Safe Overclocking of Tightly Coupled CGRAs and Processor Arrays using Razor

Alex Brant, Ameer Abdelhadi,
Douglas Sim, Shao Lin Tang, Michael Yue,
Guy Lemieux

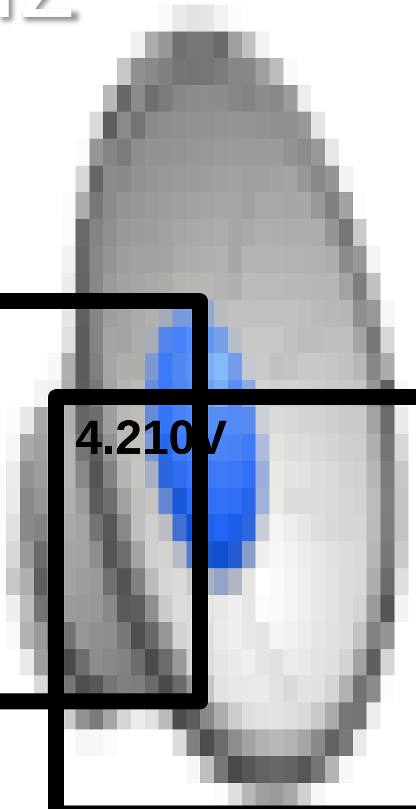
Overclocking… is it safe?

- Clock frequency determined by 2 things:
  - CAD timing analysis (timing margins)
  - speed binning of actual wafers + chips (variation)

- Can you go faster?
  - Yes, if your chips are fast
  - Yes, if your data is not “worst-case”, eg carry propagation
  - Yes, if you do not want “safe” timing margin guardbands

Overclocking… is it safe?

3.8GHz 4.2 V



3.818GHz !!!

4.210V

Overclocking… is it safe?

- How fast is too fast?
  - Blows up
  - Fails to POST
  - Fails to boot
  - Blue screen of death
  - Random crashes
  - Data errors in documents and spreadsheet
- When these problems go away, is it safe?

Overclocking… is it safe?

- Root cause: timing errors
  - Problem 1: can we **detect** them?
    - Yes, e.g. using **Razor**
  - Problem 2: can we **correct** them?
    - Yes, using **Razor** with feed-forward pipelines
      - Pipeline must be ‘replayed’, input data ‘unfetched’
    - Not possible with general sequential logic
      - Need ‘spare’ cycles to ‘unfetch’ input data
      - Cyclic dependencies make this difficult

Overclocking… is it safe?

- If not for general logic, what about…
  - Traditional CPU pipelines?
    Yes:
    - Feed-forward, correctable by Razor-replay
  - Multi-core CPUs?
    Yes:
    - Each CPU is a traditional pipeline
    - Loosely coupled
      - Other CPUs tolerate race conditions

Overclocking… is it safe?

- If not for general logic, what about…
  - Ambric-style processor arrays?
    Yes:
    - Like multi-core CPUs
    - Loosely coupled
      - Neighbour CPUs tolerate uncertainty of arrival time
  - Tightly coupled processor arrays or CGRAs?
    No: neighbour CPUs cannot tolerate delays!

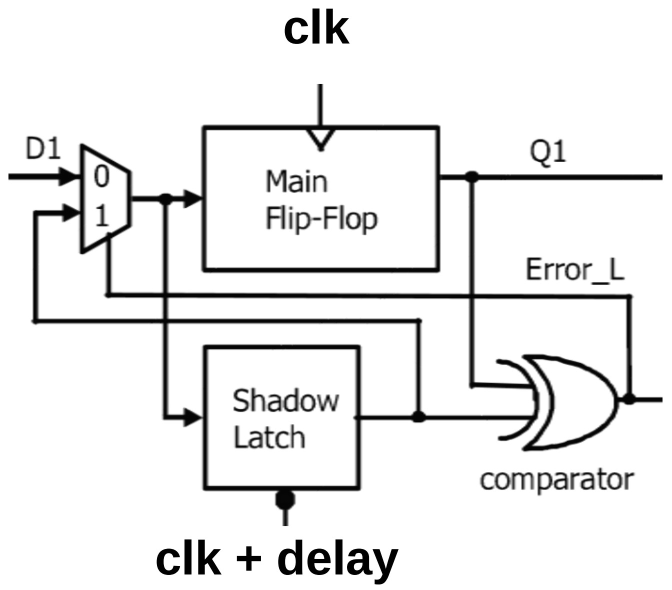
Main Contribution

- Extends Razor error correction to…
  - tightly coupled processor arrays, CGRAs
  - time-multiplexed FPGAs/CGRAs

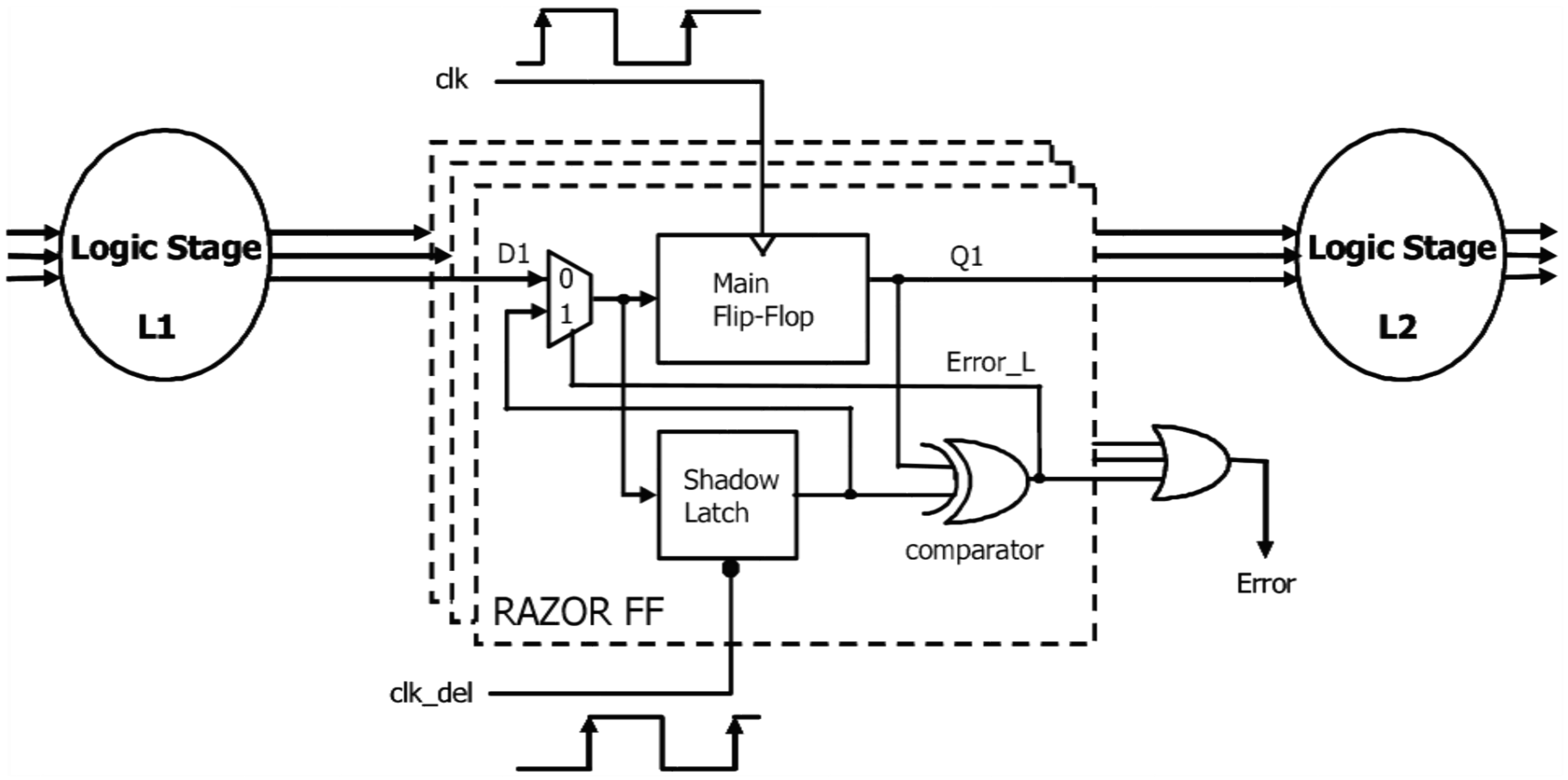
Tightly coupled means…

- Pre-scheduled communication
  - **Data must be present during cycle X**
- No “data presence” indicators / handshakes

Razor FF in Pipeline

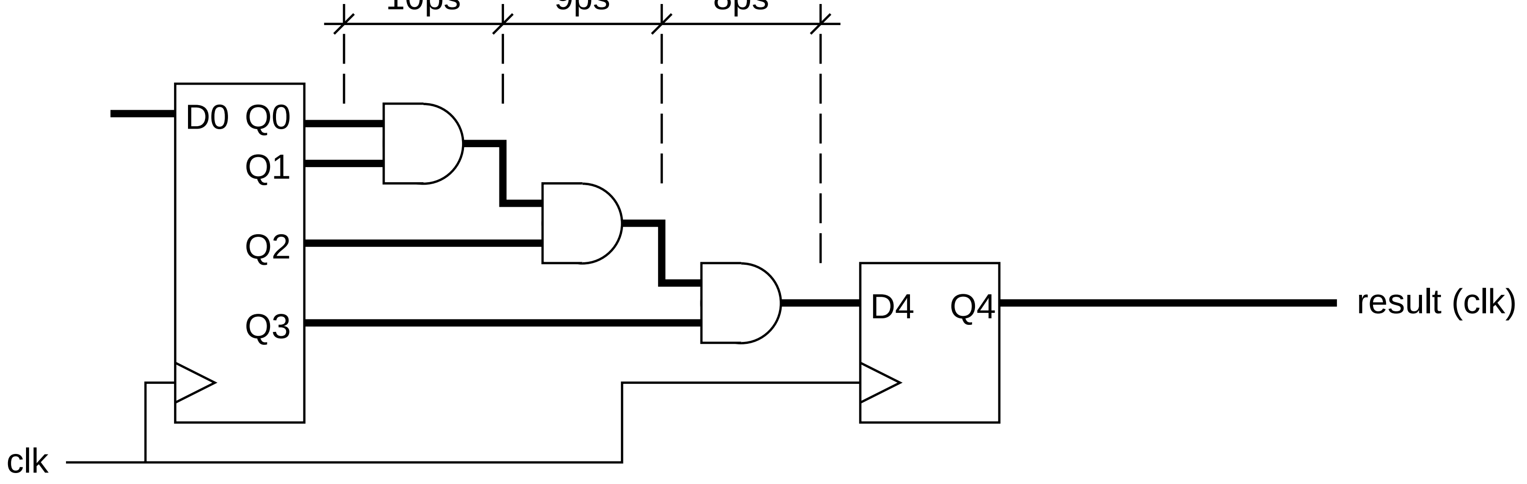


Razor FF in Pipeline



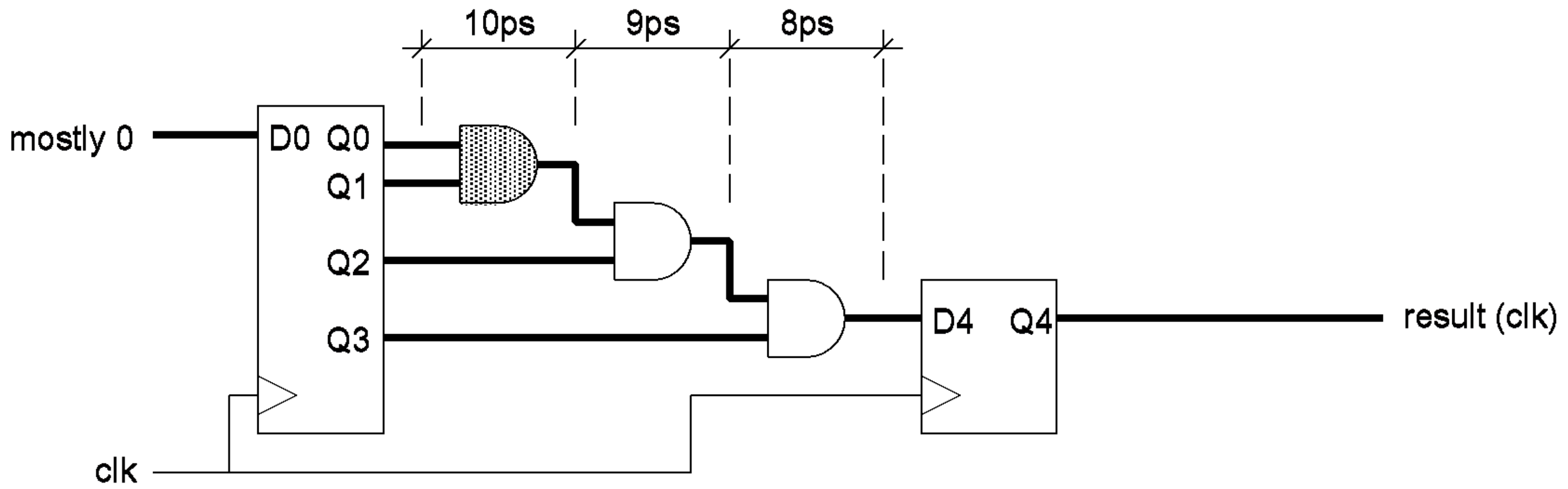
How can we overclock?

10+9+8 = 27ps clock



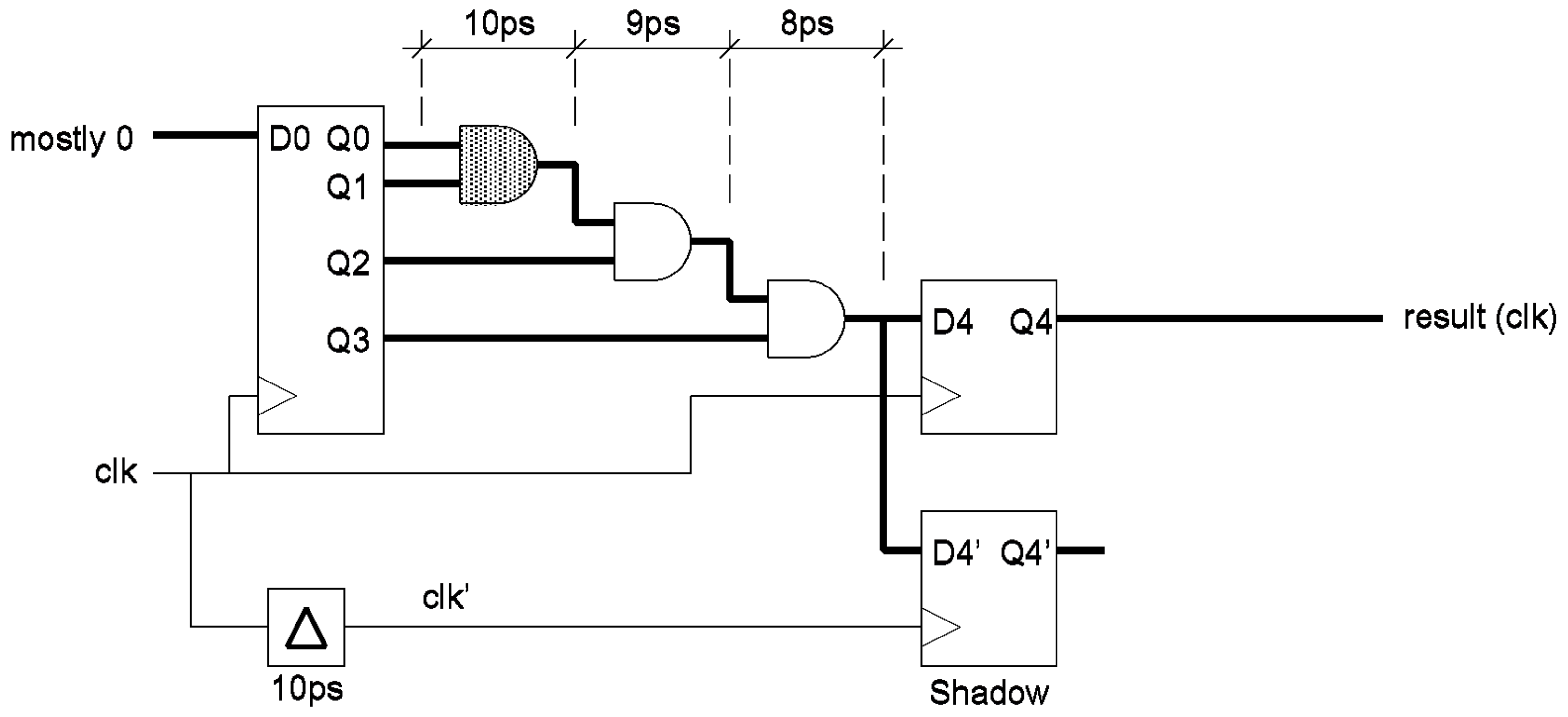
How can we overclock?

