Note On Update (May 2018)

The original tutorial of the software derivation of the text is based on Xilinx Vivado/SDK version 2016. The procedure to embed the elf file to MicroBlaze MCS configuration bit stream is cumbersome and slow. The newer versions streamline the process. Subsections A.5.3 to A.5.6 are updated to demonstrate the features of the new version.

A.5 SHORT TUTORIAL ON FPRO SYSTEM DEVELOPMENT

The FPro system development involves the derivation of hardware and software. The procedure consists of following steps:

1. Create a design project.
2. Add or create a MicroBlaze MCS instance.
3. Add or create HDL codes with an MCS instance.
4. Add a constraint file.
5. Perform synthesis, implementation, and bitstream generation.
6. Export hardware configuration.
7. Derive software and generate the executable file (elf file).
8. Set up a terminal emulator program.
9. Embed the elf file into FPGA's memory module, regenerate bitstream, and Program an FPGA device.

The procedure expands the previous hardware development procedure in Section A.2 and incorporates three additional steps (Steps 6, 7, and 8) to accommodate the software development. The tutorials of the three steps are provided in the following subsections.

Note that Vivado Design Suite can serve as a platform for SoC development. The IP Integrator process of Flow Navigator is for this purpose. However, the platform is intended for full-featured MicroBlaze and AXI-based IP cores. Support for MicroBlaze MCS is limited and its development does not follow Vivado's general IP-based flow. The FPro system in the book is constructed from scratch and does not use any Vivado's built-in IP integration facilities.

A.5.1 Derive FPro system hardware

We use the vanilla FPro system discussed in Section 10.7 for the tutorial. The `cpu.xci` IP file and HDL files can be found in the companion website. The FPro system hardware can be derived as follows:

1. Create a design project: same as in Section A.2.1.
2. Add a MicroBlaze MCS instance: add `cpu.xci` to project. The `.xci` may need to be updated for the new Vivado version. Follow the procedure in Section A.4.3 to recreate the instance if needed.
3. Add HDL codes: add the following HDL files:
   - Top-level design and slot definition: `mcs_top_vanilla` and `chu_io_map`
   - MMIO subsystem and bridge: `mmio_sys_vanilla`, `chu_mmio_controller`, and `chu_mcs_bridge`
   - GPI, GPO, and timer MMIO cores: `chu_gpi`, `chu_gpo`, and `chu_timer`

- UART core: `chu_uart`, `uart`, `uart_rx`, `uart_tx`, `baud_gen`, `fifo`, `fifo_ctrl`, and `reg_file`

4. Add a constraint file: same as in Section A.2.4.
5. Perform synthesis, implementation, and bitstream generation: same as in Section A.2.5.

A.5.2 Export hardware configuration

Many features of MicroBlaze MCS core can be customized and thus the configuration of each instance can be different. This configuration information can be obtained and encapsulated in a *hardware description file*. In Vivado version 2016, the file can be obtained as follows:

1. Select the File ≻ Export ≻ Export Hardware... menu and a subwindow appears.
2. In the Export to field, navigate to destination folder and then click the Ok button.
3. The hardware description file (with the extension of .hdf) is generated in the designated folder.

Since an FPro system is designed manually from scratch and not from IP Integrator, the hardware description file only contains the information about the MicroBlaze MCS configuration, not the entire FPro system. Thus, this step does not need to be repeated if the same MicroBlaze MCS instance is used.

A.5.3 Derive software

We use Xilinx SDK (software development kit) as the platform for software development. It is based on Eclipse IDE (integrated development environment) with a custom Xilinx plug-in module. A typical Xilinx SDK window is shown in Figure A.13. The basic software model constitutes a three-layer hierarchy:

- Hardware platform specifications
- BSP (board support package)
- Application program

An embedded system is built around a specific application and its configuration is tailored to support the application. *Hardware platform specification* is the bottom layer that captures the relevant hardware information required for software development and deployment. *BSP* is the middle layer. It is a software library that contains drivers and start-up routines based on the information from a specific hardware platform specification file. The term is borrowed from traditional embedded system development, in which the system is usually realized by a custom printed circuit board. An *application program* is the top-layer that uses the routines in the BSP library to access hardware.

