



Synx: cross-core synchronization

Pravin Kumar Ravi

Staff Engineer, Qualcomm

Oct 7, 2026 @ Driver Core MC, LPC, Prague, Czech

pravin.ravi@oss.qualcomm.com



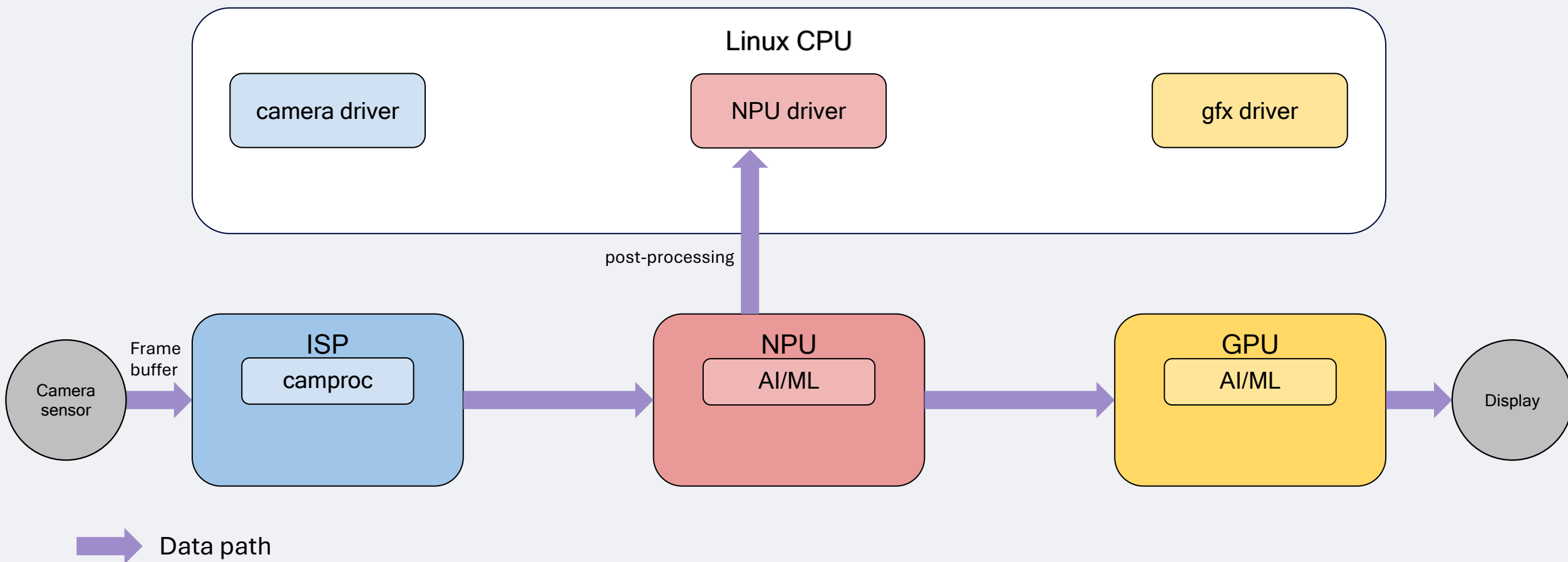


Agenda

- Real-time heterogeneous use cases
- The unsolved synchronization gap
- Preferred model
- Discussions

