PM-QoS? Naah..It is PnP QoS

Sundar Iyer, Mark Gross, Premanand Sakarda, Ajaya Durg, Muthukumar Kalyan, Anand Bodas, Manoj Dawarwadikar

Mobile & Comms. Group, Intel

Special Thanks to:

Ticky Thakkar, Jasmin Ajanovic, Paul Chelladurai

Agenda

- Increasing (or decreasing?) complexity SoCs
- Power or Performance: A very strong “AND”
- What are we (Intel) facing?
- How we think is the deliverance?

smartphone/tablets/devices: Linux

- Android pushes out in practically any hand-held
- Overall experience/features remain same mostly like browsing, media, camera

BUT…..

- Different expectations, different tunings, different resolutions, different usage models
- Potentially different fusing for SoC internals

SoC: Squared Order of Complexity

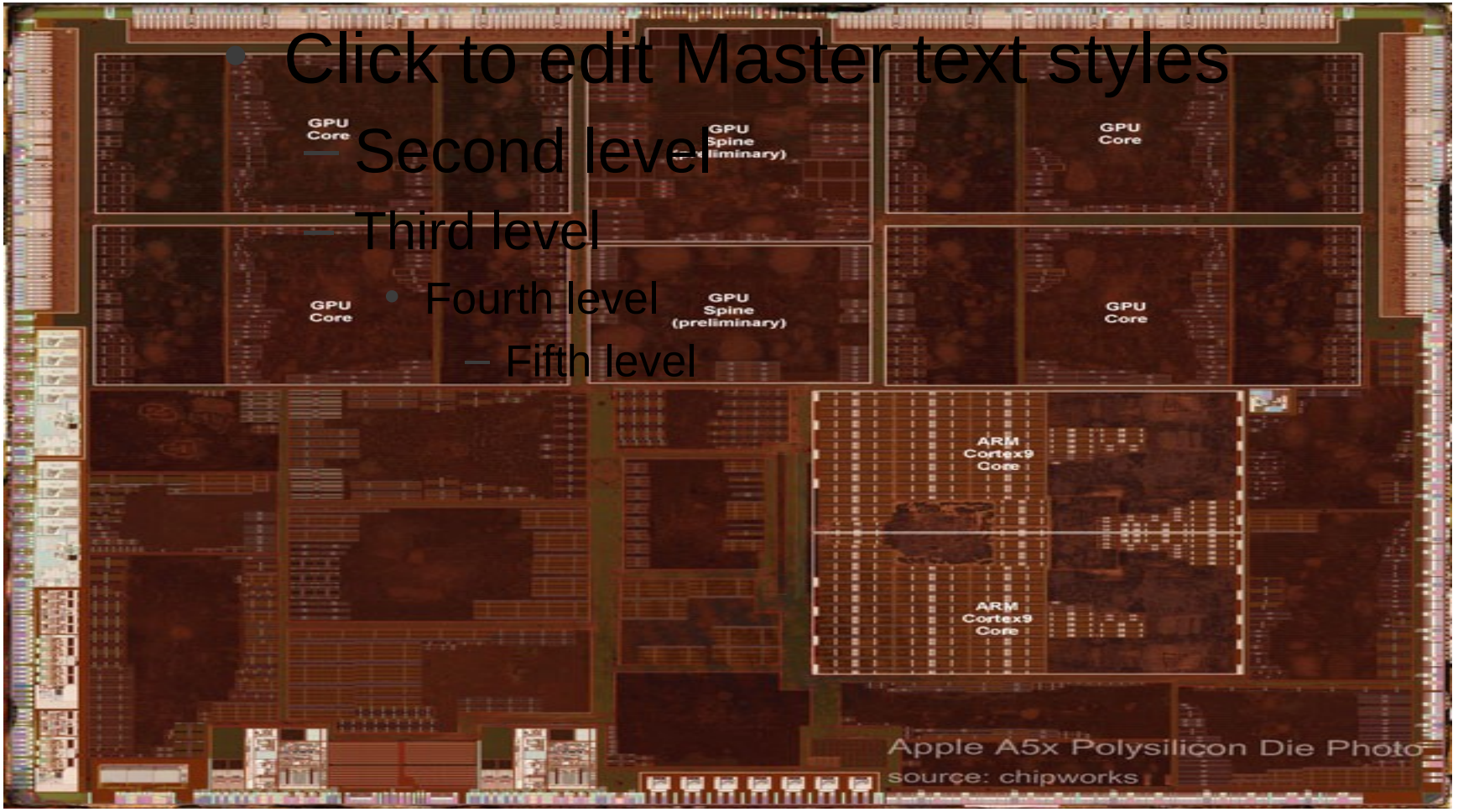
- Mostly same IPs, features, functionality BUT
  - Different Interaction between multiple components/accelerators/underlying fabrics
  - Subtle plumbing changes
- Inter-component dependencies
  - Underlying fabric and accelerator interactions
  - Common shared power rails/platform power/performance states

