Scaling interconnect bus

Georgi Djakov / Vincent Guittot

LEADING
COLLABORATION
IN THE ARM
ECOSYSTEM

OSPM 2018 - Pisa

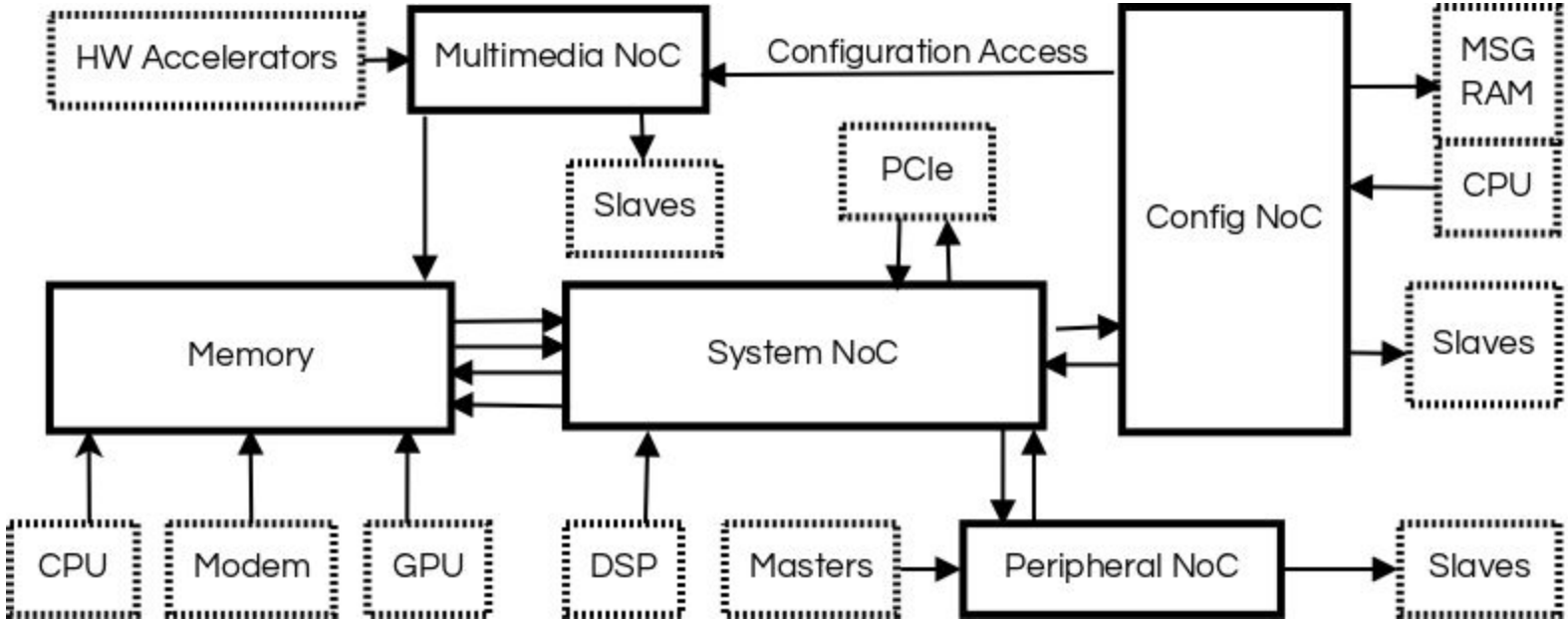
Agenda

- Some background
- The problem
- Current status
- Discussion on open issues

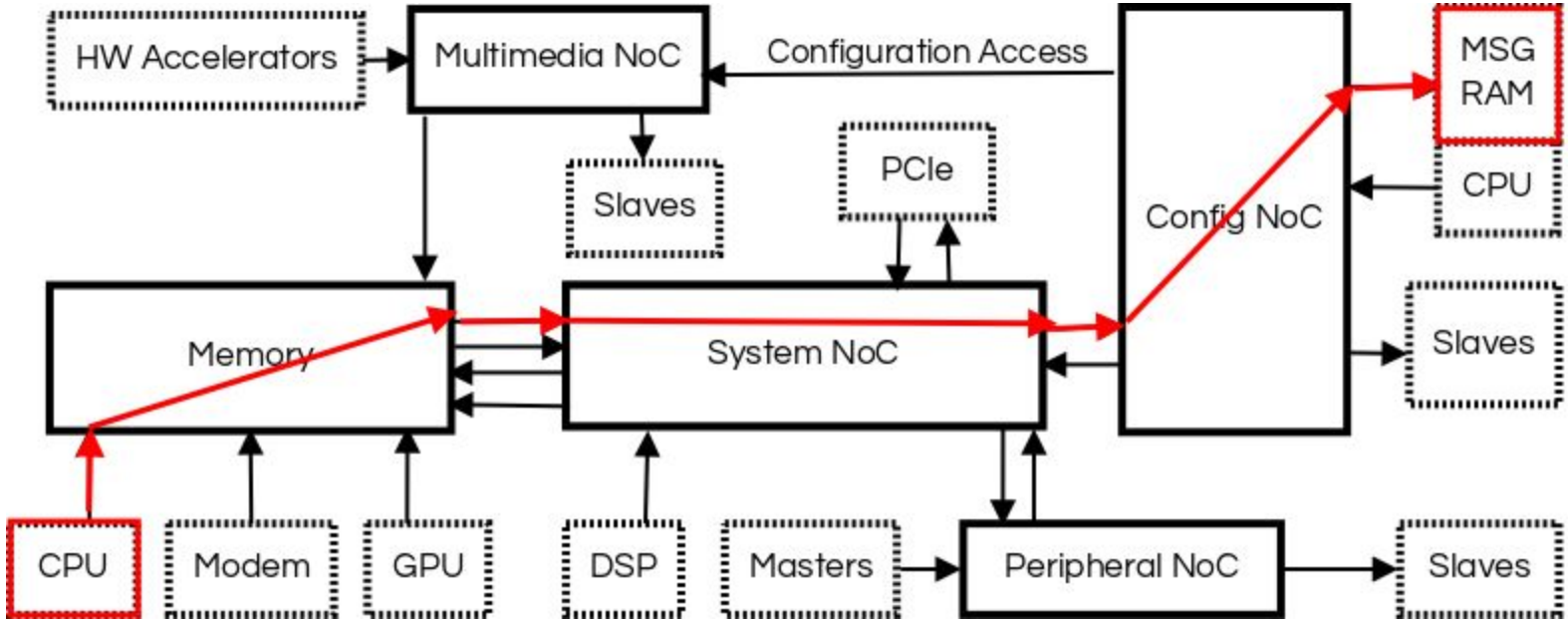
Some background

- ARM SoC architecture becomes more complex
  - More and more features (IP cores)
  - Many components talking to each other
  - Multiple sources of traffic
  - Concurrent transfers
  - Predictability (many interrupts, DDR utilisation)
- Evolution of on-chip interconnects
  - Buses, crossbars
- Network-On-Chip (NoC)
  - Packet transport protocol - scalability
  - Shorter wires - power efficiency
  - QoS, load balancing

An example topology



An example topology



On-Chip interconnects and Linux

- On-chip interconnect buses can handle high throughput data transfers, but most of the time they may be idle.
