# CAeSaR: unified Cluster-Assignment Scheduling and communication Reuse for clustered VLIW processors

Vasileios Porpodas [†] and Marcelo Cintra [†⋆]

School of Informatics, University of Edinburgh[†]
Intel Labs Braunschweig[⋆]

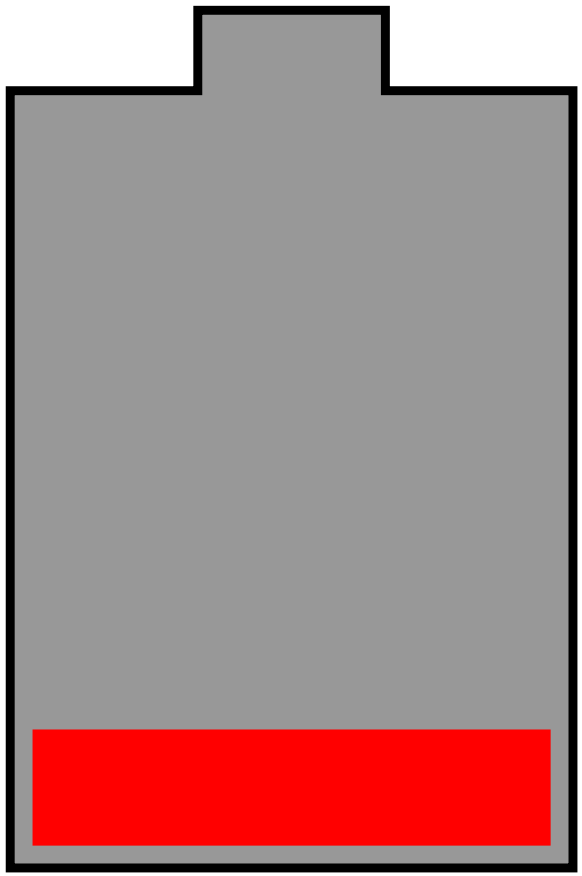
CASES 2013

# Energy efficiency, VLIWs and Scheduling

- Mobile era.

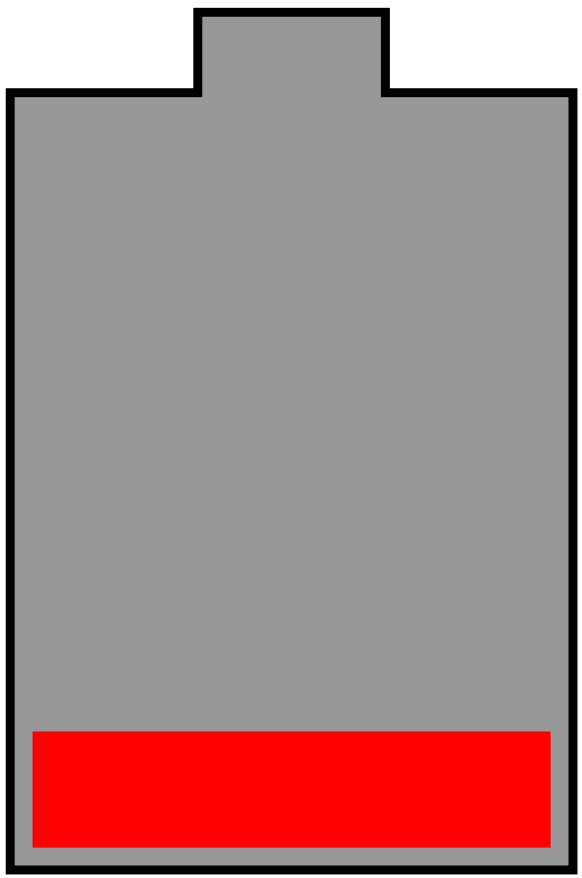
# Energy efficiency, VLIWs and Scheduling

- Mobile era.
- Energy becomes a major design constraint.



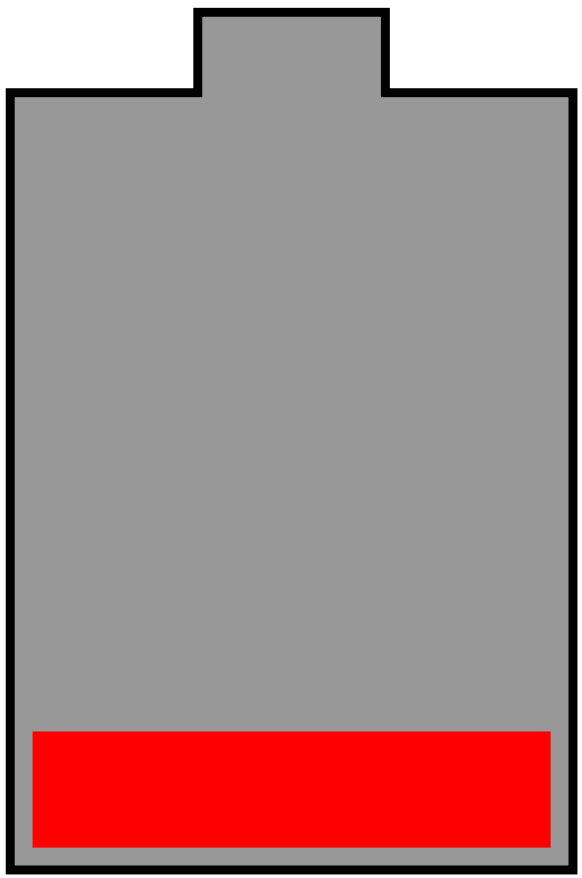
# Energy efficiency, VLIWs and Scheduling

- Mobile era.
- Energy becomes a major design constraint.
- Hardware Instruction scheduling consumes a lot of energy.



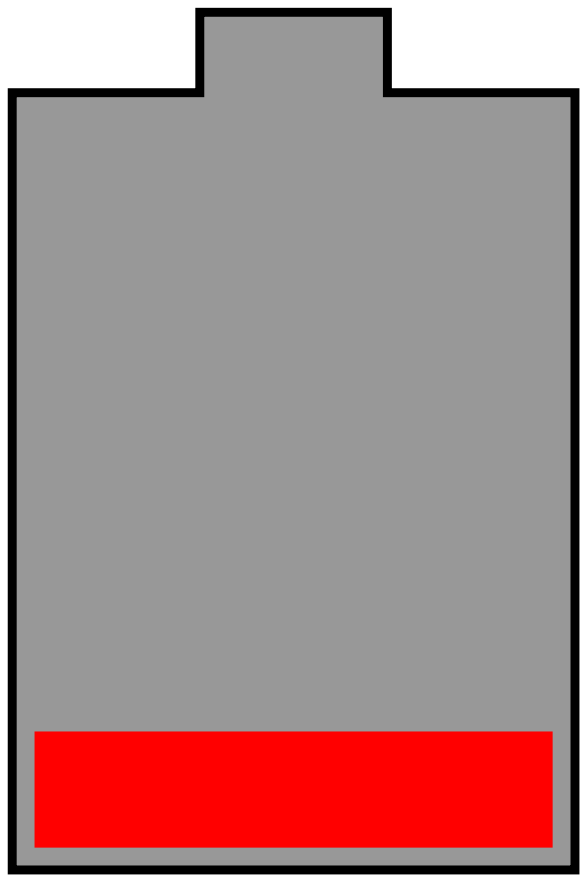
# Energy efficiency, VLIWs and Scheduling

- Mobile era.
- Energy becomes a major design constraint.
- Hardware Instruction scheduling consumes a lot of energy.
- VLIW processors: high-performance and statically scheduled.



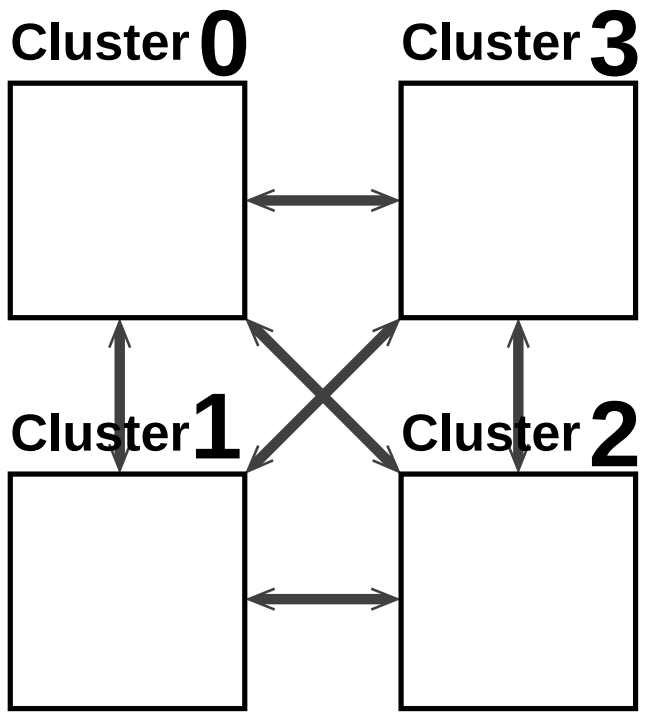
# Energy efficiency, VLIWs and Scheduling

- Mobile era.
- Energy becomes a major design constraint.
- Hardware Instruction scheduling consumes a lot of energy.
- VLIW processors: high-performance and statically scheduled.
- Clustered VLIW = scalable VLIW.



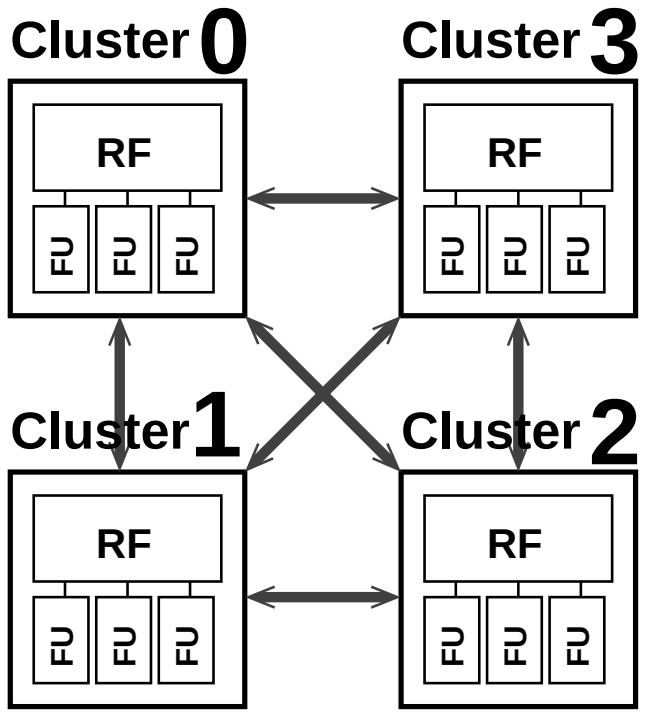
# Clustered VLIW

- Scalable
- Energy efficient



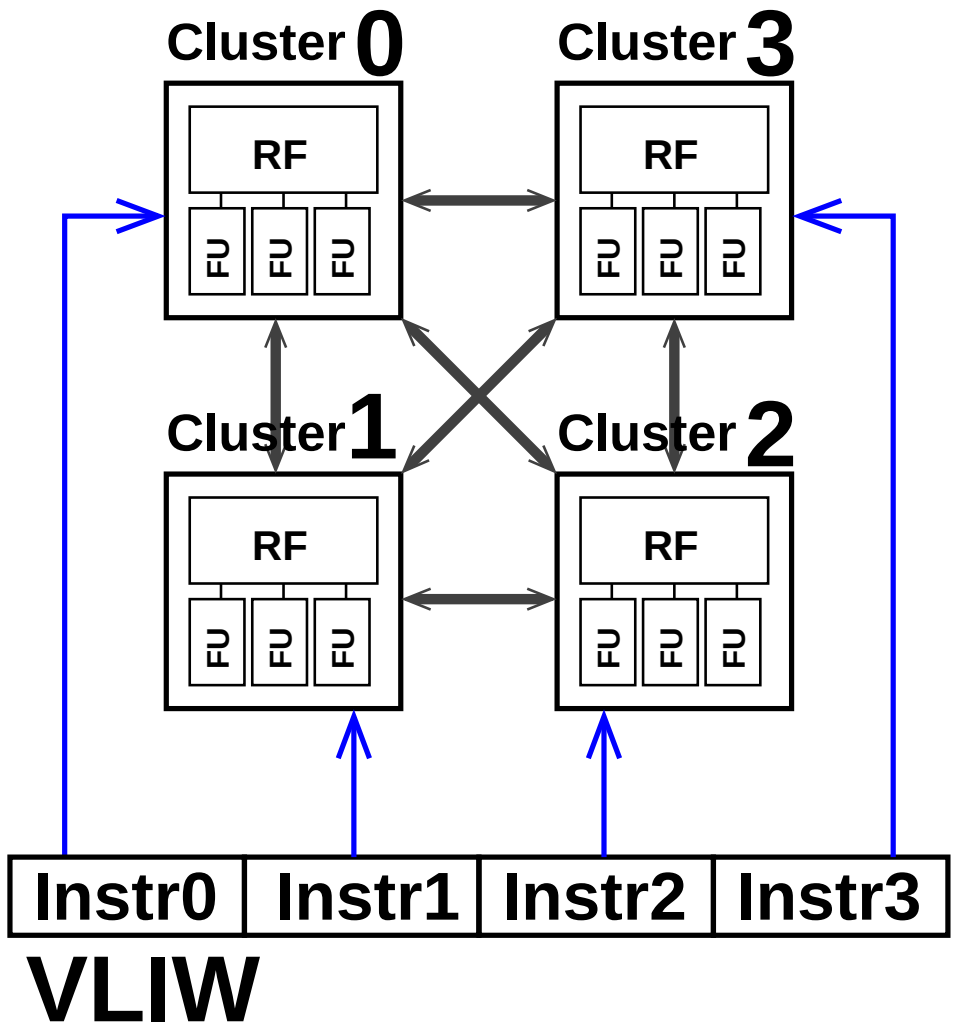
# Clustered VLIW

- Scalable
- Energy efficient
- High frequency
- Inter-Cluster delay



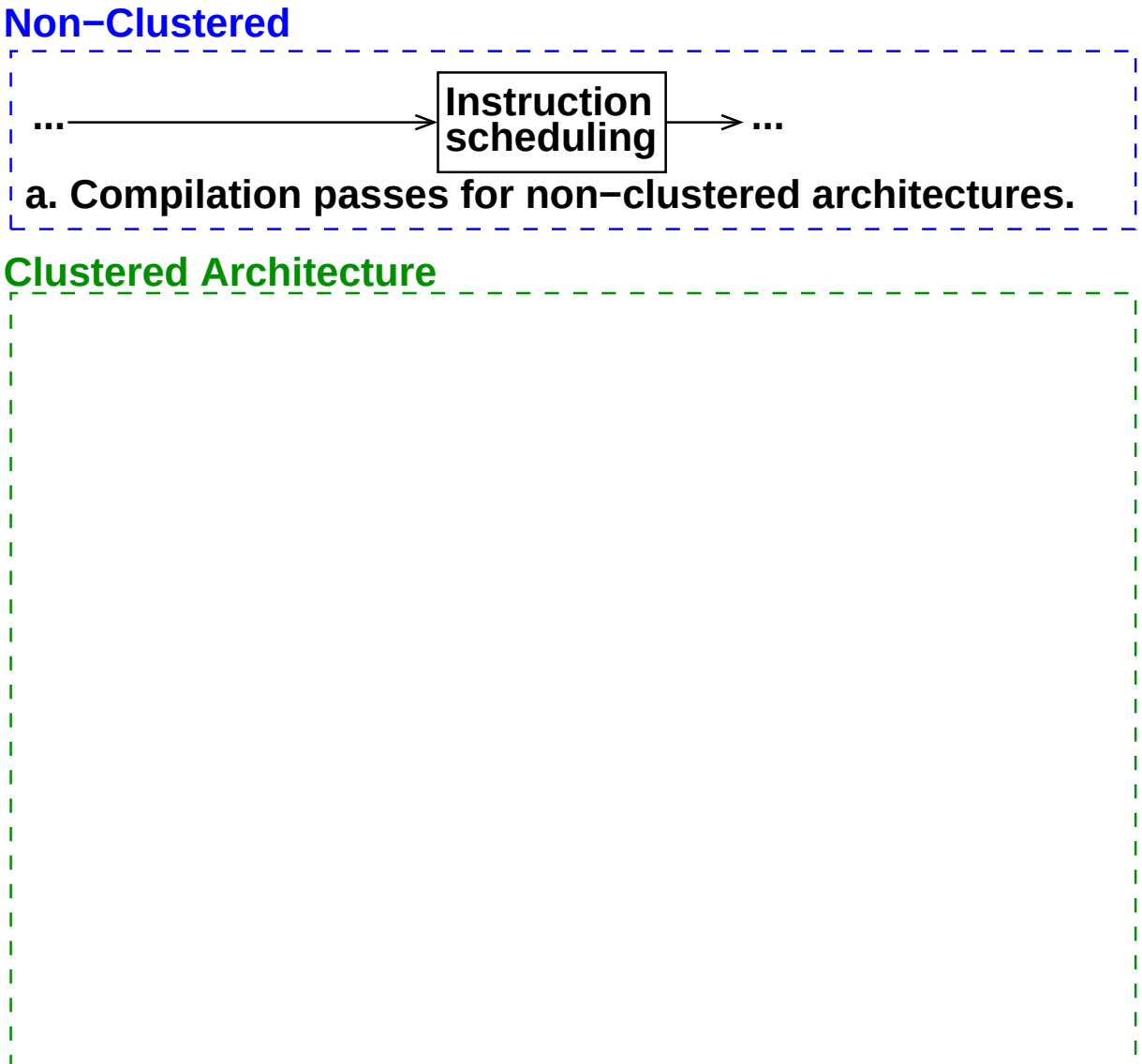
# Clustered VLIW

- Scalable
- Energy efficient
- High frequency
- Inter-Cluster delay
- Relies on compiler
- Statically scheduled
- Explicit ILP



