A Dynamic Stream Link for Efficient Data Flow Control in NoC Based Heterogeneous MPSoC

Claude Helmstetter[1], Sylvain Basset[1], Romain Lemaire[1], Fabien Clermidy[1],

Pascal Vivet[1], Michel Langevin[2], Chuck Pilkington[2], Pierre Paulin[2], Didier Fuin[3]

[1]*CEA-Leti, Minatec Campus, Grenoble, France*
[2]*STMicroelectronics, Ottawa, Canada*
[3]*STMicroelectronics, Grenoble, France*

18th Asia and South Pacific Design Automation Conference, ASP-DAC 2013
Jan. 22-25, 2013 – Yokohama, Japan

Heterogeneous Multicore SoCs

- Embedded computing evolution
  - New *application*: 3D imaging, multi-antenna wireless baseband processing
  - Common *characteristics*: Complex operations, high data rate, real-time requirements
  - *Efficiency* requirement: Performance versus Power consumption
- New architectures for Systems-on-Chip
  - *Complexity*: Increasing number of computing cores
  - *Heterogeneity*: CPU, DSP, Reconfigurable HW cores
  - *Parallelism*: Dataflow programming

Stream-Based Communications

- From bus-based architecture to *Network-on-Chip*
  - Bandwidth requirement due to increasing number of computing cores
  - Well-defined communication *interfaces* required
    - Ease of integration, reuse and programming
  - Bus protocols not well-adapted for NoC communication with dataflow programming
    - Address management issue
- Introduction of *stream-based* NoC protocols
  - Local control by *Network Interface* (NI)
  - Packet-switched data transfers + Flow control (credits)

Dynamic Stream-Based Link Management

- *Configuration* of stream-based communication links
  - How and when communications can be *configured*
  - How and when communication links are *started/stopped*
- *Synchronization* requirements
  - How to synchronize communication links
  - How to synchronize communications and computation
- Issue: *dynamic* communication parameters
  - *Non-predictable, data-dependent* control
  - Control latencies become critical (reaction time)

➔ Novel NI architecture supporting dynamicity

- *SIF*: *S*treaming *I*nter*F*ace
- *Objectives*: fast reconfiguration and limited control workload

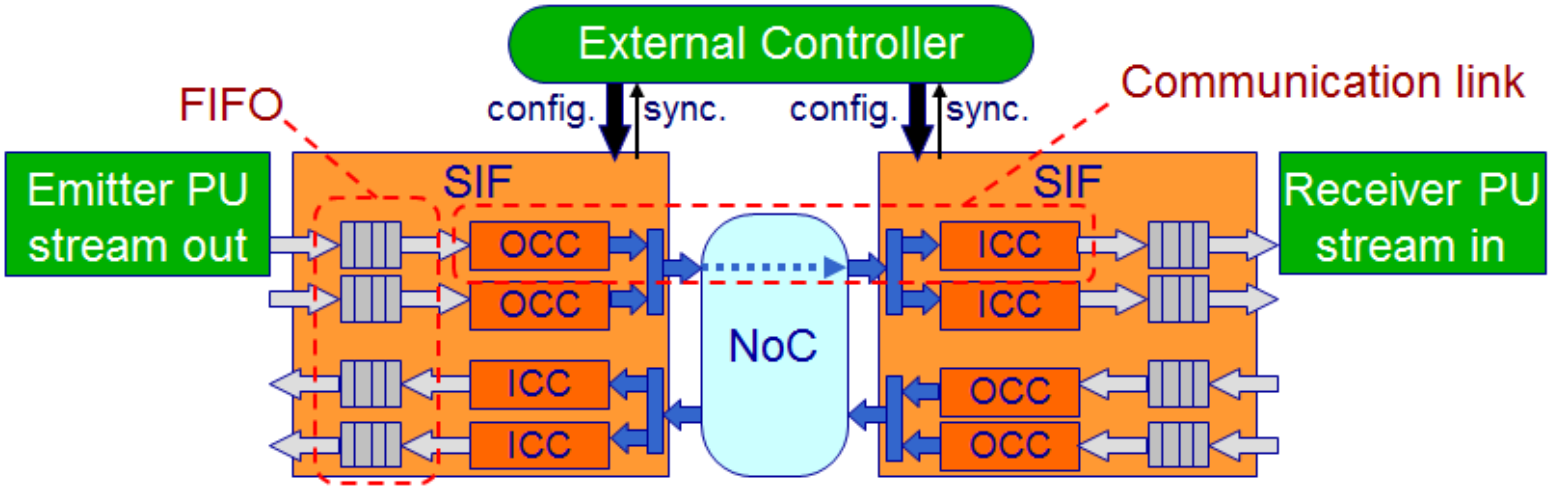
Outlines

- Introduction
- Streaming interface and stream link protocols
- Hardware Implementation
- Evaluation and simulation results
- Conclusion

