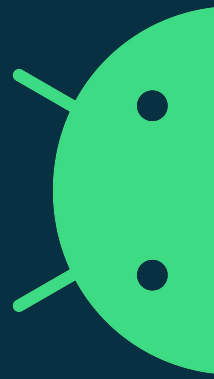
Page-based Hardware Attributes (PBHA)



Linux Plumbers Conference, 2021

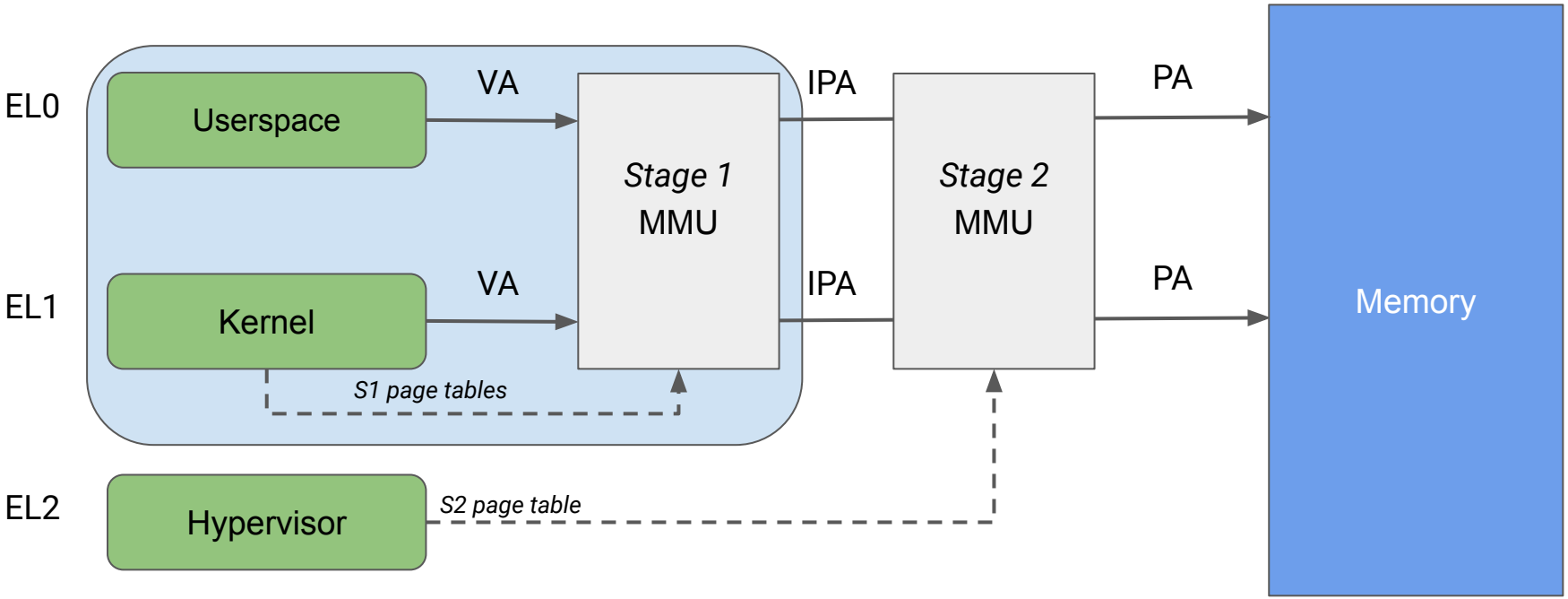
Will Deacon <will@kernel.org>

Introduction

Who am I and what am I hacking on?

- Active upstream kernel developer, co-maintaining the arm64 architecture port, locking, atomics, memory model, TLB, SMMU, ...
- Part of the Android Systems Team at Google
  - Over 3 *billion* active devices
  - Android Common Kernel
  - SoC agnostic
  - Some visibility into upcoming SoC designs
  - Upstream-first approach
- Active engagements with Arm architecture