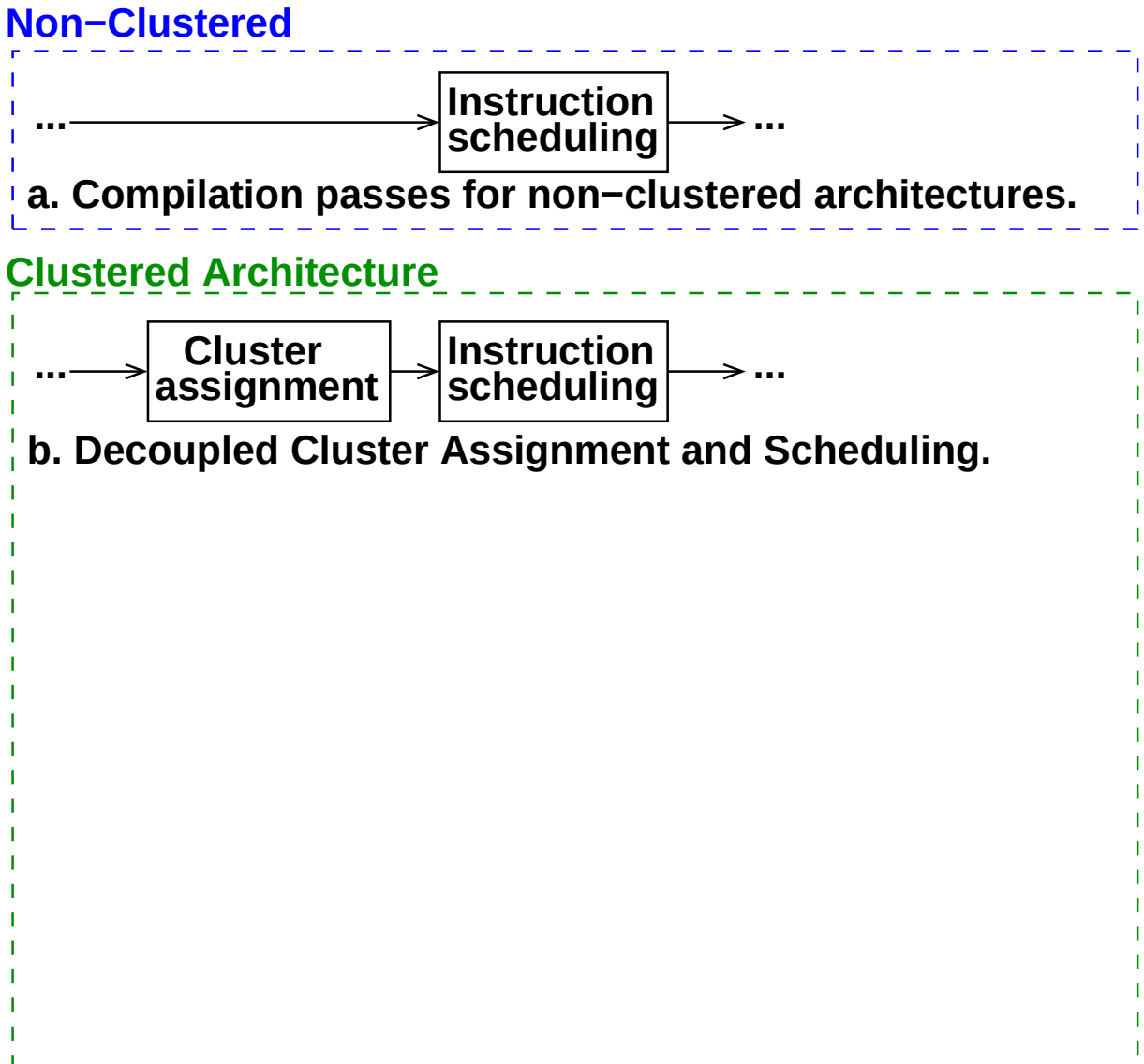
# Scheduling Overview

- Scheduler for monolithic VLIW



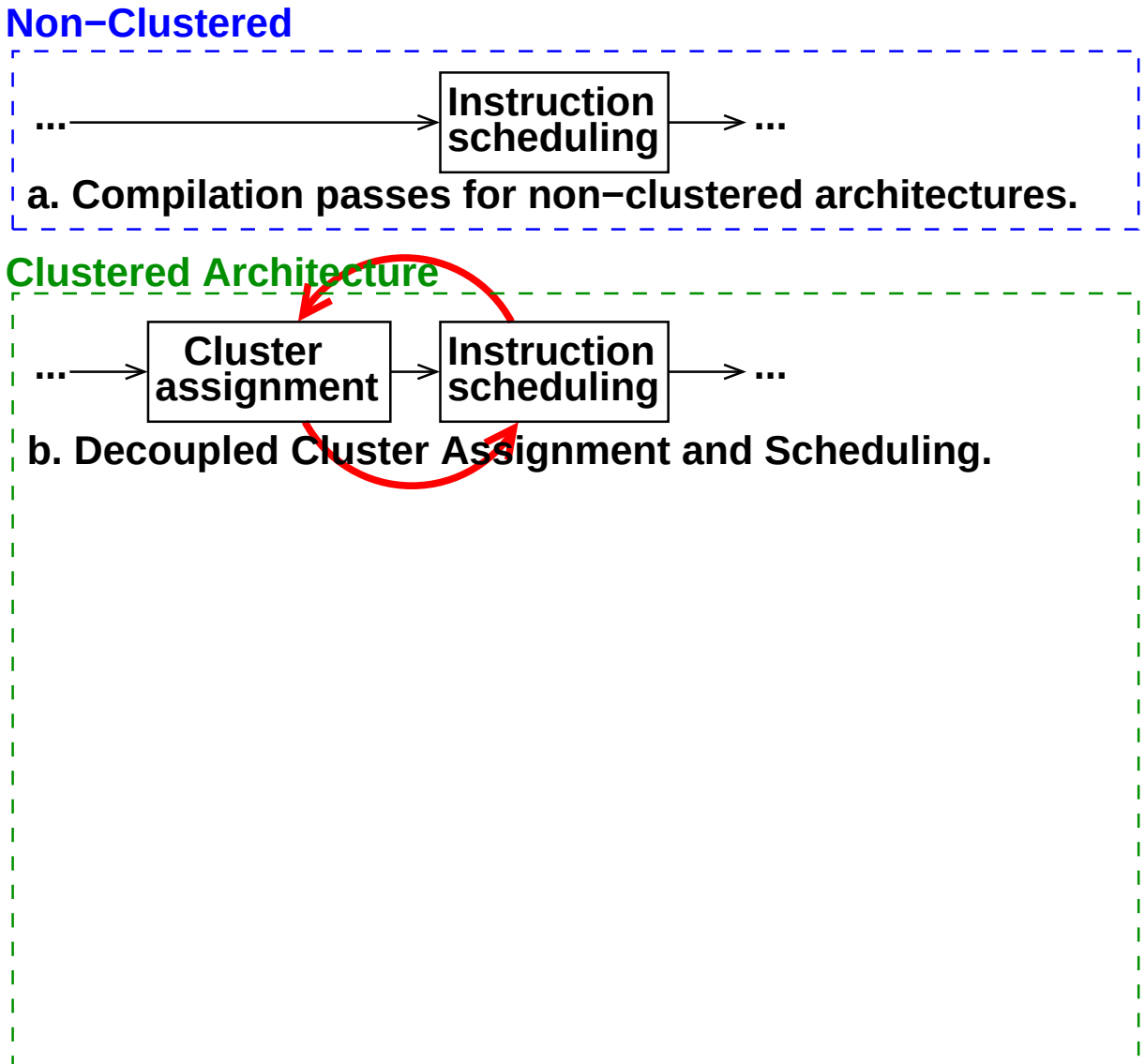
# Scheduling Overview

- Scheduler for monolithic VLIW
- 2-stage scheduling for clusters



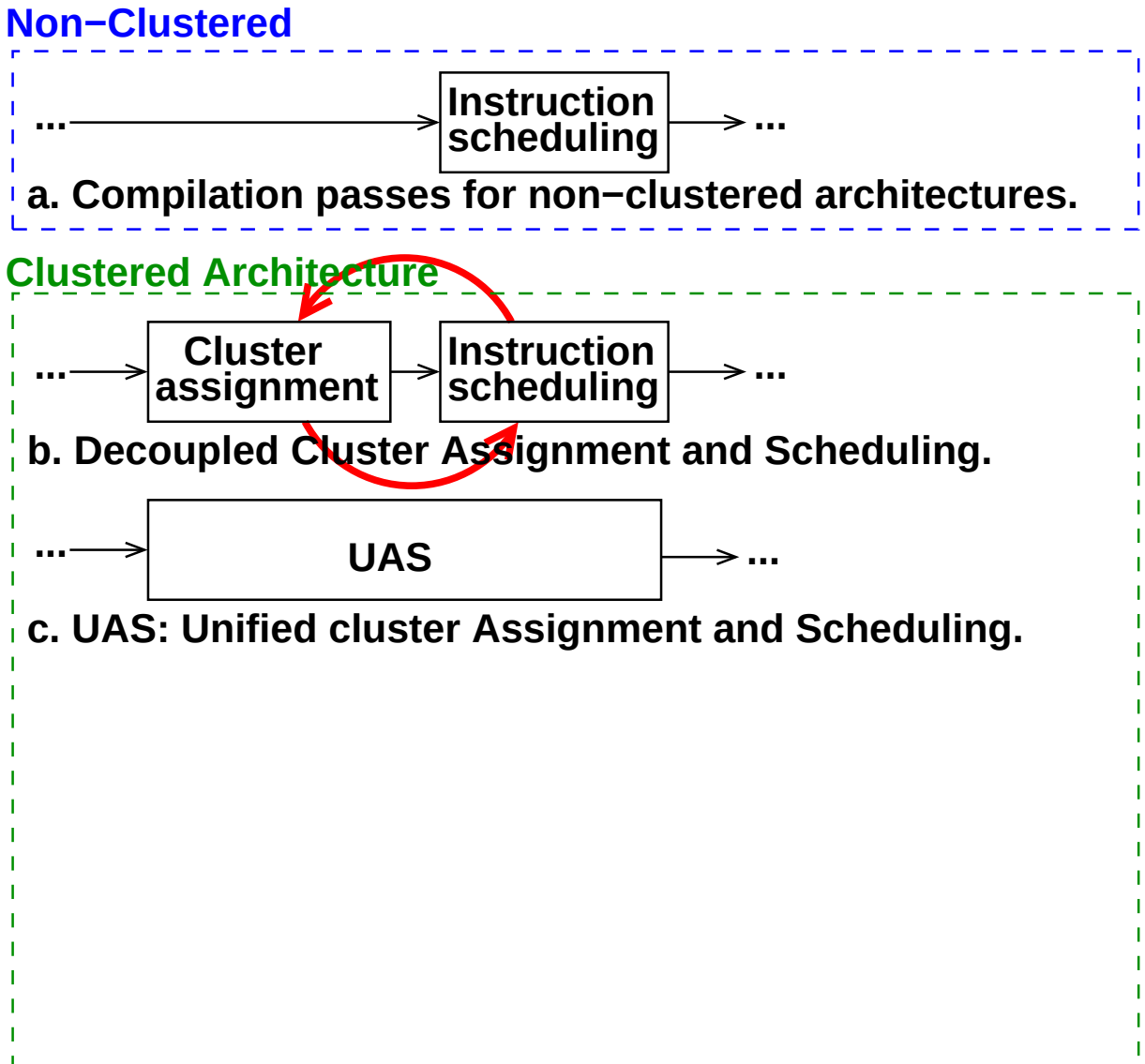
# Scheduling Overview

- Scheduler for monolithic VLIW
- 2-stage scheduling for clusters



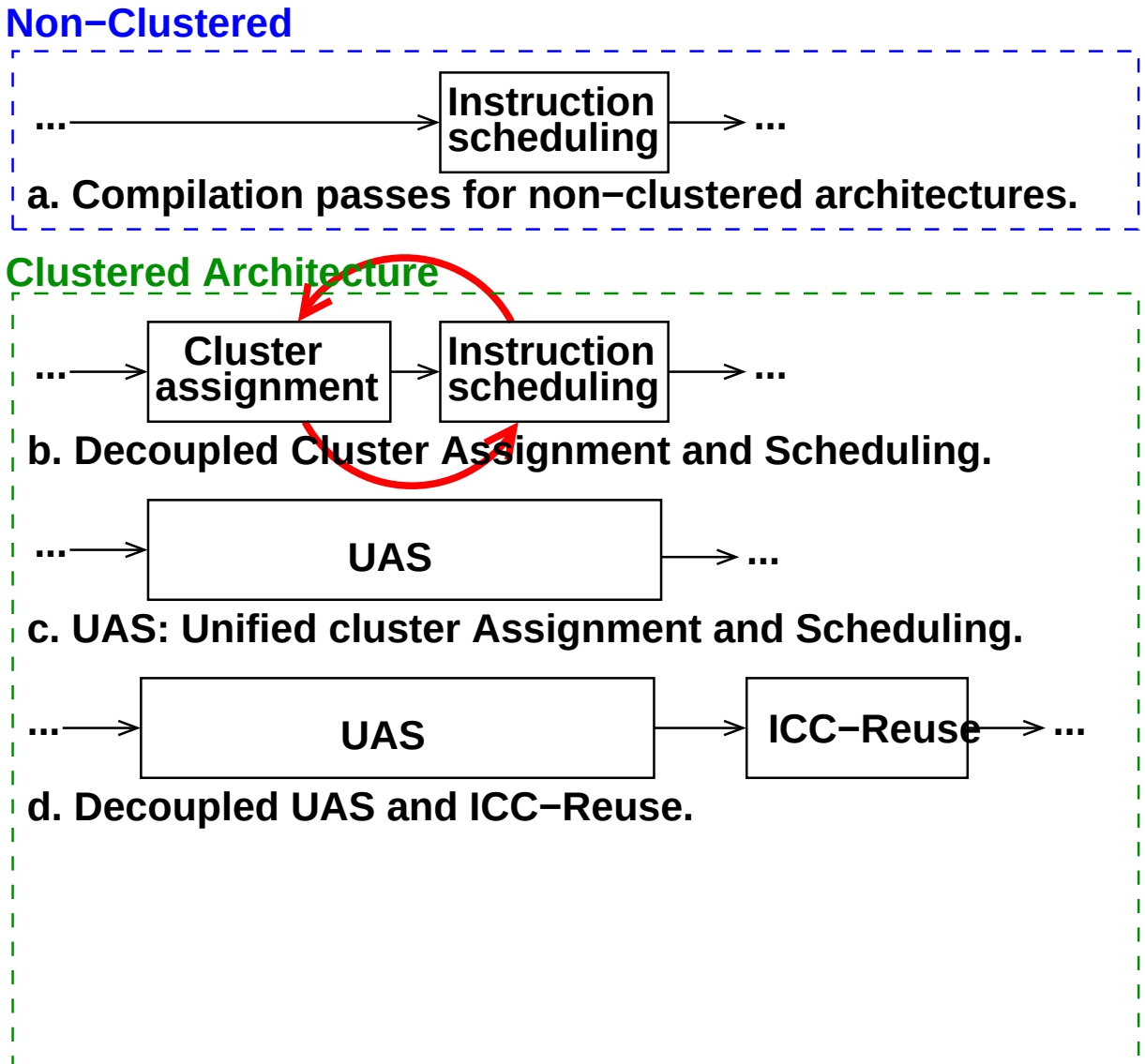
# Scheduling Overview

- Scheduler for monolithic VLIW
- 2-stage scheduling for clusters
- Unified scheduler (State-of-the-art)



