Dependence Graph Preprocessing for Faster Exact Modulo Scheduling in High-level Synthesis

Julian Oppermann, Melanie Reuter-Oppermann, Lukas Sommer, Oliver Sinnen, Andreas Koch
Agenda

- (Short) introduction to modulo scheduling
- Proposed preprocessing approach
- Results and insights
- **Loop pipelining**
  = increase throughput by overlapping iterations
Modulo Scheduling

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

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

![Diagram showing Modulo Scheduling with iterations and time steps]

- **Iterations**
  - 0
  - 1
  - 2

- **Time steps**
  - 0
  - 1
  - 2
  - 3
  - 4
  - 5
  - 6
  - 7
  - 8
  - 9
  - 10

- **ld [0]**
- **ld [1]**
- **add**
- **ld [1]**
- **st [2]**
- **ld [2]**
- **add**
- **ld [2]**
- **st [3]**
- **ld [3]**
- **add**
- **st [4]**

- **Initiation Interval (II)**
Modulo Scheduling

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

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

- Subject to
  • inter-iteration dependences
Modulo Scheduling

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

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

- Subject to
  • inter-iteration dependences
  • resource constraints
Modulo Scheduling

- **Loop pipelining**
  - increase throughput by overlapping iterations

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

- Subject to
  - inter-iteration dependences
  - resource constraints

- Minimise
  1. initiation interval
Modulo Scheduling

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

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

- Subject to
  - inter-iteration dependences
  - resource constraints

- Minimise
  1. initiation interval
  2. schedule length
An **instance** of the **modulo scheduling problem** (MSP) is defined by:
An instance of the modulo scheduling problem (MSP) is defined by:

- Operations $i$
  - latency: 0 (combinatorial), 1, 2, …
Formal Definition

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

- Operations $i$
  - latency: 0 (combinatorial), 1, 2, …

- Edges $i \rightarrow j$
  - (delay), e.g. to control chaining
  - $<$distance$>$, $\geq 1$ for inter-iteration dependences („backedges“)
Formal Definition

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

- **Operations** $i$
  - latency: 0 (combinatorial), 1, 2, …

- **Edges** $i \rightarrow j$
  - (delay), e.g. to control chaining
  - $<\text{distance}>, \geq 1$ for inter-iteration dependences („backedges“)

- **Resource model**
  - distinct types with given #units
  - unlimited operations
Modulo scheduling is usually applied to:
Modulo scheduling is usually applied to:
- Very-long-instruction-word (VLIW) architectures
- majority of literature targets VLIW compilers
Motivation

- Modulo scheduling is usually applied to:
  - Very-long-instruction-word (VLIW) architectures
    - majority of literature targets VLIW compilers
  - High-level synthesis (HLS)
    - larger and denser dependence graphs
    - not all operations are resource-constrained
Motivation

- Modulo scheduling is usually applied to:
  - Very-long-instruction-word (VLIW) architectures
    - majority of literature targets VLIW compilers
  - High-level synthesis (HLS)
    - larger and denser dependence graphs
    - not all operations are resource-constrained

- Observed scalability issues in highly-tuned approach
  → Can’t we just simplify the problem?
Benchmark Set

- 21 medium to large MSP instances
  - from CHStone and MachSuite
  - extracted from a typical HLS flow
Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**

- Median values
  - 471 operations
Benchmark Set

- 21 medium to large MSP instances
  - from CHStone and MachSuite
  - extracted from a typical HLS flow

- Median values
  - 471 operations
    - 12% resource-limited operations
Benchmark Set

- 21 medium to large MSP instances
  - from CHStone and MachSuite
  - extracted from a typical HLS flow

- Median values
  - 471 operations
    - 12% resource-limited operations
  - 1578 edges
Benchmark Set

- **21** medium to large MSP instances
  - from **CHStone** and **MachSuite**
  - extracted from a **typical HLS flow**

- Median values
  - 471 operations
    - 12% resource-limited operations
  - 1578 edges
    - 63% non-dataflow edges (memory dependences, chaining control)
(Short) introduction to modulo scheduling

