Heterogeneous Latch-based Asynchronous Pipelines

Girish Venkataramani
Tiberiu Chelcea
Seth C. Goldstein

Carnegie Mellon

Presenter:
Tobias Bjerregaard

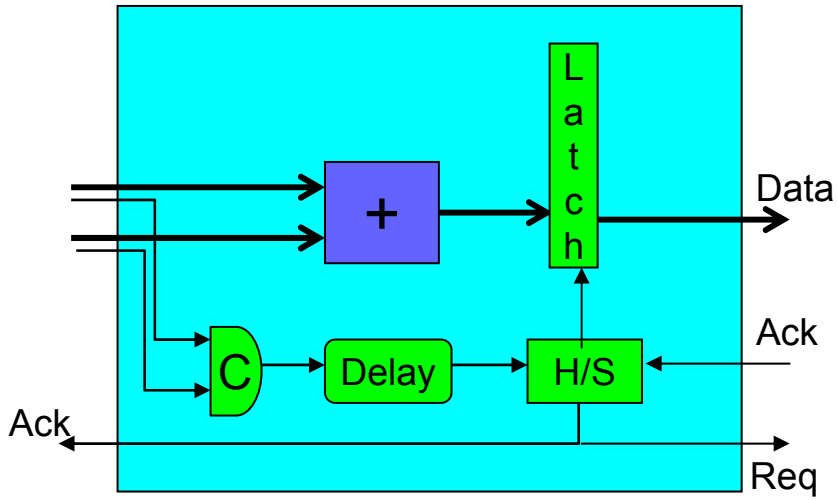
teklatech

Outline

- Introduction
- Latch Selection Algorithm
- Experimental Results
- Conclusions

Motivation

- Normally open latches are attractive for bundled data designs, e.g., Mousetrap, [Singh, ICCD 01]



\+ High-performance:

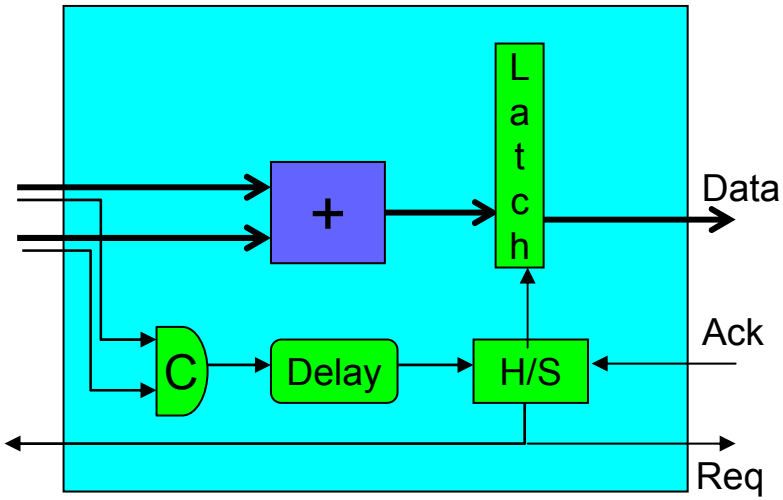
Short critical path to open latch

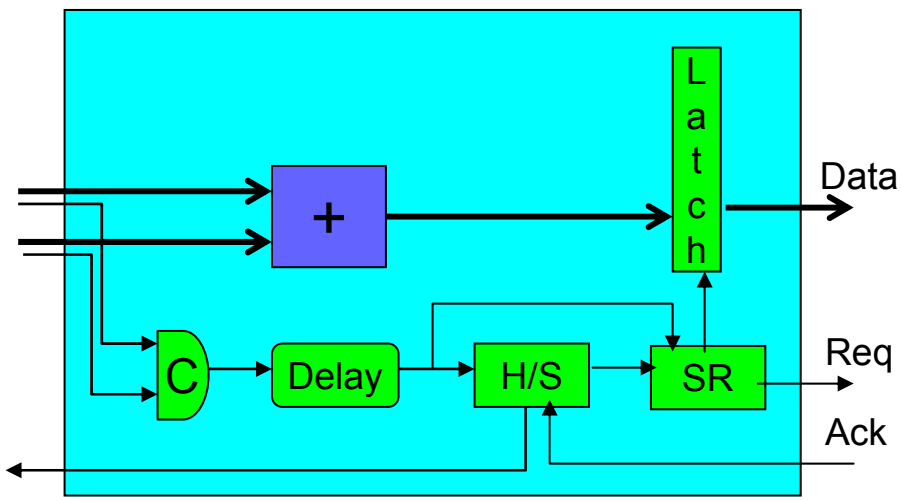
– Power hungry:

