Comparison of Memory Access Strategies in Multi-core Platforms using MAST

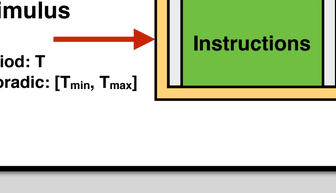
Juan M. Rivas[1], J. Javier Gutiérrez[2], Julio L. Medina[2] and Michael González Harbour[2]
[1]PARTS Research Center, Université Libre de Bruxelles (Belgium) - [2]Software Engineering and Real-Time, University of Cantabria (Spain)

[1]jrivasco@ulb.ac.be, [2]{gutierjj, medinajl, mgh}@unican.es

The Challenge

Challenge: To calculate latencies in an engine management system

The Platform



Memory access times (cycles)

	LRAM0	LRAM1	LRAM2	LRAM3	GRAM
CORE0	1	9	9	9	9
CORE1	9	1	9	9	9
CORE2	9	9	1	9	9
CORE3	9	9	9	1	9

200 MHz system-wide

FIFO arbitration at memories

Real-time situation: Amalthea Model

http://www.amalthea-project.org/

Amalthea Tasks

- 21 Tasks
- Statically assigned to a core
- Fixed Priority: preemptive/ cooperative
- Released by stimuli: periodic/ sporadic (arbitrary phasing)
- D=T
- Series of Runnables

Amalthea Runnables

- 1250 Runnables
- Read labels (memory)
- Instructions: constant/deviation
- Write labels (memory)

Labels

- 10000 Labels
- Mapped to GRAM/LRAM
- Local RAM = 1 cycle
- Non-Local RAM = 9 cycles

Approach: RTA with MAST

MAST Tool

Tool to model, analyze and optimize real-time systems

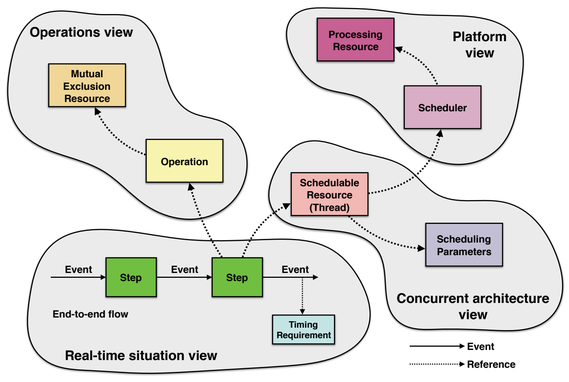
Open source, available at mast.unican.es (Windows and Linux)

Schedulabilty analysis tools	Optimization tools	Other tools	Support
• Holistic • Offset-based • Offset-based slanted • Offset-based w/ precedence • Offset-based brute force	• Simulated annealing • UD • ED • PN • NPD • EQS • EQF • HOSPA	• Simulator • Sensitivity analysis • Graphical editor • Results viewer	• Shared resources • Multipath e2 flows • Sporadic and Polling Servers • FP+EDF scheduling • Networks (AFDX, CAN) • Partitioned systems • Generator (GEN4MAST)

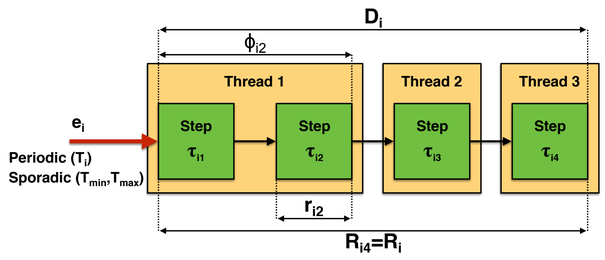
MAST Model

Aligned with OMG MARTE (SAM profile)

MAST model overview



MAST model for analysis



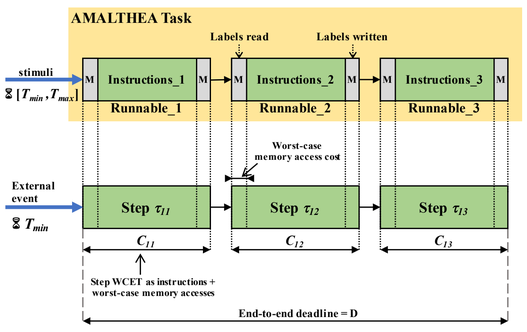
Results from response-time analysis:

- Worst-case Local response time: r_{ij}
- Worst-case Global response time: R_{ij}
- Best-case response times: r^b_{ij}, R^b_{ij}

