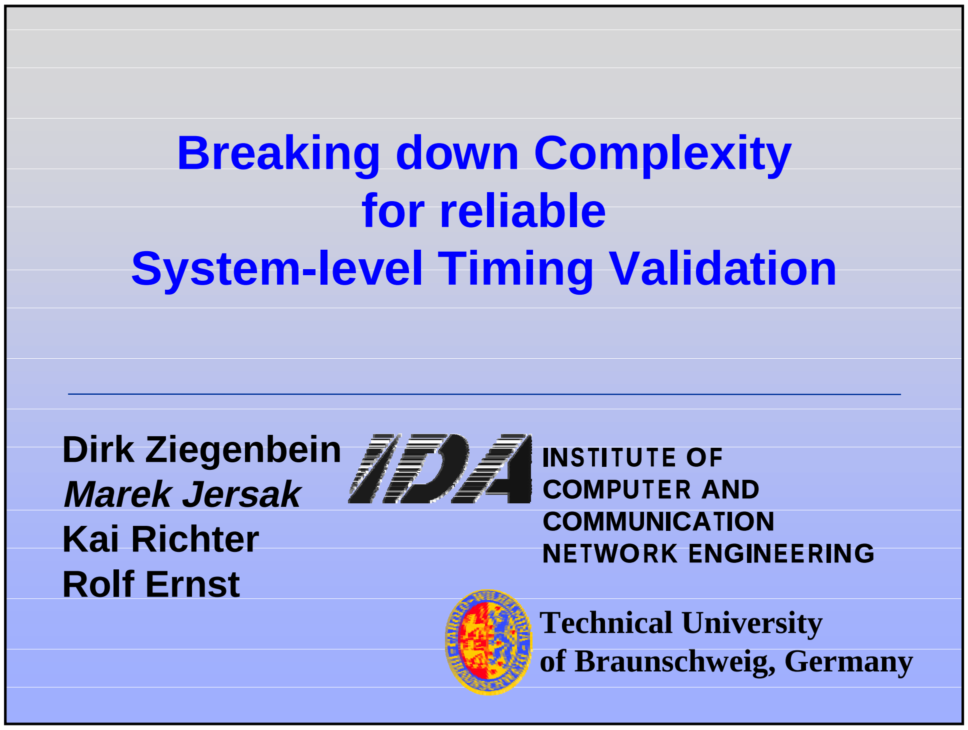
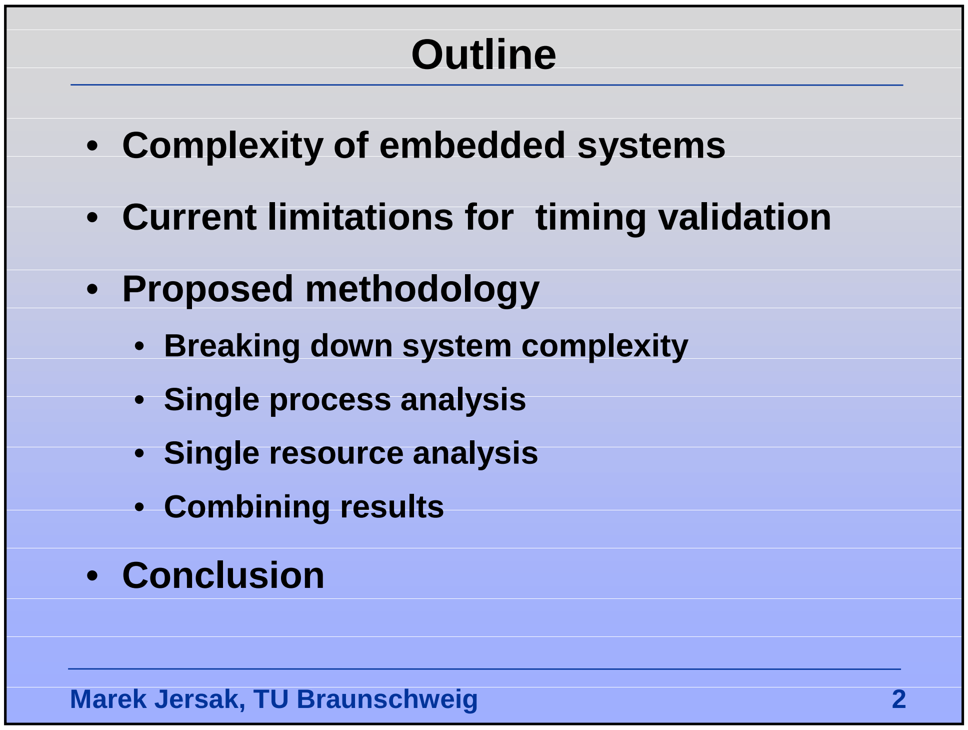


