POSIX modeling in SystemC

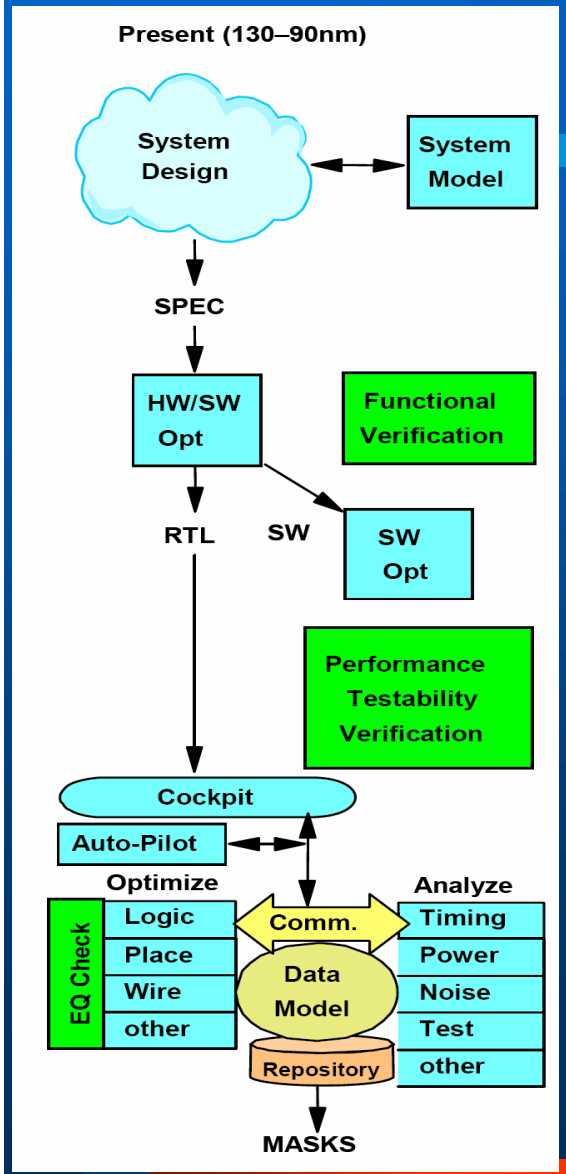
Hector Posadas, Jesús Ádamez, Pablo Sánchez &
Eugenio Villar
University of Cantabria

