POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

Tracing embedded heterogeneous systems

PROGRESS REPORT MEETING, DECEMBER 2015

THOMAS BERTAULD

DIRECTED BY MICHEL DAGENAIS

Presentation plan

Introduction -

Why tracing heterogeneous embedded systems ?

- Systems designed for specific needs/tasks
- Often used for real-time applications like signal processing
- Can be used anywhere
- Power-efficient

Introduction -

Goals

- Heterogeneous systems imply different kinds of processors
  - Not all of them are directly tracable
  - DSPs and generic calculation processors are not meant to run any OS (*bare-metal*)
- How do the different processing units interact ?
  - Use-case 1 : a classical CPU sends work to a DSP and retrieves the results
  - Use-case 2 : some classical processor monitors a bare-metal processor activity

The Parallella board -

Specifications

- ARM Cortex A9 running Linux
- 1 GB DDR3
- Epiphany chip :
  - 16 processors mesh (*eCores*)
  - 32-bits RISC CPU
  - 512KB distributed on-chip shared memory (32KB per CPU)
  - Superscalar architecture (internal pipeline)
  - 2 32-bits events counters
- 32MB external shared memory
- http://parallella.org/

The Parallella board -

Benefits and drawbacks

- Cheap (~100$ per board)
- Open-source design
- Can be used inside a cluster
- Power-efficient
- 16 multi-purposes CPU…
- … but no optimization for signal processing
- Only 32KB of local memory per core
- One way interrupt mechanism
- Only partial support of nested interrupts
- LibC not fully integrated (no *malloc*…)
- Lack of community support and documentation
- Hard to debug

The Parallella board -

Benchmarks

- *eCore* -> EDRAM 1 byte read: ~**540** cycles
- *eCore* -> EDRAM 1 byte write: **17** cycles
- Interrupt handling: ~**300 000** cycles
- Internal synchronization between 16 *eCores*: ~**200** cycles

BareCTF -

Tracing bare-metal systems

- Python tool created by Philippe Proulx (EfficiOS)
- Targets bare-metal systems
- Generates CTF traces
- Easy-to-use (configuration by YAML files)
- Lightweight
- https://github.com/efficios/barectf