Data glitches spill over to downstream stages

Motivation

- Self-Resetting (SR) Latches address the glitching problem, [Chelcea, DAC 07]
  - D-Latch closed during active computation (filters glitches)
  - D-Latch is opened just before stage computation stabilizes





\+ 2x improvement in energy-delay*

– 10% performance slowdown*

* Mediabench suite, [Lee, Micro 97]

Contributions

- Build heterogeneous pipelines
  - Use D-latches for timing critical stages
  - Use SR-latches for the rest
- Module Selection problem
  - What is timing critical?
  - When is an SR-latch warranted?
- Automatic Latch Selection Algorithm
  - Experimental results:

    Heterogeneous pipelines have equivalent performance to D-latches and are more energy-efficient than either homogeneous D-latch or SR-latch pipelines

Outline

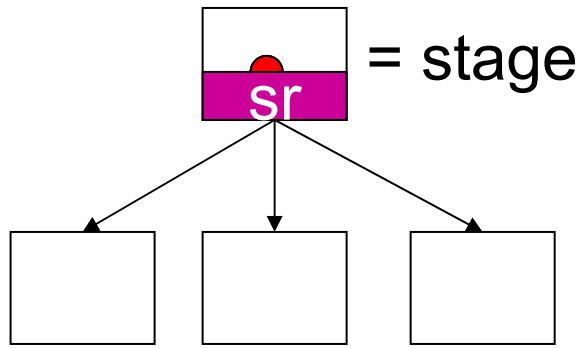
- Introduction
- **Latch Selection Algorithm**
- Experimental Results
- Conclusions

Latch Selection Algorithm

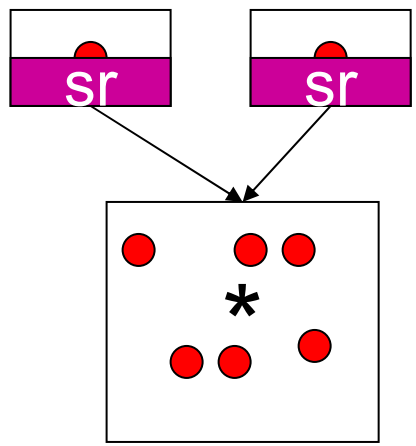
- Objectives: Get best of both worlds
  - Performance of D-latches
  - Energy efficiency of SR-latches
- Approach: Balance the use of SR-latches
  - Too many ➔ bigger and slower designs
  - Too few ➔ high energy consumption
- Algo properties: Three heuristics used to track timing criticality and estimate effect of datapath glitches

Power Heuristics

- Data glitches are proportional to the datapath fanout
  - Use SR-latches if fanout >= 2



- Protect computation-intensive stages
  - Assign SR-latches to inputs
  - Bit-operations on datapath used to estimate computation intensiveness



Timing Criticality

- SR-latch controllers introduce delay when opening latches
  - Use D-latch if stage is timing critical

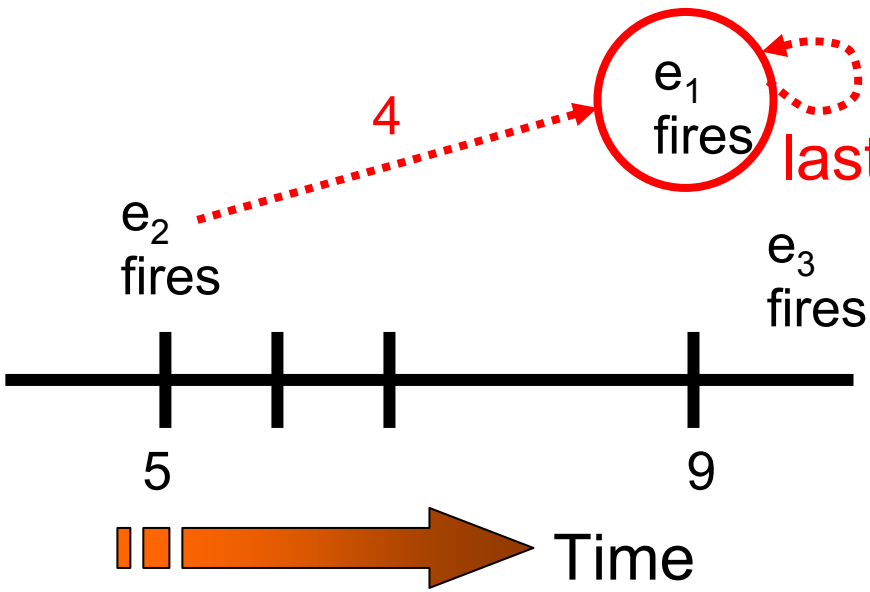
- Determine the system's critical stages using the Global Critical Path (GCP), [Venkataramani, DAC 07]