Breaking down Complexity for reliable System-level Timing Validation

Dirk Ziegenbein
Marek Jersak
Kai Richter
Rolf Ernst

INSTITUTE OF COMPUTER AND COMMUNICATION NETWORK ENGINEERING

Technical University of Braunschweig, Germany



Outline

- Complexity of embedded systems
- Current limitations for timing validation
- Proposed methodology
  - Breaking down system complexity
  - Single process analysis
  - Single resource analysis
  - Combining results
- Conclusion

Embedded System Design

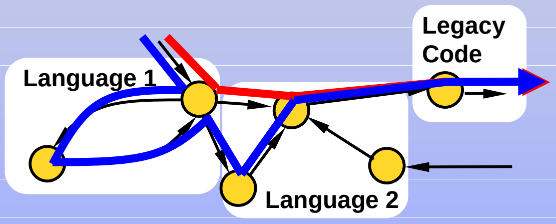
Industry Needs

- High performance, low cost, low power
  - ➔ Specialized languages, optimized architectures
- More and more features, short time-to-market
  - ➔ Platform-based design, application and architecture reuse, IP integration

System size and heterogeneity result in huge system complexity

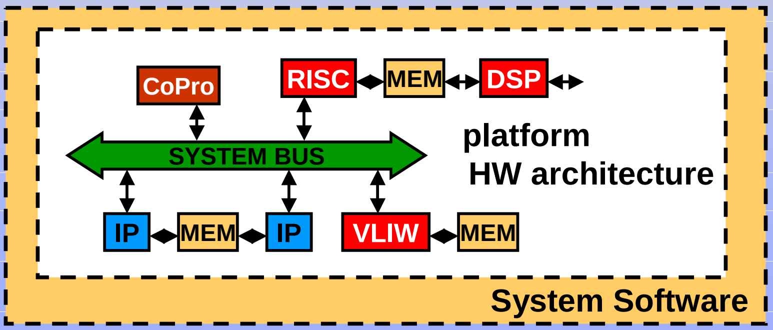
Application Complexity

- Multi-language design, e.g. *Dataflow* (voice processing), *FSMs* (protocol), legacy code
- Complex dependencies (contexts, scenarios)



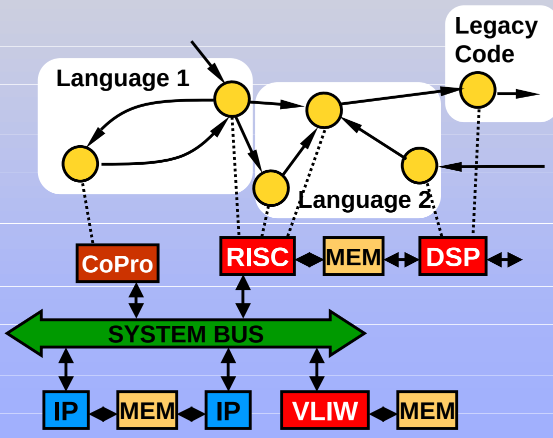
Architecture Complexity

- Heterogeneous platforms and SoC
- Complex on-chip and distributed networks
- System software (RTOS, drivers)



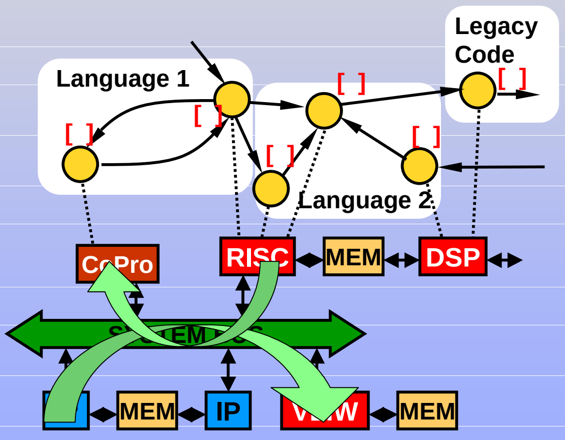
Integration Complexity

- Heterogeneous component and language integration [VSIA, Accellera]