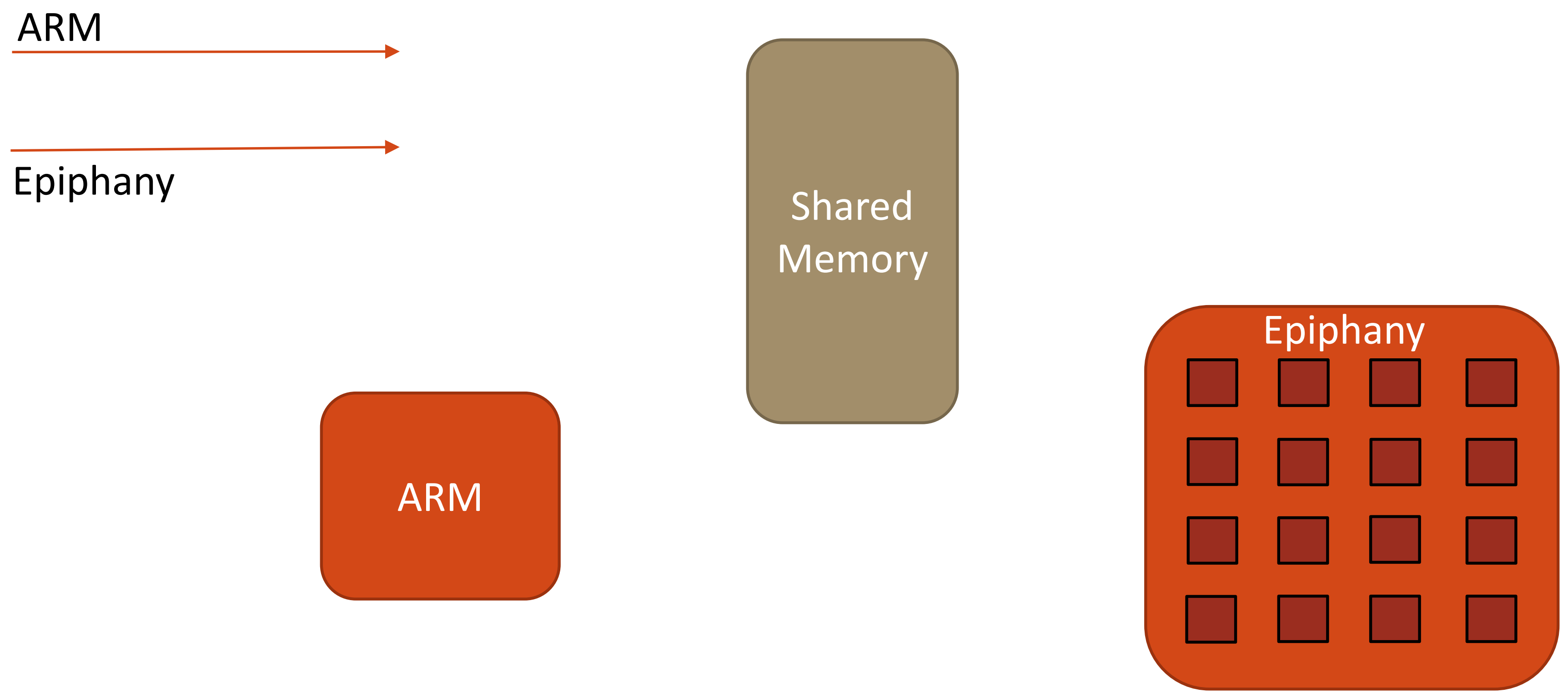
The synchronization process -

Facts and goal

- We can use LTTng to trace the ARM side of the Parallella
- We can use BareCTF to trace the Epiphany side of the Parallella
- A hardware barrier can be used for internal synchronization
- All *eCores* share the same clock rate
- How do we synchronize and correlate ARM and Epiphany traces ?