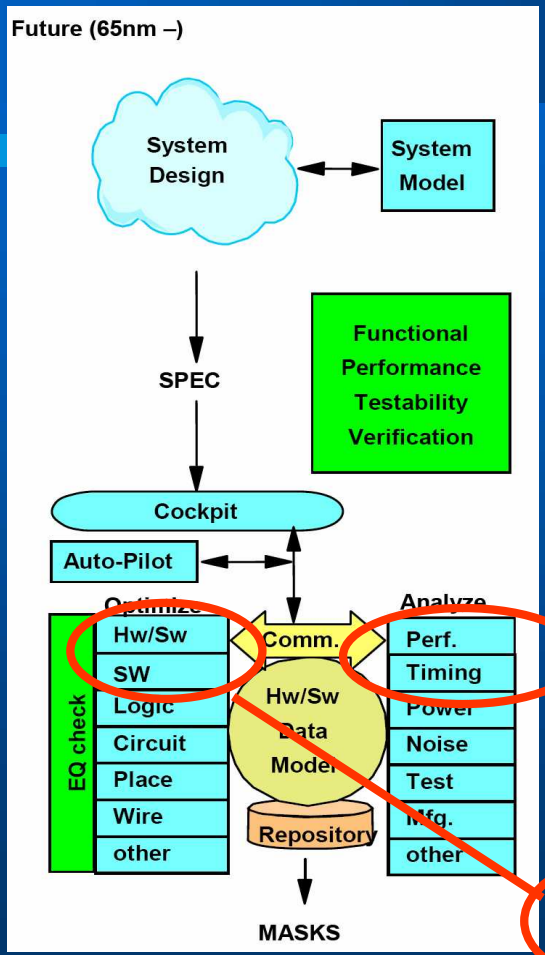
Francisco Blasco
DS2

Outline

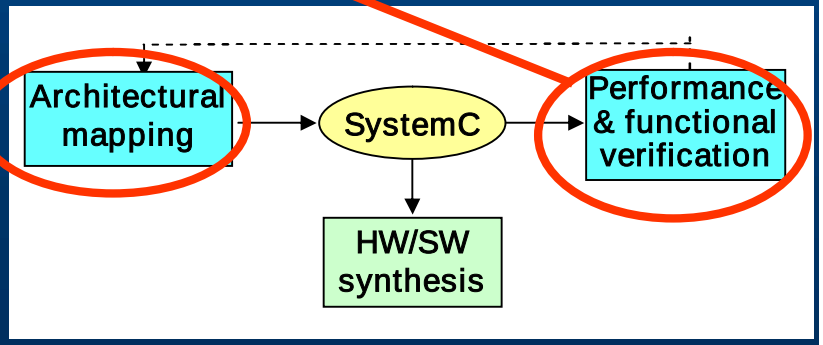
- Motivation
- POSIX modeling
  - Functionality
  - Timed simulation
- Results
- Conclusions

ITRS: Design flow evolution



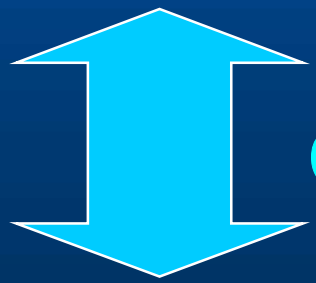


- New SystemC framework
  - SW and HW refined together



SW modeling nowadays

- Current industrial techniques not adequate for HW/SW co-design flow
  - Commercial RTOS simulators
    - Functional but untimed models
    - Difficult HW integration
    - Limited performance estimations



Our approach

  - ISS is too slow for complex systems
    - Refiment process slow

HW & SW modeling in SystemC

- SystemC used for HW design
- SystemC lacks a standard model of SW
  - Execution of SW code is untimed
  - Actual task execution order difficult to model in SystemC
