Low-level IPC Topic

These are minutes from two meetings on this topic 2021-12-05 and 2021-12-13.

Preparation:

see also HV Project Deep-dive topics for 2021 June and forward

Purpose:

Discuss and share knowledge on the current landscape of low-level IPC (a.k.a. ICC / Inter-chip communication) on heterogeneous SoCs.
Discuss opportunities for improvement and alignment if that can reduce development effort / cost / risk for ECU development.

Agenda

- Presentation of one option for IPC (Cortex-R7 to Cortex-A5x cores) : **Thomas Bruss (Renesas)**
- Discussion of presentation
- Further discussion, driven by the questions belo

Minutes/Findings

- "It depends on your input requirements"
  - (Gunnar): Yes but... how many tradeoffs do we have. How many choices are required? Investigate this deeper...
- Ring-buffer vs. Shared Memory are two methods that were highlighted
  Thomas' presentation IVI focused?
  - Yes, one key described use case was trigger on VBLANK, transfer graphics
- IC-COM
  - Little information in the group
  - AUTOSAR choice?
  - Bosch provided open implementation
  - ...uncertain in the group if this is very relevant
- OpenAMP
  - Low latency optimized
  - For resource constrained e.g. MCUs
  - Few queues
  - Small shared memory kilobyte-range
  - => not for high bandwidth
  - RP message
- Thomas' presentation: RP message used a control channel, graphics transfer used a (shared mem) side channel
- Hardware features
  - Mailboxes, Dual-port memories
  - => Can be used but nowadays you can usually map large RAM areas as shared memory
  - Data exchange needs to be orchestrated if using shared memory hence protocols, like VIRTIO
- Gunnar: Are these shared-memory implementations considering write-protecting the buffer (e.g. Sealing feature in Linux)?
  - Stephen: Does this become limited to requiring Linux on both sides?
  - Gunnar: I don't know if other OSes implement a similar feature
- José: MiPi. Higher-level protocols exist. Similar to SOME/IP.
- Discussion question: "Is diversity/fragmentation of (low-level) IPC/ICC choice a challenge for efficient development?"
  - Stephen: Definitely
  - Dan: Generally Linux is enabled first on hardware. When you switch to RTOS you are lagging behind. We are working on combining Linux and RTOS by allowing RTOS to "reuse" certain services from Linux. It could include initializing the hardware, or higher-level services like filesystem access.
  - Gunnar: Does Linux drive diversity in the low-level IPC/ICC choice?
  - Dan: Usually the hardware brings the diversity
  - Gunnar: ... because SVs bring a favorite choice or the actual hardware features?
- José: Programming language differences also matter. Linux can hide the mechanism of communication. Tools create C++ and other high-level language but I prefer C. Frameworks like flatbuffer have C support but it is not as good as for other languages.
- Gunnar: Understood. I'm guessing on low-level IPC / ICC a lot of the implementations are in kernel-space for Linux, or in RTOS, so this means it is likely written in C (*quick mention: Rust coming(?) also in Linux kernel*)
- **Discussion question**: "What (if anything) is preventing simply agreeing on one (few) choice(s)?"
  - Stephen: Lack of discussion is one part.
- Bernhard: A lot of safety island focus currently. Implementation challenges may include limited address space (e.g. 32 bit) for some involved cores.
  - Bernhard: Further industry feedback needed.
- ... fiunal consensus seems to be that convergence towards one/few standards is worth doing.
- **Gunnar: Challenge to all: Figure out needed stakeholders to come together to make those agreements.**

Next week availability?

- Apologies: Stephen, Bernhard, Adam, at least.
- Conclusion: Restart meeting series next year, with this topic and other Deep Dive topics

Discussion questions:

Factors affecting choice

What factors affect the choice of IPC?
- Technology / optimization?
- Tier-1 preference (previous experience)?
- Silicon Vendor preference?
- Why is a certain choice preferred?

Are there any standard ways that can be agreed upon?
Is IC-COM a standard? AUTOSAR preferred choice?
Are there others?
Does AUTOSAR have specific choices, and how does that apply on non-AUTOSAR systems?

Is diversity/fragmentation of IPC choice a challenge for efficient development?

What (if anything) is preventing simply agreeing on one choice?

Detailed operation

What and how does it work
e.g. shared memory, buses (SPI..), PCI, ?
Development effort: the driver must be implemented

Relationship to (other parts of) SW stack?

Flexibility (available implementation) towards upper and lower protocols (think OSI-stack).
Are higher-level protocols layered on top of the simple communication, or not? How does the choice of low-level IPC affect that?
In relation to connecting downward to hardware, what are the implementation challenges?

Required Hardware Features

Does it depend on particular hardware features (hardware-supported mailboxes, dual-port memories, com. buses,...?)
(E.g. if you choose A, then you have simple integration of technologies 1, 2, and 3, but for choice B available integrations are more limited)

CPU architecture differences?

- RISC-V, Arm, SH, ...

Applicability

- Communication between different cores on an SoC.
- Inter-ECU? Between SoCs / CPUs.
  E.g. shared PCI network,
  Full network (Ethernet)