GeMS: A Generator for Modulo Scheduling Problems

Julian Oppermann[1] Sebastian Vollbrecht[1] Melanie Reuter-Oppermann[2] Oliver Sinnen[3] Andreas Koch[1]

1 Embedded Systems and Applications Group, Technische Universität Darmstadt
2 Discrete Optimization and Logistics, Karlsruhe Institute of Technology
3 Parallel and Reconfigurable Computing Lab, University of Auckland

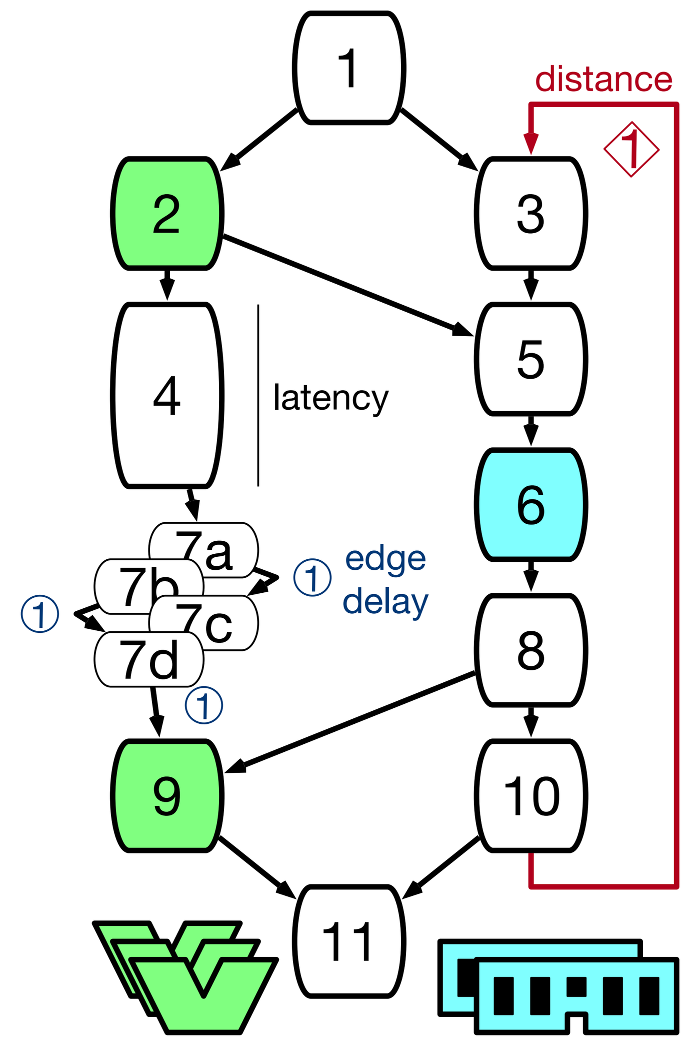
Why generate problems?

- Finding an **optimal** solution to the MSP is NP-hard
- But, we were stubborn…
  - observed **most** MSPs in a high-level synthesis context **can** be solved with an **exact, ILP-based** scheduler
  - only a **handful** instances are slow or intractable, too few to reason about
- Generated problems "fill the gaps" between the benchmark instances
  - small/large, sparse/dense, few/many limited operations, …
  - investigate what's **"hard"** for a particular scheduler
  - long-term goal: build an **oracle** that picks the "right" scheduler for a given instance

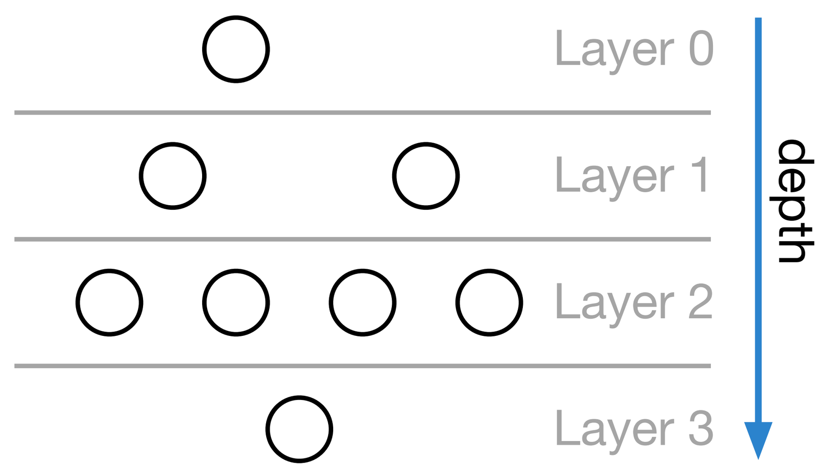
Formal definition

An instance of the **modulo scheduling problem** (MSP) is defined by:

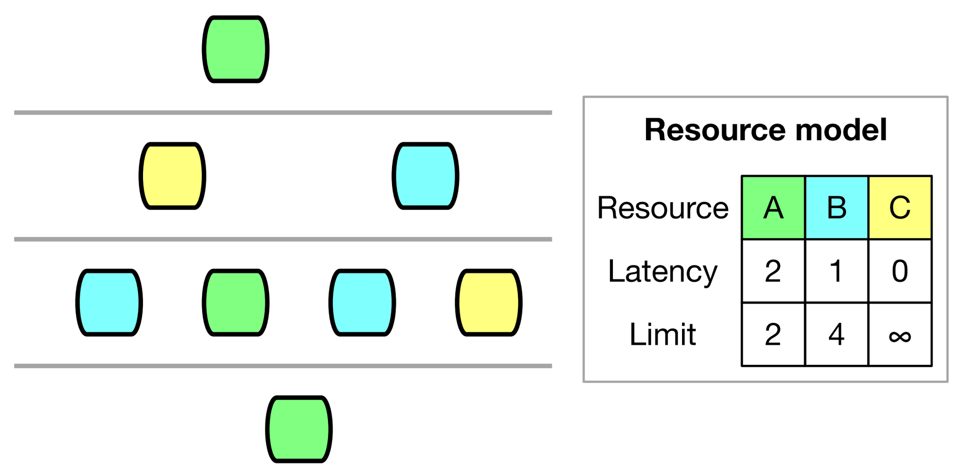
- **Resources types** *r*
  - latency
  - # available units (or ∞)
- **Operations** *i*
  - mapping to resource type
- **Edges** *i* → *j*
  - delay (e.g. to control operator chaining)
  - distance (≥ 1 for inter-iteration dependences)
- **Solution:** initiation interval (II), start times for operations



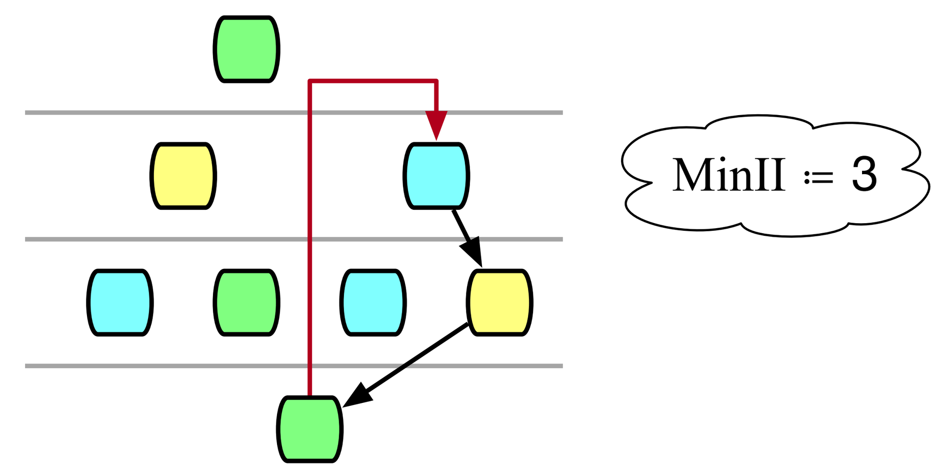
Generation approach



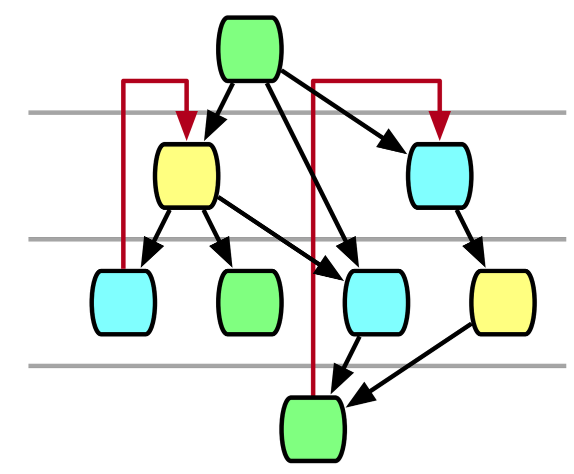
1 Build layer structure



2 Map operations to (user-specified) resource types



3 Establish MinII (optional)



4 Construct edges

Generating graphs with known MinII

- **MinII** = lower bound for optimal II, induced by cycles and resource constraints
  - schedulers usually try several candidate IIs until a feasible solution is found
  - important to keep number of tried candidate IIs the same when comparing scheduler runtimes
- GeMS allows a **desired MinII**, and whether the MSP shall be **feasible or infeasible** at that MinII, to be specified
  - if needed, picks operations to construct a cycle (step 3) to raise the graph's MinII
  - checks prevent that edges (generated in step 4) change the desired MinII or its feasibility
  - **the rest of the MSP is still randomly generated!**