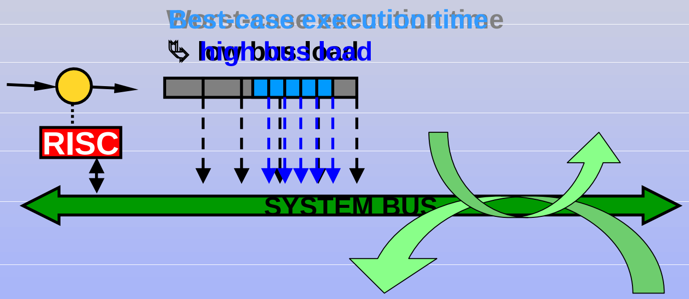
Timing Validation Complexity

- Process execution time intervals
- Complex run-time interdependencies



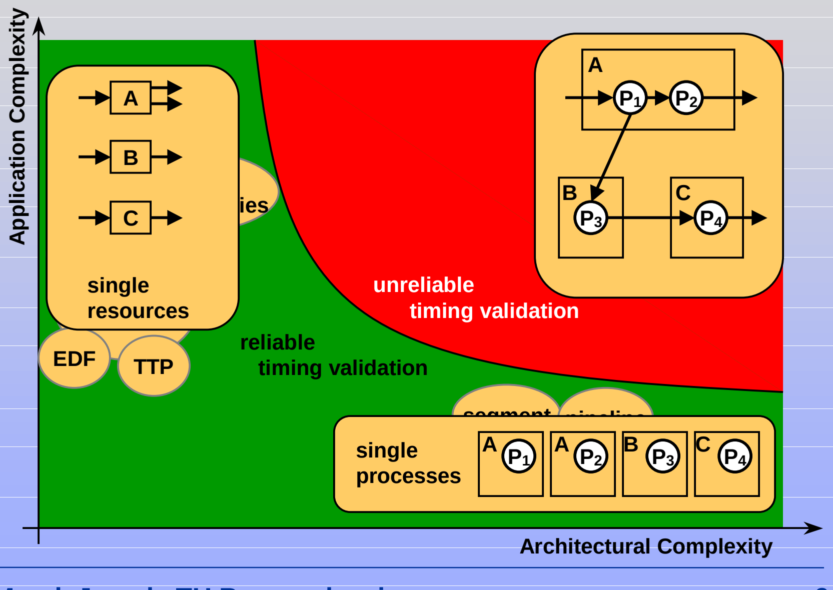
Limits of Simulation-based Validation

- *System* performance corner cases different from *component* performance corner cases



- Simulation limited to problems with known corner cases or when full coverage is feasible

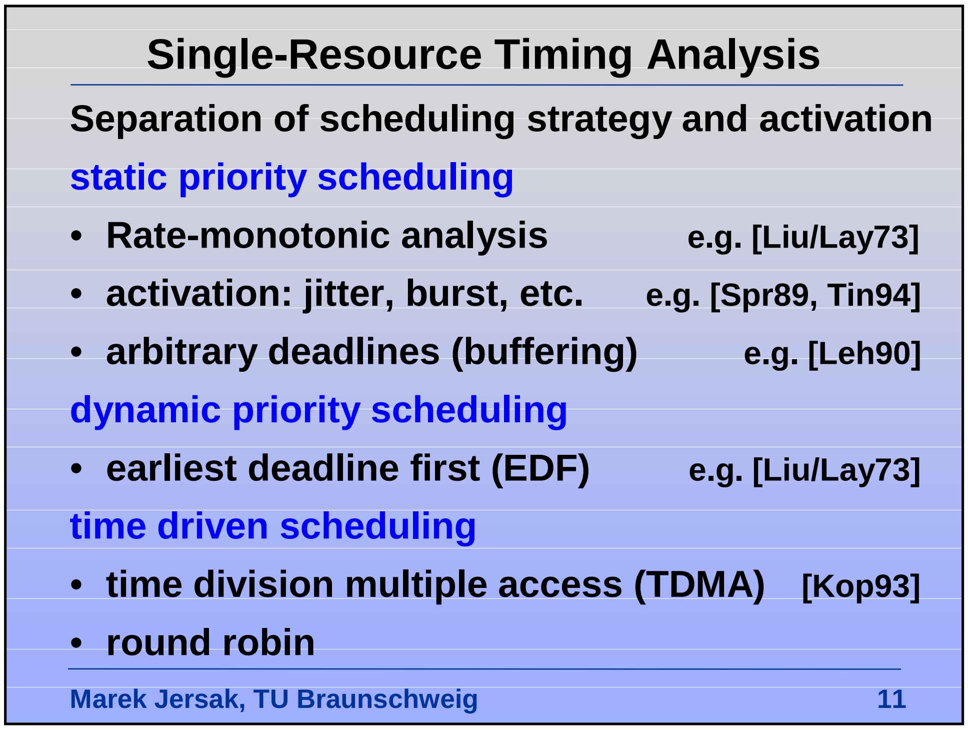
Reliable vs. Unreliable Timing Validation



Single-Process Timing Analysis

Separation of path analysis and architecture modeling

- Mok, Puschner, Park (Iteration bounds for loops)
- Gong and Gajski (Branching probabilities)
- Li and Malik (Implicit path enumeration)
- Ye, Wolf, Ernst (Segment-based analysis)
- First commercial approaches (AbsInt)



Single-Resource Timing Analysis

Separation of scheduling strategy and activation

static priority scheduling

- Rate-monotonic analysis e.g. [Liu/Lay73]
- activation: jitter, burst, etc. e.g. [Spr89, Tin94]
- arbitrary deadlines (buffering) e.g. [Leh90]