Heterogeneous use case example 1

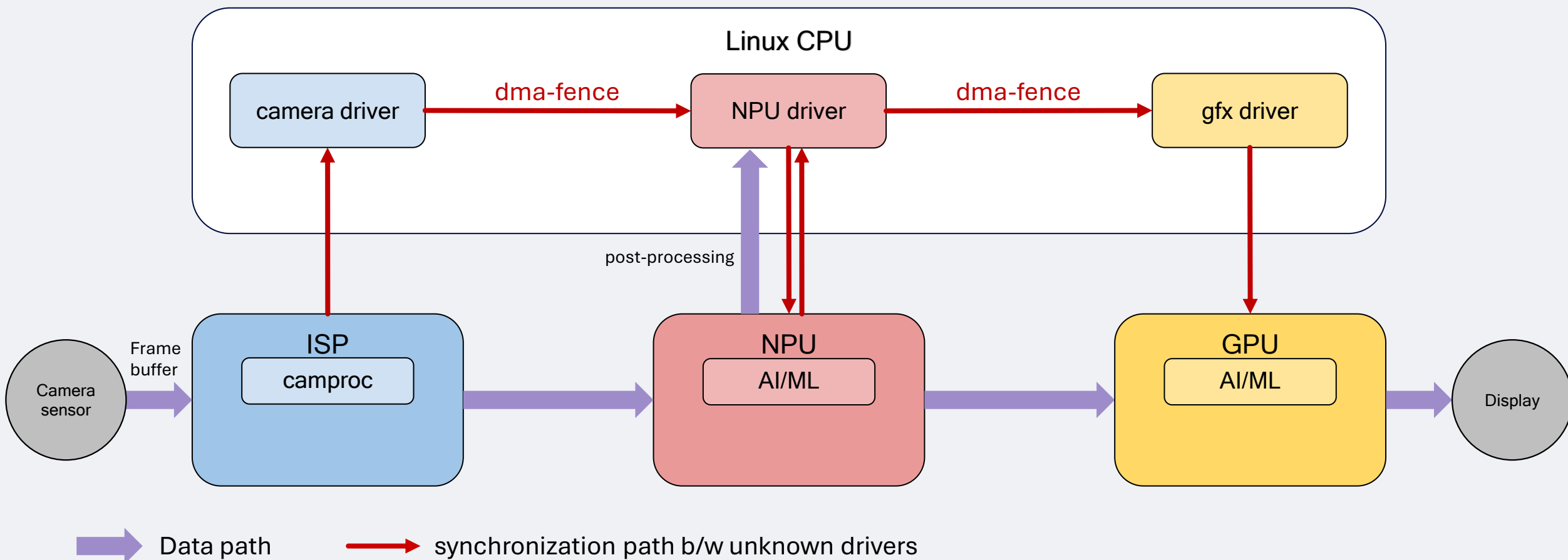
AI CAMERA



Heterogeneous use case example 1

AI CAMERA

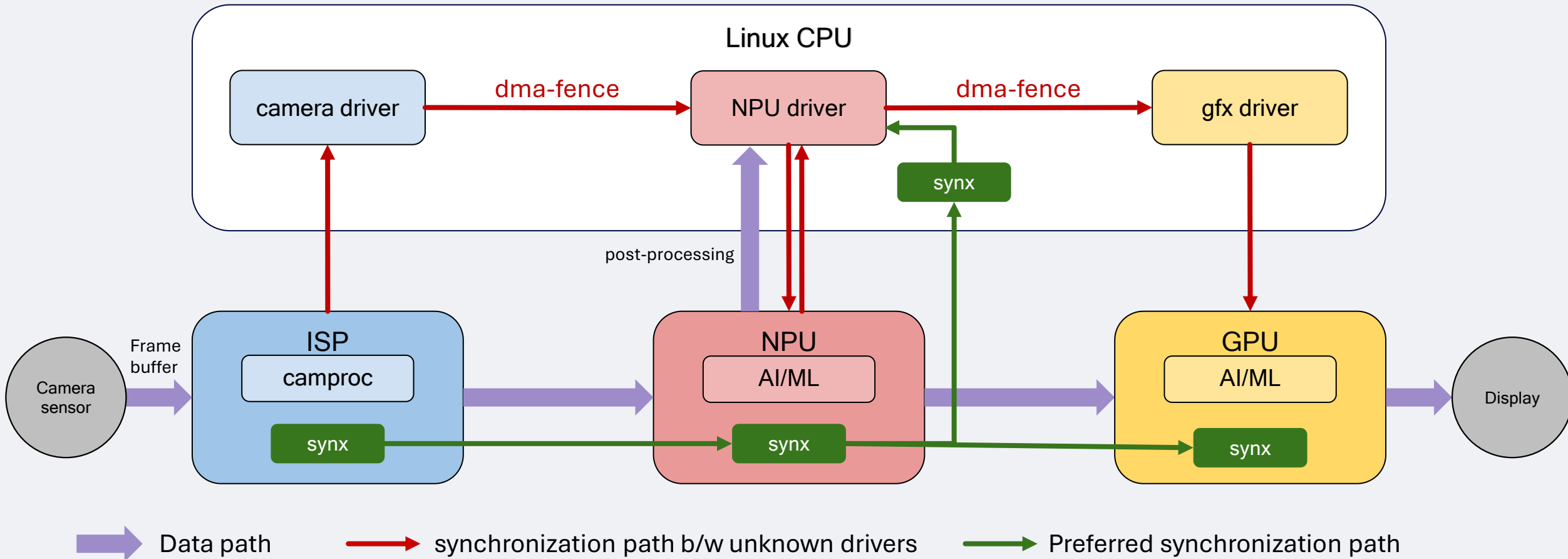
Cross-core signal happens through host Linux
Cost: an extra CPU wake-up + host round trip on every cross-core dependency, even when Linux has nothing to do with the work.



