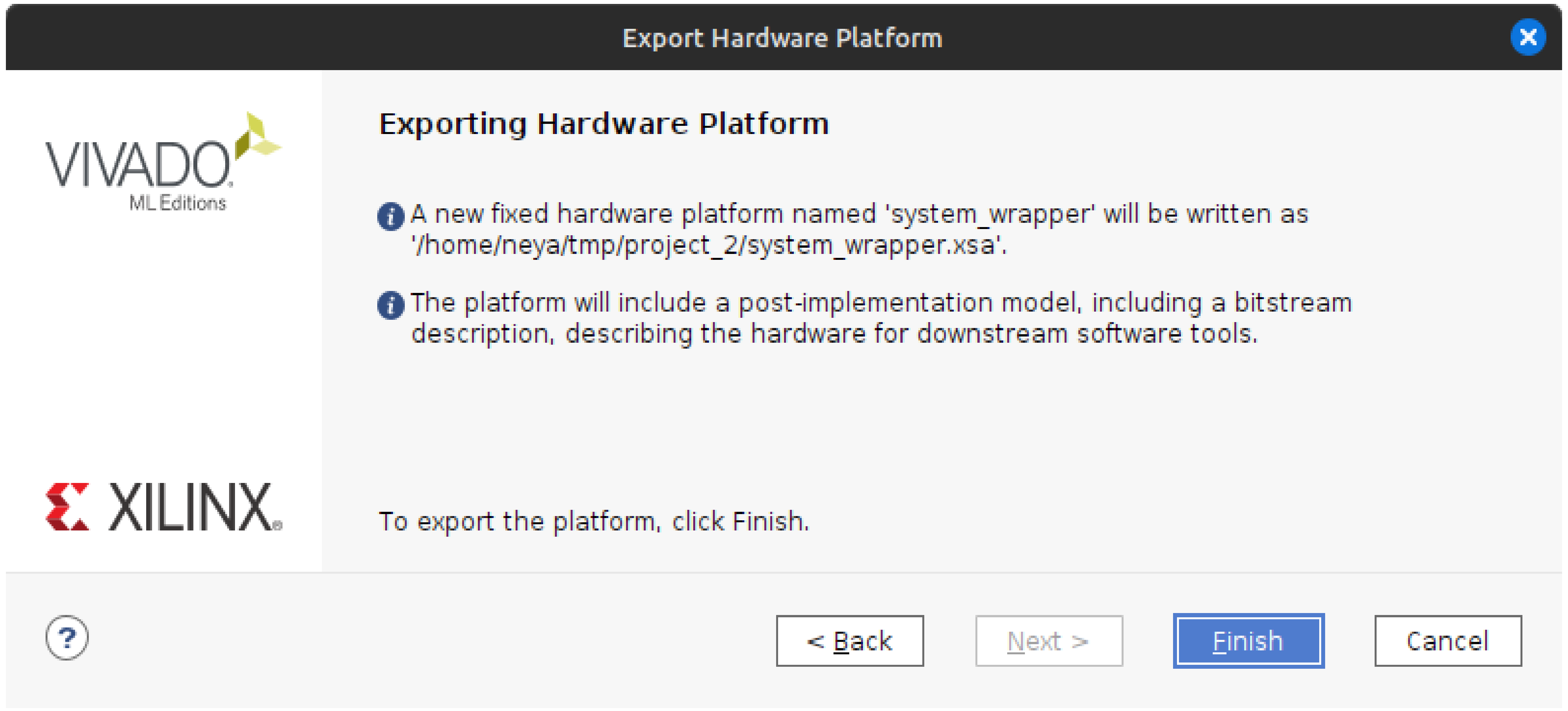
Enter the name of your hardware platform file, and the directory where the XSA file will be stored.

XSA file name: system_wrapper

Export to: /home/neya/tmp/project_2

The XSA will be written to: /home/neya/tmp/project_2/system_wrapper.xsa

Знакомство с Vitis



Export Hardware Platform

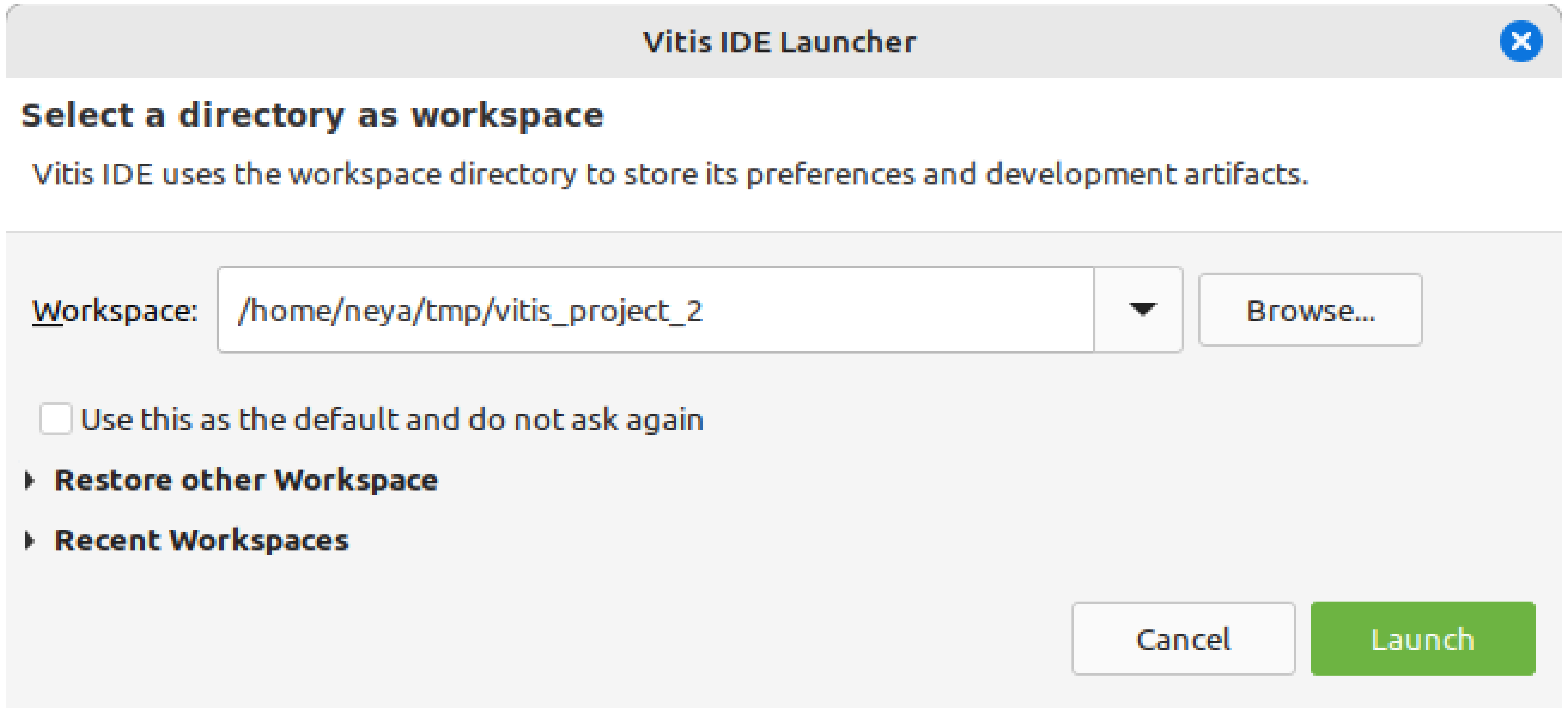
Exporting Hardware Platform

A new fixed hardware platform named 'system_wrapper' will be written as '/home/neya/tmp/project_2/system_wrapper.xsa'.

