A Simulation Framework for Design of Mixed Time/Event-Triggered Distributed Control Systems with SystemC/TLM

Zhenkai Zhang, Joseph Porter

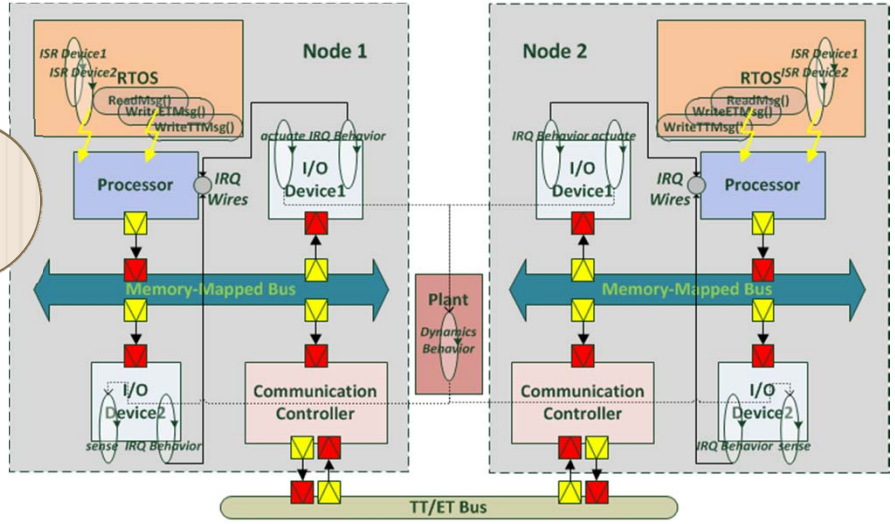
Xenofon Koutsoukos, Janos Sztipanovits

ISIS, EECS

Vanderbilt University

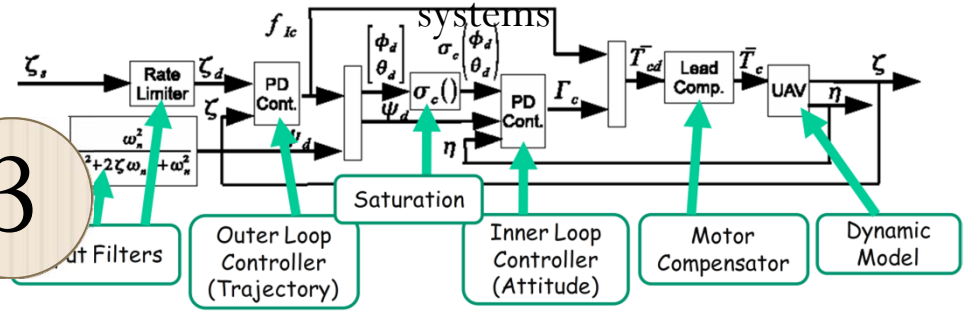
Three Highlights

1



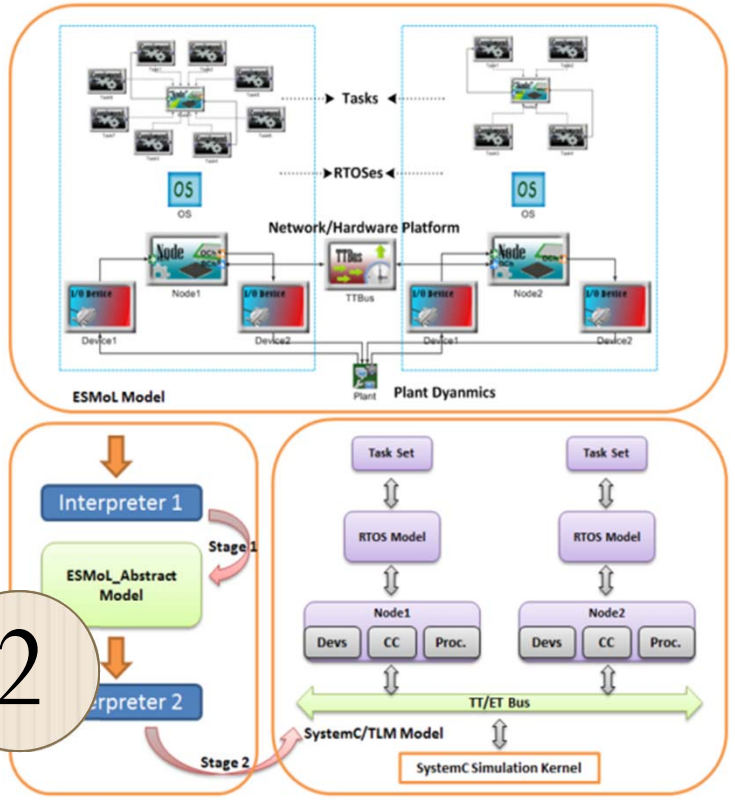
Using abstract computation and communication models to establish a universal simulation framework for mixed TT/ET systems

3



Preliminary results of a Quadrotor flight control system with different communication system options

