

Developing Zynq Software with Xilinx SDK

Lab 1

Explore a Zynq Hardware Platform



November 2013
Version 03

Lab 1 Overview

Zynq-7000 All Programmable SoC application development starts with a Zynq Hardware Platform. This hardware platform defines how the ARM Processing System (PS) is configured as well as providing the actual hardware design itself for the Programmable Logic (PL). This hardware platform must be configured, designed, and built in Vivado. Upon completion of this build process, the results may be exported out of Vivado for use in the Xilinx Software Development Kit (SDK).

This course focuses on the software application development in SDK, so we will make use of a Zynq hardware platform that was previously designed and built in Vivado and then exported for SDK use. For additional instruction on how to design the Zynq hardware platform, please attend the *Developing Zynq-7000 AP SoC Hardware*.

Before performing software application development in SDK, it is worthwhile to understand what the hardware platform archive contains.

Lab 1 Objectives

When you have completed Lab 1, you will know:

- What files are included in a Zynq hardware platform archive
- What data is contained in each file
- How to analyze a hardware platform archive to determine what that hardware platform is

Experiment 1: Review the Hardware Platform Archive

This experiment shows where you will find the exported Zynq Hardware Platform within a Vivado project directory structure. The purpose of each file will be explained.

Experiment 1 General Instruction:

Browse the directory structure for the pre-built Zynq platform from Vivado. Review the archive that Vivado exported for SDK. Learn what the purpose of each file is.

Experiment 1 Step-by-Step Instructions:

1. Open Windows Explorer.
2. Browse to the provided, pre-built and pre-exported Zynq hardware platform within the Vivado project in the following folder:

`C:\Speedway\ZynqSW\2013_3\ZynqDesign`

3. This is the Vivado project containing the hardware platform design files. Notice the multiple directories for the hardware project, as shown below.

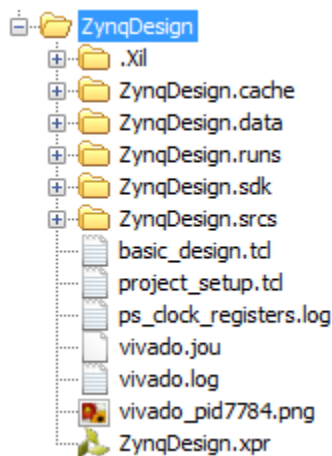


Figure 1 – Directory Structure for a Vivado Hardware Project

4. Now browse into the SDK directory.

- `ZynqDesign.sdk\SDK\SDK_Export\hw`

NOTE: If you have inherited a Vivado project from which Vivado auto-launched SDK, the SDK Workspace will be located at the SDK_Export level, as indicated by the presence of a .metadata folder there. We will be using a different, custom location for the SDK workspace.

The contents of the **hw** directory contain everything necessary to develop software applications within SDK. For hardware and software engineers working together, the contents of this directory could be the only items transferred from the hardware team to the software team. For software engineers working exclusively in SDK, they do not need anything else from the rest of the Vivado project.

5. Examine the files contained in this directory, as shown below.








Name	Date modified	Type	Size
 ps7_init.c	10/30/2013 6:30 PM	UltraEdit Document (.c)	536 KB
 ps7_init.h	10/30/2013 6:30 PM	UltraEdit Document (.h)	6 KB
 ps7_init.html	10/30/2013 6:30 PM	Firefox HTML Document	2,857 KB
 ps7_init.tcl	10/30/2013 6:30 PM	TCL File	33 KB
 ps7_summary.html	10/30/2013 6:30 PM	Firefox HTML Document	0 KB
 Z_system.xml	10/30/2013 6:30 PM	XML Document	697 KB
 Z_system_wrapper.bit	10/30/2013 6:30 PM	BIT File	3,951 KB

Figure 2 – Zynq Hardware Platform Export from Vivado

Here is a description of each file and its purpose.

ps7_init.c	C code file defining all the register settings to properly initialize the ARM processing system. Used to create the First Stage Boot Loader (FSBL) executable.
ps7_init.h	Header file used with ps7_init.c which is also used in the creation of the FSBL.
ps7_init.html	Information file that describes how the ARM processing system is configured in this hardware platform and what all the register settings are. This is the best file for a software engineer to first examine as it is the 'datasheet' for the hardware platform.
ps7_init.tcl	This is the equivalent of ps7_init.c, but in TCL format. This file performs the ARM initialization when using SDK in JTAG mode, such as when debugging.
ps7_summary.html	Legacy file that was superseded by ps7_init.html. Currently serves no purpose. Will be removed for 2013.4.
Z_system.xml (<project>.xml)	This is the hardware specification file. The name 'Z_system' is based upon what the designer chose (whereas all the ps7* files have fixed names). This is the file that SDK uses to become aware of the hardware platform features, which it in turn uses in generating a BSP.
Z_system_wrapper.bit (<project>_wrapper.bit)	This is the PL configuration bitstream. This file is used to configure the PL, which gives the PL its function and identity. 'Z_system_wrapper' is the name of the top-level HDL file that the hardware designer used.