Proposed preprocessing approach

Results and insights
Basic Insight

Some operations are critical for the scheduling result, as they influence the

- **feasibility**, and/or the
- **objective value** of the solution
Basic Insight

- Some operations are **critical** for the scheduling result, as they influence the
  - **feasibility**, and/or the
  - **objective value** of the solution

- Others can always be scheduled ASAP
Basic Insight

- Some operations are critical for the scheduling result, as they influence the
  - feasibility, and/or the
  - objective value of the solution

- Others can always be scheduled ASAP
  - find subgraphs of non-critical operations
Basic Insight

- Some operations are **critical** for the scheduling result, as they influence the
  - **feasibility**, and/or the
  - **objective value** of the solution

- Others can always be scheduled ASAP
  - find subgraphs of **non-critical** operations
  - replace by a single edge with appropriate delay
Approach Overview

Original instance
Approach Overview

1. Construct reduced instance
   a) Determine critical operations
   b) Construct new edges
   c) Filter redundant edges

Original instance → Reduced instance
Approach Overview

1. Construct reduced instance
   - a) Determine critical operations
   - b) Construct new edges
   - c) Filter redundant edges

2. Modulo Scheduler

Solution for reduced instance

Original instance → Reduced instance
Approach Overview

1. Construct reduced instance
   - a) Determine critical operations
   - b) Construct new edges
   - c) Filter redundant edges

2. Modulo Scheduler
   - Solution for reduced instance

3. Complete schedule
   - Solution for original instance
1a) Determine Critical Operations

- Feasibility
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
  - endpoints of backedges
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
  - endpoints of backedges

- Objective
1a) Determine Critical Operations

- Feasibility
  - resource-limited operations
  - endpoints of backedges

- Objective
  - source and sink nodes
1b) Construct New Edges

- Single-pass data-flow analysis over dependence graph
1b) Construct New Edges

- Single-pass data-flow analysis over dependence graph

- For each operation, compute the longest paths to the nearest preceding critical operations
  - "length" = accumulated latencies and delays
1b) Construct New Edges

- Single-pass data-flow analysis over dependence graph

- For each operation, compute the \textit{longest paths} to the \textit{nearest} preceding critical operations
  - „length“ = accumulated latencies and delays
  - paths are \textit{reset} when encountering a critical operation
1b) Construct New Edges

- Example of data-flow analysis
b) Construct New Edges

- Example of data-flow analysis

![Graph example]

- Path length to preceding critical operation
1b) Construct New Edges

- Example of data-flow analysis

```
A IN  -  OUT  0
B IN  -  OUT  0
C IN  1  OUT  1
D IN  -  OUT  -
E IN  1  OUT  -
F IN  -  OUT  -
G IN  -  OUT  -
H IN  -  OUT  -
J IN  -  OUT  -
```

- Path length to preceding critical operation
- "Reset" paths to other critical ops
1b) Construct New Edges

- Example of data-flow analysis

- Path length to preceding critical operation
1b) Construct New Edges

- Construct edges between reachable critical operations
Construct edges between reachable critical operations

- need to subtract source operation’s delay
1b) Construct New Edges

- Construct edges between reachable critical operations
  - need to subtract source operation’s delay

- Backedges are copied over from original instance
1c) Filter Redundant Edges

- Precedence constraints may be modelled transitively
1c) Filter Redundant Edges

- Precedence constraints may be modelled transitively
- Find and remove such edges
2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler
2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler
  
  - **Here**: using **exact** formulations with integer linear programs (ILP)
2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler

- **Here:** using **exact** formulations with integer linear programs (ILP)

- Eichenberger & Davidson (1997) "ED"
2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler

  - **Here**: using **exact** formulations with integer linear programs (ILP)

  - Eichenberger & Davidson (1997) „ED“

  - Oppermann et al. (2016) „Moovac“
2) Modulo Scheduling

- **Reduced** instance is now scheduled with an arbitrary modulo scheduler

  - **Here**: using **exact** formulations with integer linear programs (ILP)
  
