ILP-based Modulo Scheduling for High-level Synthesis

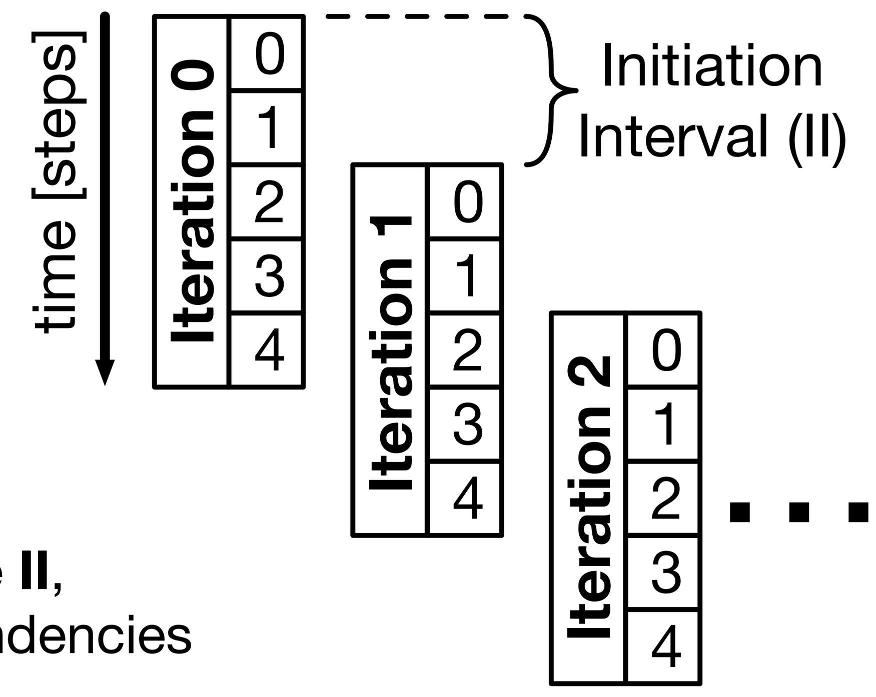
Julian Oppermann **Andreas Koch**
Embedded Systems & Applications Group
Technische Universität Darmstadt, Germany

Melanie Reuter-Oppermann
Discrete Optimization & Logistics Group
Karlsruhe Institute of Technology, Germany

Oliver Sinnen
Parallel & Reconfigurable Computing Group
University of Auckland, New Zealand

Loop pipelining

- Start new loop iterations after a fixed number of time steps, called **Initiation Interval (II)**
- Partially overlapping execution of subsequent loop iterations → **resource constraints** on **congruence classes (modulo II)** of time steps
- Primary objective: Find schedule with **smallest feasible II**, subject to resource constraints and inter-iteration dependencies



Typical modulo scheduler

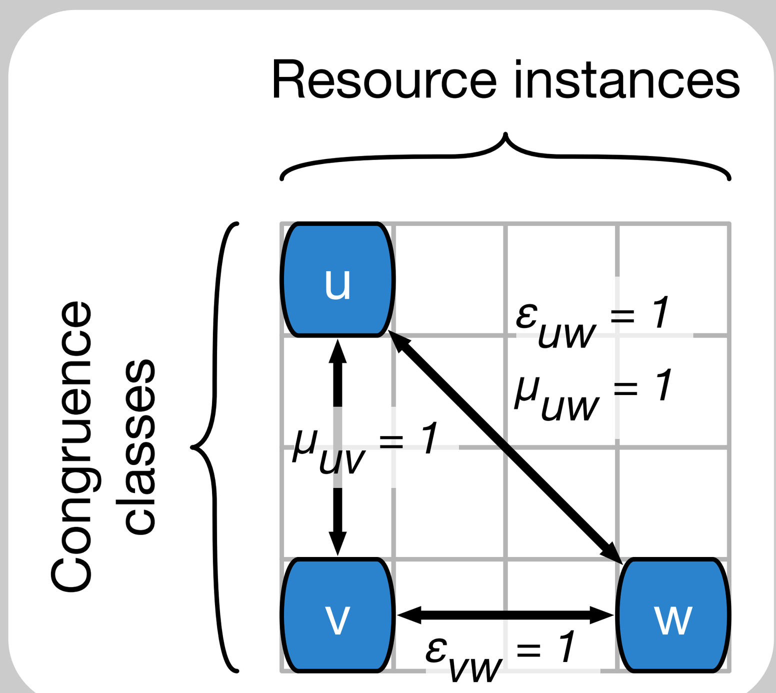
- Determine lower and upper bound for the II
- Select a **candidate II** from that range and try to find a feasible modulo schedule
  - Input: candidate II, precedence edges, resource constraints, operation latencies
  - Output: start times for operations, or attempt fails

