Zephyr Project
Developer Summit

# Multi-domain Logging

Krzysztof Chruściński

## Ordered, human-readable logs from multiple domains on a single output

with low CPUs load

## Agenda

- Assumption, Preconditions, Challenges
- Single domain logging recap
- Multi-domain architecture
- Log message
  - Identification
  - Formatting
  - Ordering
- Potential improvements
- Other options

Krzysztof Chruściński

- Contributing to Zephyr since 2018
- Nordic Semiconductor
- Logging subsystem maintainer
- Other areas: shell, drivers, kernel

## Assumptions & Preconditions

- Domain = binary product (e.g., ARM TrustZone – 2 domains on a single CPU)
- Logging available from any context, ideally non-intrusive
- String formatting
  - Time consuming
  - Depends on string length and type and number of arguments
- Sending data over backend
  - Time consuming
  - Limited to thread context?
- Domain memory access limitations

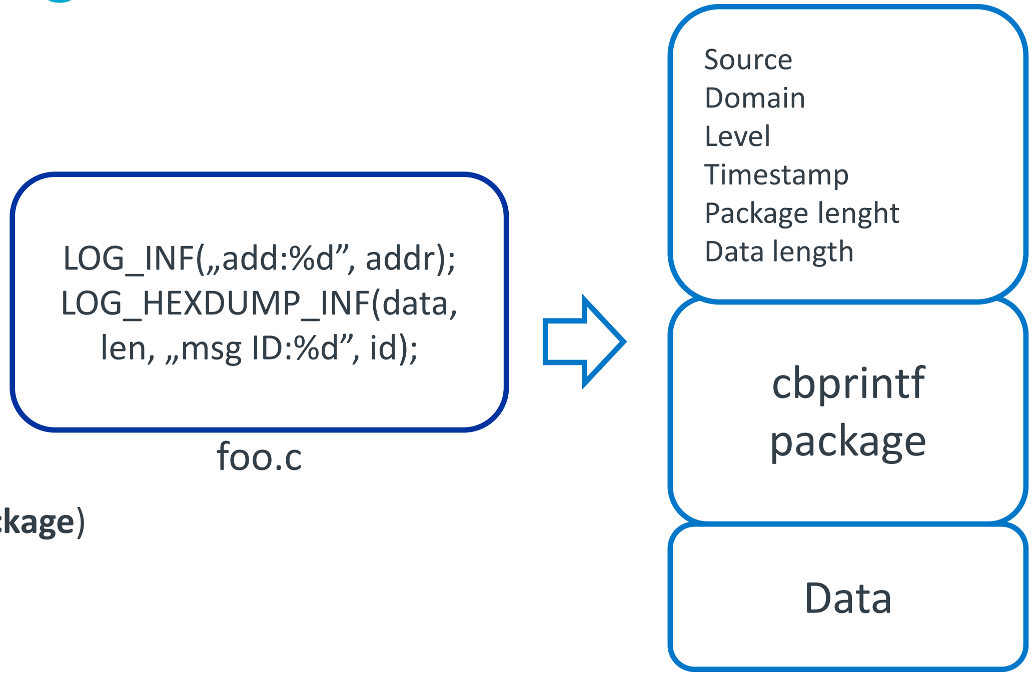
**LOG_INF**(„Johann Gambolputty de von Ausfern-schplenden-schlitter-crasscrenbon-fried-digger-dingle-dangle-dongle-dungle-burstein-von-knacker-thrasher-apple-banger-horowitz-ticolensic-grander-knotty-spelltinkle-grandlich-grumblemeyer-spelterwasser-kurstlich-himbleeisen-bahnwagen-gutenabend-bitte-ein-nürnburger-bratwustle-gerspurten-mitzweimache-luber-hundsfut-gumberaber-shönendanker-kalbsfleisch-mittler-aucher von Hautkopft of %s”, „Ulm”)

## Challenges

- Performance
  - Processing and IPC communication
- Synchronization
  - Common timestamp source
- Ordering
  - by time of generation
- User friendliness
  - Same look&feel as for single domain