The platform will include a post-implementation model, including a bitstream description, describing the hardware for downstream software tools.

To export the platform, click Finish.

Знакомство с Vitis



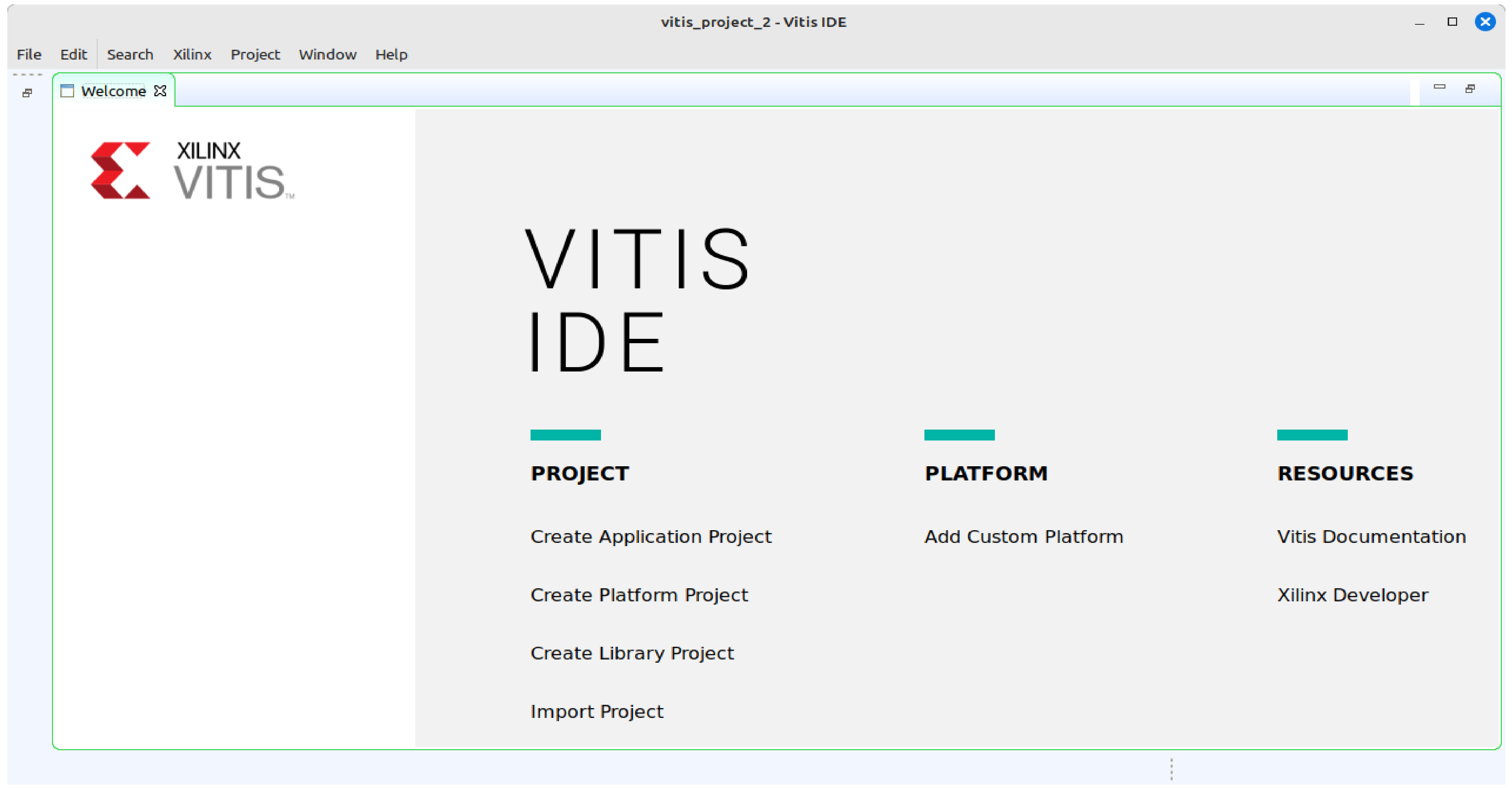
Select a directory as workspace

Vitis IDE uses the workspace directory to store its preferences and development artifacts.