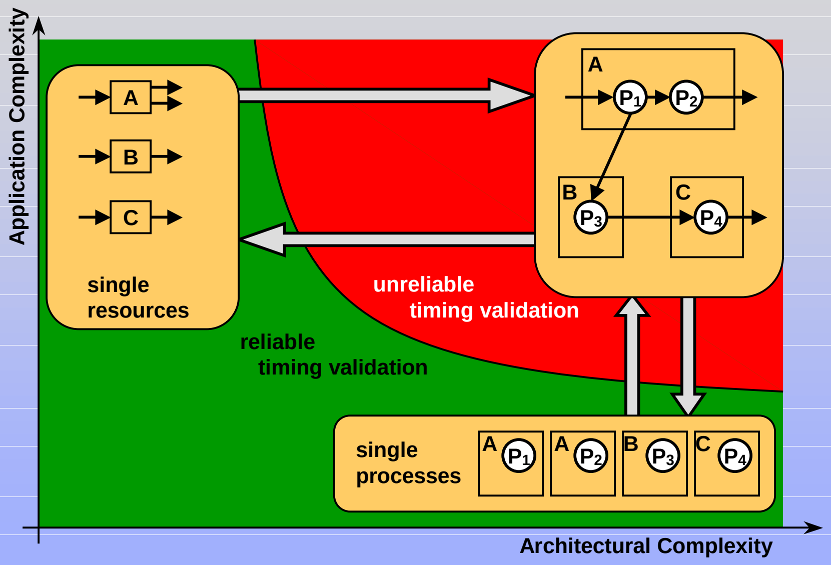
dynamic priority scheduling

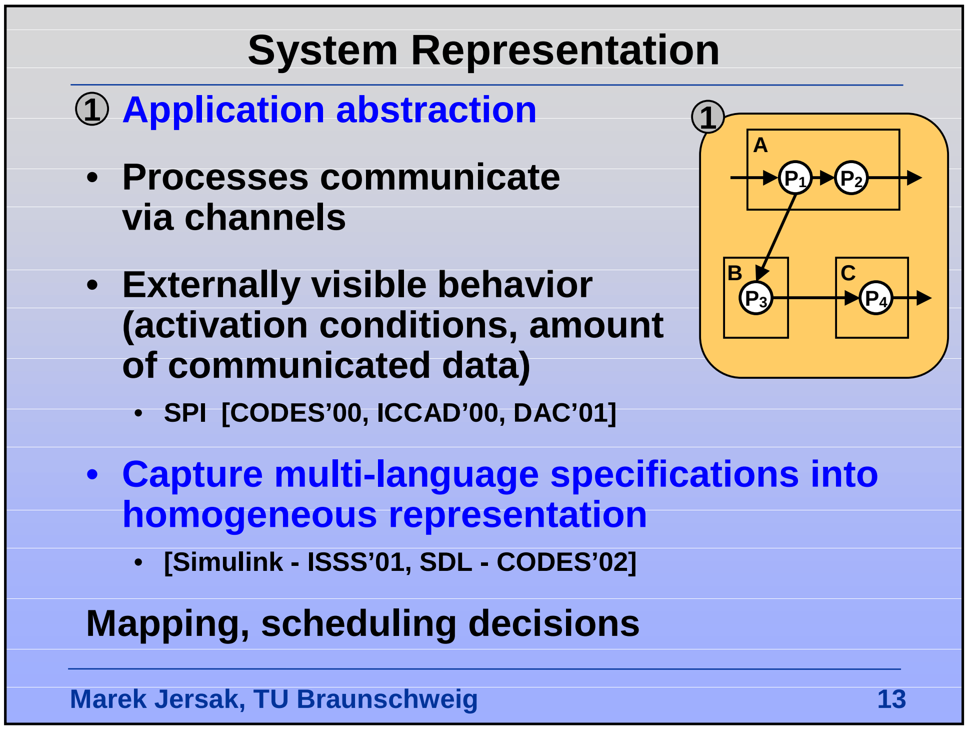
- earliest deadline first (EDF) e.g. [Liu/Lay73]

