Improving HLS Generated Accelerators Through Relaxed Memory Access Scheduling

Johanna Rohde

Christian Hochberger

TU Darmstadt

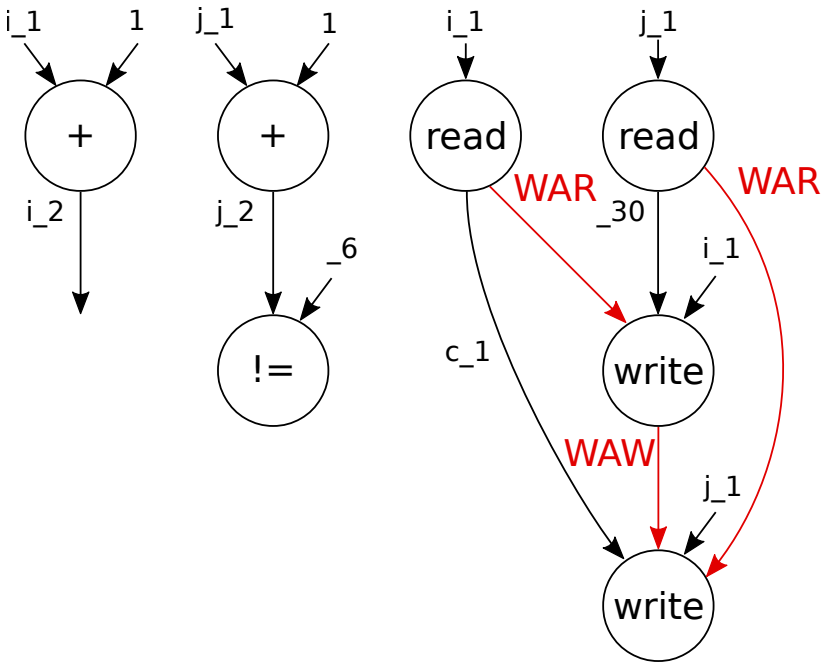
Computer Systems Group

Introduction

- High-Level-Synthesis (HLS) can be used to create hardware accelerators from software code.
- For a meaningful acceleration, accelerator must be capable to autonomously access the memory.
- Such memory accesses can be a major bottleneck because multiple accesses to the memory can target the same memory location.
- In some cases it can be proven at compile time that two accesses are independent. Otherwise, their order must be preserved which typically leads to an increased cycle times for the execution of one iteration.
- 3 types of memory dependences:
  - Read-After-Write (RAW)
  - Write-After-Write (WAW)
  - Write-After-Read (WAR, not optimized by our approach. Please refer to our paper for a detailed discussion)

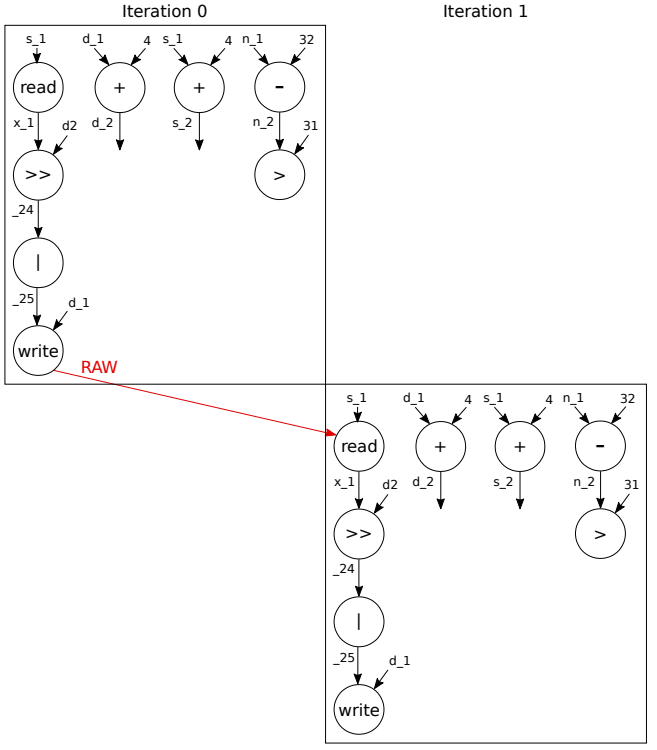
Problem Statement

- Example 1: WAW Dependence
  - Memory dependences are represented by red edges.
  - The data flow graph (DFG) has a critical path of 3.
  - The critical path could be 2 if it was not for the WAW dependence.



