Implementation of Real-Time Data-Flow Synchronous Programs on a Many-Core Architecture

Matheus Schuh
C. Maiza, P. Raymond, B. D. Dinechin

CAPITAL Workshop
24/02/2020
### Basic Concepts

#### Real-Time Systems
- Must provide valid outputs before a deadline
- Timing constraints are part of the specification
- Soft/Hard Real-Time: according to criticality of application

#### Synchronous Data-Flow Languages
- Network of nodes
- Dependencies and thus order requirements
- Lustre, SCADE, Blech
From Single-Core to Many-Core Code Generation

Static sequential schedule from data dependencies

Simple execution

Parallelism extraction

Communication

Synchronization

Scheduling decisions: #cores order

WCET

WCRT = WCET + interferences

Matheus Schuh
Synchronous Data-Flow Programs on Many-Core
Interference and Response Time

Worst-Case Response Time (WCRT)

- Most precise approach is too complex
- Naive approach is too pessimistic

Timing analysis

- Knowledge of hardware: MPPA2 / MPPA3
- Knowledge of software: Synchronous Data-Flow
- Time-triggered execution: interference control

ढ Multi-Core Interference Analysis (MIA) tool
## Multi-Core Interference Analysis

### Inputs
- Initial release date of tasks
- Period and deadline
- Dependencies between the tasks
- WCET in isolation + WC number of accesses

### Outputs
- Schedulable / Not Schedulable
- Final release dates of tasks
  - Minimize interference
  - Predictable runtime
Contribution: Improvement on MIA Tool

Original MIA

- Complexity of $O(n^4)$
  - Where $n$ is the number of tasks
- Stopped converging for hundred of tasks

New interference calculation algorithm

- Accepted paper @ DATE 2020
- Complexity of $O(n^2)$
  - No nested loops within all tasks
  - No fixed-point iteration
Framework Execution Model

- Platform
  - Bare metal
  - Mono-rate non-preemptive static schedule
  - Mapping between runnables and cores done by external tool

- Task activation
  - Time-triggered execution
  - MIA: release dates respecting data dependencies and timing

- Banked Memory
  - One bank for each core: code, input buffers and local variables
  - Execute in a local bank, write to a remote bank
  - Interference on communication only
Framework Overview

Data-flow application

Parallelism Extraction

Functional Code

Mapping + Non-preemptive scheduling

Communication and Dependency graph

Code Generation System + Communication

WCET Analysis

Executable for Kalray

MIA

release dates
Different approaches

- Temporal Isolation: Quentin Perret
  - Application domain: avionic
  - Phased execution that forces isolation

- Run-time adaptation: Stefanos Skalistis
  - Parallel interference-sensitive run-time adaptation mechanism
  - Based on the actual execution time of tasks

- Interference Delay into schedulability analysis: Benjamin Rouxel
  - Contention-aware scheduling strategies
  - Minimize the pessimism of the global response time

- Compiler-level Integration: Dumitru Potop-Butucaru
  - Real-time systems compilation
  - Allows interferences for better efficiency
Thesis objectives

- Right abstraction level for efficient implementation of Real-Time applications
  - RTOS\(^1\)
  - High-level communication layer, such as DDS\(^2\)
  - More generic than bare metal w/o losing flexibility

- Versatile model of computation
  - Programming Environments
    - Lustre/SCADE
    - Simulink
    - Giotto
  - Mixed criticality

\(^1\)Real-Time Operating System

\(^2\)Data Distribution Service
Revisited Framework Overview

Versatile Model of Computation
- SDF (Lustre, SCADE)
- HLD (Simulink)
- ADL (Giotto, Prelude)

High-level communication

RTOS

Kalray Low-Level libraries
- libc, libm, libgcc, libgloss

Kalray k1c Hardware

Thesis Contribution
Ongoing/Future work

**Ongoing**

- PREM on MPPA2
- SCADE MPPA3 Integration
  - Interference Model of MPPA3
  - Code Generation Framework

**Future**

- Experiments with RTOS tasks generation
- Logical Execution Time (LET)
Thanks for your attention!

Questions?

You can find me at
matheus.schuh@univ-grenoble-alpes.fr
http://www-verimag.imag.fr/~schuhm/