

Distributed Sequencing for Resource Sharing in Multi-Applicative Heterogeneous NoC Platforms

Yvain Thonnart, Romain Lemaire, Fabien Clermidy

CEA – LETI – Minatec, Grenoble, France

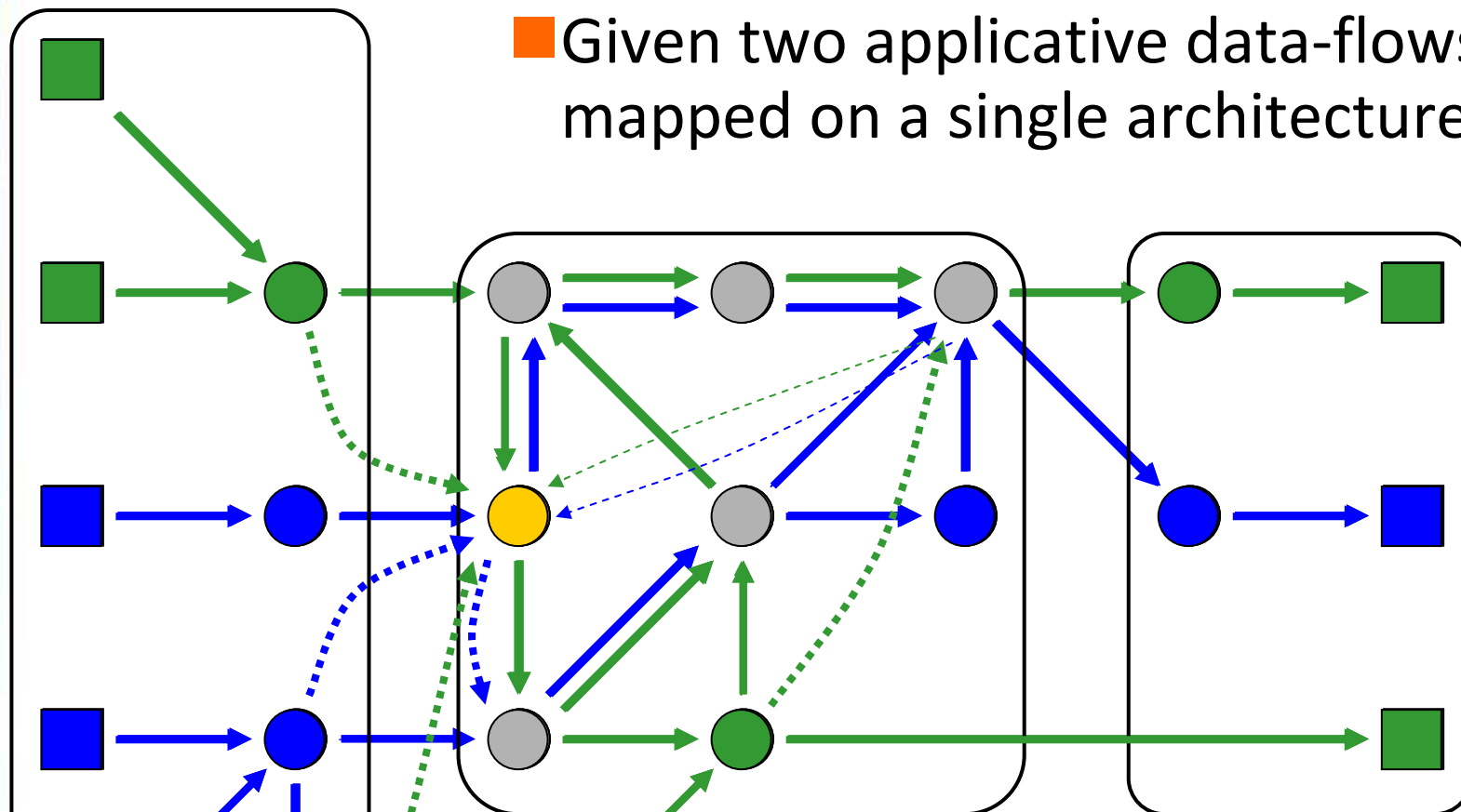
{firstname.lastname}@cea.fr

IEEE/ACM international symposium on Networks on Chip

May 6th 2010



On merging multiple applications in a single NoC



■ Given two applicative data-flows mapped on a single architecture

➔ Define **synchronizations between resources** allowing to interleave sessions on multiple tasks

Outline

■ Motivations

- User need
- Increase in resource utilization

■ Concepts

- Coarse grain segmenting
- Local synchronization

■ Implementation

- Sequencing protocol in network interfaces
- Micro-architecture

■ Results

- Area overhead
- Case study
- Utilization improvement & associated savings

■ Conclusion

User need for multi-application

■ Functionality Integration

- Technology allows for more in a single chip
- Users want even more functionality in handheld devices

■ Independent parallel processings

- Telecom baseband
- Audio decoding
- Video decoding
- ...