9+8 = 17ps clock, most of the time



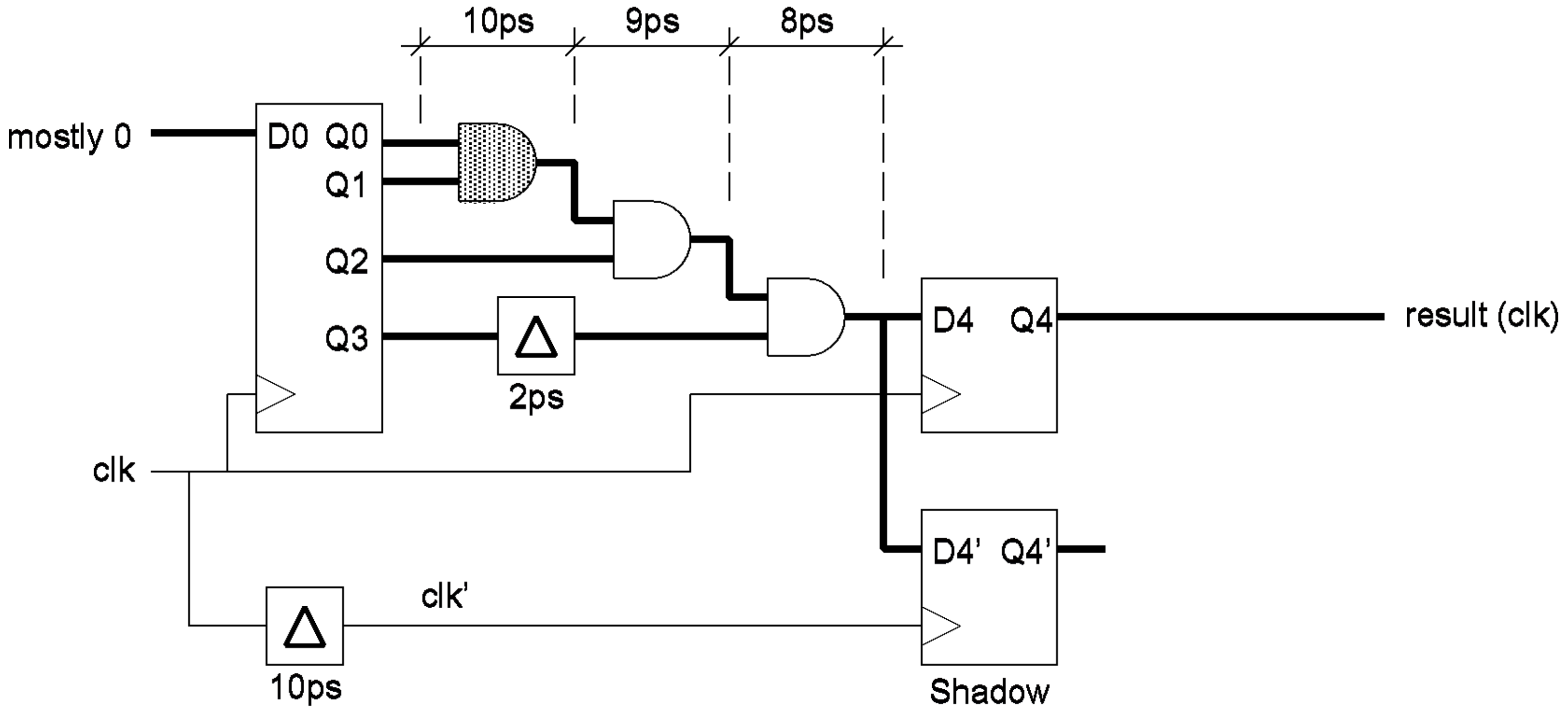
How can we overclock?

9+8 = 17ps clock, most of the time



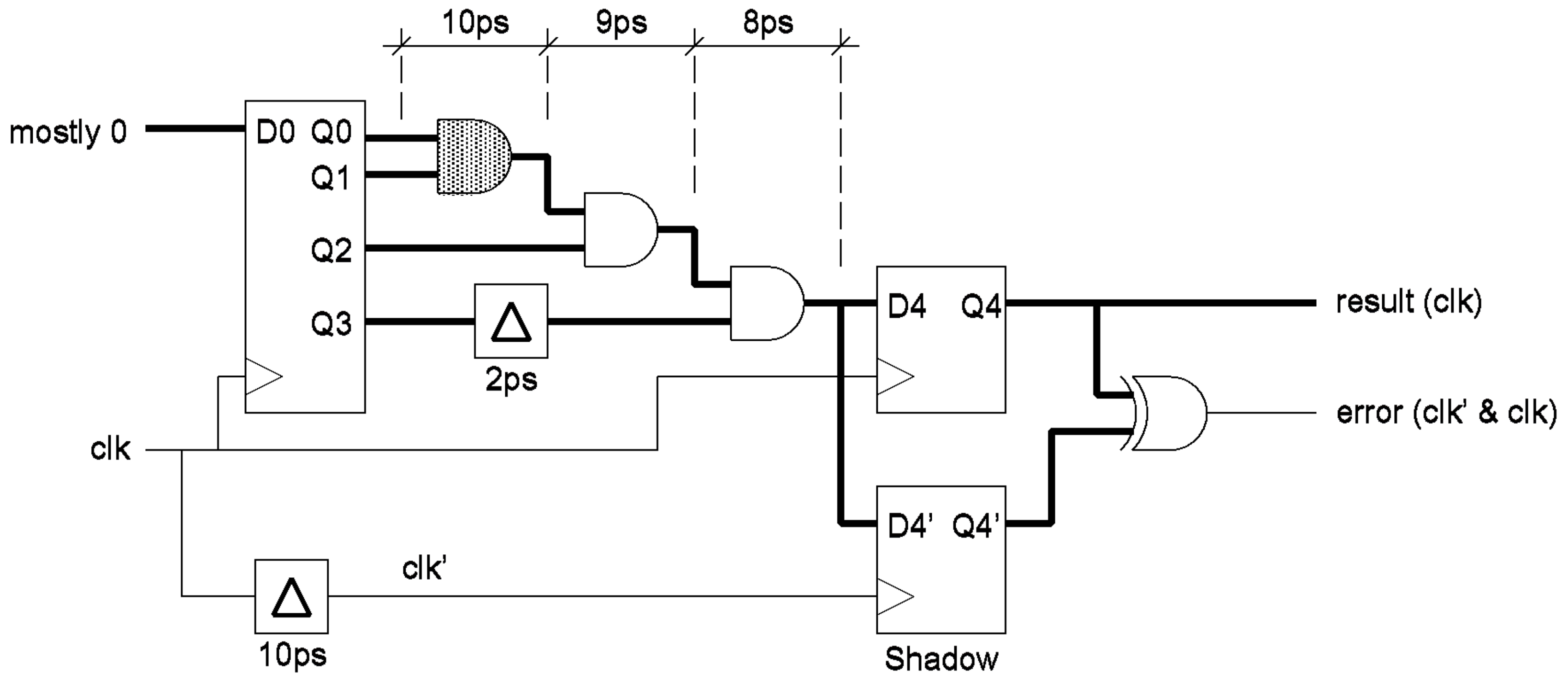
How can we overclock?

9+8 = 17ps clock, most of the time



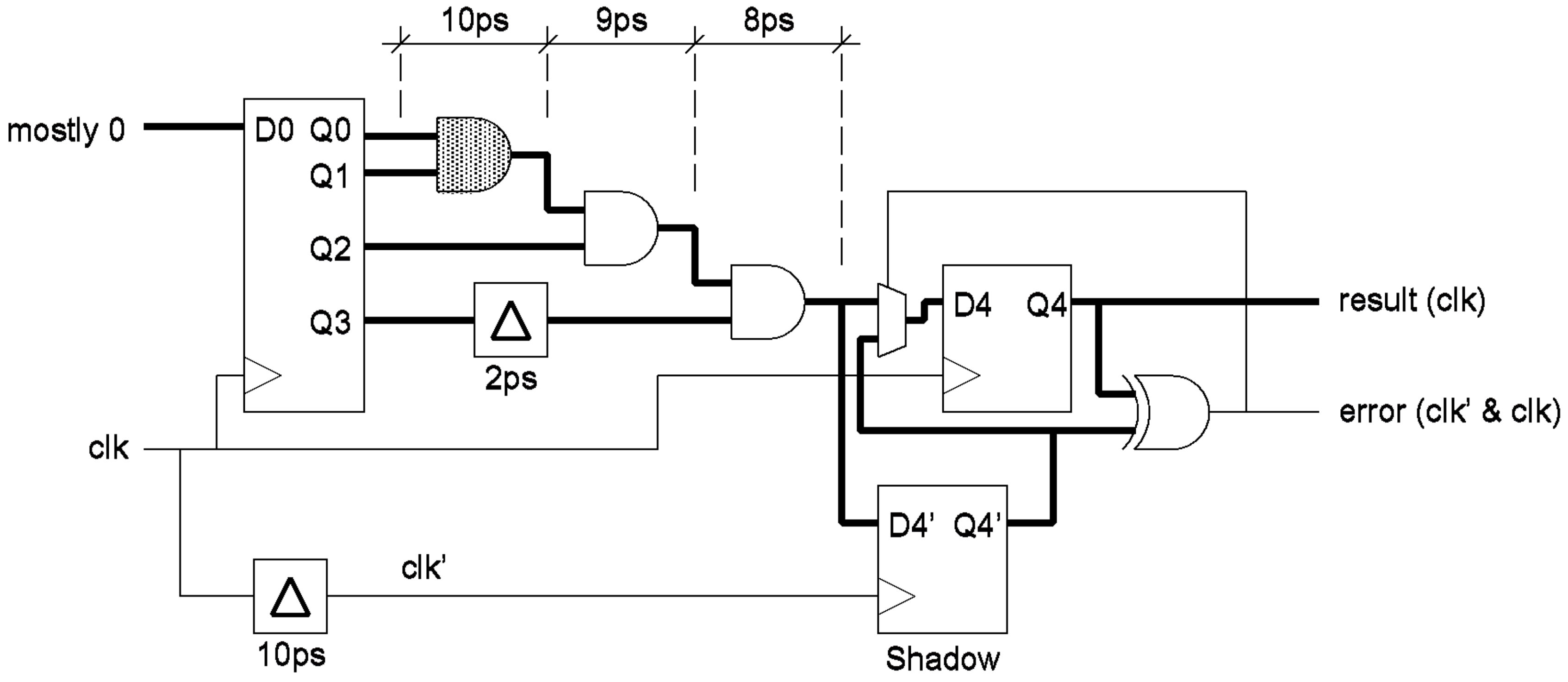
How can we overclock?

9+8 = 17ps clock, most of the time



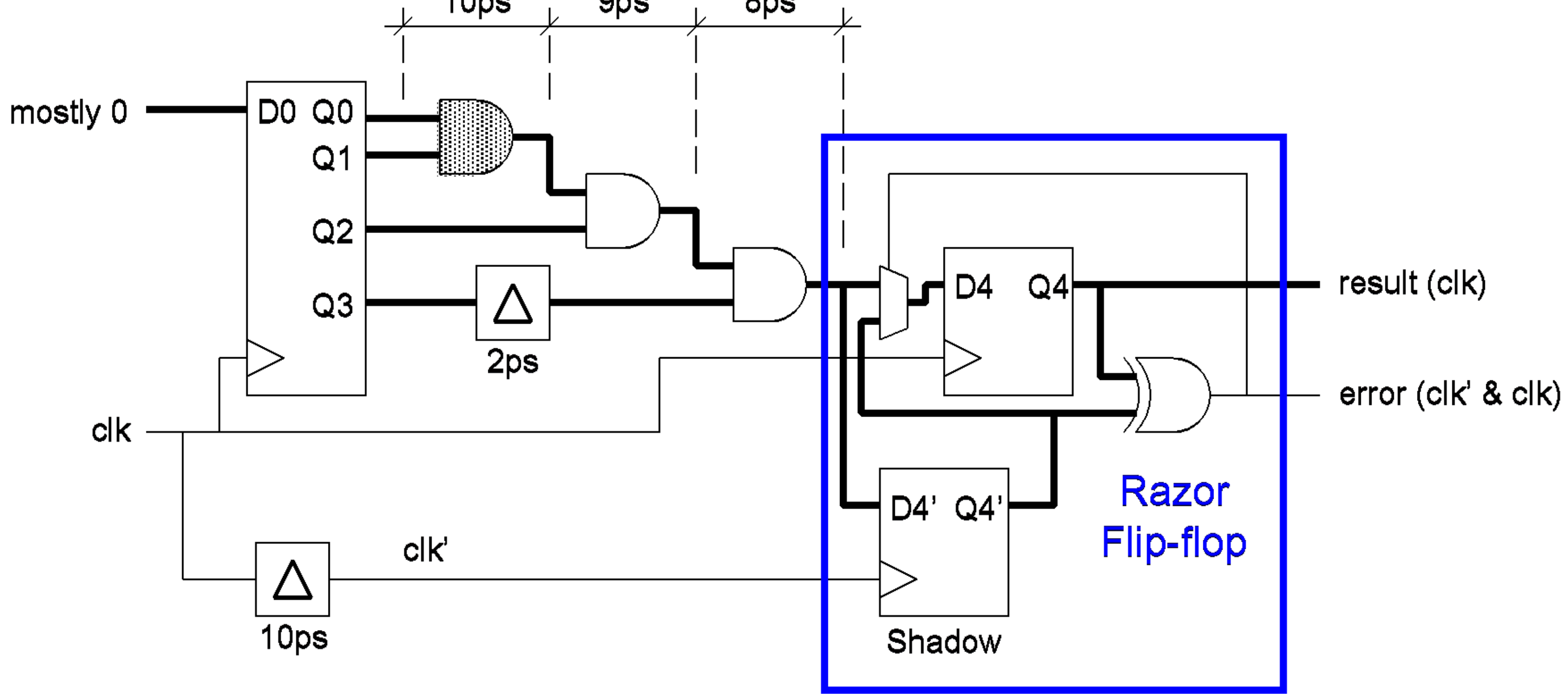
How can we overclock?

9+8 = 17ps clock, most of the time



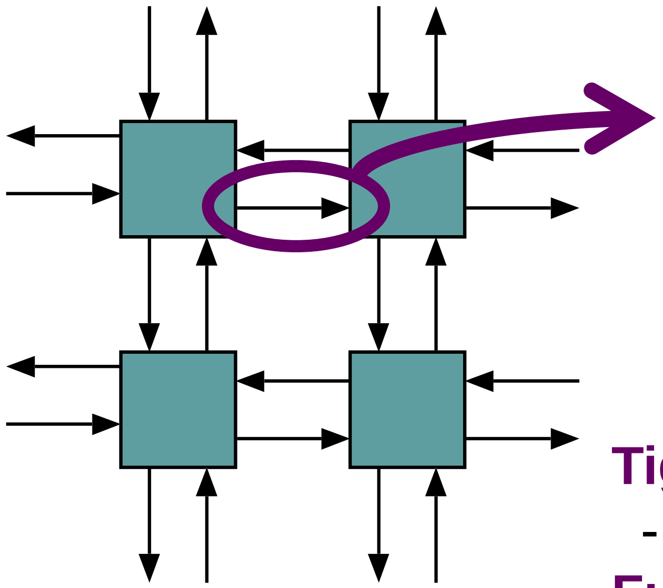
How can we overclock?