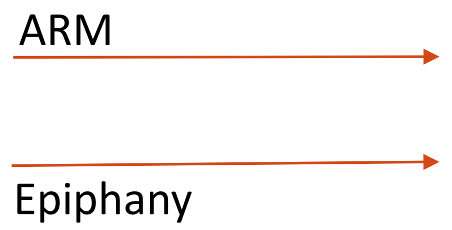
The synchronization process -

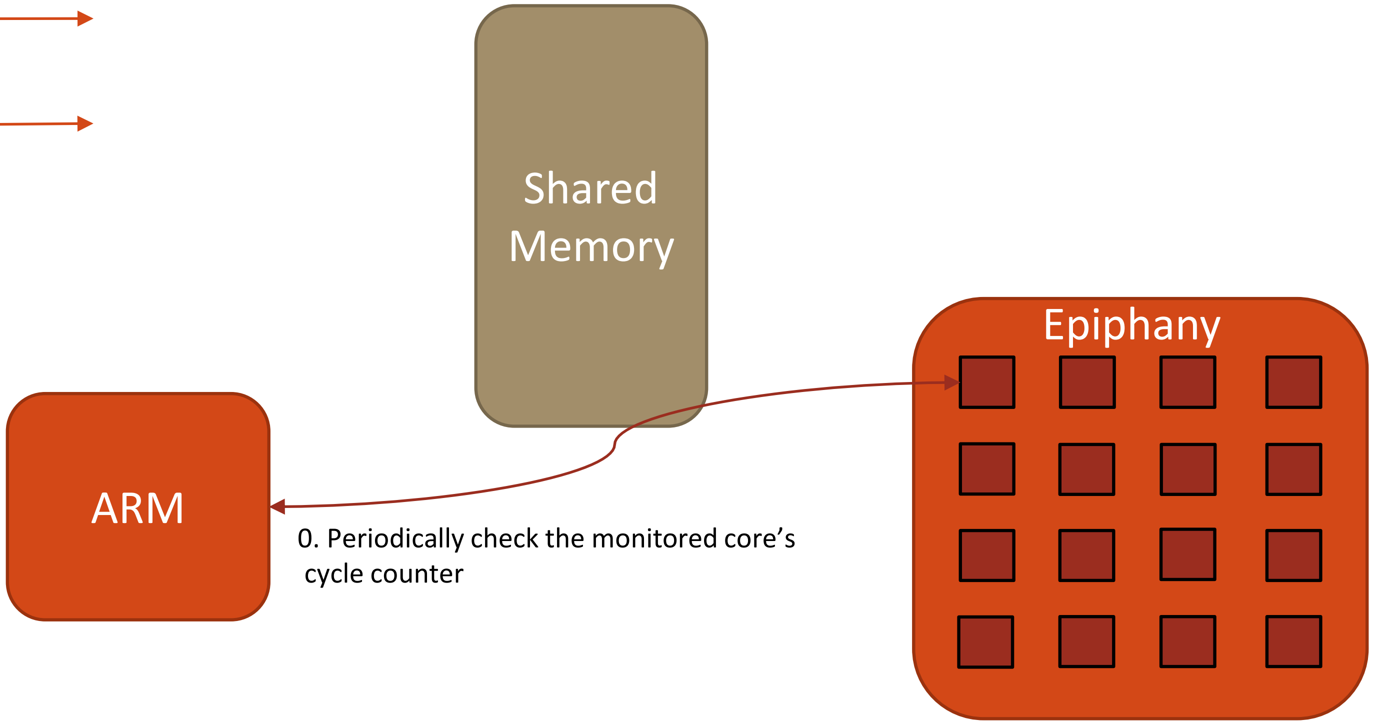
Description



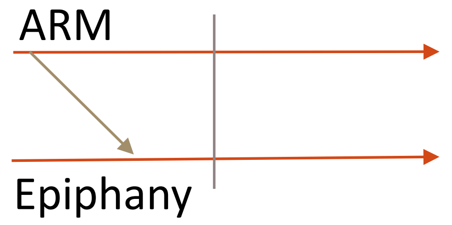
The synchronization process -