SoC Platform : Latest Mantra(s)

- Power AND Performance
  - Platform must deliver peak performance
  - Battery life must be long, yet not infinite!
  - While reliably delivering QoS for UX and device use cases.
- Beyond the AP Core
  - Increasingly complex non-core components
  - Scalability needs to be inclusive of non-core

Increasing non-CPU weight: A5X

- Click to edit Master text styles
  - Second level
    - Third level
      - Fourth level
        - Fifth level



Guaranteed non-CPU Performance

- How can non-CPU units (Graphics, Imaging, Video) run in guaranteed performance mode?

- Each accelerator is
  - unique
  - different set of constraints
  - different set of dependencies w.r.t CPU

What have we (Intel) faced?

- Few prominent issues
- Memory Self Refresh latencies impact performance
- Fabric speed common denominator for all CPU/non-CPU workloads

Security Encryption Performance

- Accelerated workload, very low CPU utilization
- Idle CPU pro-actively seeks deeper C-states; SoC kicks in higher DDR self refresh latencies based on C-states
- Memory throughput hurts for crypto engine IP block throughput.
- SOLUTION: PM_QOS_MEMORY_SELF_REFRESH_LATENCY QoS Class

Graphics Performance boost

- If Graphics requires very high throughput, needs the fabric to be running in burst mode
- Fabric burst mode tied to CPU frequency; GFX accelerated workloads tend to have CPU not running full steam
- Need to convert graphics burst into a comparable CPU frequency
- SOLUTION: platform dependent CPUFreq QoS request for such workloads.

CPU throughput QoS?

- https://lkml.org/lkml/2012/3/4/178
- https://lists.linux-foundation.org/pipermail/linux-pm/2012
- IMO These where shot down because the use of KHz units for CPU throughput in QoS API is not portable across uArch or ISAs.
- Could a platform shim be used?

Our deliverance?

- What if QoS is parameterized for every IP block and fabric?
  - Makes it generic across SW
  - Make is easier to map requirements
  - Exposes non-portable low level abstractions (kernel only?)
- What if non-portable QoS settings are accessed mostly through a per-platform shim?
  - Platform-shim interfaces remain portable across ISAs while mapping to non-portable (1 to many) ones?
  - Allows conversions specific to platform
  - Allows solid abstractions to enable scaling across ISA’s and uArch’s

A example QoS class for UX (Interactive Governor)

- QoS can be completed via
  - Specifying X fps (i.e. 60)
  - Specifying timeout
- Based on above parameters for Intel SoCs
  - Minimum cpu freq needed for burst mode memory
  - P-unit hint to push GPU ip block into burst mode
- Other SoC's
  - Goose memory bandwidth to max
  - Limit lowest CPU freq.
  - Limit lowest GPU freq.

A example QoS class for Imaging

- QoS can be completed via
  - Specifying X fps
  - Specifying resolution, XxY or 1080p/720p
  - Specifying output format, VP8, h264...
- Based on above parameters,
  - Limit Minimum memory bandwidth,
  - Limit Minimum Imaging IP (ISP) frequency
  - Limit Minimum MIPI frequency or throughput for sensor to ISP fabric
  - Limit video encoder IP block's min freq and related fabric clocks rates
  - Limit eMMC minimum data rates...TBD

Further Idea(s)

- W.R.T previous example, application can specify the accelerator loads….BUT
- Is it possible to also “account” CPU activity?
  - Number of threads an app spawns
  - A % of CPU utilization that the app will consume
  - A characterization in terms of MIPS, cycles/sec?
  - Deadline scheduling?
  - Active cooling through idle injection?
  - Timed out QoS or “burst” requests?