    - Scheduling capabilities cannot be directly modeled
  - Refined SW code usually includes system calls
    - SystemC does not provide an RTOS API

State of the art

- Abstract RTOS models in SpecC and SystemC
  - Specific RTOS functions
    - Simulation code not valid for implementation
  - Low-level, timing behavior is not adequately modeled
    - “wait” statements in predefined points
      - Time-slicing
      - Preemption
      - Interrupts…
    - Communication issues
      - Priority inheritance

SW modeling in SystemC

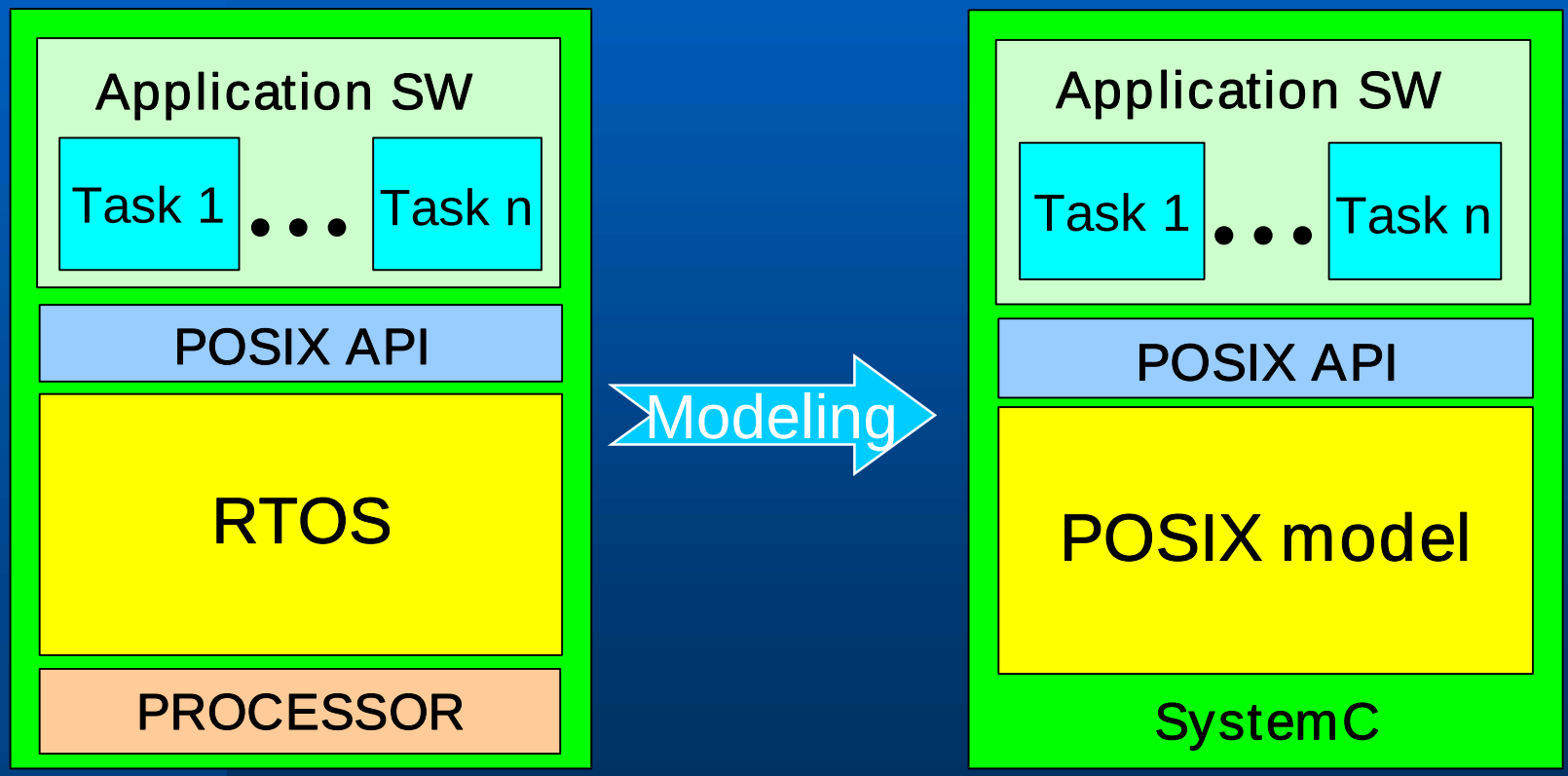
- A complete SystemC RTOS model is required
  - All common RTOS features for optimum refinement
    - Scheduling & preemption
    - Communication & synchronization
    - Timing functions: clocks, timers
    - …
  - Refined SW has to be directly implementable
    - Real API