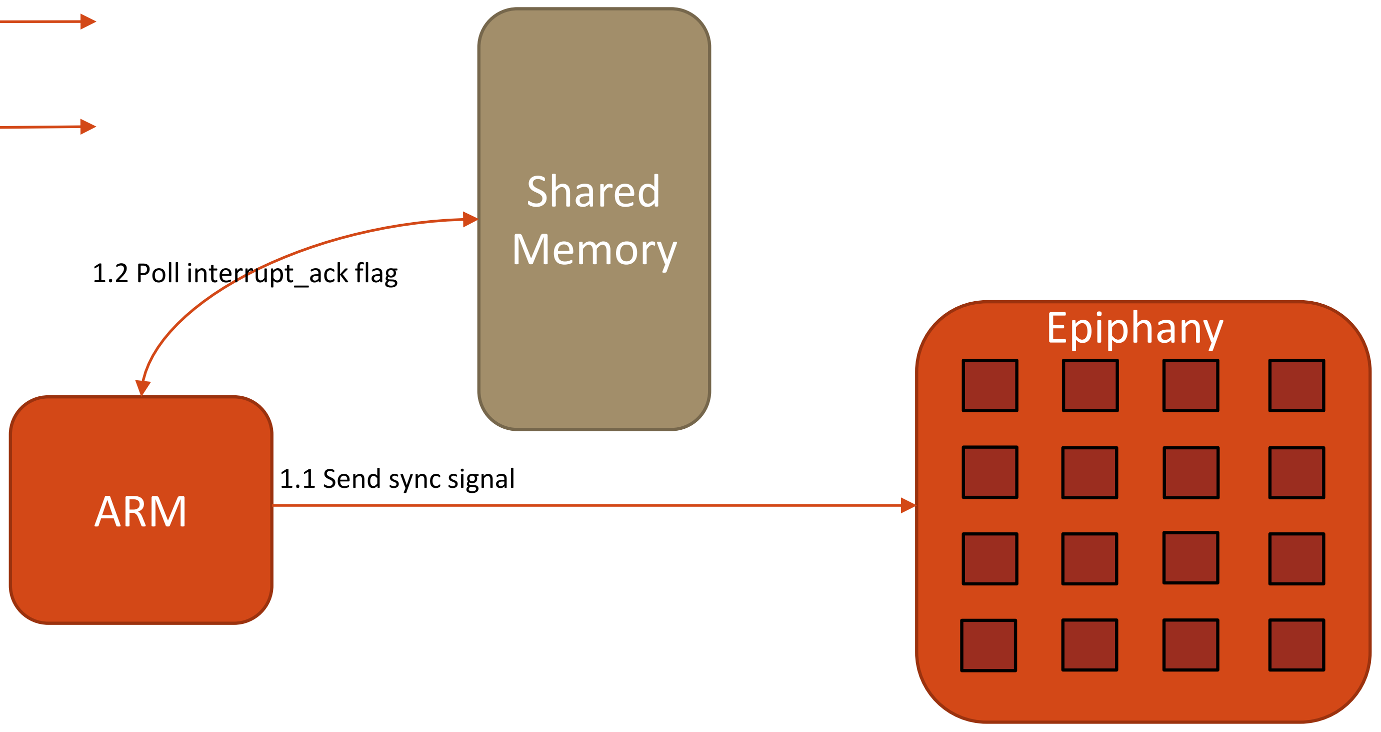
Description



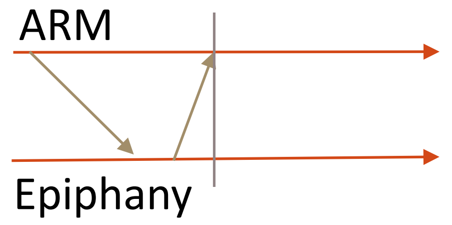


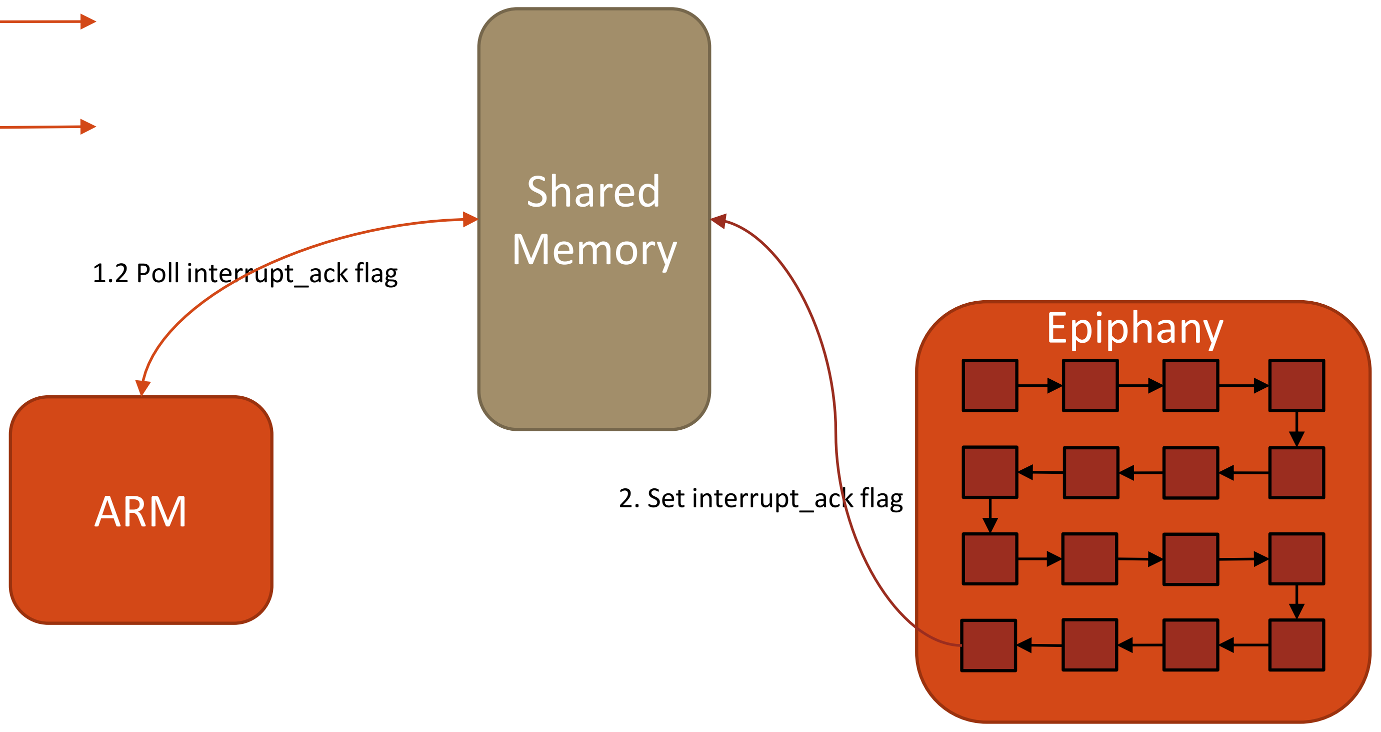
The synchronization process - Description