2



Integration with a model-based design tool to improve the framework's usability

Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Introduction

- Mixed Time/Event-Triggered (TT/ET) Distributed Control Systems
  - Holistic view of TT/ET Distributed Control Systems
    - Computation: TT tasks + ET tasks
    - Communication: TT (static) traffic + ET (dynamic) traffic
  - Factors influencing the decision for choosing TT/ET
    - Predictability (TT)
    - Composability (TT)
    - Fault-tolerance (TT)
    - Flexibility (ET)
    - Resource Utilization (ET)
  - Increasingly found in CPSs
    - Automotive

Introduction

- Mixed TT/ET Distributed Control Systems are hard to model, design, and analyze.
  - There have been efforts to deal with these difficulties
    - [Yokoyama RTAS '05] – modeling
    - [Pop CODES+ISSS '03] – design optimization
    - [Pop ECRTS '03] – schedulability analysis
    - [Phan RTSS '09] – timing analysis of TT/ET mode-switching systems
  - We focus on how to evaluate a designed TT/ET system at early design stages to provide performance feedback via simulation
    - Complement to formal methods
    - When formal analyses are hard to do

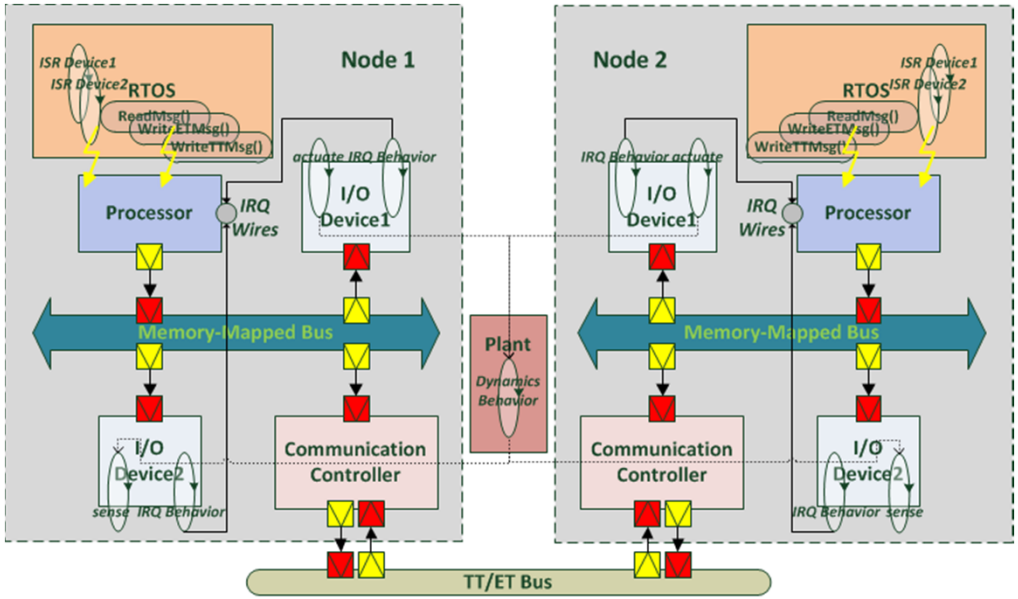
Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Virtual Platform Model

- SystemC/TLM is used to model the virtual platform
  - SystemC has been standardized by IEEE
  - Widely used in hardware/software codesign
  - Supports modeling and simulation at various abstraction levels
    - Transaction level modeling (TLM) for communication
      - A transaction is a set of data being exchanged
      - Focuses on what (rather than how) data is being transferred.
      - Abstracts away some communication details while keeping certain accuracy

Virtual Platform Model



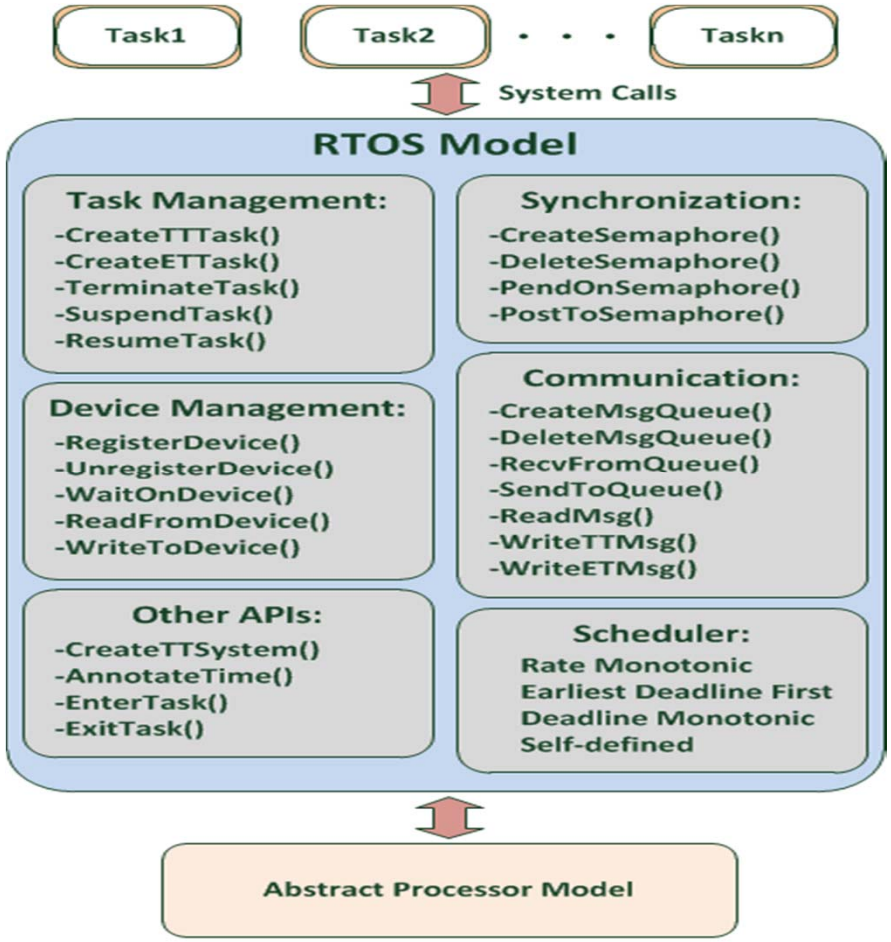
- TT/ET computation behavior is captured by an abstract RTOS model
- TT/ET communication behavior is captured by an abstract TT/ET communication system model
- Platform is formed by connecting abstract hardware models

Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

RTOS Model

- Provides services to upper TT/ET tasks via system calls

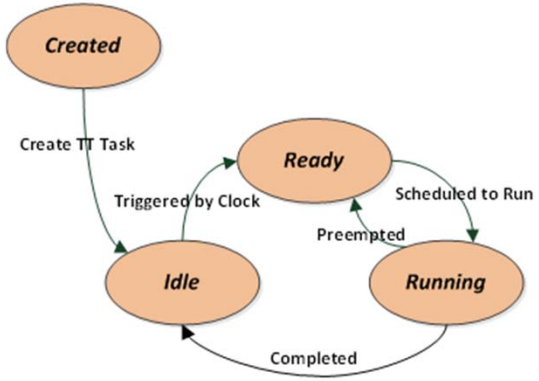


RTOS Model

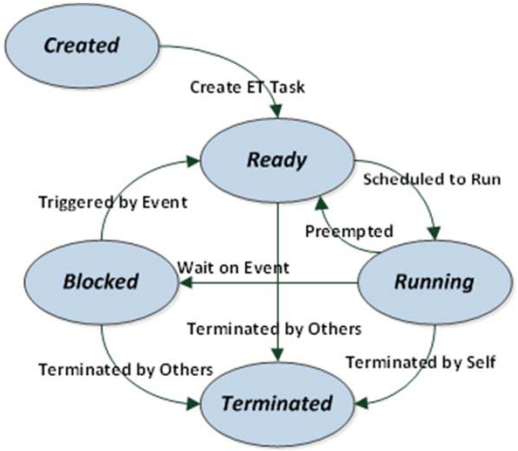
- Task management
  - A task corresponds a SystemC *SC_THREAD* process
    - Create TT task
    - Create ET task
  - Each process pends on its own *sc_event* object controlled by scheduler
  - Adding timing annotations in the task process to model execution time
    - Zero simulation time if no explicit timing annotation
    - Timing annotation API ensures synchronization

RTOS Model

- Scheduling
  - Scheduling Policies
    - RM
    - EDF
    - DM
  - A single ready queue
    - TT task enters *Ready* state statically according to a schedule table
    - ET task enters *Ready* state dynamically when the event happens



Time-Triggered Task State Transitions



Event-Triggered Task State Transitions

RTOS Model

- Inter-task communication
  - On one node
    - TT – TT tasks: shared memory
    - ET – ET tasks: shared memory with semaphore synchronization
    - TT – ET tasks: message queue
      - TT side has an agent – state-message keeper (non-blocking read)
      - ET side uses blocking read
  - On different nodes
    - Message passing
      - TT message
      - ET message

RTOS Model

- Interrupt handling
  - SystemC has disadvantages for interrupt handling modeling
    - Non-interruptible *wait-for-delay* time advance
    - Non-preemptive simulation processes
  - We use *wait-for-event* other than *wait-for-delay* to advance execution time
    - An *event-OR-list* is composed of a timeout event and an interrupt event
    - The timeout event will be notified after the given delay passes if no interrupt happens - the amount of execution time is finished.
    - When an interrupt occurs, the interrupt event is notified and the timeout event's notification is canceled.
    - Processes with higher priorities will run
    - New timeout will be calculated by subtracting the execution time already passed.