## Logging recap – message content

- Who?
  - Source ID, Domain ID
- When?
  - Timestamp
- What?
  - Level
  - *String with arguments (**cbprintf package**)
  - *Data (for hexdump messages)



```
LOG_INF(„add:%d”, addr);
LOG_HEXDUMP_INF(data,
len, „msg ID:%d”, id);
```

## Cbprintf package – formatted string snapshot

- Printflike(„foo %s %d”, str, val)
  - „foo %s” – Variable in RO data. Pointer.
  - str – variable on stack. Pointer
  - val - Value
- Statically create using C11 _Generic keyword
  - Format string (*fmt*) not touched

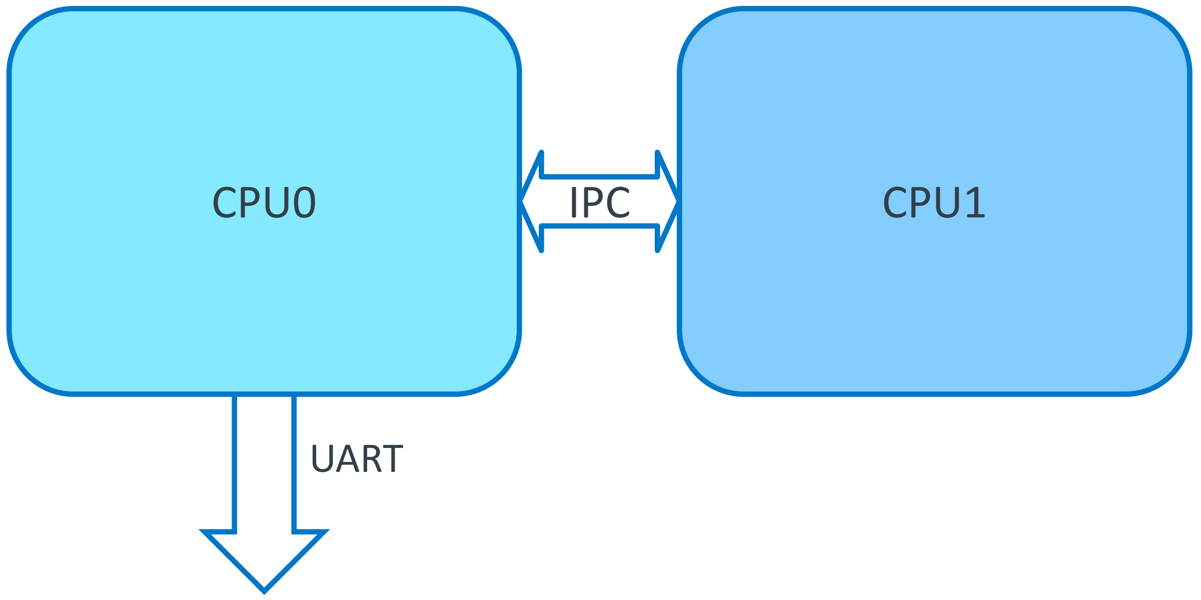
```
#define FOO(x) _Generic(x,      \
        int: printk(„int”),         \
        default: printk(„else”))
```

| |
|---|
| **Header: va_list size, number of attached strings** |
| fmt |
| *va_list frame |
| *RW strings prefixed with location index |