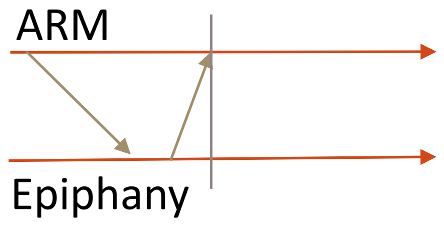
The synchronization process -
Description

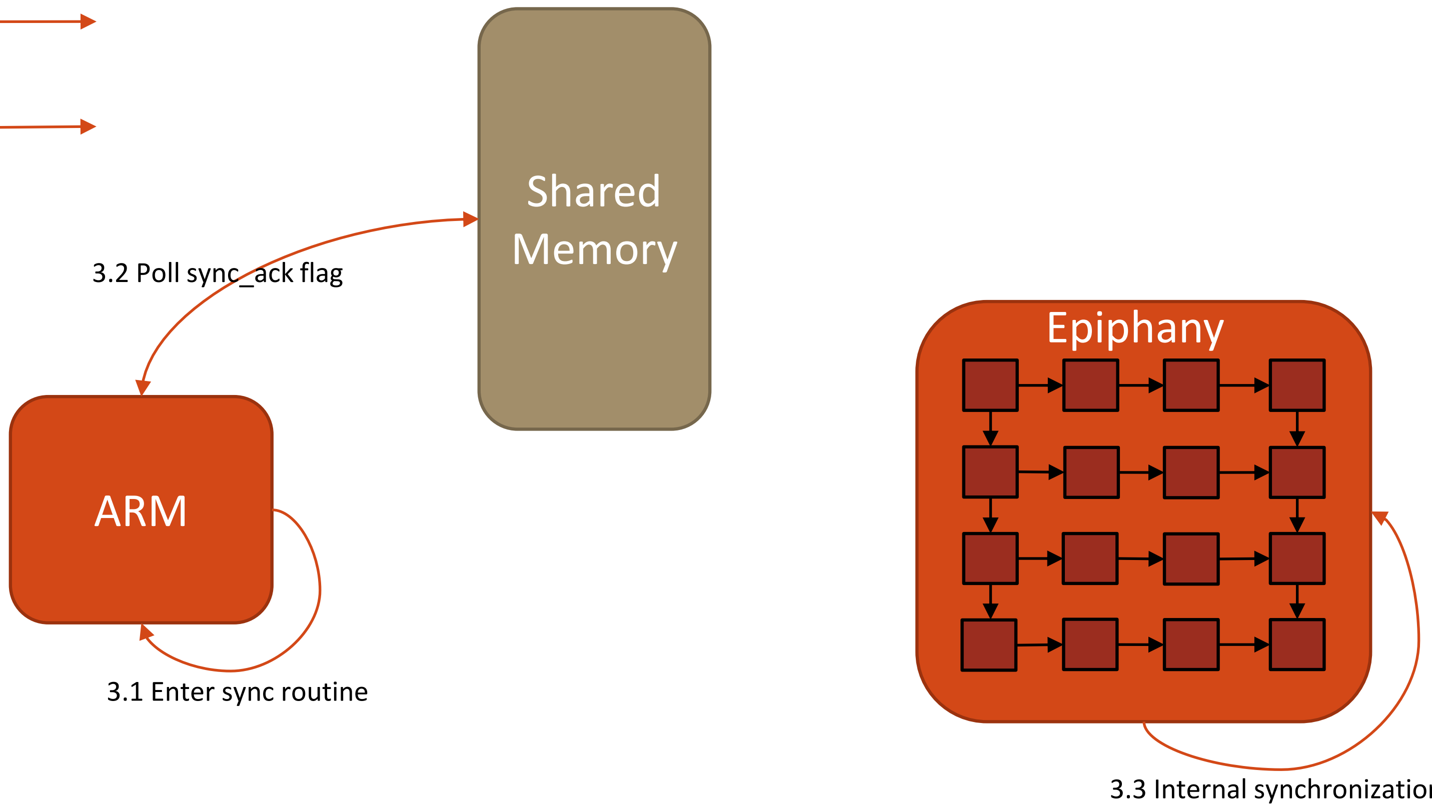




The synchronization process -

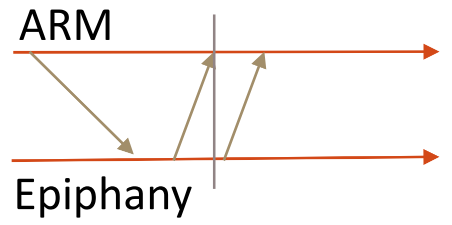
Description

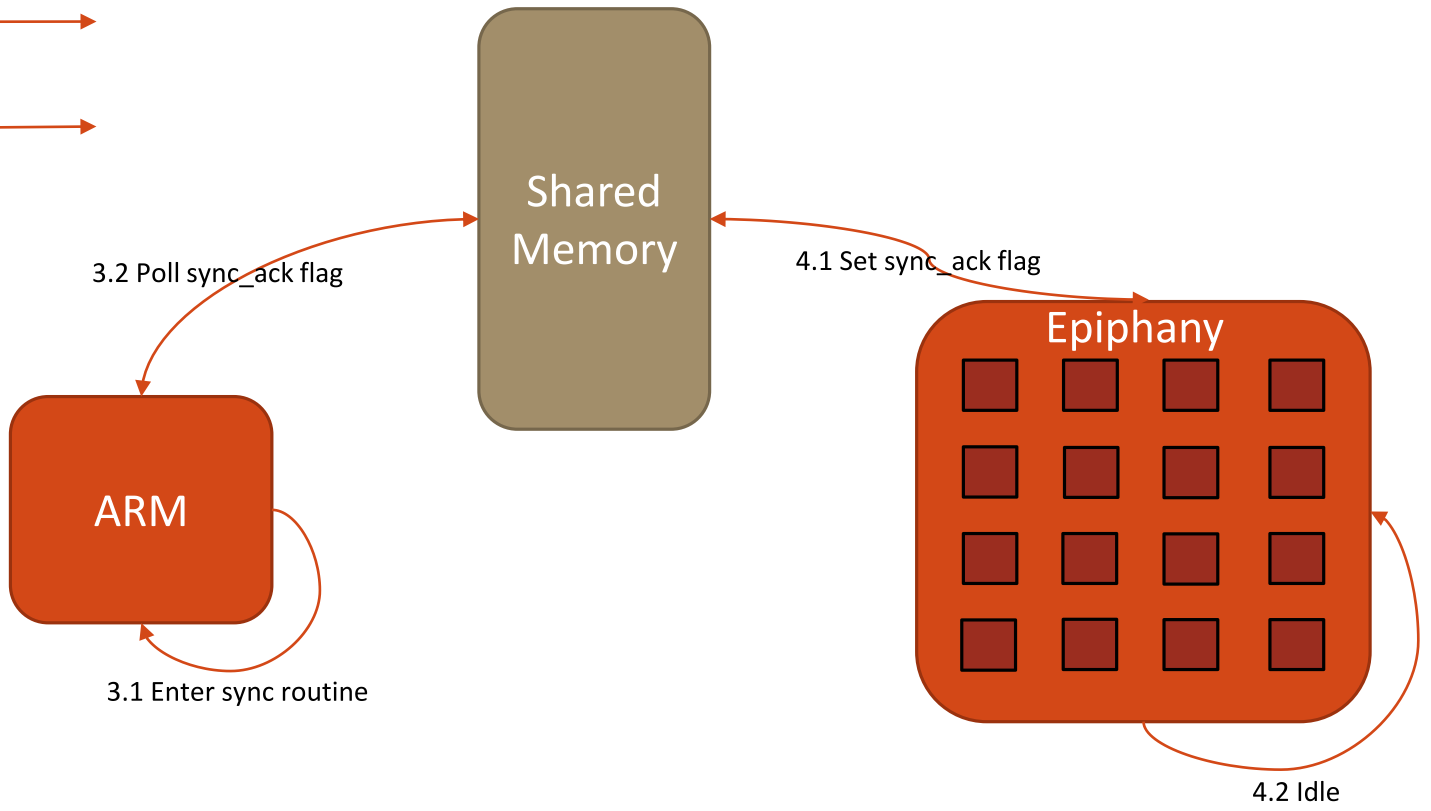




The synchronization process -

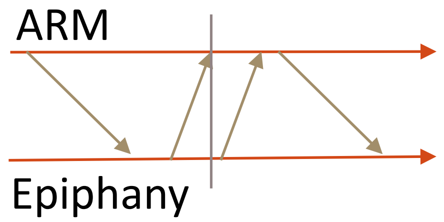
Description

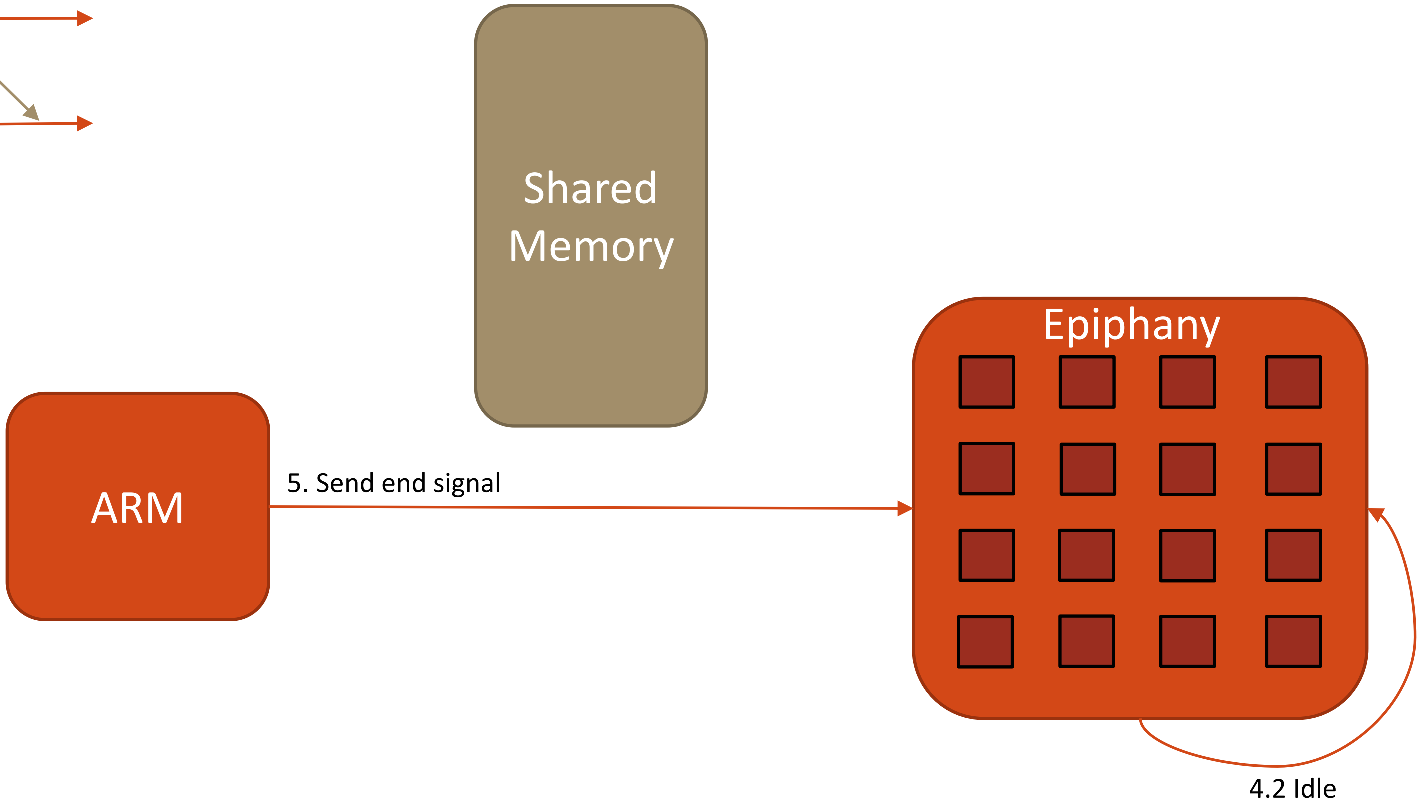




The synchronization process -