Outlines

- Introduction
- Streaming interface and stream link protocols
  - Stream link definition
  - Static stream link
  - Iteration-based dynamic stream link
  - Mode-based dynamic stream link
- Hardware Implementation
- Evaluation and simulation results
- Conclusion

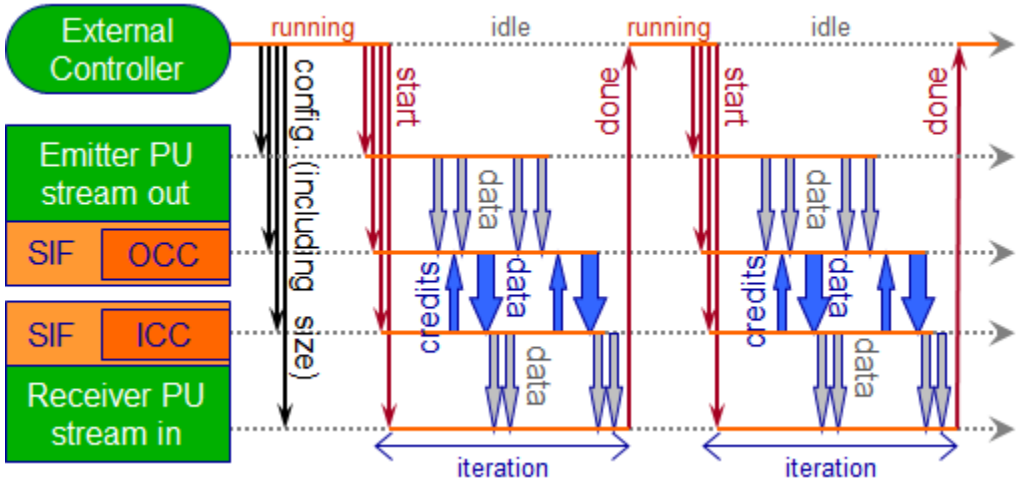
Stream Link Definition



- *Processing Units* (PU)
  - Receive/send data using FIFO interfaces only (dataflow)
  - Optional additional ports for control (registers, IT wires, …)
- *Streaming Interfaces* (SIF)

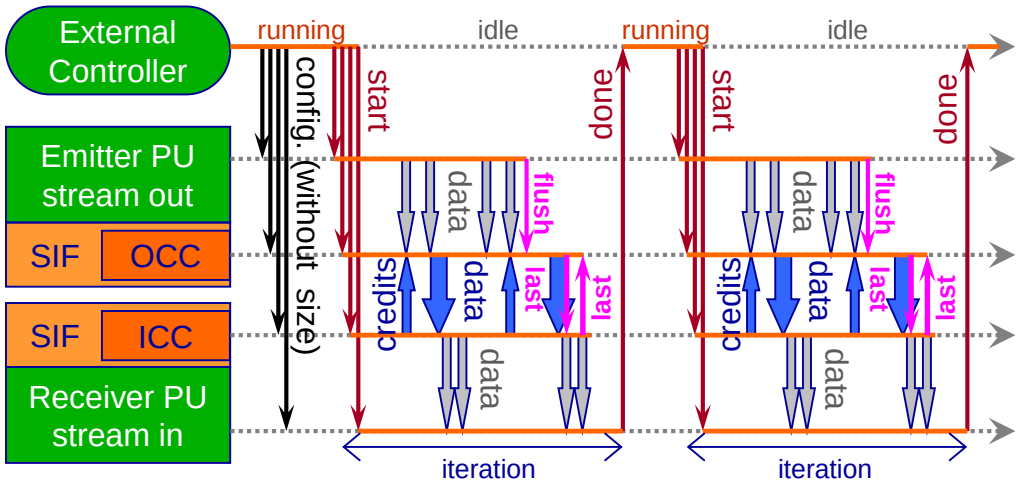
Output Communication Controller (*OCC*)	Packetise data, add routing information (packet header)
Input Communication Controller (*ICC*)	Depacketise data, send *credit* backward (flow control)

