International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 05 | May -2020 ISSN: 2582-3930

Booting Testcase Development for XILINX FPGA Devices

Dr. Anwar Bhasha Pattan, Ch.Yugasri Devi, K.Bhavani Sai Pooja, G.Manisha, Ch.Rohini Reddy

BVRIT HYDERABAD College of Engineering for Women
Rajiv Gandhi Nagar, Bachupally, Hyderabad

Abstract—

This paper focuses on developing the booting test cases for XILINX FPGA devices like ZYNQ and ZYNQMP. Booting is the process of starting a device. It can be initiated by hardware such as a button press, or by a software command. The heart of booting process is a file which can be of the extension .bin/.mcs specifically for XILINX FPGA devices. This file is called as "BOOTIMAGE" which is responsible for whole booting process of the device. A boot image contains the complete data (also called as partition), optionally contains the secure cases (if user seeks for secure process).

Keywords— Booting, Boot Image, ZYNQ, ZYNQM P

I INTRODUCTION

In present day market, the XILINX FPGA devices that are getting released are gaining huge positive response from the customers because of the good secure performance of the product/device. Xilinx FPGAs and system-on-chip (SoC) devices typically have multiple hardware and software binaries used to boot them to function as designed and expected. These binaries can include FPGA bit streams, firmware images, boot loaders, operating systems, and user-chosen applications that can be loaded in both non-secure and secure methods. In this paper we focus on developing the booting test cases for the Xilinx FPGA devices like ZYNQ and ZYNQMP.

The heart of booting process is a file which can be of the extension .bin/.mcs specifically for XILINX FPGA devices. This file of the format .bin/.mcs is called as the "BOOTIMAGE". The tool that generates BOOTIMAGE is "BOOTGEN". The secure boot of Xilinx devices uses public and private key cryptographic algorithms. BOOTGEN also supports encryption and authentication. Each partition can be encrypted and authenticated with the BOOTGEN. The output is a single file that can be directly programmed into the boot flash memory of the system.

BOOTGEN comes with both a GUI interface and command line option.[2] The tool as integrated into the software development toolkit, SDK, for generating basic boot images using a GUI, but the majority of BOOTGEN options are command line-driven. Command line options can be scripted. The BOOTGEN tool is driven by a boot image format (BIF) file configurations file with a file extension of *.bif.

II BOOTIMAGE LAYOUT

To build any BOOTIMAGE, there are two steps:
1) Create .bif file
2) Run the BOOTGEN executable to get BOOTIMAGE

Each device requires files in a specific format to generate a boot image for that device. Fig:1 shows ZYNQMP BootImage Layout. For ZYNQ device, the layout is similar to that of ZYNQMP, but PUF Helper Data partition is absent.

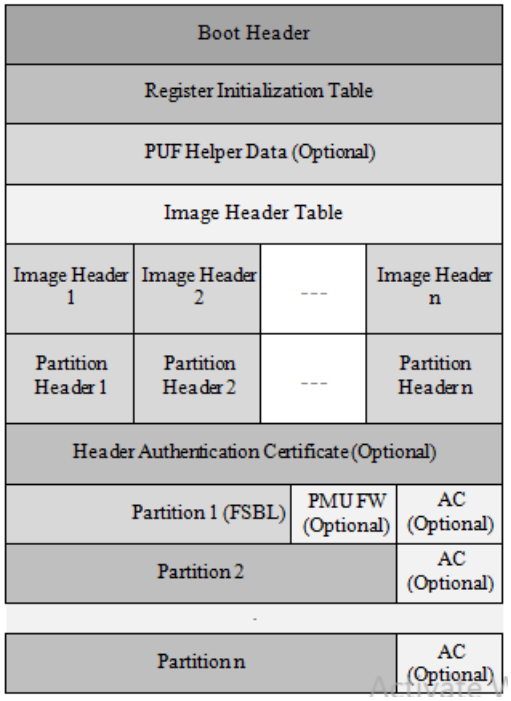

Fig1: The boot image layout for Xilinx ZYNQMP devices

***Boot Header*:**
The Boot Header is a structure that contains information related to booting the primary boot loader, such as the FSBL [3].

Register Initialization Table:
The Register Initialization Table in BOOTGEN is a structure of 256 address-value pairs used to initialize PS registers for MIO multiplexer and flash clocks [3].

***Image Header Table*:**
BOOTGEN creates a boot image by extracting data from ELF files, bit stream, data files, and so forth. These files, from which the data is extracted, are referred to as images [3]. Each image can have one or more partitions. The Image Header

table is a structure, containing information which is common across all these images.