9+8 = 17ps clock, most of the time



Array Architecture

Processor Array



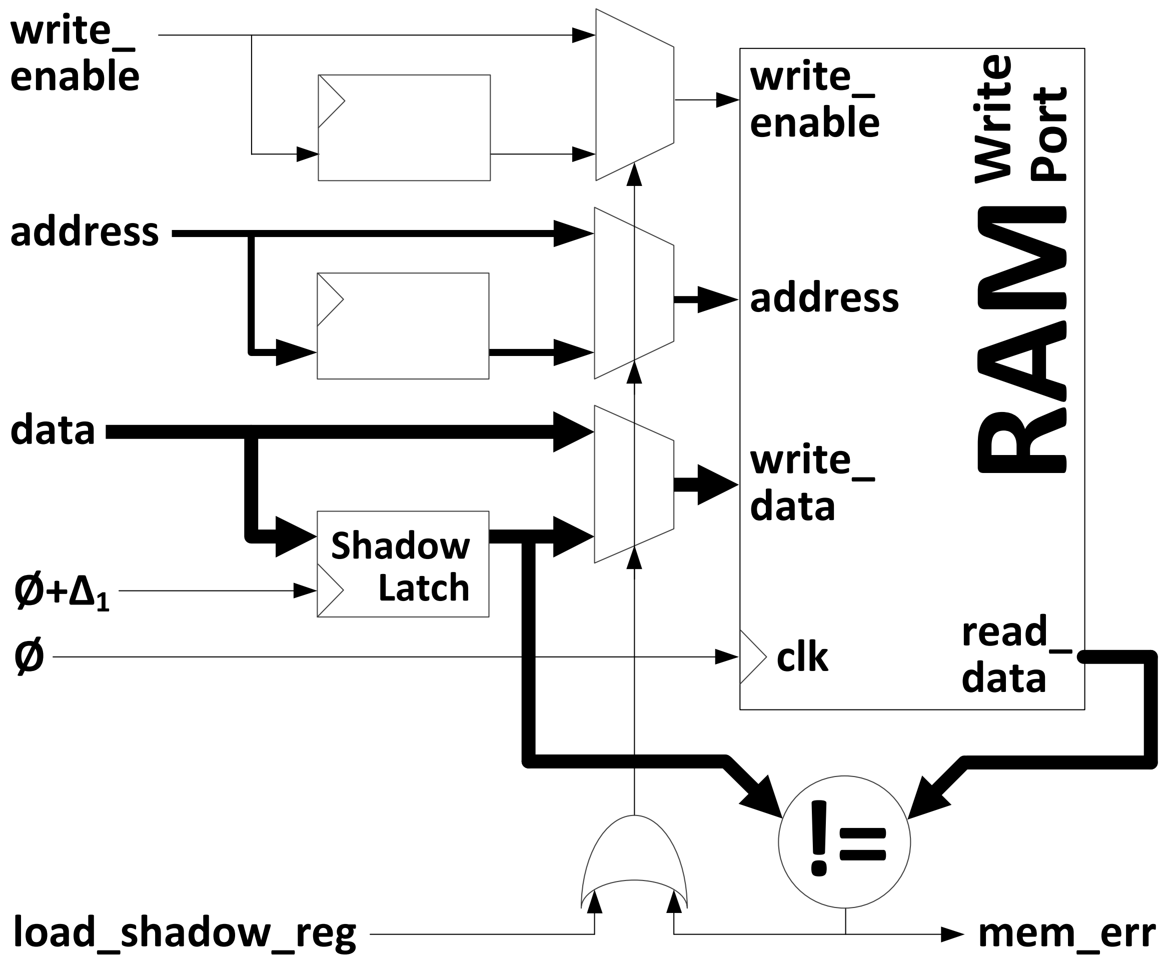
Tightly coupled communication:

- can be FIFO-based or ‘mailbox’-based

Fully bypassed:

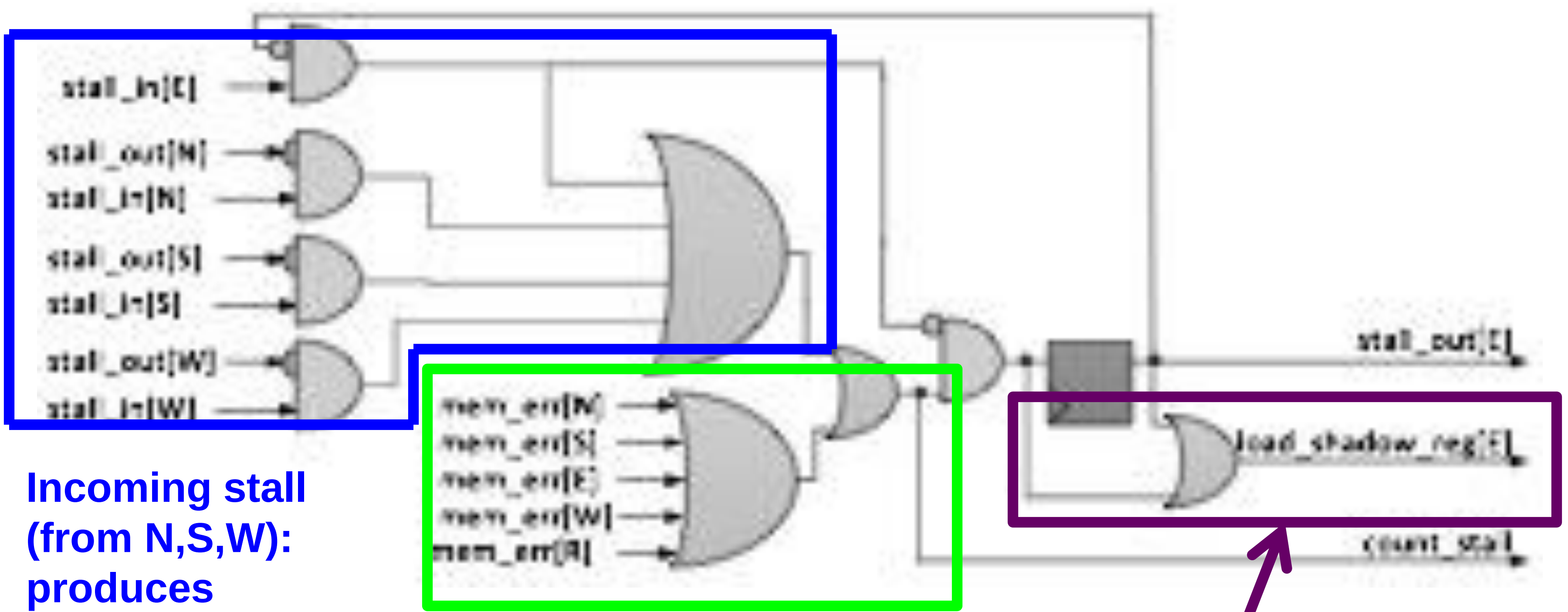
- write on cycle X
- read on cycle X+1, ie address provided cycle X

Add "Razor" to Block RAM



Processor Error Detection

(for East direction only)



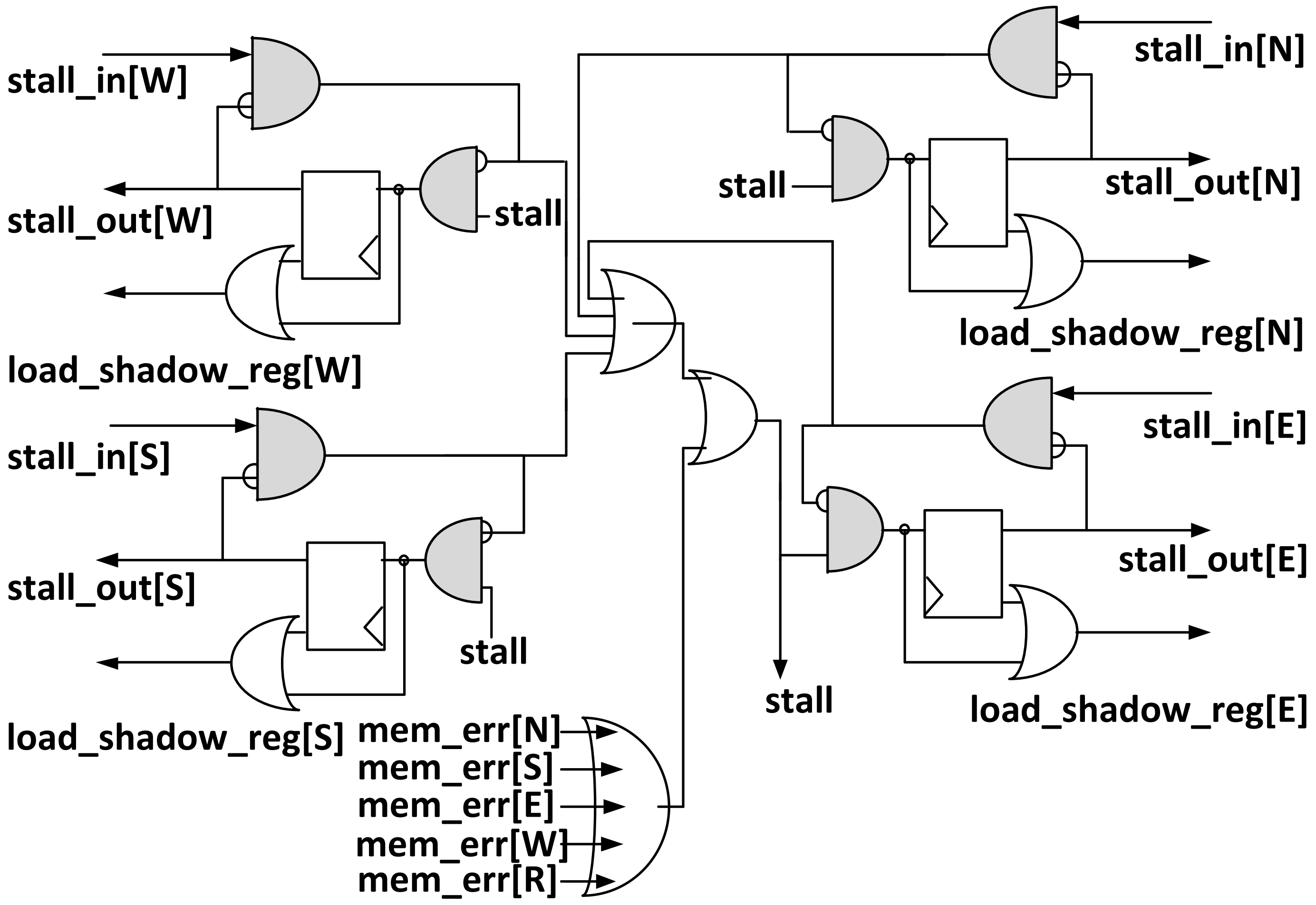
Incoming stall (from N,S,W): produces Outgoing E stall

Processor memory error: Causes stall

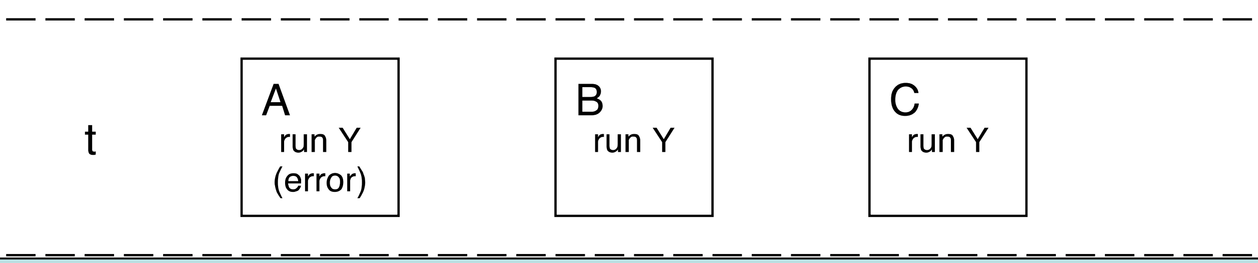
Early warning signal: prevents incoming data

Processor Error Detection

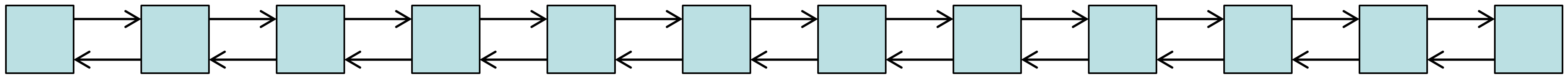
(all four directions)



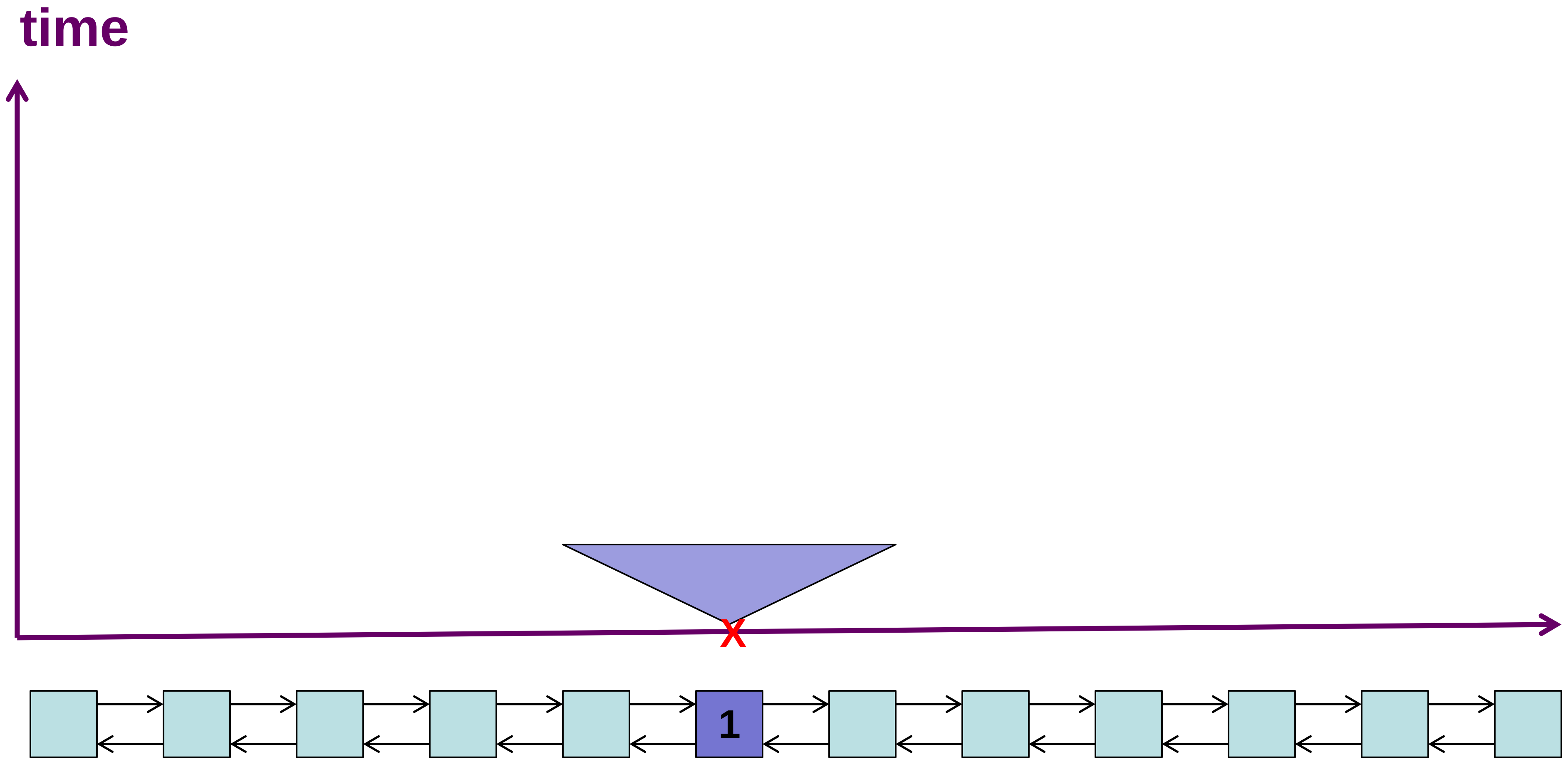
Writes entering error region….