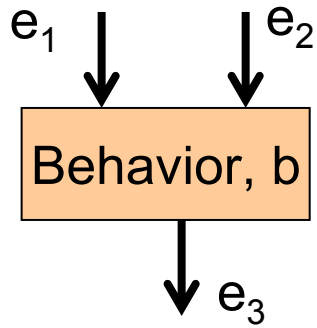
Timing Analysis

- Analysis produces steady-state event firing times
  - Events are handshake signal transitions
  - Behaviors are dependence relations between events
- Cycle time: Time difference between an event recurrence
- Alternative representation of cycle time
  - Set of slack values: time difference between input events
  - Global Critical Path (GCP): longest zero-slack path
  - Global slack: Timing budget for GCP tolerance
    - How much can stage be slowed without changing GCP

Event Slack

- Use concept of Time Separation of Events (TSEs) to compute *slack, GCP and global slack*





Slack(e_2, b) = 4
Slack(e_1, b) = 0

0-slack input
is locally critical
[Fields, ISCA 01]

Global Critical Path (GCP)

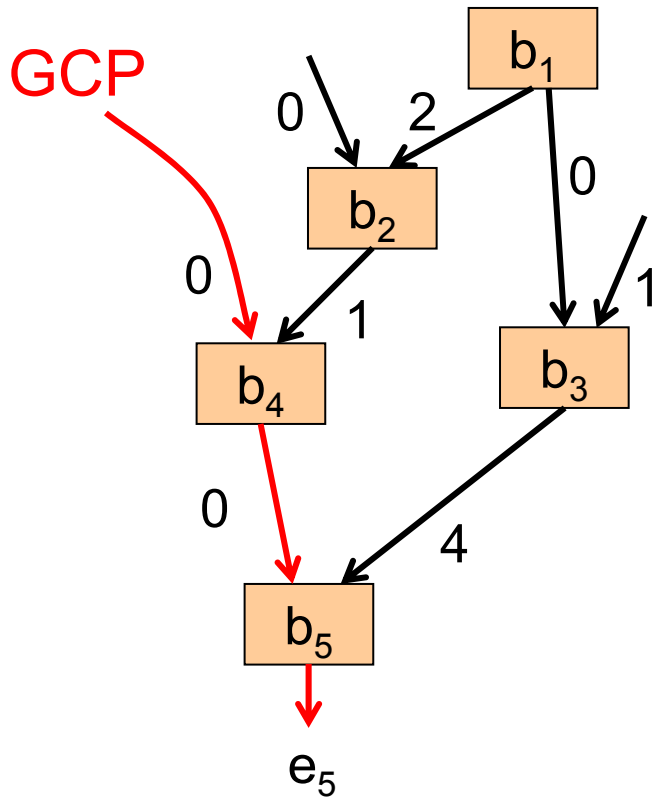
- GCP is longest path of zero-slack input events, [Venkataramani,DAC 07]
  - Equivalent to the critical cycle

- Bottom-Up computation of Cycle time:
  - Length of GCP cycle = cycle time

GCP is the sequential critical path of the system.
It represents the primary performance bottleneck

Global Slack

- Minimum cumulative slack to the GCP
  - If event is on the GCP
    - GSlack(b_4) = GSlack(b_5) = 0
  - Otherwise:
    - GSlack(b_2) = 1
    - GSlack(b_3) = 4
    - GSlack(b_1) = Min(0+4, 2+1) = 3

