

SDK Projects

Table of contents

- 1 [Table of contents](#)
 - 2 [Create SDK Project from Vivado](#)
 - 2.1 [Variant \(a\): Use additional TE Script functions](#)
 - 2.2 [Variant \(b\): Use Xilinx GUI Export](#)
 - 3 [Create Software Application with SDK Template](#)
 - 4 [Modify BSP-Settings](#)
 - 5 [Debug Software Application](#)
 - 6 [Examples](#)
 - 6.1 [Xilinx "Hello World" on ZynqMP](#)
 - 6.2 [Xilinx "Hello World" on Zynq](#)
 - 6.3 [Xilinx "Hello World" on MicroBlaze](#)
 - 7 [Convert Application ELF to SREC](#)
 - 8 [References](#)
-

Create SDK Project from Vivado

Variant (a): Use additional TE Script functions

1. Requirements (for HDF-Export with Bitfile):
 - a. Project must be started with TE-Batch file or TE Scripts must be loaded. See [Initialise TE-scripts on Vivado/LabTools](#).
 - b. "Generate Bitstream" must be finished (Use GUI or typ on Vivado TCL-Console to generate Bitstream:"TE::hw_build_design").
2. Typ "TE::sw_run_sdk" on Vivado TCL-Console (for more options type "TE::sw_run_sdk -help")
3. SDK Project will be generated and opened on subfolder /workspace/sdk/. Local Library folder /sw_lib will be included automatically.

Variant (b): Use Xilinx GUI Export

1. Requirements (for HDF-Export with Bitfile):
 - a. "Generate Bitstream" must be finished.
2. For HDF export click: FileExportExport Hardware (Select "Include bitstream" and your preferred folder)
3. To Launch SDK click: FileLaunch SDK (Select your HDF location and preferred SDK workspace)
4. SDK Project will be generated and opened (default location /vivado/<project name>.sdk).
5. (optional) Include local library folder in SDK, click: Xilinx ToolRepositoriesNew Local Repositories
 - a. for TE-SW-templates use the path: <reference design>/sw_lib

Create Software Application with SDK Template

1. Click:FileNew Application Project

2. Select Name and OS (Example Name:Hello World, OS:standalone)

New Project

Application Project
Create a managed make application project.

Project name:

☒ Use default location
Location:
Choose file system:

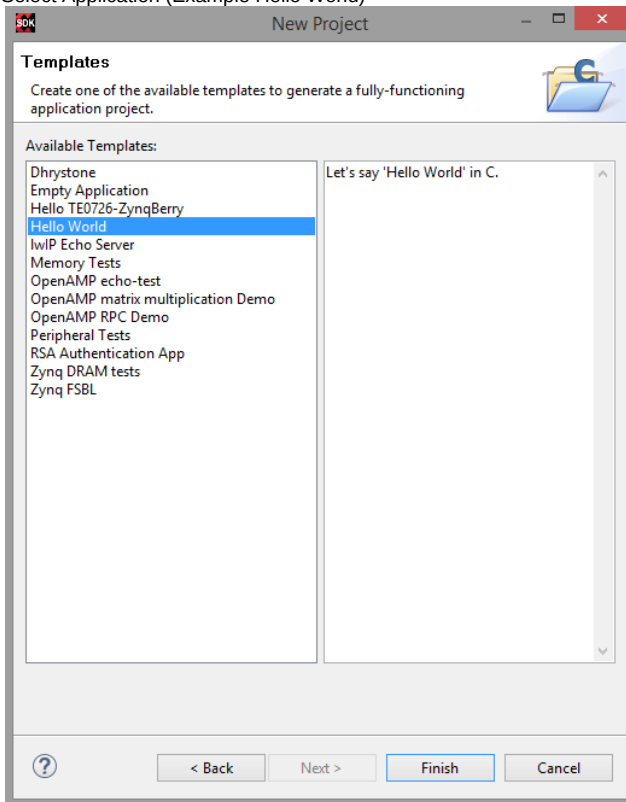
OS Platform:

Target Hardware
Hardware Platform:
Processor:

Target Software
Language: ☒ C ☐ C++
Compiler:
Board Support Package: ☒ Create New
☐ Use existing