- Simultaneous data flows across the SoC with different sources and destinations, interleaved traffic.
- Interconnect buses can be configured according to the use-case and demand.
- Each SoC vendor has its own custom implementation in downstream kernels.
- Need a common solution in the upstream Linux kernel.

Current status

- Interconnect framework
  - re-configure the hardware dynamically according the use-case
- Interconnect providers
  - contains the topology
  - vendor specific implementation for set() and aggregate()
- Interconnect consumers
  - use get/set/put API functions

Provider API

Register interconnect topology from a SoC platform driver.

- int **icc_provider_add**(struct icc_provider *provider);
- struct icc_node ***icc_node_add**(int id);
- int **icc_link_create**(struct icc_node *node, const int dst_id);
- int **icc_provider_del**(struct icc_provider *provider);

Consumer API

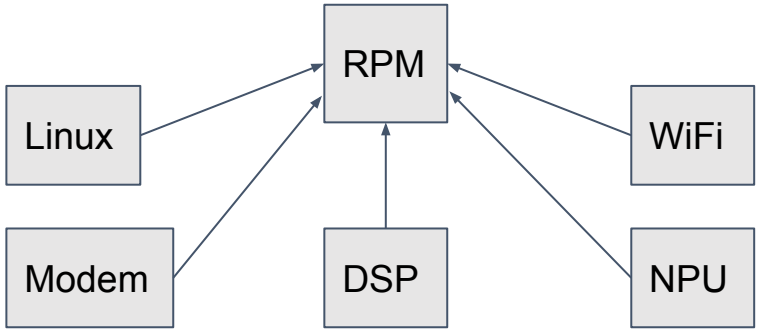
Consumer drivers express their bandwidth needs.

