

PROARTIS

Probabilistically Analysable Real-Time Systems

System Software



Tullio Vardanega (UniPD)

**ARPA Workshop
Berlin, January 22, 2013**

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 249100.



www.proartis-project.eu

Outline

- *System-level challenges to be faced in PROARTIS*
- *System-level challenges impeding outside of PROARTIS*
- *PROARTIS rationale*
- *An understanding of time composability*
- *Results*

System-level challenges within PROARTIS

- **C1:** *Determining the impact of the PTA method on the architecture and execution of the software system*
 - What analysis hypotheses must hold true for the software system
- **C2:** *Determining (a) what the software elements of the execution stack must do and (b) what they may do to facilitate the analysis and draw maximum benefit from it*
 - The *must* part reflects the obligations that follow from challenge C1
 - The *may* part captures gain opportunities
 - In the quality of the analysis result
 - In the non-intrusiveness of the application of the analysis in the industrial development process

Perimeter of investigation

- ***Operating System level***

- Whether and how constant-time OS services can be delivered to the application
 - With no perturbation effect on the WCET of the application
 - In the direction of *time composability* between OS and application
 - That's the way OS-level scheduling may facilitate PTA for the application

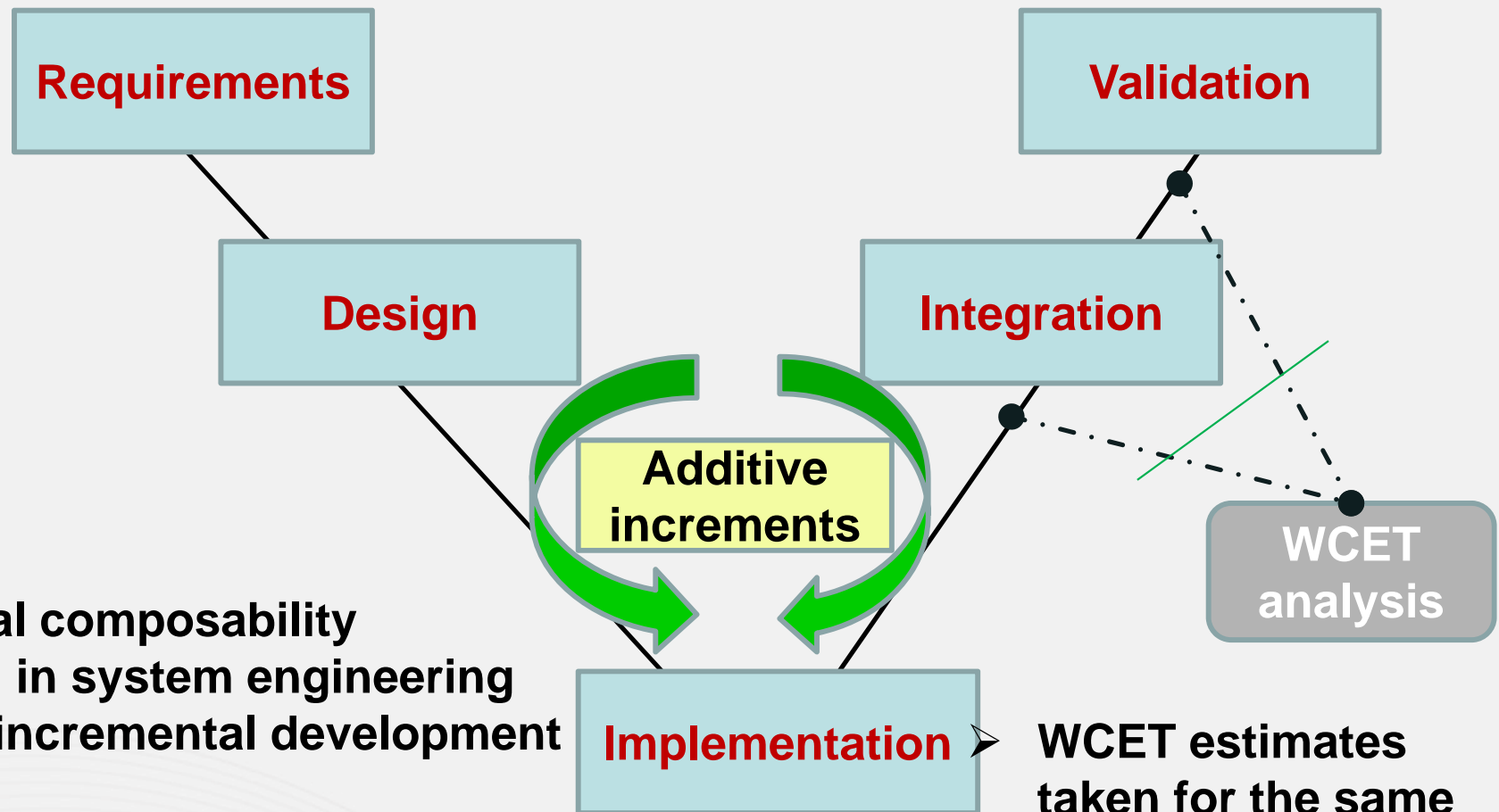
- ***Programming paradigms patterns and styles***