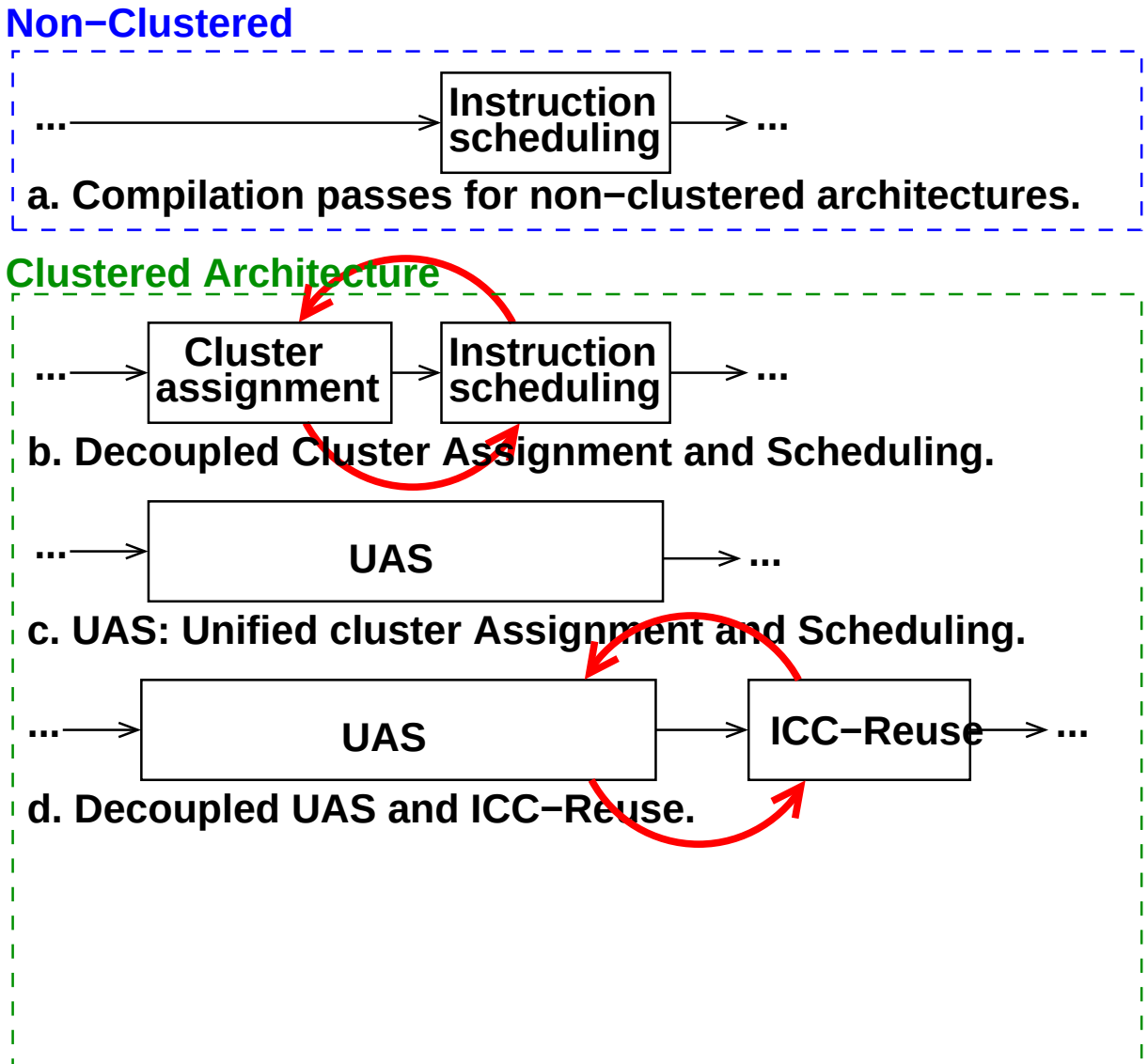
# Scheduling Overview

- Scheduler for monolithic VLIW
- 2-stage scheduling for clusters
- Unified scheduler (State-of-the-art)
- 2-stage ICC-reuse



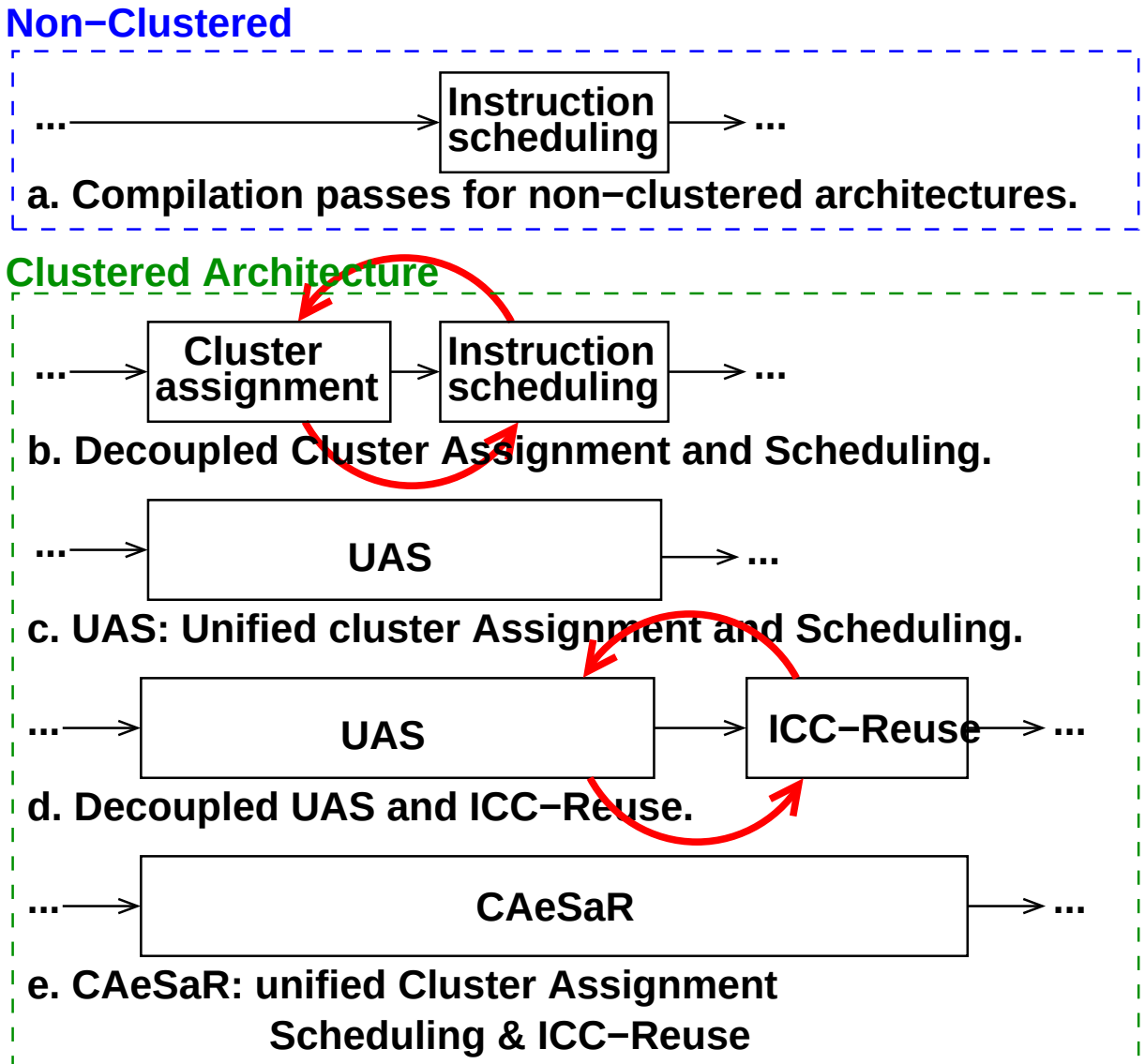
# Scheduling Overview

- Scheduler for monolithic VLIW
- 2-stage scheduling for clusters
- Unified scheduler (State-of-the-art)
- 2-stage ICC-reuse



# Scheduling Overview

- Scheduler for monolithic VLIW
- 2-stage scheduling for clusters
- Unified scheduler (State-of-the-art)
- 2-stage ICC-reuse
- Unified ICC-reuse



# Outline

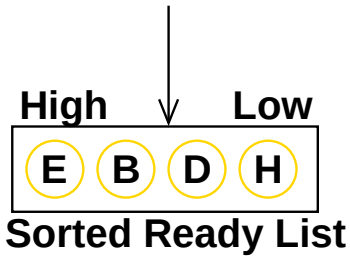
Introduction

CAeSaR Scheduling

Experimental Setup and Results

Conclusion

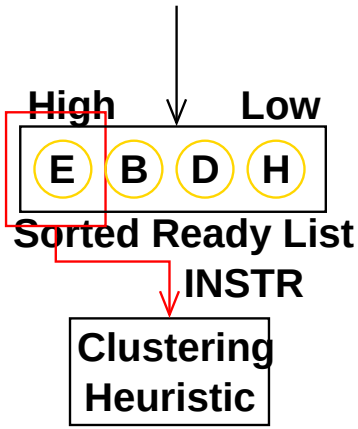
# CAeSaR Scheduler




- State-of-the-art (UAS)

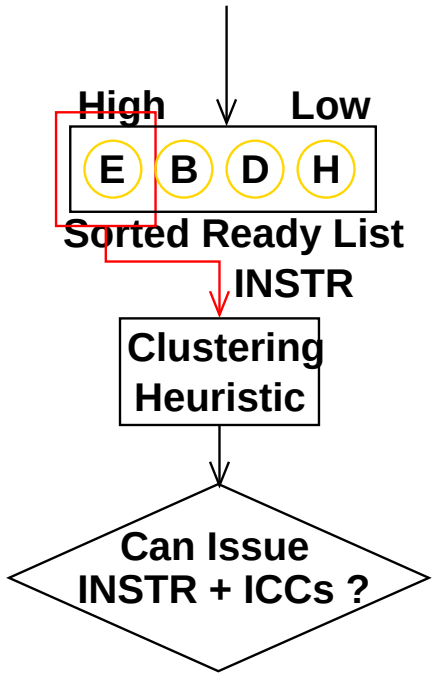
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR



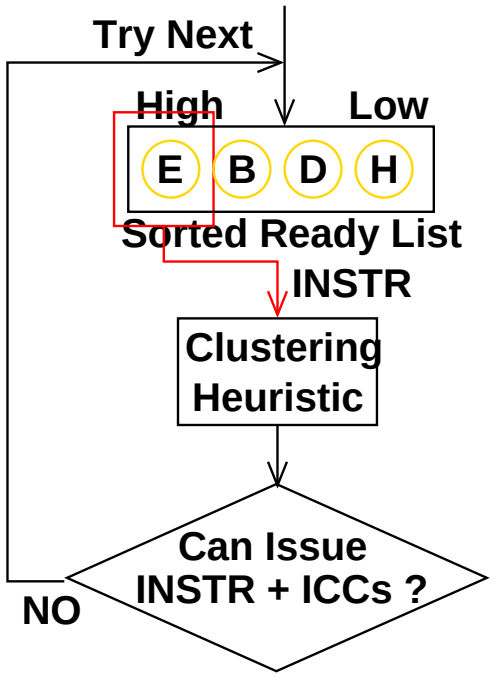
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR



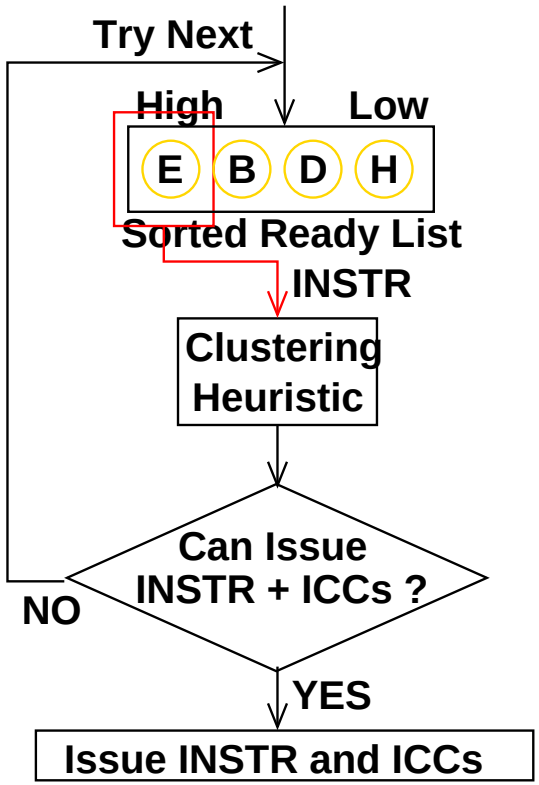
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR



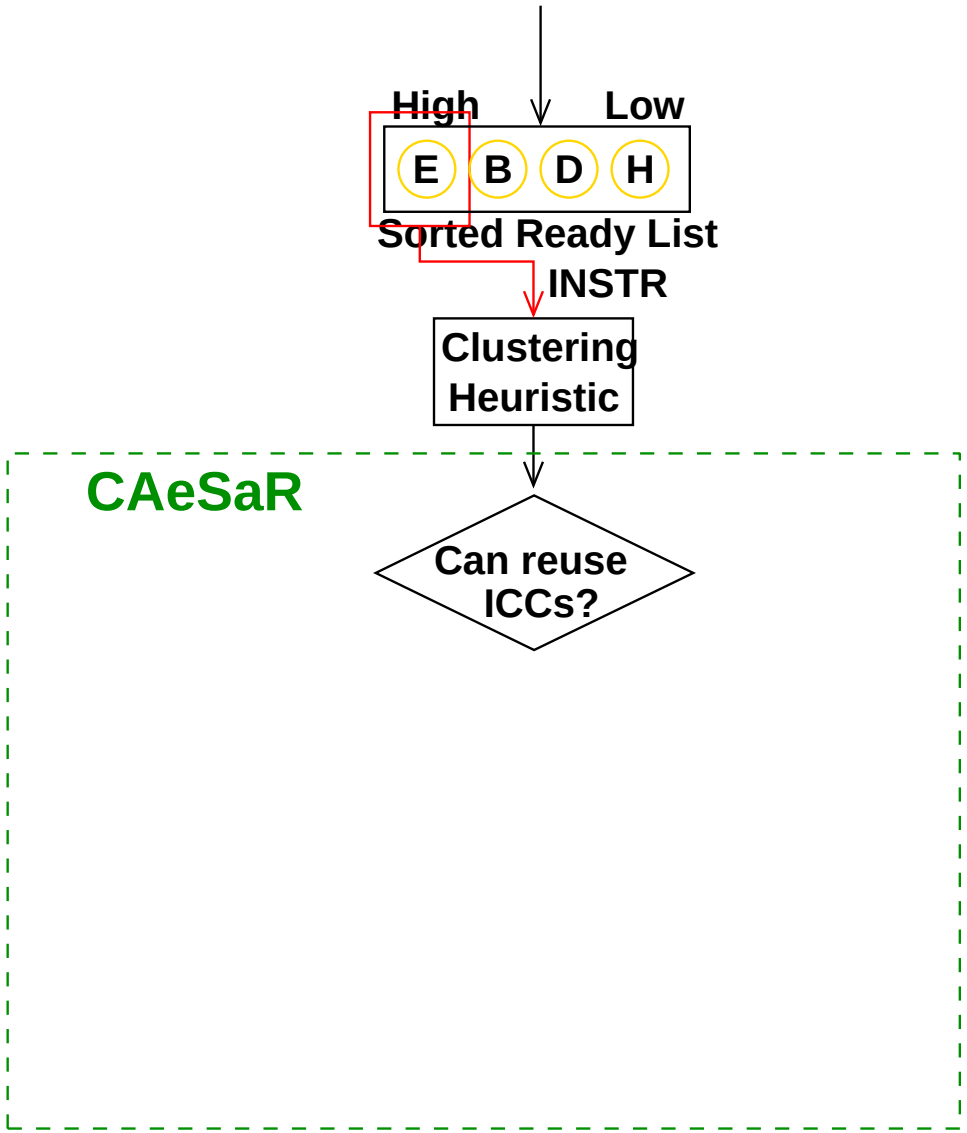
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE



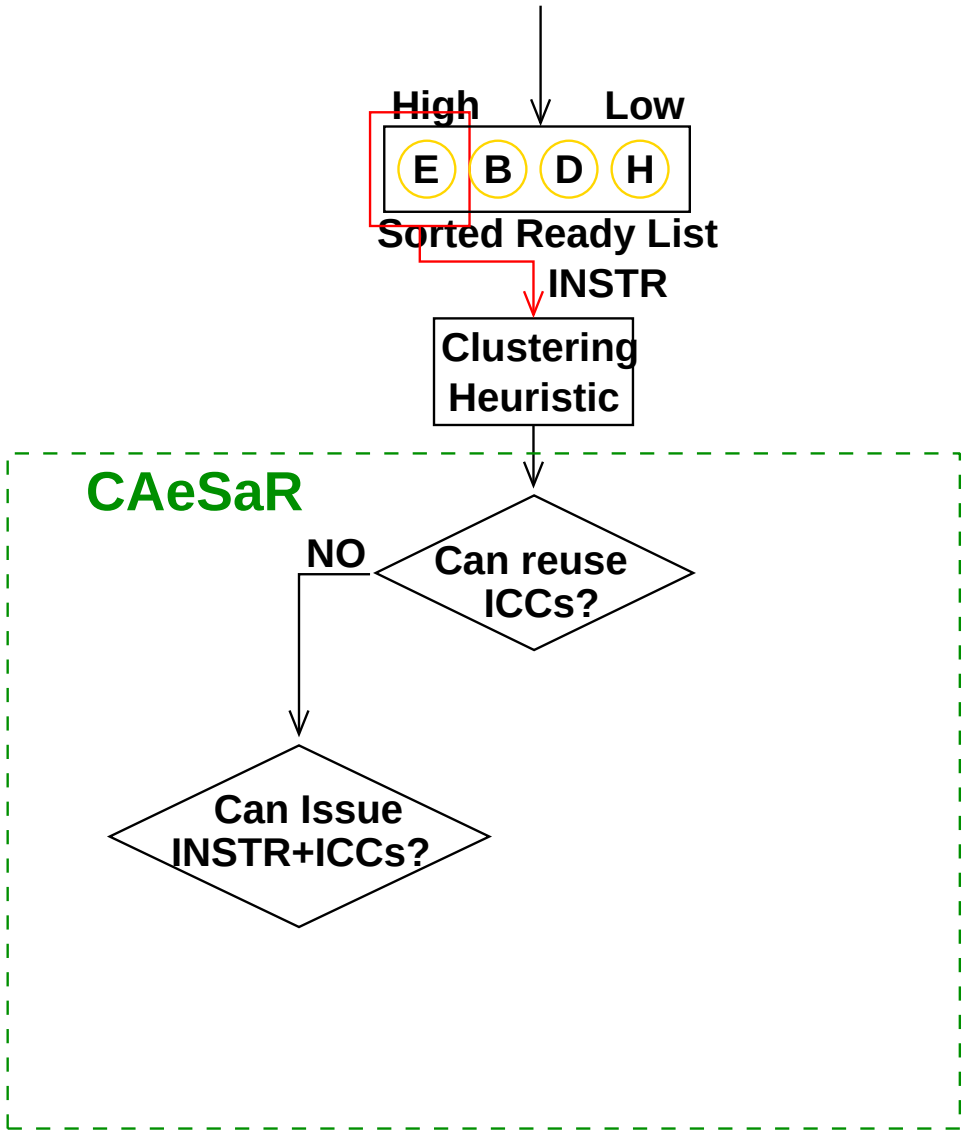
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR



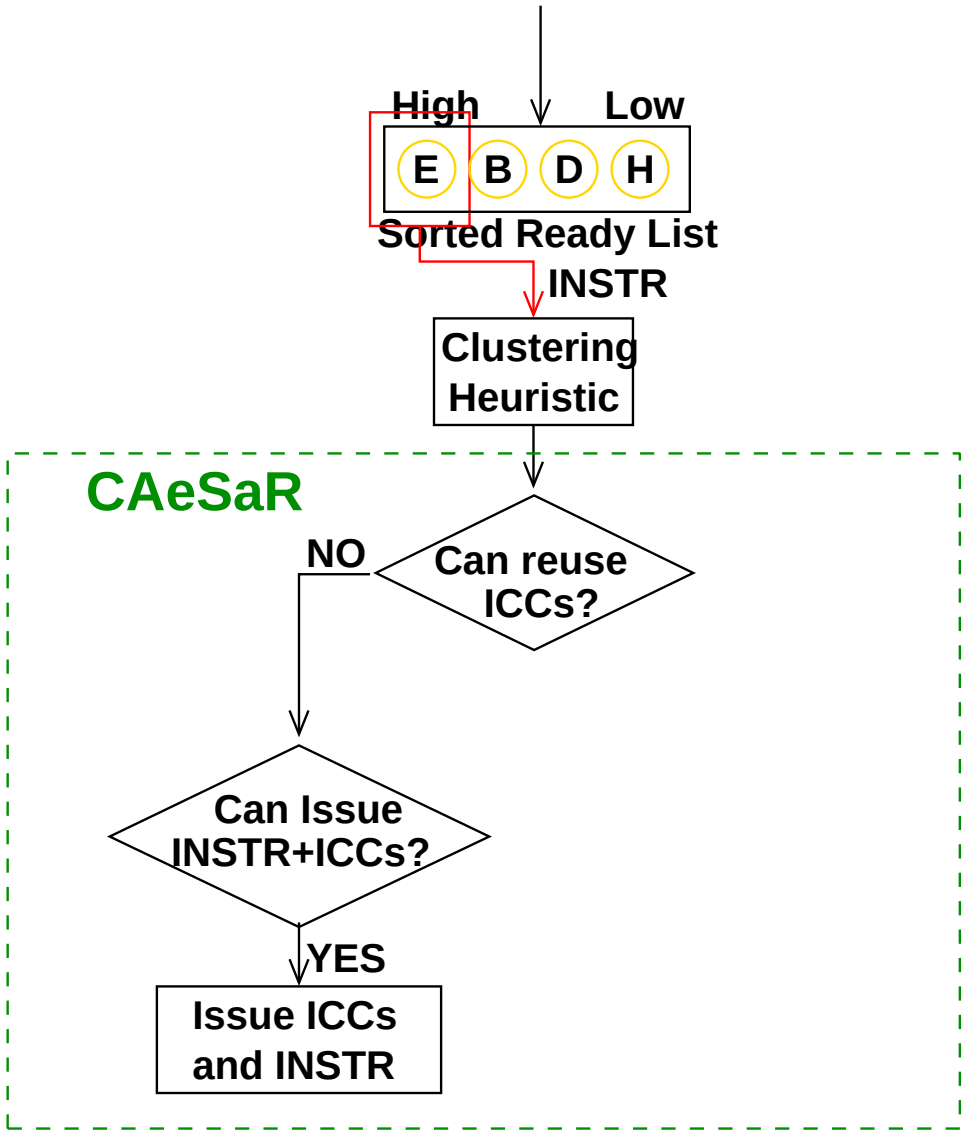
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR



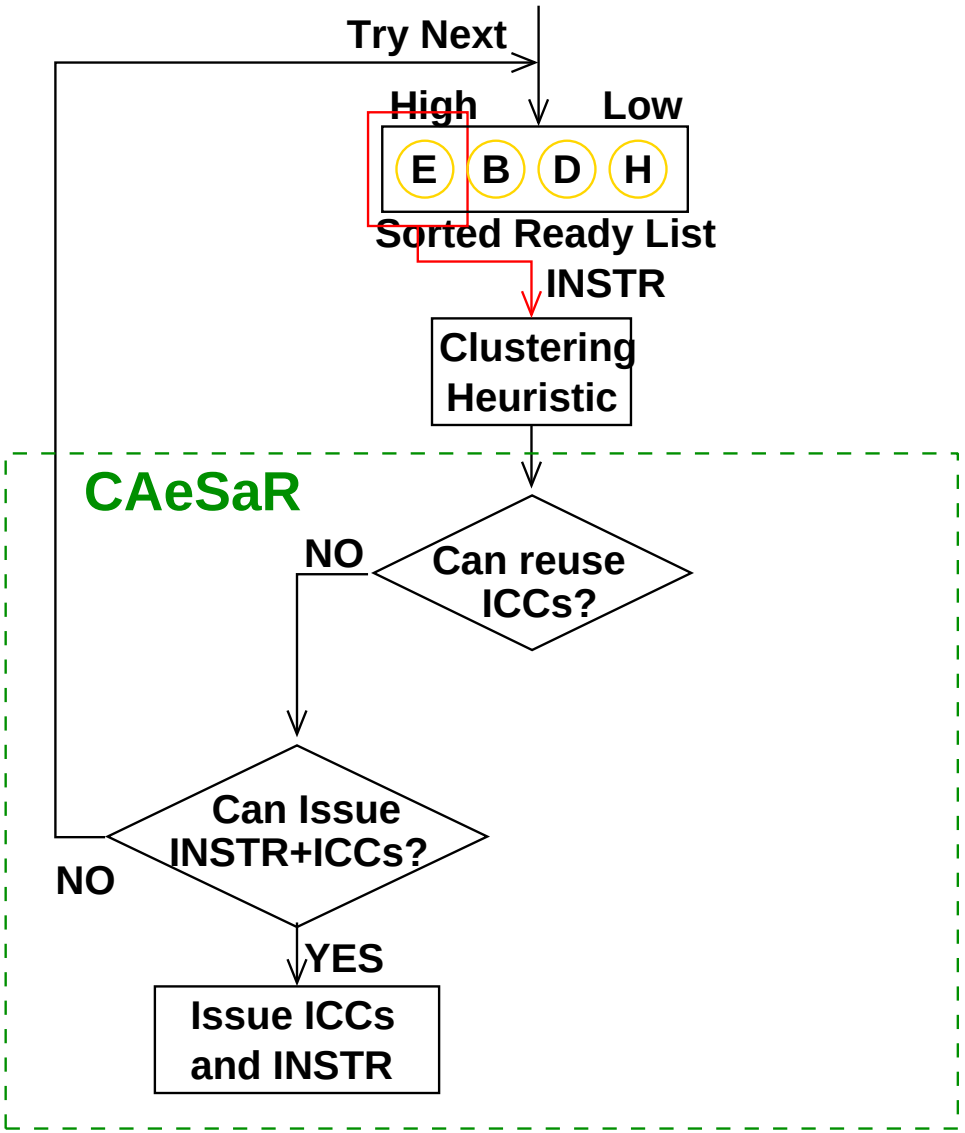
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR



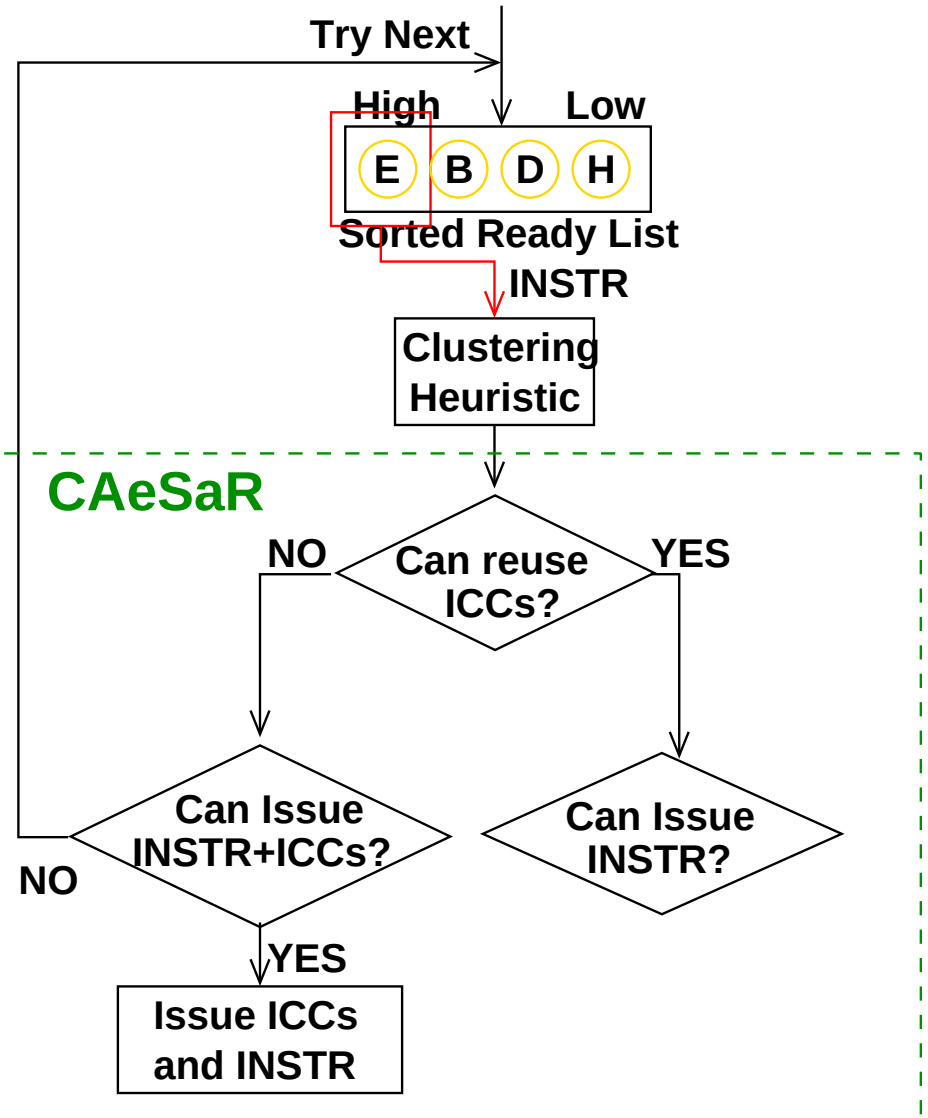
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR



