SHAKTI BOOT SEQUENCE

User Manual

developed by: SHAKTI Development Team @ iitm '19

shakti.org.in

contact @ shakti[dot]iitm[@]gmail[dot]com

0.1 Proprietary Notice

0.2 Release Information

Version	Date	Changes
0.1	January 24, 2020	Initial Release

Table of Contents

SECTION 1

Introduction

In stand alone mode, the Arty-35t board when booted starts executing the code autonomously. The application is no longer downloaded from the PC through a debugger and executed. Instead, it is stored in the flash memory. When the system starts, the boot loader loads the application from the flash memory to the physical memory (RAM). Then the control transfers to the application residing in RAM. This mode of running the application is usually used for standalone systems.

This document describes:

(1) The overall boot process.

(2) An overview of the boot options available on the SHAKTI Arty35T.

(3) Bare-metal boot examples that can be run on the SHAKTI Arty 35T.

There are two parts to run an application in standalone mode. First, there should be a boot process, which ensures the boot application is run after reset i.e. without any external intervention. This boot code must be available in a ROM or similar structure (Boot ROM). Second, the bare metal user application should be present in a erasable / programmable memory (for example: flash memory). If the board is reset, the Boot ROM code will act like a loader which will load the bare metal user application into the RAM and start executing from it. Figure 1 describes the boot process.
In this figure the Boot ROM code (which acts as a loader) takes code from the flash and writes to RAM, and then jumps to the starting address of the user application.

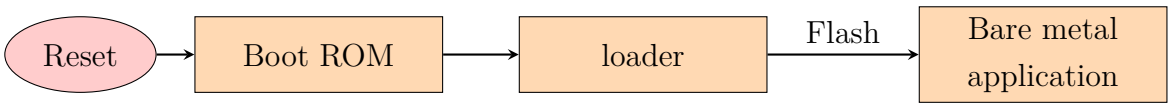

Figure 1: SHAKTI Boot flow

1.0.1 Prerequisite

In order to run the example application in standalone mode, the following are required:

- Digilent Artix 35T board with SHAKTI E class SoC programmed [1].
- Host PC running Ubuntu 16.04.[1]
- SHAKTI tool chain installed.

[1]We have tested with Ubuntu 16.04.

SECTION 2

Boot Overview

This section presents an overview of the different boot options and capabilities available for booting applications on SHAKTI for Arty35T boards.

The SHAKTI boot process starts when the processor is released from reset, and jumps to the Boot ROM address space. Typically, the main steps in boot flow are:

- Boot loader will detect the selected boot source i.e. SPI Flash.
- Perform necessary initialization for the loader to work.
- Load the bare metal application from Flash to physical memory and jump to it.

2.1 Boot Sources

The boot can happen with either one of the following modes:

- SPI
- Debugger

2.1.1 Boot in Debug mode

The system boots in Debug mode when switches sw1 and sw2 are low. This is the mode in which the Debugger controls the whole system. This method is usually used in debugging programs.

2.1.2 Boot in QSPI mode

When booting from SPI, the loader images are always located in the Boot ROM. The loader is called after the boot initialization code. The bare metal user application is stored in the flash memory, inside 64KB sectors. The loader loads the contents in flash from the location 0x000B0000 inside flash memory. The bare metal user application code should have been stored in the flash memory before you can boot from this option. Figure 2 shows the complete boot process. Initially the user has to build the baremetal application using the tool chain and get the code ELF ready. This ELF is written to the flash successfully. Then a RESET is issued to the board. On RESET the boot process takes over as mentioned earlier. If the code execution terminates (depending on user logic) RESET of the board will repeat this process.

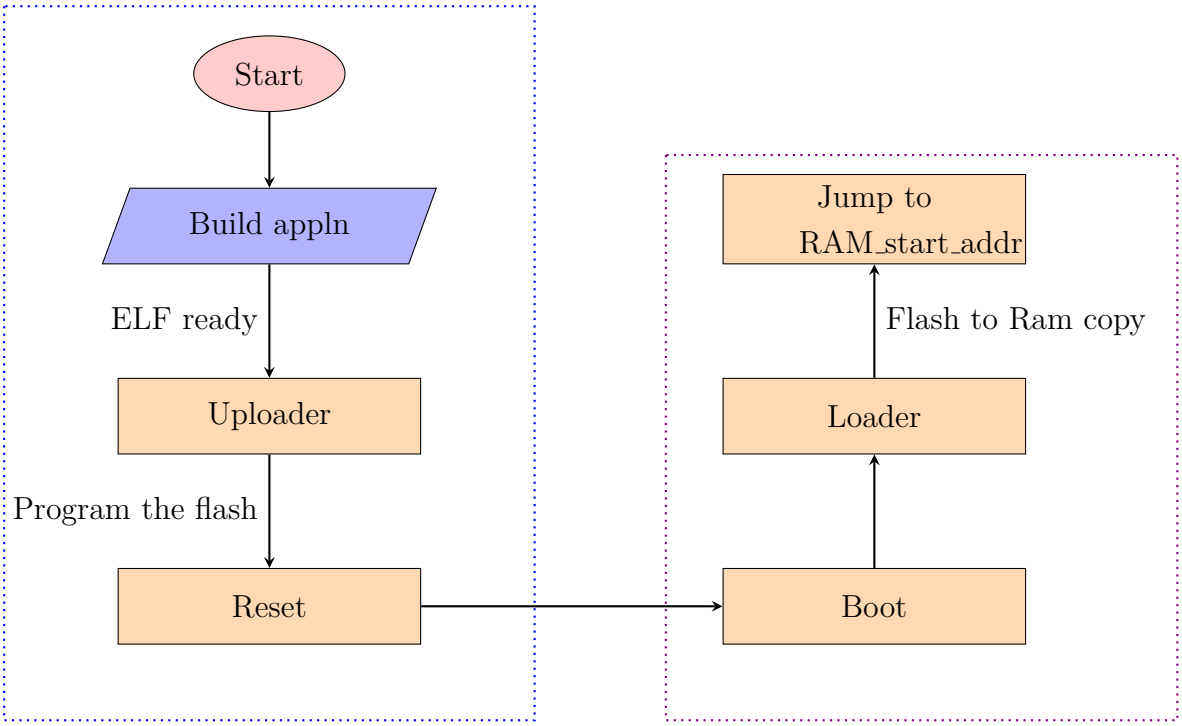

Figure 2: High level working diagram of Standalone mode

Boot Duration

In time critical applications, the duration of the boot process is very critical, and it needs to meet strict time or space constraint. We have seen that the QSPI boot takes less than 1 sec to boot for a boot code size of around 1130 X 32 bits.

We now list the steps that will generate the standalone user application.

2.2 Steps to generate standalone user application

The `make upload` command is used to build and upload the application to the flash automatically. The shakti-sdk has an *uploader* tool that is used to load a content (such as ELF) to flash, after building the image.

```
$ cd shakti-sdk
$ make upload PROGRAM= bare metal appln TARGET=artix7_35t
```

Interpreting above commands:

- PROGRAM is the new bare metal user application that is created. It is listed by typing `"make list_applns"`.
- TARGET= artix7_35t, refers to the board.

3

SECTION

Bibliography

[1] SHAKTI E-class SoC on Artix 7 35T board https://gitlab.com/shaktiproject/cores/shakti-soc/-/tree/master/fpga/boards/artya7-35t/e-class