3. Click: Next

4. Select Application (Example Hello World)

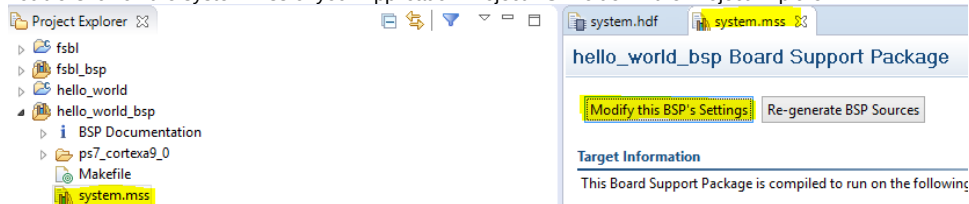


5. Click Finished

6. Project with Xilinx Hello Word will be generated automatically

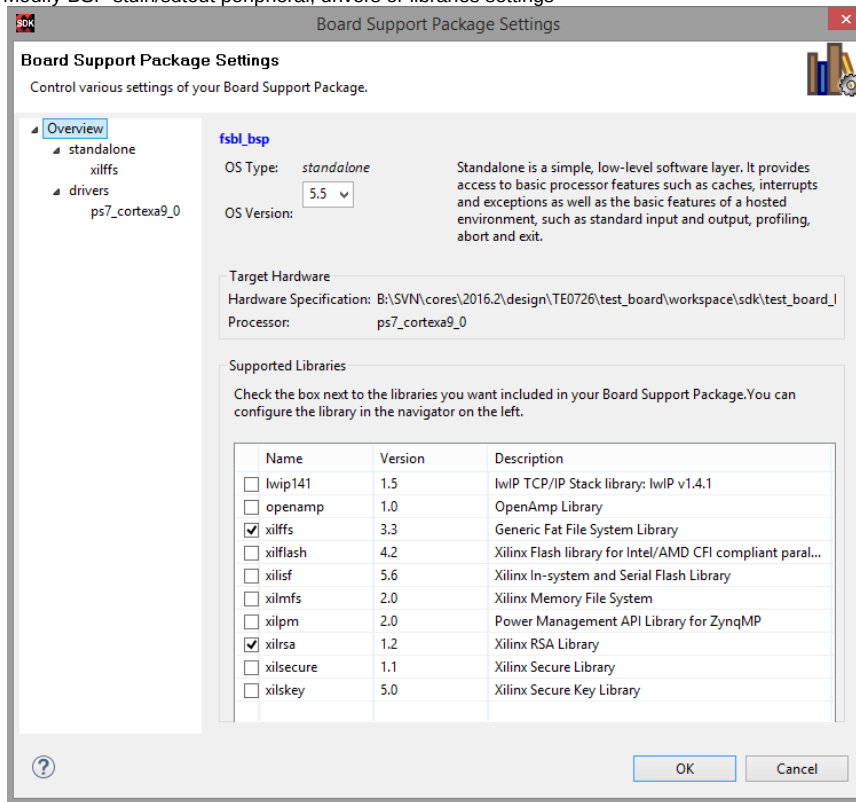
Modify BSP-Settings

1. Double Click on the system.mss of your Application Project BSP folder in the Project Explorer.



2. Click Modify this BSP's Settings.

3. Modify BSP stdin/stdout peripheral, drivers or libraries settings



Debug Software Application

1. Right Click on the Software Project in the Project Explorer.
2. Click: Debug As Launch on Hardware (System Debugger)
3. View will be changed to Debugging and Application will be started on Hardware with break point on first main entry. (Click F6 (step over) or F5 (step into) to step to the next line).

Examples

 This examples shows only the basic steps without TE-Module specific settings

Xilinx "Hello World" on ZynqMP

Same procedure as on Zynq device.

Xilinx "Hello World" on Zynq



This Hello World example is not usable for TE0722 without DDR

1. Create SDK Project:
 - a. See [Create SDK Project from Vivado](#)
2. Create FSBL:
 - a. See [Create Software Application with SDK Template](#)

New Project

Application Project
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

Target Hardware

Hardware Platform:

Processor:

