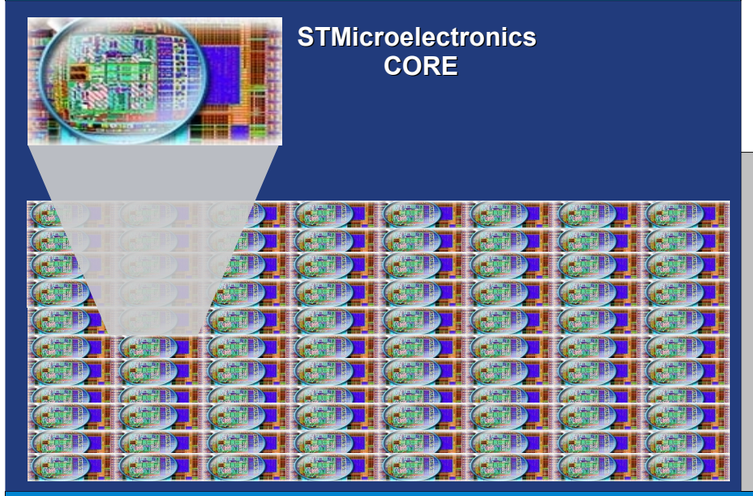
MPSoC Observation with Fractal

Carlos Prada[1], Vania Marangozova-Martin[1], Kiril Georgiev[1], Jean-François Méhaut[1], Miguel Santana[2]
[1]FirstName.LastName@imag.fr, [2]Miguel.Santana@st.com

Future Multi - Processor Systems on Chip are many-core architectures



Supporting parallelism needs new programming models and development tools

- Rapid software development
- Software tuning : debugging and performance optimization

NEED for new observation tools for MPSoC

Related Work on Observation

- **MPSoC** : low-level and platform-specific [1][2]
- **Parallel systems**: closely related to the parallel programming model [3][4]
- **Component systems**: high-level of abstraction non related to low-level performance metrics [5]

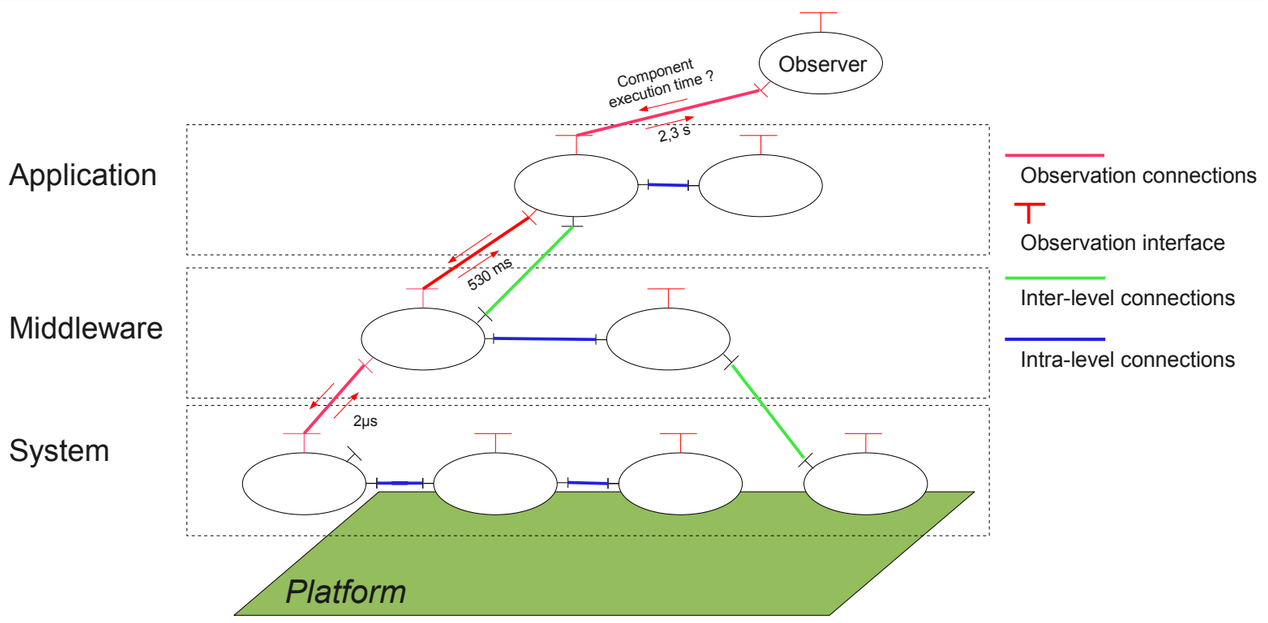
EMBera: A Fractal-based Model for MPSoC Observation

Objectives:

- Observe different types of applications
- Observe different hardware MPSoC families
- Be configurable to a specific execution platform:
  - Select events to observe (multi-level)
  - Provide good performance

EMBera:

- Fractal primitive active components
- Provided / Required interfaces
- Connections
- **Observation control interface**



Method for requesting observation information (recursive approach)

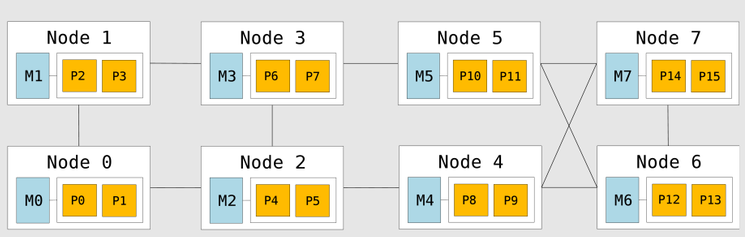
EMBera Prototype:

New Implementation of Model in C Language

EMBera on SMP – Linux

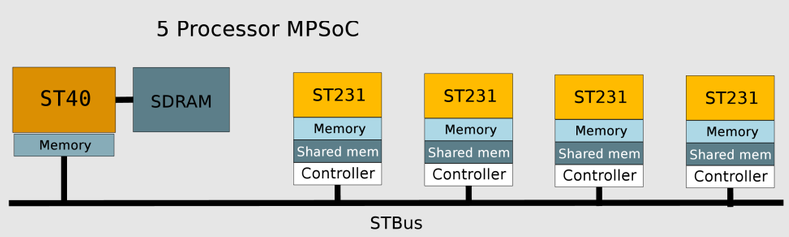
- Application: set of POSIX threads (Linux process)
- Component: one thread + control structures
- Communication: simple asynchronous send / receive operations

16-core SMP



EMBera on STi7200 MPSoC – OS21

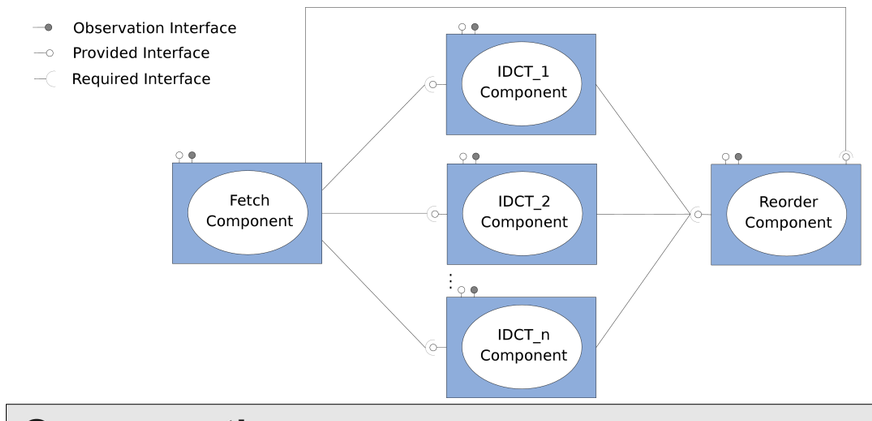
- Application: set of processes (OS21 Tasks)
- Component: one OS21 task + control structures + distributed objects
- Communication: Based on ST communication software



Observation Information

Three levels of observation: Operating System, Middleware, Application

Target application: EMBera componentized **MJPEG Decoder:** (input file: video stream – 578 images)



Observation of component IDCT2:

	SMP – Linux	MPSoC - OS21
Operating system - Component Allocated Memory	8 350 kB	60kB
Operating system – Component Execution Time	4,08 s	105 s
Middleware - **Send** execution time	81,23 µs	0,8 ms
Application - # data exchanges among components	3 462	5 193

Open questions

- What functions should be provided by the observation interface?
- Towards an Abstract Device Interface for Observation(ADI)?
- How do we select the events to be observed?
- How do we set the treatments to apply?
- How do we manage multi-level information?

References:

[1] Dynamic Kernel Tracing with KPTrace. Website. http://www.stlinux.com/docs/manual/development/advanced_development30.php.
[2] SpyKer. Website. http://www.lynuxworks.com/products/spyker/spyker.php3
[3] Sameer Shende and Allen Malony. The TAU parallel performance system. *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.
[4] POSIX Thread Trace Toolkit (PTT). Website. http://nptltracetool.sourceforge.net/.
[5] Nick Trebon, Allen Morris, Jaideep Ray, Sameer Shende, and A. D. Malony. Performance Modeling of Component Assemblies. *Research articles. Concurrent Computing: Practice and Experience*, 19(5):685–696, 2007.