Questions:

Answer the following questions:

- What tool suite creates the Zynq hardware platform archive?

- Which file in the hardware platform archive is most comparable to a 'datasheet' for the hardware platform?

Experiment 2: Examine the Hardware Platform

If the hardware team provides you with a Zynq hardware platform, how do you determine what hardware is available in that hardware platform? This experiment will use the files in the hardware platform archive to determine what is in the hardware definition.

Experiment 2 General Instruction:

Use the HTML and XML files to determine what is in the hardware platform.

Experiment 2 Step-by-Step Instructions:

1. Open the ps7_init.html in a browser application. This file is an HTML representation of all the ARM register settings based upon how the hardware platform was defined in the Xilinx Vivado design tools. Although low-level, it is useful to be aware of the information available here.

ZYNQ PS REGISTER SUMMARY VIEWER

This design is targeted for xc7z020board (part number: xc7z020clg484-1)

Zynq Design Summary

Device	xc7z020
SpeedGrade	xc7z020
Part	xc7z020clg484-1
Description	Zynq PS Configuration Report with register details
Vendor	Xilinx

MIO Table View

MIO Pin	Peripheral	Signal	IO Type	Speed	Pullup	Direction
MIO 0	GPIO	gpio[0]	LVC MOS 3.3V	slow	enabled	inout
MIO 1	Quad SPI Flash	qspi0_ss_b	LVC MOS 3.3V	slow	enabled	out
MIO 2	Quad SPI Flash	qspi0_io[0]	LVC MOS 3.3V	slow	disabled	inout
MIO 3	Quad SPI Flash	qspi0_io[1]	LVC MOS 3.3V	slow	disabled	inout
MIO 4	Quad SPI Flash	qspi0_io[2]	LVC MOS 3.3V	slow	disabled	inout
MIO 5	Quad SPI Flash	qspi0_io[3]	LVC MOS 3.3V	slow	disabled	inout
MIO 6	Quad SPI Flash	qspi0_sclk	LVC MOS 3.3V	slow	disabled	out
MIO 7	GPIO	gpio[7]	LVC MOS 3.3V	slow	disabled	out
MIO 8	Quad SPI Flash	qspi_fbclk	LVC MOS 3.3V	slow	disabled	out
MIO 9	GPIO	gpio[9]	LVC MOS 3.3V	slow	enabled	inout
MIO 10	GPIO	gpio[10]	LVC MOS 3.3V	slow	enabled	inout
MIO 11	GPIO	gpio[11]	LVC MOS 3.3V	slow	enabled	inout
MIO 12	GPIO	gpio[12]	LVC MOS 3.3V	slow	enabled	inout

2. The first thing to notice is the targeted Xilinx device. The example above shows the xc7z020 with a -1 speedgrade and a CLG484 package. This makes sense as this example was built for the Avnet ZedBoard, which has that same part.
3. The next table shows the MIO pin configurations. This table also provides clues regarding which PS peripherals are enabled and assigned to MIO. Examine the table and compare with the expected list of PS peripherals below. You can

expect that when you later build a BSP, the Xilinx tools will include the drivers for these peripherals.

- a. GPIO
- b. Quad SPI Flash
- c. USB
- d. Ethernet
- e. SD
- f. UART

4. The next table provides information about the DDR Memory. This is useful in telling you all the operating parameters of the DDR controller, but more basically, the total size of the RAM, which in this case is 512 MB (32-bit bus, composed of two 16-bit ICs that are each 2048 Mbits).

DDR Memory information

Parameter name	Value	Description
Enable DDR	1	Enable DDR Controller of Zynq PS
Enable DDR	1	Enable DDR Controller of Zynq PS
Memory Part	MT41J128M16 HA-15E	
DRAM bus width	32 Bit	Select the desired data width. Refer to the Technical Reference Manual(TRM) for a detailed list of supported DDR data widths
ECC	Disabled	ECC is supported only for data width of 16-bit
BURST Length (lppdr only)	8	Select the burst Length. It refers to the amount of data read/written after a read/write command is presented to the controller
Internal Vref	1	
Operating Frequency (MHz)	533.333333	Chose the clock period for the desired frequency. The allowed freq range (200 - 667 MHz) is a function of FPGA part and FPGA speed grade
HIGH temperature	Normal (0-85)	Select the operating temperature
DRAM IC bus width	16 Bits	Provide the width of the DRAM chip
DRAM Device Capacity	2048 Mbits	
Speed Bin	DDR3_1066F	Provide the Speed Bin

Figure 3 – DDR Memory Information from HTML

5. Continue to scroll through this file to examine various settings for the PLL, clock, control, and peripheral registers.

For a more detailed explanation of the registers and their purposes, please refer to the [Zynq-7000 Technical Reference Manual](#).