Static Stream Link Protocol



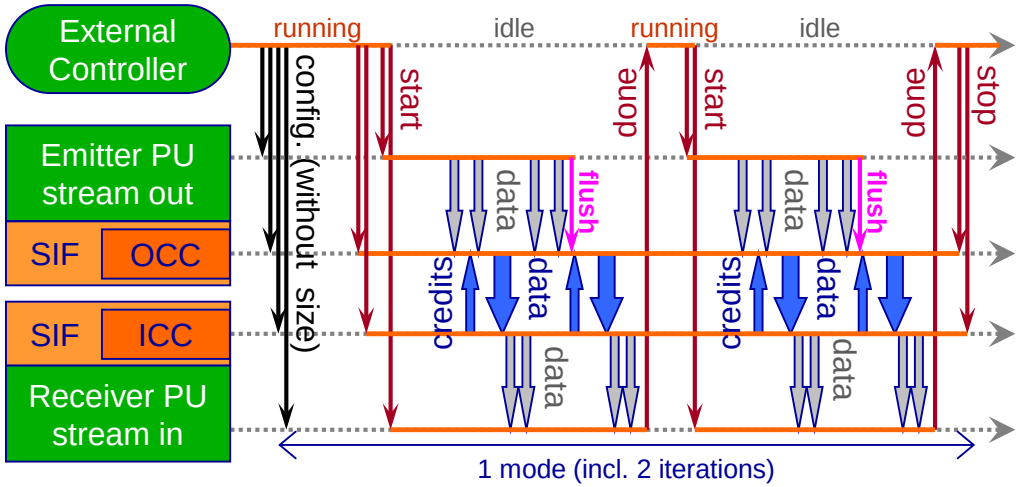
- SIF are configured with a predefined total number of data to transfer: *iteration* size
- Stream link closed locally using a simple counters
- Iteration size must be predictable at configuration time

Iteration-Based Dynamic Link



- The emitter PU decides to close the link (flush)
- The OCC sends a packet flagged as *last*
- Acknowledgement sent by the ICC
  - Required for the data/credit mechanism reset
- No need to predict iteration size at start time
- Require additional communications