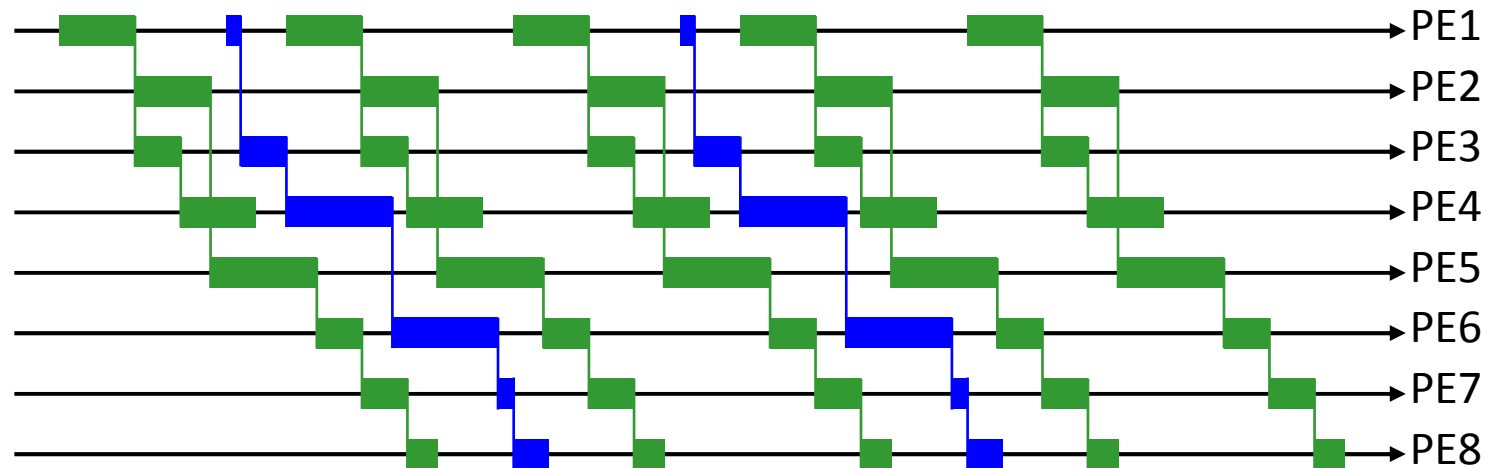
■ Seamless transitions

- Environment changes
 - ◆ Handover between base stations when changing zones
 - ◆ Handover between protocols
 - When environment becomes too noisy
 - When improved service becomes available

Real-time signal processing case study

Real-time deadlines to meet

- Long processing pipelines through dedicated operators
- Forward latency constrained
- ➔ High peak processing power needed
- ➔ Low utilization, with long idle phases



➔ Potential utilization increase with simultaneous multiple applications

Streamed Processing Elements

■ Credit-based streaming model

- Data pulled by receivers when they can absorb them
- Credit packets from receivers to senders notify the amount of data that can be transferred

■ Functional Processing Elements (PEs)

- Perform computation on the data streams
- Most work in tight-flow: no need for large buffering
 - ◆ sequence of PEs often as a composition of functions (F o G o H ...)

■ Memory buffers (MBs)

- Streamed accesses to large buffers
- May participate to computation, or only buffering
 - ◆ Complex reshufflings, Transposition of the stream
- programmable multiple writers / multiple readers

Multi-Task Sequencing: A meta-Programming Model

■ Based on task graphs

- Each node is an elementary task
- Each vertex is a communication between tasks
- Task graph is mapped on the NoC, with each node mapped on a resource

■ Proposal: merging of task graphs

- Relies only on **graph layout, not on programming model**
- Underlying execution semantics can be different
 - ◆ Synchronous Dataflow
 - ◆ Dynamic Dataflow
 - ◆ Kahn process networks...
- Requirement is that **consistent phases** can be identified
 - ◆ Needed for task segmenting