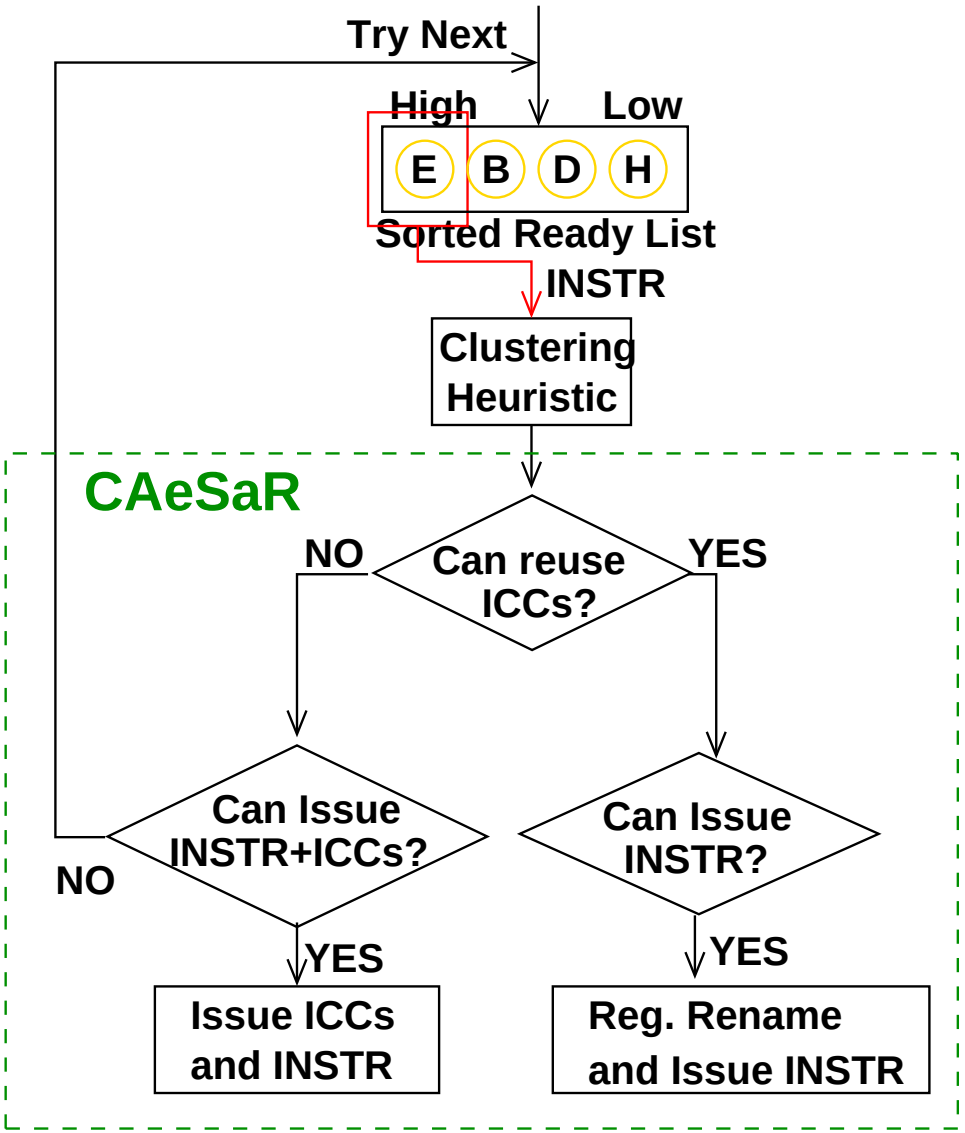
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR
- ICC-reuse



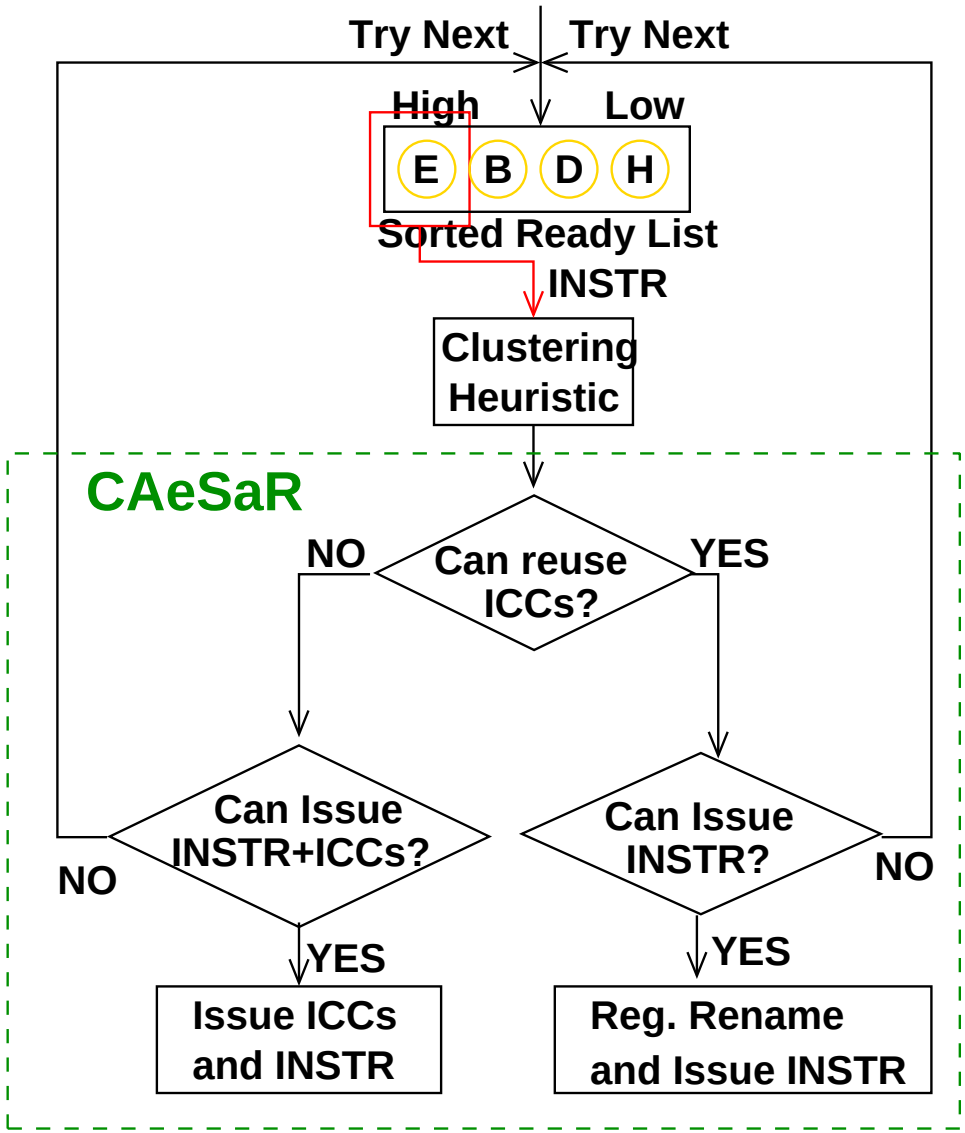
# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR
- ICC-reuse



# CAeSaR Scheduler

- State-of-the-art (UAS)
- High Priority INSTR
- If can issue, DONE
- CAeSaR
- ICC-reuse



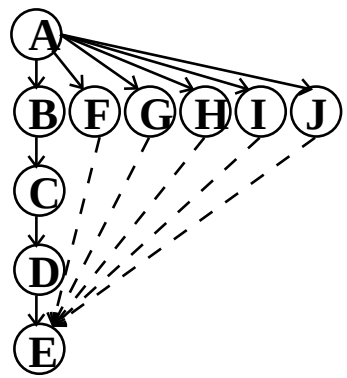
# Example

**Monolithic**

| | | |
|---|---|---|
| 0 | A | |
| 1 | B | F |
| 2 | C | G |
| 3 | D | H |
| 4 | I | J |
| 5 | E | |

**a. VLIW**

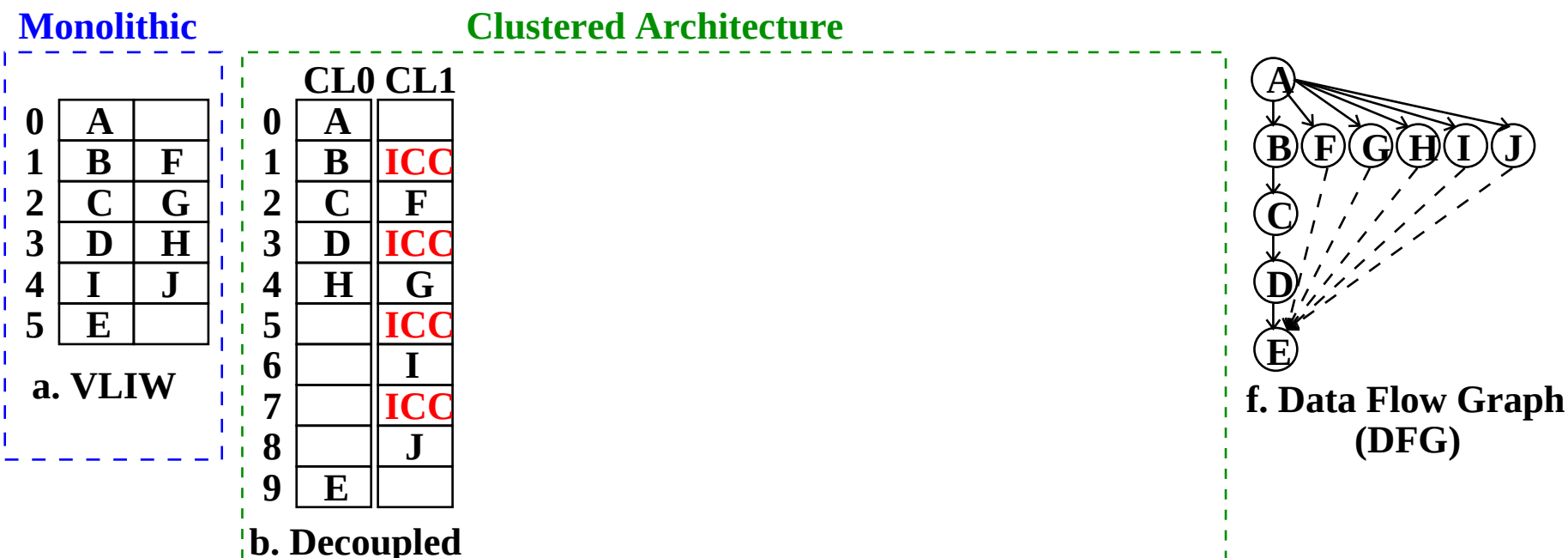
**Clustered Architecture**




**f. Data Flow Graph (DFG)**

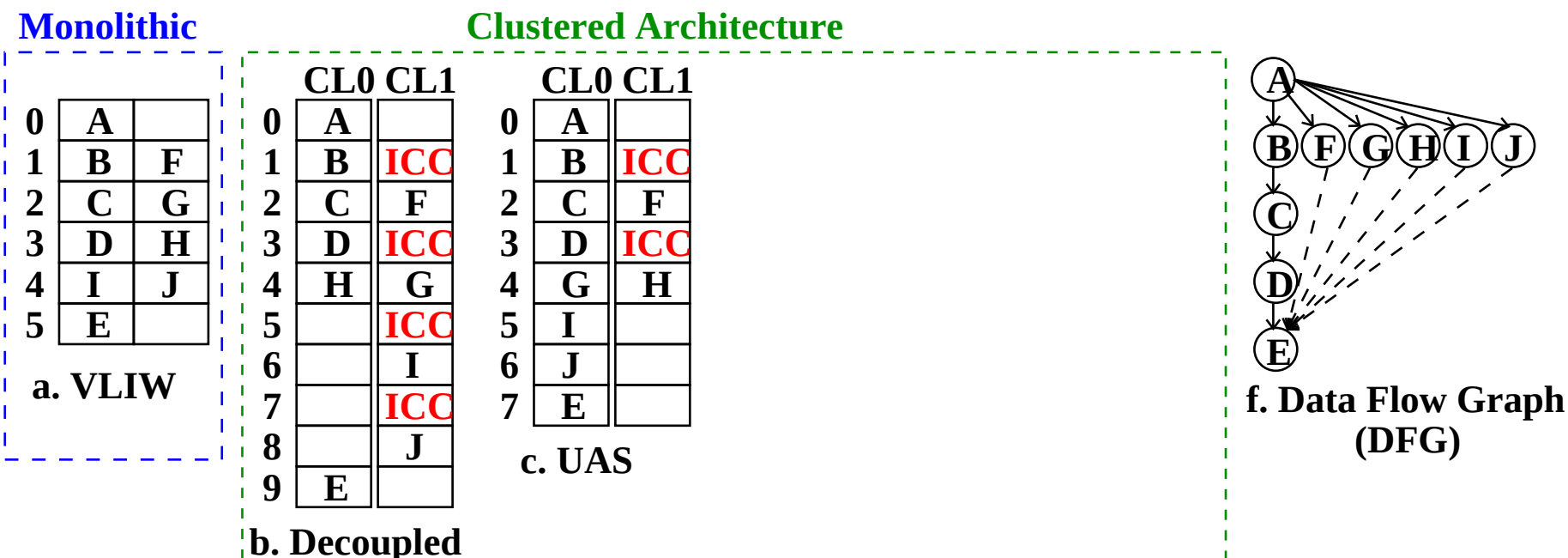
# Example

- 2-step scheduler generates bad schedule



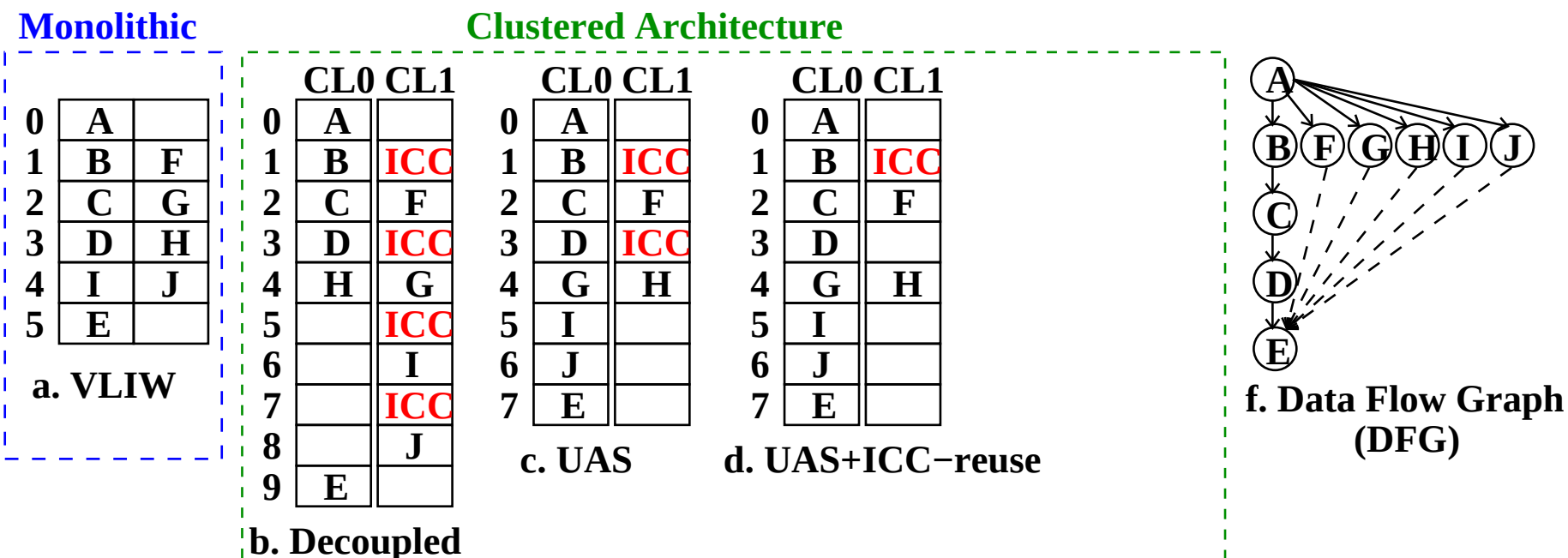
# Example

- 2-step scheduler generates bad schedule
- UAS improves it by being ICC-aware



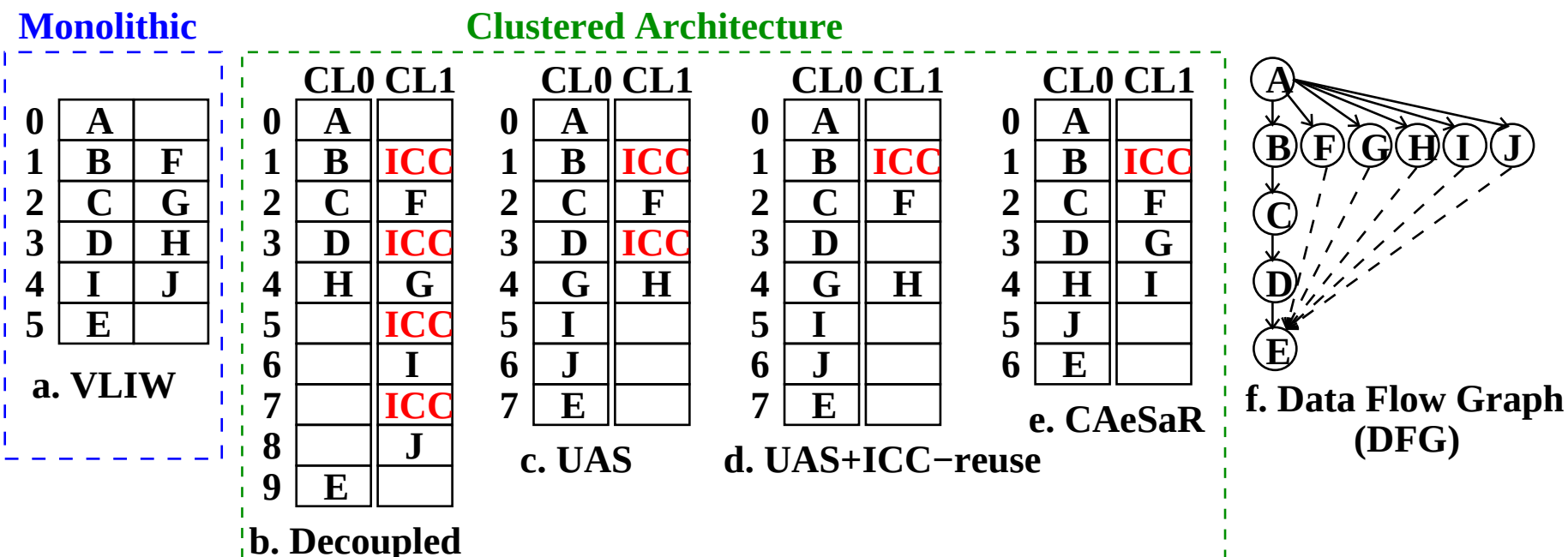
# Example

- 2-step scheduler generates bad schedule
- UAS improves it by being ICC-aware
- UAS with ICC-reuse has fewer ICCs