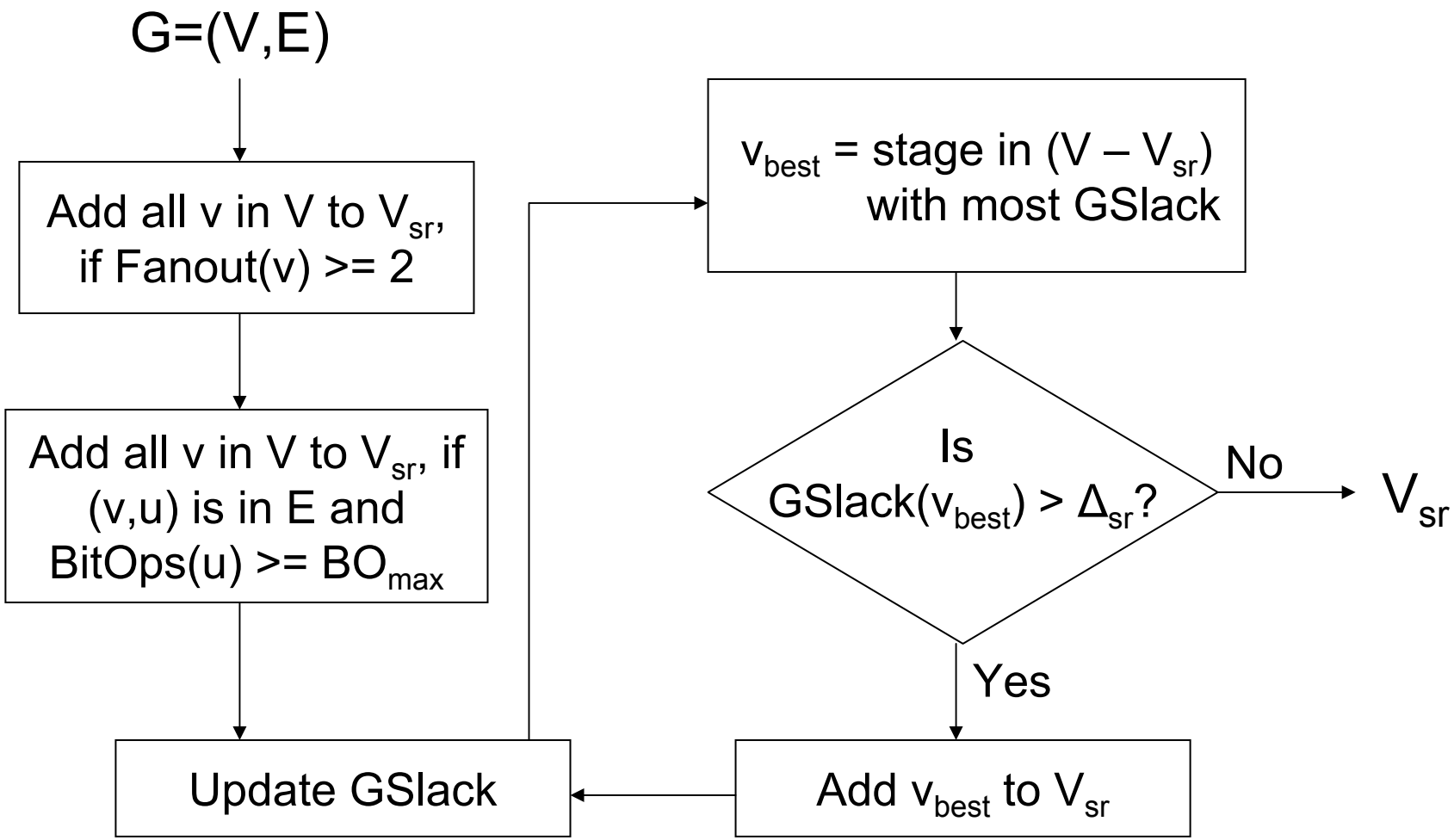


Global slack is a measure of how much a behavior can be delayed without affecting global performance

Timing Criticality Heuristic

- Let Δ_{sr} be delay overhead introduced by SR-latches
- Iterative algorithm: Assign an SR-latch when global slack is larger than Δ_{sr}
  - Update timing
  - Repeat; look for more opportunities

