2024.11.14 Syswonder 组会报告

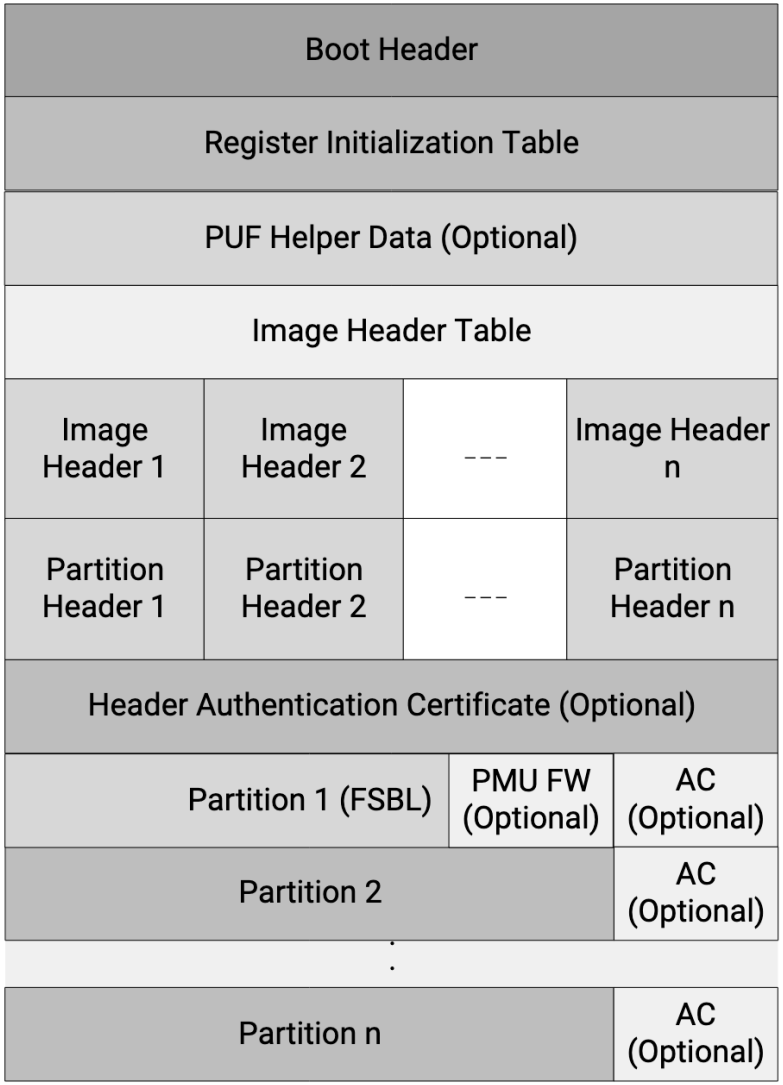
Introduction to Xilinx BIF(Boot Image Format) and Bootgen

韩喻泷

School of Computer Science, Peking University

2024 年 11 月 28 日

Boot Image Layout for Zynq UltraScale+ MPSoC



1. **Boot Header**: meta infos about FSBL and PMU firmware.
2. **Register Initialization Table**: addr-value pairs for init PS registers for MIO multiplexer and flash clocks.
3. **PUF Helper Data**: some infos for PUF(Physical Unclonable Function) to create the KEK(Key Encryption Key).
4. **Image Header Table**: pointers to first partition header and image header.
5. **Image Header**: each image can have multiple partitions. This contains image name, pointer to next image header, and partition count along with the corresponding partition header.
6. **Partition Header**: contains load and exec addrs, partition id, the corresponding image header, the next partition header. Most importantly, the offset of the partition data from the start of the boot image.
7. **Header Authentication Certificate**: infos about partition authentication.

