Productizing the Linux boot time tweaks and tricks – an engineering problem !

Khasim Syed Mohammed
khasim@ti.com

Engineering Lead – Texas Instruments

20 Sept 2024

Scope

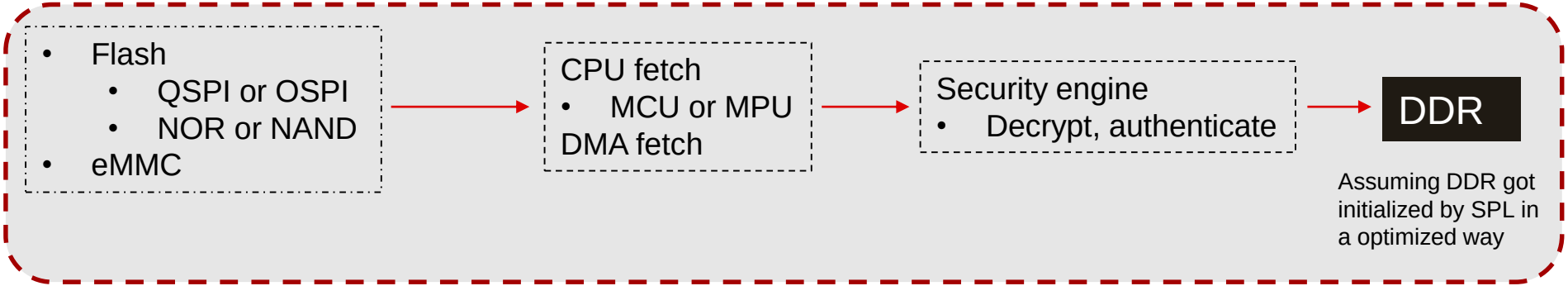
Boot time plays an important role in defining the user experience of a product, the more time it takes in getting the device into action - the quicker it is pulled out of the stands.

Linux & it's stacks can be tweaked to boot as quickly as possible but **the challenge is beyond just optimizing the time or path it takes :–**

- it gets into defining the use cases to go after
- productizing these features
- deploying in test farms
- delivering to customers.

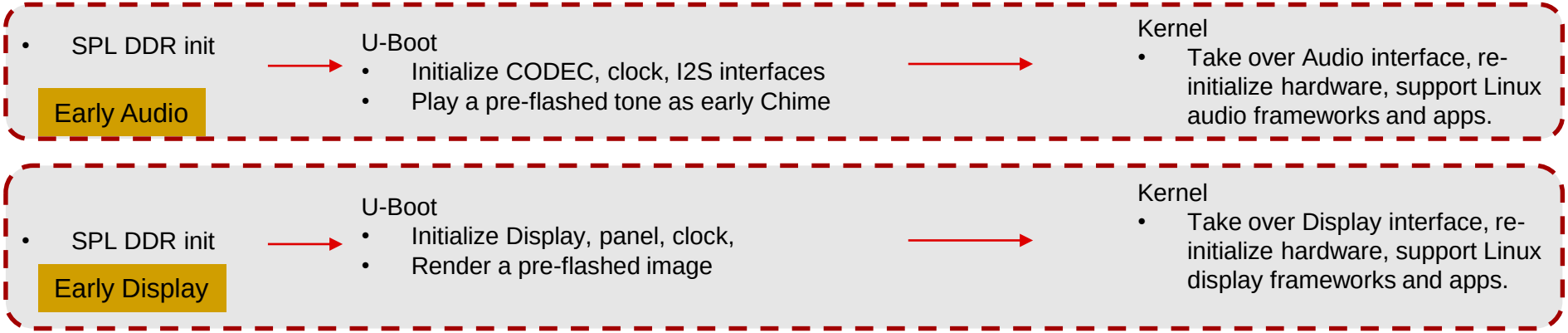
Boot != hitting kernel prompt

Problem 1 : Identifying those minimum & complicated Fixed Functions



- They impact boot time of Linux and boot loaders
- Very closely tied to hardware : clock, CPU speed, bus width, mechanism supported in hardware IPs (DMA/Authentication schemes, etc).
- Its mainly SOC architecture, platform owner responsibility – than open source community.