time driven scheduling

- time division multiple access (TDMA) [Kop93]
- round robin

Idea: Combine Reliable Results



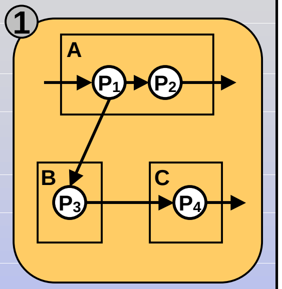


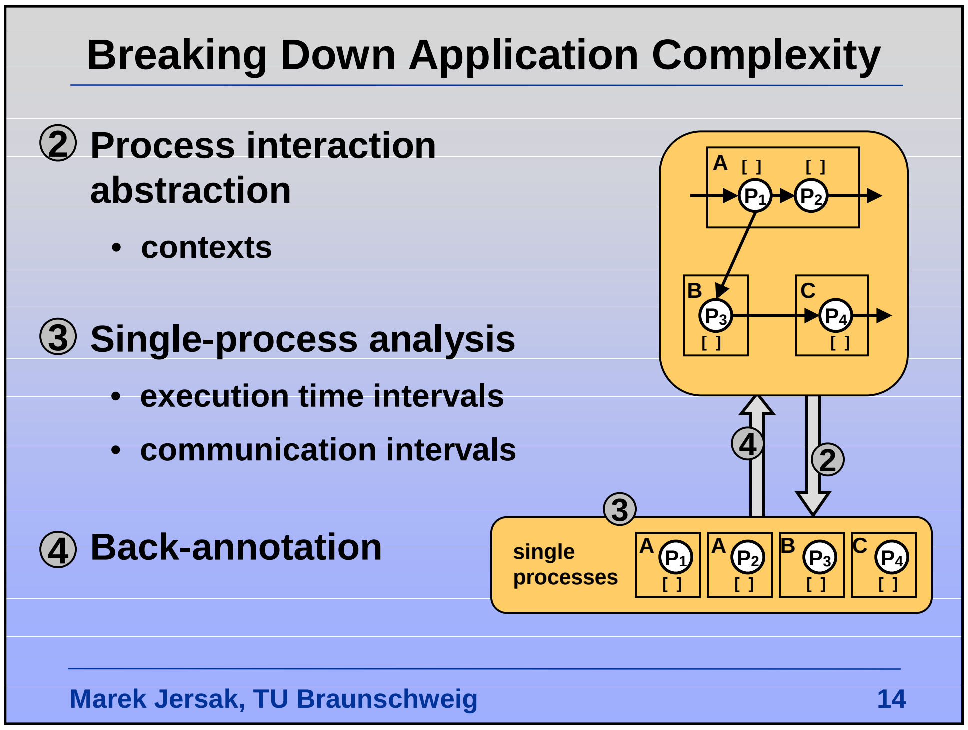
System Representation

① **Application abstraction**

- Processes communicate via channels
- Externally visible behavior (activation conditions, amount of communicated data)
  - SPI [CODES'00, ICCAD'00, DAC'01]
- **Capture multi-language specifications into homogeneous representation**
  - [Simulink - ISSS'01, SDL - CODES'02]

Mapping, scheduling decisions





Breaking Down Application Complexity

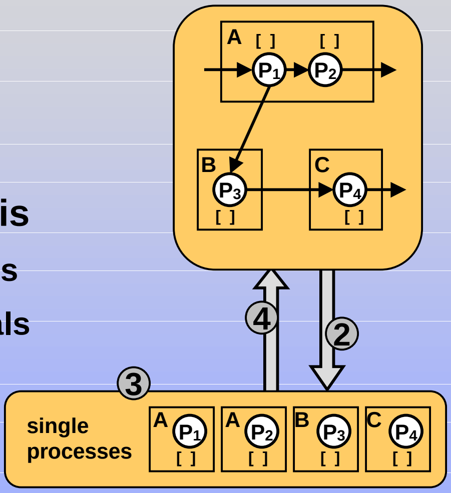
② Process interaction abstraction

- contexts

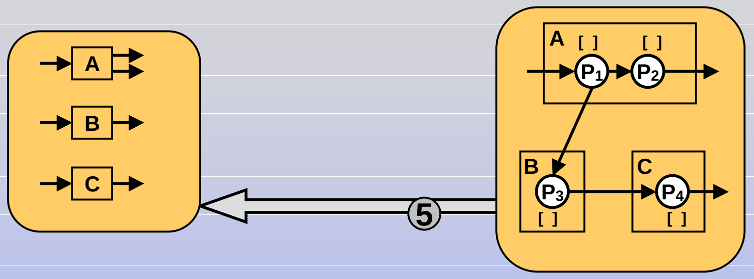
③ Single-process analysis

- execution time intervals
- communication intervals

④ Back-annotation



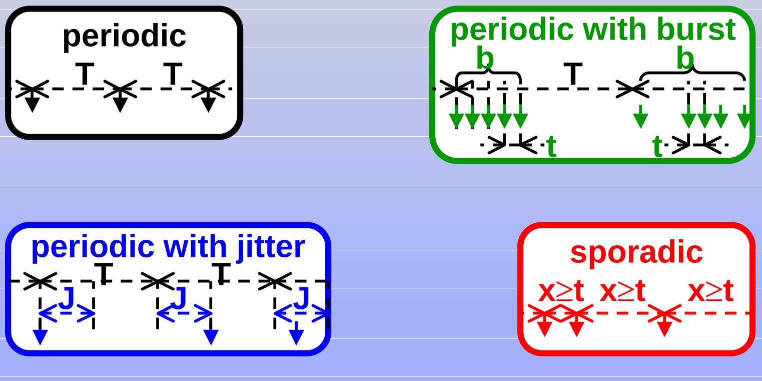
Breaking Down Architecture Complexity



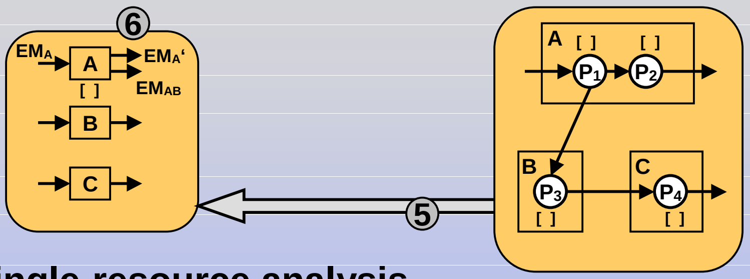
⑤ Resource interaction abstraction

Event Models

Available timing-analysis techniques require activation abstraction into event models