- With the above working, the application concerns could be limited to facilitating timing analysis at its own level
 - No specific PTA requirements on that level

System level challenges outside PROARTIS

- *Pre-PROARTIS timing analysis badly clashes against **incremental development** and qualification*
- *Incremental development is the obvious way to mitigate development and schedule hazards*
 - Industry needs it badly
 - Its use presupposes composability in *all* dimensions
- *Timing analysis techniques assume **time composability***
 - But software systems in general are not time composable
 - State-dependent perturbation effects can only be studied “per system” if excessive pessimism is to be avoided
 - To overcome this intrinsic contradiction pre-PROARTIS timing analysis is normally applied to the final system only
 - Which is very bad news to incremental development

Incremental development and composability



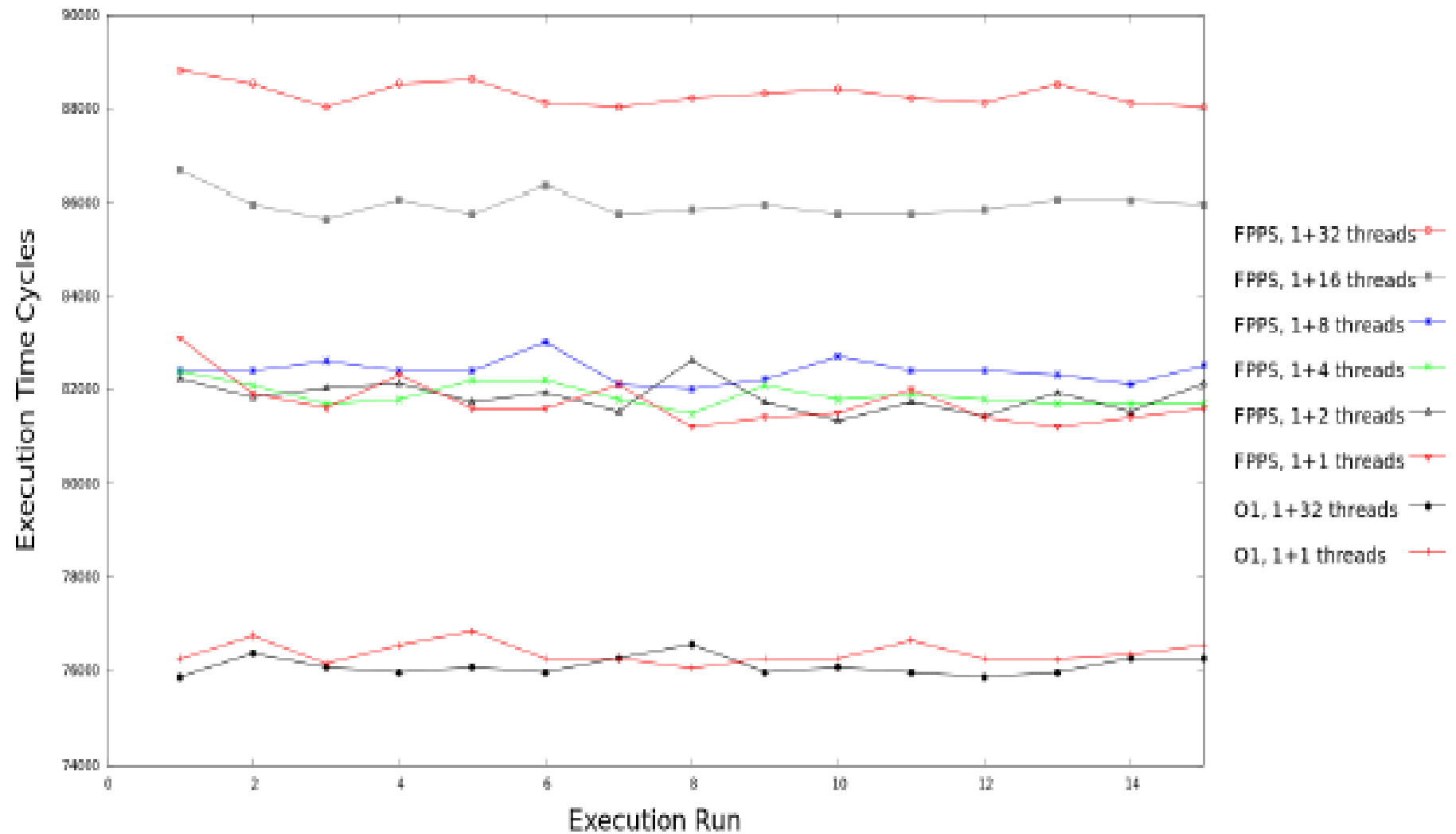
- Functional composability is a given in system engineering
- It serves incremental development very well
- This is NOT the case for time composability