Mode-Based Dynamic Link

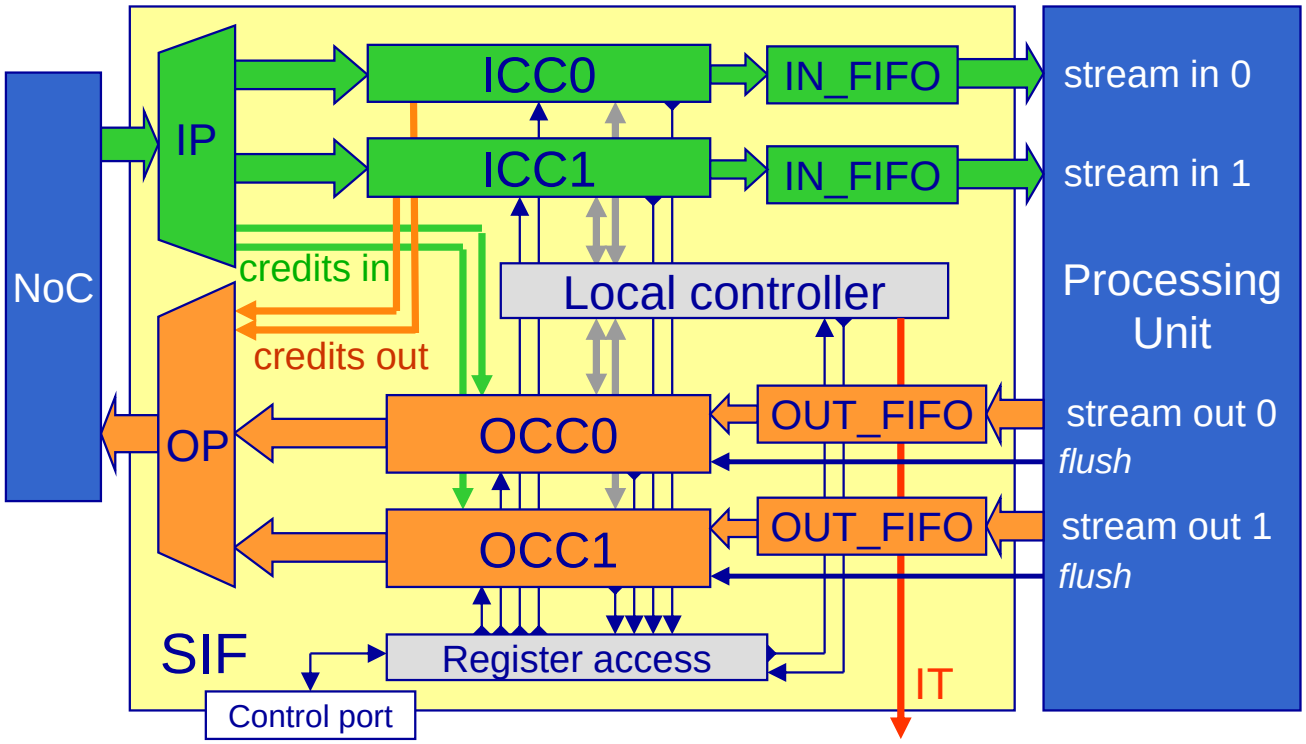


- *Mode:* Time between 2 datapath reconfigurations
- Mode-Based Dynamic Link:
  - PU controlled at iteration level
  - Stream links stopped only on mode boundaries
- Suppress close/reopen delay between iterations
- Longer reconfiguration time between modes

Outlines

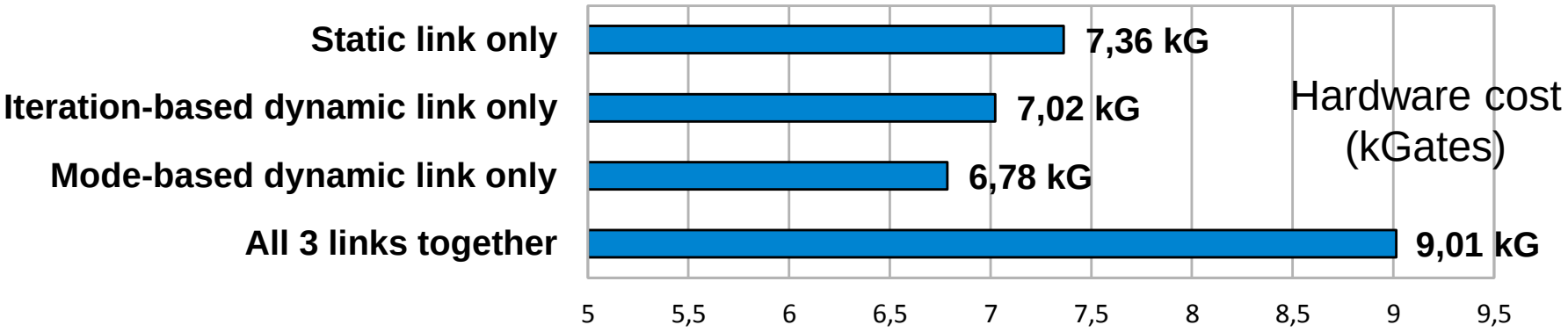
- Introduction
- Streaming interface and stream link protocols
- *Hardware Implementation*
  - *Architecture overview*
  - *Cost overhead for dynamic link protocols*
- Evaluation and simulation results
- Conclusion

Architecture Overview



- Architecture fully *customizable* at design time
  - Number of ICC/OCC, FIFO depth, datawidth, …
  - Selection of stream link protocols (possibly combined)

Silicon cost of dynamic link protocols



- Technology: CMOS 32nm @ 600MHz
  - 1 ICC, 1 OCC, 32-data FIFO depth, 64-bit data width
- Relative costs:

Mode-based	Smallest design
Iteration-based	Extra-cost due to last packets protocol
Static	Overhead from additional counters