Single-Resource Analysis



⑥ Single-resource analysis

Requires

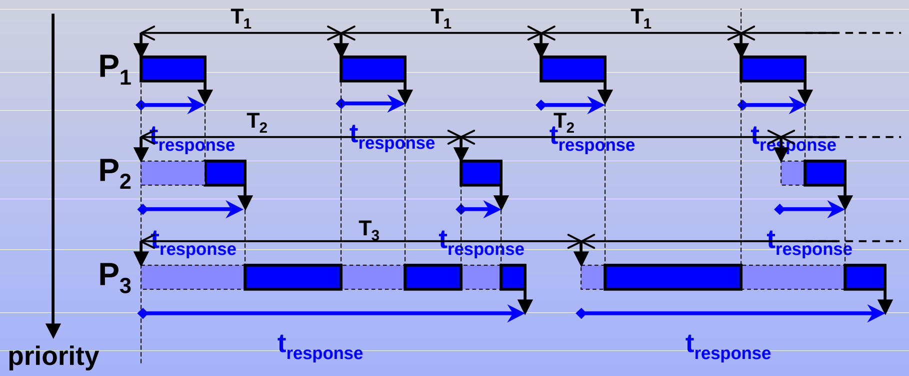
- input event models
- core execution time intervals

Generates

- Worst/best-case response times
- Output event models

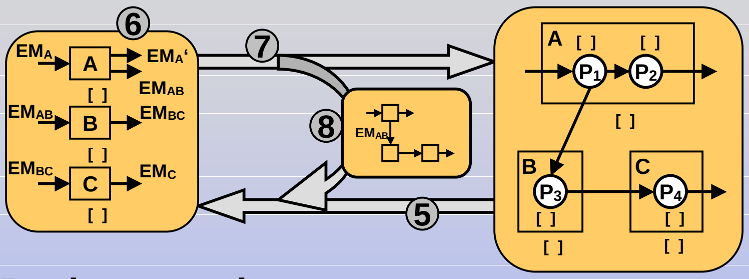
Example: Static Priority Scheduling

Given: Periodic input events with period T_x, Core Execution Times



Response: Periodic with jitter

Propagation of Event Models

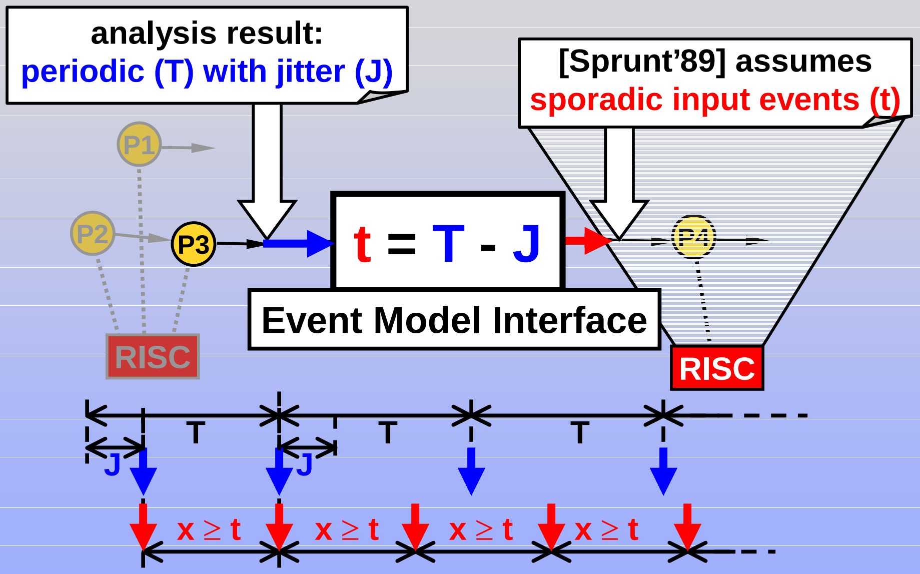


⑦ Back-annotation

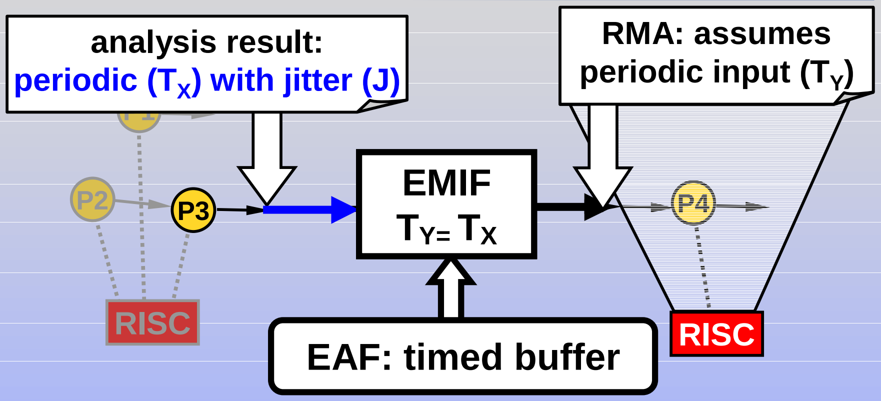
⑧ Output event models serve as input event models for analysis of the next resource