SW modeling in SystemC

- Complete, standard RTOS API required
  - POSIX is the most common RTOS API
  - Contains all elements required for optimum refinement
- A time estimation technique required
  - Not part of this work
  - PERFidy used [1]
  - Any other dynamic technique possible

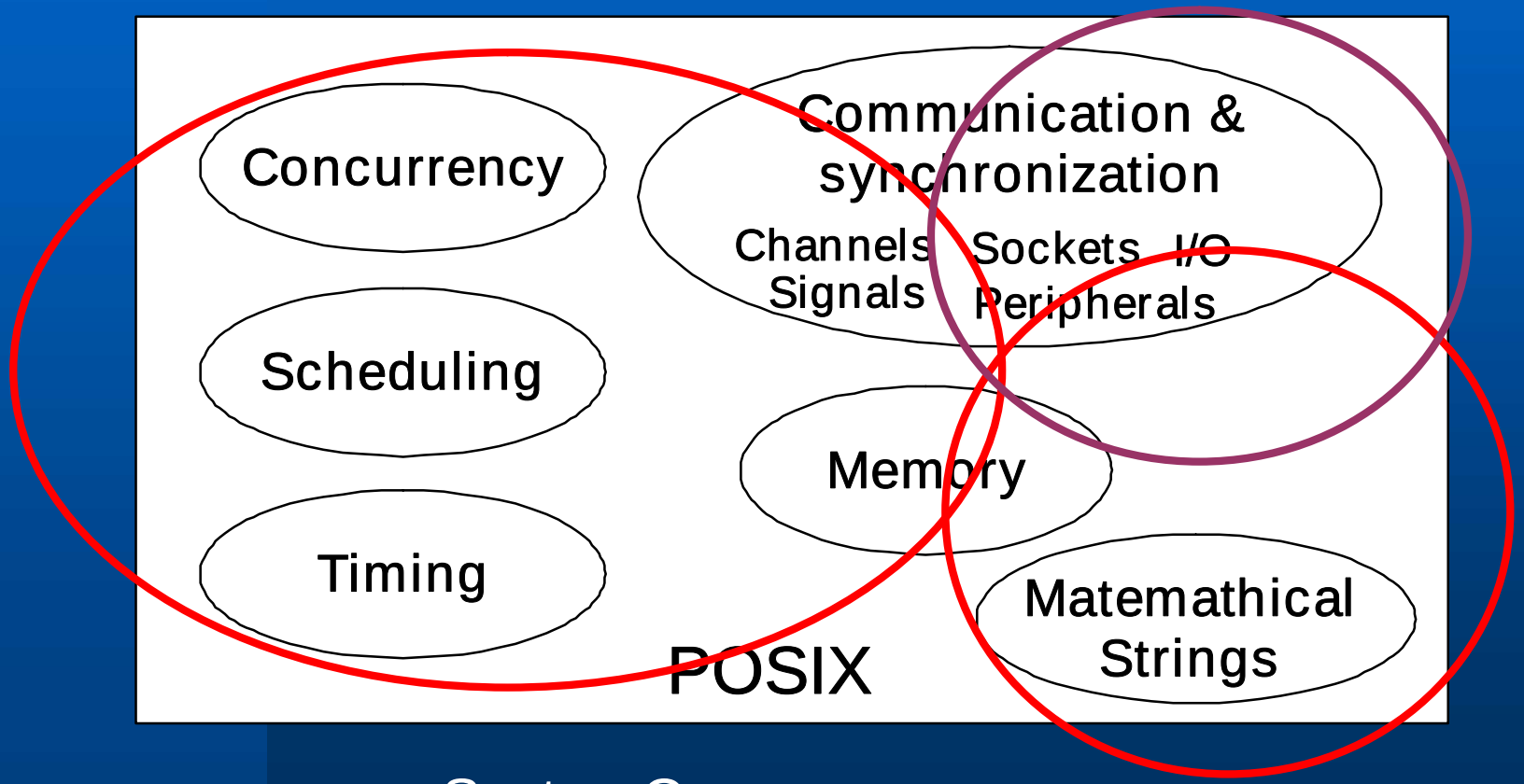
[1] H. Posadas, P. Sanchez, E. Villar, F.Blasco:"System-level performance estimation in SystemC", DATE'04

