To synchronise, or not to synchronise, that is the Question: Resource-aware task graph scheduling using ILP on multi-core

Benjamin Rouxel, Isabelle Puaut and Steven Derrien
ALF/Cairn team, University of Rennes 1 and INRIA, France

MOTIVATION

Multi-core architectures have introduced new interferences conflicts in the estimation of the Worst Case Execution Time (WCET). I/O ports and shared variables are not anymore the only shared resources in the architecture. Now buses and NoCs must also be considered as such shared resources requiring a contention analysis in order to compute a safe WCET [1].

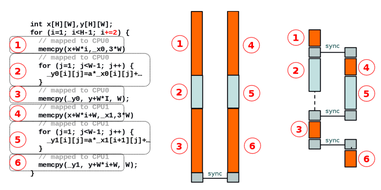

Figure 1: Problem illustration – From left to right:
a code sample, concurrent execution of task using a shared resource, synchronized task using a shared resource

- two parallelisable inner-loops (no data-dependencies)
- loop body surrounded by fetch/store data
- fetch/store tasks interfere – order is unpredictable
- WCET of each task includes worst-case concurrency

Synchronisation avoids interferences.

CONTRIBUTION

- static mapping/scheduling of tasks onto cores
- synchronisation added at the mapping/scheduling phase
- mutual exclusion between concurrent tasks
- absence of synchronisation = overlapping the execution of tasks on different cores

Comparison of the result from the mapping/scheduling step with and without synchronisation.

TASK MODEL

- streaming applications suite [2]
- represented with a Directed Acyclic Graph (DAG), a task graph (figure 2)
- communication task surround computation one [3], figure 3

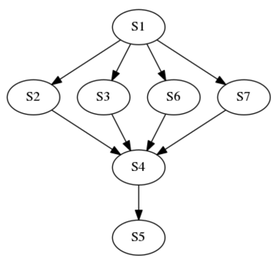

Figure 2: Simple task graph example

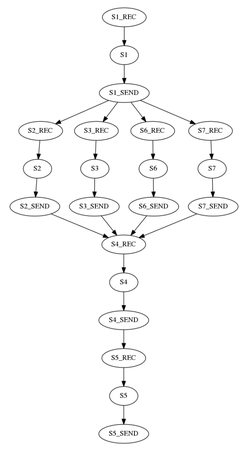

Figure 3: Augmented task graph with communication tasks

SOLVING METHOD

- exact method Integer Linear Programming (ILP) (solver: CPLEX – https://huit.re/CPLEX)
- time-triggered clustered non-pre-emptive scheduling model
- no synchronisation cost

General ideas of the constraint's system:

- variables:
  - mapping of a task on a processor
  - task's ordering
  - release time of each tasks
- constraints:
  - unicity: a task is mapped to only one processor
  - conflict: ordering task, 2 tasks can't be both before/after each other
  - causality: task's order must respect de causality chain
  - communication task specificities: communication tasks have to be on the same processor as their related computation one (w.r.t the order)
- Objective function: minimizing the schedule length (ending time of last task)

ARCHITECTURE AND COMMUNICATION MODEL

Chip architecture:

- processor with scratchpad memory (SPM) per core (Patmos [4])
- BUS with Time Division Multiplexing (TDM) and arbitration policy
- inter-core communications from SPM to SPM through the BUS

$worst-case\ interference\ latency\ =$
$waiting\ time\ +\ time\ to\ send\ data$ depends on:

- the amount of data to transmit
- the number of processors
- the arbitration policy

***waiting time* can be removed when synchronised.**

MAPPING/SCHEDULING

Mapping and scheduling of the task graph example on 4 processors with different arbitration policy and considering synchronising or not.