- Iterate steps ⑥, ⑦ and ⑧

Event Model Interface



Event Adaptation Function (EAF)

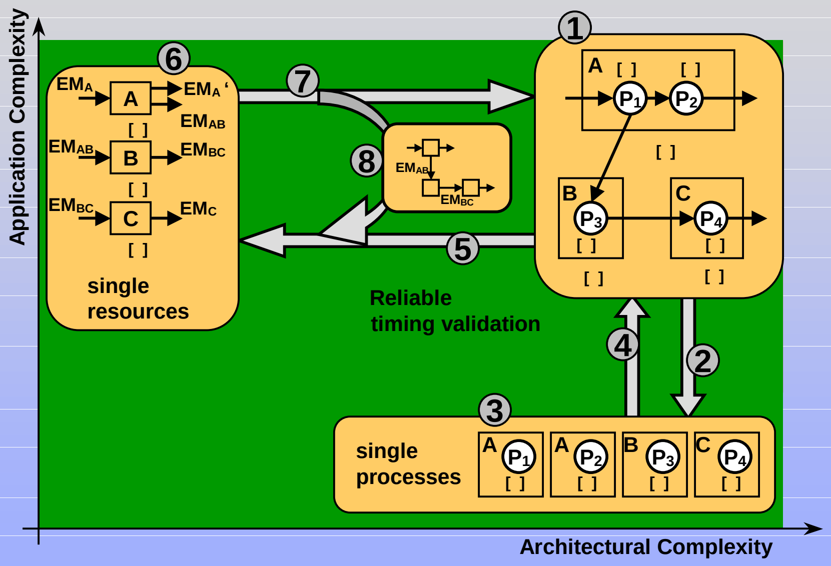

analysis result:
periodic (T_X) with jitter (J)
RMA: assumes periodic input (T_Y)
P1
P2
P3
EMIF
$T_{Y}= T_X$
P4
RISC
RISC
EAF: timed buffer

derive properties of EAF from event models:

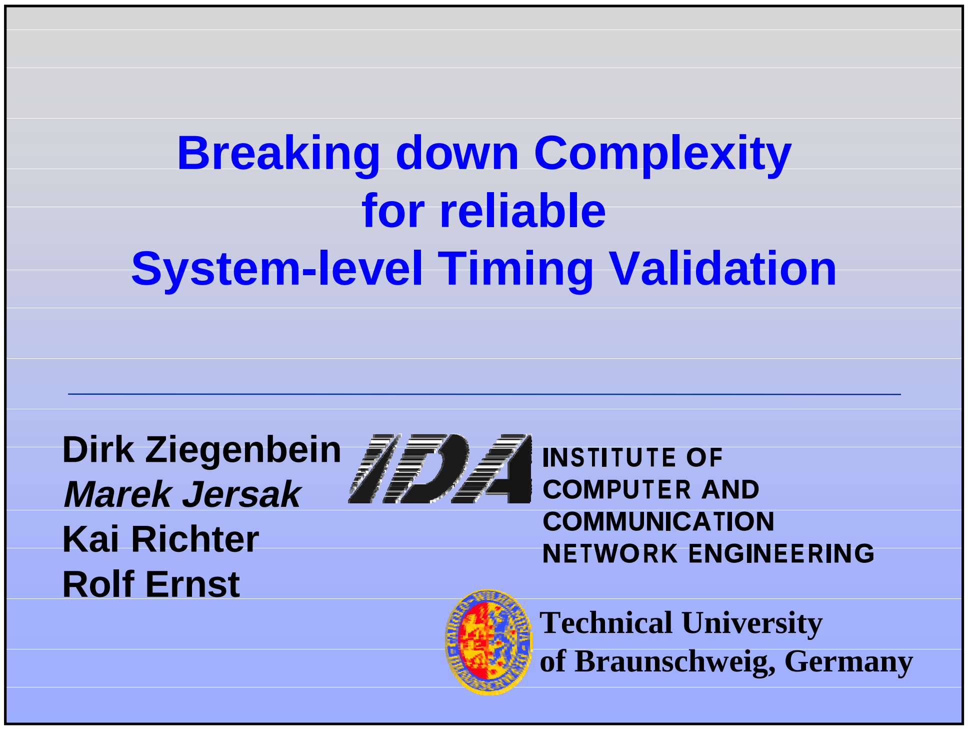
- required buffer size: 1
- maximum buffering delay: T_X