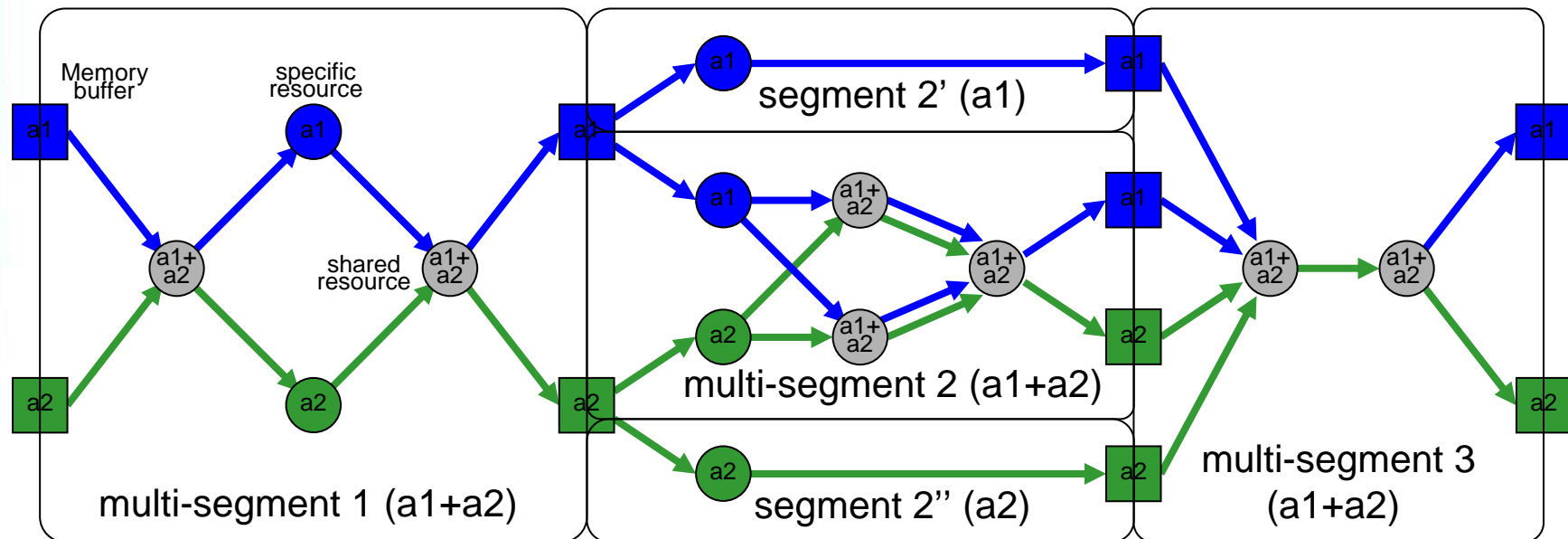
Task-Parallelization : Segments and Sessions

- **Segments:** Processing on different tasks is distributed spatially by system sectioning
 - Resources communicating together need to synchronize on a same task
 - Several tasks may run simultaneously on unrelated resources

- **Sessions:** Processing on different tasks is distributed temporally by task sectioning
 - Resources open sessions (part of processing) on a given task according to data availability at the inputs
 - Sessions are coordinated across resources of a same segment following the data-flow
 - Additional session requests are included in the NoC protocol to allow the synchronization between resources

