
SHAKTI BOOT SEQUENCE

MANUAL

DEVELOPED BY: SHAKTI DEVELOPMENT TEAM @ IITM '19 -
'20

SHAKTI.ORG.IN

CONTACT @ [SHAKTI \[DOT\] IITM \[AT\] GMAIL \[DOT\] COM](mailto:SHAKTI@IITM.GMAIL.COM)

0.1 Proprietary Notice

Copyright © 2019–2020, **SHAKTI @ IIT Madras**.

All rights reserved. Information in this document is provided “as is,” with all faults.

SHAKTI @ IIT Madras expressly disclaims all warranties, representations, and conditions of any kind, whether express or implied, including, but not limited to, the implied warranties or conditions of merchant ability, fitness for a particular purpose and non-infringement.

SHAKTI @ IIT Madras does not assume any liability rising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation indirect, incidental, special, exemplary, or consequential damages.

SHAKTI @ IIT Madras reserves the right to make changes without further notice to any products herein.

The project was funded by MeITY, Government of India

0.2 Release Information

| Version | Date | Changes |
|---------|------------------|--------------------|
| 0.1 | January 24, 2020 | Initial Release |
| 0.2 | July 17, 2020 | Review corrections |
| 0.3 | October 29, 2020 | 100T support added |

Table of Contents

| | | |
|----------|---|----------|
| 0.1 | Proprietary Notice | 1 |
| 0.2 | Release Information | 2 |
| 1 | Introduction | 4 |
| 1.0.1 | Prerequisite | 5 |
| 2 | Boot Overview | 6 |
| 2.1 | Boot Sources | 6 |
| 2.1.1 | Boot in Debug mode | 6 |
| 2.1.2 | Boot in SPI mode | 7 |
| 2.2 | Steps to generate standalone user application | 7 |

Introduction

In stand alone mode, the Arty-7 board boots and starts executing the application 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 main memory. Then the control transfers to the application residing in the main memory. 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 SoC's.
- (3) Bare-metal boot examples that can be run on the SHAKTI SoC's.

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) reads the code from the flash and writes to RAM, and then jumps to the starting address of the user application.

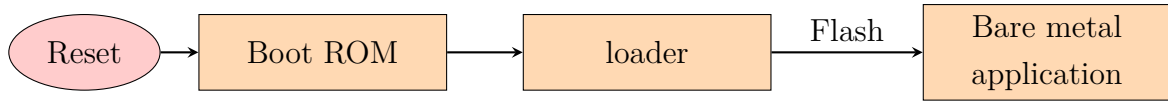


Figure 1: SHAKTI Boot flow

1.0.1 Prerequisite

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

- Digilent Artix 7 35T or 100T board with SHAKTI SoC programmed.
- Host PC running Ubuntu.¹
- SHAKTI tool chain installed.

¹We have tested with Ubuntu 16.04/18.04.

Boot Overview

This section presents an overview of the different boot options and capabilities available for booting applications on SHAKTI for Arty35T and Arty 100T 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 or Debugger.
- Perform necessary initialization for the loader to work.
- Load the bare metal application from the Flash to the physical memory and jump to the start address.

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 (0). This is the mode in which the Debugger controls the whole system. This method is usually used at application development stage or in debugging applications.

2.1.2 Boot in SPI mode

The boot loader is located in the Boot ROM. The bare metal user application is stored in the flash memory, inside 64KB sectors. The loader is called after the boot initialization code. The loader loads the contents from the flash starting at address 0x000B0000. This content is copied to the main memory. Then the control transfers to the first instruction. Figure 2 describes the complete boot process. Initially the user has to build the bare metal application and get the ELF executable ready. This ELF is written to the flash successfully, using "make upload" command. On RESET, the boot process starts as mentioned earlier.

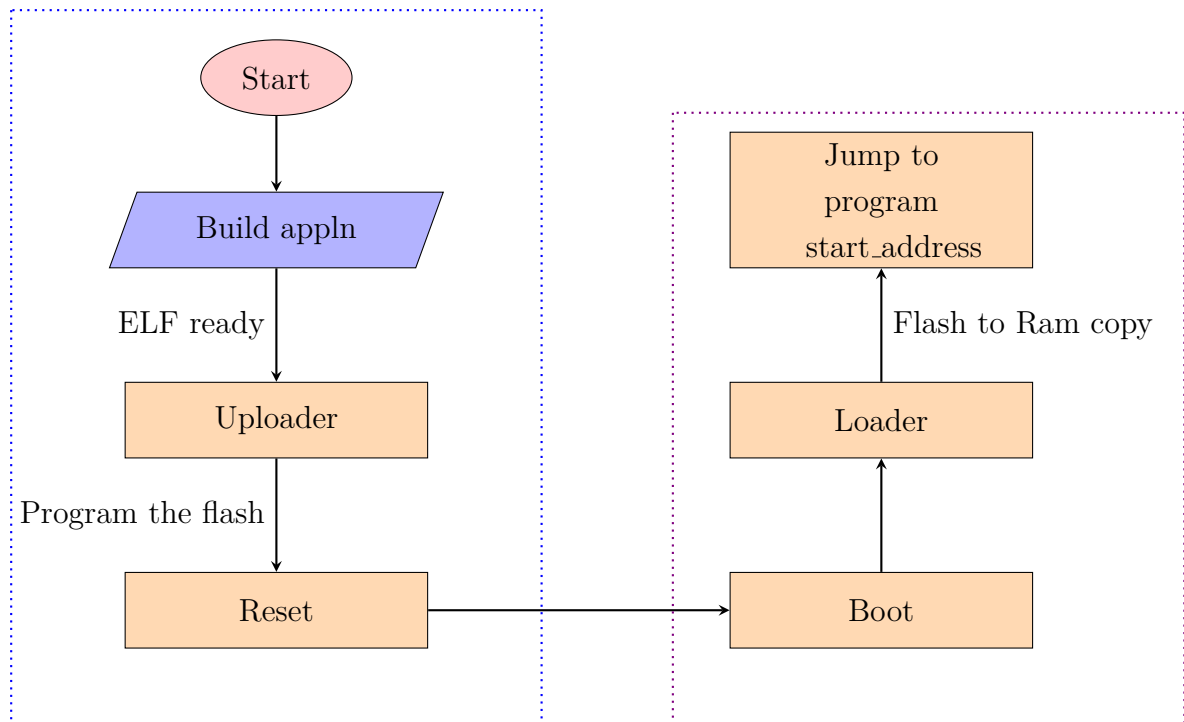


Figure 2: High level working diagram of Standalone mode

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 the ELF) to flash, after building the image.

```
$ cd shakti-sdk
$ make upload PROGRAM= bare metal appln TARGET=artix7_35t (or artix7_100t)
```

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.