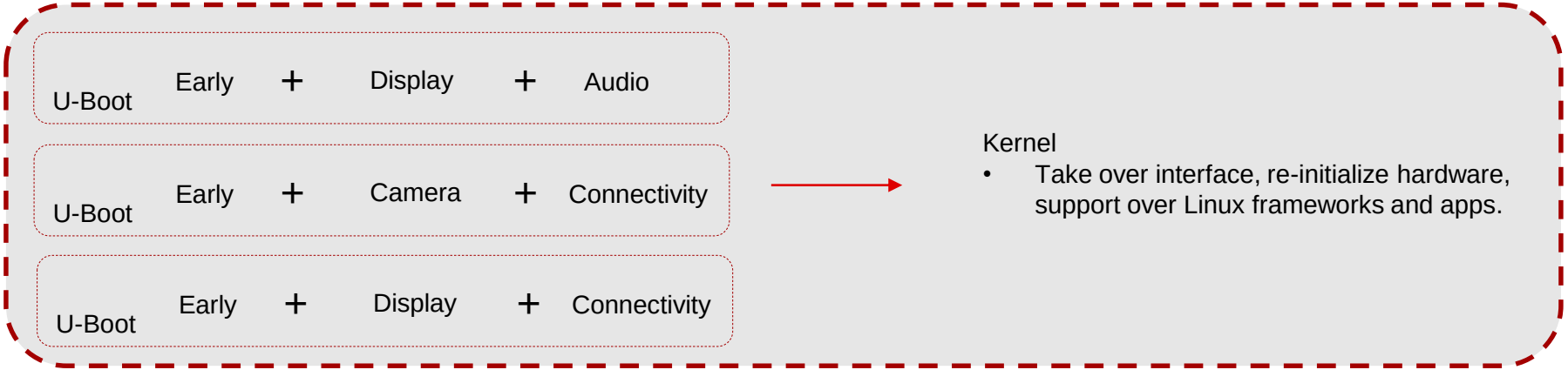
Problem 2 : Tweaking the flow for individual Early Use Cases



Challenges here :

- Need a standard mechanism to notify to Linux kernel that peripheral got initialized already.
  - **Display** : supports it through simple-framebuffer DT node.
  - **Connectivity** : No similar mechanism for Ethernet, USB, CAN – today Links go down and come back when kernel comes up due to re-initialization.
  - **Audio** : strange problem, it finishes playing the tone and then boots kernel – impacts kernel boot time or introduces glitches.

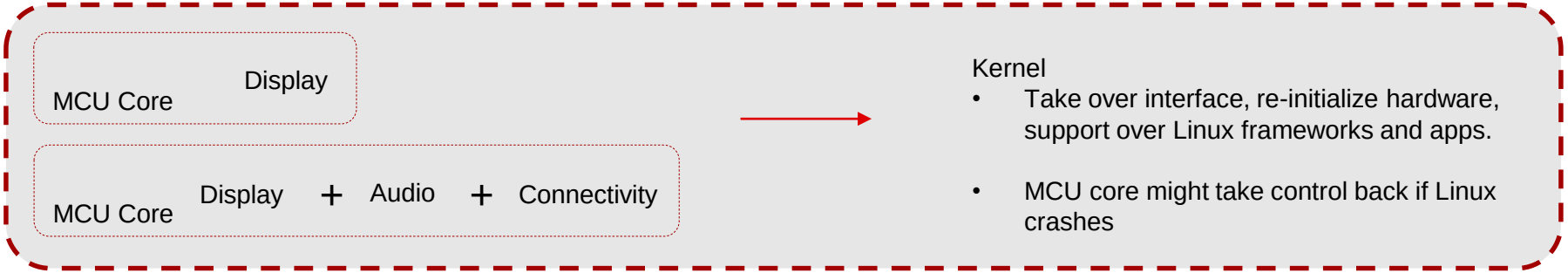
Problem 3 : Combo use cases



Challenges here :

- Need a standard mechanism to notify to Linux kernel that **multiple peripherals** got initialized already.
- Peripherals get enabled sequentially, which one goes first and second need to be prioritized as per product needs.
- Boot loaders are not multi-threaded at least U-Boot ?

Problem 4 : MCU core accelerated with Linux Late attach



Challenges here :

- Need a standard mechanism to notify to Linux kernel that peripheral got initialized already **by a remote core**.
- Safety and time critical applications go first on MCU core and leverage A-core running Linux for high end processing.

Problem 5 : Packaging and Delivery of Optimizations and Customizations

Challenges here :

- Industry has resolved most of the issues in a hacked and customized way.
  - How and where do we document these customizations for community to leverage the learnings and findings.
  - TI does it in SDK distributed through ti.com
- User space customization can be hosted as a separate OE Config or file system “wic” image with each customization – Yocto way.
  - How to host for other distributions like Debian, Buildroot.
- Rebasing with every kernel RC is painful if patches don’t go upstream.
  - Any suggestions on where to host such custom kernel configs for community to collaborate ?
  - We are on github if that is helpful ?
- Test automation – there are no boot time specific automated testing like RT-tests, etc. that gets tested in a automated environment for kernel RCs.

Key care about : How do we let community know about such things and Collaborate ?

Credits

Thanks to the following team members for helping in populating this info :

- Aashvij Shenai <a-shenai@ti.com>
- Devarsh Thakkar <devarsht@ti.com>
- Siddharth Vadapalli <s-vadapalli@ti.com>

Community Partners