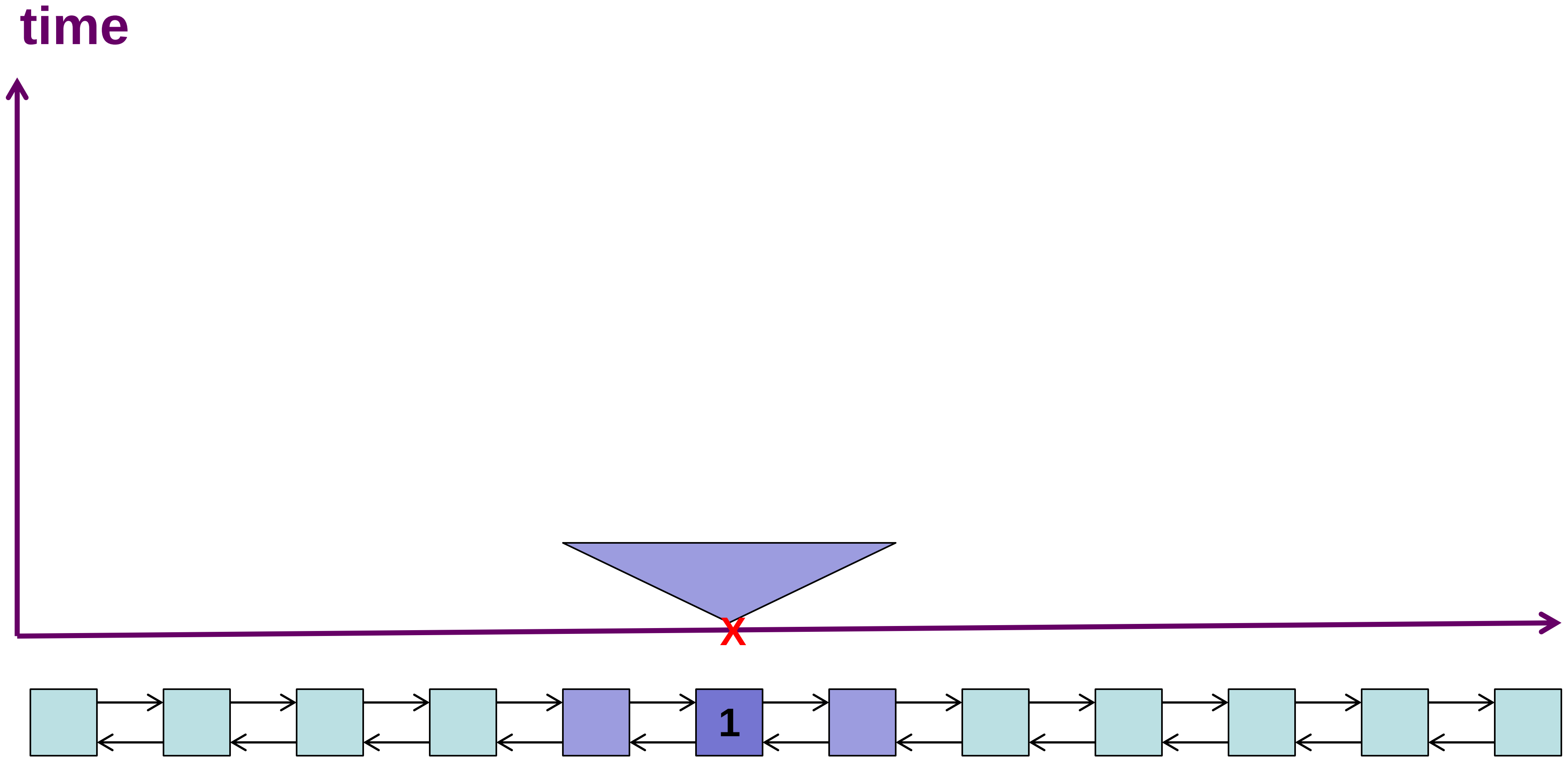
Razor Stall Propagation (1D)



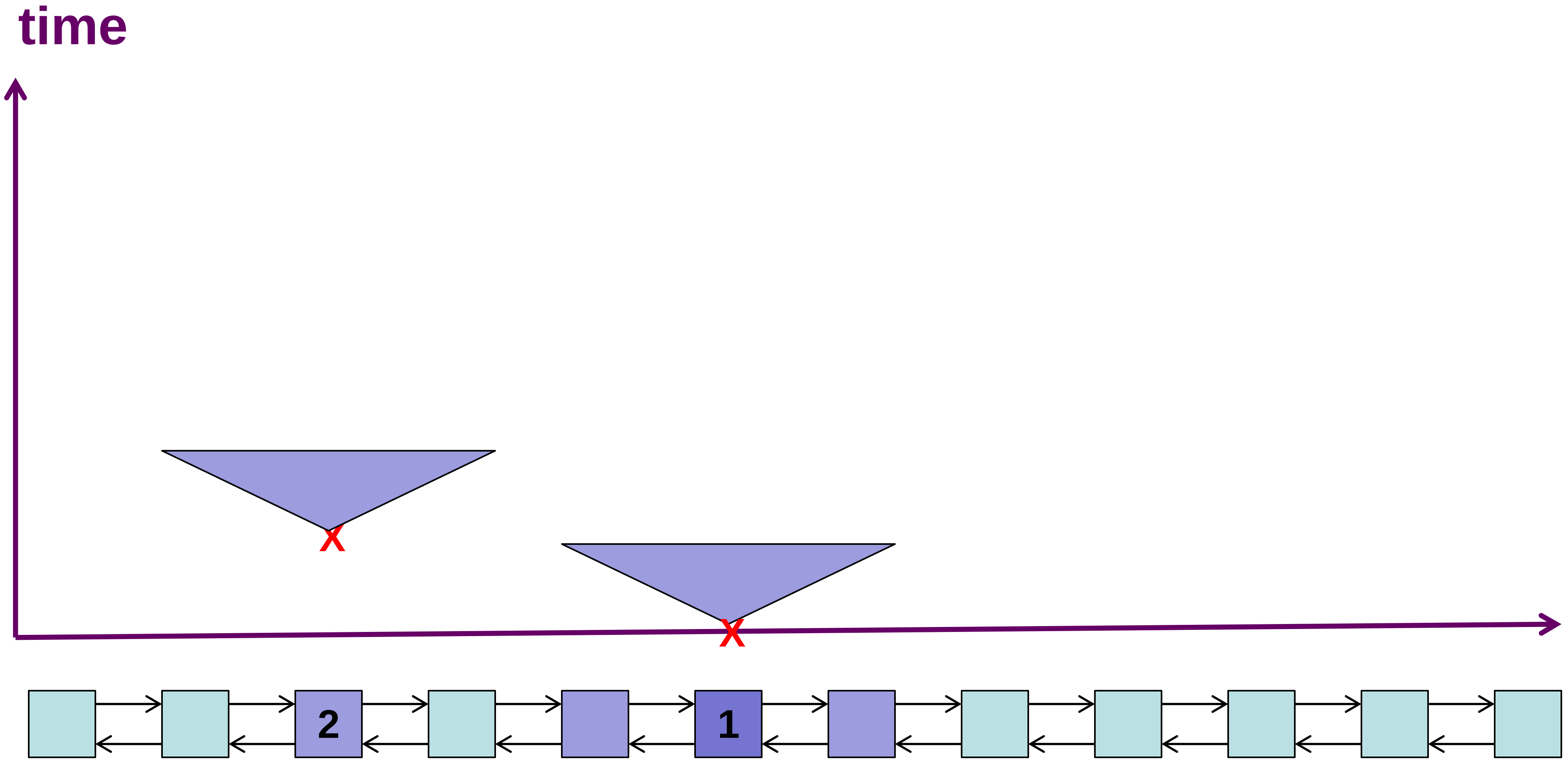
Razor Stall Propagation (1D)



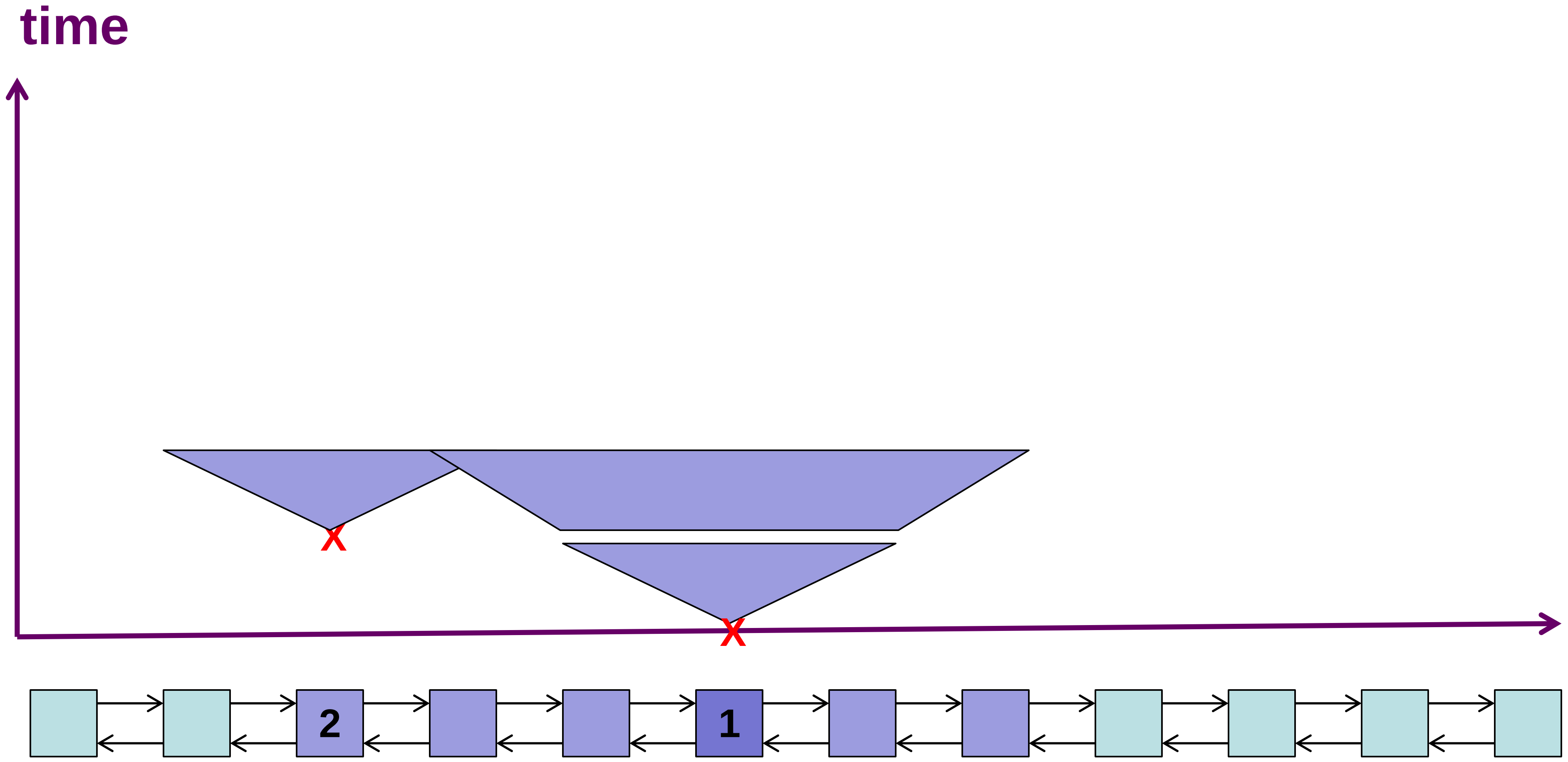
Razor Stall Propagation (1D)



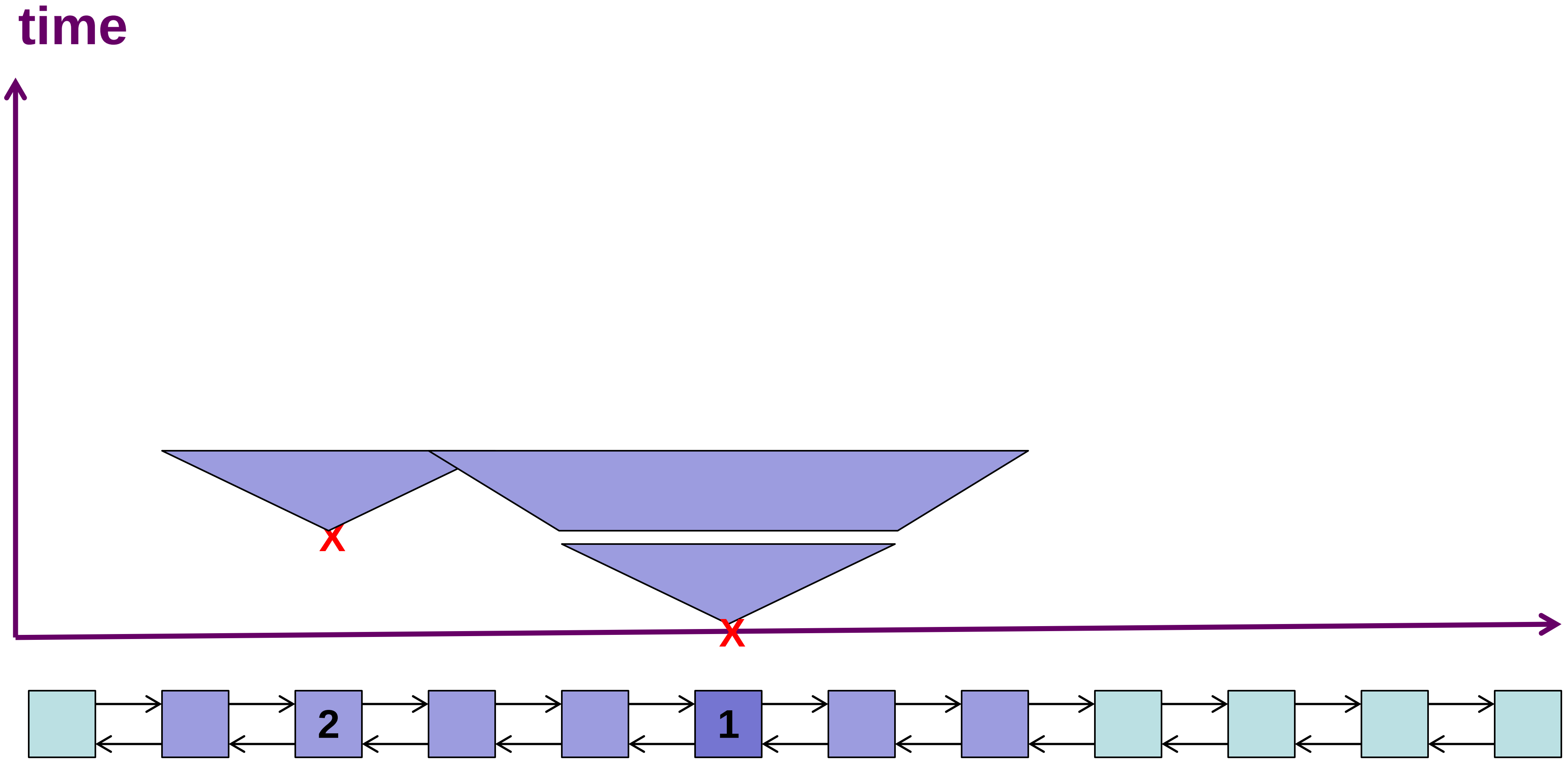
Razor Stall Propagation (1D)



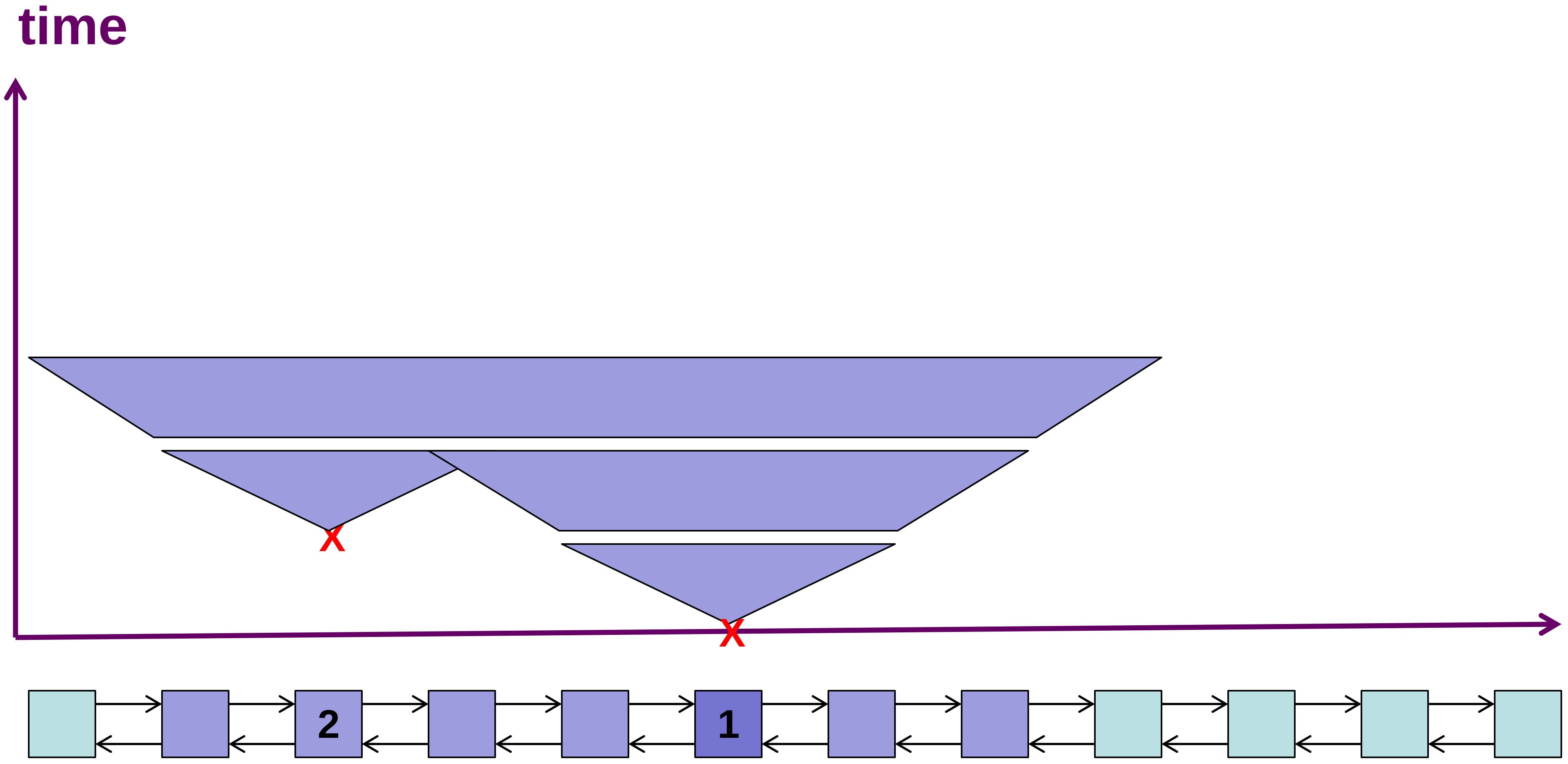
Razor Stall Propagation (1D)