SW modeling


Application SW
Task 1
...
Task n
POSIX API
RTOS
PROCESSOR
Modeling
Application SW
Task 1
...
Task n
POSIX API
POSIX model
SystemC

POSIX standard

Specific drivers required



Host RTOS facilities

- SystemC allows using host pthreads to implement SC_THREADs
- Most POSIX features cannot be modeled using host RTOS
  - The SystemC simulation has only one execution flow
  - SystemC is one host process
    - Signals, waitpid, fork, etc cannot be modeled
  - Clocks, timers, timeouts depends on host simulation time, not on estimated time for the target platform

SystemC facilities

- The RTOS model extends the SystemC kernel
  - The kernel is not modified
    - Ensures compatibility with different SystemC versions
  - A reduced set of SystemC elements is needed
    - SC_THREAD to provide concurrency
    - “wait” and “notify” to stop and resume the threads
      - Needed to model the scheduling and synchronization

Concurrency modeling

- Parallelism
  - SC_THREAD
  - Threads and processes
    - Limitations when modeling separate memory spaces
  - Dynamic creation of threads and processes
- All SC_THREADs runs in parallel
  - Tasks in the same processor cannot run in parallel
    - Scheduling

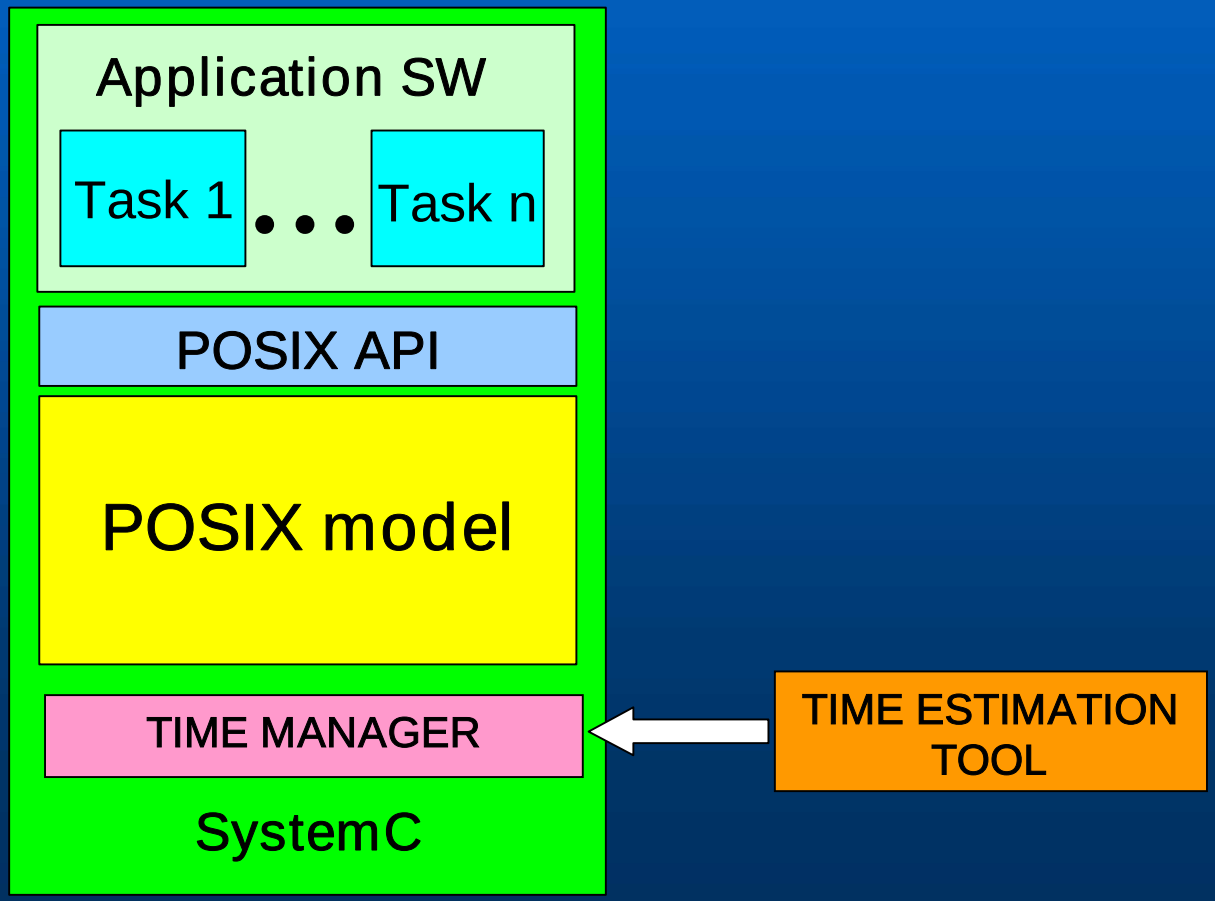
Scheduling in POSIX

- POSIX defines several policies
  - Round Robin, FIFO, Sporadic Server
    - Different exit procedure
    - FIFO
      - process stopped in a synchronization point
    - RR and SS
      - a time slice is finished or a synchronization point is reached
- Use of priorities

Scheduling modeling

- Scheduling modeled with “wait” and “notify” functions
  - Only the thread with higher priority is not blocked
  - Scheduler execution when running thread releases the processor
- Preemption has to be considered
- Time estimation tool is required
  - Time-slice based policies
  - Time events (sleeps, timeouts, timers)

SW modeling


Application SW
Task 1
. . .
Task n
POSIX API
POSIX model
TIME MANAGER
SystemC
TIME ESTIMATION TOOL