Description





Results -

General

- Synchronization is working
  - Takes an average **0,1 ms** to complete on host
- Partial integration with bareCTF
- It is possible to allow nested interrupts of the same kind
  - Need to overwrite a system register
  - Not very safe
- Another method using only one interruption has been tested
  - Average cycles taken by first method: ~**300 000** (without first interruption)
  - Average cycles taken by second method: ~**a few thousand**
  - Drawback: poll shared memory instead of idle and less precise synchronization

Results -

Limitations

- BareCTF + synchronization code on a core
- One core is periodically checked
- Need to use shared memory
- Huge overhead due to interruptions
- Internal synchronization can’t be shared by different workgroups

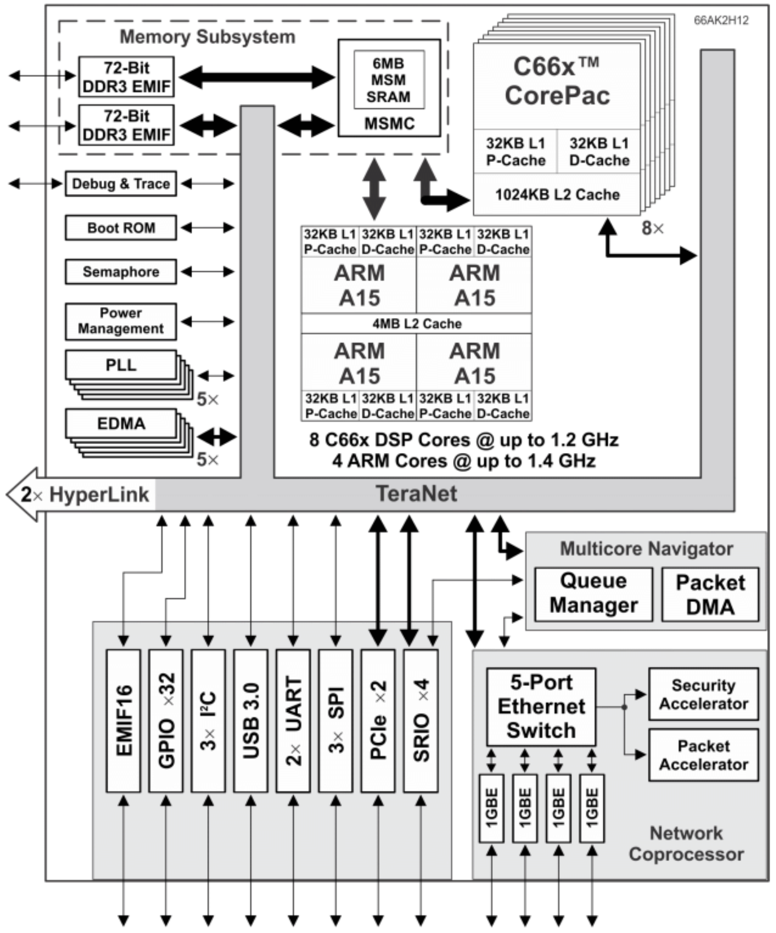
Future work

- Try the DMA engine of the Parallella
- Create a TraceCompass view
- Finalize bareCTF integration
- See how this method can be integrated to other devices
- Estimate the optimal synchronization rate

Future work -

The TI board

- 4 ARM Cortex A15
- 8 C66x DSPs
- 6 MB shared memory
- Cache system



Thank you for your attention !

Contact : thomas.bertauld@gmail.com