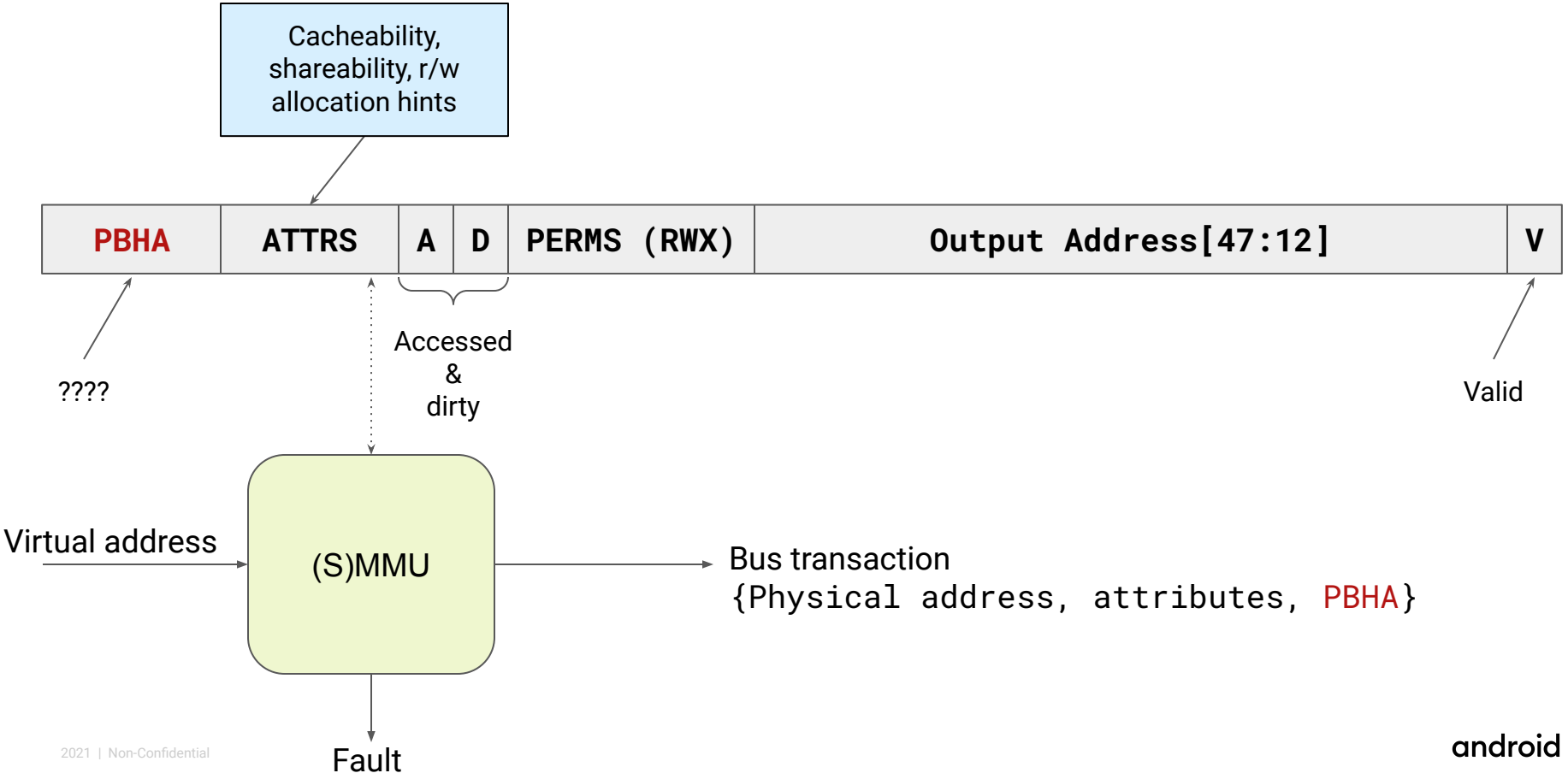
Recap: arm64 MMU (CPU and SMMU are similar)



Page tables used for address translation, permission checking and memory attributes

android

Rough overview of the arm64 PTE format



android

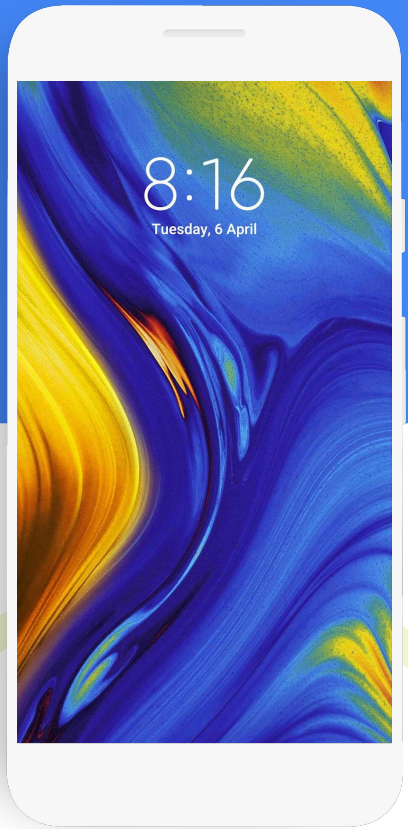
“When [PBHA is enabled then] hardware can use that PBHA bit for IMPLEMENTATION DEFINED purposes [and] the value of 0 in the PBHA bit is a safe default setting that gives the same behavior as when the PBHA bit is not used for IMPLEMENTATION DEFINED purposes.”