This software model is automated for a system derived from Vivado's IP Integrator. Since the FPro system is designed from scratch, the first two layers are just for MicroBlaze MCS and the BSP only contains a start-up routine. We need to manually include the software I/O drivers in an application program.

The software development constructs the three layers in sequence. The basic steps are the following:

1. Select Xilinx SDK from the Windows start menu or click on the SDK icon. Do not launch it from Vivado.

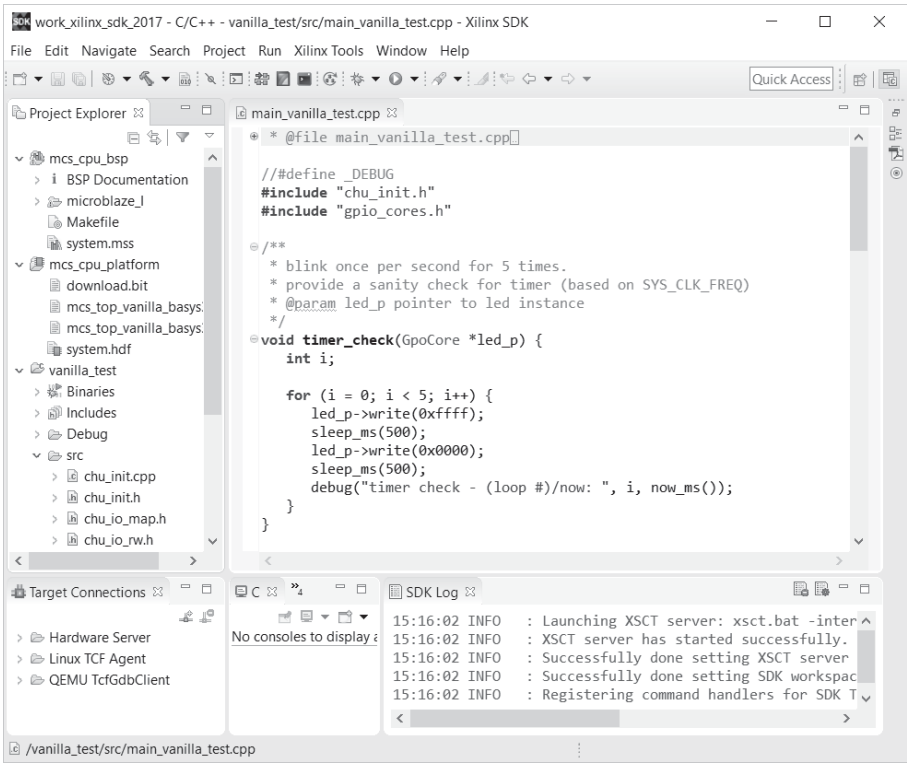

Figure A.13 Xilinx SDK.

2. Create or select a workspace.
3. Select File ≻ New ≻ Others and a window appears. Expand the Xilinx folder and then select Hardware Platform Specification. The New Hardware Project dialog appears.
   (a) Enter a name, say mcs_cpu_platform, for the hardware project.
   (b) Click on Bitstream and BMM/MMI Files to expand the panel, as shown in Figure A.14.
   (c) In Bitstream field, navigate to the Vivado implementation folder (normally proj_name.runs/impl_1) and select the .bit file.
   (d) In BMM/MMI field, navigate to the Vivado implementation folder and select the .mmi file.
   (e) In Target Hardware Specification field, navigate to the destination folder specified in Section A.5.2 and select the .hdf file. Click Finish to generate the hardware platform specifications. The new hardware platform folder appears on the Project Explorer subwindow.
4. Select File ≻ New ≻ Board Support Package and a dialog appears. Enter a name, say mcs_cpu_bsp, for the BSP project, select the previously created mcs_cpu_platform in Target Hardware, and then select standalone for the OS