Outline

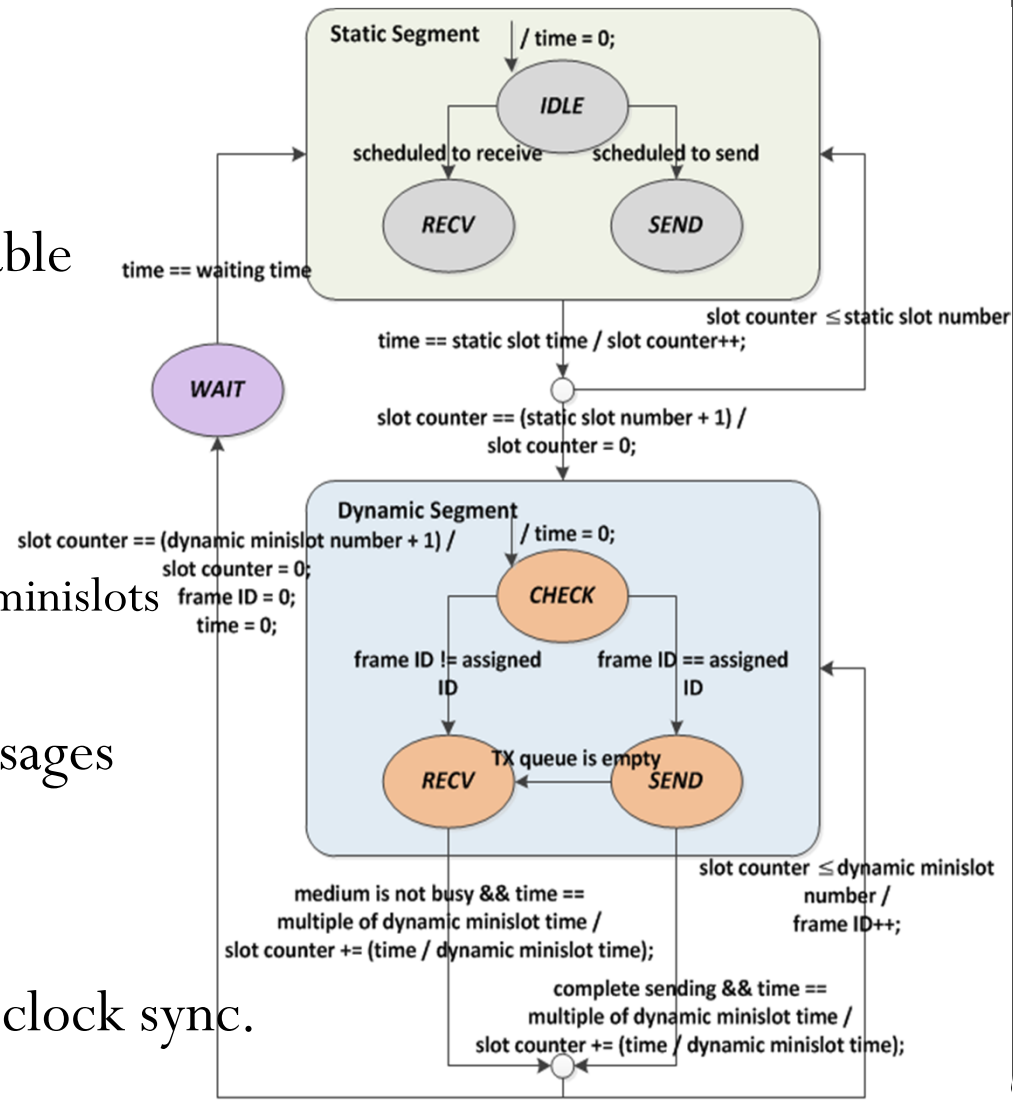
- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Communication System Model

- Abstract communication system model is based on FlexRay
- Data granularity – message level
- Communication controller model + TT/ET bus model
  - Controller transmits/receives messages
  - Controller is both an initiator and a target for TT/ET bus transactions
  - TT/ET bus relays transactions to controllers
- Bus communication is organized in cycles
  - TT static segment
  - ET dynamic segment
  - Waiting segment

Communication System Model

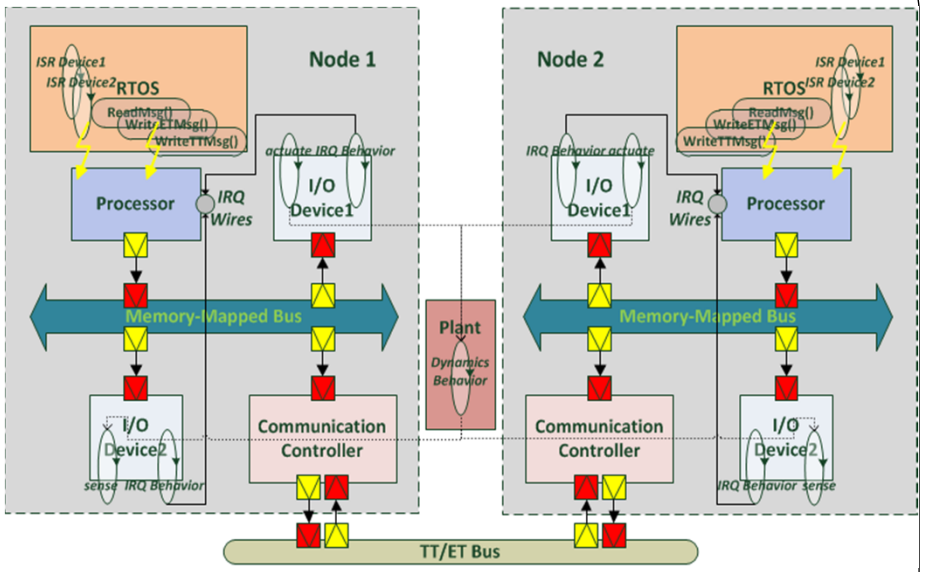
- TT static segment
  - TDMA
  - A predefined schedule table
- ET dynamic segment
  - Flexible TDMA
    - Minislots
    - Each DYN slot: varying #minislots
    - Frame ID as arbitration
  - A queue of dynamic messages
    - Sorted by priorities
- Waiting segment
  - Mimics time elapsed for clock sync.



Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Hardware Model



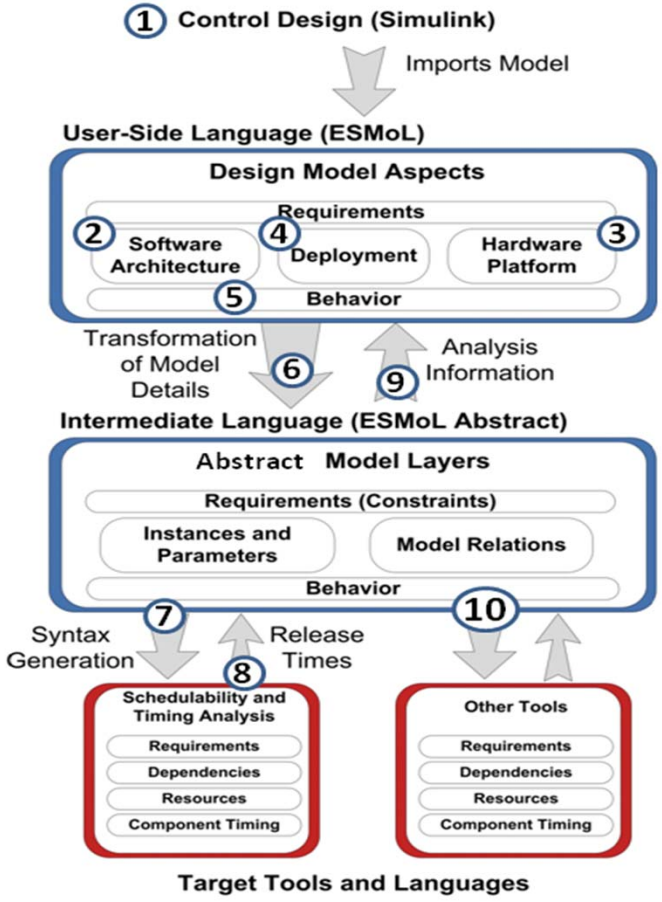
- Interfaces
  - HW – HW: TLM sockets
  - HW – PHY: shared variables
- Abstract processor model
  - Hardware abstraction level (HAL) model + abstract RTOS model
    - Communication with other peripherals through TLM sockets
    - Collect interrupt requests (IRQs)
- I/O devices (sensors and actuators)
  - Periodic IRQ behavior
  - Sporadic IRQ behavior
  - *SC_THREAD* process to interact with the plant model
- Models are connected through buses

Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

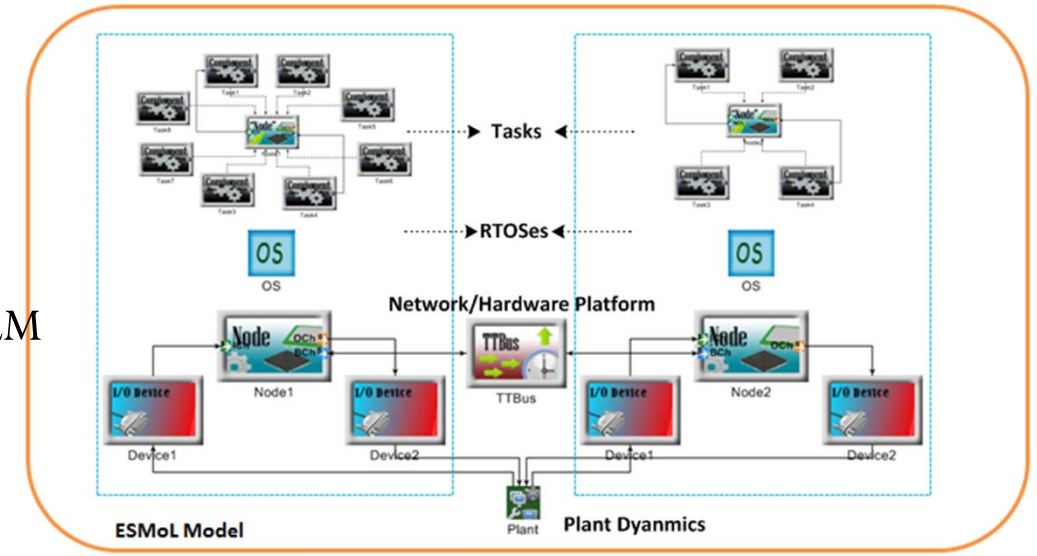
ESMoL Design Flow

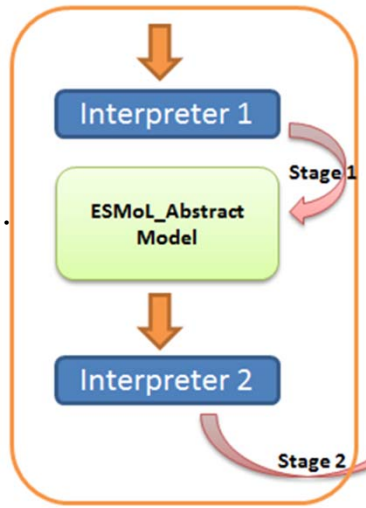
- ESMoL (Embedded System Modeling Language) is used to specify the TT/ET distributed control systems formally
  - Simulink model (functionality)
    - Imported into ESMoL
  - Logic software architecture model
    - Data dependencies
  - Platform model
    - Hierarchical hardware platform
  - Deployment model
    - Task & message mapping
  - Timing model
    - Trigger type (TT/ET)
    - WCET, deadline, etc.