  - Eichenberger & Davidson (1997) „ED“
  
  - Oppermann et al. (2016) „Moovac“

→ different approaches to model operations’ start times and resource limits
3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations
3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations
  - feasible/optimal for **original** instance
3) Schedule Completion

- Modulo scheduling the reduced instance yields:
  - initiation interval
  - start times for critical operations
  → feasible/optimal for original instance

- To be computed: start times for non-critical operations
3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations
  → feasible/optimal for **original** instance

- To be computed: start times for non-critical operations
  - fix start times of critical operations in **original** instance, and schedule!
3) Schedule Completion

- Modulo scheduling the **reduced** instance yields:
  - initiation interval
  - start times for critical operations
  → feasible/optimal for **original** instance

- To be computed: start times for non-critical operations
  - fix start times of critical operations in **original** instance, and schedule!
  → easy (polynomially) to solve, because no longer resource-constrained
Agenda

☑️ (Short) introduction to modulo scheduling

☑️ Proposed preprocessing approach

▶ Results and insights
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time

- **Time limit** (per candidate II): 3 minutes
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time

- **Time limit** (per candidate II): 3 minutes

- **CPLEX** 12.6.3 as ILP solver, 8x multithreaded
Evaluation Setup

- **21 instances** from CHStone and MachSuite that took longer than 10 sec to schedule with ED or Moovac
  - 167 other loops were scheduled optimally with both approaches in less time

- **Time limit** (per candidate II): 3 minutes

- **CPLEX** 12.6.3 as ILP solver, 8x multithreaded

- Ran on 24-core Xeon E5-2680 v3, 2.8 GHz, 64 GB RAM
Results: Graph Reduction

![Graph Reduction Chart]

- **# operations**
- **# edges**

### Instance

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21

### Reduced to...

- 0%
- 10%
- 20%
- 30%
- 40%
- 50%
- 60%
- 70%
- 80%
- 90%
- 100%
Results: Graph Reduction

- Time for complexity reduction: always < 0.5 sec
Results: ILP Reduction

<table>
<thead>
<tr>
<th>Instance</th>
<th>Mean # Variables</th>
<th>Mean # Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>ED: variables</td>
<td>ED: constraints</td>
<td>Moovac: variables</td>
</tr>
<tr>
<td>0 %</td>
<td>10 %</td>
<td>20 %</td>
</tr>
<tr>
<td>0 %</td>
<td>10 %</td>
<td>20 %</td>
</tr>
</tbody>
</table>
Results: ILP Reduction

Reduced to...

<table>
<thead>
<tr>
<th>Instance</th>
<th>ED: variables</th>
<th>ED: constraints</th>
<th>Moovac: variables</th>
<th>Moovac: constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40%</td>
<td>30%</td>
<td>5%</td>
<td>5%</td>
</tr>
<tr>
<td>2</td>
<td>30%</td>
<td>20%</td>
<td>4%</td>
<td>4%</td>
</tr>
<tr>
<td>3</td>
<td>20%</td>
<td>10%</td>
<td>3%</td>
<td>3%</td>
</tr>
<tr>
<td>4</td>
<td>10%</td>
<td>0%</td>
<td>2%</td>
<td>2%</td>
</tr>
<tr>
<td>5</td>
<td>0%</td>
<td>0%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>6</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>7</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>8</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>9</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>10</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>11</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>12</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>13</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>14</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>15</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>16</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>17</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>18</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>19</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>20</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>21</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean #</th>
<th>ED</th>
<th>ED (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>37.2 k</td>
<td>5.1 k</td>
</tr>
<tr>
<td>Constraints</td>
<td>113.4 k</td>
<td>26.7 k</td>
</tr>
</tbody>
</table>
Results: ILP Reduction