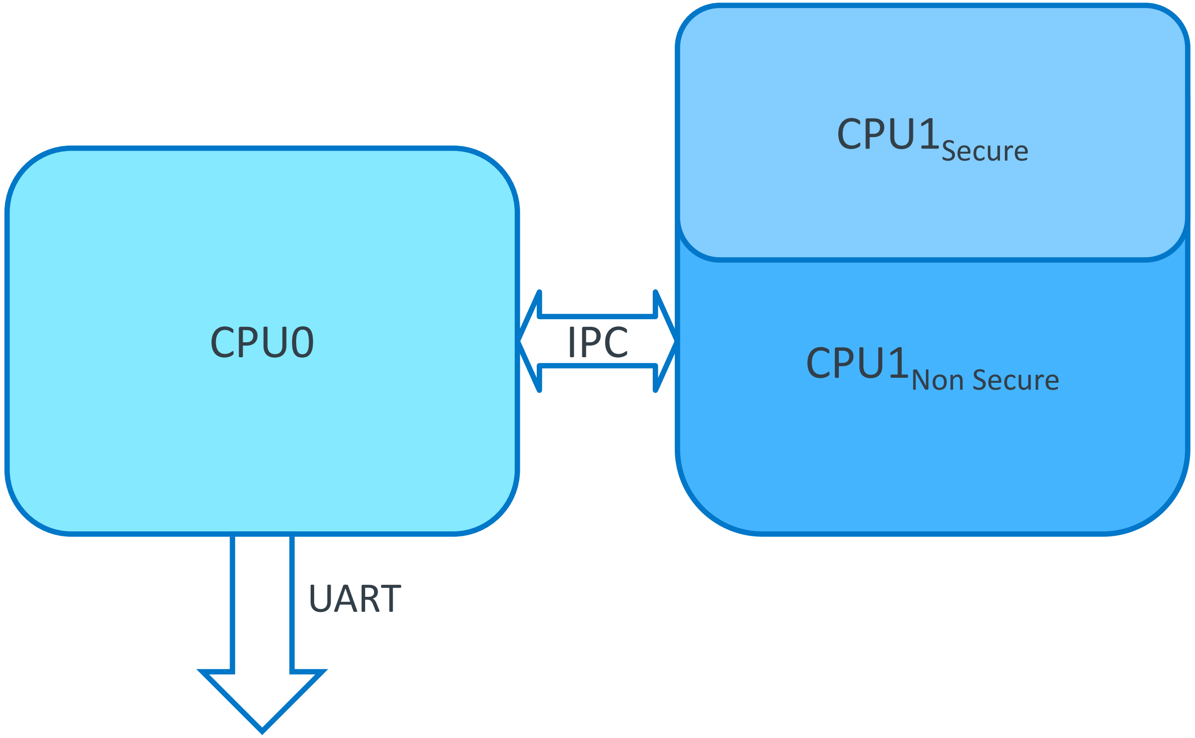
## Deferred logging in three stages

- Message creation
  - Any context
  - Create message with timestamp, allocate and commit
- Message processing
  - Known, low priority context
  - No real time requirements
  - Get message and pass it to all* enabled backends
- Backend processing
  - *Converting to human-readable string
  - Transferring (UART, flash, etc.)