Currently disabled upstream.

Page-based hardware attributes (PBHA)

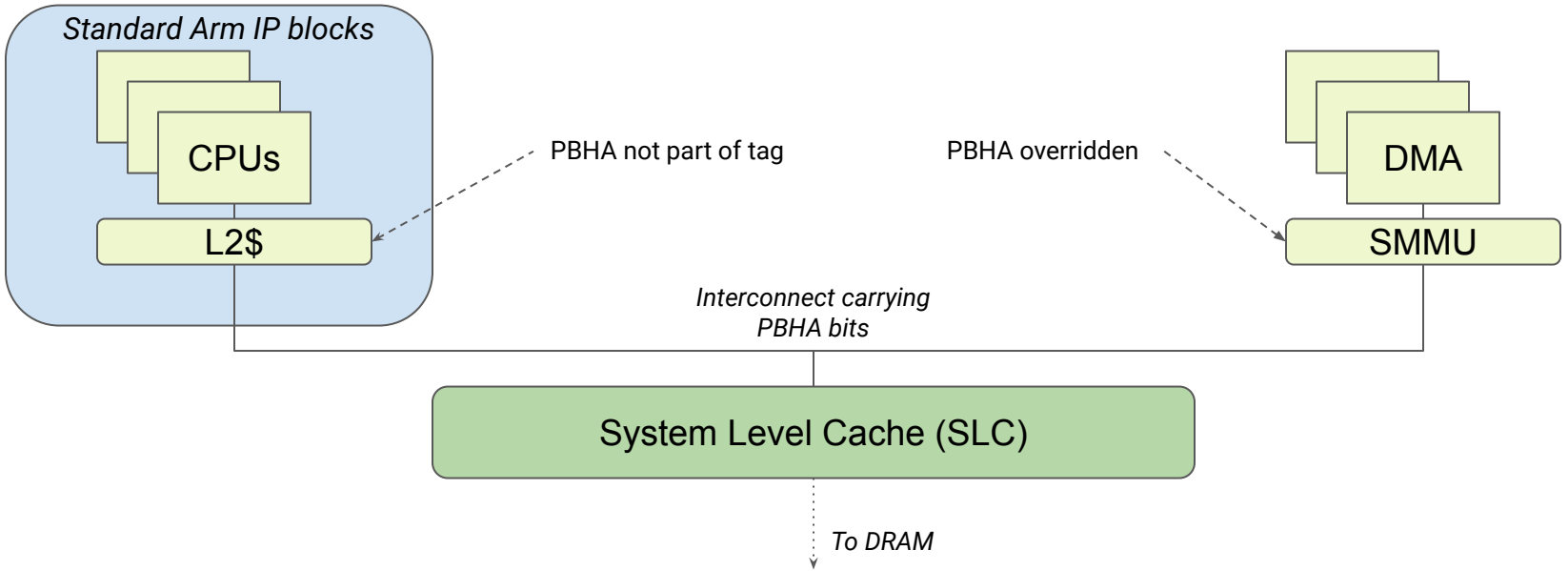
IMPLEMENTATION DEFINED means we have no idea what these bits do when enabled!

- Are they merely performance hints or do they affect functionality?
- Must all mappings of a given page use the same bits?
  - Including between CPU and devices?
  - What about speculative accesses (e.g. via the linear map)?
- What happens when the SMMU is disabled?
- How many bits are available and how do they correspond to hardware behaviours?
- Can the bit assignments change at runtime?
- How do we expose them to device drivers?
- How do we expose them to userspace?
- How do they interact with virtualisation?



android

Example of how these bits could be used in a system



An “invisible SLC” sits immediately before the memory controller and is able to cache *all* transactions

- Performance and prefetch hints
- Caching policy (beyond r/w allocation)
- QoS
- Data format

Problems supporting PBHA for an SLC in Linux

- Architecture
  - Doesn't specify how stage-1 and stage-2 interact (defined on a per-CPU basis)
  - Need to probe number of bits supported by each CPU

- Functional changes (e.g. data format) appear to be infeasible
  - Presence of cacheable alias with "incorrect" bits can lead to data corruption
  - How to ensure congruence between all virtual aliases (and efficiently)?

- Performance hints look more tractable
  - Firmware description of PBHA encodings
    - Require these to be static once Linux is running?
  - Plumb into DMA_ATTR_*?
    - Looks horribly opaque...
  - Per-page mapping, not per-transaction
  - Needed for anonymous memory or only for driver ioctl()s?

- QoS via resctrl ?

If you know of any other practical use-cases then please shout!

How do we fix it?

Thanks!

<will@kernel.org>