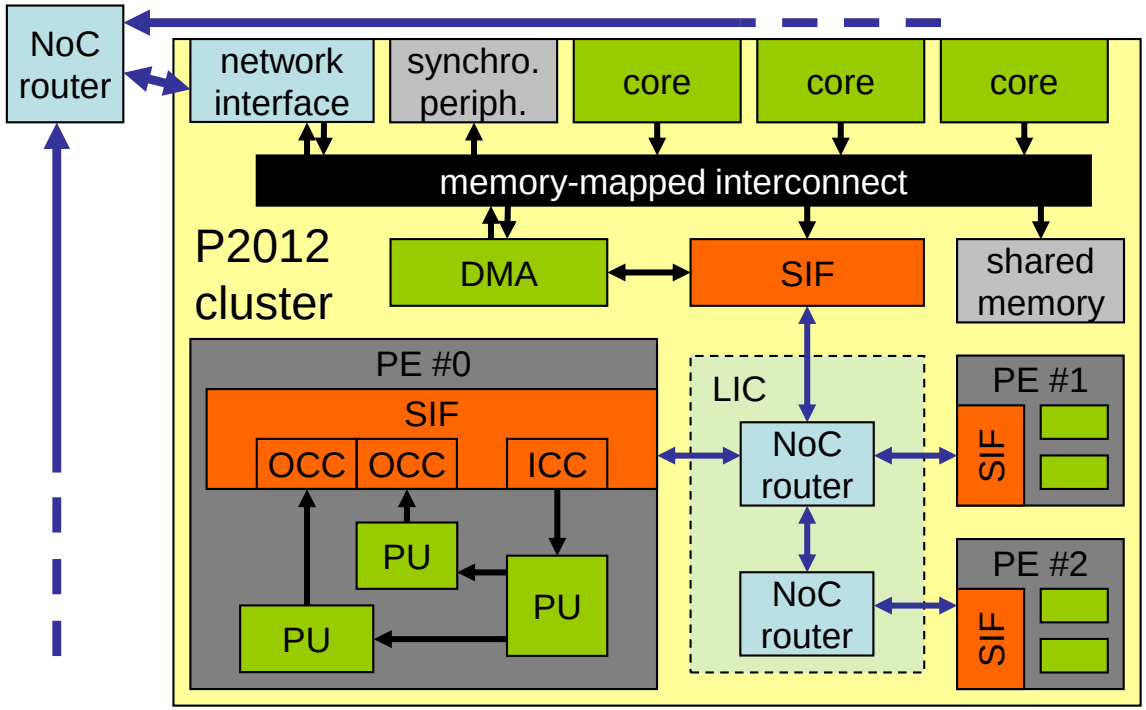
- Parts of the design shared when combining protocols

Outlines

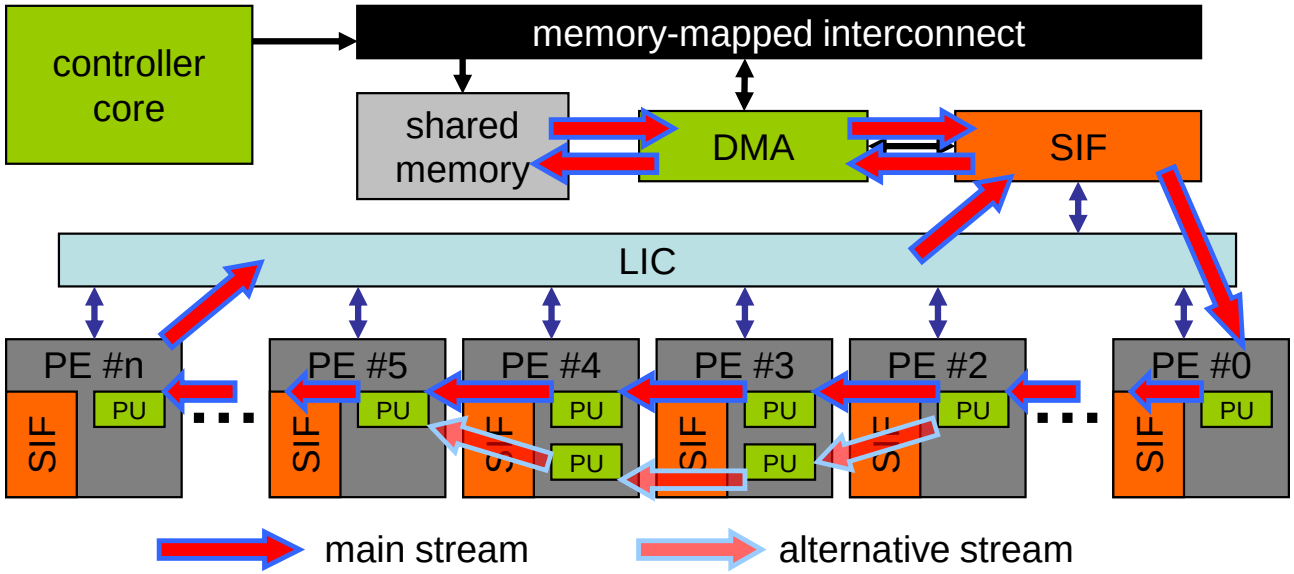
- Introduction
- Streaming interface and stream link protocols
- Hardware Implementation
- *Evaluation and simulation results*
  - *Case study and benchmark*
  - *Mode-based dynamic link*
  - *Iteration-based dynamic link*
- Conclusion