Authentication Certificate:

The Authentication Certificate is a structure that contains all the information related to the authentication of a partition.

III ENCRYPTION

Encryption is the most effective way to achieve data security. We use a key to encrypt the data which we get from AES algorithm. Any key generated from this algorithm is generally called as AES key. Symmetric key Cryptography exits in this process. Single key (same key) is used for both encryption & decryption. AES key is used to encrypt the partitions. AES key is stored either in EFUSE or BBRAM in the device, for decryption purpose.

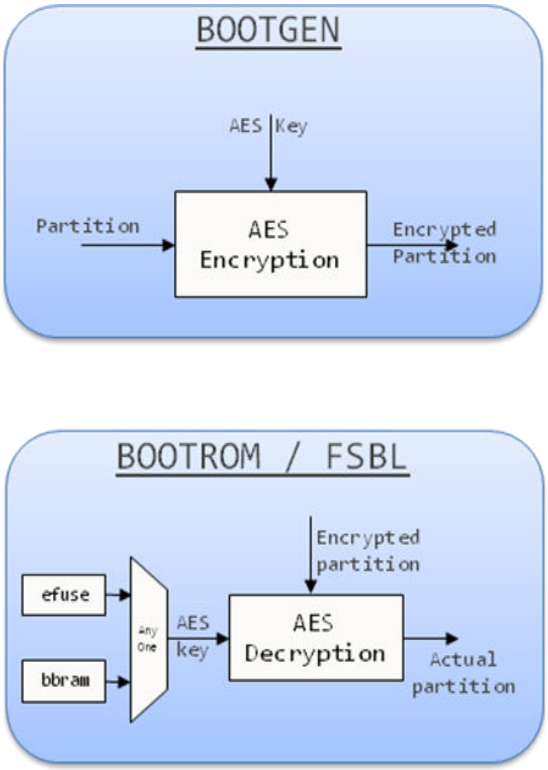

Fig 2: Block diagram showing the process of Encryption and Decryption

While developing the test cases for ZYNQMP devices, there is an extra secure step that can be used during encryption and decryption. ZYNQMP devices support multiple keys. They are:

- EFUSE RED KEY
- EFUSE BLACK KEY
- EFUSE GREY KEY
- BBRAM RED KEY
- BLACK KEY in BootHeader
- GREY KEY in BootHeader

For better understanding, Fig: 4, Fig: 5, Fig: 6 are describing encryption and decryption process in ZYNQMP

RedKey:

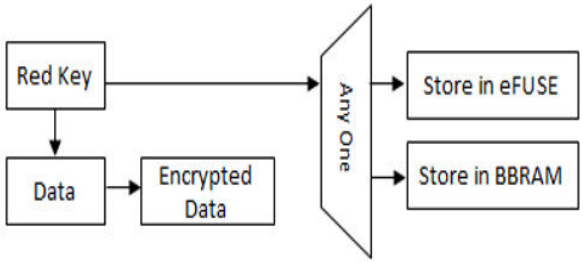

Fig 3: Encryption using redkey

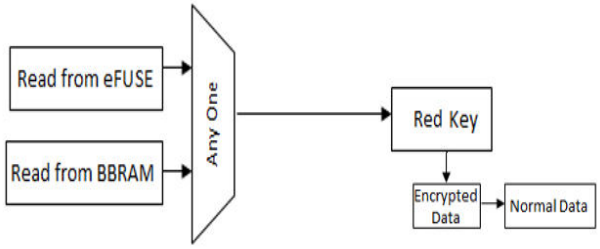

Fig 4: Decryption

BlackKey:

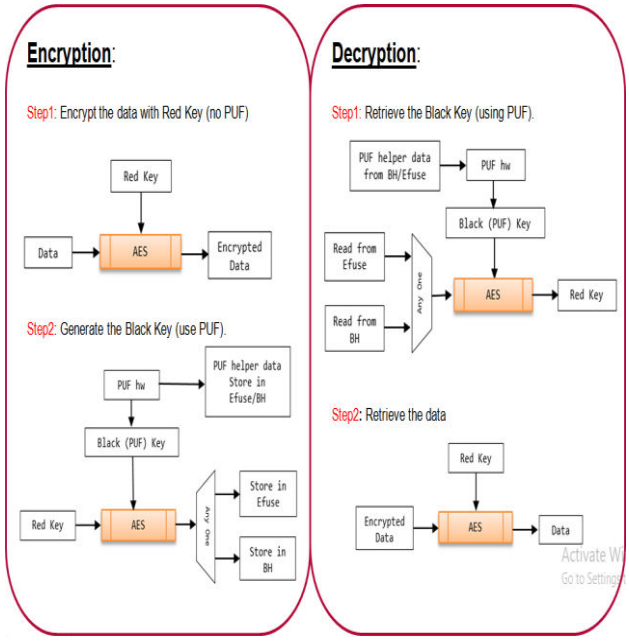

Fig 5: Encryption and Decryption using black key

GreyKey:

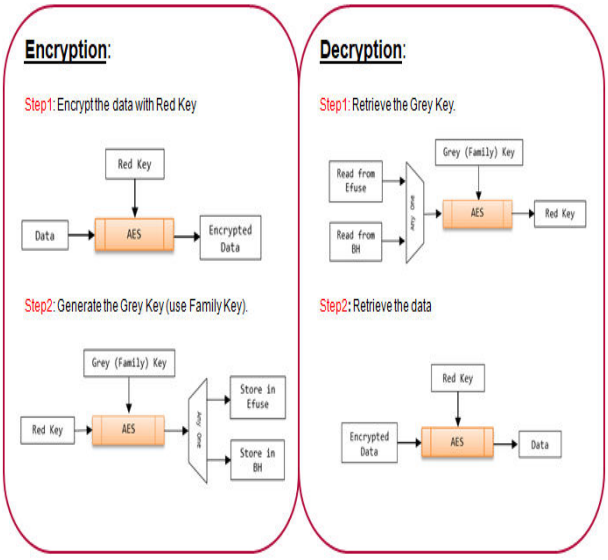

Fig 6: Encryption and Decryption using grey key

IV AUTHENTICATION

Encryption provides the basic design security to protect the design from copying or reverse engineering, while authentication provides assurance that the bitstream provided for the configuration of the FPGA was the unmodified bitstream created by an authorized user.[7] Authentication technology provides access control for systems by checking to see if a user's credentials match the credentials in a database of authorized users or in a data authentication server [4].

In the process of authentication keys can be generated using either ECDSA algorithm or RSA algorithm. The key that we use for signing the partition will not be used while verifying. This is the reason why the data is more secured. In authentication, there exists Asymmetric key cryptography. Two different keys are used, one for signing and other for verifying – Primary key, Secondary key.

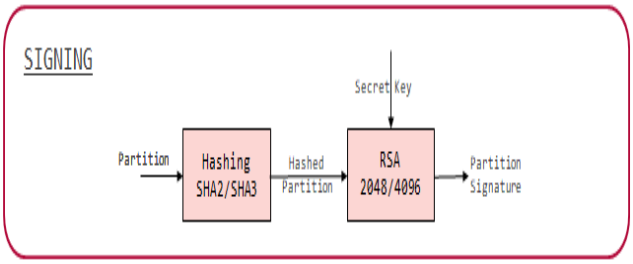

Fig 7: Signing the Partition

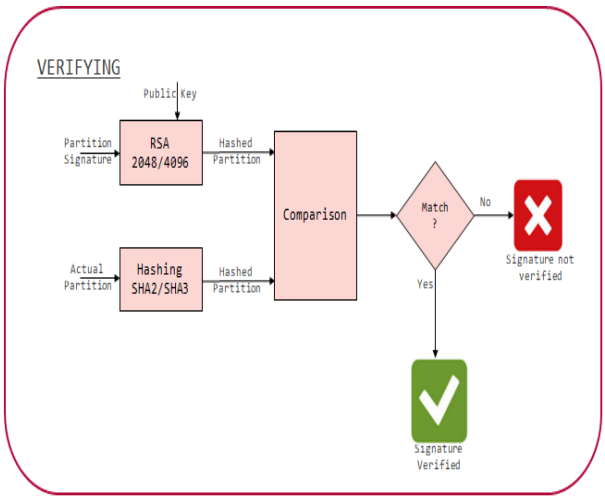

Fig 8: Verifying the Partition

In Xilinx SoCs, two sets of key pairs are used in this process

- Primary Key Pair
  - Primary Public Key (PPK)
  - Primary Secret Key (PSK)
- Secondary Key Pair
  - Secondary Public Key (SPK)
  - Secondary Secret Key (SSK)

Primary key pair is used to authenticate the secondary key pair. Secondary key pair authenticates the actual partition.