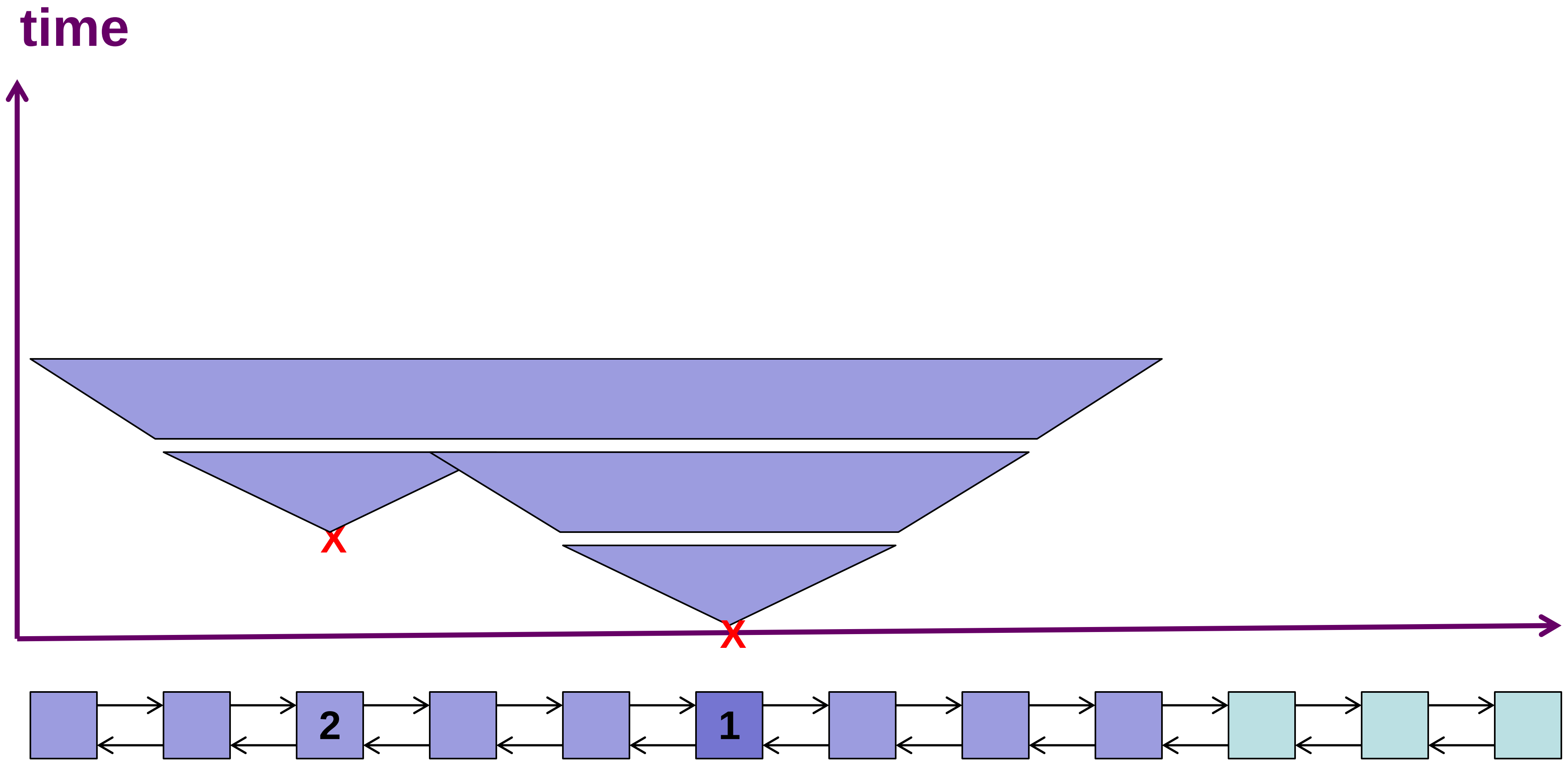
Razor Stall Propagation (1D)



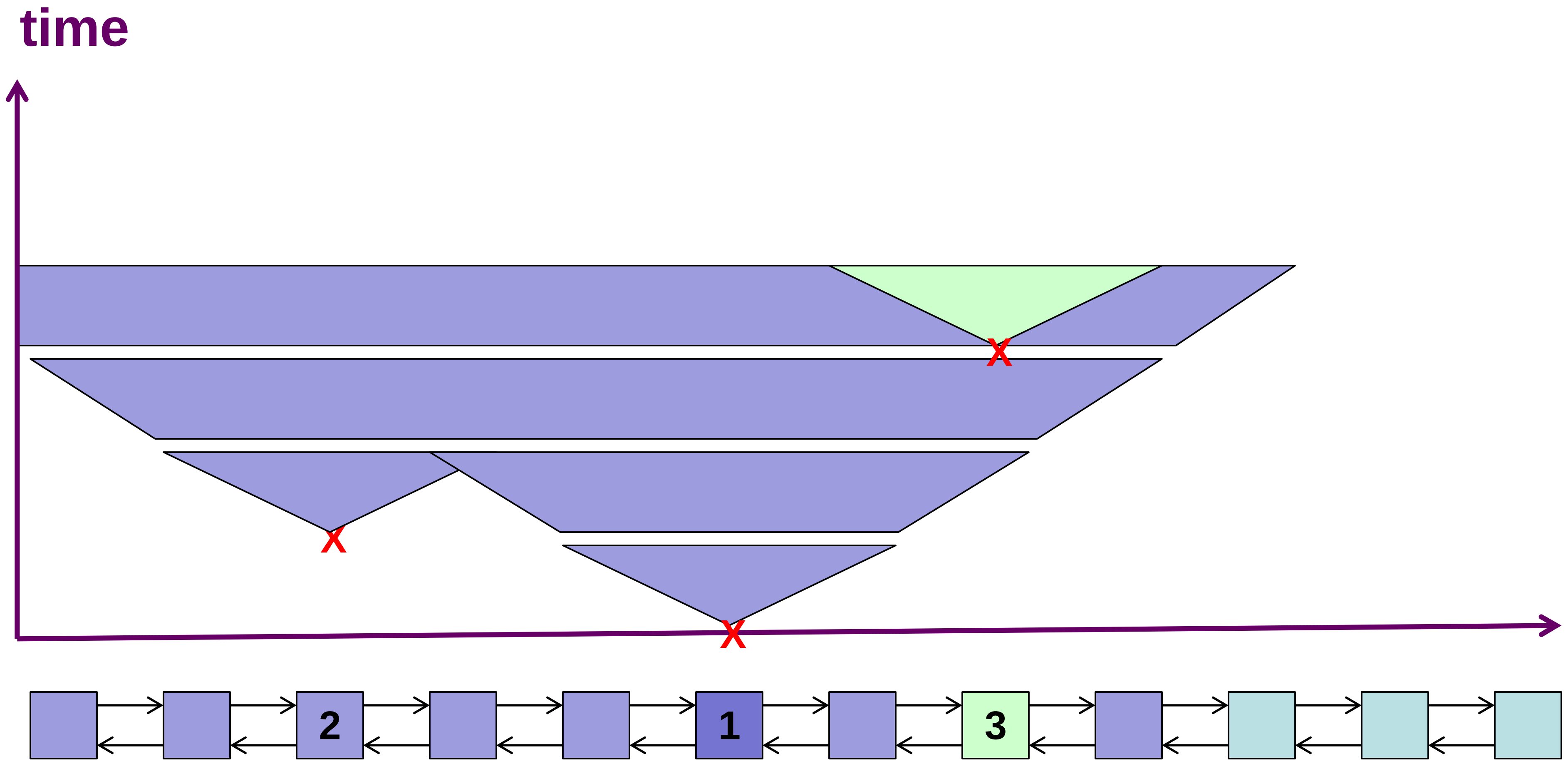
Razor Stall Propagation (1D)



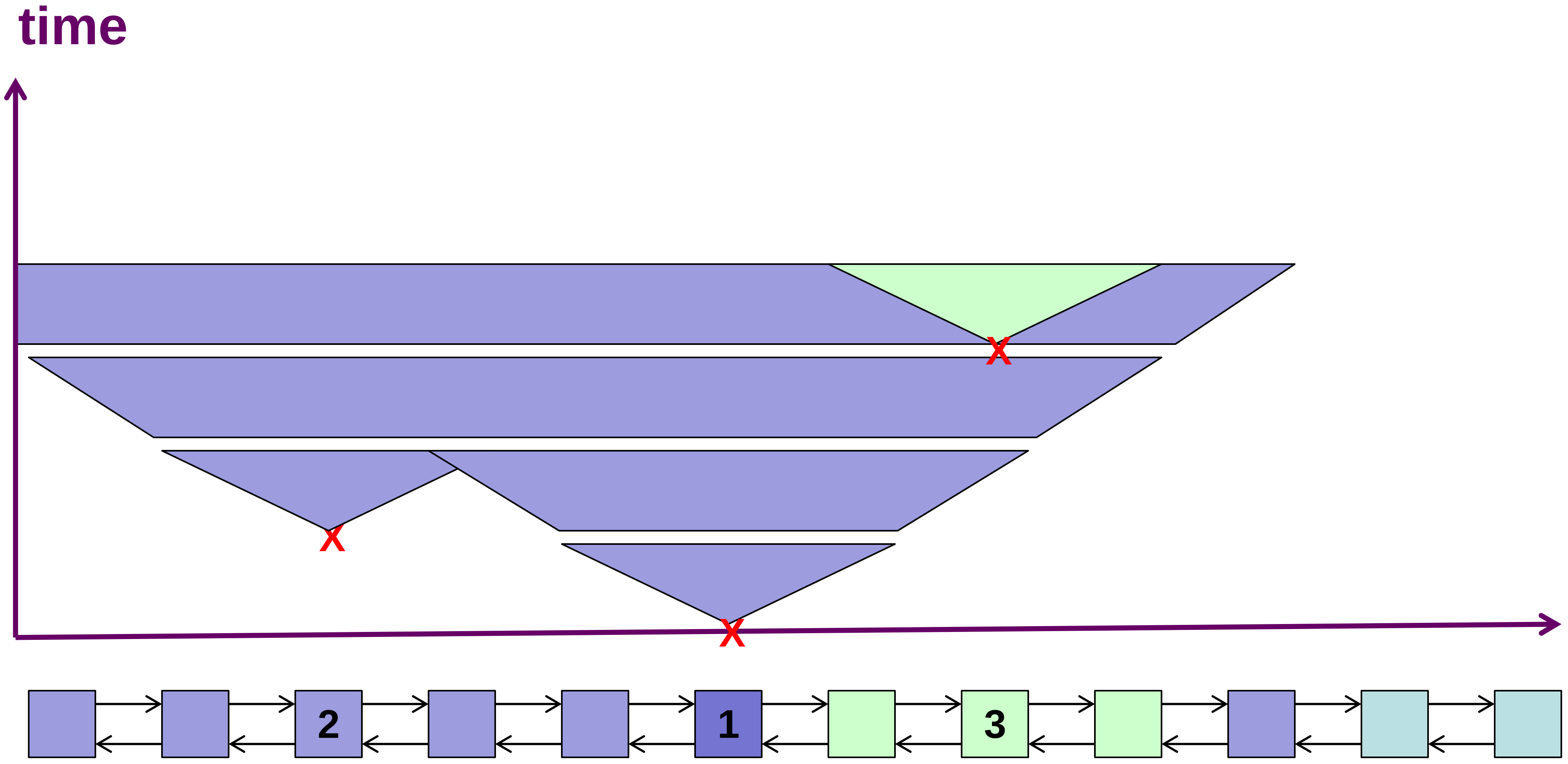
Razor Stall Propagation (1D)



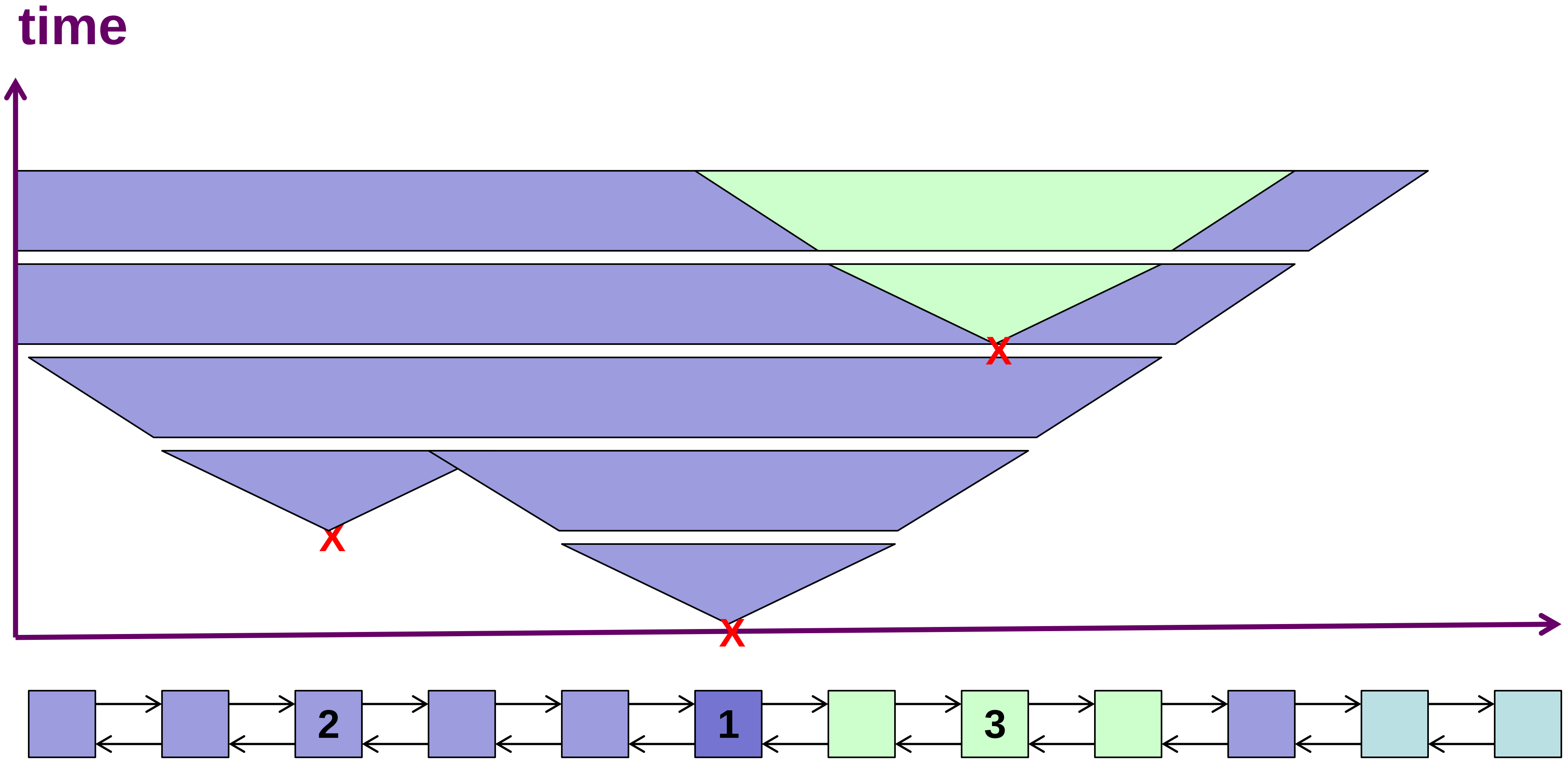
Razor Stall Propagation (1D)