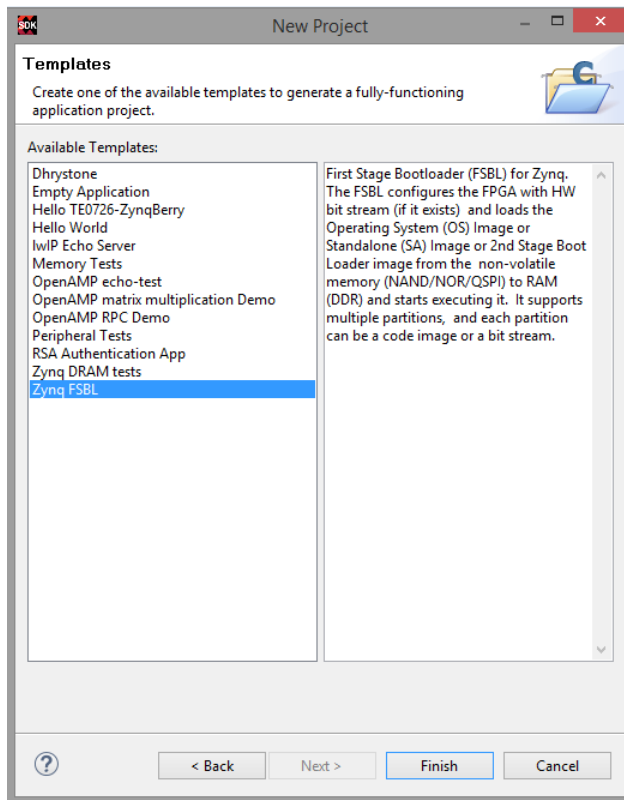
Target Software

Language: ☒ C ☐ C++

Compiler:

Board Support Package: ☒ Create New ☐ Use existing

b.



- c. (optional) If necessary, modify FSBL-BSP stdin/stdout peripheral, drivers or libraries settings for the FSBL Application
 - i. See [Modify BSP-Settings](#)
- 3. Create Hello_World:
 - a. See [Create Software Application with SDK Template](#)

New Project

Application Project
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

Target Hardware

Hardware Platform:

Processor:

Target Software

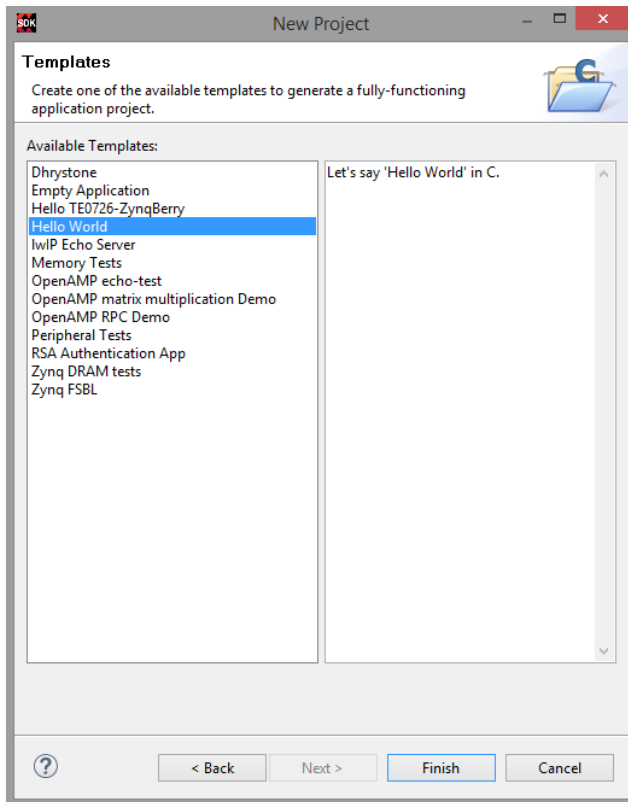
Language: ☒ C ☐ C++

Compiler:

Board Support Package: ☒ Create New

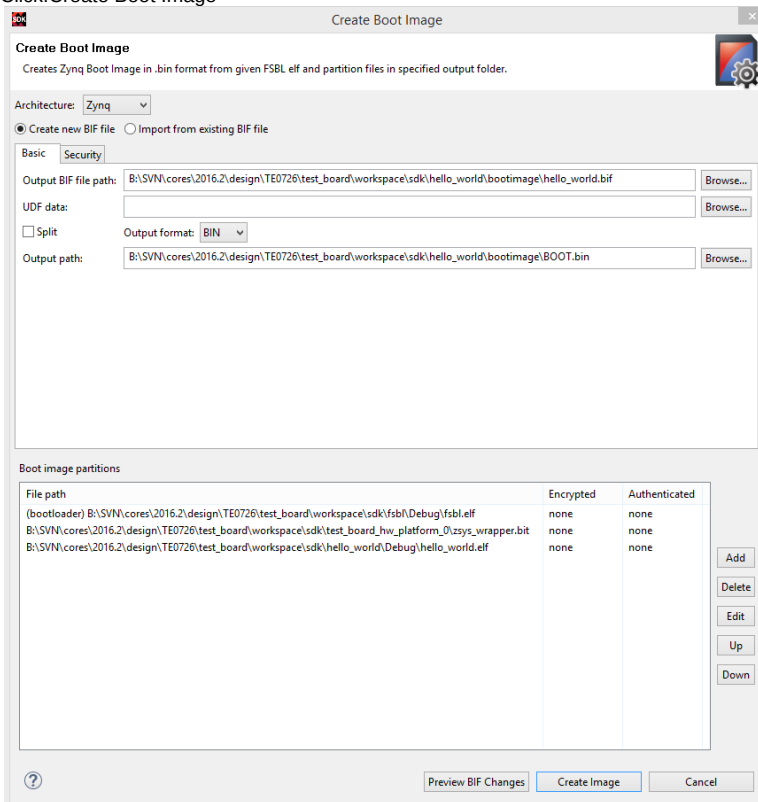
☐ Use existing

b.



- c. (optional) If necessary, modify FSBL-BSP stdin/sdtout peripheral, drivers or libraries settings for the Hello World Application
 - i. See [Modify BSP-Settings](#)
- 4. Create Boot.bin
 - a. Right Click on the hello_world Project in the Project Explorer.

b. Click: Create Boot Image



All partition settings will be set automatically, if Build Process of FSBL and Hello World was successful.

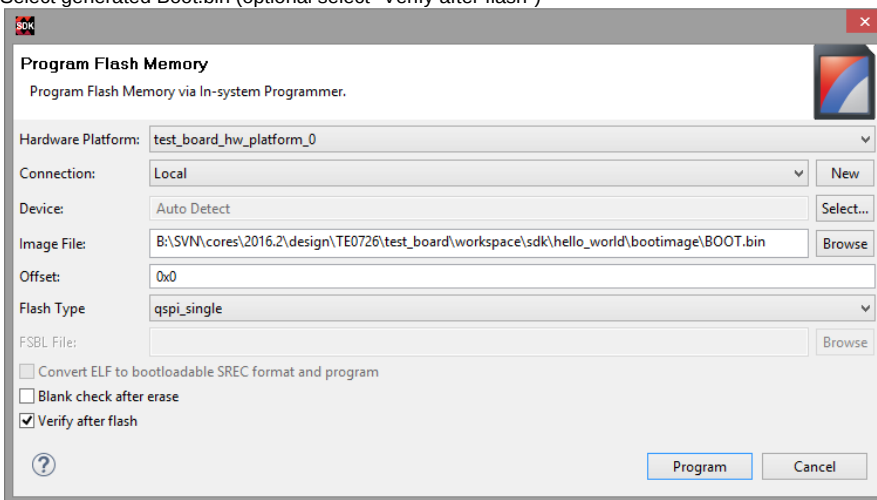
c. Create Image

5. Program Flash

a. Click: Xilinx Tools Program Flash

b. **Important since Vivado 2017.4:** FSBL is needed on setup for QSPI programming. Reference Designs include special FSBL to program QSPI Flash without changing boot mode to JTAG

c. Select generated Boot.bin (optional select "Verify after flash")



Note: Flash Type depends on HW: Select "qspi_single" or "qspi_dual_parallel", see <Reference Design>/board_files/*_board_files.csv

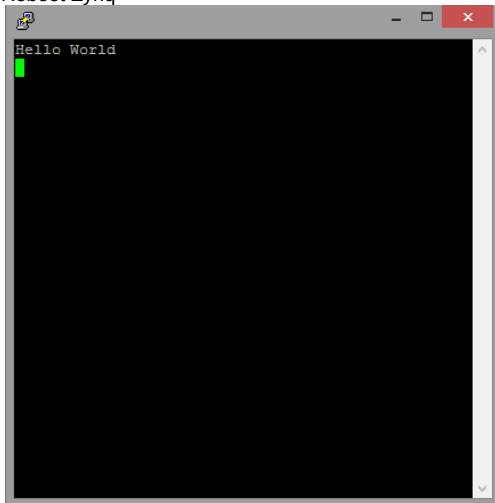
d. Program Flash

6. Connect Serial Console

a. COM Port: See OS Device Manager

b. Speed: depends on your Vivado Project. On Zynq Devices Default 115200

7. Reboot Zynq



Xilinx Hello World appears only one time on startup, so use HW-Reset Button on Module or Vivado Hardware Manager "Boot from Configuration Memory Device" Command to reboot PS. Alternatively modify helloworld.c to run print "Hello World" in endless loop.