V TEST CASES DEVELOPED

Zynq:

(1) ENCRYPTION:

```
the_ROM_image:
{
        [keysrc_encryption] bbram_red_key
        [aeskeyfile] vnc_bbram_7020.nky
        [bootloader,encryption=aes] zynq_fsbl.elf
        [encryption=aes] system.bit
        [encryption=aes] hello_world_a9_0.elf
}
```

The code above is the encrypted bif file to create BOOTIMAGE for booting ZYNQ device. The storage location of the encrypting key here in this example is BBRAM and vnc_bbram_7020 is the key used. File zynq_fsbl.elf is the boot loader. Partitions in this .bif file are system.bit, hello_world_a9_0.elf.

(2) AUTHENTICATION:

the_ROM_image:

```
{
        [pskfile] pskfile.txt
        [sskfile] sskfile.txt
        [bootloader, authentication=rsa] zynq_fsbl_rsa.elf
        [authentication=rsa] hello_world_a9_0.elf
}
```

The code above is the authenticated bif file to create BOOTIMAGE for booting zynq device. Primary and Secondary keys used to authenticate the partitions are present in the text files pskfile.txt and sskfile.txt. ELF (Extensible Linking Format) file zynq_fsbl_rsa.elf is the boot loader which holds the information of the device specifications and other required information for loading data into the device. Partition in this .bif file is hello_world_a9_0.elf, this file hold the hello world application data .

ZynqMp:

(1) ENCRYPTION:

(a) the_ROM_image:

```
{
        [keysrc_encryption] bbram_red_key
        [aeskeyfile] vnc_bbram_7020.nky
        [bootloader,encryption=aes] zynqmp_fsbl.elf
        [encryption=aes] system.bit
        [encryption=aes] hello_world_a53_0.elf
}
```

(b) the_ROM_image:

```
{
        [keysrc_encryption] bbram_red_key
        [aeskeyfile] vnc_bbram_7020.nky
        [bootloader,encryption=aes] zynqmp_fsbl.elf
}
```

The code given above are two examples showing encrypted bif file to create BOOTIMAGE for booting ZYNQMP device. The storage location of the encrypting key here in this example is BBRAM and vnc_bbram_7020 is the key used. File zynqmp_fsbl.elf is the boot loader. Partitions in given .bif files are system.bit, hello_world_a53_0.elf. Partition is optional, can be seen in second example (i.e (b)).

***(2) AUTHENTICATION*:**

```
the_ROM_image:
{
        [pskfile] pskfile.txt
        [sskfile] sskfile.txt
        [bootloader, authentication=rsa]
        zynqmp_fsbl_rsa.elf
        [authentication=rsa] hello_world_a53_2.elf
}
```

The code given above is the authenticated bif file to create BOOTIMAGE for booting ZYNQMP device. Primary and Secondary keys used to authenticate the partitions are pskfile.txt and sskfile.txt. File zynqmp_fsbl_rsa.elf is the boot loader. Partition in this .bif file is hello_world_a53_2.elf.

VI CONCLUSION

In this paper the development of the test cases have been done for few FPGA devices. The FPGA devices that are chosen in this work belong to Xilinx FPGA devices which are ZYNQ, ZYNQMP. All the test cases for ZYNQ and ZYNQMP are done for encryption and authentication. Developed test cases thereby sent as input to the BOOTGEN tool then generated boot images and program flashed (Run on board) on the family of devices of ZYNQ (zc702, zc706 etc) and ZYNQMP (zc1715, zc102 etc).

Boot images generated from ZYNQ and ZYNQMP test cases mentioned in this paper are running successfully without any errors. ZYNQ boot images are tested on zc702,zc706 boards of ZYNQ family and ZYNQMP are tested on zc1715,zc102 boards of ZYNQMP family.

This study can be applied on more problems to generate new test cases for various FPGA devices. The scope of the test case development in today's world is more as industries (like Intel, Qualcomm etc) of various sectors give importance in testing their device before releasing into the market.

REFERENCES

[1] The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000. All Programmable SoC - by Louise H. Crockett.

[2] Bootgen user guide – ug1283 document.

[3] Xilinx Software command line tool (XSCT) – document.

[4] Network security in 'Identify and access management', Search security website.

[5] Xilinx Developer Forum (XDF)-2018- Introducing-the-versal-architecture.pdf

[6] ZynqMp book: Exploring ZYNQMP SoC with pynq and machine learning applications – by University of Strathclyde Glasgow

[7] Using Encryption and Authentication to Secure an UltraScale/UltraScale+ FPGA Bitstream - Author: Kyle Wilkinson