Application Graph merging and segmenting

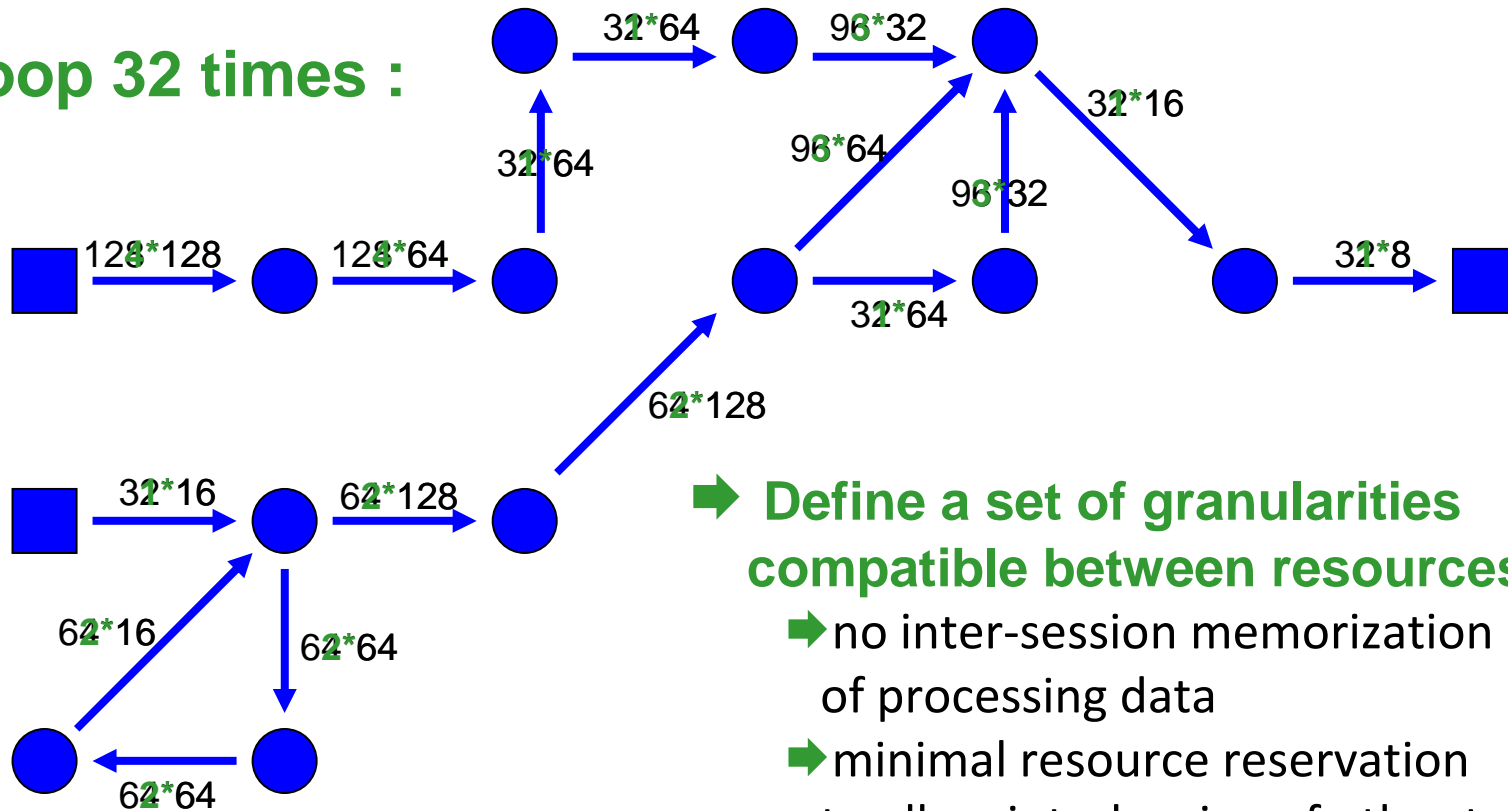
- MBs delimit system sections on which sessions are opened independently
 - PEs need to synchronize **only in their own segment**
 - **Different segments** may simultaneously process sessions on **different tasks**
 - MBs are dedicated to store data of a single task
- Session granularity is defined at section level
 - **different granularities** may be used along sections



Session Dimensioning, simple SDF example

- Given an applicative data-flow on a segment and its dimensioning

loop 32 times :



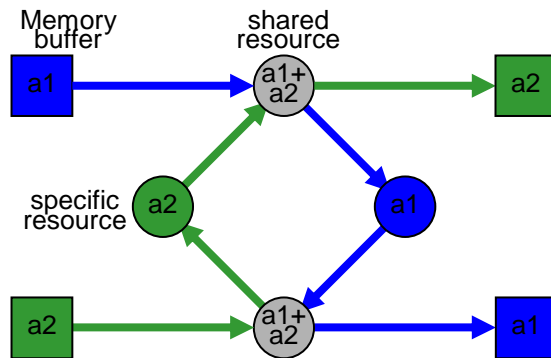
- Define a set of granularities compatible between resources
 - no inter-session memorization of processing data
 - minimal resource reservation to allow interleaving of other tasks
 - not necessarily a fixed loop : more complex scheduling is allowed

Distributed scheduling for compound task graph

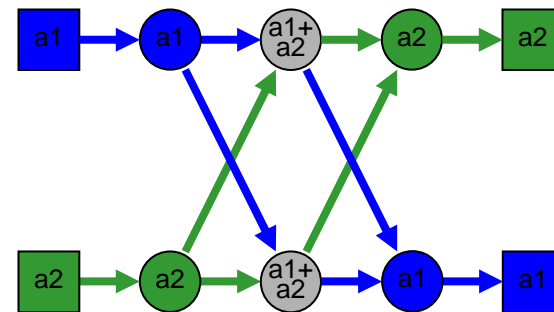
- **Centralized scheduling** would be possible, but:
 - There can be many segments, and many PEs in each segment to reconfigure between sessions
 - ◆ Non-negligible load on a centralized controller
 - Reconfiguration would be done only at the end of the processing in the last PE of the segment
 - ◆ If forward latency in the segment is quite long, this may be delaying reconfiguration of the first PEs
- We propose a **distributed sequencing protocol** in band with the dataflow
 - The **occurrence of data** at the inputs for a task **triggers reconfiguration** of the PE on that task
 - **No need for a global controller**

Application Composition: why isn't it so simple ?