Approach: RTA with MAST

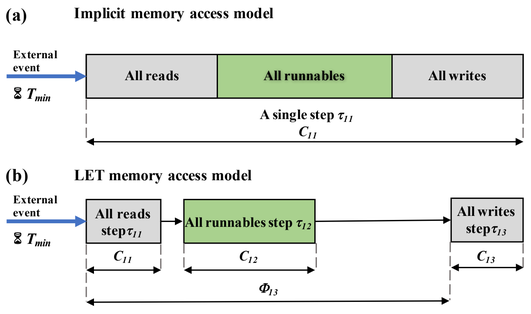
Amalthea to MAST transformation

Explicit memory access model



- Each Runnable can access memory unrestricted

Implicit and LET memory access model

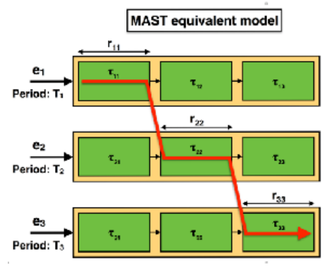


- All memory access are moved to the beginning and end of the tasks
- Local memory accesses costs added to instructions WCET

Event-chains

Latency model of data flows among non consecutive runnables

Event-chains crossing different tasks



Explicit

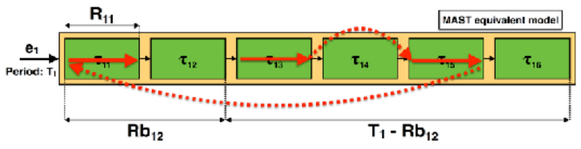
$$L = r_{11} + T_2 + r_{22} + T_3 + r_{33}$$
$$L^b = r^b_{11} + r^b_{22} + r^b_{33}$$

Implicit and LET

$$L = R_1 + T_2 + R_2 + T_3 + R_3$$
$$L^b = R^b_1 + R^b_2 + R^b_3$$

Local response times turn into global ones

Event-chains in the same task



Explicit

$$L = (T_1 - R^b_{12}) + R_{11}$$
$$L^b = (T_1 - R_{12}) + R^b_{11}$$

Implicit

$$L = (T - R^b) + R$$
$$L^b = (T - R) + R^b$$

LET

$$L = T + R$$
$$L^b = T + R^b$$

Results

Considerations

Clock speed increased to 300 MHz

Original 200 MHz yields utilizations above 100%

Priorities with Implicit and LET

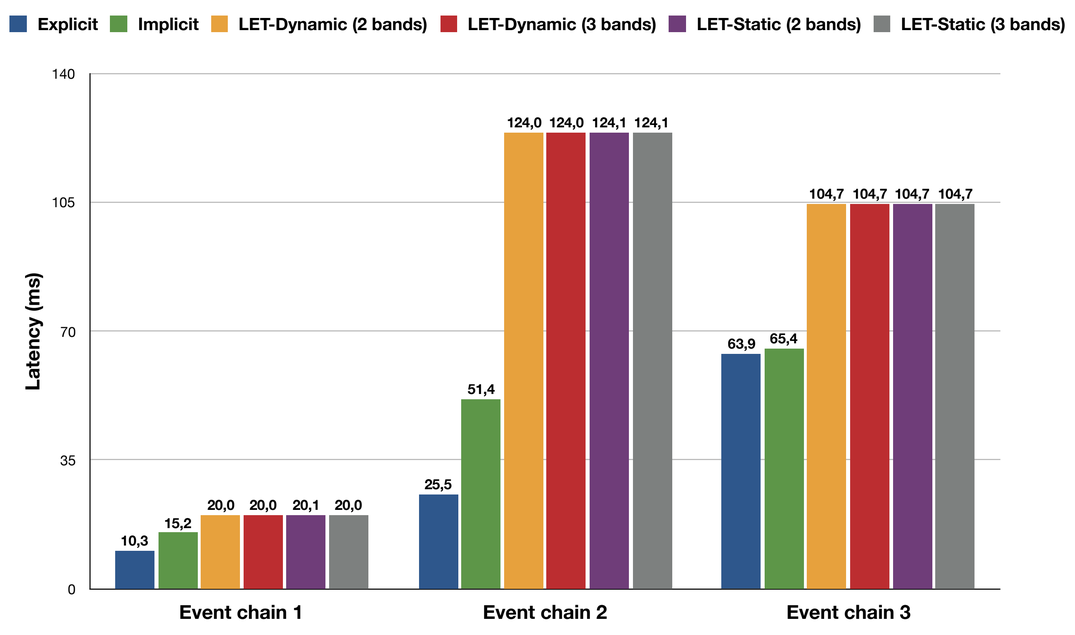
Instructions always execute in a lower priority band

- 2 priority bands: Same priority for Write and Read
- 3 priority bands: Read at middle priority, Write at high priority

Two types of offsets (ϕ) assignment for LET

- LET-Static: Offsets are set equal to periods
- LET-Dynamic: Offsets so that WCRTs of write operations equal to the periods

Event-chain latencies



Conclusions

None of the strategies is a clear winner

No solution can reduce jitter and latencies at the same time

Implicit model has similar latencies than Explicit for the tasks and higher latencies for the event chains without getting a reduction of jitter

Penalty to keep data consistency

LET has a good control of jitter at the cost of a significant increase of latencies for both tasks and event chains.