Xilinx "Hello World" on MicroBlaze

1. Create SDK Project:
 - a. See [Create SDK Project from Vivado](#)
2. Create Hello_World:
 - a. See [Create Software Application with SDK Template](#)

New Project

Application Project
Create a managed make application project.

Project name:

☒ Use default location

Location:

Choose file system:

OS Platform:

Target Hardware

Hardware Platform:

Processor:

Target Software

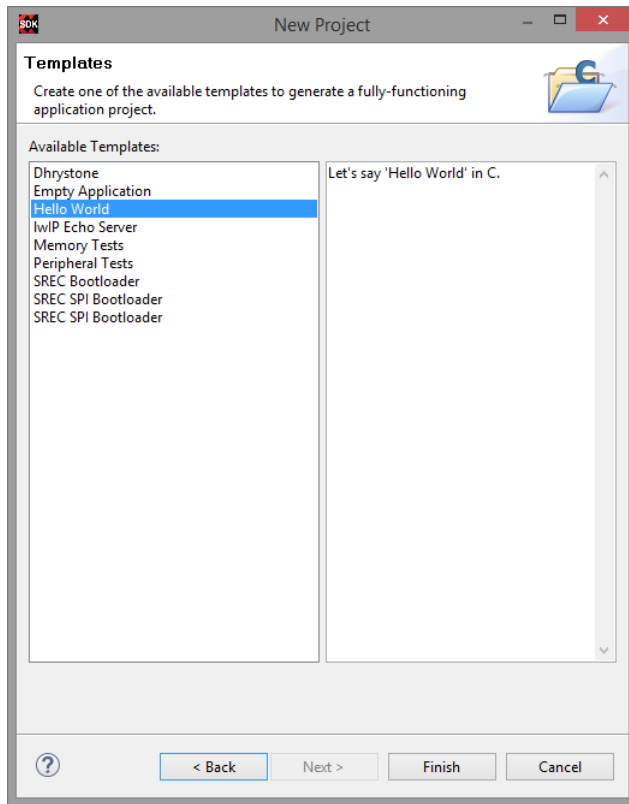
Language: ☒ C ☐ C++

Compiler:

Board Support Package: ☒ Create New

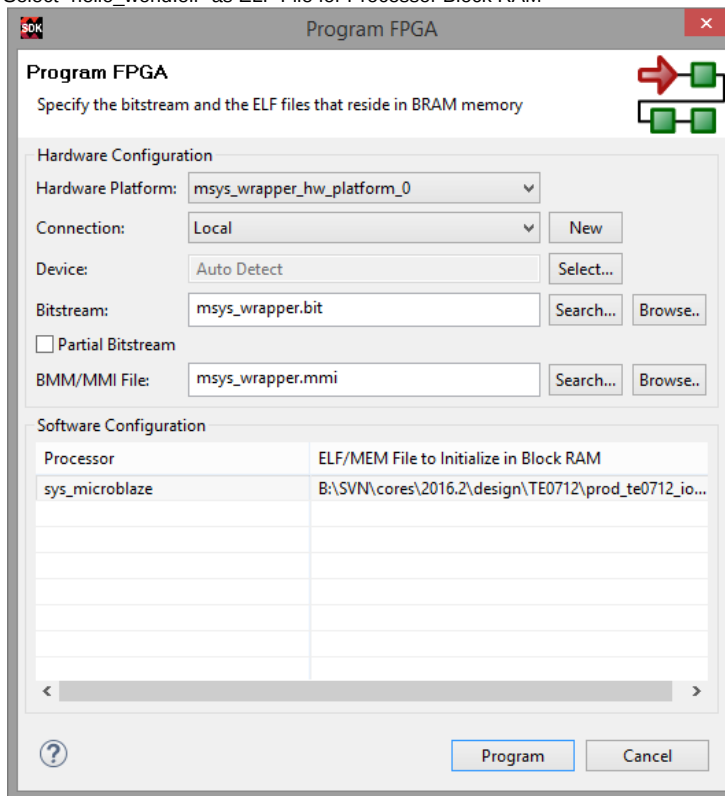
☐ Use existing

b.