- We must guarantee that all shared resources switch to the same task **together**
- Risk of deadlock for multiple shared resources
 - When accessed in different orders



“Canonical” example:
Presence of cycles



Fork example:
Race condition

Introducing Sequencing Dependencies

■ What should be prohibited:

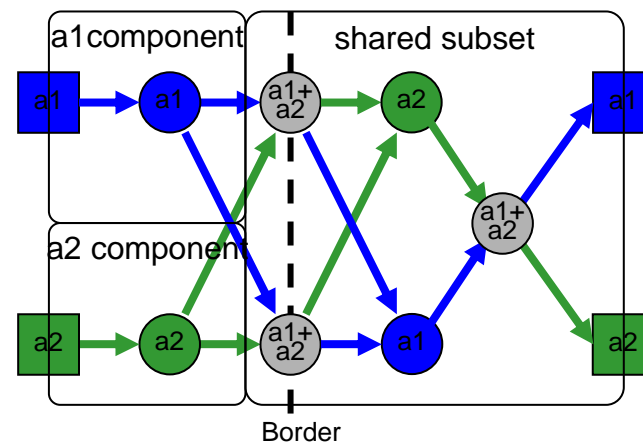
- Multiple independent decisions to switch on a task

■ What to do then ?

- Identify the **border** of the shared subset
- Let a **single entry point** in the border
- Create **virtual dependencies**

to and from this entry point:
These will propagate the decision to all the shared subset

➔ That particular PE picked in the border is now sequencing the whole segment



Fixing details...

■ Cycles within a task graph

- Differentiate **primary SRs** that will trigger a session and **secondary SRs** that will only initiate communication
- Make a **spanning tree** from the **sequencing node**
 - ◆ Arcs on spanning tree are primary, others secondary

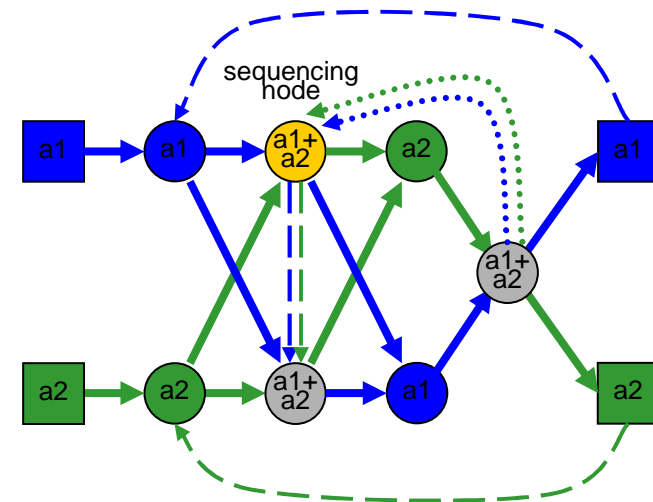
■ Add Inter-segment flow control

- start session only when MBs have **enough data/free space** for the whole session