Heterogeneous use case example 1

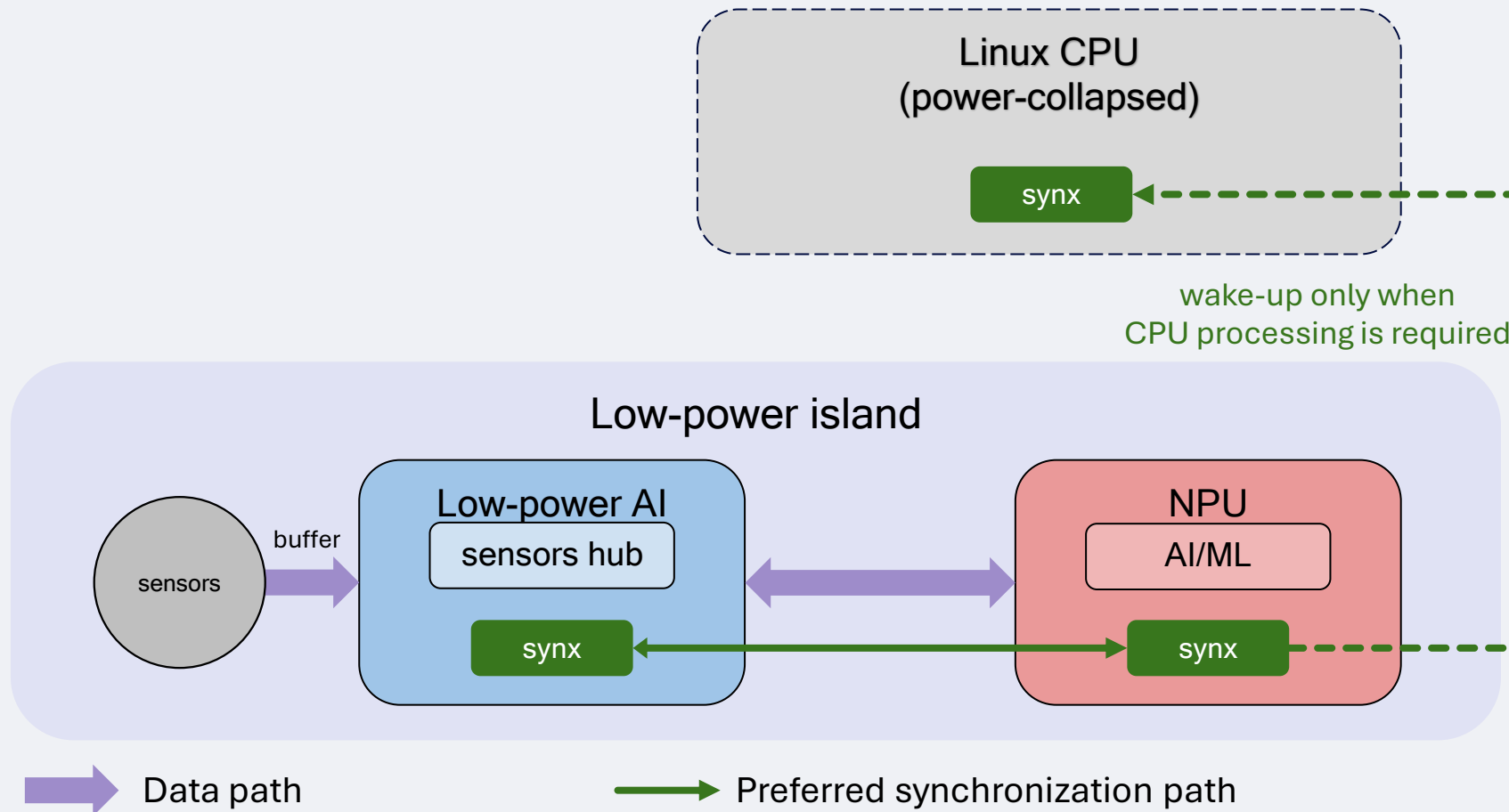
AI CAMERA

Cross-core signal happens through host Linux
Cost: an extra CPU wake-up + host round trip on every cross-core dependency, even when Linux has nothing to do with the work.