Problem Statement

- Example 2: RAW Dependence
  - By pipelining consecutive iterations, the next iteration is started before the previous one has finished.
  - In the example, loop pipelining is not possible because the read instruction has to be scheduled after the write instruction.



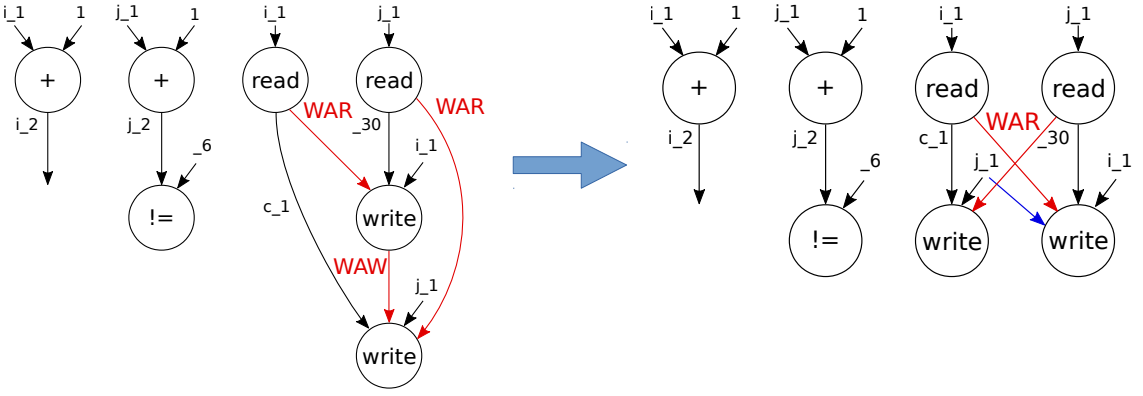
Relaxing Memory Dependences

TECHNISCHE UNIVERSITÄT DARMSTADT

- In hardware accelerators additional logic can be added in order to resolve out-of-order memory dependence conflicts at runtime.
- Idea:
  - Enable a speculative out-of-order execution of memory accesses. This will lead to a more relaxed DFG providing additional degrees of freedom to the scheduling algorithm in order to create shorter and faster schedules.
  - The correctness of the data is ensured at runtime by bypassing and write squashing.
  - Note, this approach does not replace static dependence analysis. It rather adds on top of it by focusing on those dependences for which the synthesis tool was not able to prove or disprove aliasing.

Relaxing Memory Dependences

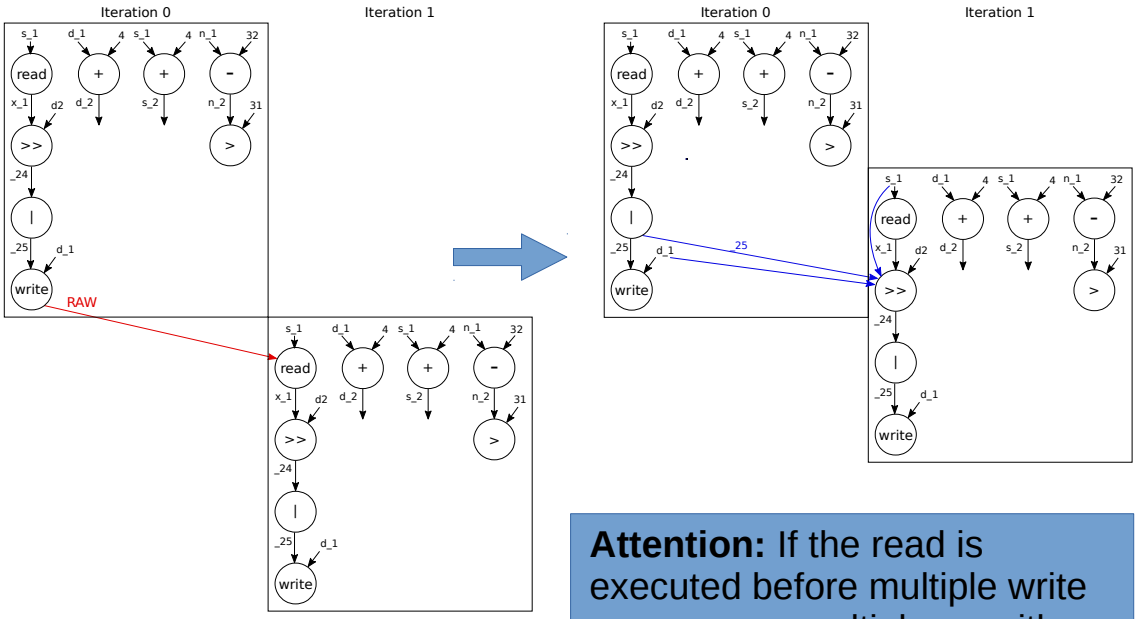
- Write Squashing for WAW Dependences
  - **Idea**: Write squashing refers to the process of disabling a scheduled write instruction at runtime.
  - **Hardware**: A comparator controlling the enable signal of the first write access.
  - **Consequence**: The addresses of both memory accesses need to be known for first write access.
  - **Changes to the DFG**: The WAW dependence edge is replaced with an edge that connects the producer of the second write address with the first write node.