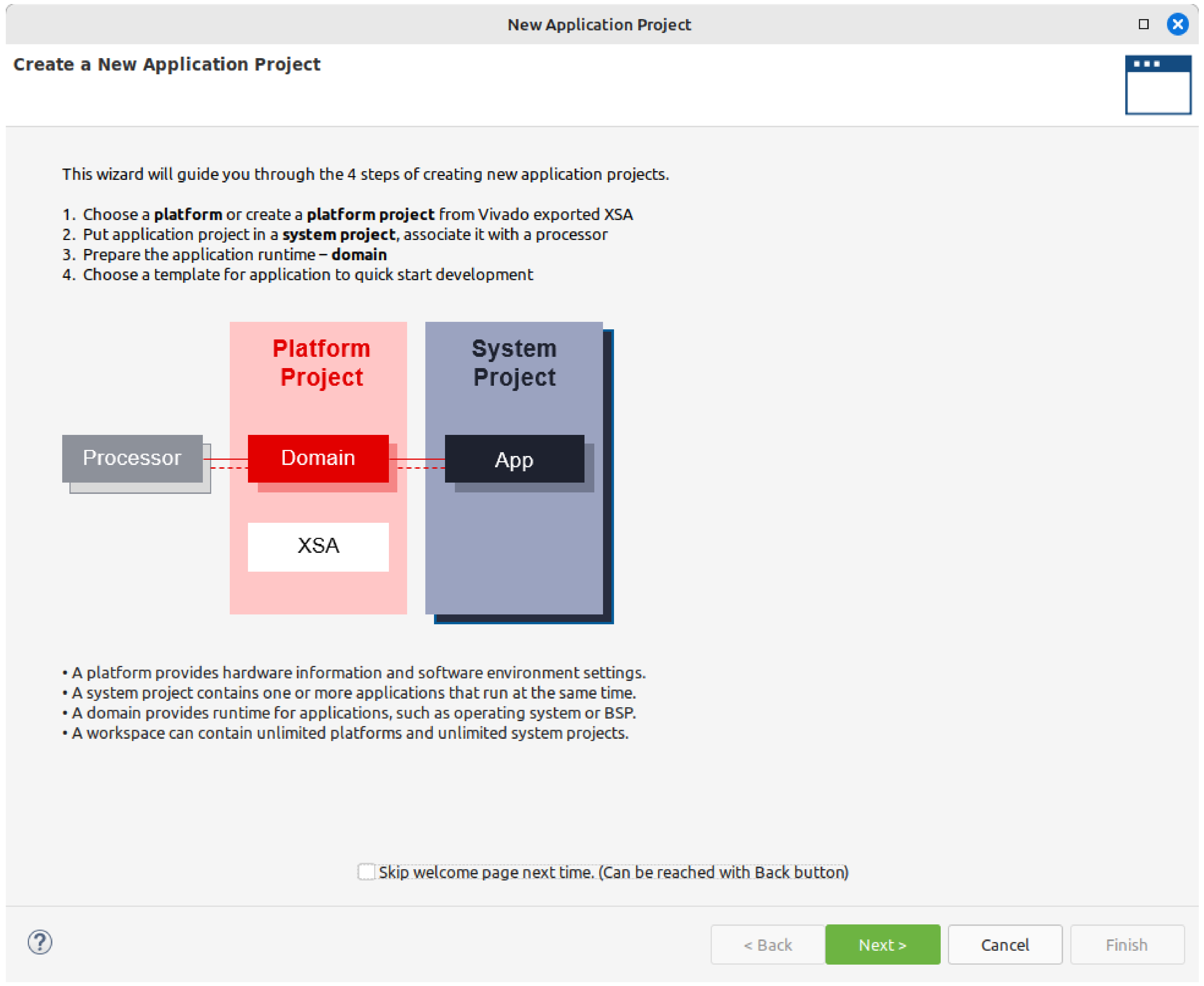
Workspace: /home/neya/tmp/vitis_project_2

☐ Use this as the default and do not ask again

Знакомство с Vitis



Знакомство с Vitis



New Application Project

Create a New Application Project

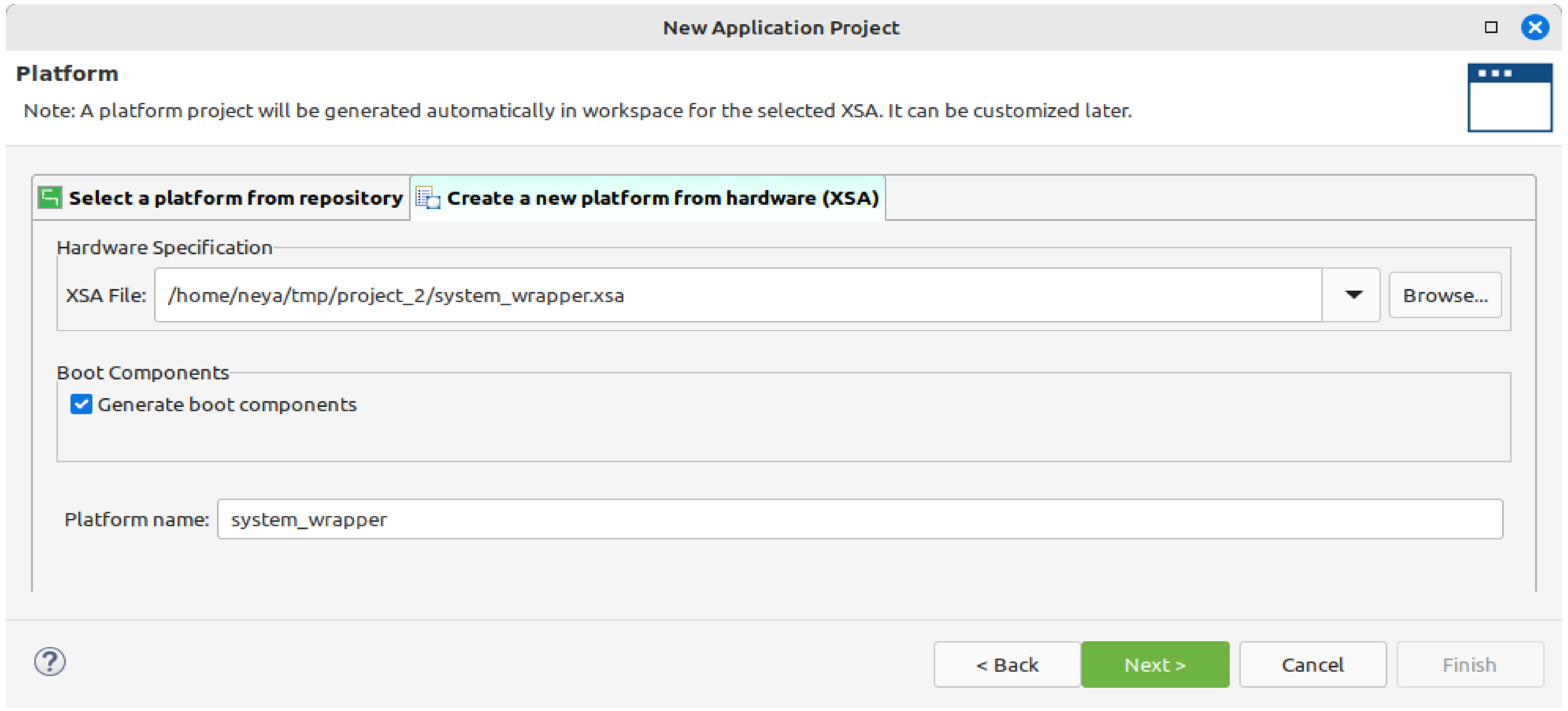
This wizard will guide you through the 4 steps of creating new application projects.

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

- A platform provides hardware information and software environment settings.
- A system project contains one or more applications that run at the same time.
- A domain provides runtime for applications, such as operating system or BSP.
- A workspace can contain unlimited platforms and unlimited system projects.

☐ Skip welcome page next time. (Can be reached with Back button)

Знакомство с Vitis


New Application Project

Platform

Note: A platform project will be generated automatically in workspace for the selected XSA. It can be customized later.