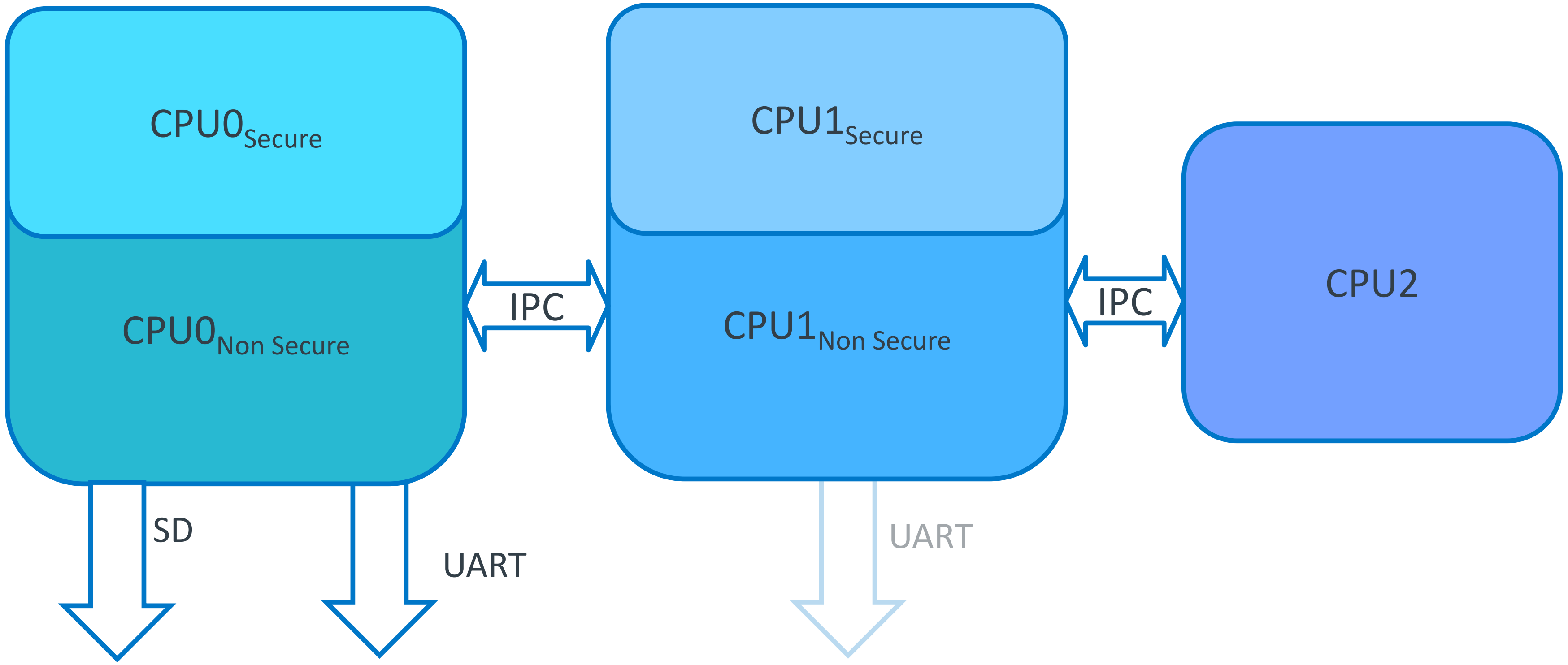
## Multi-domain architecture



## Multi-domain architecture



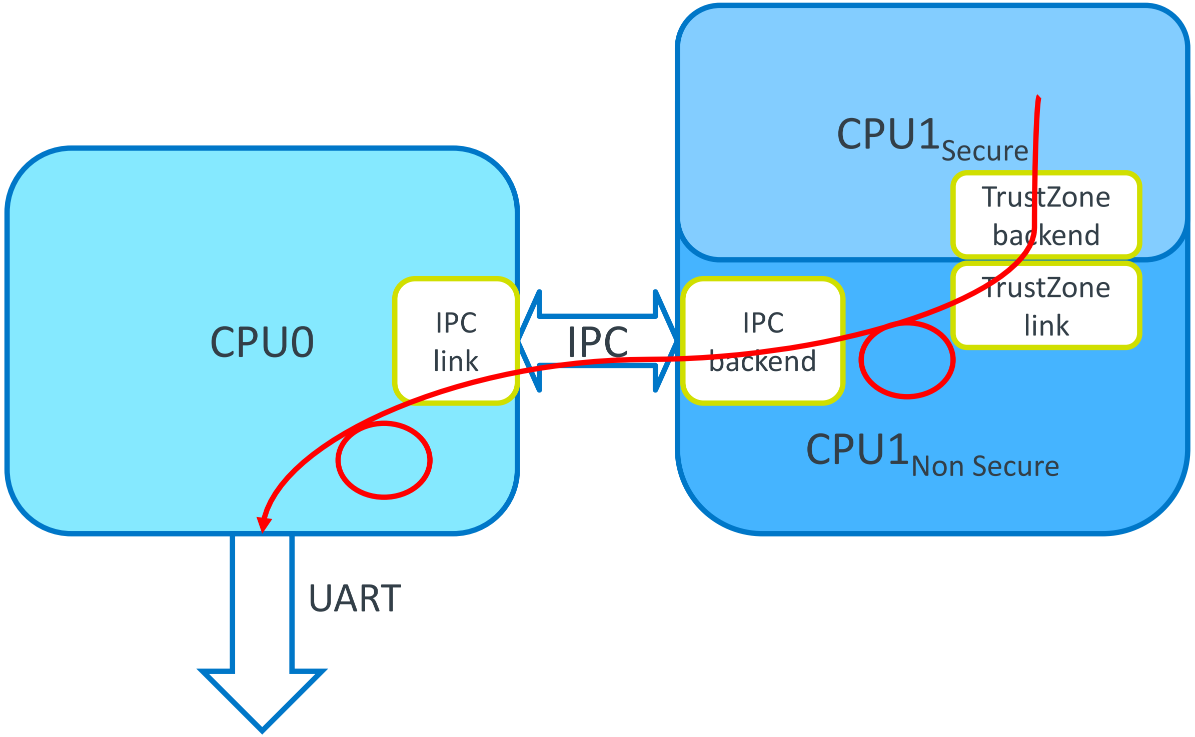
## Multi-domain architecture



## Logging Link

- Pairs with log backend
- Logging backend passes the message
- Link receives message from backend and enqueue locally