Case Study: the P2012 SoC

- Generic many-core SoC for high performance computing
  - Extensible with hardware Processing Elements (PE)
  - Developed by STMicroelectronics and CEA/LETI
- P2012 cluster architecture:



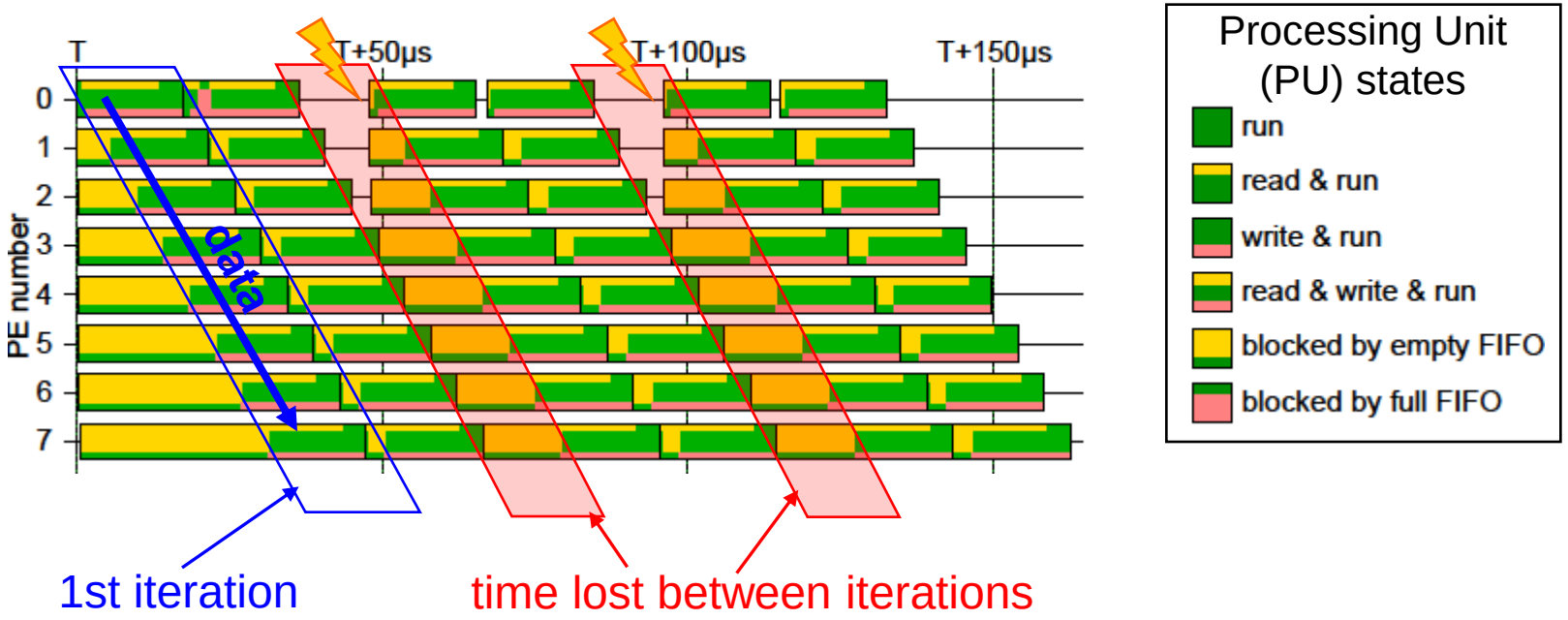
Benchmark Overview



- Linear data-flow going through the DMA and *n* Processing Elements (PE)
- Many parameters: number of PEs (*n*), iteration size, PU rate, latencies, FIFO depth, …
- *Alternative Stream* to test path reconfigurations

Mode-based Dynamic Link

Execution trace of the benchmark with mode-based dynamic link



- 1st and 2nd iterations run perfectly fine
- … but time lost before 3rd and 5th iterations
- Problem: PE #0 is restarted too late (cf.)