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

Hardware Specification

XSA File: /home/neya/tmp/project_2/system_wrapper.xsa

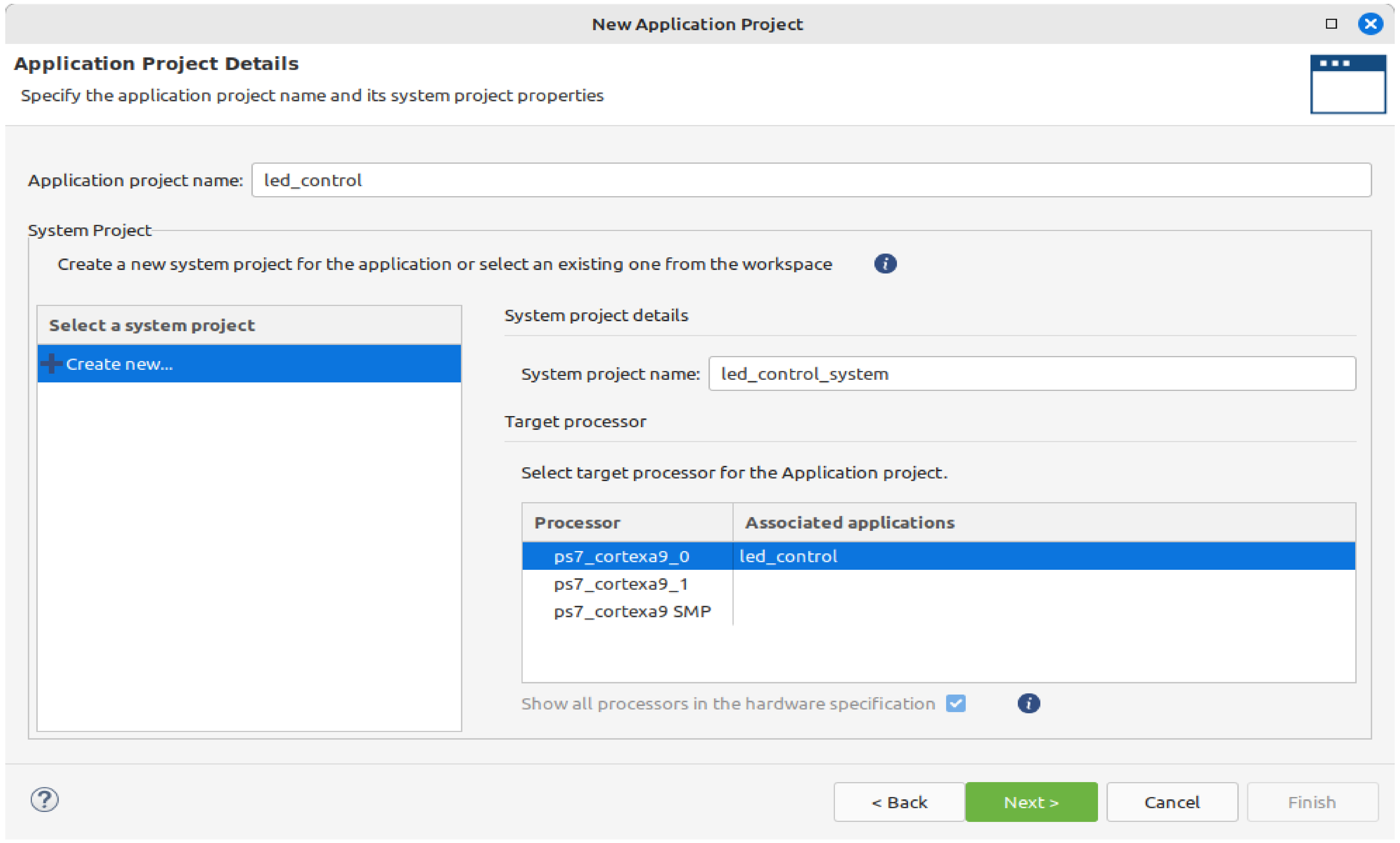
Boot Components

[✓] Generate boot components

Platform name: system_wrapper

< Back
Next >
Cancel
Finish

Знакомство с Vitis


New Application Project

Application Project Details

Specify the application project name and its system project properties

Application project name: led_control

System Project

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

Select a system project

Create new...

