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

- 38 TUTORIALS
  - 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  $\succ$  Export  $\succ$  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.

### SHORT TUTORIAL ON FPRO SYSTEM DEVELOPMENT 39



Figure A.13 Xilinx SDK.

- 2. Create or select a workspace.
- 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

### 40 TUTORIALS

Project					$\times$
Project					
o a hardware specification fil	e.				
s_cpu_platform					
ation					
_xilinx_sdk_2017\mcs_cpu_pla	atform			Brow	se
ile system: default 🖂					
Specification					
to the hardware specificatior	n file exported from	n Vivado.			
sides in SDK/SDK_Export/hw	/ folder relative to	the Vivado project	location.		
The and associated bitstream	r content win be co	pied into the work	space.		
				Bi	owse
3MM/MMI Files					
I like to associate a bitstrean at below.	n different than the	one that was expo	rted from Vivado,		
F:\work_vivado_2017\nexys	4_sv_vanilla\nexys4	_sv_vanilla.runs\imp	ol_1\mcs_top_vanilla	a.bit Bro	owse
F:\work_vivado_2017\nexys	4_sv_vanilla\nexys4	_sv_vanilla.runs\imp	ol_1\mcs_top_vanilla	a.mmi Bro	owse
	Project Project a hardware specification fil s_cpu_platform ation xilinx_sdk_2017\mcs_cpu_plat le system: default Specification to the hardware specificatior sides in SDK/SDK_Export/hw file and associated bitstrean at below. F\work_vivado_2017\nexys- F\work_vivado_2017\nexys-	Project Project a hardware specification file. s_cpu_platform ation xilinx_sdk_2017\mcs_cpu_platform le system: default Specification to the hardware specification file exported from sides in SDK/SDK_Export/hw folder relative to to file and associated bitstream content will be co SMM/MMI Files I like to associate a bitstream different than the at below. F\work_vivado_2017\nexys4_sv_vanilla\nexys4 F\work_vivado_2017\nexys4_sv_vanilla\nexys4	Project Project Project a a hardware specification file.  s_cpu_platform ation xilinx_sdk_2017\mcs_cpu_platform le system: default  Specification to the hardware specification file exported from Vivado. sides in SDK/SDK_Export/hw folder relative to the Vivado project file and associated bitstream content will be copied into the work MM/MMI Files I like to associate a bitstream different than the one that was expo at below. F\work_vivado_2017\nexys4_sv_vanilla\nexys4_sv_vanilla.runs\imp	Project  project  a hardware specification file.  s_cpu_platform  ation  xilinx_sdk_2017\mcs_cpu_platform  le system: default  Specification S	Project  Pro

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:

# SHORT TUTORIAL ON FPRO SYSTEM DEVELOPMENT 41

New Board	Support Package Project	_	×
Xilinx Board Support Package Project			
Create a Board Suppo	ort Package.		
Droject name: mes.c	nu hen		
Project name:mcs_c	pu_bsp		
✓ Use default location	on		
Location: F:\work_ap	pendix\mcs_cpu_bsp	Brows	se
Choose file	system: default 🗸		
Target Hardware			
			Nava
Hardware Platform:	mcs_cpu_platform	~	New
CPU:	сри	$\sim$	
Board Support Pack	age OS		
standalone xilkernel	Standalone is a simple, low-level provides access to basic processo as caches, interrupts and exception	software I or features ons as well	ayer. It such as the
?	Finish	Cano	cel

Figure A.15 BSP project dialog.

SDK	New Pro	oject	_ 🗆 🗙
Application Proj	ect		T.G.
Create a managed	make application proje	ct.	
Project name: van	illa test		
✓ Use default loc	ation		
Location: F:\work	appendix\vanilla test		Browse
Choose f	ile system: default 🗸		
OS Platform: star	ndalone		~
Target Hardware			
Hardware Platfor	m: mcs_cpu_platform		∨ New
Processor:	cpu		~
Target Software			
Language.	0000000		
Board Support Pa	ckage: Create New	vanilla_test_bsp	
	Use existing	mcs_cpu_bsp	~
? < Back	Next >	Finish	Cancel

Figure A.16 Application project dialog.

```
42 TUTORIALS
```



Figure A.17 Project Explorer subwindow.

📱 Problems 🖉 Tasks 🖳 Console 🛛 🔲 Properties 🖉 Terminal	- [	3
우 순 🔄 🖬 🖬 = 🚉 🖻 🗨 🗨	- 🗈	•
CDT Build Console [vanilla_test]		
1 1		^
'Invoking: MicroBlaze Print Size' mb-size vanilla_test.elf  tee "vanilla_test.elf.size" text data bss dec hex filename 21412 1044 2136 24592 6010 vanilla_test.elf 'Finished building: vanilla_test.elf.size'		~
<	>	

Figure A.18 File size information.

### SHORT TUTORIAL ON FPRO SYSTEM DEVELOPMENT 43

Session	Basic options for your PuTTY session			
Logging     Terminal         Keyboard         Bell         Features         Window         Appearance         Behaviour         Translation         Section         Colours         Connection         Data         Proxy         Tenhet         Rlogin         SSH         Serial	Specify the destination you want to con Serial line COM4 Connection type: ORaw Telnet ORlogin (C Load, save or delete a stored session Saved Sessions	Speed 9600 SSH © Serial		
	Default Settings	Load Save Delete		
	Close window on exit: Always Never Only on clean exit			

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.
- 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:

- In Vivado Design Suite, select the Tools ≻ Associate ELF Files... menu and a dialog appears.
- 2. Navigate to the folder 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.

#### 44 TUTORIALS

SOK Program FPGA			×
Program FPGA			ᠿ
Specify the bitstrea	am and the ELF files that reside in BRAM m	emory	-0-0
Hardware Configur	ation		
Hardware Platform	: mcs_cpu_platform ~		
Connection:	Local ~	New	
Device:	Auto Detect	Select	
Bitstream:	mcs_top_vanilla_basys3.bit	Search	Browse
Partial Bitstream			
BMM/MMI File:	mcs_top_vanilla_basys3.mmi	Search	Browse
Software Configura	ition		
Processor E	LF/MEM File to Initialize in Block RAM		
microblaze_I F	\work_xilinx_sdk_2017\vanilla_test\Debug	vanilla_tes	t.elf
(?)	Program	Ca	incel

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., placementand-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.