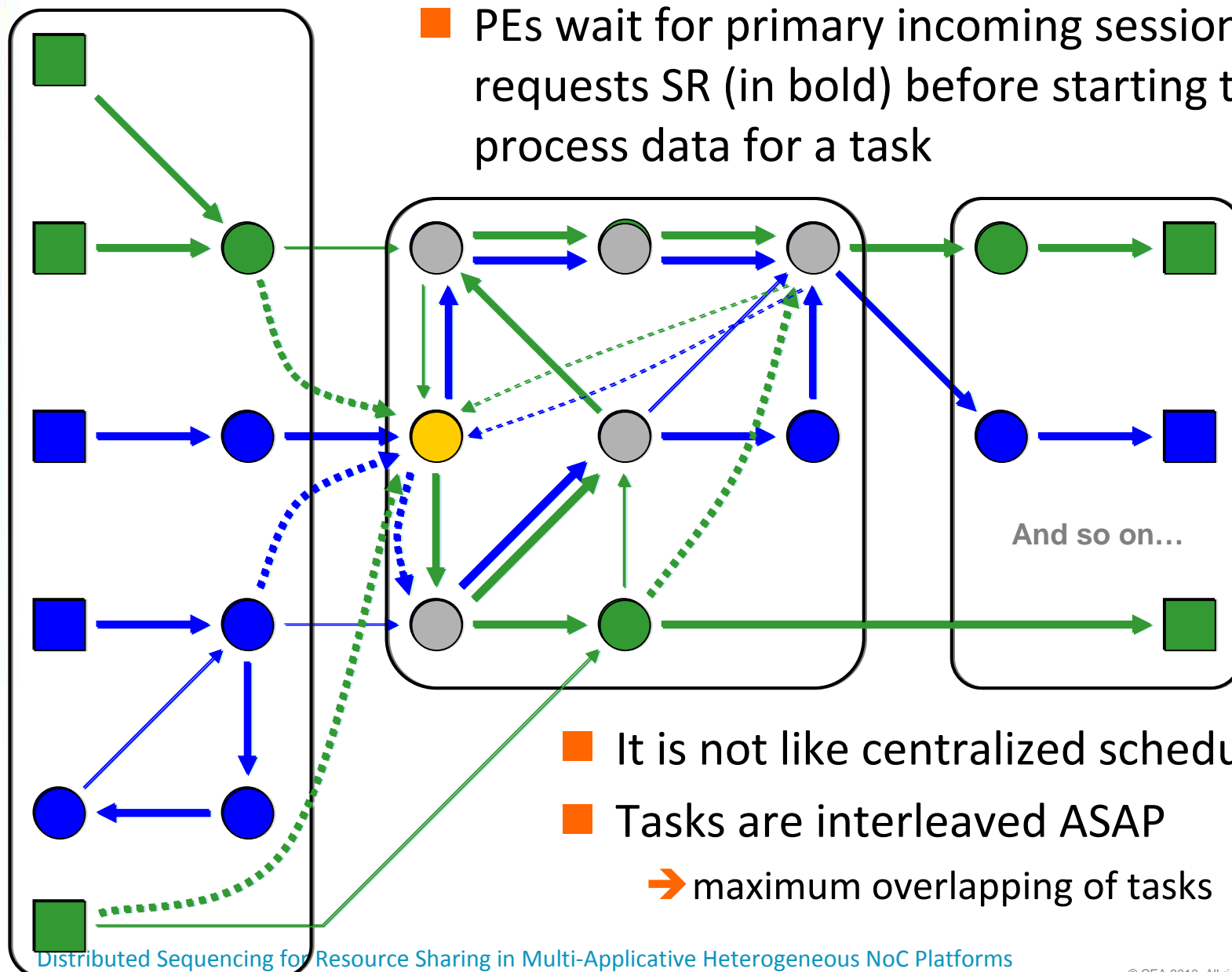
■ Ensure propagation on the whole segment

- Add secondary SRs from the leaves of the spanning tree to the sequencing node

- ◆ Only to be formally correct



How it works...



Session Request (SR) Message Format

■ A single flit

- Global task identifier
- Source identifier (local to destination)

Session request	Task Identifier	Source Identifier	Destination Identifier
-----------------	-----------------	-------------------	------------------------

SR	Tid	Sid	Path
----	-----	-----	------

(opcode)

(for table set selection)

(reference in the tables)

(routing in the NoC)

Task configuration tables for a PE

SR_in_needed_begin

(1 bit per Sid)

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

→ Sid present in this table are primary links

→ Lsb's indicate effective data transfers (to be received before credit emission)

SR_in_needed_end

(1 bit per Sid)

0	0	1	1	0	0	1	1
---	---	---	---	---	---	---	---

→ Sid present only in this table are secondary links

SR_out

Valid Sid Path

Valid	Sid	Path
1	2	Path to PE B
1	1	Path to PE C
0	XX	XXXXXXXX
0	XX	XXXXXXXX

→ At the beginning of the session, valid entries are sent to downstream PEs

→ Sid are local to the destination PE

Task monitoring tables for a PE

SR_in_received_curr

(1 bit per Sid)

0	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

SR_in_received_next

(1 bit per Sid)

