SymTA/S
Compositional performance analysis

ARTIST Workshop on tool platforms for modeling, analysis, and validation of embedded systems

Arne Hamann
Rolf Ernst
Outline

- Performance verification flow
  - Process execution model
  - Component and communication execution model
  - Global system execution model
- Compositional system level analysis
  - Iterative system level analysis approach
  - Considering task dependencies
- The SymTA/S tool
- Conclusion
Performance verification flow
Target architecture performance – general view

Global system execution model

Component and communication execution model

Process execution model
Process execution model

- Influenced by
  - execution path
    - data dependent
  - execution path timing
    - target architecture dependent
  - process communication
    (here: message passing)
    - execution path dependent
  - communication volume
    - data and type dependent

execution time analysis
Process timing and communication

- State of industrial practice - simulation/performance monitoring
  - trigger points at process beginning and end
  - data dependent execution → upper and lower timing bounds

- Simulation challenges
  - coverage?
  - cache and context switch overhead due to run-time scheduling with process preemptions

- Alternative - formal analysis of individual process timing
  - provides conservative bounds
  - serious progress in recent years
Formal process execution time analysis

- Active research area with dedicated events (e.g. Euromicro WS)
- Formal analysis using simple processor models
  - Li/Malik (Princeton) (95): Cinderella
- Detailed execution models with abstract interpretation
  - Wilhelm/Ferdinand (97 ff.): commercial tool AbsInt
- Combinations with simulation/measurement of program segments
  - Staschulat/Ernst (99 ff.): SymTA/P
- All tools provide (conservative) upper execution time bounds (WCET) or time intervals (WCET/BCET)
Component and communication execution model

- Influenced by
  - resource sharing strategy
  - process activation

single component real-time analysis
Component and communication execution model

- Resource sharing strategy

- Process and communication scheduling
  - static execution order
  - time driven scheduling
    - fixed: TDMA
    - dynamic: Round-Robin
  - priority driven scheduling
    - static priority assignment: RMS, SPP
    - dynamic priority assignment: EDF

- Timing depends on environment model
  - determines frequency of process activations or communication
Multiple Scheduling Strategies

- FCFS scheduling
- TDMA scheduling
- static priority scheduling
- CoPro
- RISC
- MEM
- DSP
- IP
- VLIW
- proprietary (abstract info)
- earliest deadline first scheduling
- static execution order scheduling
Scheduling Analysis Techniques

Buttazzo 1993

Liu/Layland 1973

Kopetz 1993

CoPro

RISC

MEM

DSP

IP

MEM

IP

VLIW

SYSTEM BUS

Lee/Messerschmidt 1989

from IP vendor

Sha 1994

from IP vendor

Lee/Messerschmidt 1989
Example: Rate Monotonic Scheduling (RMS)

- Very simple system model
  - periodic tasks with deadlines equal to periods
  - fixed priorities according to task periods
  - no communication between tasks
  - (theoretically) optimal solution for single processors
  - several practical limitations but good starting point

- Schedulability tests for RMS guarantee correct timing behavior
  - processor utilization (load) approach
  - response time approach (basis for many extensions)
RMS Theory – The response time approach

- Critical instant:
  all tasks start at $t=0$ ("synchronous assumption" to ensure maximum interference in the beginning of task execution)
- when each task meets its first deadline, it will meet all other future deadlines (proof exists!)
- test by "unrolling the schedule" (symbolic simulation)
RMS Theory – The response time formula

\[ R_i = C_i + \sum_{j \in \text{hp}(i)} C_j + \left\lfloor \frac{R_i}{T_j} \right\rfloor \leq D_i = T_i \]

- Response time
- Core execution time
- Fix-point problem
- # of preemptions
- Interference term \( I_i \)
Example: Static priority w/ arbitrary deadlines

- Assumption:
  - tasks with periods $T$, worst-case execution times $C$
  - static priorities
  - deadlines (arbitrary) larger than the period
Analysis uses “Busy Window” approach (Lehoczky)

\[ w_i(q) = q \cdot C_i + \sum_{j \in \text{bp}(i)} C_j \left[ \frac{w_i(q)}{T_j} \right] \]

\[ R_i(q) = w_i(q) - (q - 1) \cdot T_2 \]

find fix point where equations hold!
Other Extensions in Literature

- Jitter and burst activation
- Static and dynamic offsets between task activations
- Different task modes
- Execution scenarios
- Blocking and non-preemptiveness
- Scheduling overhead → context switch time
- etc...
Global system execution model