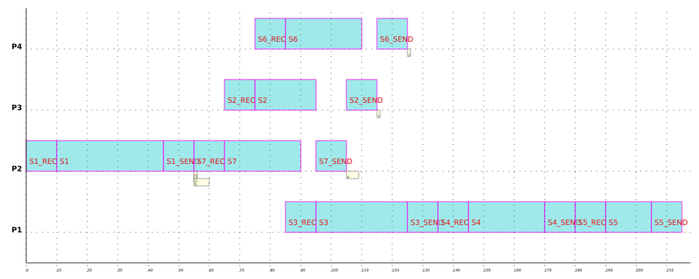

Figure 4: FAIR arbitration with mutual exclusion – Global WCET is 215

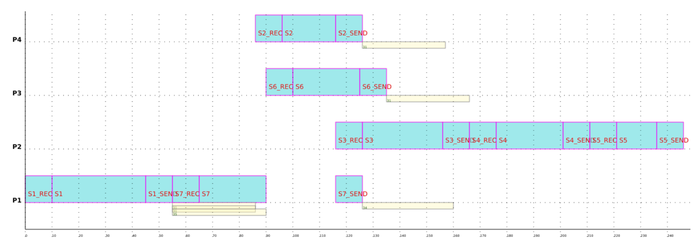

Figure 5: FAIR arbitration without synchronisation – Global WCET is 246

FIRST RESULTS

We scheduled the previous task graph on a range number of processors, from 2 to 10. As illustrated by figure 8 our intuition was right. Indeed the addition of synchronisation for FAIR arbitration policy does really improve the WCET thus limiting the contention overhead. However, adding synchronisation do worst on TDM arbitration policy.

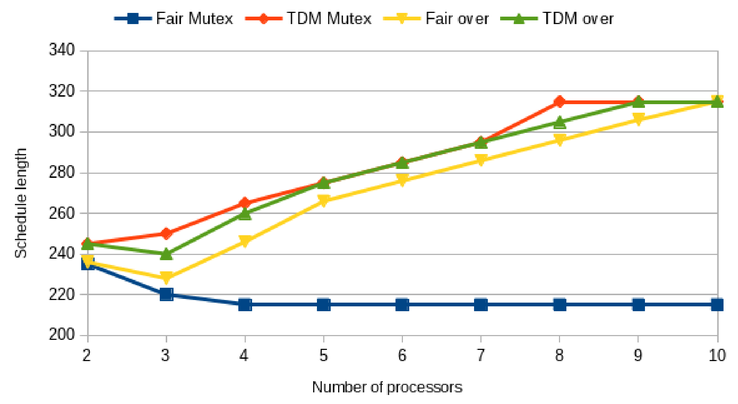

Figure 8: Comparison of the global WCET depending on the number of processors

WHAT'S NEXT?

- to map/schedule the StreamIT benchmark suite[2]
- to improve the ILP, partial overlapping/synchronising
- to handle NoC architecture
- to build heuristics, better scale on larger task graph

REFERENCES

[1] G. Fernandez, J. Abella Ferrer, E. Qui nones Moreno, C. Rochange, T. Vardanega, F. J. Cazorla Almeida, *et al.*, "Contention in multicore hardware shared resources: Understanding of the state of the art," 2014.

[2] W. Thies, M. Karczmarek, and S. Amarasinghe, "Streamit: A language for streaming applications," in *Compiler Construction*, pp. 179–196, Springer, 2002.

[3] P. Tendulkar, P. Poplavko, I. Galanommatis, and O. Maler, "Many-core scheduling of data parallel applications using smt solvers," in *Digital System Design (DSD), 2014 17th Euromicro Conference on*, pp. 615–622, IEEE, 2014.

[4] M. Schoeberl, F. Brandner, S. Hepp, W. Puffitsch, and D. Prokesch, "Patmos reference handbook," *Technical University of Denmark, Tech. Rep*, 2015.

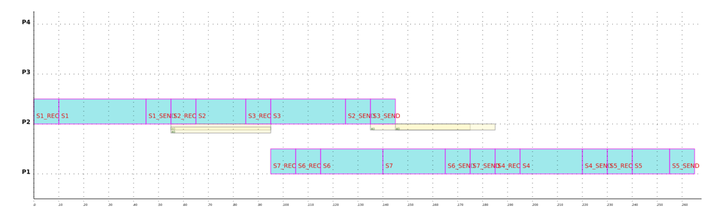

Figure 6: TDM arbitration with mutual exclusion – Global WCET is 265

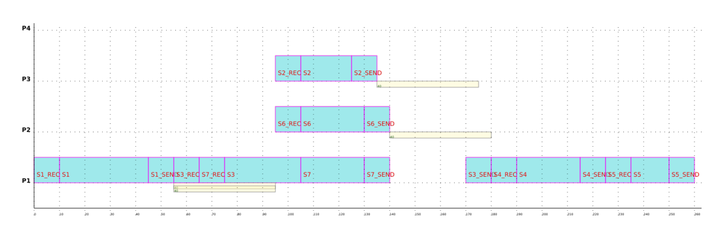

Figure 7: TDM arbitration without synchronisation – Global WCET is 260

① UNIVERSITÉ DE RENNES 1 ② Inria INVENTEURS DU MONDE NUMÉRIQUE ③ UNIVERSITÉ DE RENNES 1 ④ Inria INVENTEURS DU MONDE NUMÉRIQUE