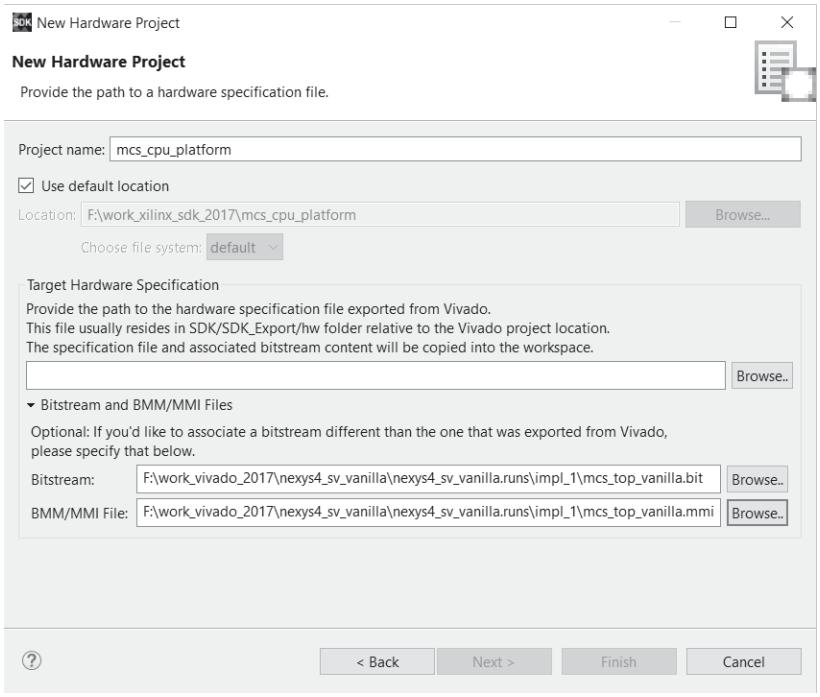

Figure A.14 Hardware project dialog.

field, as shown in Figure A.15. Click finish to generate BSP. The new BSP folder appears the Project Explorer subwindow.

5. Select File ≻ New ≻ Application Project and a dialog appears. Enter a name, say vanilla_test, for the application project, select the previously created mcs_cpu_platform and mcs_cpu_bsp, and then click on C++ button, as shown in Figure A.16. Click Finish to create a new application project. The new application project folder appears in the Project Explorer subwindow.
6. Import the main program file and driver files discussed in Chapter 9. An easy way to do this is to open Windows File Explorer and drag these files into the src folder. The Project Explorer subwindow of the completed project is shown in Figure A.17.
7. By default, Xilinx SDK is configured to build a project automatically and thus the vanilla_test.elf file is compiled and linked automatically after the files are dragged into the project. Note that the file size is displayed in the console tab, shown in Figure A.18.

A.5.4 Set up a terminal emulator program

To display the UART output character stream, a terminal emulator program is needed. We use a program, PuTTY, for this purpose. PuTTY is a telnet client and can be downloaded for free. The procedure to set up PuTTY is as follows:

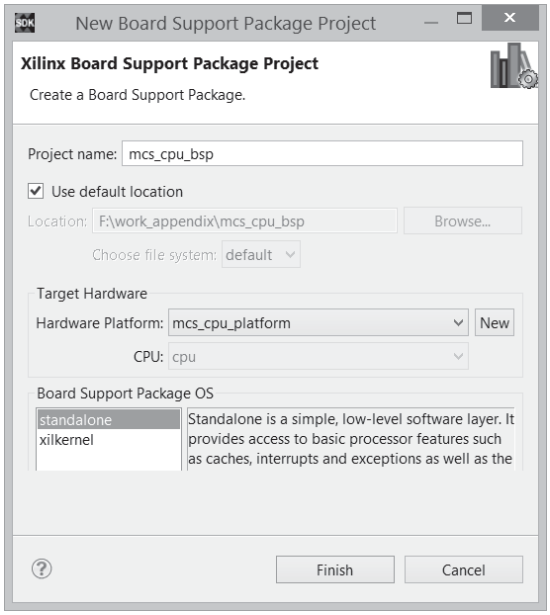

Figure A.15 BSP project dialog.

