Performance Verification and Architecture Exploration using SymTA/S

Kai Richter

INSTITUTE OF COMPUTER AND COMMUNICATION NETWORK ENGINEERING

Technical University Braunschweig, Germany

Nutrition Facts

• System integration examples
• Performance verification challenges
• Compositional scheduling analysis
• SymTA/S demo
• Conclusion and outlook
• Discussion
Distributed Architecture Example

(Source: ATZ 2001)

- heterogeneous control units from multiple suppliers
- several coupled networks with different protocols

System-on-Chip Example: Viper Settop Box

(Source: Philips)

- MIPS bridge
- MIPS C-Bridge
- MPEG-2 video decoder
- Adv. image composition Processor
- Video input processor
- Memory-based scaler
- MPEG system proc.

IDA, TU Braunschweig
Function Integration Challenge

- subsystems are distributed over a vehicle
- subsystems are integrated and share resources

HW and SW System Integration

- many suppliers
- many standards
- many tools
- strongly networked

=> integration is key
Example: Deadline Monotonic Scheduling

- complex execution sequence – creates output jitter and bursts
- found in communication scheduling and multiprocessing

Distributed System Integration

- component best-case timing can turn into system worst-case timing (scheduling anomaly)
- requires expert knowledge about entire system
- who finds / provides such simulation patterns?
Distributed Application Example

performance verification:
• end-to-end deadlines
• bus and CPU overload  tough challenge !!!
• buffer memory overflow

Influences on System Performance
• distributed functions
• share resources = preempt & delay each other
• interact = talk to & depend on each other

=> variety of complex performance dependencies

• what are the critical corner cases ?

• how can they be covered during optimization and verification ?
Single-CPU Scheduling Analysis (RMA)

**Given:** Periods(T) & Core Execution Times (C)

Liu/Layland 1973:
- Worst-Case Response Times

\[ t_{\text{resp},i}^n = C_i + \sum_{j=1}^{i-1} C_j \left[ \frac{t_{\text{resp},j}}{T_j} \right] \leq D_i \]

Multiple Scheduling Strategies

- FCFS scheduling
- Static priority scheduling
- TDMA scheduling
- Proprietary (abstract info)
- Earliest deadline first scheduling

IDA, TU Braunschweig
Corresponding Analysis Techniques

- Buttazzo 1993
- Liu/Layland 1973
- Kopetz 1993
- Lee/Messerschmidt 1989
- Sha 1994

Integration ???

- Buttazzo 1993
- Liu/Layland 1973
- Kopetz 1993
- Lee/Messerschmidt 1989
- Sha 1994

from IP vendor
Current Practice in Performance Verification

- subsystem scheduling analysis
  e.g. rate-monotonic & others, first tools available
- complex system integration?
  ➔ unknown corner case coverage
  ➔ simulation bound to miss critical corner cases
  ➔ system integration is not reliable today

What else can we do?

"holistic" formal methods
- extends "classical" scheduling analysis into complex system-level timing equations
- specialized, mostly homogeneous - limited flexibility

"compositional" techniques
- composition of local techniques (re-use of e.g. rate-monotonic scheduling & existing tools)
  ➔ configurable, flexible
  ➔ supports heterogeneous systems
**"Compositional" Scheduling Analysis**

- independently scheduled components (CPUs, busses)
- coupled by data ("event") flow (communication)
- interpreted as activating events (event models)
- coupling corresponds to event propagation

**Known Event Models**

- observation: real-time analysis assumes similar event models at input
  - periodic event stream
  - periodic event streams *with jitter*
  - sporadic events with minimum event separation

- supported by a host of scheduling strategies
Challenges

• components may use different event models

• output timing usually more irregular than inputs timing
  • jitter increases and can lead to heavy bursts
  => complex dynamic dependencies
  • specifically painfull for dependency cycles

Event Model Interface (EMIF)

RMA: assumes periodic (T) with jitter (J)

[Sprung’89] assumes sporadic input events (t)

CPU1

CPU2

\[ t = T - J \]
Traffic Shaping = Event Adaptation

analysis result: periodic (T) with burst (t,n)

requires periodic events (T)

\[ T_Y = \frac{T_X}{b_X} \]

\[ \text{max buffering delay} \quad \text{buf}_{\text{max}} = T_X - (b_X-1)t_X \]

- calculate size from maximum delay:

Cyclic Event Stream Dependency

activation by RTOS

simple periodic

sens

preemption

CPU

preemption

HW

interference

sens

sporadic w/jitter

periodic w/jitter

periodic

interference

NoC
Cyclic Event Stream Dependency

Event Models

- strictly periodic: $T$
- periodic with jitter: $T, J < T$
- periodic with bursts: $T, J \geq T$
- sporadic: $t$
- sporadic with jitter: $T, J < T$
- sporadic with bursts: $T, J \geq T$
Benefits I

1. event models & stream propagation displays the global consequences of
   - scheduling
   - integration
   - buffering

2. known scheduling & analysis techniques can be applied locally to each component

=> system-level performance becomes comprehensible and controlable
Benefits II

3. SymTA/S runs fast, and a large design space can be explored

=> system architecture can be systematically dimensioned and optimized

Performance Verification and Architecture Exploration

www.symta.org

Kai Richter