Attention: If multiple writes are executed out-of-order, multiple comparators have to be created and joint by an or-gate.

Relaxing Memory Dependences

- Bypassing for RAW Dependences
  - **Idea**: Execute the read access speculatively before the write and compare the addresses once the data is used. In case of a conflict, the data from the write operation is bypassed and replaces the data read from memory.
  - **Hardware**: A multiplexer of which the select signal is wired to a comparator.
  - **Consequence:** Both addresses as well as the written value need to be known at the successors of the read node.
  - **Changes to the DFG**: The edge of the RAW dependence is removed. New edges for the addresses and the bypass value are inserted.



Attention: If the read is executed before multiple write accesses, a multiplexer with multiple input ports has to be created.

Relaxed Memory Access Scheduling (RMAS)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem: Patching all memory dependences before scheduling is not always possible because the number of comparators for a single write-squash and the number of multiplexer input ports has to be limited.
- Therefore, we developed an algorithm that automatically detects those memory dependences that are critical to the performance.
- Summary:
  - The Relaxed Memory Access Scheduling (RMAS) algorithm searches a DFG for memory dependences that are on the critical path.
  - If possible, those dependences are patched as explained before.
  - This process is repeated until no further improvements can be made.
- For more information, please refer to the paper.

Experimental Results

- The evaluated system consists of a SpartanMC soft-core, the hardware accelerators and a cache system. Everything was synthesized for a Nexys Video Artix-7 (XC7A200T) FPGA board.
- On average, the latency was reduced by 6.46 %. This includes those benchmarks, that were not effected by the optimization because they do not have any critical memory dependences
- On average, inserting additional multiplexer and comparators into the accelerator did not have an significant effect on the critical path
- On average, the number of LUTs increased by 10% and the number of flip-flops increased by 2%.

	Δ Cycles	Ratio Frequency	Ratio LUT	Ratio Flip-Flops
adpcm	0.21 %	1.07	1.00	1.00
bcnt	1.09 %	1.00	1.00	1.00
blit	-39.85 %	1.01	1.42	1.14
compress	-8.05 %	1.05	1.12	0.99
crc	-22.36 %	0.98	1.30	1.07
engine	0.00 %	1.00	1.00	1.00
g3fax	0.03 %	1.01	1.00	1.00
huffman	-0.55 %	1.04	1.26	1.04
jpeg	-0.31 %	0.95	1.04	1.00
pocsag	0.00 %	0.99	1.00	1.00
ucbqsort	-1.22 %	1.01	0.98	0.97
Average	-6.46 %	1.01	1.10	1.02

Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- We have presented an algorithm that relaxes the scheduling of memory accesses by inserting read-bypasses and write squashes into performance critical parts of a DFG.
- Some applications do not contain critical memory accesses. They are neither improved, nor slowed down.
- Some applications contain memory accesses that could be relaxed but the gain was only in parts of the code that did not contribute much to the overall runtime.
- Finally, some applications contain memory accesses that greatly benefit from the relaxed scheduling and gain substantial runtime.
- **On average, the improvement is about 6.5%. The amount of additional hardware resources is in a similar range.**

The End

Questions?

Feedback?

Interested in discussing the details of our paper?

Please contact me:

Johanna Rohde

rohde@rs.tu-darmstadt.de