Code example

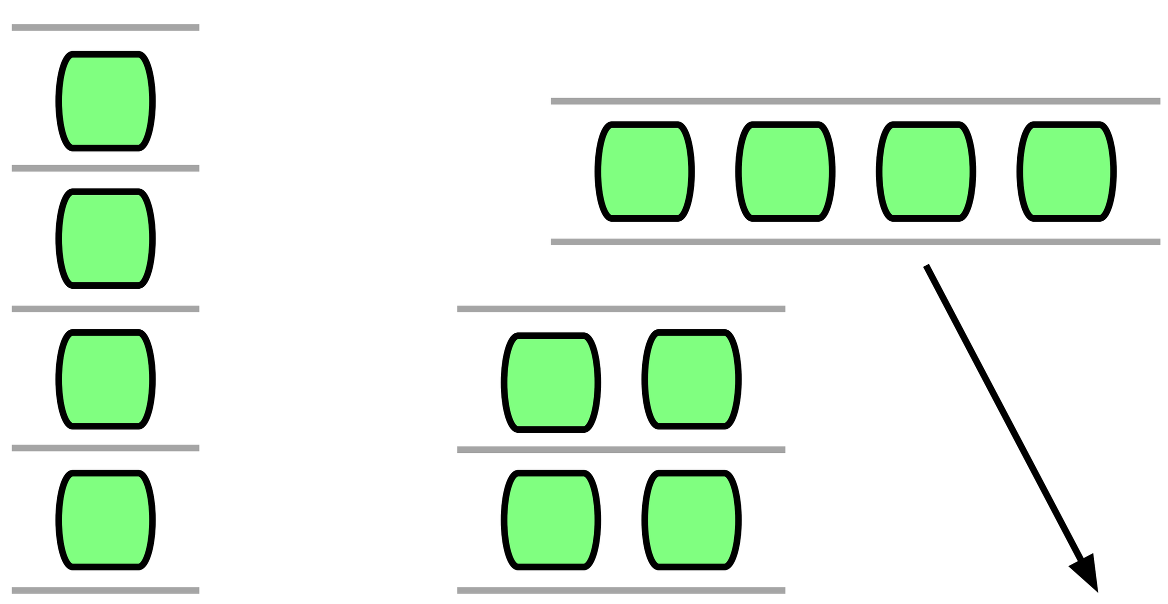
```
Resource resA = new Resource("A", 2, 2); Resource resB = new Resource("B", 1, 4);
Resource resC = new Resource("C", 0);

GraphGenerator gen = new GraphGenerator(
      new FixedShapeLayerCreator(/* nodes in layer */ 1, 2, 4, 1),
      new DistributionNodeCreator(new ProbabilityDistribution<>(resA, resB, resC)),
      new EdgeCreator(
          /* edge delay        */ new ConstantValueComputer(0),
          /* backedge delay    */ new ConstantValueComputer(0),
          /* backedge distance */ new ConstantValueComputer(1)),
      /* forward edges */ new ProbabilityEdgeIncluder(0.0075),
      /* backedges     */ new ProbabilityEdgeIncluder(0.0030)
);
GraphFileUtils.graphToHatScheTFiles(gen.createGraph(/* seed */ 42), "graph");
```

- GeMS is a **toolkit** written in Java, offers no CLI
- Graph representation is simple (~nodes+edges)
  - supplied export facilities: DOT, and format used by **HatScheT scheduler library**

Case study

- **Question**: How does the Moovac formulation [CASES'16] cope with symmetry?
- **Experiment**
  - 1 resource type with 2 instances
  - 48 operations in **different layer structures** compete for this resource type
- **Result/insight**
  - the more operations in parallel, the harder for Moovac to find/prove an optimal solution



#layers x #ops	48x1	24x2	16x3	12x4	8x6	6x8	4x12	2x24	1x48
avg time [s]	3.0	116.5	3600	3600	3600	3600	3600	3600	3600
avg gap [%]	opt.	opt.	29	45	61	69	77	88	88*

Average over 10 random instances ∗) No solution found for 2 instances

Source code available

GeMS: HatScheT:

Outlook

- Add support for specifying the number of incoming edges (e.g. #operands)
- Finer control over the MSP's II (e.g. "be feasible at MinII+3")

{oppermann, vollbrecht, 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