Power management in Linux together with secure firmware

Joakim Bech and Vincent Guittot

An environment with shared resources

- Linux is (usually) the main component controlling devices
  - Clocks
  - Power Domains
  - Hardware IP
  - etc ...
- Secure side firmware also controls and uses components
  - Crypto blocks
  - Perhipherals
  - External cache maintenance
- Secure side directly controls a few things
  - PSCI

Linaro
connect
Budapest 2017

ENGINEERS
AND DEVICES
WORKING
TOGETHER

Crypto hardware blocks

- It’s not uncommon that crypto hardware blocks are shared between normal and secure world
- Without proper synchronization, things can go really wrong, example:
  - Secure side does a crypto job and normal world decides to turn off the block
  - Both normal and secure side tries to use the crypto block at the same time
- I.e., if the hardware itself cannot handle multiple sources, then there is a need for some kind of synchronization primitive shared between secure and non-secure world

Linaro
connect
Budapest 2017

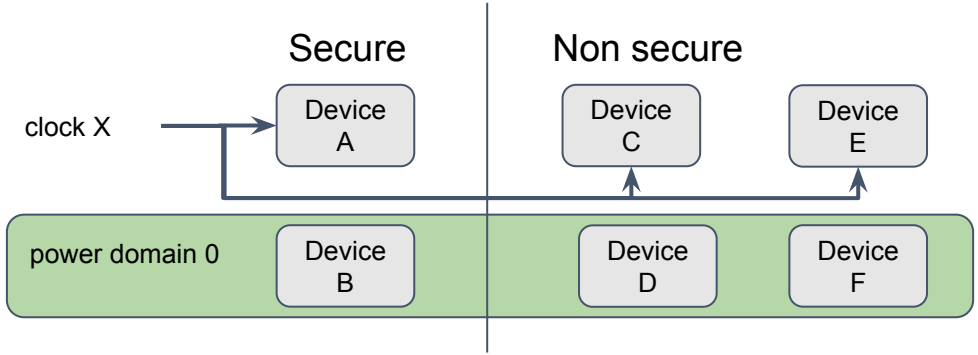
ENGINEERS
AND DEVICES
WORKING
TOGETHER

CPU Latency

- Using PSCI (secure side) is mandatory on ARMv8-A
- Latency for powering up and down CPU/clusters have increased compared to previous architectures
  - The addition of switching and running code in secure side also seems to have added a significant cost in terms of latency
- Ongoing effort to consolidate CPU idle state decision into the scheduler
- Some state doesn’t need any save of the context
  - Like CPU retention or Cluster retention

Device Latency

- Manage shared power resources between secure and non secure devices



- Non secure side
  - Secure world interaction for every device state change is not efficient
  - power domain can consolidate non secure state
- Secure side
  - Can’t trust non secure world

Linaro connect
Budapest 2017

ENGINEERS AND DEVICES WORKING TOGETHER

Suspend to RAM - done right

1. ARM recommends:
   1.1. REE saves states, REE transitions to TEE, TEE saves states
   1.2. TEE communicates with PMIC to put the device into low power standby
2. Partners to users of OP-TEE reported that they often see:
   2.1. REE saves states, REE transitions to TEE, TEE saves states
   2.2. TEE transitions back to REE
   2.3. REE communicates with PMIC to put the device into low power standby

- I.e., REE PMIC bypass the TEE => (Usually) not good!
  - This can lead to memory protected by the TEE can become available to the REE
  - There is a also a risk of corrupting TEE states
- Partners expressed wishes that TEE's supports both TEE and REE controlled PMIC flow.

Thank You

#BUD17
For further information: www.linaro.org
BUD17 keynotes and videos on: connect.linaro.org
joakim.bech@linaro.org and vincent.guittot@linaro.org