Satisfying Real-Time Requirements of Multicore Software on ARINC 653: The Issue of Undocumented Hardware

Bjorn Andersson

Dionisio de Niz

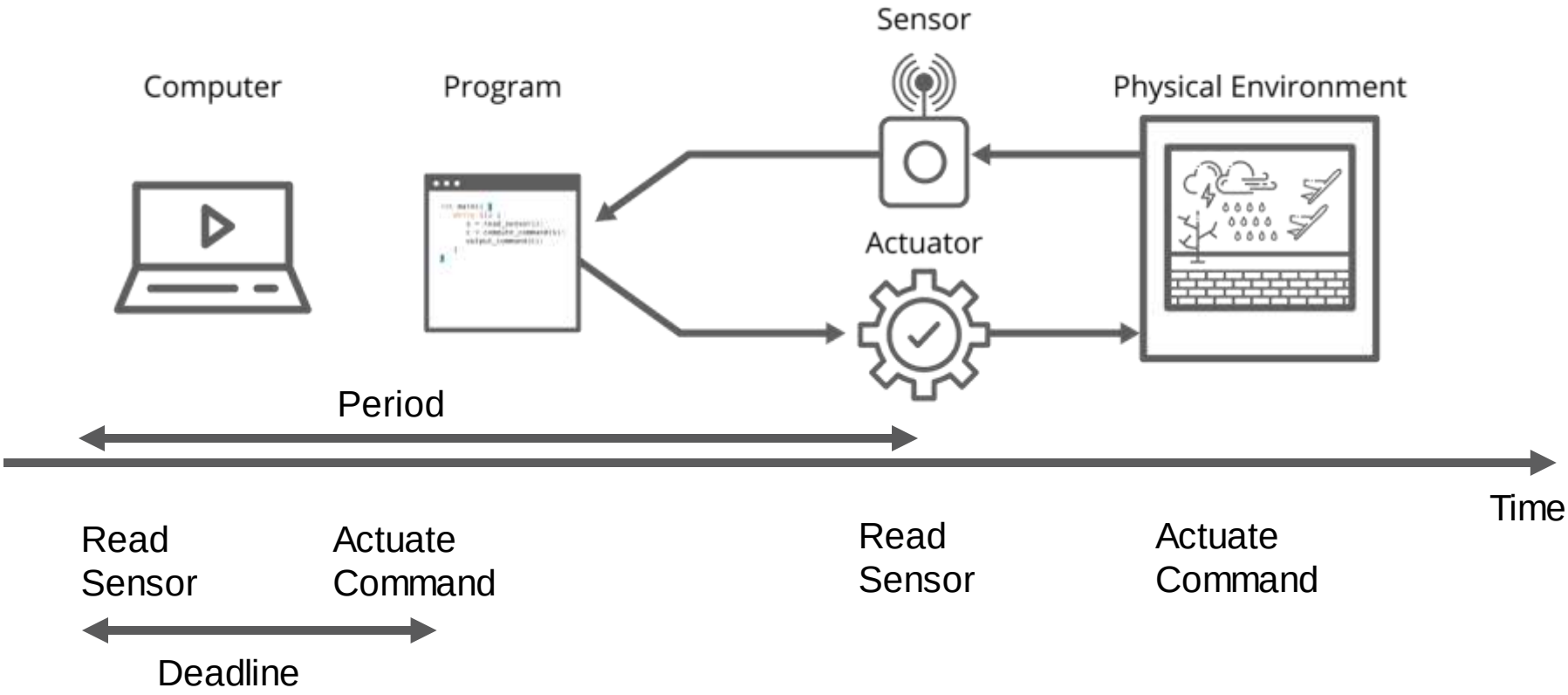