System project details

System project name: led_control_system

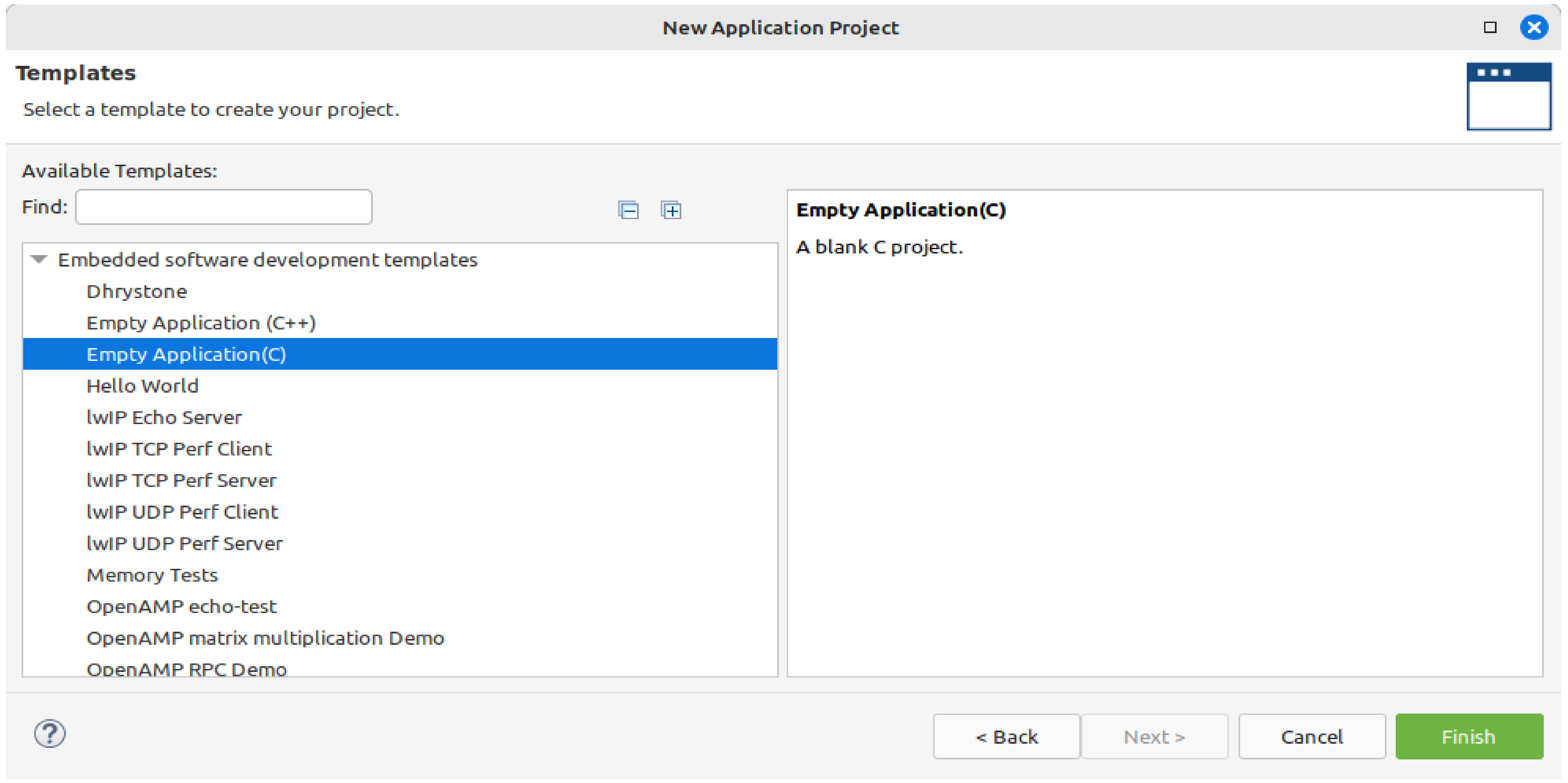
Target processor

Select target processor for the Application project.

Processor	Associated applications
ps7_cortexa9_0	led_control
ps7_cortexa9_1	
ps7_cortexa9 SMP	

Show all processors in the hardware specification ✓

Знакомство с Vitis


New Application Project

Templates

Select a template to create your project.

Available Templates:

Find:

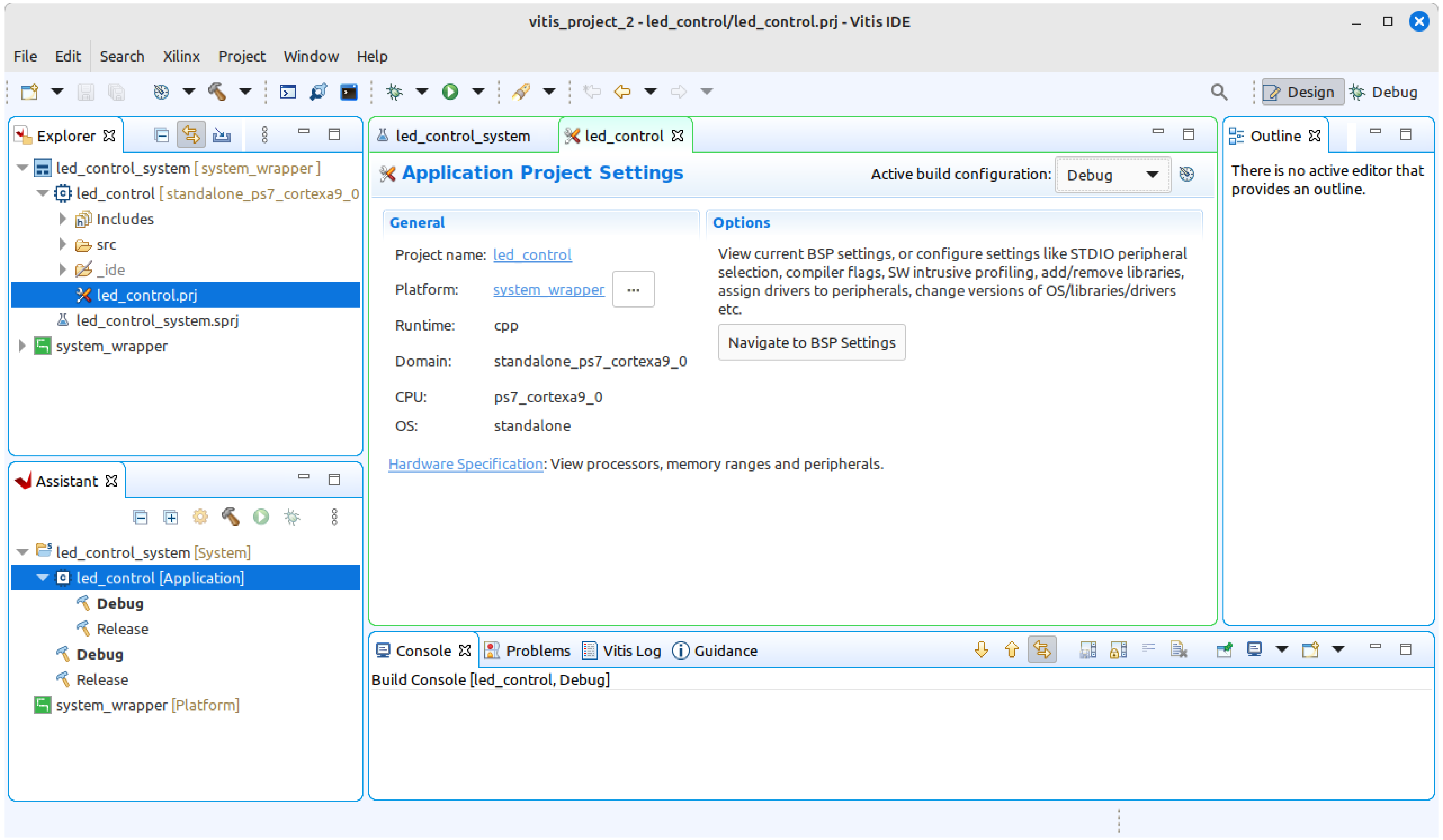
- Embedded software development templates
  - Dhrystone
  - Empty Application (C++)
  - Empty Application(C)
  - Hello World
  - lwIP Echo Server
  - lwIP TCP Perf Client
  - lwIP TCP Perf Server
  - lwIP UDP Perf Client
  - lwIP UDP Perf Server
  - Memory Tests
  - OpenAMP echo-test
  - OpenAMP matrix multiplication Demo
  - OpenAMP RPC Demo

Empty Application(C)

A blank C project.

< Back | Next > | Cancel | Finish

Знакомство с Vitis


vitis_project_2 - led_control/led_control.prj - Vitis IDE

Application Project Settings

Active build configuration: Debug

General

Project name:	led_control
Platform:	system_wrapper
Runtime:	cpp
Domain:	standalone_ps7_cortexa9_0
CPU:	ps7_cortexa9_0
OS:	standalone

Options

View current BSP settings, or configure settings like STDIO peripheral selection, compiler flags, SW intrusive profiling, add/remove libraries, assign drivers to peripherals, change versions of OS/libraries/drivers etc.