➤ WCET estimates taken for the same program at different steps of integration may be very different!

Time isolation

- *Some software architectures postulate on time isolation*
 - My friend IMA does that
 - And teaches and preaches it to the rest of the world ☹
- *But true time isolation can only be attained with physical isolation*
 - Which uses federation, the exact negation of integration
- *You gain time isolation if you have time composability but you don't want mad overprovisioning to achieve it*
 - You are not time composable if your response time jitter causes overprovisioning
- *So now you know what you are after*

The evil of tick scheduling

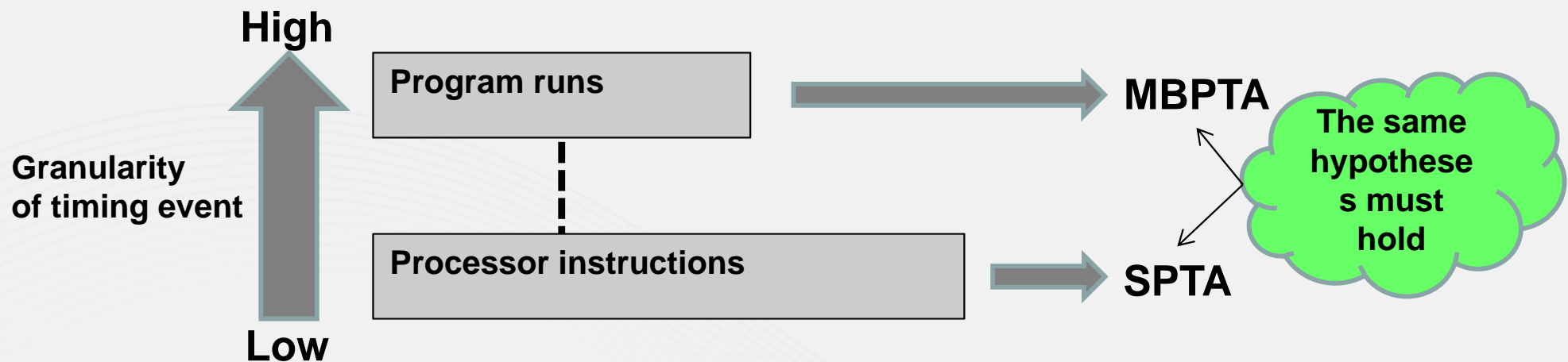


PROARTIS rationale /1

- *Processor HW removes the dependence of timing behaviour from the history of execution by design*
 - Or makes it probabilistically analyzable
- *This feature is provided at the level of processor instruction*
 - A conveniently high level of abstraction from the HW perspective
- *Execution time variation at the application level can therefore only stem from one of two sources*
 - **S1**: Data-dependent choice of execution paths
 - **S2.a**: Jittery response time of OS calls
 - **S2.b**: Perturbation effects caused by the OS execution on HW state upon return from the call