Mark Klein

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

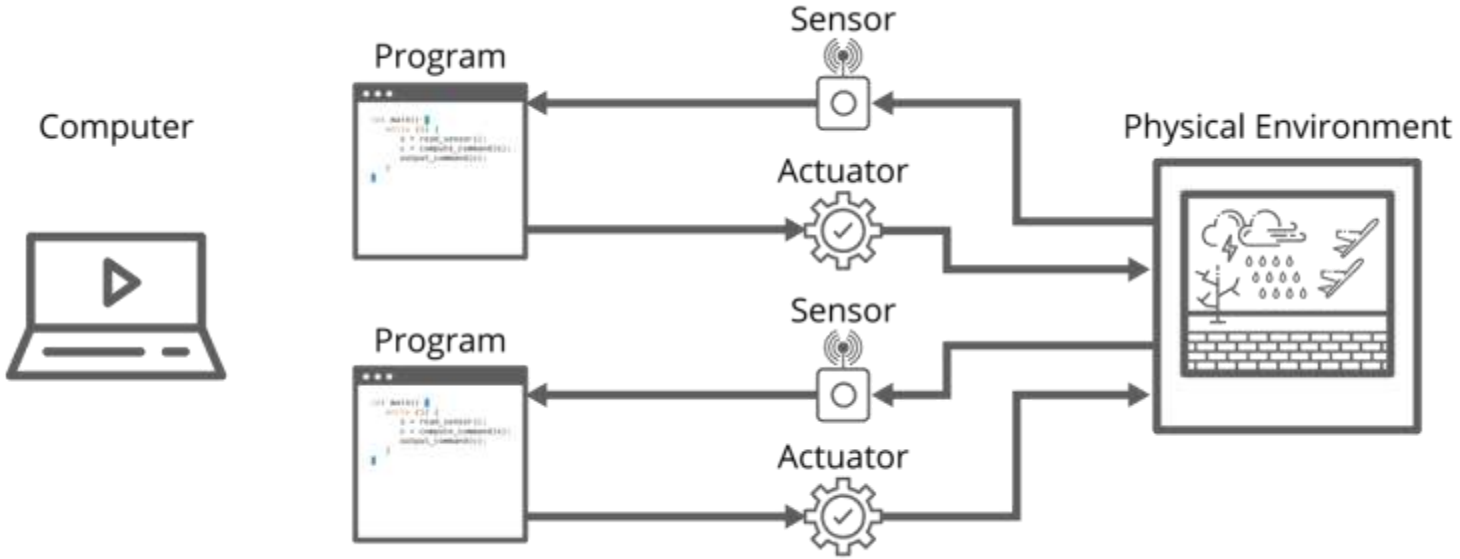
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM22-0733