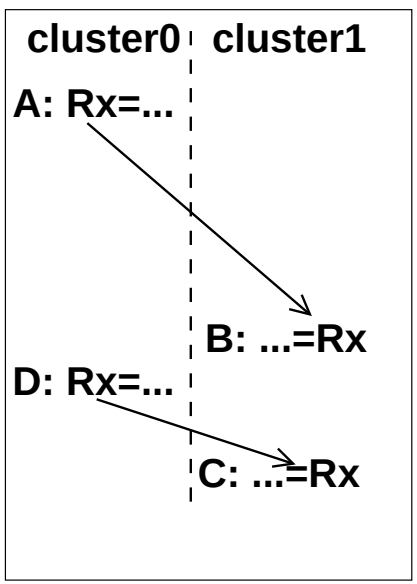
# Example

- 2-step scheduler generates bad schedule
- UAS improves it by being ICC-aware
- UAS with ICC-reuse has fewer ICCs
- CAeSaR solves phase-ordering



# Register File Coherence

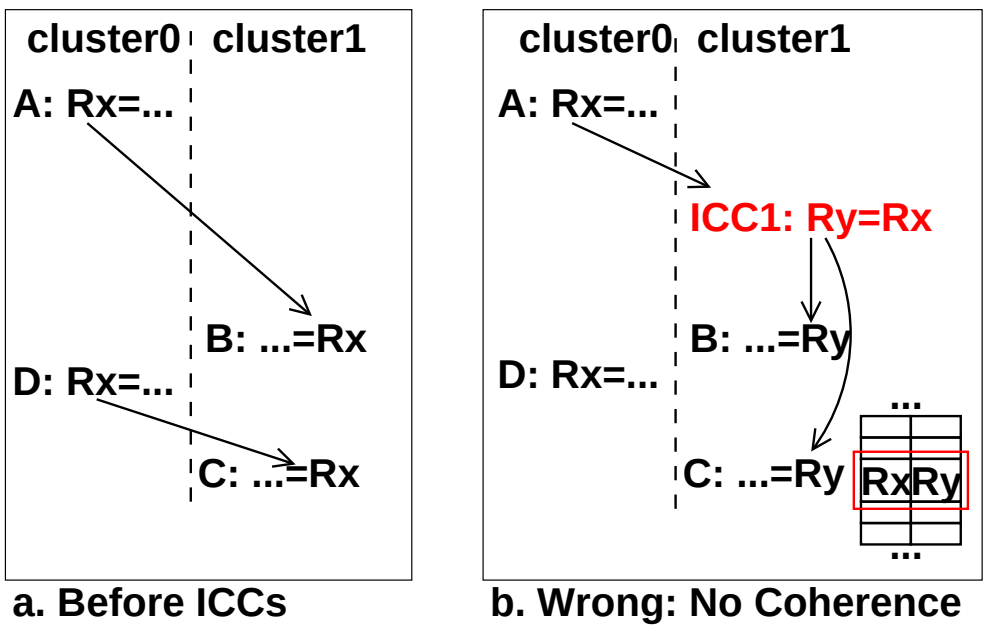
- Reused value must be correct




a. Before ICCs

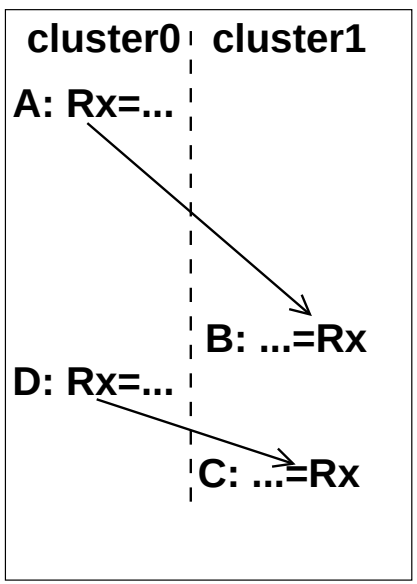
# Register File Coherence

- Reused value must be correct

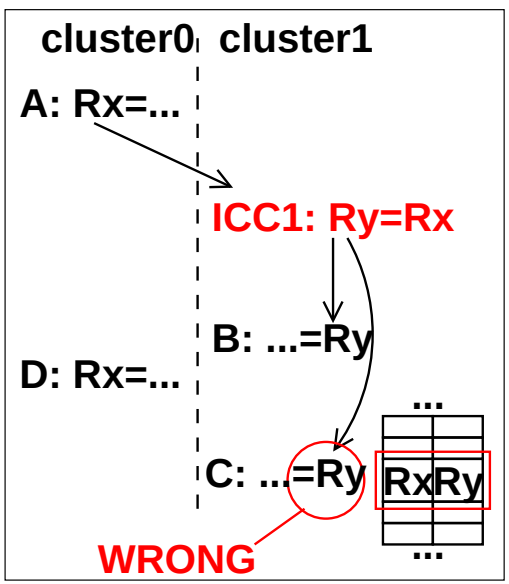


# Register File Coherence

- Reused value must be correct



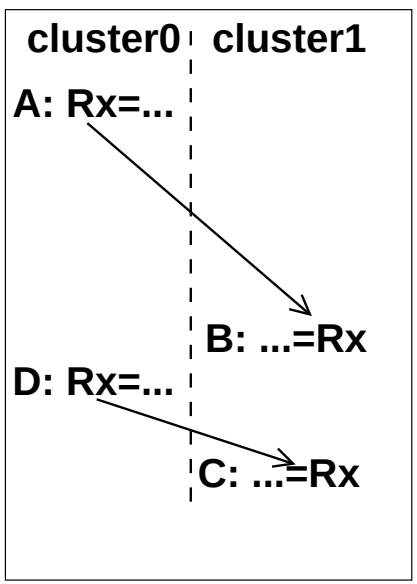

a. Before ICCs



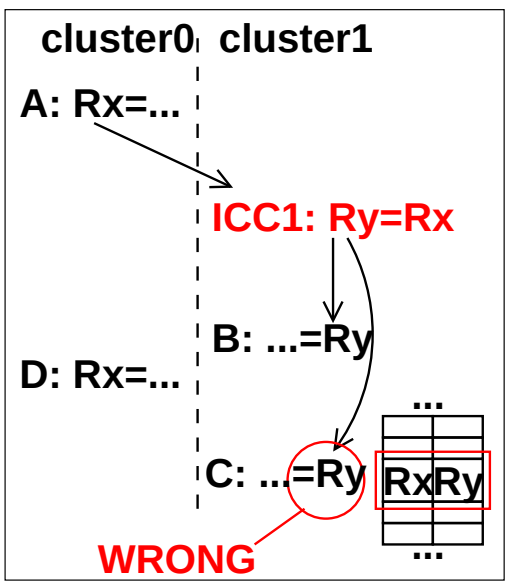

b. Wrong: No Coherence

# Register File Coherence

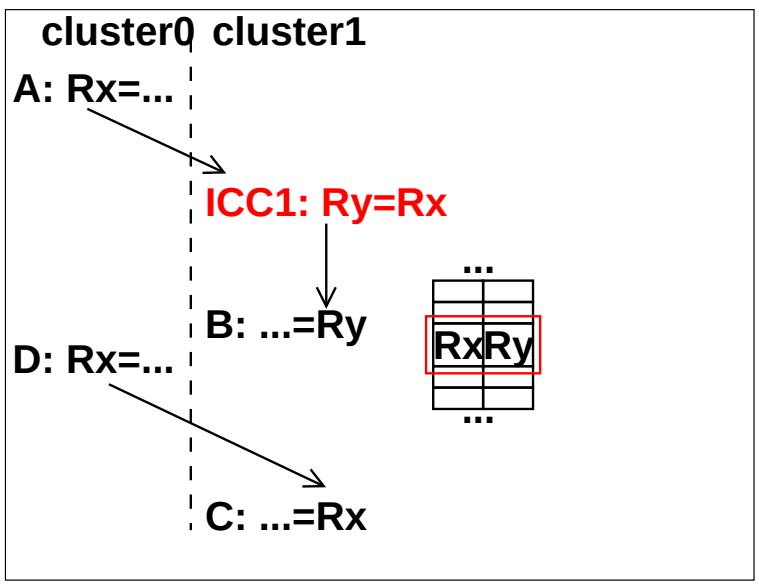
- Reused value must be correct
- CAeSaR follows an approach similar to write-invalidate cache coherent protocols




a. Before ICCs



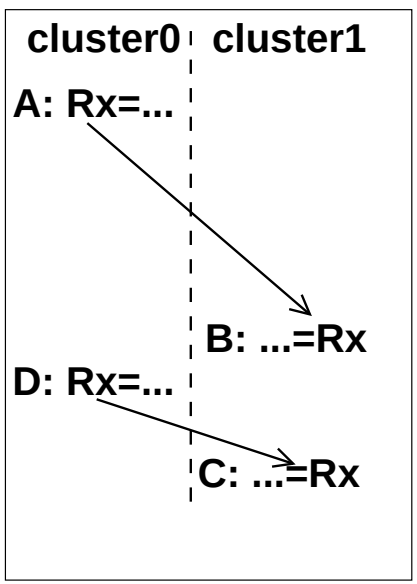

b. Wrong: No Coherence



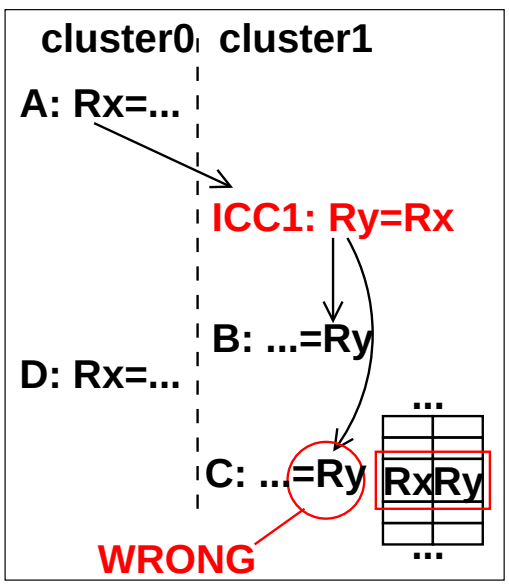

c. CAeSaR Register Coherence

# Register File Coherence

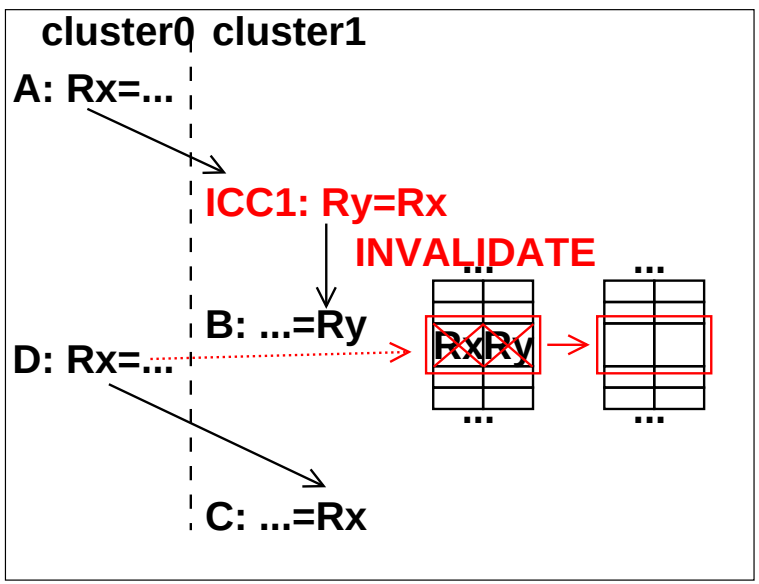
- Reused value must be correct
- CAeSaR follows an approach similar to write-invalidate cache coherent protocols




a. Before ICCs



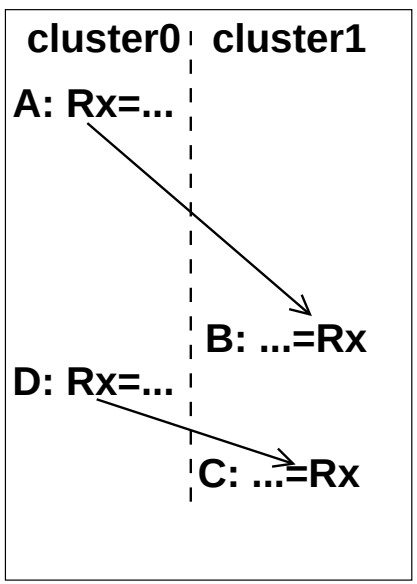

b. Wrong: No Coherence



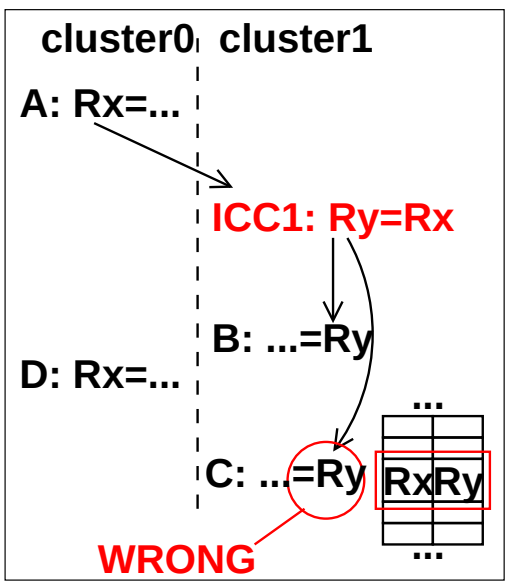

c. CAeSaR Register Coherence

# Register File Coherence

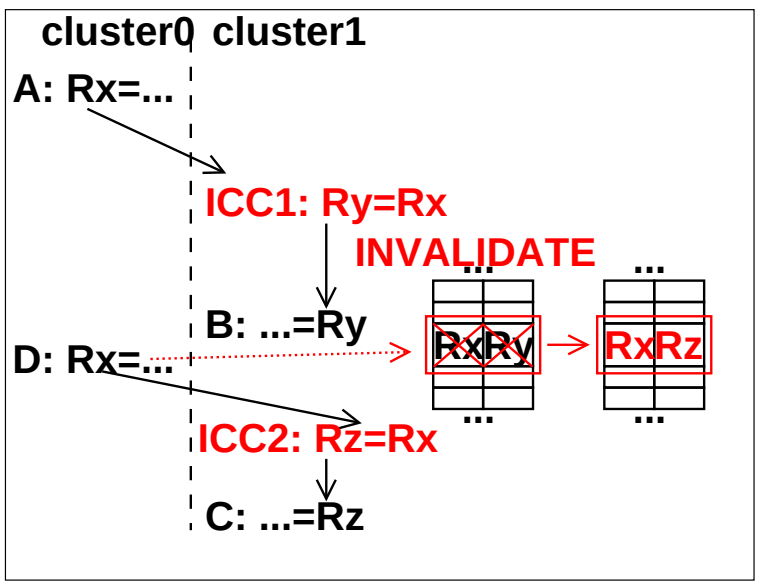
- Reused value must be correct
- CAeSaR follows an approach similar to write-invalidate cache coherent protocols