Latch Selection Algorithm Overview



Complexity: $O(|V||E| + |V|^2)$

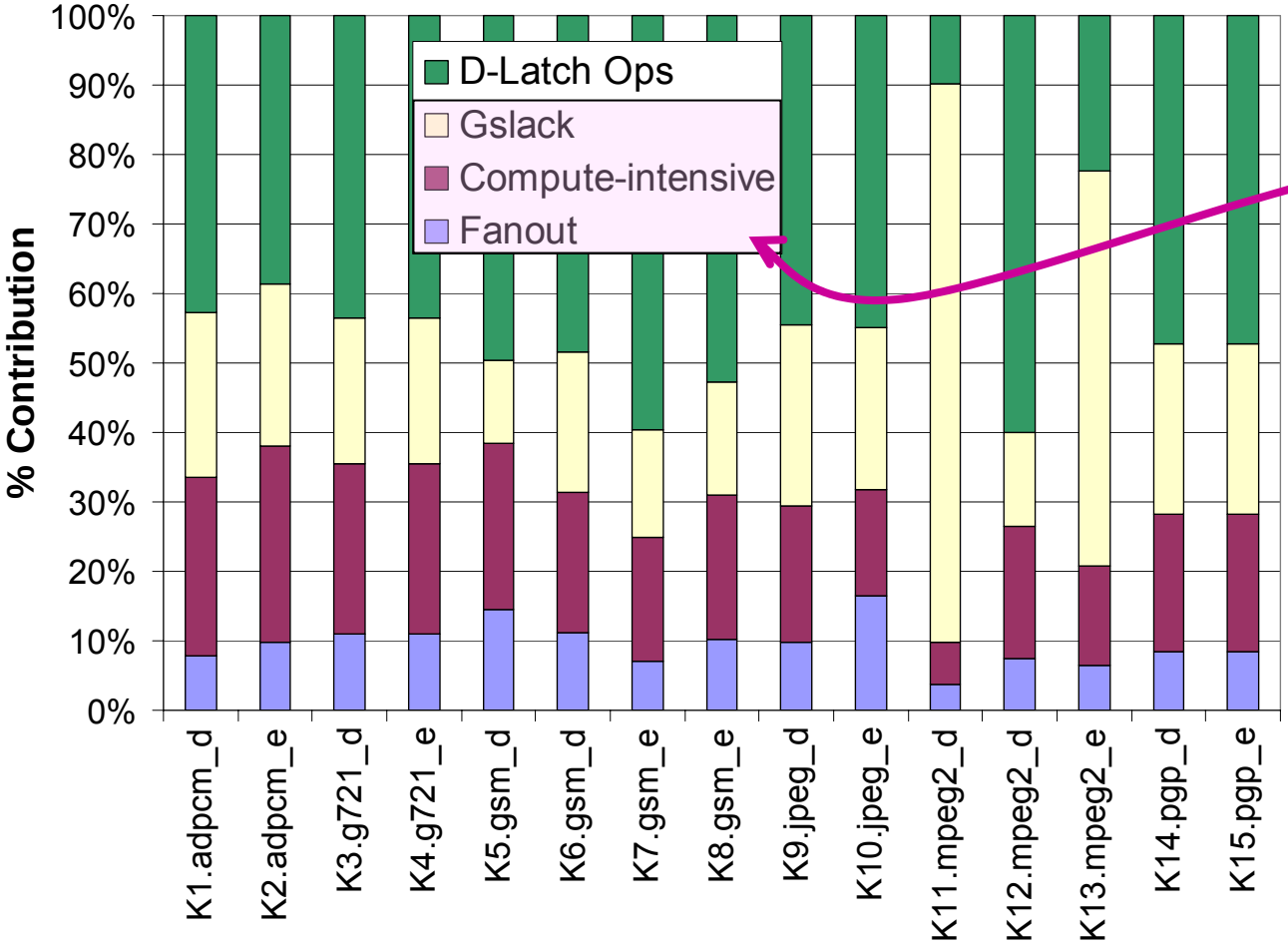
Outline

- Introduction
- Latch Selection Algorithm
- **Experimental Results**
- Conclusions

Experimental Setup

- Implemented latch selection algorithm within CASH, a compiler synthesizing 4-phase bundled circuits from C [IWLS 04]
- Applied on 15 Mediabench kernels [Lee, Micro 97]
- Circuits mapped to [180nm/2V] STMicro standard-cell library
- Synopsys DC used to estimate energy, Modelsim used for gate-level timing estimation