Marek Jersak, TU Braunschweig 21

Everything Together



Marek Jersak, TU Braunschweig 22

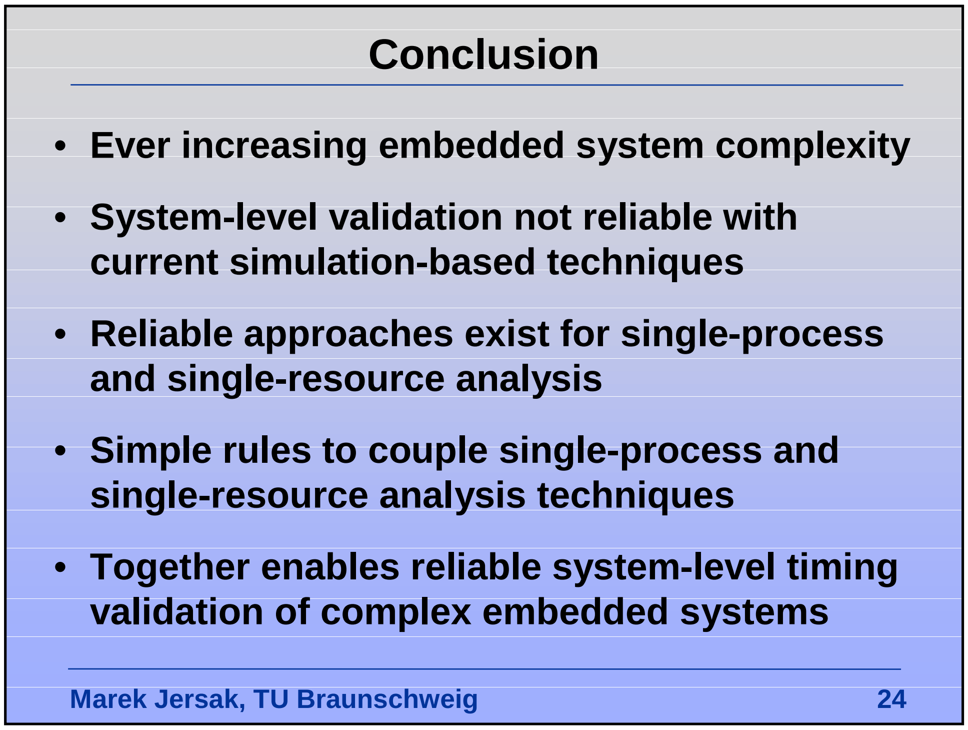


Breaking down Complexity for reliable System-level Timing Validation

Dirk Ziegenbein
Marek Jersak
Kai Richter
Rolf Ernst

INSTITUTE OF COMPUTER AND COMMUNICATION NETWORK ENGINEERING

Technical University of Braunschweig, Germany



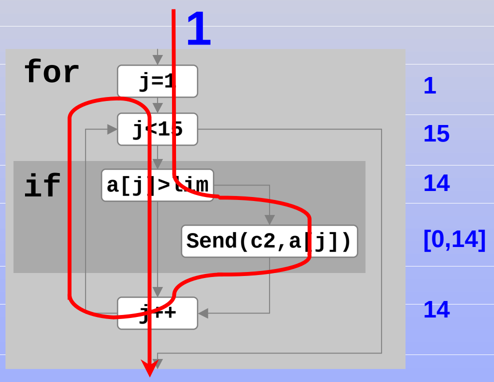
Conclusion

- Ever increasing embedded system complexity
- System-level validation not reliable with current simulation-based techniques
- Reliable approaches exist for single-process and single-resource analysis
- Simple rules to couple single-process and single-resource analysis techniques
- Together enables reliable system-level timing validation of complex embedded systems

Marek Jersak, TU Braunschweig 24

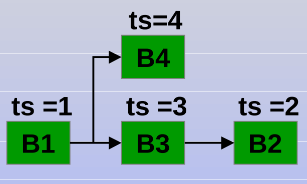
Single-Process Timing Analysis (SYMTA)

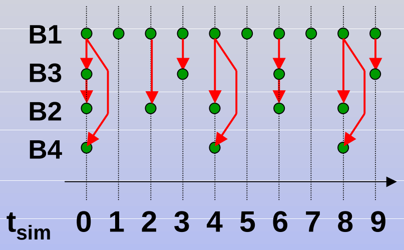
- Analysis of control structures (path classification)
  - Obtain execution number interval for each path
  - Execution of segments to obtain cost intervals (execution time, communication ...)
  - Conservative combination considering state of pipeline, cache ...



Application Capture: Example Simulink

- Coordination model: Time-driven, idealized timing





Coordination abstraction

- capture relative rates and data-dependencies into dataflow representation
- relax timing constraints

Host model

- Use RTW to generate C-code
- Use target-specific compiler