- influenced by
  - communication pattern
  - shared memory access
  - environment model
Compositional performance analysis
Tasks are coupled by event sequences

Composition by means of event stream propagation
- apply local scheduling techniques at resource level
- determine the behavior of the output stream
- propagate to the next component
Idea

- Use stream model describing the distribution of activating events as intermediate mathematical formalism.
- E.g. arrival curve functions of network calculus:
  - $\eta^+(\Delta t)$ maximum number of activating events occurring in time window $\Delta t$.
  - $\eta^-(\Delta t)$ minimum number of activating events occurring in time window $\Delta t$.
  - $d_-$ minimum event distance - limits burst density.
Input – output event model relation

- Any scheduling increases jitter
- Jitter grows along functional path
- Increasing jitter leads to
  - burst and transient overloads
  - higher memory requirements
  - power peaks
System analysis loop

environment model

map to input event model

local analysis

schedulability?

YES

YES

derive output event model

convergence?

NO

NO

infeasible configuration

feasible configuration
Considering task dependencies
Taking global dependencies into account

- Utilized stream model is state-less

- Classical critical instance assumption is save but often overly conservative
  - Reason: activating events in different event streams are often time-correlated which rules out the simultaneous activation of all tasks

- Solution: consider „inter-context“ dependencies between tasks to tighten analysis results
  - Idea: propagate offset information along event streams
Motivating Example

- Static priority preemptive scheduling on all resources
- Compositional performance analysis approach
Lehoczky (1990)

- Ignore correlation between tasks!
Lehoczky (1990)

- Ignore correlation between tasks!
Lehoczky (1990)

T3
T5
T8

Priority

R4

CET = [2,2] Priority=Mid

CET = [2,2] Priority=High

CET = [2,2] Priority=Low

P3 = 50 J3 = 8

P5 = 50 J5 = 8

P8 = 50 J8 = 6

R_W^8 = 6

critical instant

P5 = 50
J5 = 8

T5

T3

T8

t
t

2

2

2
• Periodic arrival of events at system inputs as timing-reference
Tindell (1994)

Global Offset $\Phi_i = \text{earliest activation time of } T_i \text{ relative to the periodical arrival of an external event at the system input}$
Tindell (1994)

\[ \Phi_5 = 2 \]

\[ \Phi_8 = 4 \]

\[ P_5 = 50 \]
\[ J_5 = 8 \]
\[ \Phi_5 = 14 \]

\[ P_3 = 50 \]
\[ J_3 = 8 \]
\[ \Phi_3 = 2 \]

\[ P_8 = 50 \]
\[ J_8 = 6 \]
\[ \Phi_8 = 4 \]

CET = [2,2]
Priority=Mid

CET = [2,2]
Priority=Low

CET = [2,2]
Priority=High

Priority

T3
T5
T8

External event arrival

Critical instant

t

t
Tindell (1994)

external event arrival

Priority

T₃

T₅

T₈

critical instant

CET = [2,2]

Priority = Mid

CET = [2,2]

Priority = High

CET = [2,2]

Priority = Low

P₅ = 50

J₅ = 8

Φ₅ = 14

P₃ = 50

J₃ = 8

Φ₃ = 2

P₈ = 50

J₈ = 6

Φ₈ = 4

R₄ = 4
Further Techniques

- Relative offsets and relative jitter (Henia et al.)
  - Extends idea of global offsets
  - Describes the earliest activation time of a task relative to a timing-reference \textit{ref}
  - Reference is not necessarily a periodic external event
  - Enables tighter response time calculation
- Precedence relations
  - Explicitly considers precedence relations between tasks (i.e. task i cannot start until task j has finished execution)
  - Orthogonal to offset based techniques
Conclusion

- Abstract stream models enable early system performance analysis ...
- ... requiring only key performance data
- Advantage: very fast analysis ...
  - 10s of tasks: order of milliseconds
  - 100s of tasks: order of seconds
- ... allows the application of advanced analysis features
  - System sensitivity analysis
  - System exploration including robustness optimization
- Presented formalisms implemented in a tool called SymTA/S
- Tool commercialized by Symtavision
SymTA/S Tool Suite

- Exploration
- Sensitivity Analysis
- Analysis Engine
- verified system

flow integration:
- data bases
- tools

3rd party

open

analysis libraries for ECUs, buses, etc ...:
- SPP/DMA/RMA
- EDF
- TDMA
- RR

Industry related:
- OSEK flavours
- CAN

idea, specification, sketch, existing system
SymTA/S screenshot
Thank you for your attention !!!