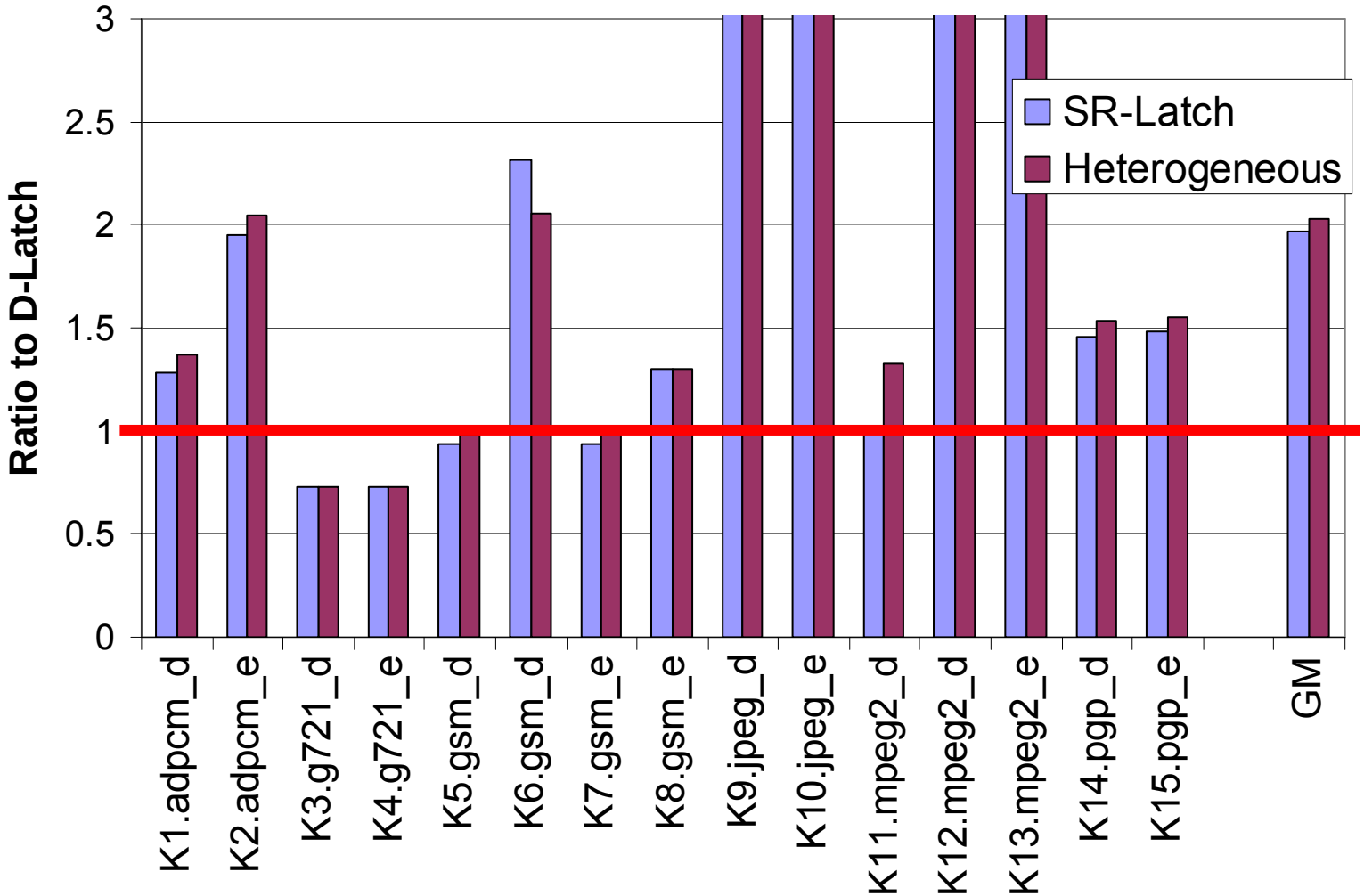
Impact of Heuristics



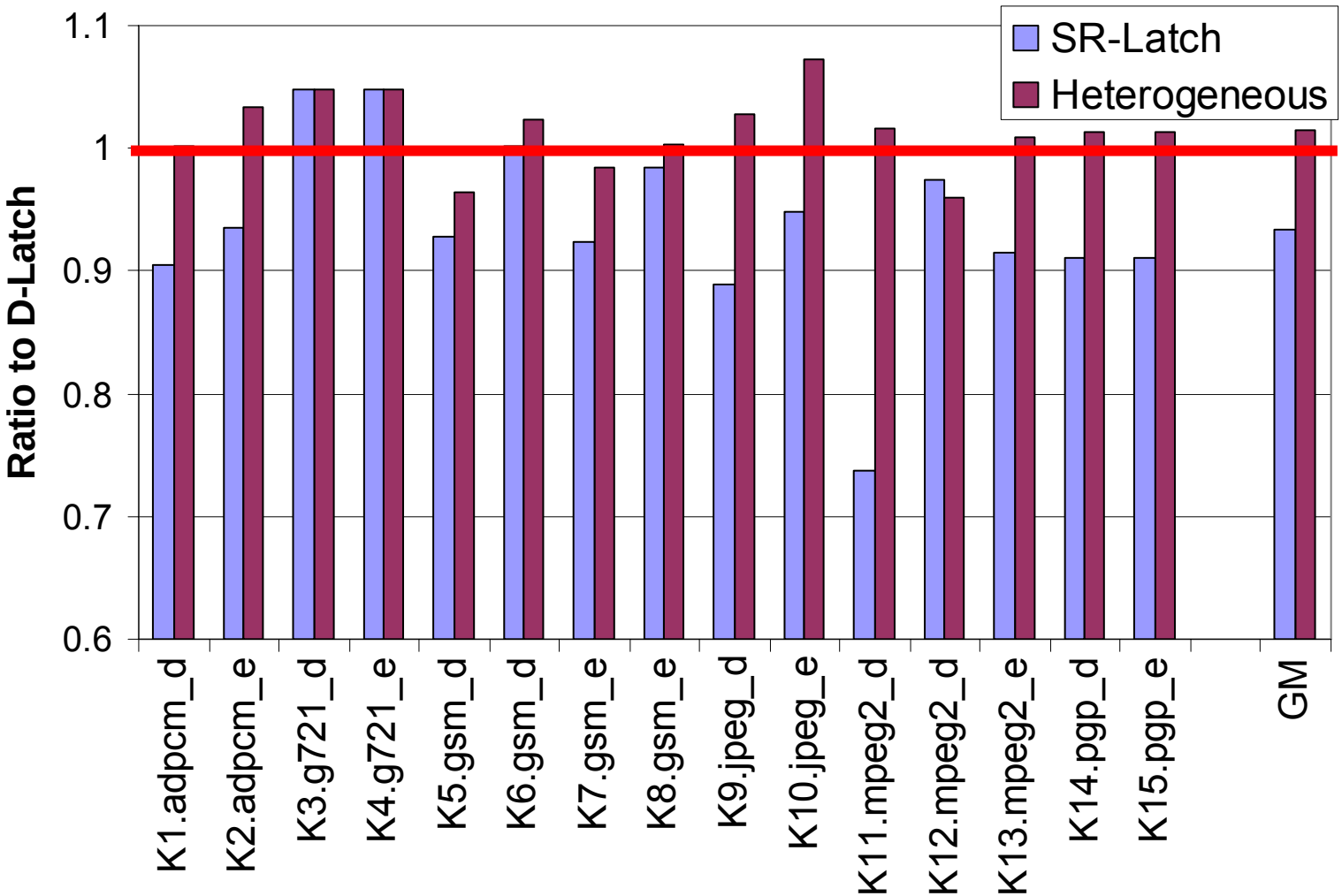
SR-Latch Stages

Combined effect of heuristics contributes to energy efficiency

Energy-Delay



End-to-End Execution Time



Outline

- Introduction
- Latch Selection Algorithm
- Experimental Results
- Conclusions

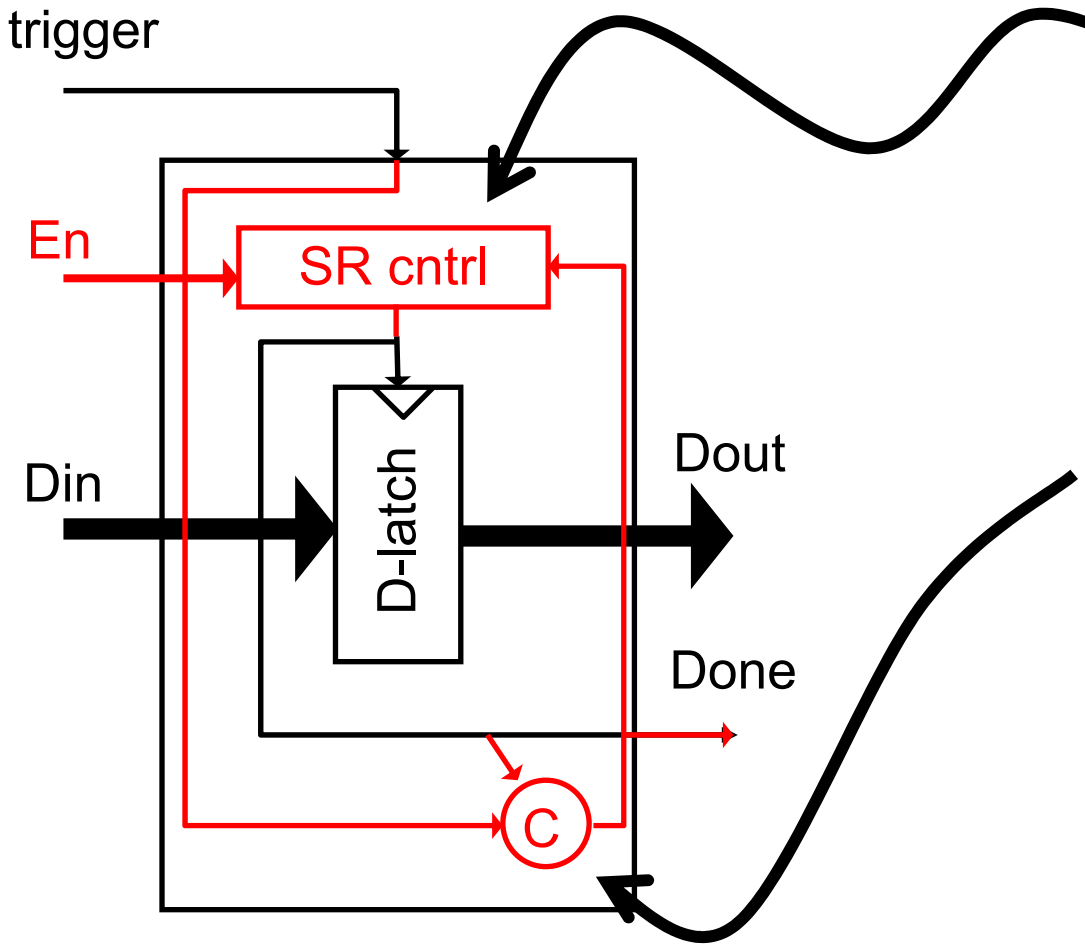
Conclusions