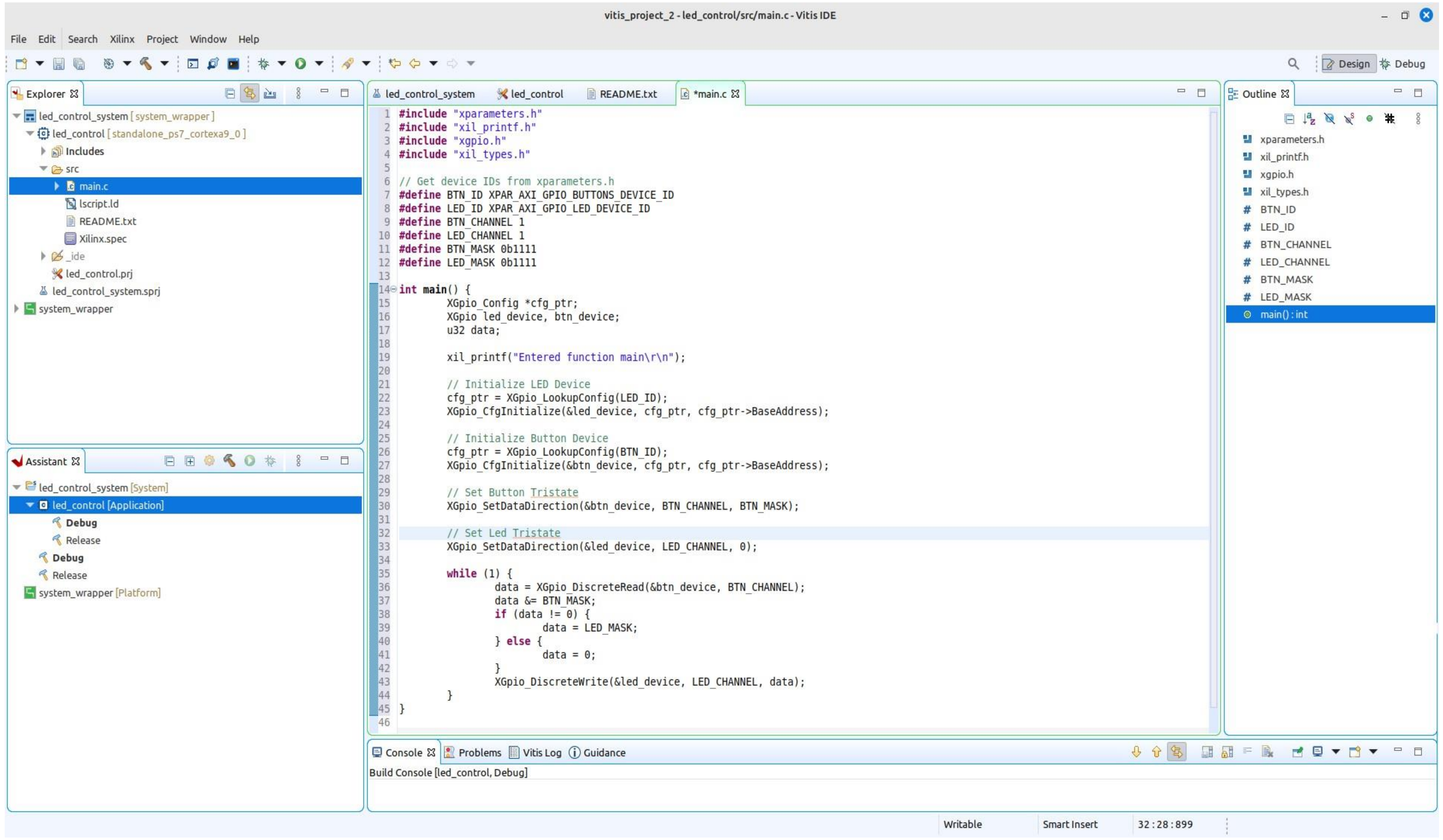
Navigate to BSP Settings

Hardware Specification: View processors, memory ranges and peripherals.

There is no active editor that provides an outline.

Build Console [led_control, Debug]

Знакомство с Vitis

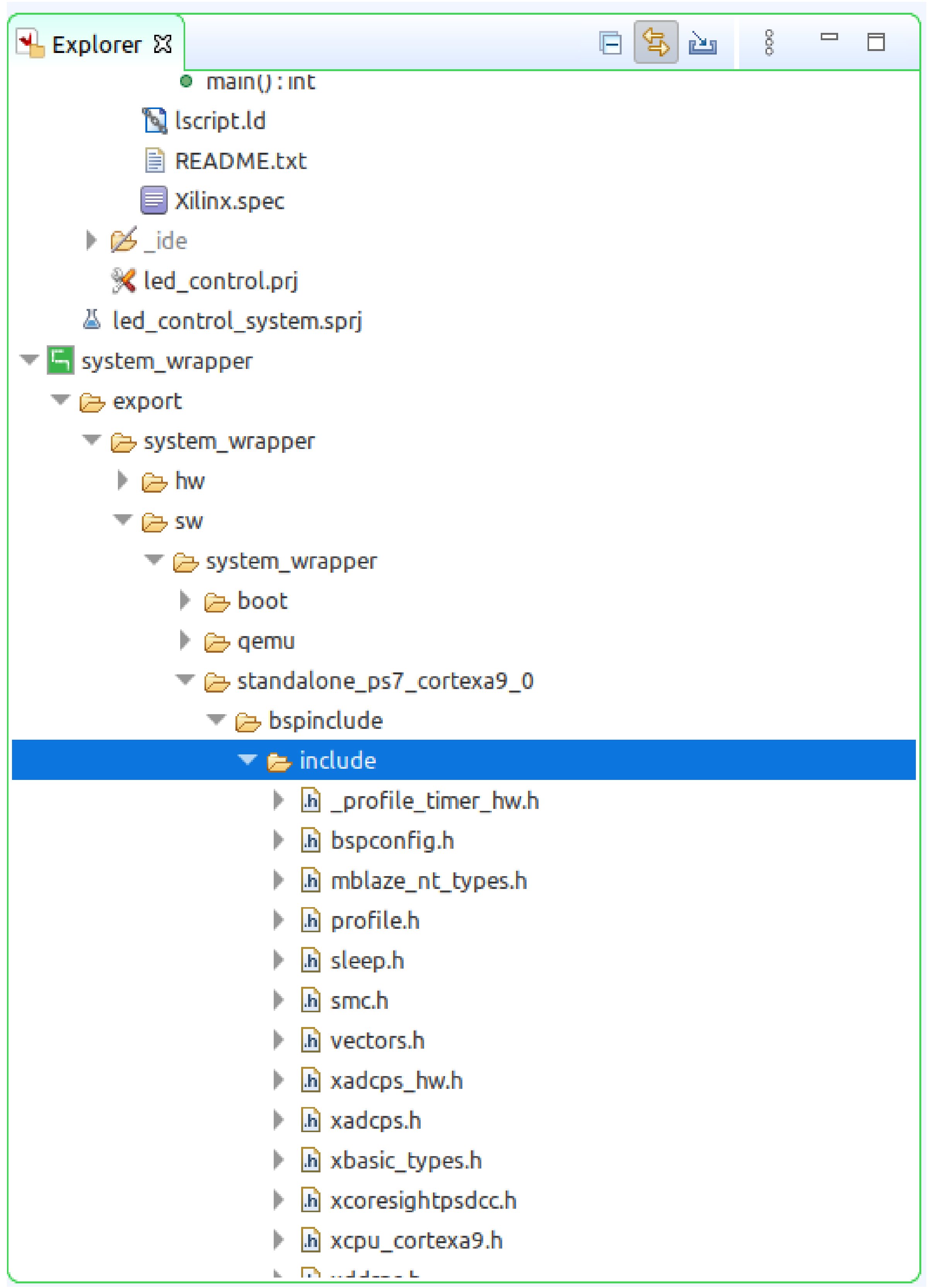


```c
#include "xparameters.h"
#include "xil_printf.h"
#include "xgpio.h"
#include "xil_types.h"

// Get device IDs from xparameters.h
#define BTN_ID XPAR_AXI_GPIO_BUTTONS_DEVICE_ID
#define LED_ID XPAR_AXI_GPIO_LED_DEVICE_ID
#define BTN_CHANNEL 1
#define LED_CHANNEL 1
#define BTN_MASK 0b1111
#define LED_MASK 0b1111

int main() {
	XGpio_Config *cfg_ptr;
	XGpio led_device, btn_device;
	u32 data;

	xil_printf("Entered function main\r\n");

	// Initialize LED Device
	cfg_ptr = XGpio_LookupConfig(LED_ID);
	XGpio_CfgInitialize(&led_device, cfg_ptr, cfg_ptr->BaseAddress);

	// Initialize Button Device
	cfg_ptr = XGpio_LookupConfig(BTN_ID);
	XGpio_CfgInitialize(&btn_device, cfg_ptr, cfg_ptr->BaseAddress);

	// Set Button Tristate
	XGpio_SetDataDirection(&btn_device, BTN_CHANNEL, BTN_MASK);

	// Set Led Tristate
	XGpio_SetDataDirection(&led_device, LED_CHANNEL, 0);

	while (1) {
		data = XGpio_DiscreteRead(&btn_device, BTN_CHANNEL);
		data &= BTN_MASK;
		if (data != 0) {
			data = LED_MASK;
		} else {
			data = 0;
		}
		XGpio_DiscreteWrite(&led_device, LED_CHANNEL, data);
	}
}
```