a. Before ICCs




b. Wrong: No Coherence




c. CAeSaR Register Coherence

# Outline

Introduction

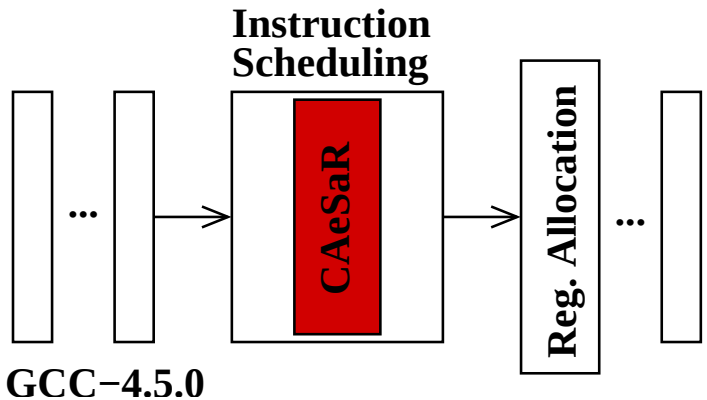
CAeSaR Scheduling

Experimental Setup and Results

Conclusion

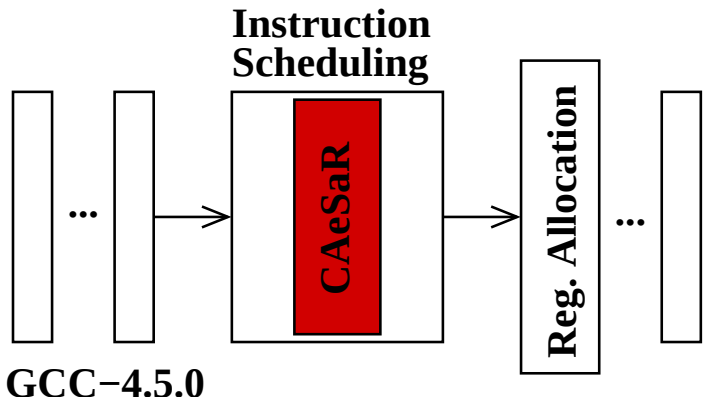
# Experimental Setup

- Compiler: GCC-4.5.0, Modified Haifa-Scheduler



# Experimental Setup

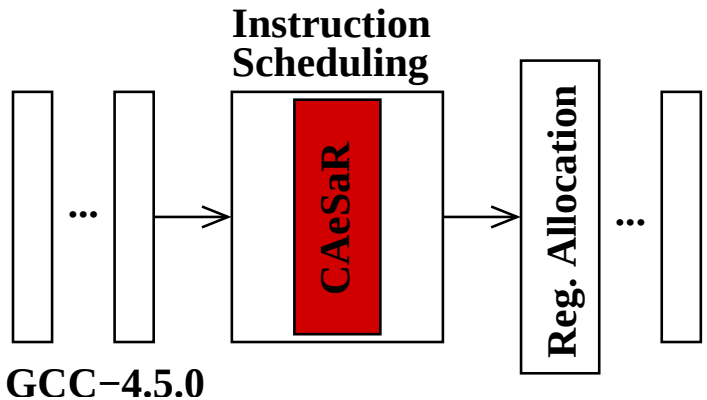
- Compiler: GCC-4.5.0, Modified Haifa-Scheduler




- Architecture
  - IA64-based 4 issue clustered VLIW
  - 2,4 clusters
  - 1 cycle Inter-Cluster Latency

# Experimental Setup

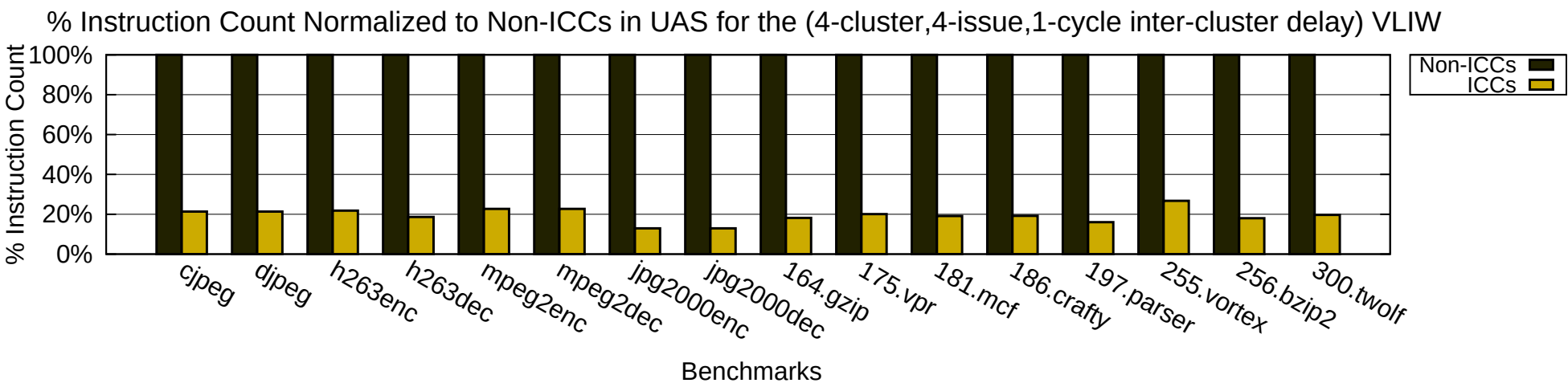
- Compiler: GCC-4.5.0, Modified Haifa-Scheduler




- Architecture
  - IA64-based 4 issue clustered VLIW
  - 2,4 clusters
  - 1 cycle Inter-Cluster Latency
- Benchmarks: SPEC CINT2000 and MediabenchII Video

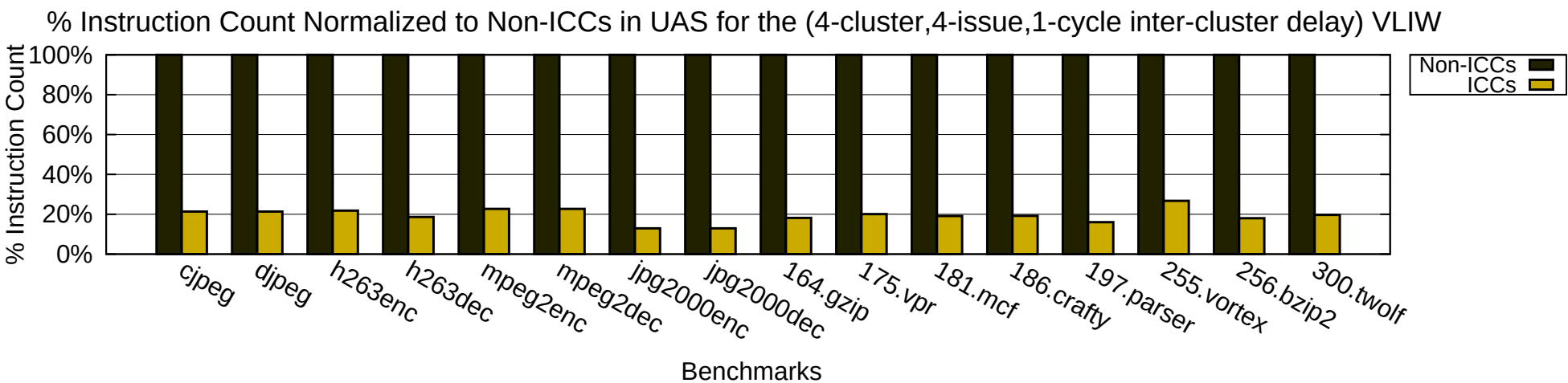
# Results: 4-cluster 4-issue, ICC ratio

- ICCs are a significant bottleneck



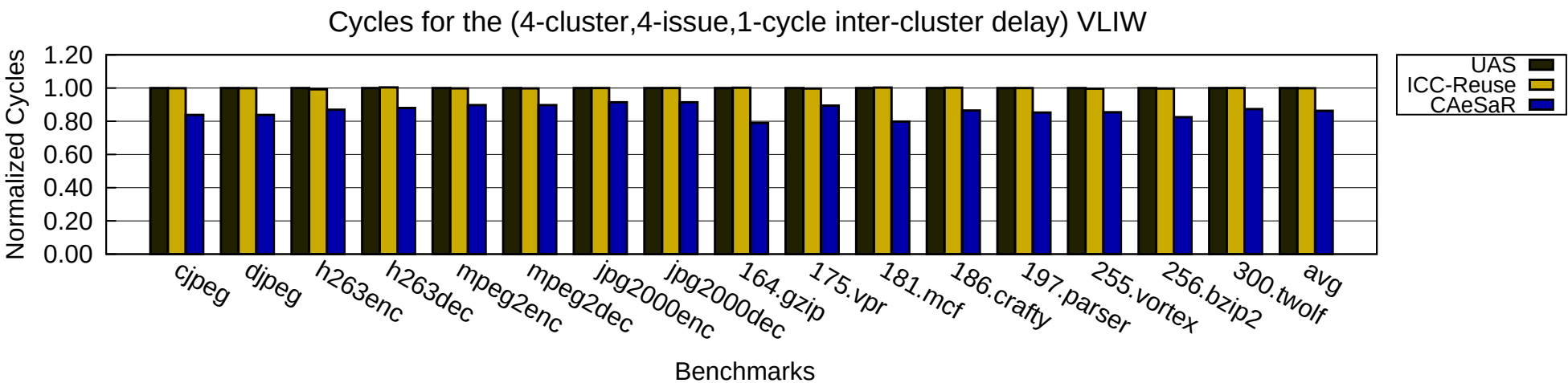
# Results: 4-cluster 4-issue, ICC ratio

- ICCs are a significant bottleneck
- 1 ICC in 5 non-ICCs



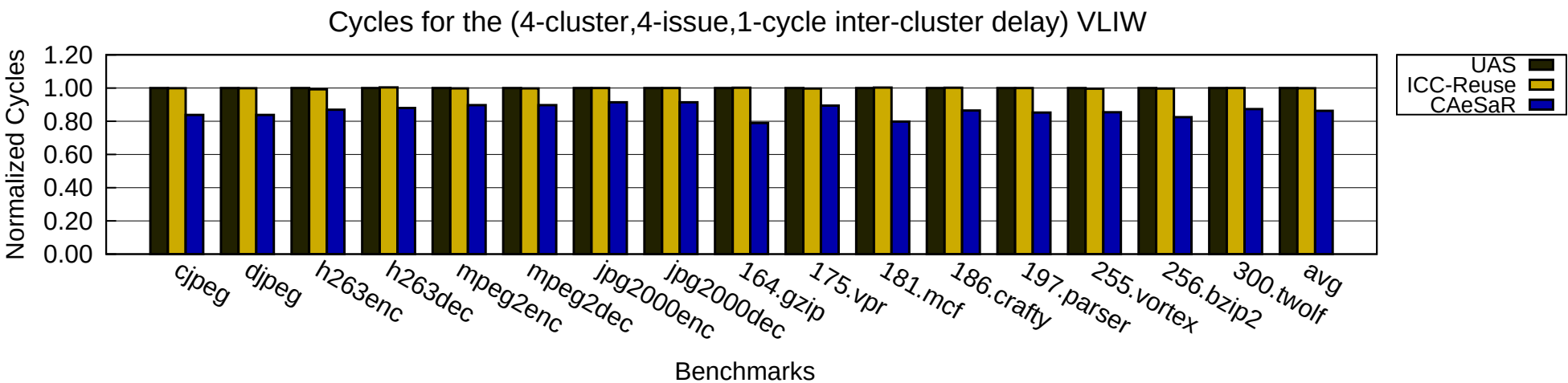
# Results: 4-cluster 4-issue, Sched cycles

- CAeSaR generates 13.8% more compact schedules on average



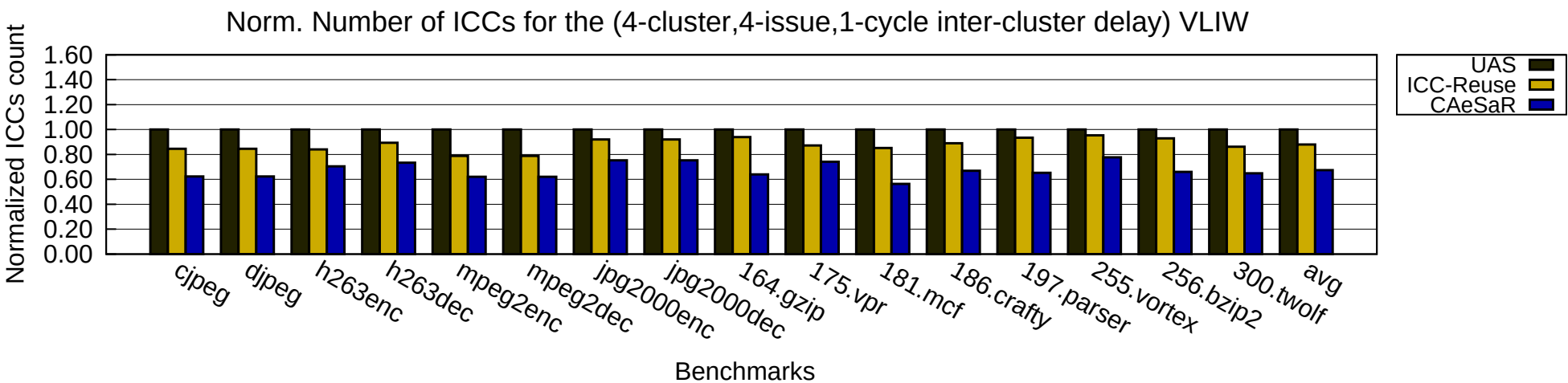
# Results: 4-cluster 4-issue, Sched cycles

- CAeSaR generates 13.8% more compact schedules on average
- ICC-reuse same performance as UAS (expected)