Xilinx Boot Image Format(BIF)

- Xilinx **Boot Image Format** (BIF) is a text file (`*.bif`) that specifies the memory layout of the boot image.
- The BIF file is used by the Xilinx *Bootgen* tool to create a boot image.
- Bootgen can be used in Xilinx Vitis GUI or CLI.
- Docs: `https://docs.amd.com/r/en-US/ug1283-bootgen-user-guide`
- Github: `https://github.com/Xilinx/bootgen`

Example BIF for linux kernel[1]

```
1 the_ROM_image:
2 {
3 [bootloader, destination_cpu = a53-0]fsbl_a53.elf
4 [destination_cpu=pmu]pmu_fw.elf
5 [destination_cpu=a53-0, exception_level=el-3, trustzone]
      bl31.elf
6 [destination_cpu=a53-0, exception_level=el-2] u-boot.elf
7 [offset=0x1E40000, load=0X10000000, destination_cpu=a53-0]
      image.ub
8 }
```

Listing 1: Example BIF for linux kernel

- Supported files by bootgen: .bin, .dtb, image.gz, .elf, .bit, etc.

[1] Xilinx Bootgen User Guide, UG1283 (v2022.2) December 14, 2022

How to run hvisor using JTAG?

- in petalinux generated boot.scr, jtag uses these default addresses(and configurable via menuconfig):
  - 0x100000 for dtb
  - 0x200000 for kernel
  - 0x4000000 for ramdisk
  - 0x10000000 for uboot FIT(Flat Image Tree) image[2]
  - 0x20000000 for the boot.scr script
- **for hvisor with root linux**: since we have a lot of things to pack that need to be loaded using uboot, using uboot **Flat Image Tree**(FIT) image is a good choice. 😀

[2] https://www.thegoodpenguin.co.uk/blog/u-boot-fit-image-overview

What's uboot FIT image[3]?

- u-boot's **Flattened Image Tree**(FIT) image uses syntax of Device Tree Source(DTS) to describe the images to be loaded, which is similar to linux's Flattened Device Tree(FDT).
- FIT source files: `*.its` (image tree source).
- You can include multiple kernel images, dtbs, ramdisks, etc. in a single FIT image.
- So we add our hvisor.bin, root linux's kernel, dtb, and ramdisk into the FIT image and packed it as `image.ub`.

[3] `https://docs.u-boot.org/en/stable/usage/fit/index.html`

Example uboot FIT image source file for hvisor

```
/dts-v1/;
/ {
    description = "hvisor FIT image";
    images { // 用于描述要加载的各个image和其他数据
        kernel@1 {
            description = "hvisor";
            data = /incbin/("hvisor.bin");
            ...
            load = <0x40400000>;
            entry = <0x40400000>;
        };
        kernel@2 {
            description = "root linux kernel";
            data = /incbin/("Image");
            ...
        };
        fdt@1 {
            description = "root linux dtb";
            data = /incbin/("linux1.dtb");
            ...
        };
        ramdisk@1 {
            description = "root linux ramdisk";
            data = /incbin/("rootfs.cpio.gz");
            ...
        };
    };
    configurations { // 通过conf实现不同场景使用不同的文件
        default = "conf@1";
        conf@1 {
            kernel = "kernel@1";
            fdt = "fdt@1";
            ramdisk = "ramdisk@1";
        };
        ...
    };
};
```

Last Week's Bugs

- The root linux won't boot and stuck when init the `liointc`(legacy io interrupt controller).
- After removing `liointc` in dts, root linux can boot but non-root cannot boot because hvisor stuck when issuing IPI from root to non-root CPU.
- The two problems are all related to the board chip registers.
- These problems have not been met in the previous 3A5000 motherboard.
- **Assumption**: the new board's UEFI firmware initialized the chip and io controller registers differently somewhat.

My Hacky Solution

- First, force the board to use *legacy interrupt mode*.
- Then, force each core to clear the boot IPI in irqchip init process.
- Finally, force re-enabled all IPI enable-bits in chip registers.
- **Result:** now the root/nonroot linux can mount `liointc` smoothly, and hvisor's IPI worked as well. ☺

Thanks for watching!

- **TODO**: debug nonroot virtio