- struct icc_path ***icc_get**(const int src_id, const int dst_id);
- struct icc_path ***of_icc_get**(struct device *dev, const char *name);
- int **icc_set**(struct icc_path *path, u32 avg_bw, u32 peak_bw);
- void **icc_put**(struct icc_path *path);

Discussion on open issues

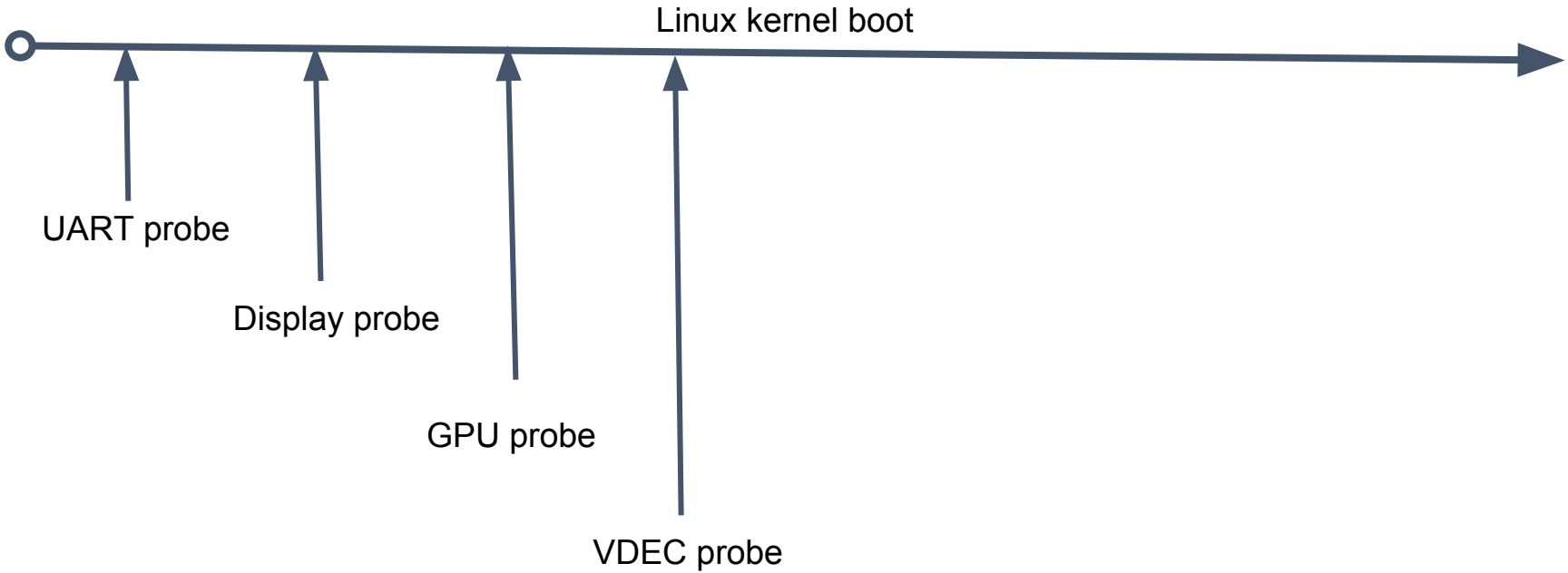
- Consumer might not know its bandwidth needs
- SoC specific predefined bandwidth values for paths, device idle states
- Active/sleep sets on Qualcomm platforms
- Extend boot constraints patchset by Viresh for interconnects?
- Merge path

Active set / Sleep set



Resources	Active Set	Sleep Set	Active*	Sleep*
Resource A	100 MHz	0 MHz	100 MHz	0 MHz
Resource B	1000 mV	[no request]	1000 mV	1000 mV
Resource C	[no request]	[no request]	[off]	[off]

First consumer vote



Thanks!