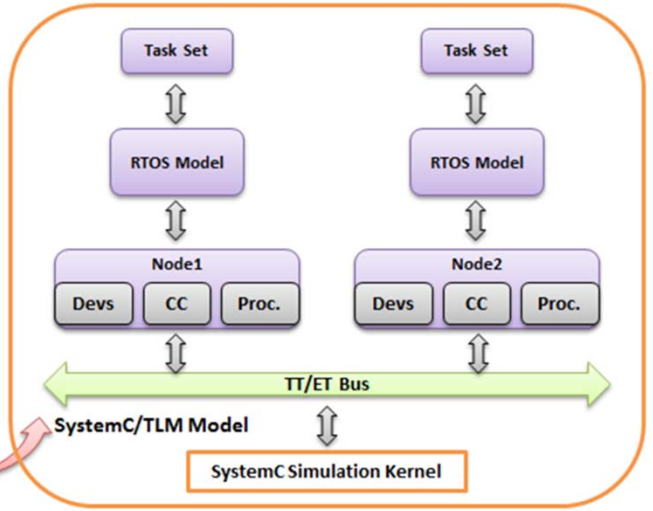


Model-Based Approach

- Two stages transformations
  - Interpreter 1
    - ESMoL => ESMoL_Abstract
  - Interpreter 2
    - ESMoL_Abstract => SystemC/TLM
      - UDM + Google Ctemplate
        - Task set class template
          - Deployment
        - Task config template
          - TT/ET, Msg, Timing, etc.
        - RTOS config template
          - Scheduling method, table, etc.
        - CC config template
          - Schedule table
      - RTW
        - Task functions
        - Plant with fixed step solver





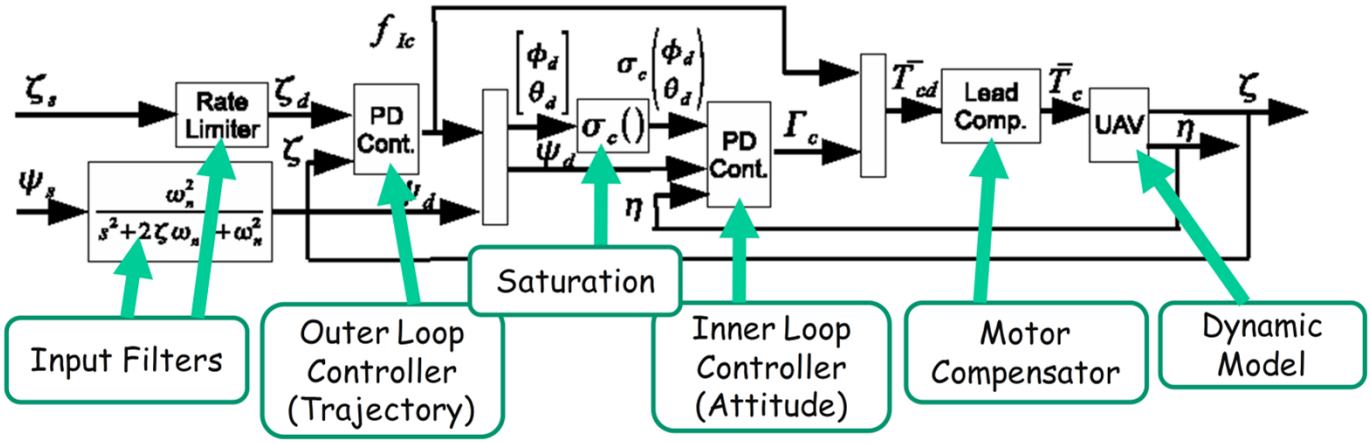


Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Case Study

- Quadrotor Flight Control System



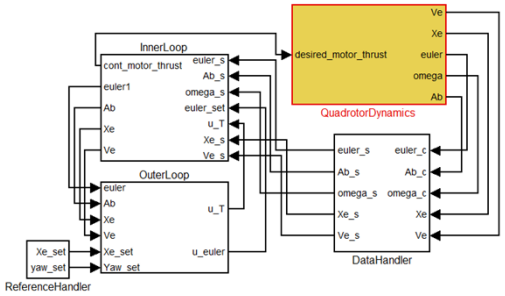
  - Two linear proportional derivative controllers
    - Outer loop: a "slow" PD inertial controller
    - Inner loop: a "fast" PD attitude controller
  - Four tasks
    - ReferenceHandler, OuterLoop, DataHandler, InnerLoop

Case Study

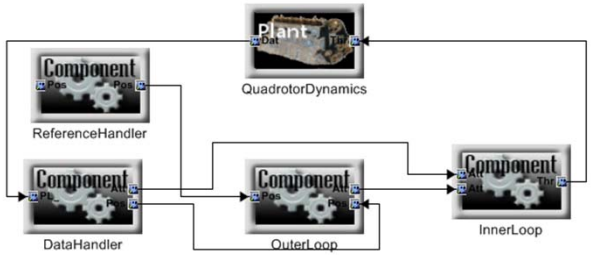
- Fixed design decisions
  - Two nodes – based on PXA255 and ATmega128
  - Sampling period: 20ms
  - On PXA255, RM is used
    - ReferenceHandler, WCET 50µs, Relative deadline 5ms, ET
    - OuterLoop, WCET 1.6ms, Relative deadline 2ms, TT at 12ms
  - On ATmega128, EDF is used
    - DataHandler, WCET 200µs, Relative deadline 4ms, ET
    - InnerLoop, WCET 600µs, Relative deadline 1ms, TT at 10ms
  - Two messages via communication system
    - DataHandler sends a msg to OuterLoop with position data (60B)
    - OuterLoop sends a msg to InnerLoop with attitude control data (24B)