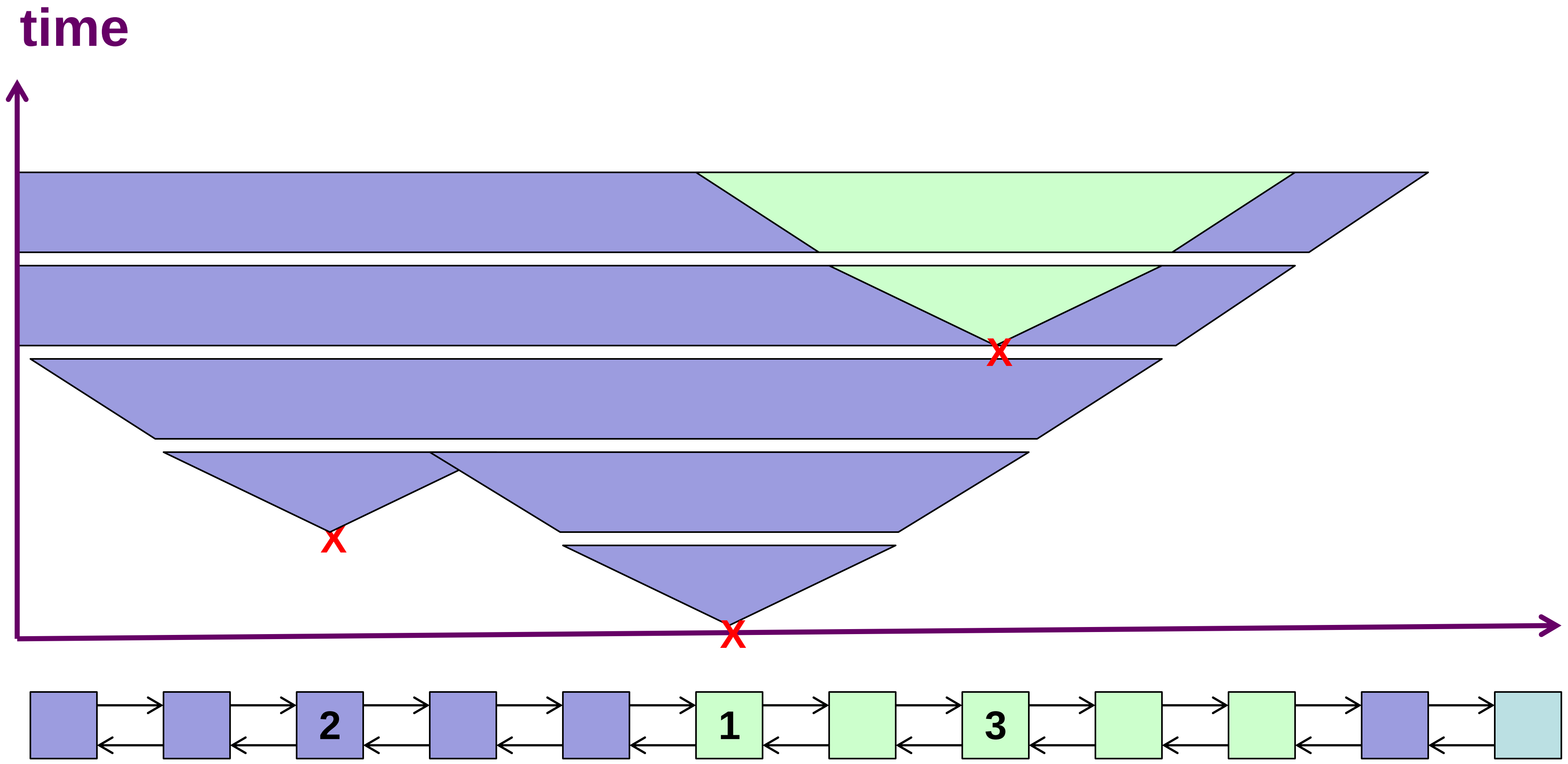
Razor Stall Propagation (1D)



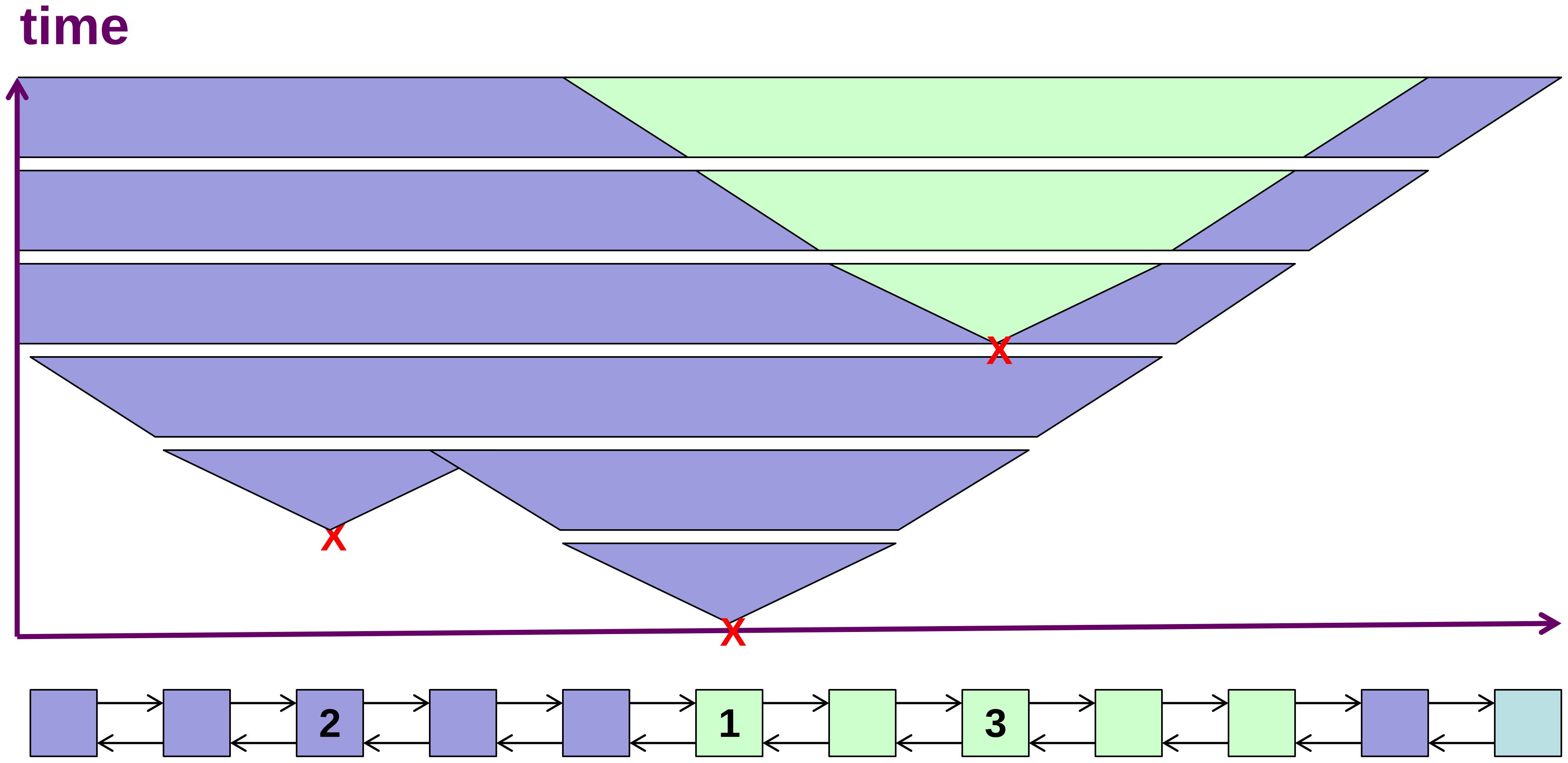
Razor Stall Propagation (1D)



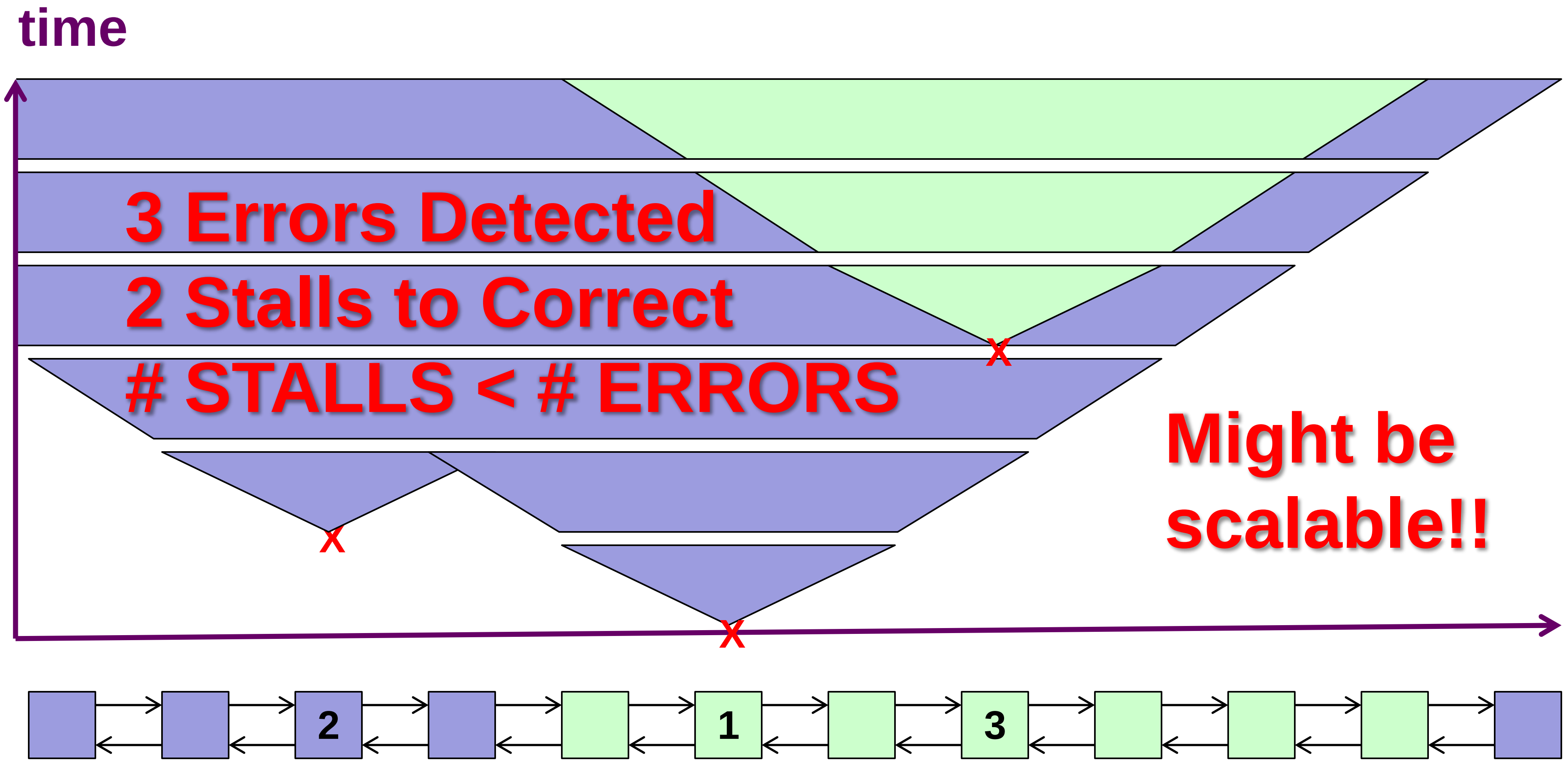
Razor Stall Propagation (1D)



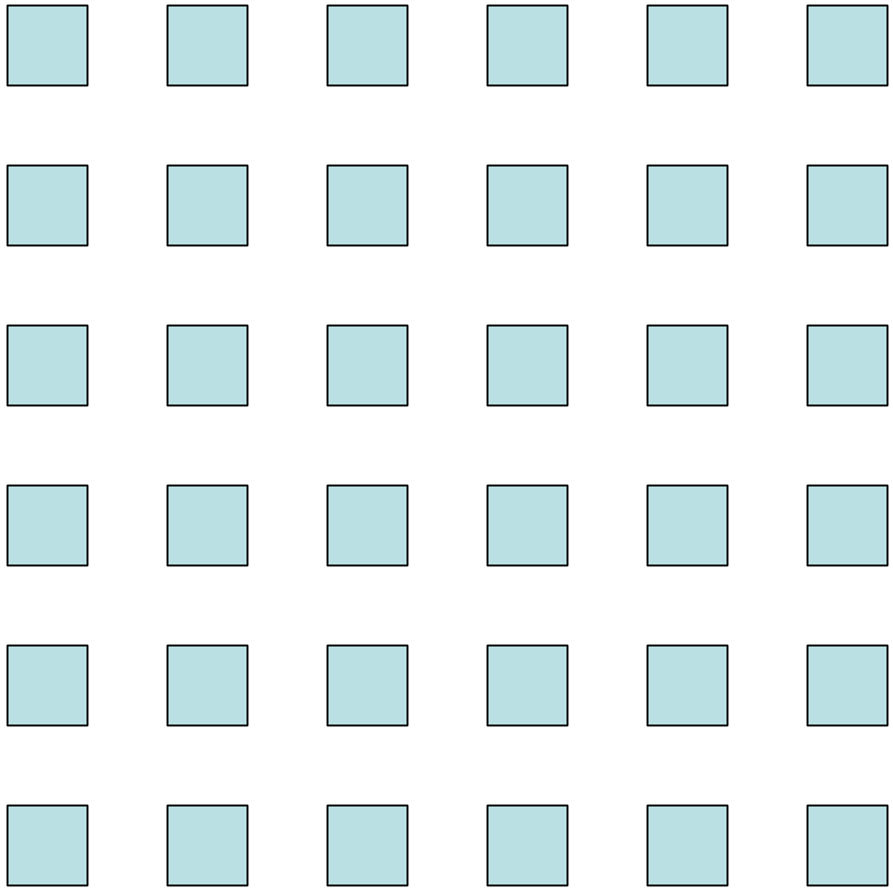
Razor Stall Propagation (1D)



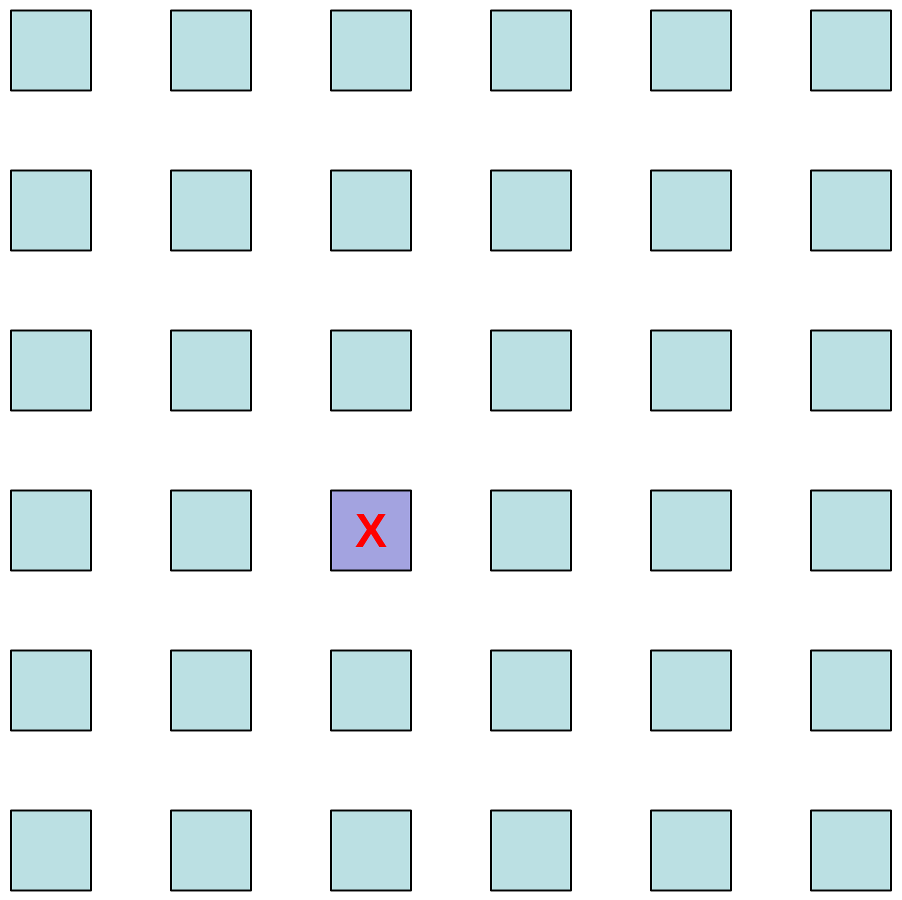
Razor Stall Propagation (1D)