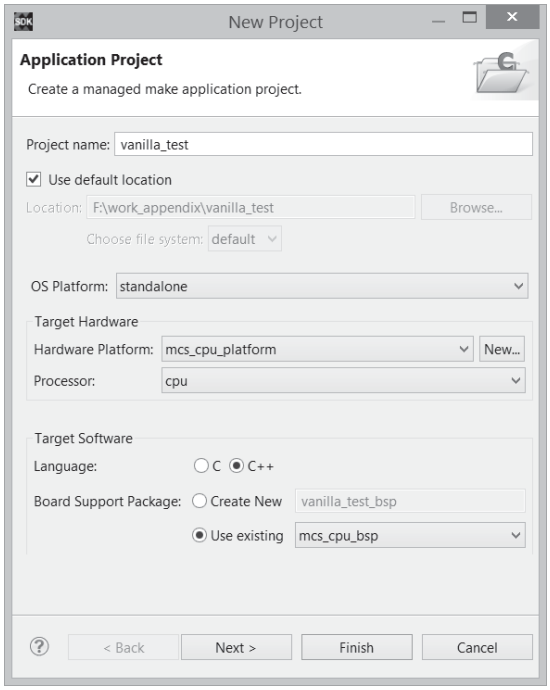

Figure A.16 Application project dialog.

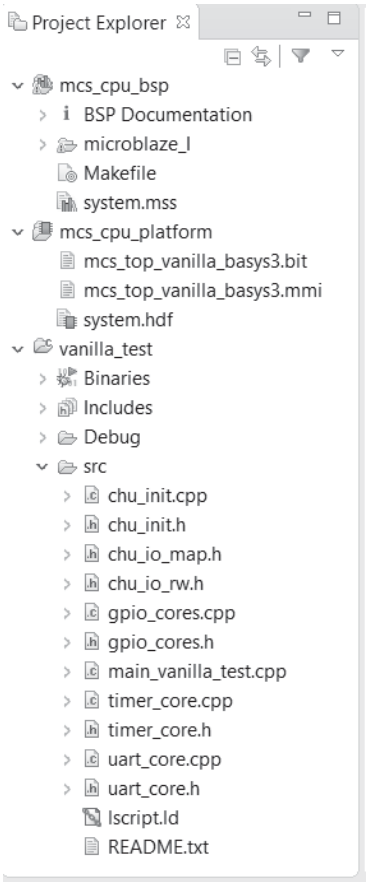

Figure A.17 Project Explorer subwindow.

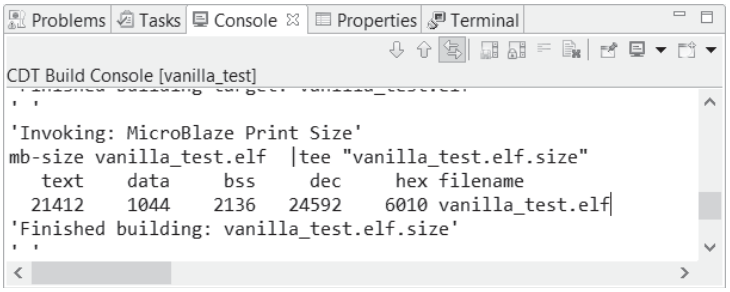

Figure A.18 File size information.

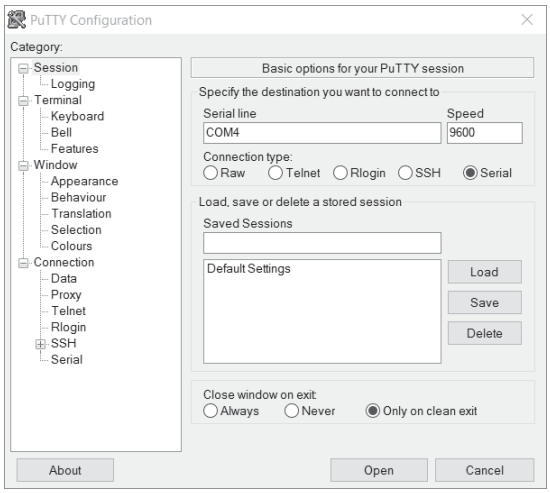

Figure A.19 PuTTY screen.