Case Study

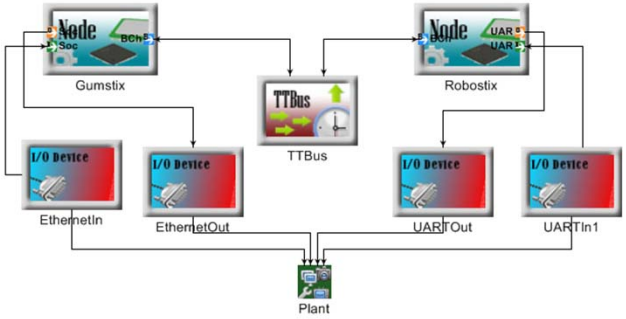
- ESMoL Models



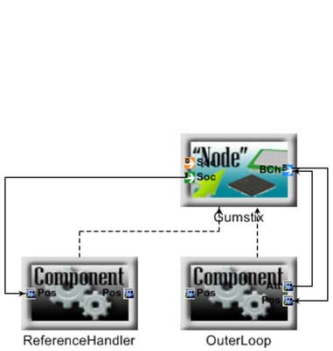
Simulink Model

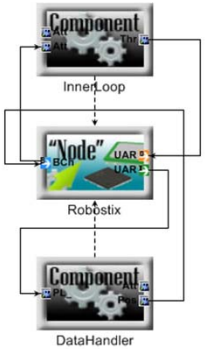


Logic software architecture

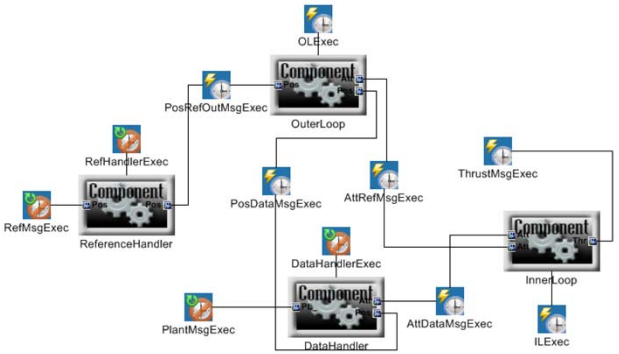


Hardware Platform





Deployment



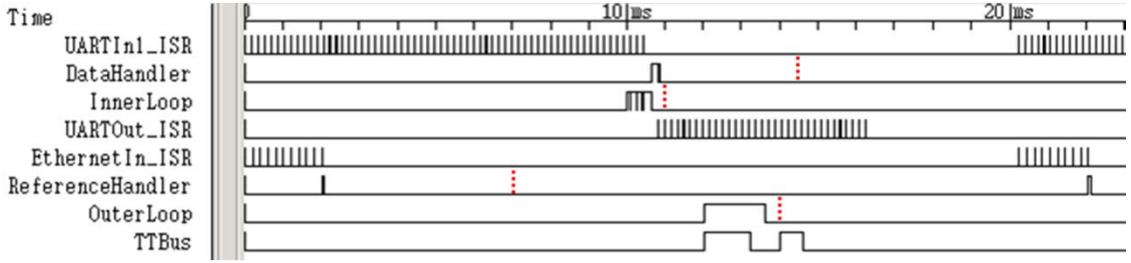
Timing

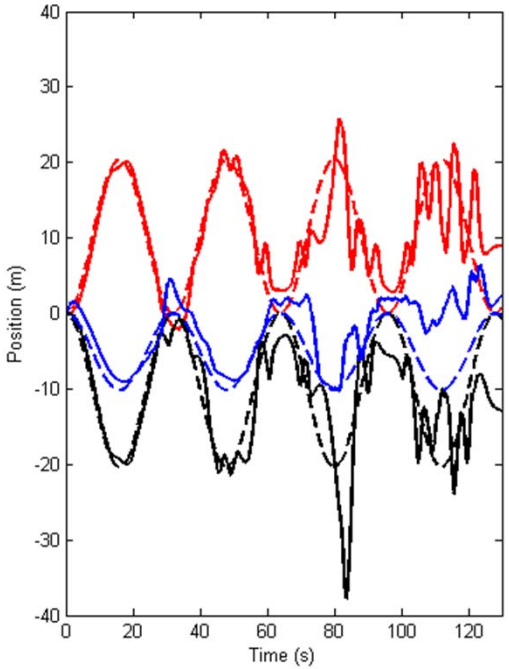
Case Study

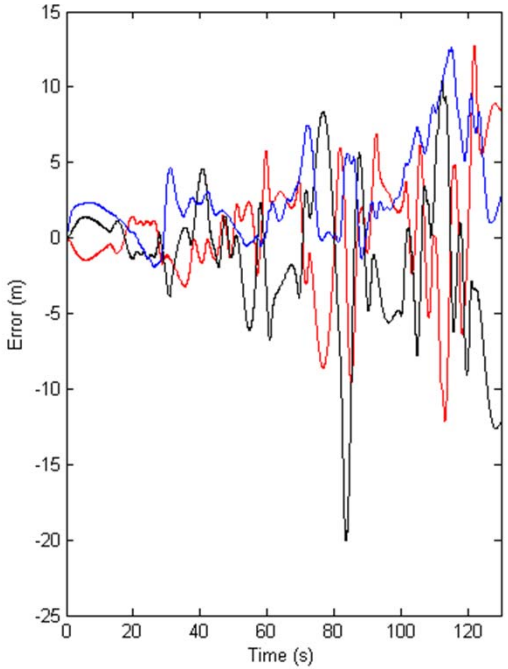
- Two options for the communication system
  - Cheaper one can provide 400kbit/s bandwidth for application
  - The other one can provide 2Mbit/s, but more expensive

Case Study

- The cheaper one with 400kbit/s bandwidth

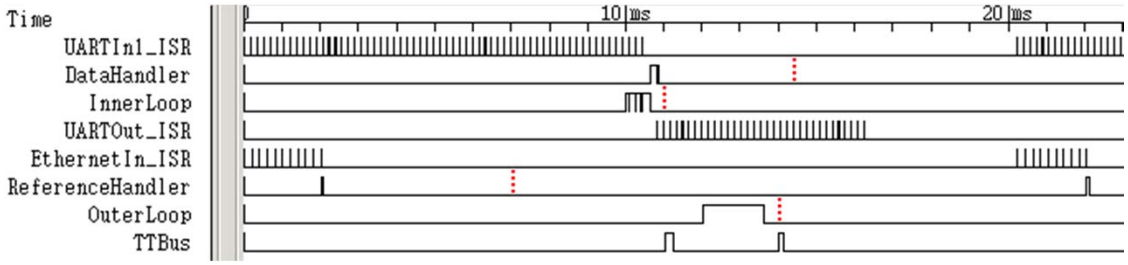


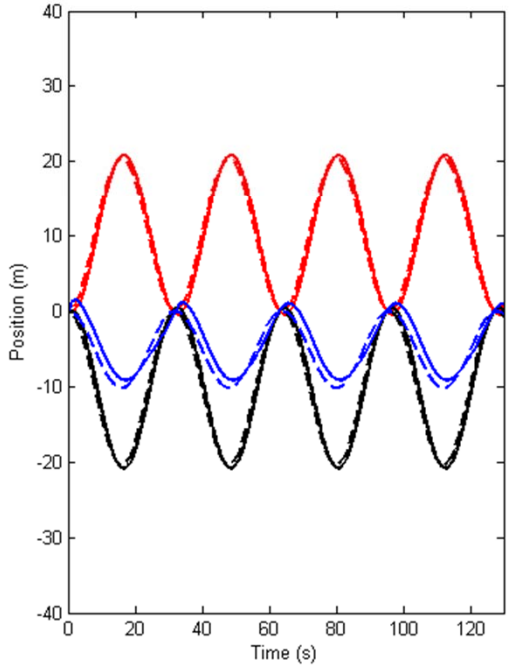


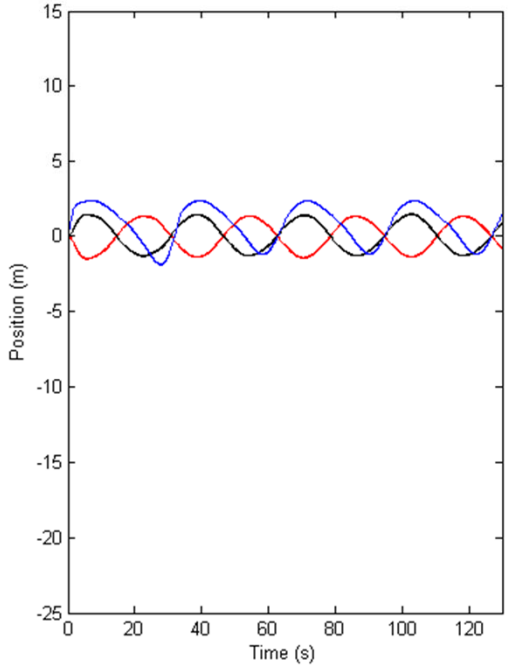


Case Study

- The more expensive one with 2Mbit/s bandwidth







Outline

- Introduction
- Virtual Platform Model
  - RTOS Modeling
  - Communication System Modeling
  - Hardware Modeling
- Integration with Model-Based Design Flow
- Case Study
- Conclusion

Conclusion

- A simulation framework for design of mixed TT/ET distributed control system is introduced
  - Virtual platform model in SystemC/TLM
    - Abstract RTOS model – capture dynamic behaviors of TT/ET tasks
    - Abstract communication system model – capture behaviors of TT/ET communication
    - Abstract hardware model – integrate models as a holistic system model
  - Model-based approach
- Case study on a Quadrotor flight control is presented
- Future work
  - More realistic clock synchronization (done)
  - Other abstraction levels

Questions?

- Thank you!