Approaches (based on Integer Linear Programs)

- Scheduling without resource constraints is easy, can be done in polynomial time with a System of Difference Constraints (SDC)
- Approaches differ in the modelling of resource constraints
  - There are A_k units of resource kind k and II-1 congruence classes
  - Each resource instance can be used at most by one operation in each congruence class

Moovac (novel)

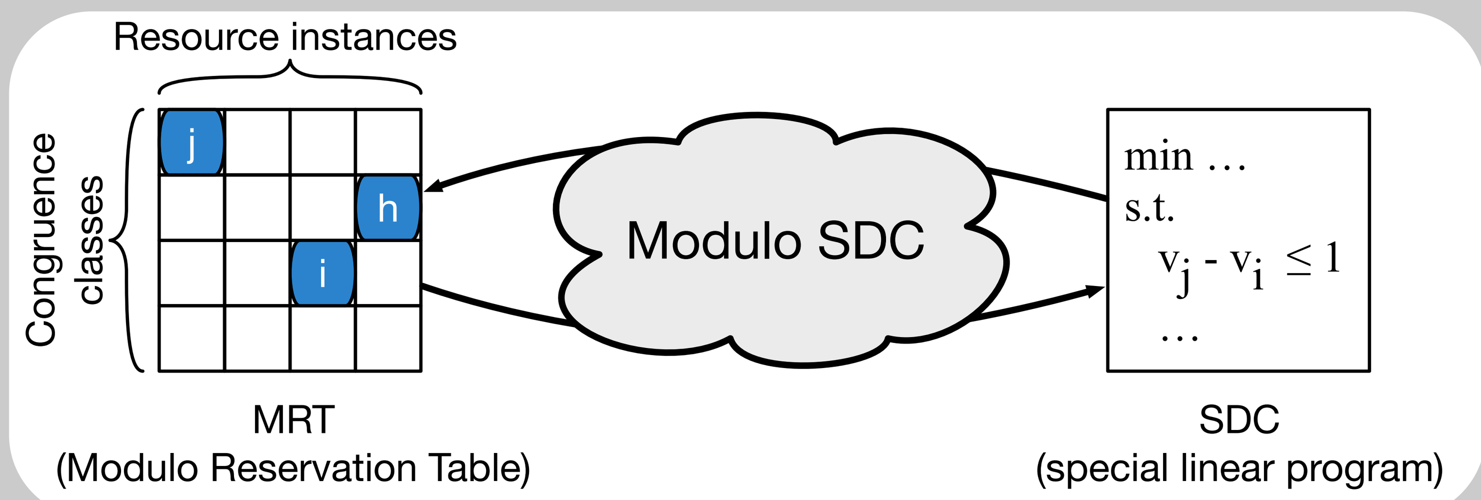
- **Exact** formulation based on an efficient task scheduler
- Uses integer variables to model operations' start times
- Resource assignment modelled by
  - integer variables
    r_i resource instance ID
    m_i congruence class ID
  - binary **overlap** variables
    ε_{ij} := 1 iff. $r_i < r_j$
    μ_{ij} := 1 iff. $m_i < m_j$



