Schedulability and Timing Analysis of Mixed Preemptive-Cooperative Tasks on a Partitioned Multi-Core System

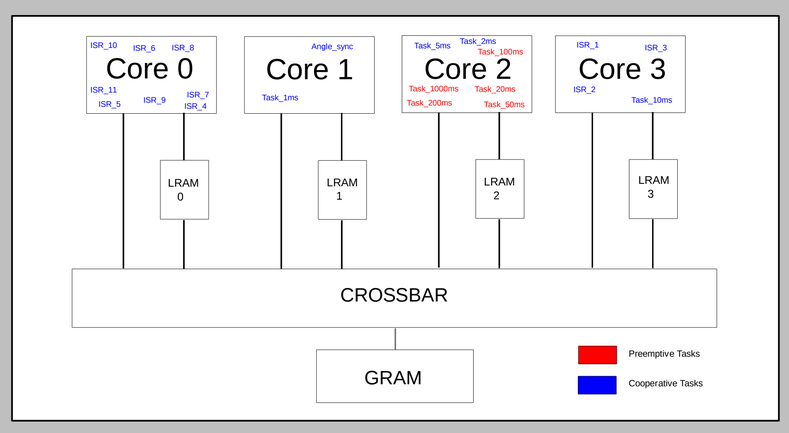
Ignacio Sañudo, Paolo Burgio and Marko Bertogna
University of Modena, Italy

System model

The FMTV verification challenge proposes **a timing and schedulability analysis of an engine management system** to be executed on a shared-memory multi-core platform.

- The application consists of statically partitioned **tasks**, each one composed of multiple **runnables** that are executed according to a **read-compute-write policy.**

The system model is composed of a microcontroller architecture with four identical cores. Each CORE is connected to a single ported local memory arbitrated according to a FIFO policy, there also exist a global memory that is shared among all cores.



Task model

- The task model consists of **statically partitioned tasks**,
- Each one composed of **multiple runnables**
- Mixed **preemptive-cooperative** task/runnable model:

  a. **Preemptive runnables** may only be preempted by higher priority runnables
  b. **Cooperative runnables** may be preempted by higer priority preemptive runnables and by higher priority cooperative runnables at boundaries

Goal

The challenges proposed by Bosch are:

I. Calculate tight end-to-end latencies **ignoring** memory accesses and arbitration
II. Calculate tight end-to-end latencies **including** memory access and arbitration accesses
III. Optimize end-to-end latencies by **mapping** the labels among the local and global memories

Challenge I

We propose a **mixed preemptive-cooperative model**

Computing the largest level-i active period $L_i = \sum_{j:P_j \geq P_i} \left\lceil \frac{L_i}{T_j} \right\rceil C_j$

Considering all jobs $K_i = \left\lceil \frac{L_i}{T_j} \right\rceil$

Blocking time $B_i = max_{j,r:P_j<P_i}\{ C_{j,r}\}$

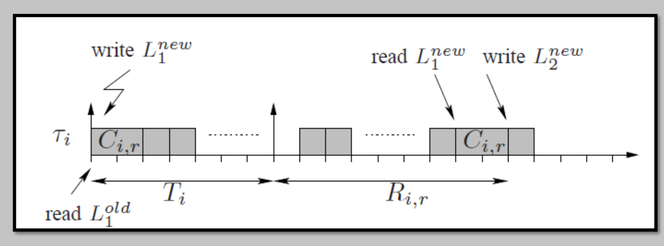
Start time $s_{i,r}^k = B_i + \sum_{j:P_j>P_i} \left(\left\lfloor \frac{s_{i,r}^k}{T_j} + 1 \right\rfloor \right) C_j + (k-1)C_i + \bar{C}_{i,r-1}$

Finishing time $f_{i,r}^k = s_{i,r}^k + C_{i,r} + \sum_{j:P_j>\theta_i} \left(\left\lceil \frac{f_{i,r}^k}{T_j} \right\rceil - \left(\left\lfloor \frac{s_{i,r}^k}{T_j} + 1 \right\rfloor \right) \right) C_j$

Response time $R_{i,r}^k = f_{i,r}^k - (k-1)T_i$

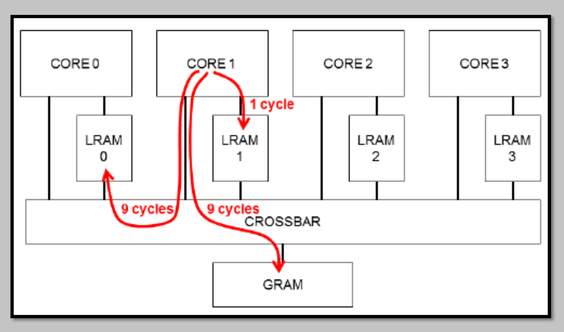
Worst case response time $R_{i,r} = max_{k\epsilon[1,k_i]}\{ R_{i,r}^k \}$

Maximum propagation delay from an initial event to the final runnable → **Effect chain**



$$\delta(EC) = \sum_{T_i \in EC} (T_i + R_{i,r})$$

Challenge II

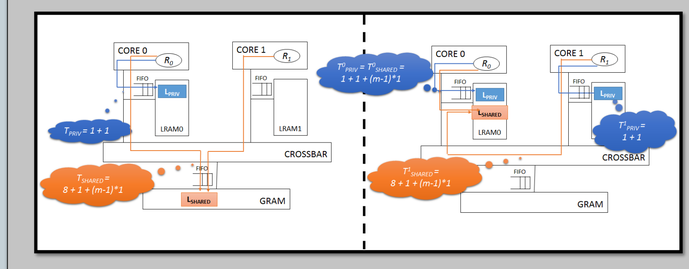


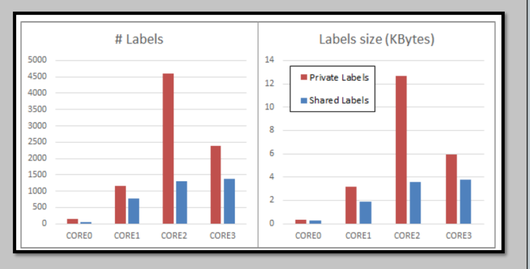
Analysis **including** memory access

- The delay for a global memory access is of 8 cycles for crossbar traversing and 1 cycle for the memory access. The memory access time has to be multiplied by the number of cores m that may concurrently access the global memory.

$$C_{i,r} = (n^I/_f) + (8+(1*\mathrm{m})*n^R) + (8+(1*\mathrm{m})*n^W)$$

Challenge III





Taking into account the label distribution among cores, we can design an efficient mapping:

- $T_{LRAM} = (m-1) * 1(FIFOqueue) + 1(memory) = m$
- $T_{GRAM} = 8(xbar) + 1(memory) + (m-1) * 1(FIFOqueue) = 8 + m$

For the **PRIVATE** labels,

- optimal choice → map labels to the local memory of the core that exclusively accesses them → (m cycles vs. $8 + m$ cycles).

For the **SHARED** labels,

- map each label to the "core-closer" LRAM that mostly accesses it → this could worsen the latencies of other runnables → there is not an optimal solution. Reasonable solution → map shared labels in closer LRAMs favoring the labels involved on an effect chain.

http://hipert.mat.unimore.it/FMTV16/

This Project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement: 688860.