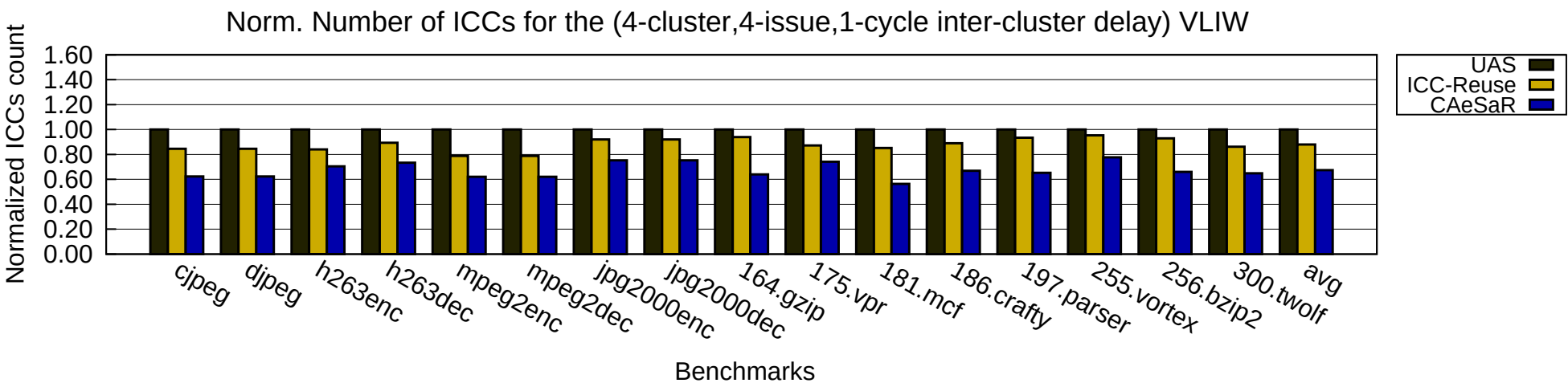
# Results: 4-cluster 4-issue, ICCs

- CAeSaR reduces the count of ICCs



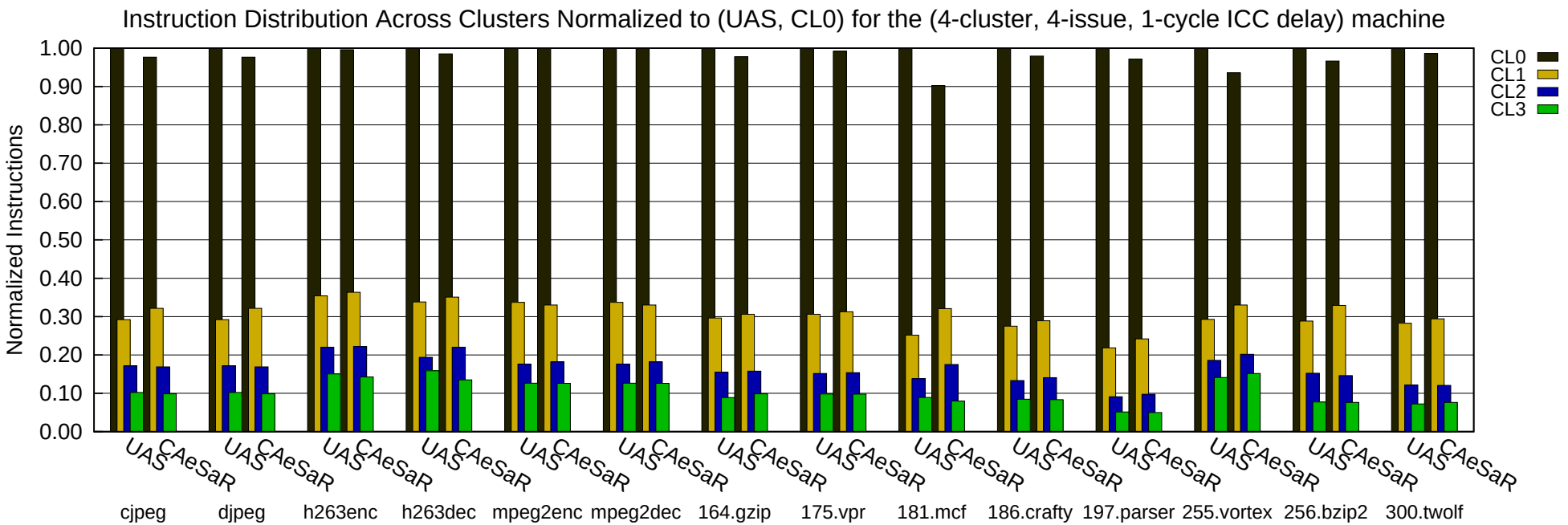
# Results: 4-cluster 4-issue, ICCs

- CAeSaR reduces the count of ICCs
- CAeSaR generates fewer ICCs than ICC-reuse (proof of phase-ordering problem)



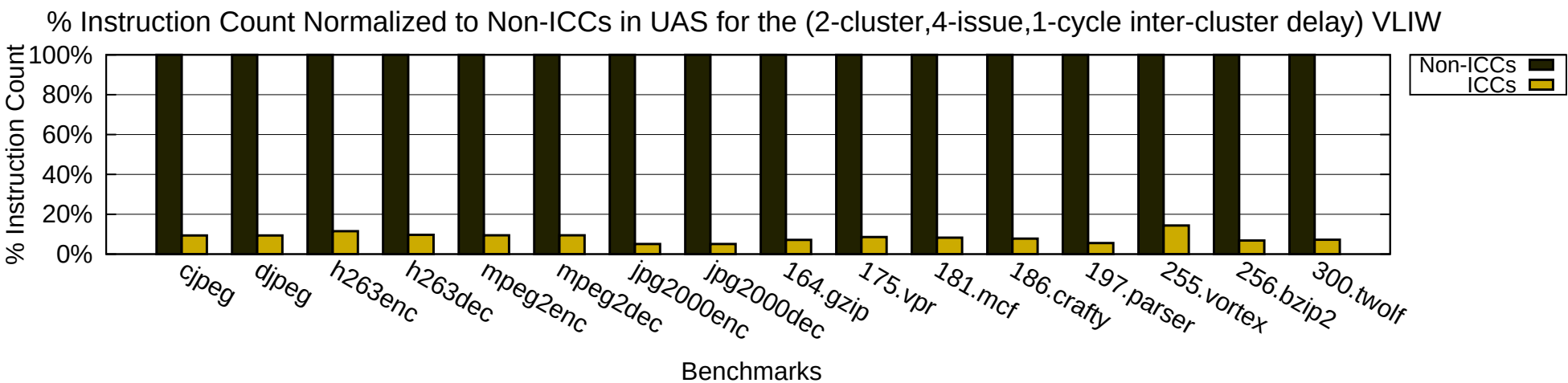
# Results: 4-cluster 4-issue, Instr Distribution

- Better resource utilization
- Less communication bottlenecks
- Potential for more ILP



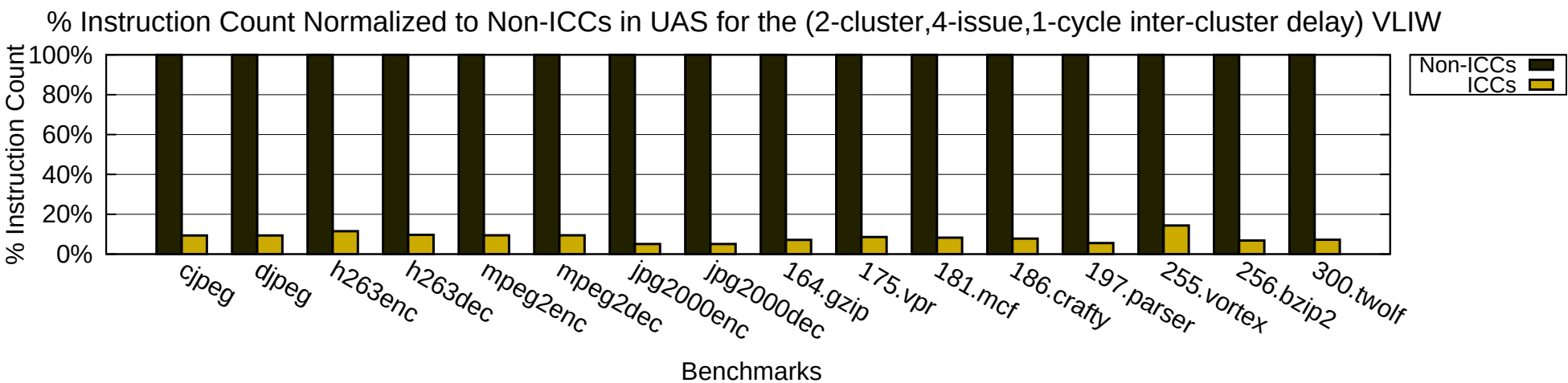
# Results: 2-cluster 4-issue, ICC ratio

- ICCs are still noticable



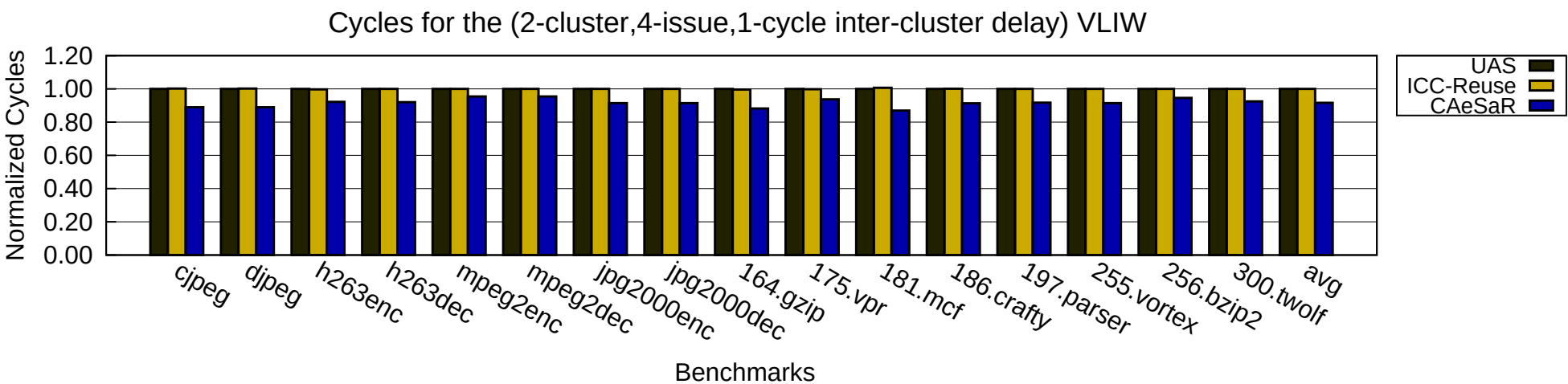
# Results: 2-cluster 4-issue, ICC ratio

- ICCs are still noticable
- 1 ICC in 10 non-ICCs



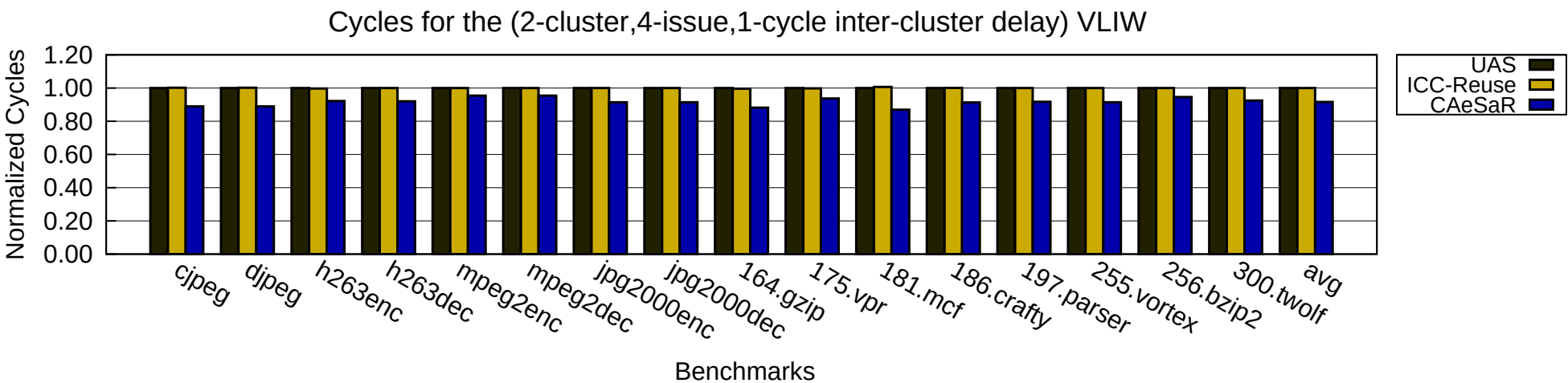
# Results: 2-cluster 4-issue, Sched cycles

- CAeSaR generates 8.4% more compact schedules on average



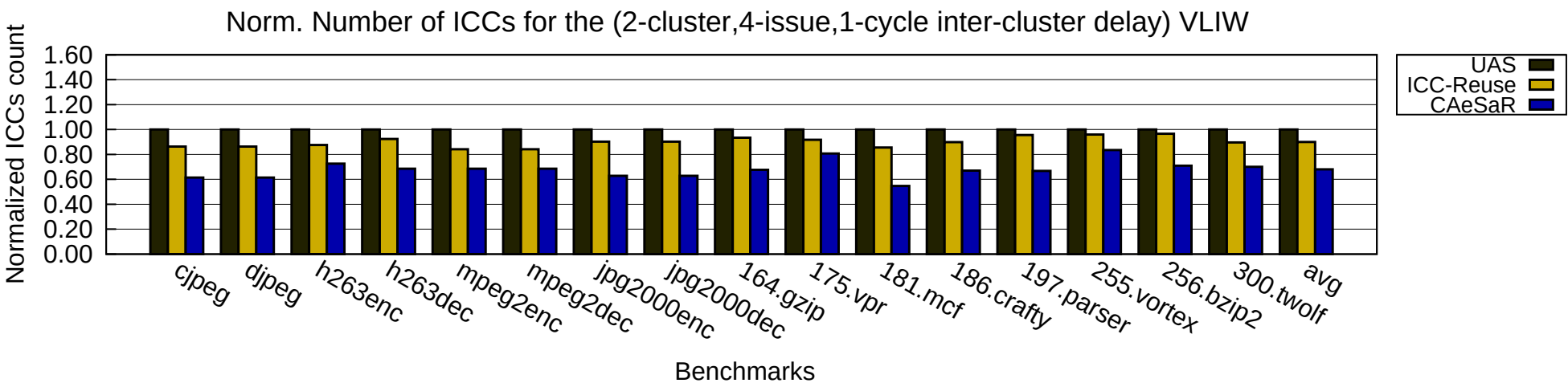
# Results: 2-cluster 4-issue, Sched cycles

- CAeSaR generates 8.4% more compact schedules on average
- ICC-reuse same performance as UAS (expected)



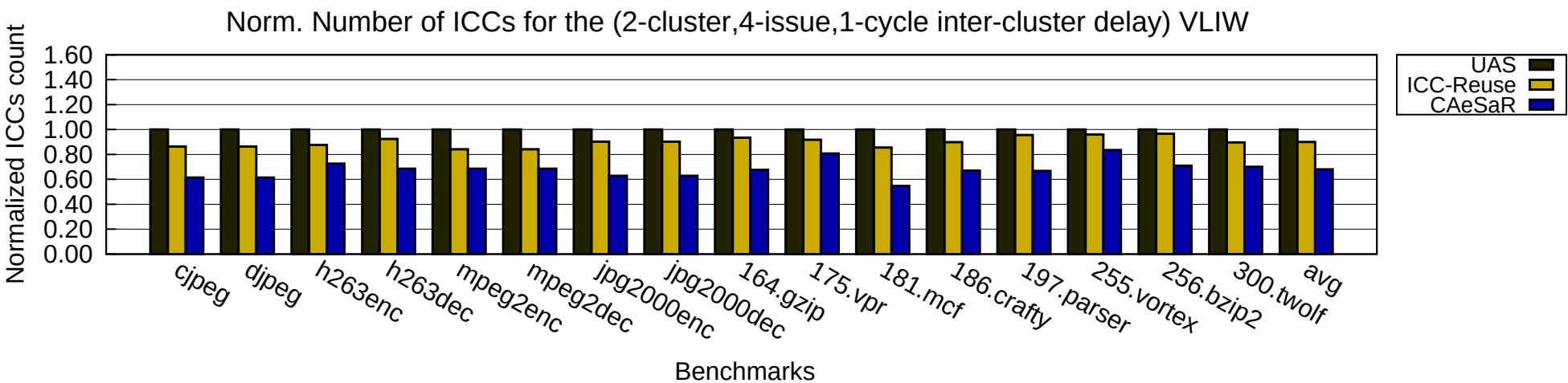
# Results: 2-cluster 4-issue, ICCs

- CAeSaR reduces the count of ICCs



# Results: 2-cluster 4-issue, ICCs

- CAeSaR reduces the count of ICCs
- CAeSaR generates fewer ICCs than ICC-reuse (proof of phase-ordering problem)



# Conclusion

Proposed CAeSaR Scheduling, a scheduler for clustered VLIWs that:

- Elliminates redundant Inter-Cluster copies
- Solves phase-ordering problem between Scheduling and ICC-reuse
- Generates more compact schedules compared to state-of-the-art

# CAeSaR: unified Cluster-Assignment Scheduling and communication Reuse for clustered VLIW processors

Vasileios Porpodas [†] and Marcelo Cintra [†⋆]

School of Informatics, University of Edinburgh[†]
Intel Labs Braunschweig[⋆]

CASES 2013