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---

→ SR received from upstream,
loaded in a 2-slot FIFO

SR_out_sent

Done

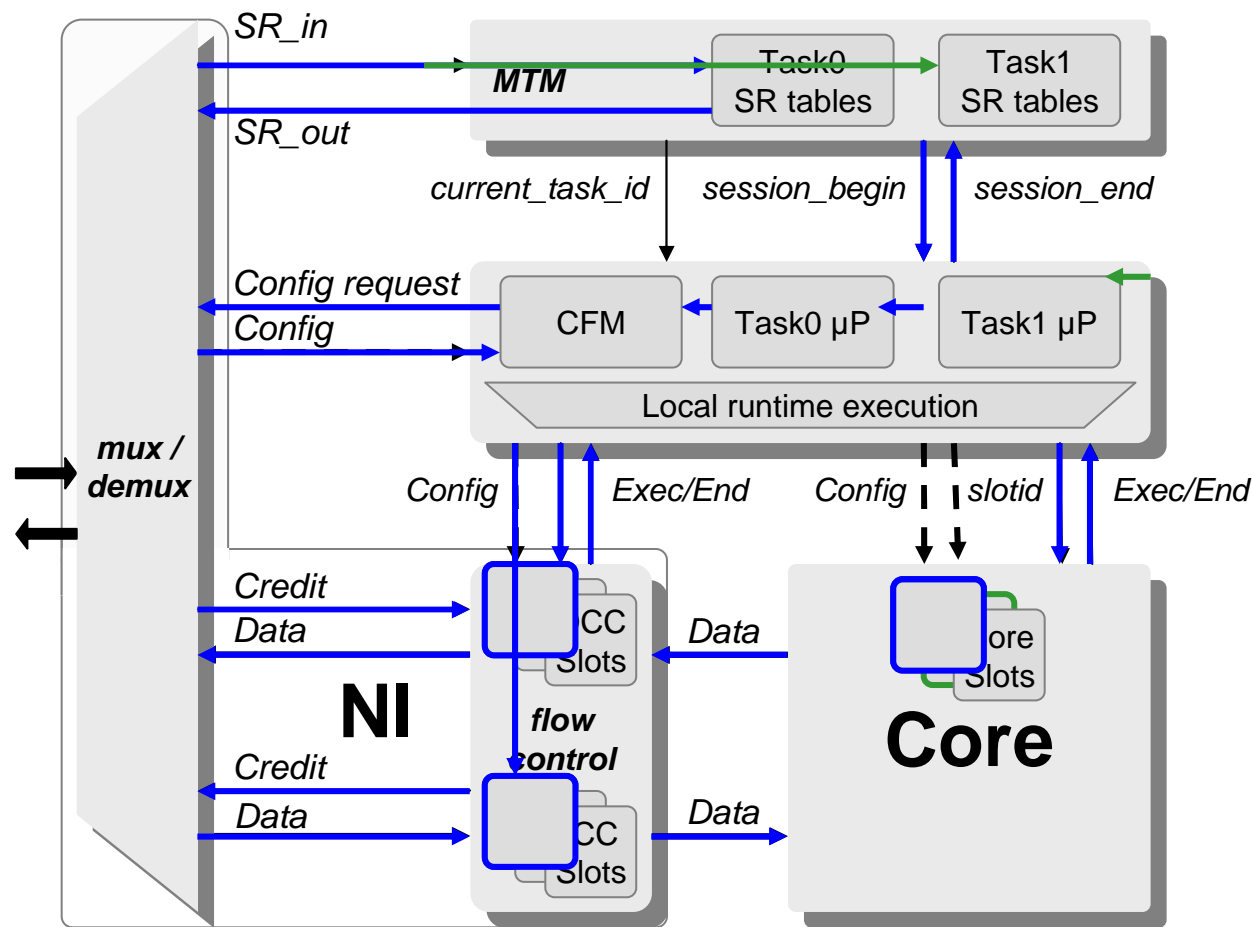
1
1
0
0

→ SR to be forwarded
downstream at beginning
of session

- Session begins when current values are equal to SR_in_needed_begin, and cannot end before current values are equal to SR_in_needed_end
- The FIFO is shifted at the end of the session
- More complex mechanisms could be envisaged for hard-real-time constraints matching on one of the tasks
- All PEs have identical hardware and behaviour, even the sequencing node!