- D-latches are power-hungry and SR-latches are slow for bundled-data pipelines
- Heterogeneous latch selection algorithm
  - Global slack to guide timing-critical selection
  - Simple heuristics to guide power-critical selection
- Heterogeneous latch pipelines are more energy-efficient than either homogeneous D-latch or homogeneous SR-latch pipelines

Thank You! Questions?

Self-Resetting (SR) Latches

[Chelcea, DAC 07]

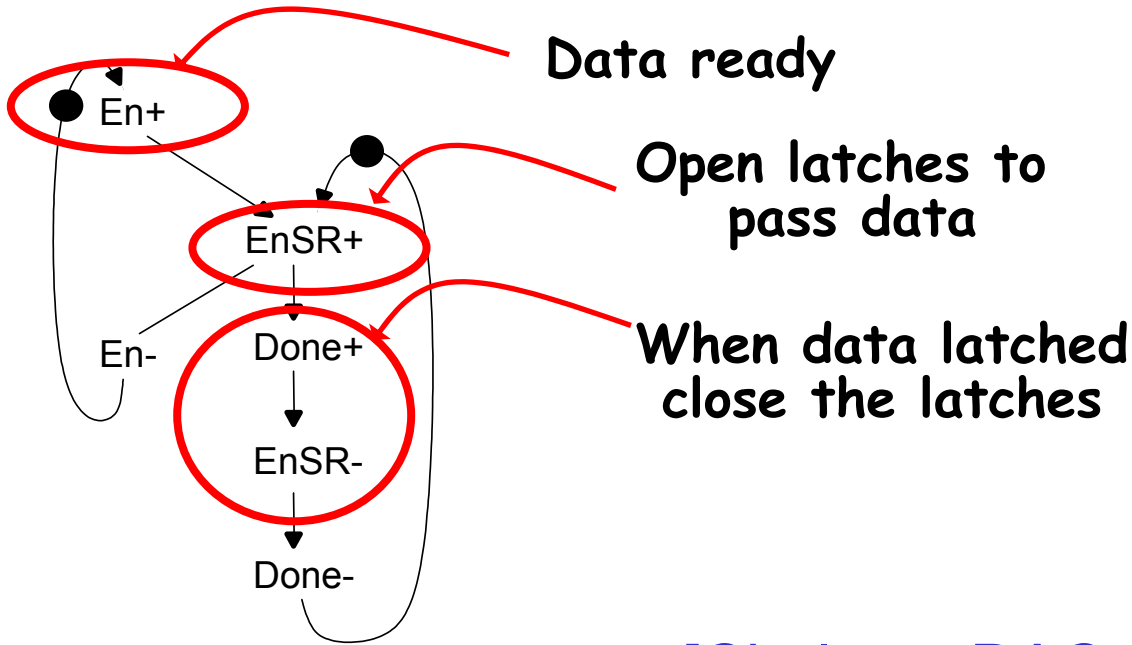


Area and control-path power overhead

Timing overhead

Benefit: Datapath power savings

SR-latch behavior



STG specification

[Chelcea, DAC 07]

- Eliminate glitches:
  - open only after data is ready
  - close as soon as data latched
- Eliminate overheads:
  - open before handshake starts