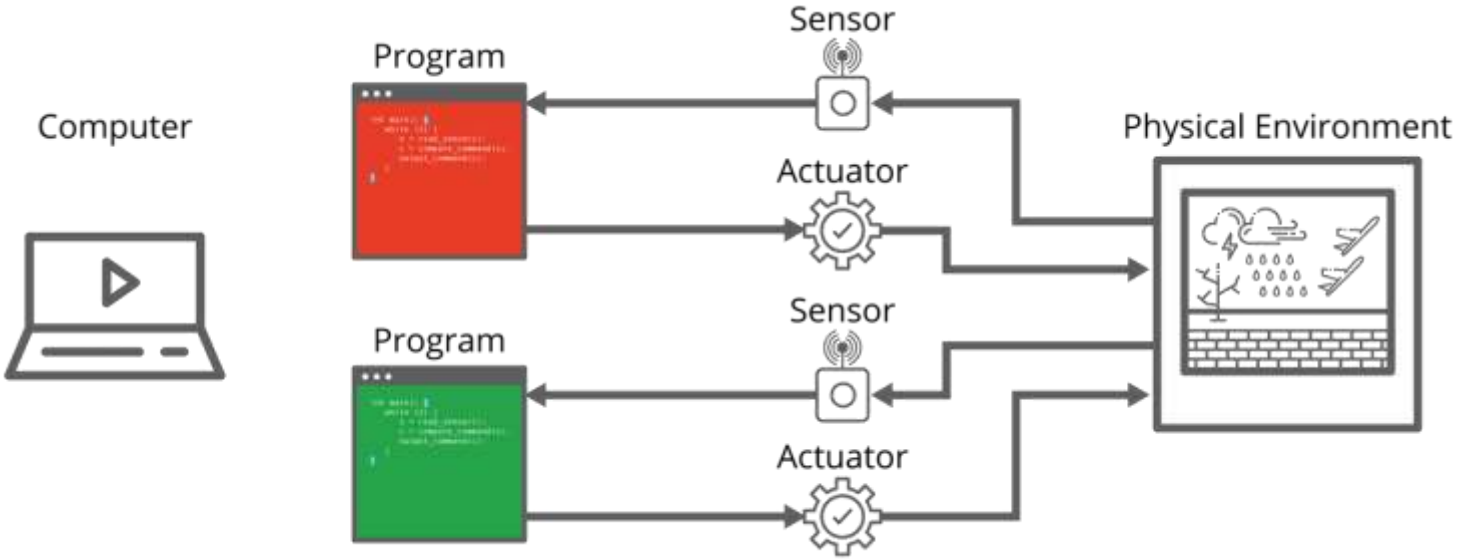
Single-core real-time systems



Single-core real-time systems



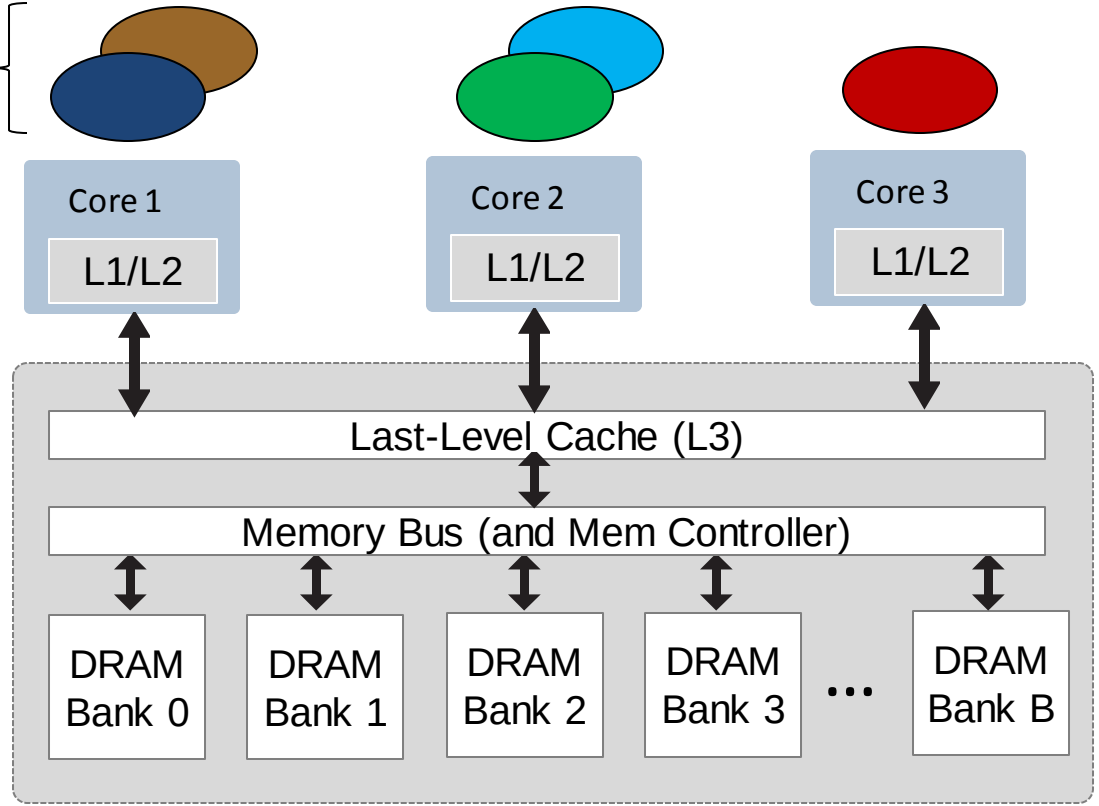
Single-core real-time systems



Multicore real-time systems

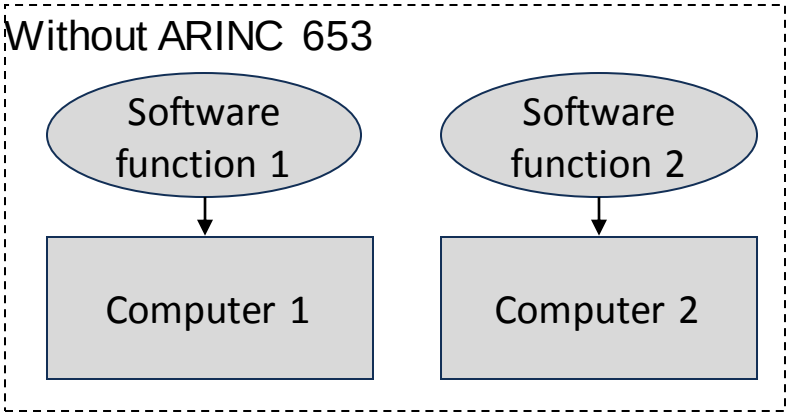
Issues

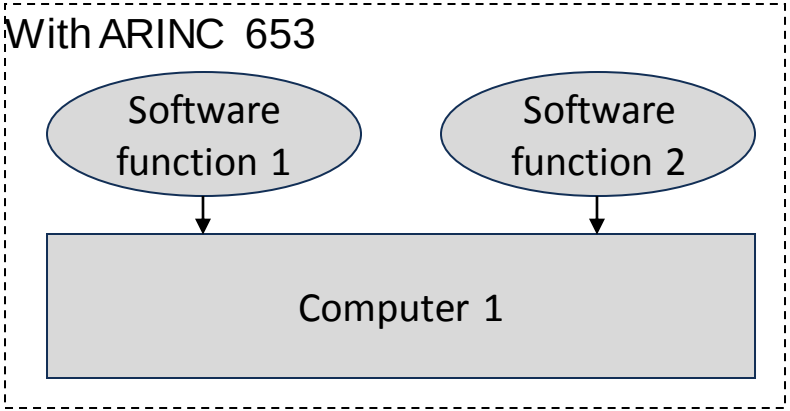
- Shared hardware resources impact timing.
- 103 times slowdown has been observed [Yun15].
- Most current methods cannot deal with undocumented resources.
- Even when resources are documented, current methods can only analyze/manage a small set of them.
- The problem is getting worse:
  * Slowdown increasing
  * More undocumented h/w



ARINC 653

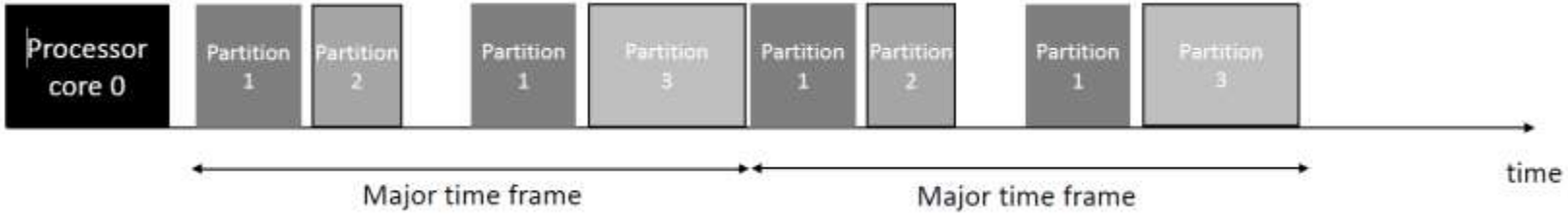
- Standard that facilitates integration of many software subsystems (called partitions) on a single computer





ARINC 653

- Standard that facilitates integration of many software subsystems (called partitions) on a single computer
- ARINC 653 uses a time-triggered scheduler to supply processing time to partitions.
- ARINC 653 initially only supported single-core systems



ARINC 653

- Standard that facilitates integration of many software subsystems (called partitions) on a single computer
- ARINC 653 uses a time-triggered scheduler to supply processing time to partitions.
- Today, ARINC 653 supports multicore systems

ARINC 653

- Standard that facilitates integration of many software subsystems (called partitions) on a single computer
- ARINC 653 uses a time-triggered scheduler to supply processing time to partitions.
- Today, ARINC 653 supports multicore systems

2.2.1 Core Modules with Multiple Processor Cores

This standard includes support for the ARINC 653 services to be utilized with a core module that contains a single or multiple processor cores. With multiple processor cores, additional scheduling paradigms are feasible on a core module:

- Multiple processes within a partition scheduled to execute concurrently on different processor cores.
- Multiple partitions scheduled to execute concurrently on different processor cores.