1. Connect the Nexys 4 DDR board to the PC's USB port and turn on the power of the board. The PC should recognize the FT2232 device of the board and treat the connection as a serial port.
2. In Windows, open Control Panel, select Device Manager, and expand Ports (COM & LPT). The board should be listed as one of the serial ports, labeled as USB Serial Port (COMn), where n is the designated serial port (i.e., COM port) number. Record this number.
3. Invoke the PuTTY program. In its application window, select the Serial button for serial port and enter the previous recorded COMn in the Serial line field. Make sure that the 9600 baud rate is specified in the Speed field. The completed configuration screen is shown in Figure A.19.
4. Click the Open button and the terminal window appears.

A.5.5 Embed elf file, regenerate bitstream, and program an FPGA device

After the elf file is generated, it can be used as the *initial values* of FPGA's BRAMs and embedded into the module definitions. The bitstream needs to to regenerated and the new bit file can be downloaded to an FPGA device. In version 2016, this task must be performed within Vivado Suite. In version 2017 or later, it can also be invoked directly from Xilinx SDK.

Vivado based process The steps are as follows:

1. In Vivado Design Suite, select the Tools ≻ Associate ELF Files... menu and a dialog appears.
2. Navigate to thefolder and select vanilla_test.elf and then click OK. The file will be shown under the ELF subfolder of the Design Sources folder in the Sources subwindow.
3. Follow the procedure in Section A.2 to regenerate the bit file.

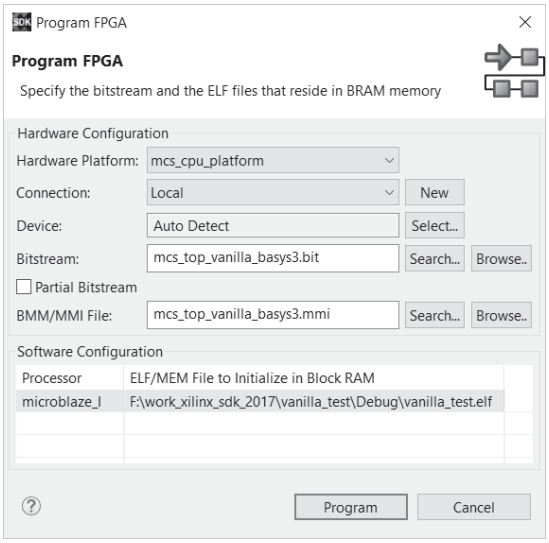

Figure A.20 Program FPGA dialog

4. Follow the procedure in Section A.2 to download the bit file to an FPGA device.

The program starts upon completion and the UART character data stream from the FPGA board will be displayed in PuTTY window.

SDK based process The steps are as follows:

1. In Xilinx SDK, select the Xilinx Tools ≻ Program FPGA menu and a dialog appears, as shown in Figure A.20. Verify that the desired hardware platform, bit stream file, and mmi file are correct.
2. In the Software Configuration subpanel, click the microblaze_1 row.
3. Click the Program button to regenerate the bit file and download it an FPGA device.

The application program starts upon completion and the UART character data stream from the FPGA board will be displayed in PuTTY window.

Comparison The SDK based process separates the software development from Vivado Suite and does not reinvoke the time-consuming implementation (i.e., placement-and-routing) process. It is the preferred method.

A.5.6 Subsequent software application development

A complete FPro hardware implementation can be encompassed in three files:

1. .hdf (hardware description) file
2. .mmi (memory map info) file
3. .bit file

The software development can be done with these files in Xilinx SDK alone. As long as the hardware portion remains the same, Vivado Suite does not needed to be invoked.