- c. (optional) If necessary, modify "Hello World"-BSP stdin/stdout peripheral, drivers or libraries settings for the Hello World Application
 - i. See [Modify BSP-Settings](#)
3. Program FPGA:
 - a. Click: Xilinx Tools Program FPGA

- b. Select "hello_world.elf" as ELF File for Processor Block RAM



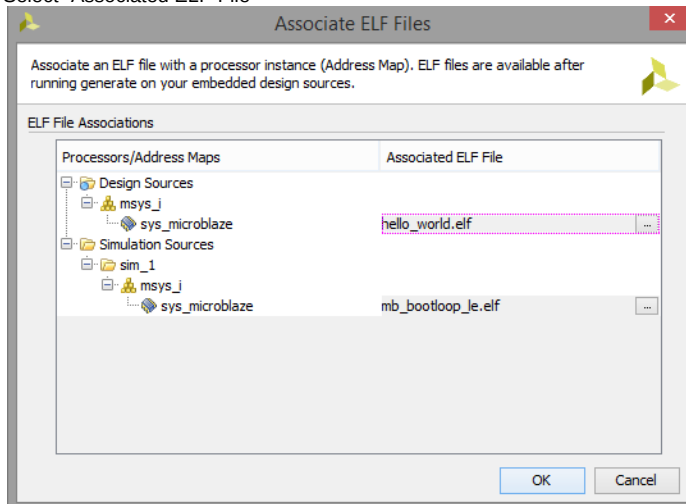
- c. Click: Program



This step configure the Bitfile with the specified ELF-files and program the FPGA

4. (Optional) include ELF-file in Vivado Project:
- a. Right Click on the Block Diagram on Vivado Project Manager source window

- b. Select "Associated ELF-File"



- c. Generate Bitfile
d. Program FPGA with Vivado HW-Manager

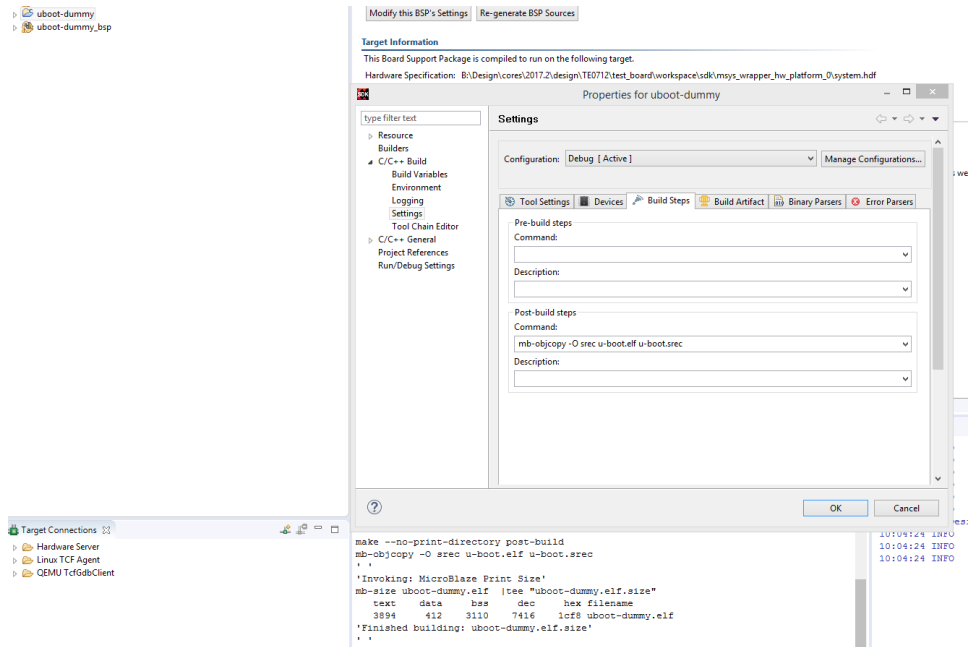


Xilinx Hello World appears only one time on startup, so use HW-Reset Button on Module or Vivado Hardware Manager "Boot from Configuration Memory Device" Command to reboot PS. Alternatively modify helloworld.c to run print "Hello World" in endless loop.

Convert Application ELF to SREC

1. Create Application (for example Hello World)
2. Open "Application" properties C/C++ Build Settings and go into Build Steps Tab.

3. Add to Post-build steps: `mb-objcopy -O srec <applicationname>.elf <applicationname>.srec`



4. Press Apply or regenerate project

References

- UltraFast Design Methodology Guide for the Vivado Design Suite (UG949)
- UltraFast Embedded Design Methodology Guide (UG1046)
- Zynq UltraScale+MPSoC Software Developer Guide (UG1137)
- Zynq-7000 All Programmable SoC Software Developers Guide (UG821)
- Vivado Design Suite User Guide - Embedded Processor Hardware Design (UG898)
- Generating Basic Software Platforms - Reference Guide (UG113)
- [PetaLinux KICKStart](#)