FSBL: Introduction Demo Script

Introduction

Here you will have the opportunity to build a default configuration of a First Stage Boot Loader (FSBL) and explore the basic components of the software.

FSBL: Introduction

Action with Description	Point of Emphasis and Key Takeaway

• Launch the Vitis™ Unified IDE.
• Close the Welcome tab if it is open.
• Click **Open Workspace** and select a directory of your choice. Ensure that the directory is empty so that its contents do not interfere with the workspace.
• Select **File** > **New Component** > **Platform** to create a new platform component.
• Set the following options:
  • Platform name: **UED_plat**
  • Click **Next** and select **Hardware Design**
  • XSA file: **$TRAINING_PATH/ CustEdIP/UED_zcu104.xsa**
  • OS platform: **standalone**
  • Processor: **psu_cortexa53_0**
  • Select the **Generate boot artifacts** option
• Click **Finish**.
This will take a minute or so to complete.
	• Create a platform component containing the FSBL and PMU firmware.

Action with Description	Point of Emphasis and Key Takeaway

Explore the generated software.
• Expand **UED_plat** > **Sources** > **zynqmp_fsbl**.
• Open **xfsbl_main.c** and **xfsbl_initialization.c**.
	• The thrust of this exercise is to:
	  • Determine the reset reason; initialize the APU; determine that the WDT did not create an interrupt during the previous boot attempt.
	  • Determine the boot mode according to the boot device and prepare that device for booting.
	  • Cycle through the partitions on the boot device and copy data to the indicated device.
	  • Pass control to the application or subsequent boot loader.

• Select **xfsbl_initialization.c**. • Advance to (302) `if(FsblInstancePtr->ResetReason == XFSBL_PS_ONLY_RESET)`	• Determine the reset reason; initialize the APU; determine that the WDT did not create an interrupt during the previous boot attempt.
• Advance to (873) `FsblInstancePtr->PrimaryBootDevice = BootMode;`	• Determine the boot mode according to the boot device and prepare that device for booting.
• Select **xfsbl_main.c**. • Advance to (196) `FsblStatus = XFsbl_PartitionLoad(&FsblInstance, PartitionNum);`	• Cycle through the partitions on the boot device and copy data to the indicated device.
• Advance to (254) `XFsbl_Printf(DEBUG_INFO,"========== In Stage 4 ========== \n\r");`	• Pass control to the application or subsequent boot loader.

Action with Description	Point of Emphasis and Key Takeaway

• Open **xfsbl_debug.h**.
• Notice the debug definitions.
	• Question: What symbol must be defined to start getting debug messages?
	• Answer:
	  • DEBUG_PRINT_ALWAYS or
	  • DEBUG_GENERAL or
	  • DEBUG_INFO or
	  • DEBUG_DETAILED.
	• Question: Which one do you think provides more information?
	• Answer: DEBUG_DETAILED. Look at line 64.

• Close the Vitis Unified IDE.	• Enables a clean exit.

Summary

Here you saw how the FSBL can be built for the specified hardware, thus supporting the Zynq™ 7000 SoC's A9 processor, the Zynq UltraScale+™ MPSoC's R5 or A53 processor, or even a MicroBlaze™ processor (in a device without the PS).

The FSBL software has a number of hooks that can be enabled to change how much text is displayed during its run as well as places to insert specific user boot code.