Heterogeneous use case example 2

AI IN LOW-POWER ISLAND MODE



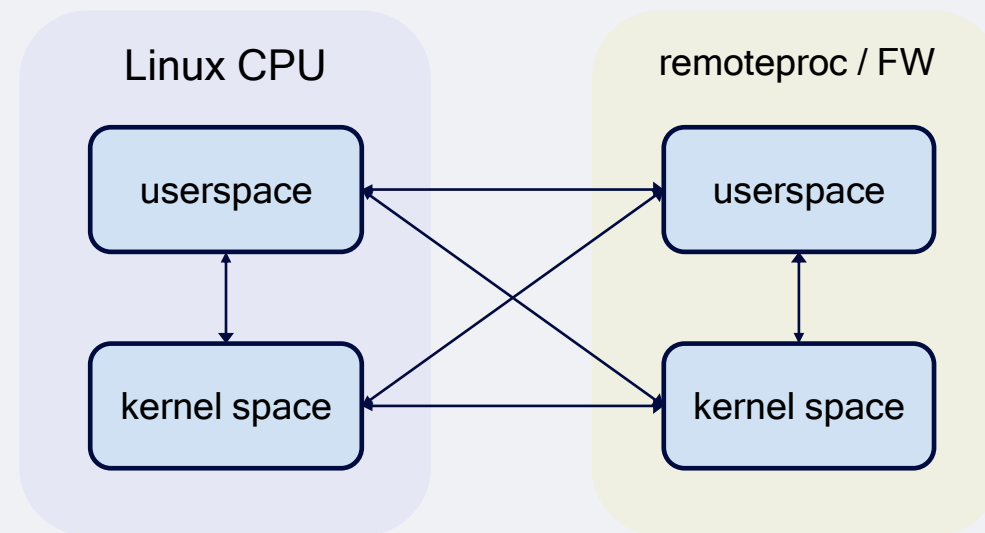
Synx API

CONSISTENT API ACROSS SUBSYSTEMS

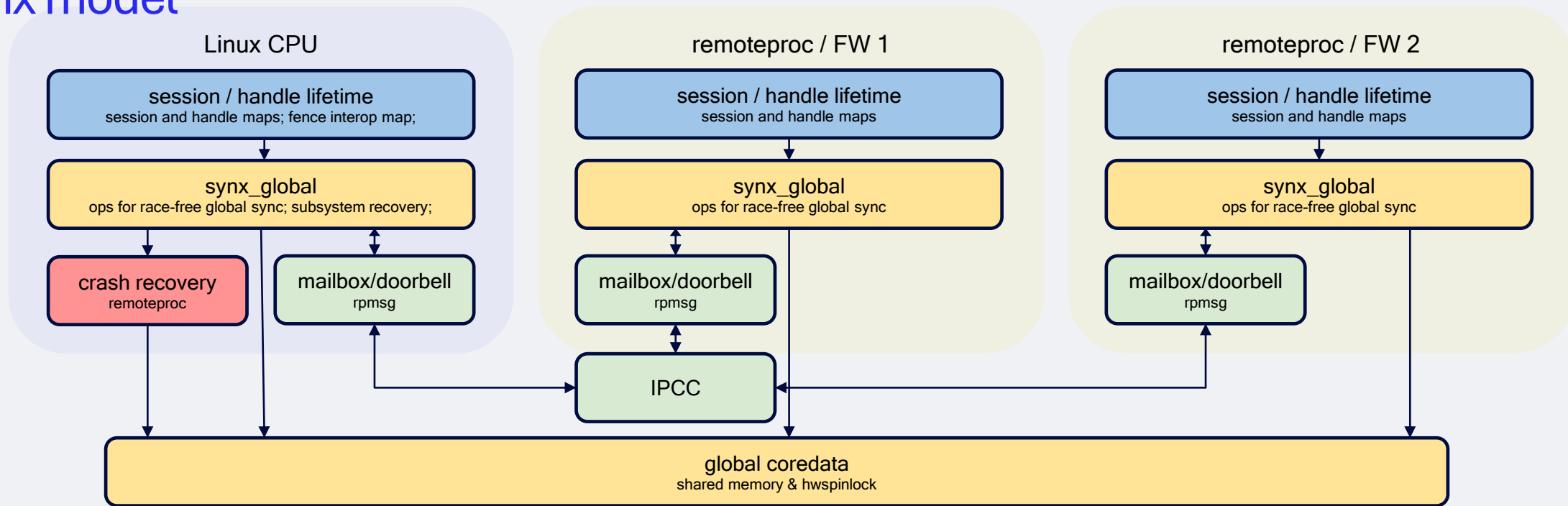
- Handle lifecycle management
- Reusable and non-reusable fences
- SoC-wide merge / composition
- Signal
- Blocking and non-blocking wait w/ optional timeout
- Subsystem crash / hang recovery (Linux-only)