Знакомство с Vitis

```c
#include "xparameters.h"
#include "xil_printf.h"
#include "xgpio.h"
#include "xil_types.h"

// Get device IDs from xparameters.h
#define BTN_ID XPAR_AXI_GPIO_BUTTONS_DEVICE_ID
#define LED_ID XPAR_AXI_GPIO_LEDS_DEVICE_ID
#define BTN_CHANNEL 1
#define LED_CHANNEL 1
#define BTN_MASK 0b1111
#define LED_MASK 0b1111

int main() {
	XGpio_Config *cfg_ptr;
	XGpio led_device, btn_device;
	u32 data;

	xil_printf("Entered function main\r\n");

	// Initialize LED Device
	cfg_ptr = XGpio_LookupConfig(LED_ID);
	XGpio_CfgInitialize(&led_device, cfg_ptr, cfg_ptr->BaseAddress);

// Initialize Button Device
	cfg_ptr = XGpio_LookupConfig(BTN_ID);
	XGpio_CfgInitialize(&btn_device, cfg_ptr, cfg_ptr->BaseAddress);

	// Set Button Tristate
	XGpio_SetDataDirection(&btn_device, BTN_CHANNEL, BTN_MASK);

	// Set Led Tristate
	XGpio_SetDataDirection(&led_device, LED_CHANNEL, 0);

	while (1) {
		data = XGpio_DiscreteRead(&btn_device, BTN_CHANNEL);
		data &= BTN_MASK;
		if (data != 0) {
			data = LED_MASK;
		} else {
			data = 0;
		}
		XGpio_DiscreteWrite(&led_device, LED_CHANNEL, data);
	}
}
```