The .c/.h files and the TCL file are alternative representations of these exact same register settings. The TCL file is used to set up the Zynq PS during a debug session, while the .c/.h are used to generate the FSBL.

6. Next, open the `Z_system.xml` file in a text editor. This is the primary file that sets up SDK with the necessary information about the hardware platform. You

would rarely need to open this file, so we will only look at a couple useful pieces of information.

7. If you ever have a question about what version of tools were used to build the hardware, you will find this in the XML file. See Line 2 below, which shows the VIVADOVERSION is equal to 2013.3. This hardware platform was built in Vivado 2013.3.

```
2 <EDKSYSTEM EDKVERSION="1.2" TIMESTAMP="Thu Sep 26 08:42:52 2013" VIVADOVERSION="2013.3">
```

Figure 4 – Tool Version

8. The part, speedgrade, and package are also repeated in this XML file.

```
4 <SYSTEMINFO ARCH="zynq" DEVICE="7z020" PACKAGE="clg484" SPEEDGRADE="-1"/>
```

Figure 5 – Part/Speedgrade/Package Target

9. Every peripheral in the system is listed in the XML as a MODULE. You can search the text file for 'MODULE' to find all of the peripherals in the hardware platform. For example, by searching on 'MODULE' you can find one of the peripherals that is built in the programmable logic fabric of this design – the PWM_Controller with Interrupt.

```
31 <MODULE FULLNAME="/PWM_w_Int_0" HWVERSION="1.0"
```

Figure 6 – PWM Controller Module

10. If you were looking specifically for ARM PS peripherals that were enabled for this design, you could search for 'MODTYPE="ps7"' since all PS peripherals have the 'ps7' prefix on their names. In addition to the peripherals, this will also show you the Cortex A9 processors, AXI interconnect, cache, dma, snoop control unit, etc.

```
5610 <MODULE HWVERSION="1.00.a" INSTANCE="ps7_uart_1" IPTYPE="PERIPHERAL" MODCLASS="PERIPHERAL" MODTYPE="ps7_uart">
```

Figure 7 – PS UART Enabled

11. The XML also reveals memory map information. This example shows the DDR mapped from 0x00100000 to 0x1FFFFFFF.

```
8829 <MODULE HWVERSION="1.00.a" INSTANCE="ps7_ddr_0" IPTYPE="PERIPHERAL" MODCLASS="MEMORY_CNTL" MODTYPE="ps7_ddr">
8830 <PARAMETERS>
8831 <PARAMETER NAME="C_INTERCONNECT_S_AXI_MASTERS" VALUE="ps7_cortexa9_0.M_AXI_DP & ps7_cortexa9_1.M_AXI_DP"/>
8832 <PARAMETER NAME="C_S_AXI_BASEADDR" VALUE="0x00100000"/>
8833 <PARAMETER NAME="C_S_AXI_HIGHADDR" VALUE="0x1FFFFFFF"/>
```

Figure 8 – DDR Memory Map

12. The `Z_system_wrapper.bit` file is binary, although there is a readable header at the beginning of the file. This header will have a timestamp as well as identify the targeted device.

Z_system_wrapper.bit x																
	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
00000000h:	00	09	0F	F0	0F	F0	0F	F0	0F	F0	00	00	01	61	00	23
00000010h:	5A	5F	73	79	73	74	65	6D	5F	77	72	61	70	70	65	72
00000020h:	3B	55	73	65	72	49	44	3D	30	58	46	46	46	46	46	46
00000030h:	46	46	00	62	00	0C	37	7A	30	32	30	63	6C	67	34	38
00000040h:	34	00	63	00	0B	32	30	31	33	2F	31	30	2F	33	30	00
00000050h:	64	00	09	31	37	3A	32	37	3A	32	37	00	65	00	3D	BA
00000060h:	FC	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000070h:	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
00000080h:	FF	00	00	00	BB	11	22	00	44	FF	FF	FF	FF	FF	FF	FF

Figure 9 – Bit File Header

Questions:

Answer the following questions:

- Which file contains the information that SDK will later use to build a BSP?

- Which file will later be used to initialize the Zynq PS during a debug session?

This concludes Lab 1.

Revision History

Date	Version	Revision
12 Nov 13	01	Initial release
22 Nov 13	02	Revisions after pilot
01 May 14	03	ZedBoard.org Training Course Release

Answers

Experiment 1

- *What tool suite creates the Zynq hardware platform archive?*

Vivado is the currently recommended tool for creating Zynq hardware platforms, although XPS can also do it.

- *Which file in the hardware platform archive is most comparable to a 'datasheet' for the hardware platform?*

ps7_init.html

Experiment 2

- *Which file contains the information that SDK will later use to build a BSP?*

The XML file – Z_system.xml

- *Which file will later be used to initialize the Zynq PS during a debug session?*

The TCL file – ps7_init.tcl