PE micro-architecture

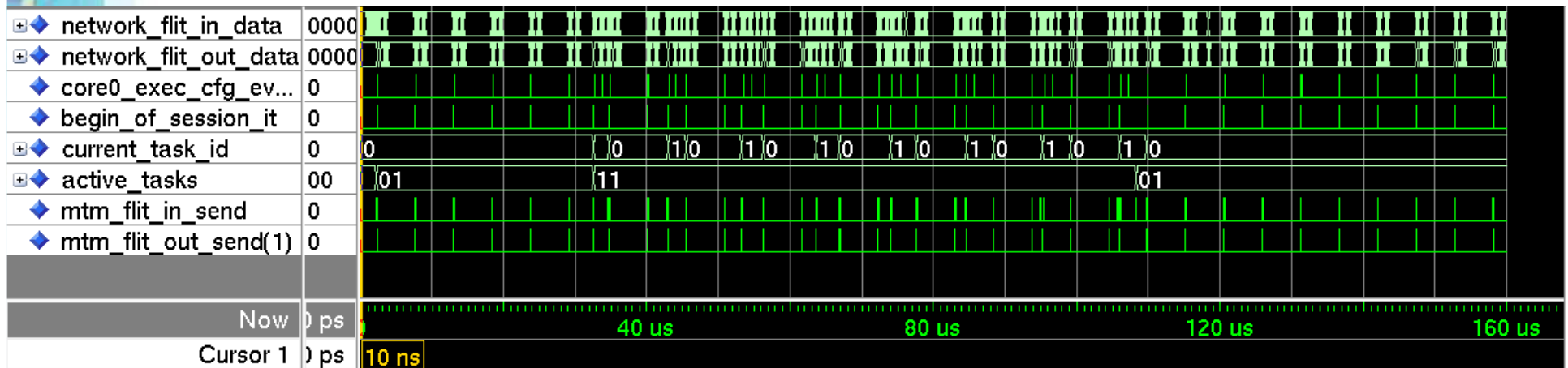
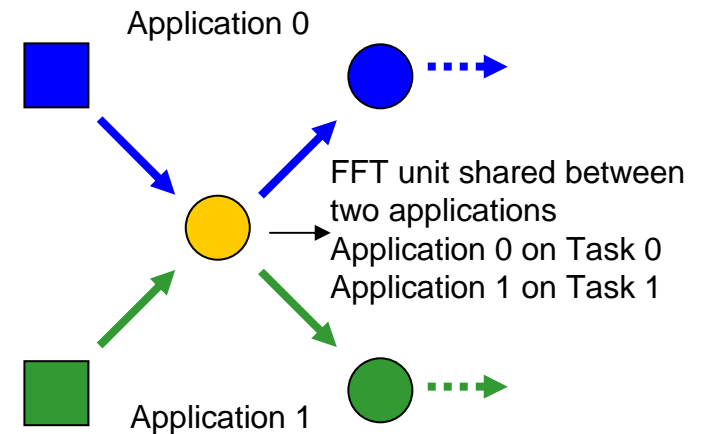
- SR configuration is distributed/activated on the bus
- ▶ SR configuration is distributed/activated on other (each) program tasks, depending on the SR received



From theory to practice...

Implementation within a telecom baseband chain

- Very demanding on FFTs
 - ◆ Tx, Rx
 - ◆ MIMO antennas
- 4x14 FFTs/iFFTs on 2048 points and 1024 points every ms



Results

■ Implemented in the MAGALI chip

- STMicroelectronics CMOS 65nm LP

■ Raw metrics

- WC frequency 500 MHz
- 10 cycle latency / PE for SR propagation
- Area for MTM+μprogs 9kGate

■ Potential savings

- Have FFT engines process both Tx and Rx for each antenna
- increase utilization from 43% and 31 % to 80%
- reduce area of 1.5 mm²
- reduce power of 12 mW



Conclusion

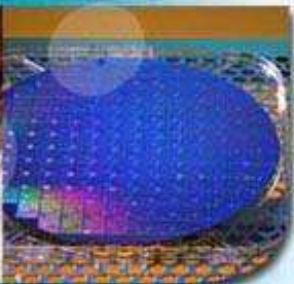
■ Realizations

- A distributed sequencing protocol to enable concurrent multi-application support between PEs on a heterogeneous NoC-based SoC
- May greatly improve resource utilization
 - ◆ Particularly in the context of reconfigurable hardware
- Does not depend on the host processor
 - ◆ Alleviates the load, reduces the number of interrupts...

■ Perspectives

- A step towards cognitive radio
- Investigate on other classes of application & programming models for different constraints

micro and nanoelectronics
microsystems
ambient intelligence
biology and health
image chain



Innovation
for industry

Loyalty
Entrepreneurship
Team work
Loyalty Innovation
Entrepreneurship
Team work
Innovation

leti

MINATEC

INSTITUT
CARNOT
CEA LETI



cea