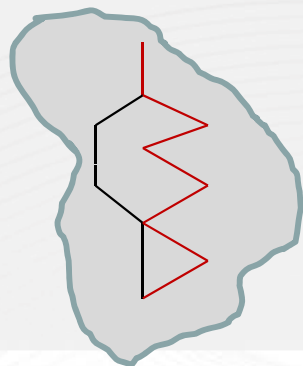
Intuition /1

- *The input data to PTA are timing events occurring at some level of execution granularity*
 - Independence is strictly required for them
 - Identical distribution makes the analysis tractable
- *For sensible and practical application of PTA the timing events in use have to have the same granularity*
- *Whereas the PROARTIS argument is built bottom up the application of PTA has to be top down*

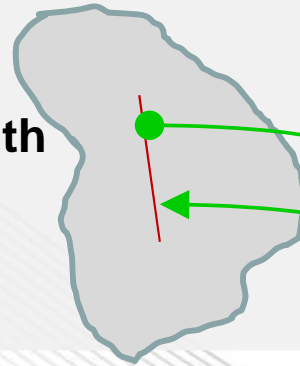


Intuition /2

- *Processor instructions in a PTA-friendly processor are time composable by design*
 - That's great for bottom-up reasoning
 - But we need to study things from the top
- *Program runs are **not** time composable: two major sources of jitter in their time behavior exist*
 - **S1**: Data-dependent execution paths within the same program
 - **Horizontal** issue: within one level of execution granularity
 - **S2**: Cost and state-perturbation effects from calls into the OS
 - **Vertical** issue: across distinct levels of execution granularity