ARINC 653

- Standard that facilitates integration of many software subsystems (called partitions) on a single computer
- ARINC 653 uses a time-triggered scheduler to supply processing time to partitions.
- Today, ARINC 653 supports multicore systems

2.2.1 Core Modules with Multiple Processor Cores

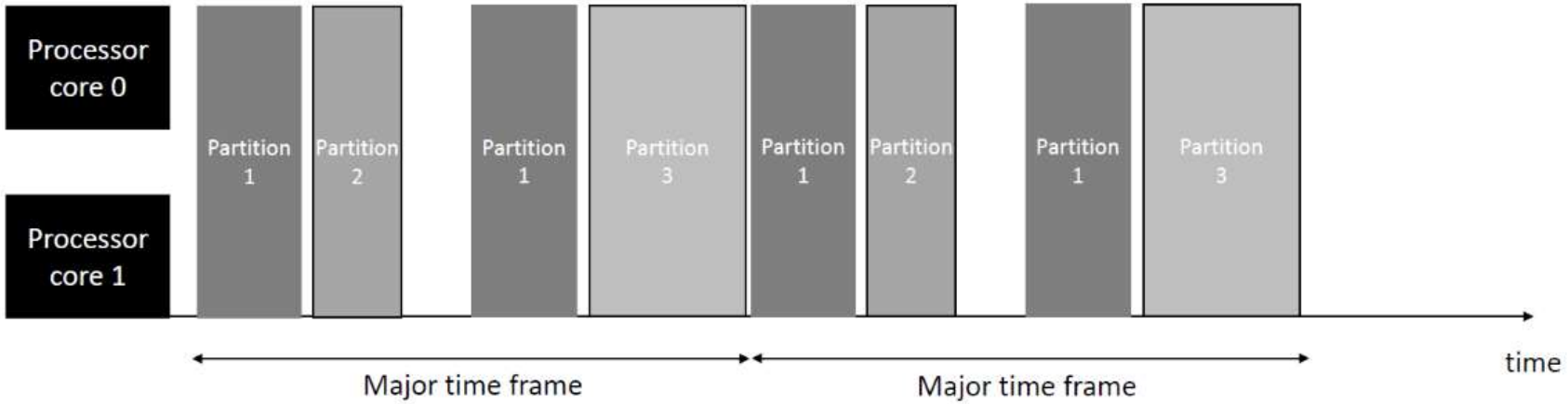
This standard includes support for the ARINC 653 services to be utilized with a core module that contains a single or multiple processor cores. With multiple processor cores, additional scheduling paradigms are feasible on a core module:

- Multiple processes within a partition scheduled to execute concurrently on different processor cores.
- Multiple partitions scheduled to execute concurrently on different processor cores.

In the rest of this talk, we will assume this paradigm.

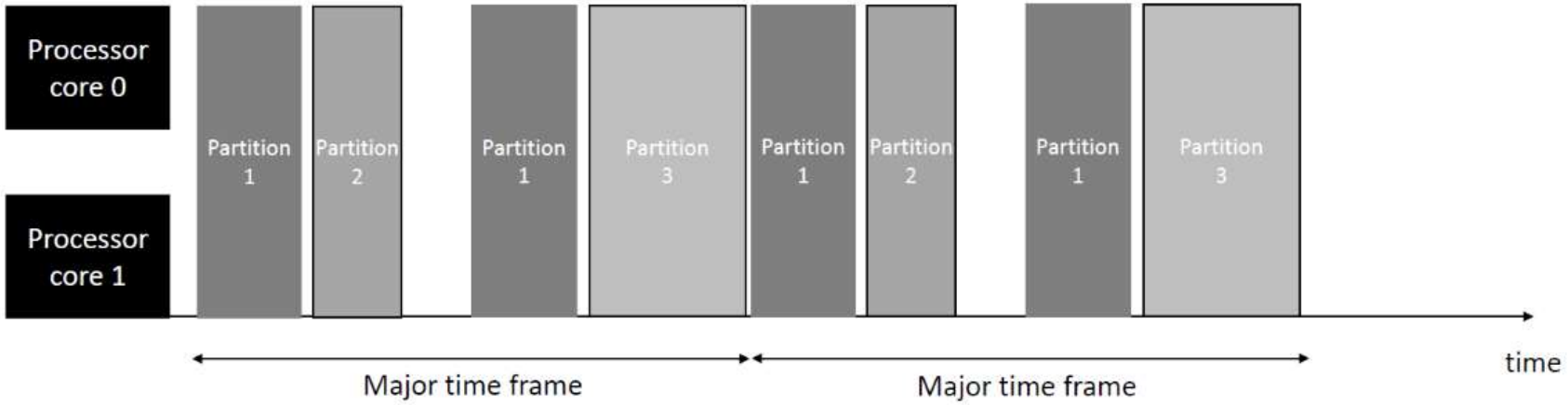
ARINC 653

- Standard that facilitates integration of many software subsystems (called partitions) on a single computer
- ARINC 653 uses a time-triggered scheduler to supply processing time to partitions.
- Today, ARINC 653 supports multicore systems



ARINC 653

- Standard that facilitates integration of many software subsystems (called partitions) on a single computer
- ARINC 653 uses a time-triggered scheduler to supply processing time to partitions.
- Today, ARINC 653 supports multicore systems



- How do we analyze real-time software executing over ARINC 653 on multicore processor with undocumented hardware?