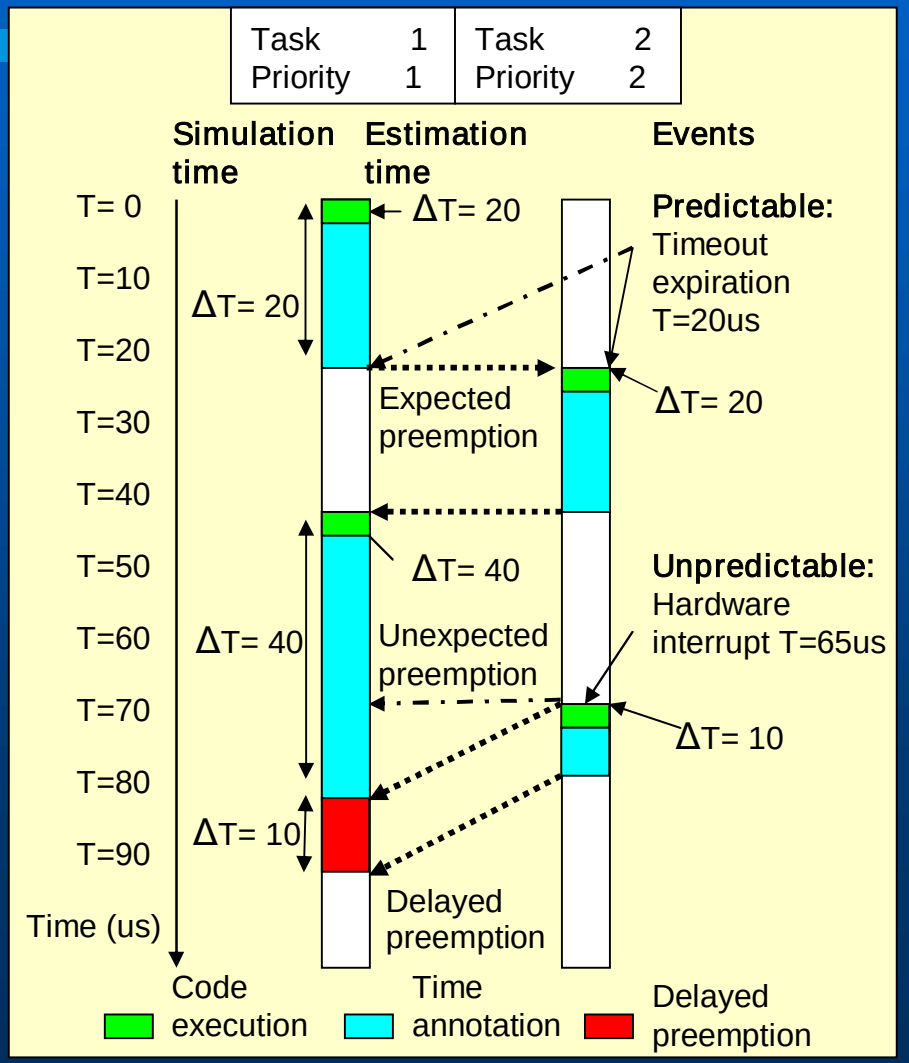
Preemption example

Task 1

B=A-6;
C=A+B;
B*=2;
…
A=C/B;
D=C+1;

Time estimation tool

Time: 20 us



Communication & synchronization

- Required a complete set of communication and synchronization mechanisms for SW refinement
  - Mutexes, semaphores, conditional variables, message queues, etc
    - Wide range of options: Priority inheritance
    - “wait” and “notify” functions of SystemC are used to model synchronization
  - Asynchronous events: POSIX Signals
    - An specific SC_THREAD is required to execute signal handlers
      - signal handling can require a temporal new thread

Clocks and timers

- POSIX standard defines several clocks
  - Real-time, monotonic
    - Are modeled with the SystemC time control
    - Timers, alarms & sleeps can be modeled with “wait(t)”
  - Thread clocks, process clocks
    - The clock values are updated with the time estimation
    - A new SC_THREAD is required to execute events depending on this clocks (timers)

Results

- The POSIX model has been used to simulate a GSM coder
  - ETSI - EN 301 245
  - 13.500 code lines
  - Manually parallelized
    - 9 concurrent processes
- Results
  - Verify the RTOS model with the real target RTOS
  - Time estimation errors compared
    - SystemC (without RTOS model) vs. time in target platform
    - SystemC + POSIX model vs. time in target platform

Results

- The error is lower when considering the effect of the RTOS
  - With the same estimation technique

Thread	SystemC (%)	POSIX (%)
pre_filtering	3.45	2.41
homing_frame_test	13.67	0.88
frame_lsp_func	4.51	0.45
frame_int_tol_fun	0.94	0.36
subframe_coder_fun	3.04	0.23
serializer_fun	6.72	6.36
vad_comp_fun	9.36	3.34
CN_encoder_fun	7.45	5.33
sid_encoding_fun	12.18	7.29
RTOS		1.84

Results

- Overhead of RTOS model is worthless respect gain with ISS

	SC (ms)	SC + P (ms)	ISS (ms)
Simulation time	1,07	1,56	124

Conclusions

- Current version of SystemC can be extended to model a complex RTOS
  - SW refinement can be done in SystemC
- SW refinement over SystemC requires a complete RTOS model
  - Standard API
  - All features present in target RTOS

Conclusions

- Close link Estimation tool – RTOS model
  - Modeling of time-slices & time events
  - RTOS model requires time estimations (process & thread clocks)
  - Estimation results better with RTOS model
- High-level modeling and timed simulation of application SW is possible at source code
  - Fast estimations with enough accuracy

Thank You !