- No resource conflict iff. $\varepsilon_{ij} + \varepsilon_{ji} + \mu_{ij} + \mu_{ji} \geq 1$

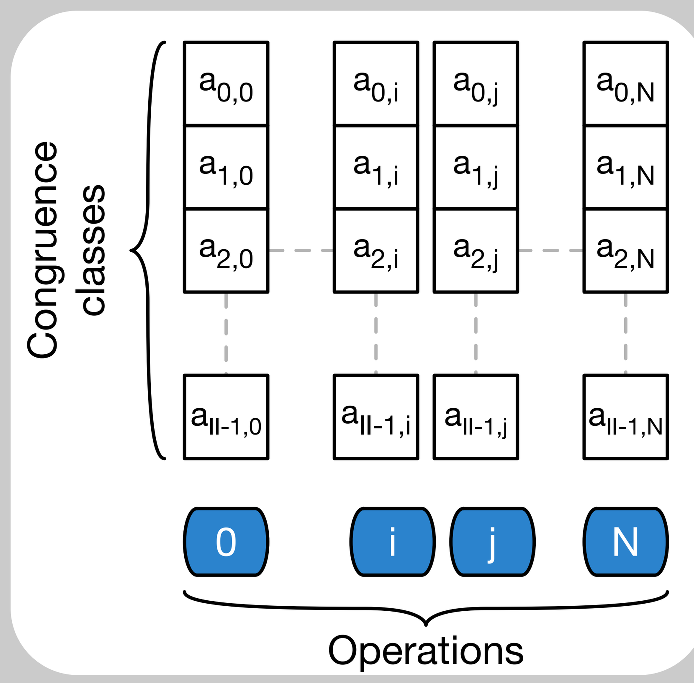
Modulo SDC

- State-of-the-art **heuristic** algorithm using an SDC and an MRT
- Start with a resource-**un**constrained schedule
- Incrementally try to assign ops to MRT / update SDC, until all resource-constrained ops fit in MRT
- Backtracking required if SDC becomes infeasible



Eichenberger's formulation

- **Exact** formulation as general ILP with time-indexed binaries:
  Variable $a_{m,i}$:= "op i starts in congruence class m"
- Resource constraints modelled per congruence class q: $\sum_x a_{q,x} \leq A_k$ (simplified)



Evaluation

- Schedulers implemented with CPLEX 12.6.3, ran single-threadedly on Intel Xeon E5-2667's at 3.3 GHz
- Time limit of 5 min or 60 min per candidate II → increment II if instance is shown to be infeasible, or no solution was found within time budget
- Attempted to schedule 225 loops from CHStone and MachSuite

Scheduling time - 5 min time limit

Graphs		Moovac		Modulo SDC		Eichenberger's ILP	
Size	*#*	*Time [min]*	*Timeouts*	*Time [min]*	*Timeouts*	*Time [min]*	*Timeouts*
all	225	489	96	753	148	932	177
small	203	3	0	131	26	5	0
large	22	486	96	623	122	927	177

- Moovac is surprisingly fast; Moovac + M. SDC synergistically is even faster: **429 min**
- Fruitless attempts dominate overall time. Heuristic can struggle with small graphs.

Result quality - 5 min time limit

- Compared to time-limited Moovac, Modulo SDC finds schedules with…
  - the same II for 217 of 225 graphs
  - a worse II for 6 of 225 graphs
  - a better II for 2 of 225 graphs

Contact {oppermann, koch}@esa.tu-darmstadt.de melanie.reuter@kit.edu o.sinnen@auckland.ac.nz

TECHNISCHE UNIVERSITÄT DARMSTADT
KIT Karlsruhe Institute of Technology
THE UNIVERSITY OF AUCKLAND Te Whare Wānanga o Tāmaki Makaurau NEW ZEALAND