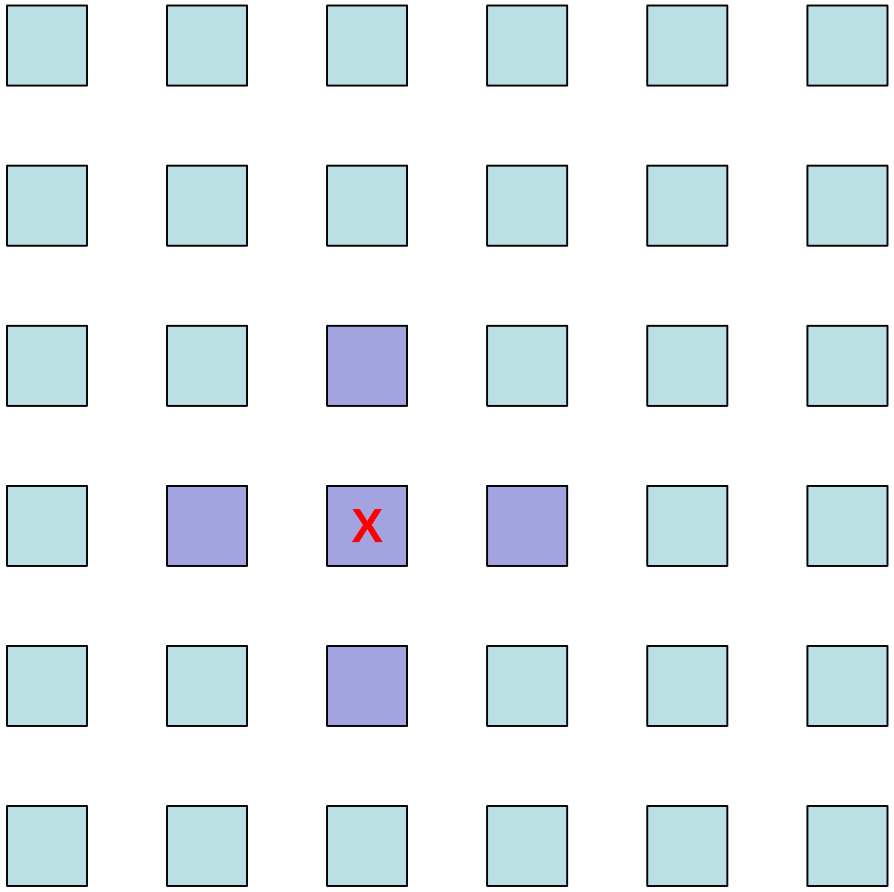
Razor Stall Propagation (2D)



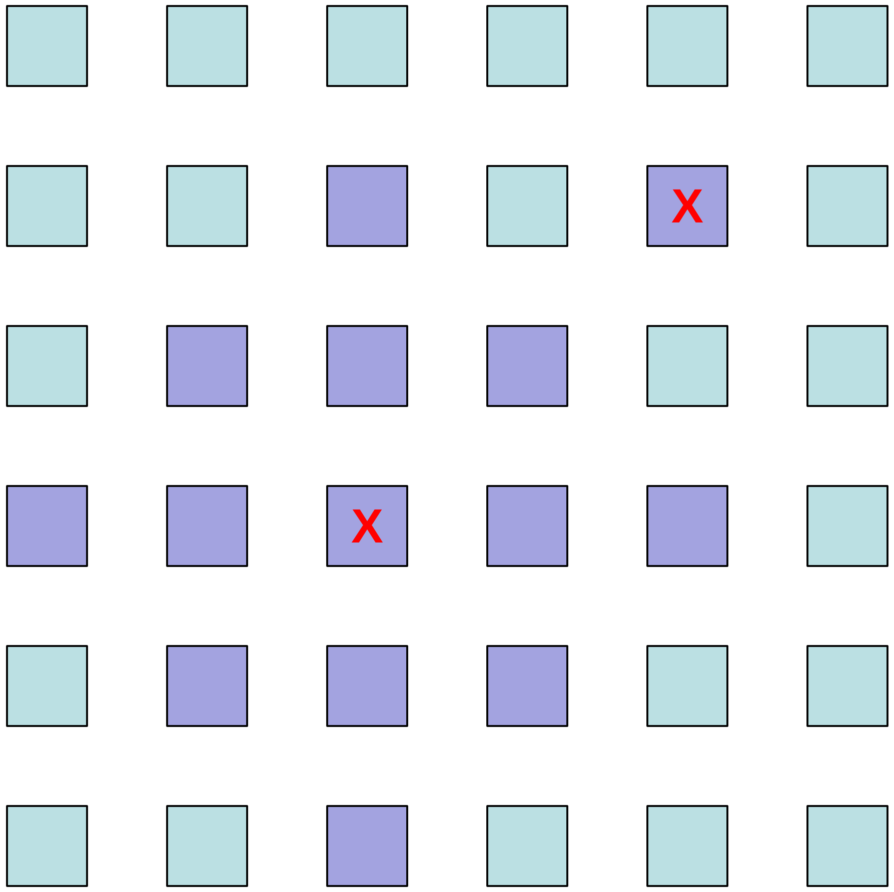
Razor Stall Propagation (2D)



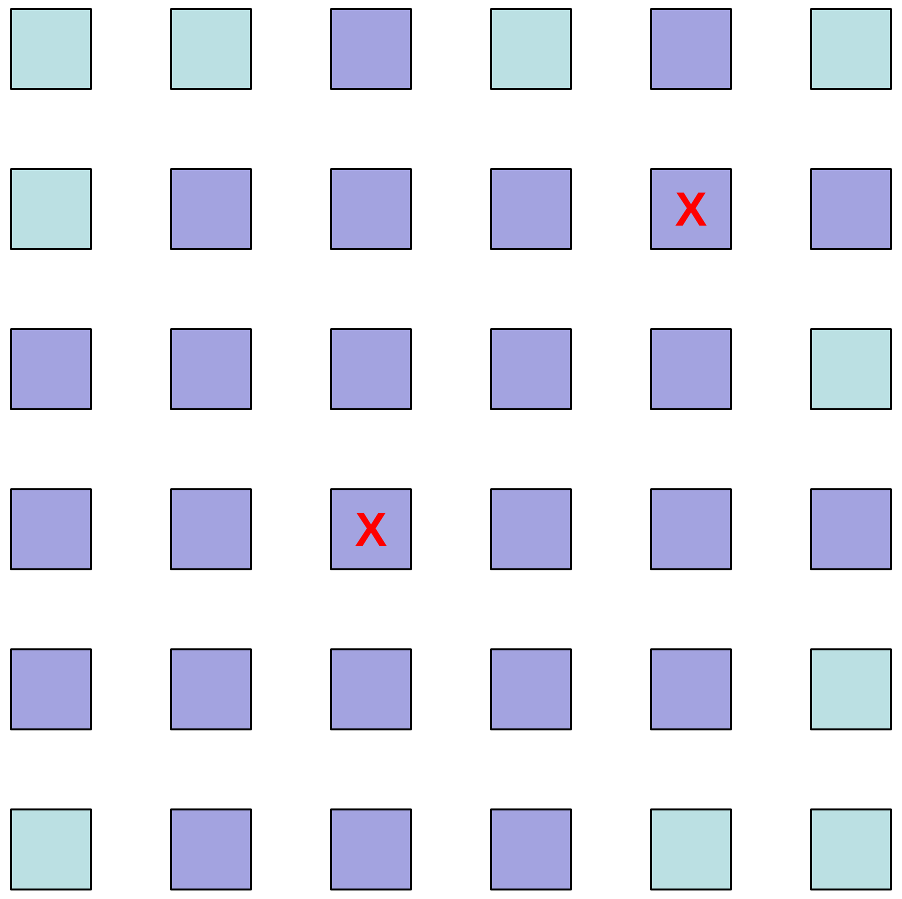
Razor Stall Propagation (2D)



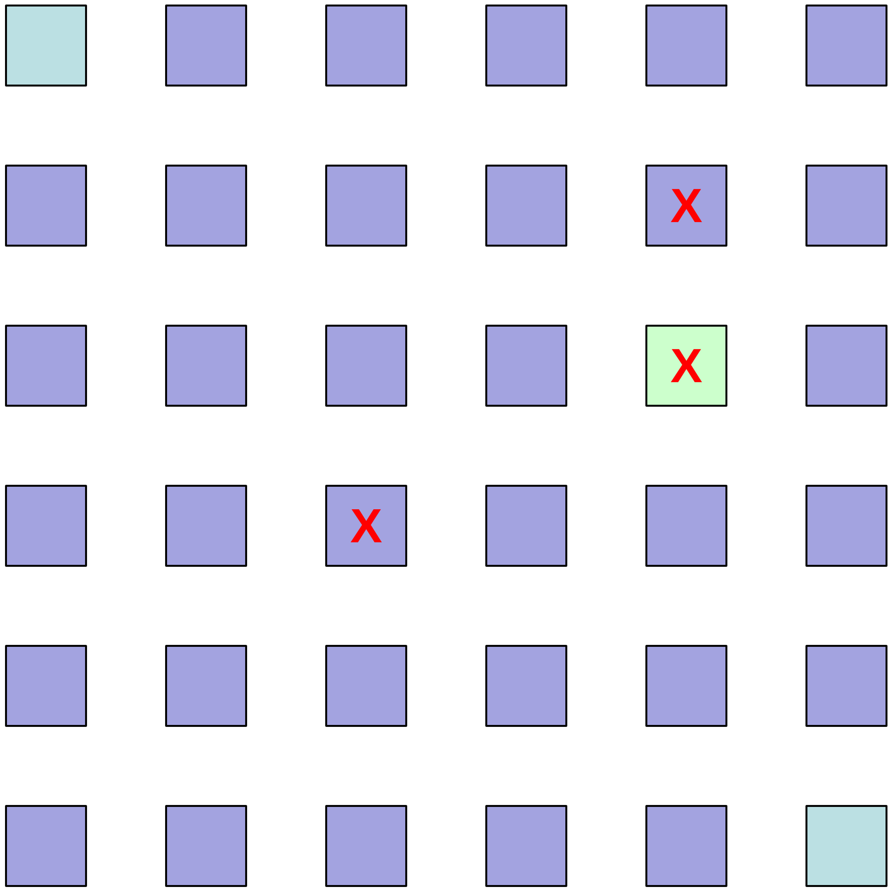
Razor Stall Propagation (2D)



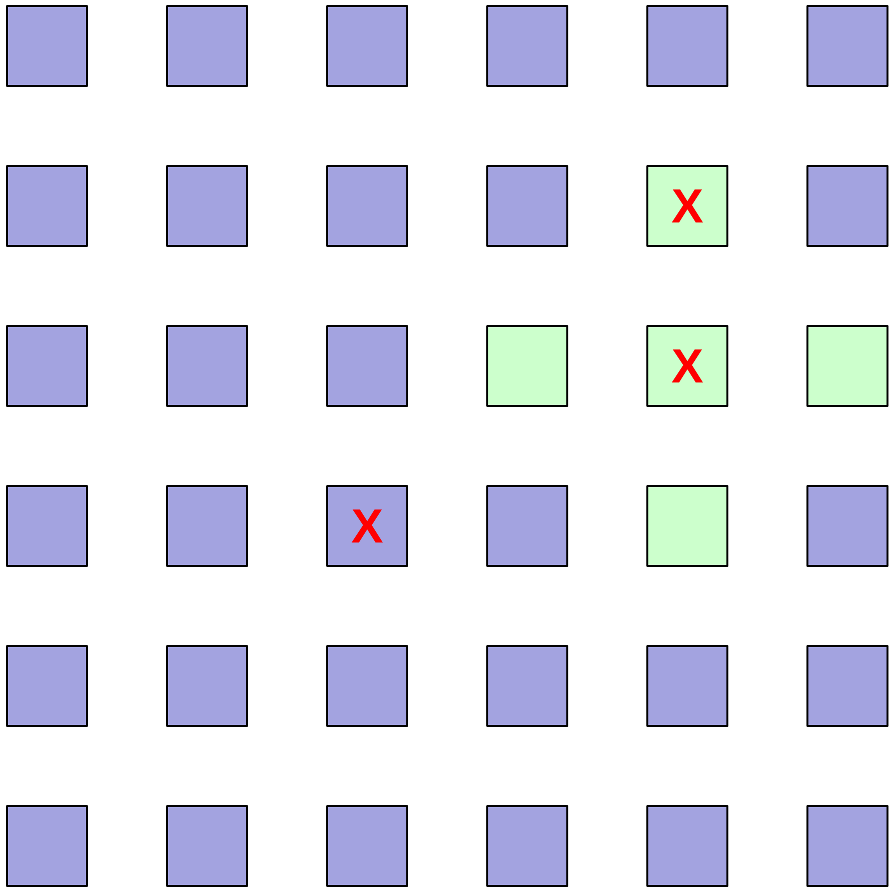
Razor Stall Propagation (2D)



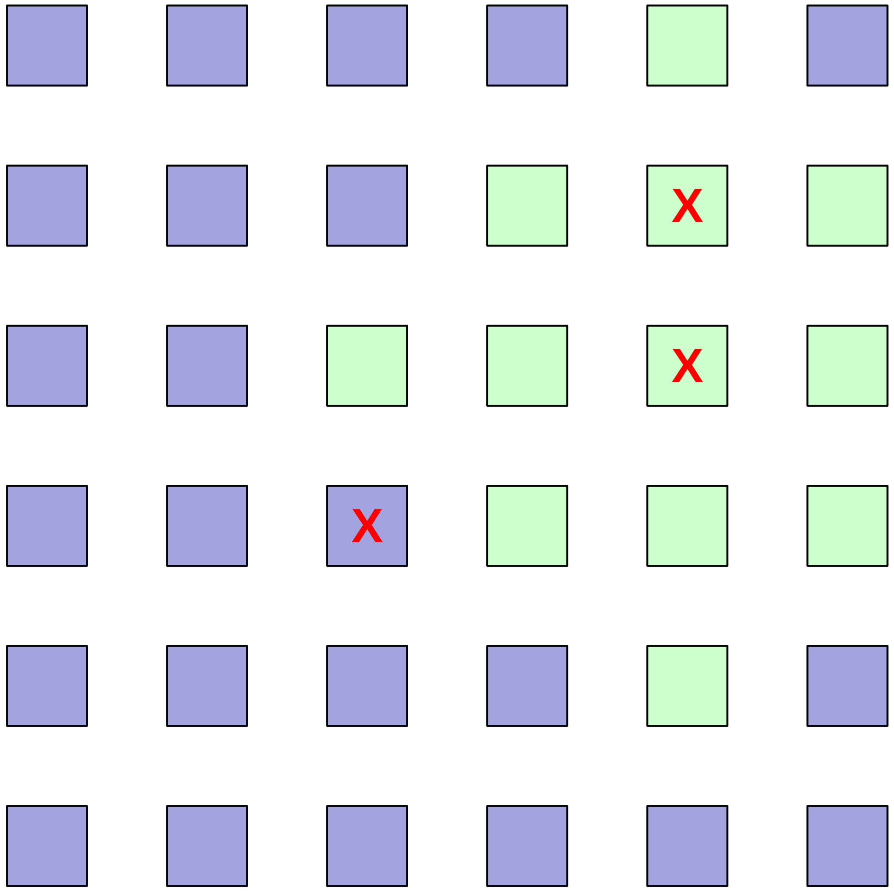
Razor Stall Propagation (2D)