# Multi-domain architecture



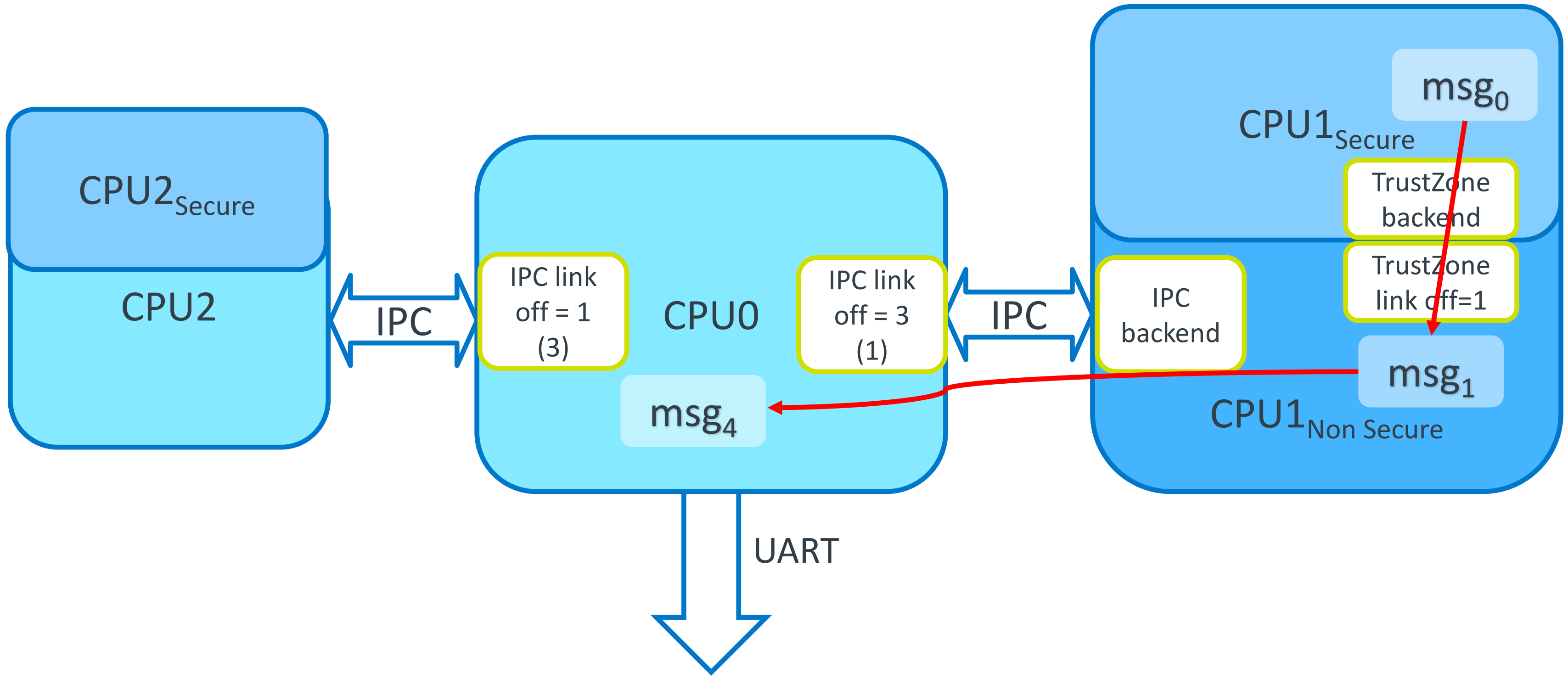
## DomainID addressing

- Static addressing
  - Domain ID assigned in Kconfig
  - More maintenance
  - Troublesome on-chip operations
- Dynamic addressing
  - Assigned during initialization
  - Less configuration maintenance but longer initialization
  - Link API for getting number of domains
- Dynamic approach used