A single program unit with many execution paths



A program run calling into the OS

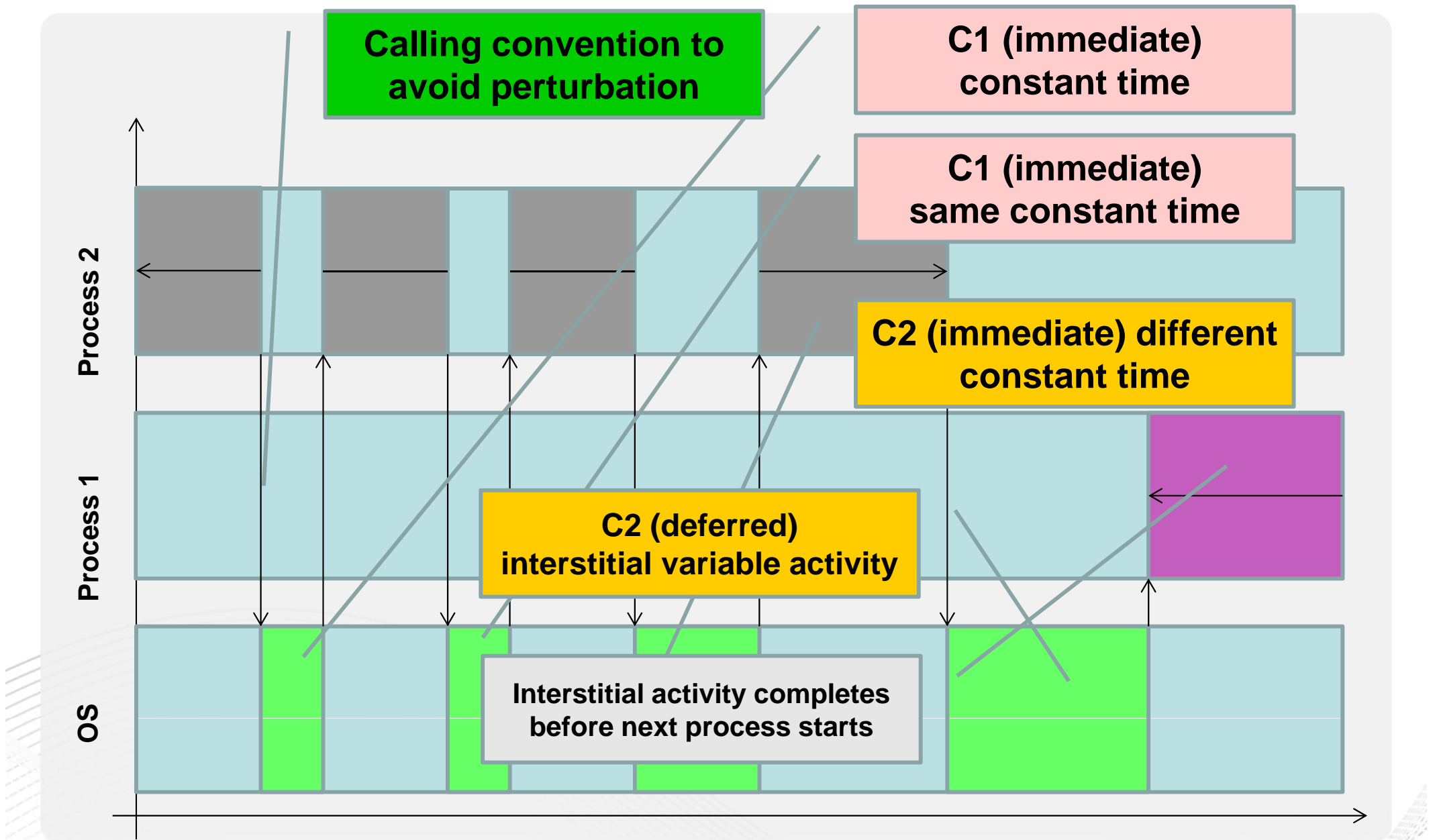
PROARTIS rationale /2

- *Source **S1** is addressed by achieving sufficient path coverage in the PTA observation runs*
 - No different from the pre-PROARTIS situation
 - Only made much simpler because PTA only needs to consider the program units in isolation
- *Source **S2.a** is addressed by redesigning the OS so that its immediate services have a near-constant tightly upper-bounded response time*
 - Immediate OS services are those that have to complete before returning control to the application
- *Source **S2.b** is addressed by ensuring that those OS services do not modify the state of the probabilistically-variable HW resources*

PROARTIS rationale /3

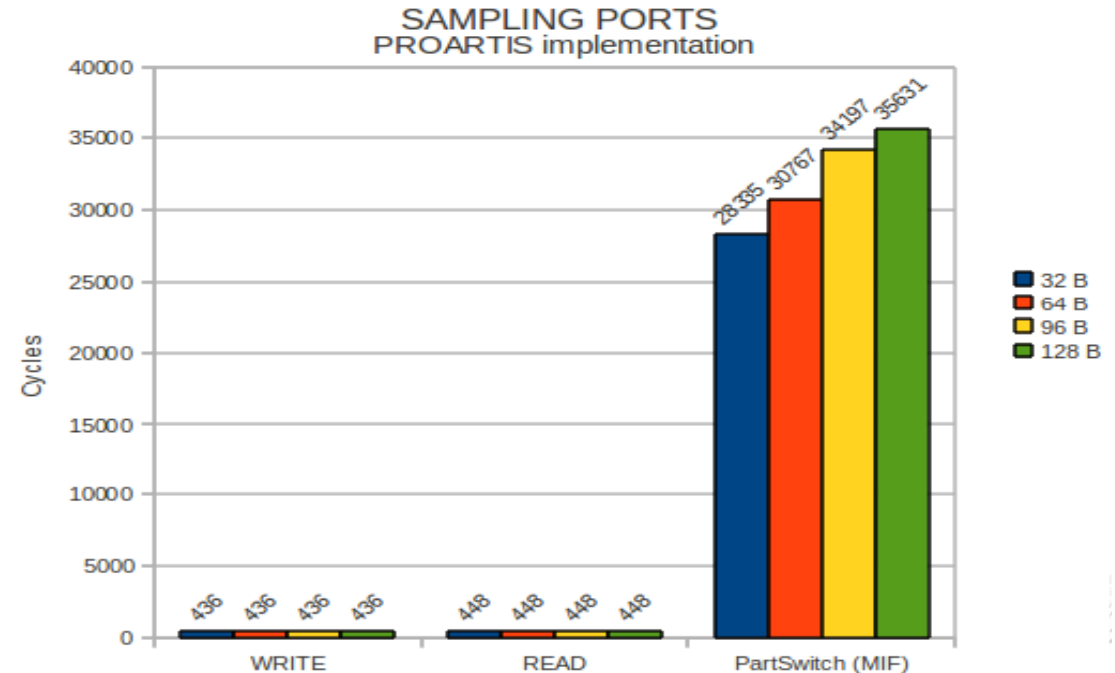
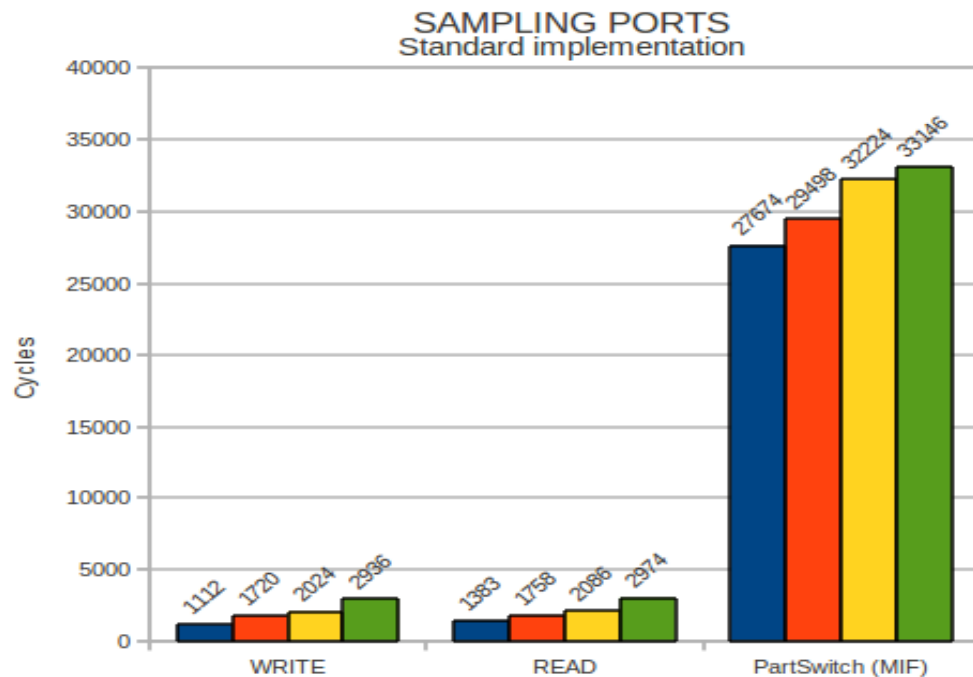
- *Not all OS services need to be immediate*
 - Some can be deferred until the calling process has completed or even later
 - Servicing of dispatching points
 - Message delivering
 - Others execute outside of critical instants (e.g., initialization, partition switch) so that their timing behavior only relevant at the level of [IMA] partitions
- *This distinction becomes handy when combined with run-to-completion semantics for ARINC processes*
 - A permissible ARINC behaviour
- *The ARINC process becomes the largest-granularity unit of composition and analysis*
 - The one that must be made time composable