Optimizing the Controller

- *Basic controller:*

```
while (not_done()) {
     wait until all PEs are ready to be configured;
     configure and start all PEs;
}
```

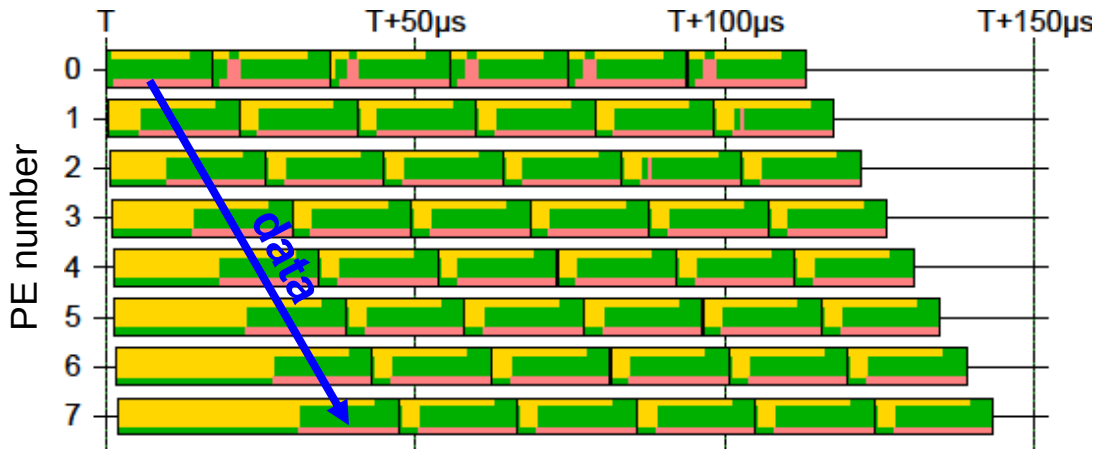
- *Idea:* do not wait for last PE before restarting the 1st
- *Optimized "asynchronous" controller:*

```
while (not_done()) {
     For all PE {
       if "this PE" is ready to be configured
           configure and start "this PE";
      }
     wait an event from any PE;

}
```

- NB: new embedded software but hardware unchanged
  - As long as the processor running the controller is fast enough

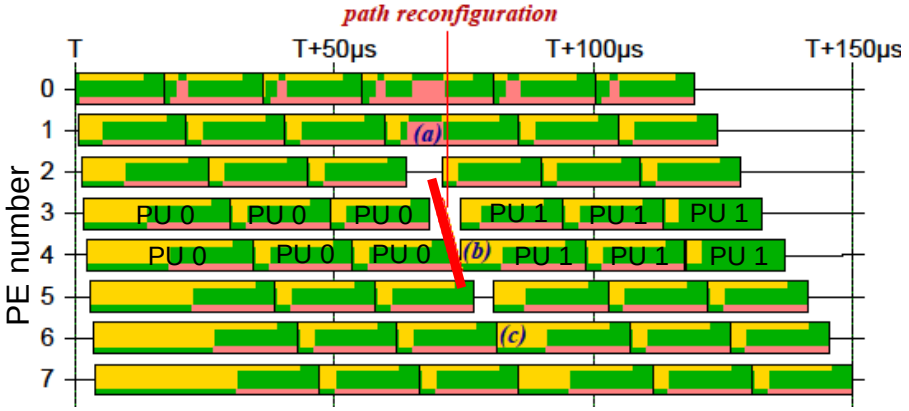
Mode-based Dynamic Link



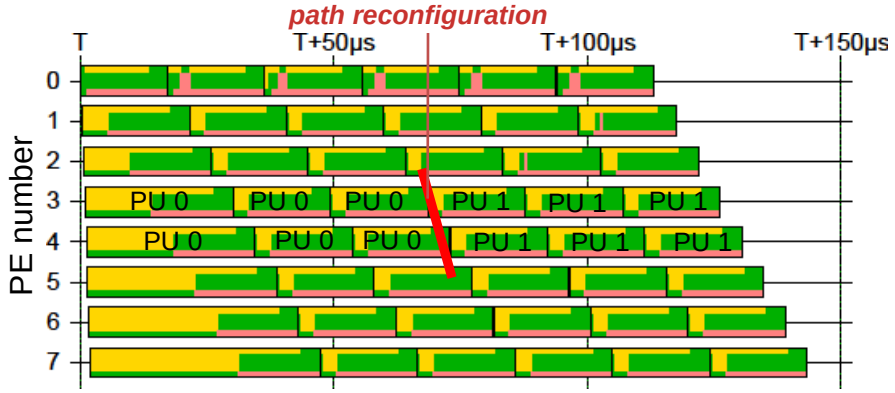
- Execution of the benchmark with optimized controller
  - No more pipeline holes between iterations