<table>
<thead>
<tr>
<th>Instance</th>
<th>ED: variables</th>
<th>ED: constraints</th>
<th>Moovac: variables</th>
<th>Moovac: constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variables</td>
<td>37.2 k</td>
<td>5.1 k</td>
<td>8.4 k</td>
<td>7.9 k</td>
</tr>
<tr>
<td>Constraints</td>
<td>113.4 k</td>
<td>26.7 k</td>
<td>28.4 k</td>
<td>27.1 k</td>
</tr>
</tbody>
</table>
Results: Runtime

- ED formulation
- Moovac formulation

Accumulated runtime

Speed-up (geomean)
Results: Runtime

- **ED formulation**
- **Moovac formulation**

<table>
<thead>
<tr>
<th>Instance</th>
<th>ED</th>
<th>ED (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accumulated runtime</td>
<td>328 min</td>
<td>268 min</td>
</tr>
<tr>
<td>Speed-up (geomean)</td>
<td>4.37x</td>
<td></td>
</tr>
</tbody>
</table>
Results: Runtime

<table>
<thead>
<tr>
<th>Instance</th>
<th>ED</th>
<th>ED (red.)</th>
<th>Moovac</th>
<th>Moovac (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Accumulated runtime</td>
<td>328 min</td>
<td>268 min</td>
<td>290 min</td>
</tr>
<tr>
<td></td>
<td>Speed-up (geomean)</td>
<td>4.37x</td>
<td></td>
<td>0.8x</td>
</tr>
</tbody>
</table>
## Results: Solution Quality

<table>
<thead>
<tr>
<th># instances</th>
<th>ED</th>
<th>ED (red.)</th>
<th>Moovac</th>
<th>Moovac (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>11</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>optimal II</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>feasible</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>no solution</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
### Results: Solution Quality

<table>
<thead>
<tr>
<th># instances</th>
<th>ED</th>
<th>ED (red.)</th>
<th>Moovac</th>
<th>Moovac (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>11</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>optimal II</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>feasible</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>no solution</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- More instances are tractable for ED formulation
## Results: Solution Quality

<table>
<thead>
<tr>
<th># instances</th>
<th>ED</th>
<th>ED (red.)</th>
<th>Moovac</th>
<th>Moovac (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>11</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>optimal II</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>feasible</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>no solution</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- More instances are tractable for ED formulation
## Results: Solution Quality

<table>
<thead>
<tr>
<th># instances</th>
<th>ED</th>
<th>ED (red.)</th>
<th>Moovac</th>
<th>Moovac (red.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal</td>
<td>11</td>
<td>14</td>
<td>13</td>
<td>12</td>
</tr>
<tr>
<td>optimal II</td>
<td>2</td>
<td>-</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>feasible</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>no solution</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

- More instances are tractable for ED formulation
- Again, minor regression for Moovac
**Discussion**

- **ED**: Significant benefits
  - both in terms of *runtime* and *solution quality*
  - additional effort for problem reduction is negligible
Discussion

- **ED**: Significant benefits
  - both in terms of **runtime** and **solution quality**
  - additional effort for problem reduction is negligible

- **Moovac**: ILP complexity dominated by resource-limited operations
  - not enough reduction potential to offset ILP solvers’ „**performance variability““
Discussion

- **ED**: Significant benefits
  - both in terms of **runtime** and **solution quality**
  - additional effort for problem reduction is negligible

- **Moovac**: ILP complexity dominated by resource-limited operations
  - not enough reduction potential to offset ILP solvers’ “**performance variability**”

- Both now **much closer** performance-wise
  - seem to **complement** each other
Can’t we just simplify the problem?
Can’t we just simplify the problem?

• Yes!
Can’t we just simplify the problem?

- Yes!
- other, similar ILP formulations exist
Conclusion & Outlook

- Can’t we just simplify the problem?
  - Yes!
  - other, similar ILP formulations exist

- Long-term goal: an oracle
  - select the „right“ modulo scheduler for a given instance
Conclusion & Outlook

■ Can’t we just simplify the problem?
  • Yes!
  • other, similar ILP formulations exist

■ Long-term goal: an oracle
  • select the „right“ modulo scheduler for a given instance
    → important to have different schedulers that scale roughly the same
Thank you!
oppermann@esa.tu-darmstadt.de