GeMS: A Generator for Modulo Scheduling Problems

Julian Oppermann, Sebastian Vollbrecht, Melanie Reuter-Oppermann, Oliver Sinnen, Andreas Koch
**Modulo Scheduling**

- **Loop/software pipelining**
  
  = increase throughput by overlapping iterations

- **Modulo schedulers**
  
  compute
  
  • initiation interval (II)
  
  • start times (= „schedule“)

- **Subject to**
  
  • inter-iteration dependences
  
  • resource constraints
Why Generate Problems?

- Finding an optimal II and schedule is NP-hard,
  - only a few problem instances are slow/intractable
  - what’s a „hard“ instance for a particular scheduler?

- GeMS-generated instances „fill the gaps“
  - small/large, sparse/dense, …

- Long-term: build an oracle

[CASES’16] Oppermann et al.: ILP-based Modulo Scheduling for High-level Synthesis
[FPL’18] Oppermann et al.: Dependence Graph Preprocessing for Faster Exact Modulo Scheduling in HLS
Generation Approach

1) Generate layer structure

Layer 0
Layer 1
Layer 2
Layer 3

depth

- **MinII** = lower bound for the optimal II
  - schedulers typically try several **candidate II**s

2) Map operations to resource types

3) Establish user-specified MinII (optional)

- GeMS picks operations to form a special cycle if needed
  - can be **feasible** or **infeasible** at that II
Case Study

- 48 operations compete for resource type with 2 units
- Using layer structures with increasing parallelism among the operations

---

[CASES’16] Oppermann et al.: ILP-based Modulo Scheduling for High-level Synthesis
GeMS is a **toolkit** written in Java → no CLI

- source available under Apache License

```java
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");
```

- Export generated graphs directly to **HatScheT** scheduler library
  - great for scheduler research!
Thanks, see you at the poster!
oppermann@esa.tu-darmstadt.de

Get GeMS

Try HatScheT,
a toolkit for schedulers