Cost of Path Reconfigurations

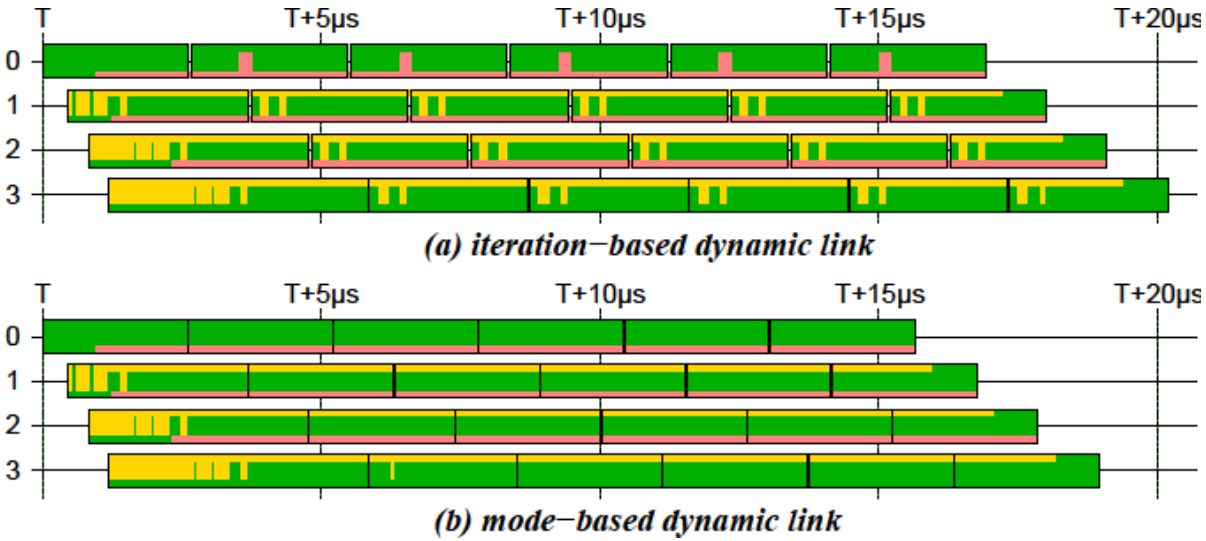
- Mode-based dynamic link
  - Even partial reconfiguration generates a pipeline hole
  - This hole can be larger if the controller is not optimized



- Iteration-based dynamic link
  - No time cost for path reconfiguration



Iteration-based vs Mode-based



- Mode-based 10% faster than iteration-based dynamic link
  - For short iterations (figure generated with 60-token long iterations)
  - Explanation: FIFOs are not flushed between iterations

Outlines

- Introduction
- Streaming interface and stream link protocols
- Hardware Implementation
- Evaluation and simulation results
- *Conclusion*

Conclusion (1/2)

- Validation of an innovative SIF architecture
  - Support non predictable communication parameters
  - 1 static & 2 dynamic protocols available
- « Best protocol » is application dependent

Application characteristics	Recommended protocol
Predictable communication	Static Stream Link
No or few path reconfiguration	Mode-based Dynamic Link
Other cases	Iteration-based Dynamic Link

Conclusion (2/2)

- External controller (embedded software) impact on performance
  - Too centralized, coarse-grain: simpler code but performance degradation
  - Asynchronous, optimized: easer parallelization, better reactivity if fast enough
- Known choices for real applications:
  - 3GPP-LTE (signal processing), Magali SoC: static link
  - H264 (video processing), P2012 SoC: mainly mode-based dynamic link

Thank you

LABORATOIRE D'ÉLECTRONIQUE
ET DE TECHNOLOGIES
DE L'INFORMATION

CEA-Leti
MINATEC Campus, 17 rue des Martyrs
38054 GRENOBLE Cedex 9
Tel. +33 4 38 78 36 25

www.leti.fr