Previous work

This Paper Compared to State-Of-The-Art

Can analyze ARINC 653	Yes	[5,6,7]	This paper
	No		[8]
		No	Yes
		Can analyze undocumented hardware	

References

[5] Y.-H. Lee, D. Kim, M. Younis, and J. Zhou, “Scheduling tool and algorithm for integrated modular avionics systems,” DASC, 2000.

[6] A. Mok, D.-C. Tsou, and R. de Rooij, “The MSP.RTL real-time scheduler synthesis tool,” RTSS, 1996.

[7] N. C. Audsley and A. J. Wellings, “Analysing APEX applications,” RTSS, 1996.

[8] B. Andersson, H. Kim, D. de Niz, M. Klein, R. Rajkumar, and J. Lehoczky, “Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times,” TECS, 2018.

Previous work

In this paper, we adapt [8] for ARINC 653.

This Paper Compared to State-Of-The-Art

Can analyze ARINC 653	Yes	[5,6,7]	This paper
	No		[8]
		No	Yes
		Can analyze undocumented hardware	

References

[5] Y.-H. Lee, D. Kim, M. Younis, and J. Zhou, "Scheduling tool and algorithm for integrated modular avionics systems," DASC, 2000.

[6] A. Mok, D.-C. Tsou, and R. de Rooij, "The MSP.RTL real-time scheduler synthesis tool," RTSS, 1996.

[7] N. C. Audsley and A. J. Wellings, "Analysing APEX applications," RTSS, 1996.

[8] B. Andersson, H. Kim, D. de Niz, M. Klein, R. Rajkumar, and J. Lehoczky, "Schedulability Analysis of Tasks with Co-Runner-Dependent Execution Times," TECS, 2018.

Previous work

This Paper Compared to State-Of-The-Art

Can analyze ARINC 653	Yes	[5,6,7]	This paper
	No		[8]
		No	Yes
		Can analyze undocumented hardware	

[8] uses a model that describes the effect of sharing of hardware resources in the memory system without actually modelling the resources in the memory system.

$\Rightarrow$

allows analysis of software executing on undocumented hardware.

Main result in this paper

Theorem 1: If $\forall \tau_i \in \tau \quad \exists t \in [0, D_i]$ $reqlpARINC653(\tau, \Pi, PART, i, t) \leq sbf_I(\tau, \Pi, PART, i, t)$, then the system is schedulable.

This is a minor adaptation from previous work [8].

This is new in this paper.

Main result in this paper

Theorem 1: If $\forall \tau_i \in \tau \quad \exists t \in [0, D_i]$ $reqlpARINC653(\tau, \Pi, PART, i, t) \leq sbf_l(\tau, \Pi, PART, i, t)$, then the system is schedulable.

How much processing time is requested

How much processing time is supplied

Main result in this paper

Theorem 1: If $\forall \tau_i \in \tau \quad \exists t \in [0, D_i]$ $reqlpARINC653(\tau, \Pi, PART, i, t) \leq sbf_I(\tau, \Pi, PART, i, t)$, then the system is schedulable.

The ARINC 653 standard can be interpreted in different ways. This yields different bounds on supply of processing time.

Main result in this paper

Theorem 1: If $\forall \tau_i \in \tau \quad \exists t \in [0, D_i]$ $reqlpARINC653(\tau, \Pi, PART, i, t) \leq sbf_I(\tau, \Pi, PART, i, t)$, then the system is schedulable.

The ARINC 653 standard can be interpreted in different ways. This yields different bounds on supply of processing time.

I allows you to specify which interpretation you want to use.

Main result in this paper

Theorem 1: If $\forall \tau_i \in \tau \quad \exists t \in [0, D_i]$ $reqlpARINC653(\tau, \Pi, PART, i, t) \leq sbf_I(\tau, \Pi, PART, i, t)$, then the system is schedulable.

For details: see paper.

Conclusion

It is possible to satisfy real-time requirements of multicore software on ARINC 653 and deal with the issue of undocumented multicore.