Time-composable service model



Some experimental results

	FPPS (standard POK)			O(1) scheduler (partition switch)			O(1) scheduler (thread switch)		
	Min	Max	Delta	Min	Max	Delta	Min	Max	Delta
2 partitions 5 threads	255	714	459	232	232	0	N/A		
3 partitions 17 threads	259	1759	1500	232	232	0	45	95	50



PROARTIS rationale /4

- *Immediate OS services can be made to have nearly constant response time*
- *They are prevented from causing perturbation effects on the HW state left on return to the application*
 - If we look at the cache we can either inhibit access (easier) or use partitioning (only if the service may benefit from the cache)
- *Hence their contribution to the execution time of the caller can be regarded as a simple additive term*
- *The units of composition at the application level have therefore become time composable!*
 - See next slide for an understanding of the time-composability problem at application level

Time composability at application level

- *Problem statement for any program unit B that may be called more than once in the CFG of a main procedure P*
 - $[B_p, \emptyset, B_q] \ [B_p, \delta_1, B_q] \ [B_p, \delta_2, B_q]$ where B_i is the i^{th} run of B in P
- *Solutions*
 - Flush HW state prior to the each execution of B
 - Time composable but for a pessimistic pWCET estimate
 - Probabilistically bound the state perturbation effect caused by application-level execution in between any two subsequent observations of B
 - The effect that δ_i can have on B_j is **completely characterized** by the **reuse distance** rd of the memory accesses made in δ_i
- *For any two programs ρ_1 and ρ_2*
 - If for each memory access in ρ_1 with $rd = x$ there is an access in ρ_2 with $rd = y : y > x$ then ρ_1 upperbounds the effect of ρ_2 on B_q

Conclusions

- *We have seen that the PTA principles can be injected bottom-up in the design of a complex embedded system*
- *Time composability is an extremely valuable property to enable incremental system development*
- *The PROARTIS HW design makes processor instructions time composable by construction*
- *We have shown that the Operating System can be made time composable toward the application*
- *With that we have provided time composability for program units at the application level*

PROARTIS

ARPA workshop



January 22, 2013

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 249100.



www.proartis-project.eu