## Dynamic domainID

- Initialization
  - For each link get number of domains
  - $\text{Link}_0$ offset = 1
  - $\text{Link}_N$ offset = $\text{Link}_{N-1}$ offset + $\text{Link}_{N-1}$ domain count
- New message in link
  - Add link offset to the domainID
- Getting information from domainID
  - Decrement link offset before the call

## domainID


CPU2$_{Secure}$
CPU2
IPC
IPC link off = 1 (3)
CPU0
IPC link off = 3 (1)
msg$_4$
IPC
UART
msg$_0$
CPU1$_{Secure}$
TrustZone backend
IPC backend
TrustZone link off=1
msg$_1$
CPU1$_{Non Secure}$

## Message formatting

- Adding prefixes:
  - Timestamp
  - Domain name
  - Source name !
  - Level
- String formatting
  - From cbprintf package !

## Getting source name

- Get_source_name(domain_id, source_id)
- How to reduce IPC communication?
- Cache
  - Dedicate memory
  - Store n most recently used domain_id, source_id
  - When entry not found in cache, evict the oldest and get new from link
- Alternative (not implemented)
  - Get number of all sources
  - Dynamically allocate space and fetch the names

# Fully self-contained package

### Transient

Size of va_list

fmt

va_list

### Self-contained

Size of va_list

Number of rw strings

fmt

va_list

RW strings (with locations)

### Fully self-contained

No deps!

Size of va_list

Number of rw+ro strings

fmt

va_list

Strings (with locations)

## Fully self-contained package

### Transient

static

Size of va_list

fmt

va_list

Locations of rw strings

Locations of ro strings

### Self-contained

Size of va_list

Number of rw strings

fmt

va_list

rw strings (with locations)

Locations of ro strings

### Fully self-contained

Size of va_list

Number of rw+ro strings

fmt

va_list

Strings (with locations)

## Conversion

- Conversion is simple:
  - Read address under location
  - Append string under that location to the package
- LOG_INF(„lorem %s %s”, „ipsum”, p_char)
  - Statically created transient package
  - In the context of log message converted to self-contained
  - In the IPC backend converted to fully self-contained

## Multi-domain logging

- Message received from the link
  - Is self contained
  - Source can be retrieved from cache or remote domain
  - Timestamp is in sync
  - But ...
  - Messages will be processed in the order of arrival to the final destination

## Ordering

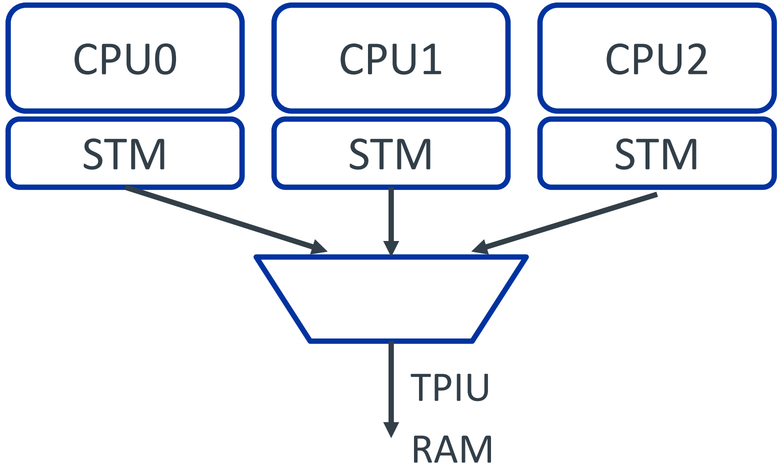
- Message buffer
  - Circular
  - Messages can only be processed in the order of arrival
  - Solution?
- Buffer for each link (+ local)
  - Peek each buffer – process the oldest message
  - At RAM cost
  - Still unordered?
  - Delay processing by the maximum latency

## PR & Potential improvements

- *PR: #43797*
- *Fmt/source name* in the memory section accessible by the final destination
  - Not accessed anywhere else (if no formatting backends used in origin domain)
  - Fmt is redundant in the package
  - Requires linker tricks

## ARM Coresight - HW solution example

- For each core set of memory mapped register interfaces (STM System Trace Macrocell)
- Writing to registers generates stream of data
- Merged stream
  - Stored in RAM (ETR)
  - Send over the debug interface (TPIU)



# Thank you

Questions?