Distributed signaling

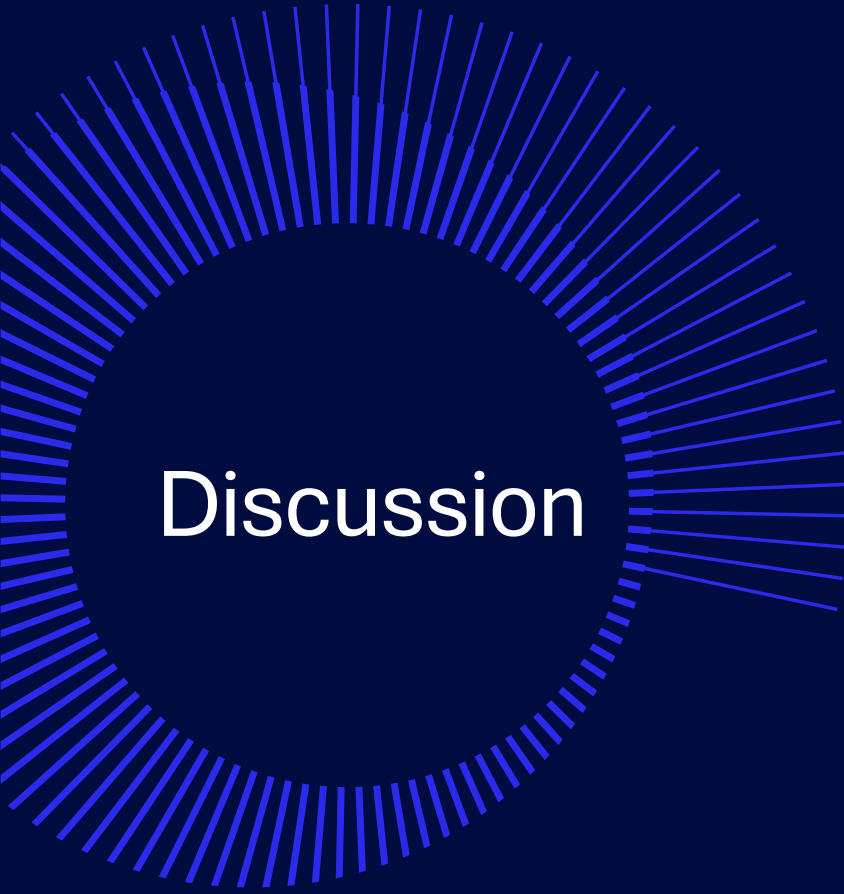
MESH NETWORK



Synx model



```
1 struct synx_global_coredata {
2     u32 handle;           // globally unique and meaningful handle
3     u16 refcount;         // each participating subsystem holds a refcount
4     u16 subscribers;     // bitmap representing participating subsystems
5     u16 waiters;          // bitmap representing waiting subsystems
6     u32 status;           // ACTIVE or SIGNED with a status
7     // allows composition of handles created across the SoC
8     u16 num_child;
9     u16 parents[MAX_PARENTS];
10    u64 security_key;      // secure the synx endpoints in untrusted spaces
11    u64 client_data;       // userdata; mostly used in reusable fencing to identify frames
12};
```



Discussion

- If you're a SoC vendor, what concerns do you face in your use case? What's your solution?
- a dma-fence extension
 - Supported drivers share direct-signaling mechanism (1) interrupt line (2) memory to poll etc. (3) error handling interface.
 - Linux-driven fencing. How to solve synchronization in island mode when CPU is powered down without subsystem-driven fences?

Thank you

Nothing in these materials is an offer to sell any of the components or devices referenced herein.

© Qualcomm Technologies, Inc. and/or its affiliated companies. All Rights Reserved.

Qualcomm and Snapdragon are trademarks or registered trademarks of Qualcomm Incorporated.
Other products and brand names may be trademarks or registered trademarks of their respective owners.

References in this presentation to “Qualcomm” may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable. Qualcomm Incorporated includes our licensing business, QTL, and the vast majority of our patent portfolio. Qualcomm Technologies, Inc., a subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of our engineering, research and development functions, and substantially all of our products and services businesses, including our QCT semiconductor business.

Snapdragon and Qualcomm branded products are products of Qualcomm Technologies, Inc. and/or its subsidiaries. Qualcomm patents are licensed by Qualcomm Incorporated.

Follow us on: [in](#) [X](#) [@](#) [▶](#) [f](#)

For more information, visit us at [qualcomm.com](https://www.qualcomm.com) & [qualcomm.com/blog](https://www.qualcomm.com/blog)