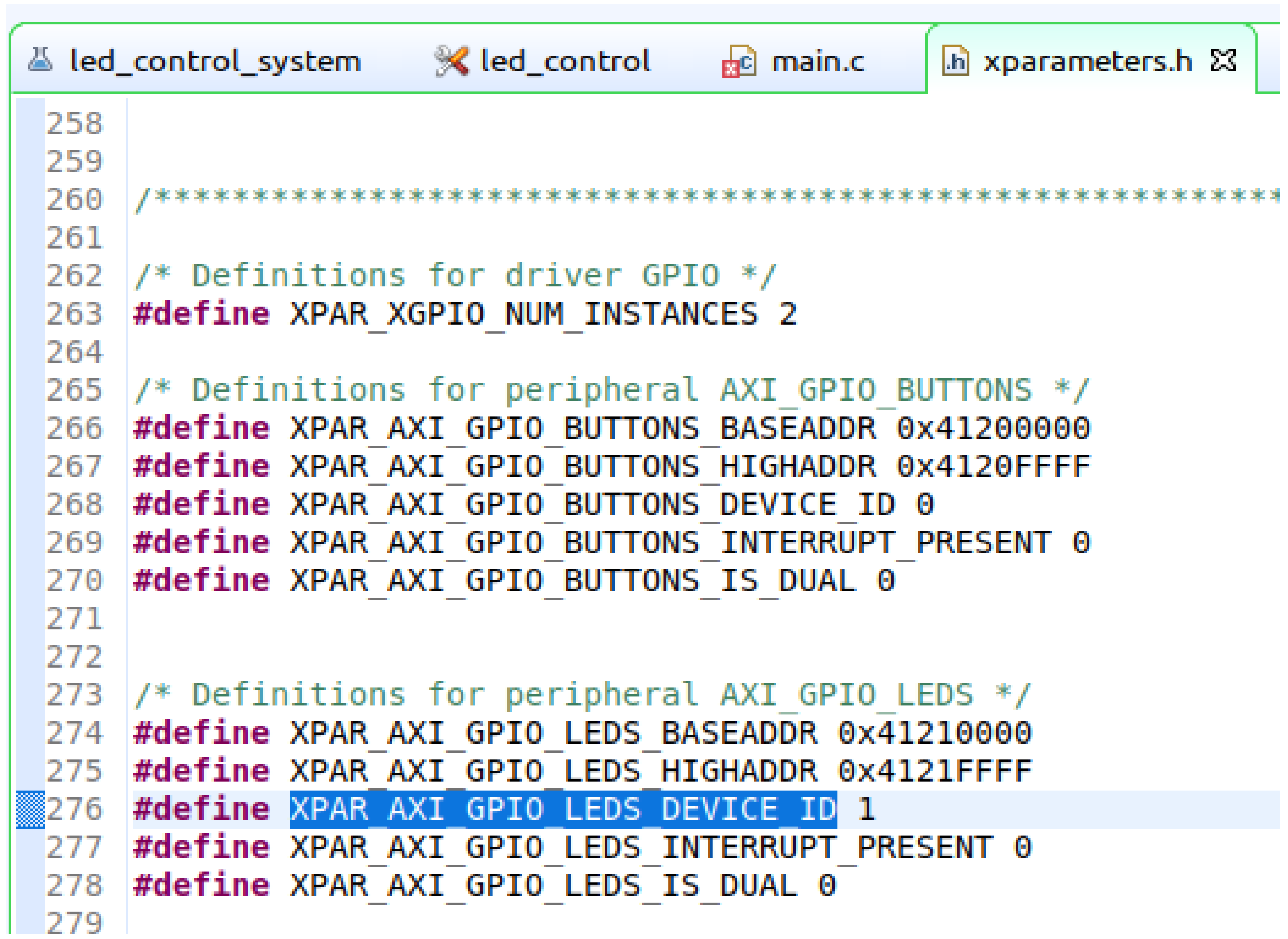
Знакомство с Vitis



Explorer

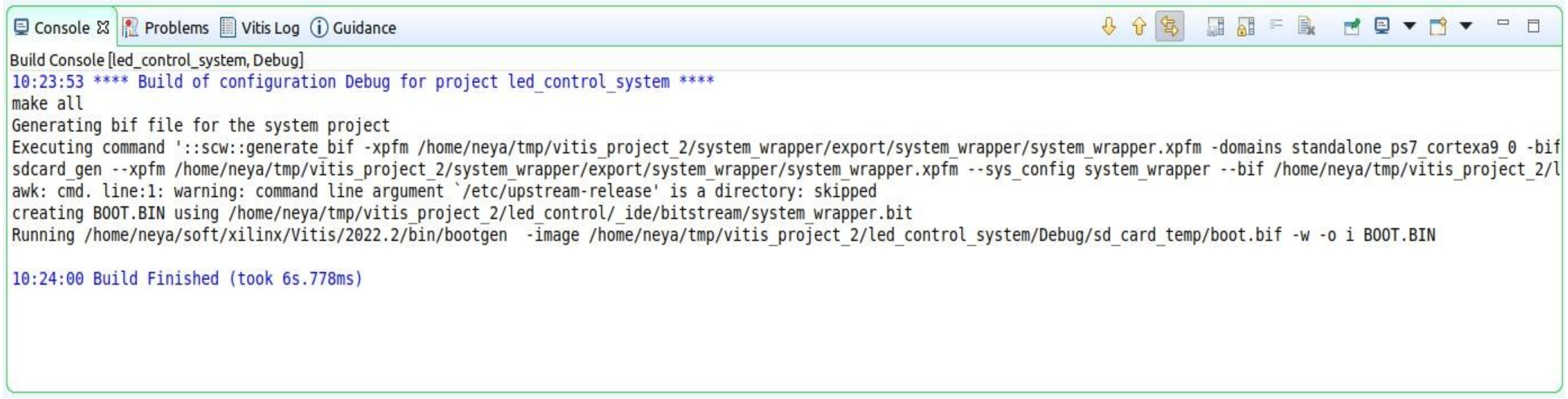
```
      main() : int
    lscript.ld
    README.txt
    Xilinx.spec
  _ide
  led_control.prj
 led_control_system.sprj
system_wrapper
 export
  system_wrapper
   hw
   sw
    system_wrapper
     boot
     qemu
     standalone_ps7_cortexa9_0
      bspinclude
       include
        _profile_timer_hw.h
        bspconfig.h
        mblaze_nt_types.h
        profile.h
        sleep.h
        smc.h
        vectors.h
        xadcps_hw.h
        xadcps.h
        xbasic_types.h
        xcoresightpsdcc.h
        xcpu_cortexa9.h
```

Знакомство с Vitis



```


/************************************************************

/* Definitions for driver GPIO */
#define XPAR_XGPIO_NUM_INSTANCES 2

/* Definitions for peripheral AXI_GPIO_BUTTONS */
#define XPAR_AXI_GPIO_BUTTONS_BASEADDR 0x41200000
#define XPAR_AXI_GPIO_BUTTONS_HIGHADDR 0x4120FFFF
#define XPAR_AXI_GPIO_BUTTONS_DEVICE_ID 0
#define XPAR_AXI_GPIO_BUTTONS_INTERRUPT_PRESENT 0
#define XPAR_AXI_GPIO_BUTTONS_IS_DUAL 0


/* Definitions for peripheral AXI_GPIO_LEDS */
#define XPAR_AXI_GPIO_LEDS_BASEADDR 0x41210000
#define XPAR_AXI_GPIO_LEDS_HIGHADDR 0x4121FFFF
#define XPAR_AXI_GPIO_LEDS_DEVICE_ID 1
#define XPAR_AXI_GPIO_LEDS_INTERRUPT_PRESENT 0
#define XPAR_AXI_GPIO_LEDS_IS_DUAL 0

```

Знакомство с Vitis


Console | Problems | Vitis Log | Guidance

Build Console [led_control_system, Debug]

```
10:23:53 **** Build of configuration Debug for project led_control_system ****
make all
Generating bif file for the system project
Executing command '::scw::generate_bif -xpfm /home/neya/tmp/vitis_project_2/system_wrapper/export/system_wrapper/system_wrapper.xpfm -domains standalone_ps7_cortexa9_0 -bif
sdcard_gen --xpfm /home/neya/tmp/vitis_project_2/system_wrapper/export/system_wrapper/system_wrapper.xpfm --sys_config system_wrapper --bif /home/neya/tmp/vitis_project_2/l
awk: cmd. line:1: warning: command line argument `/etc/upstream-release' is a directory: skipped
creating BOOT.BIN using /home/neya/tmp/vitis_project_2/led_control/_ide/bitstream/system_wrapper.bit
Running /home/neya/soft/xilinx/Vitis/2022.2/bin/bootgen  -image /home/neya/tmp/vitis_project_2/led_control_system/Debug/sd_card_temp/boot.bif -w -o i BOOT.BIN

10:24:00 Build Finished (took 6s.778ms)
```

Знакомство с Vitis

Спасибо за внимание!