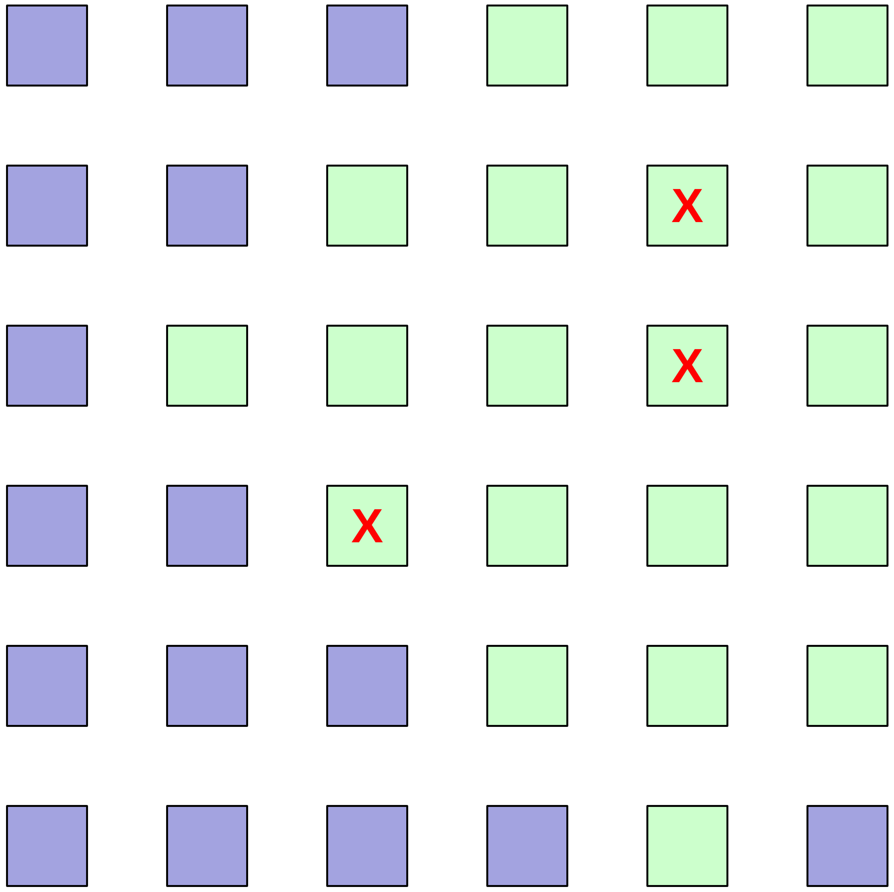
Razor Stall Propagation (2D)



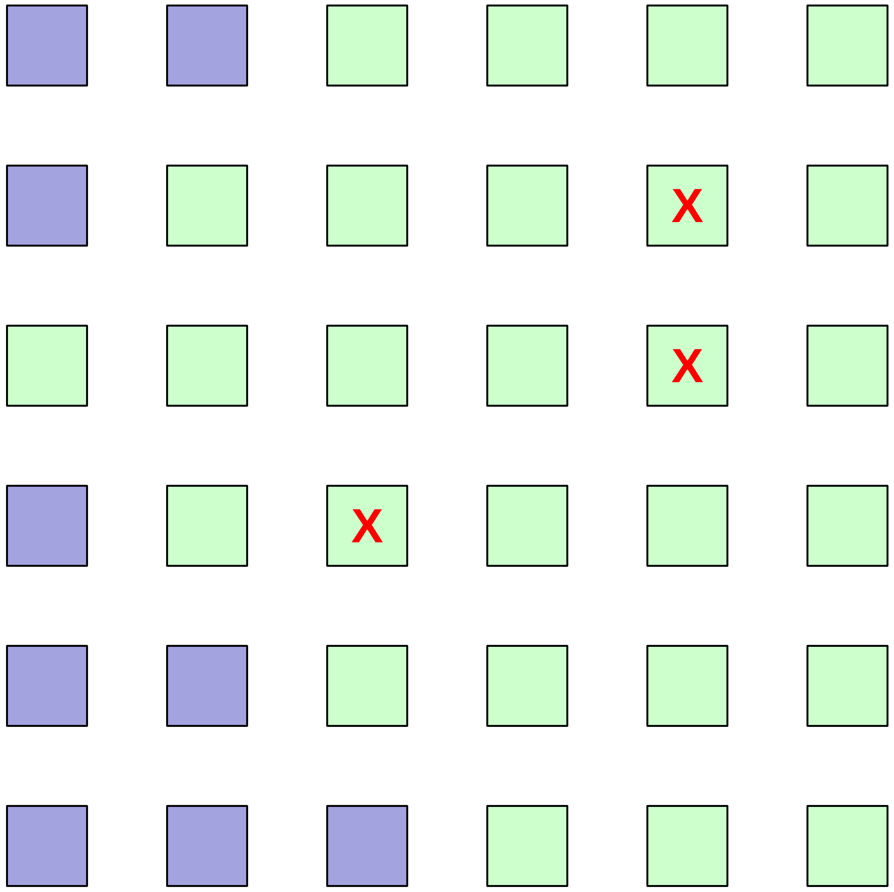
Razor Stall Propagation (2D)



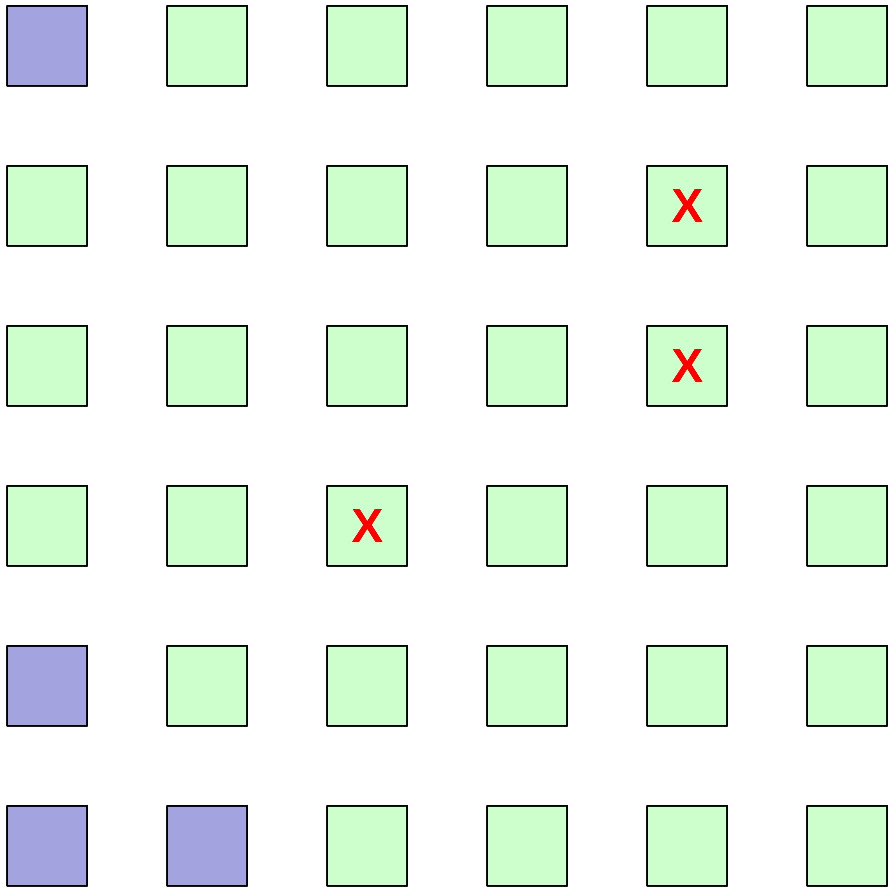
Razor Stall Propagation (2D)



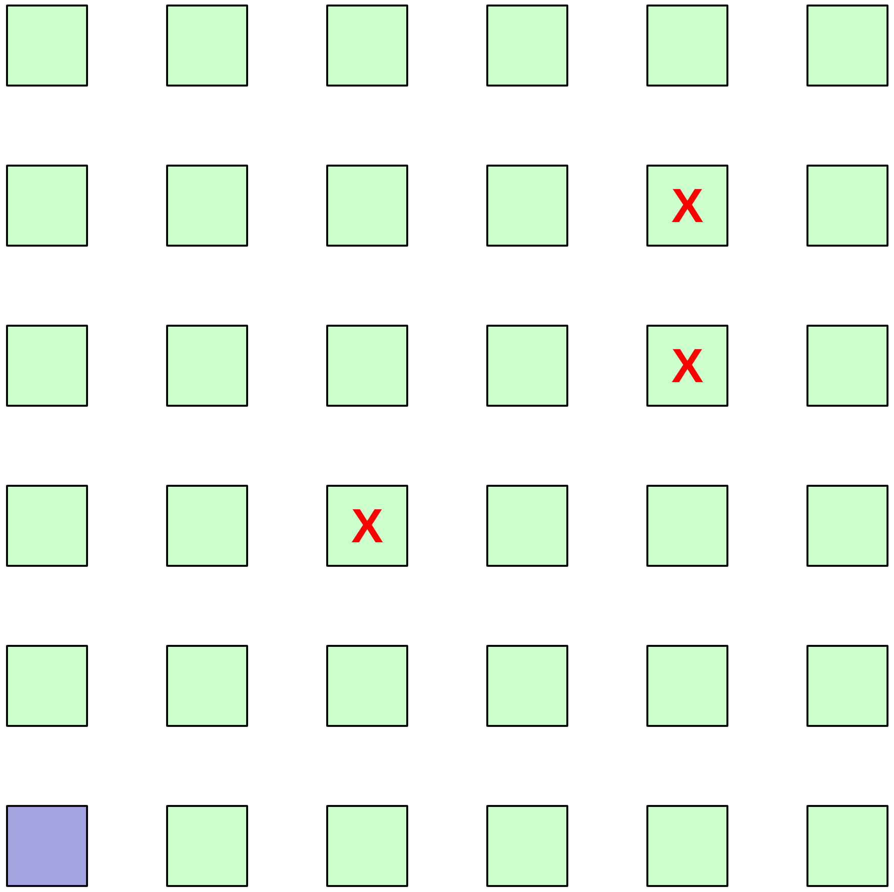
Razor Stall Propagation (2D)



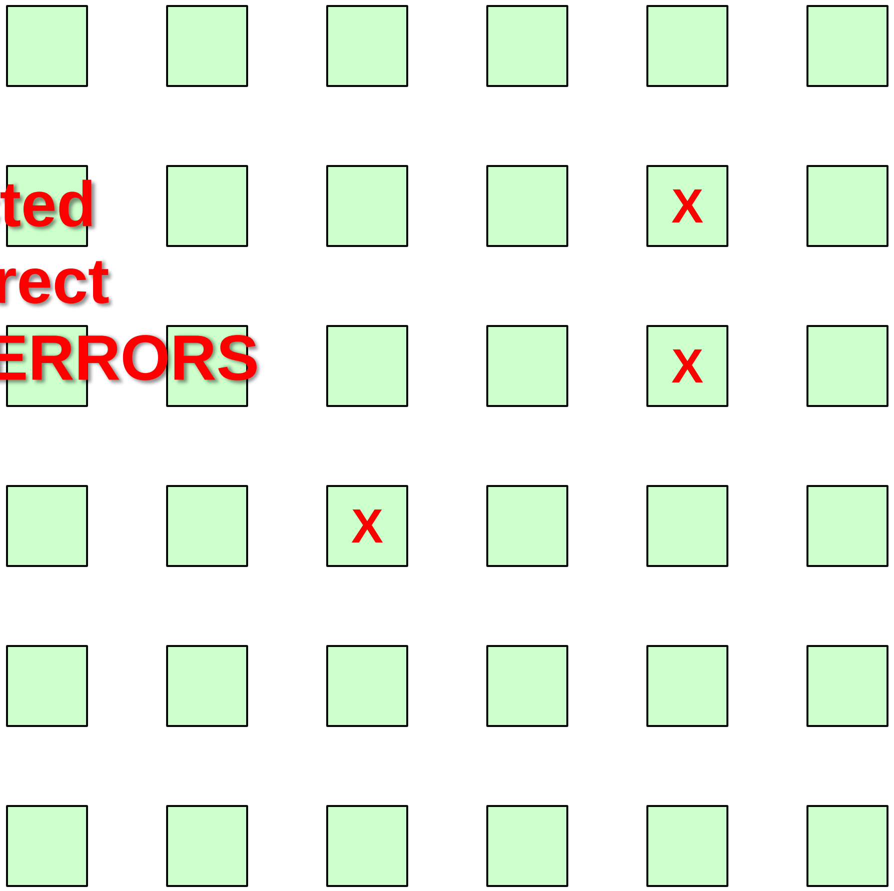
Razor Stall Propagation (2D)



Razor Stall Propagation (2D)



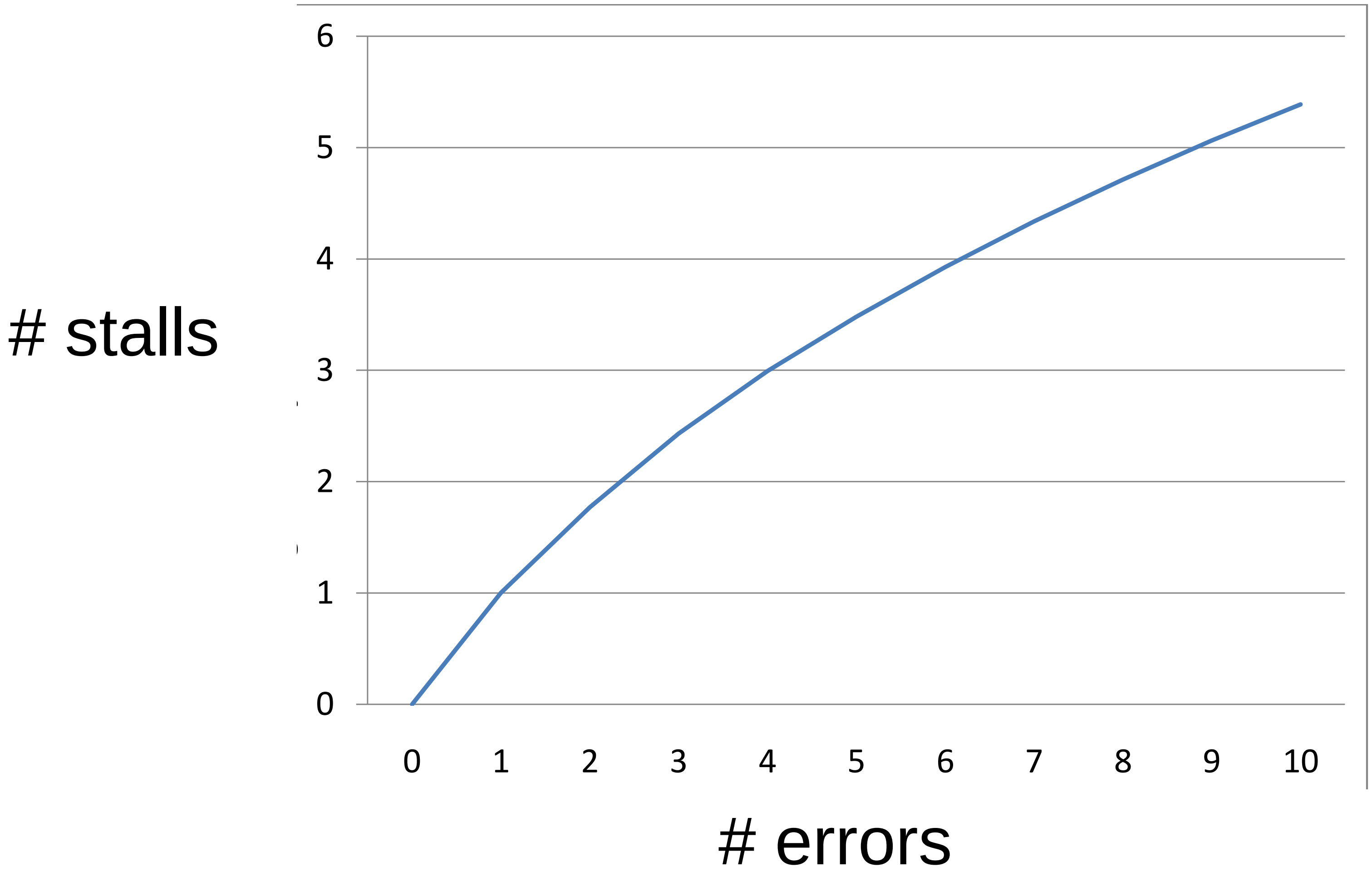
Razor Stall Propagation (2D)



3 Errors Detected
2 Stalls to Correct
# STALLS < # ERRORS

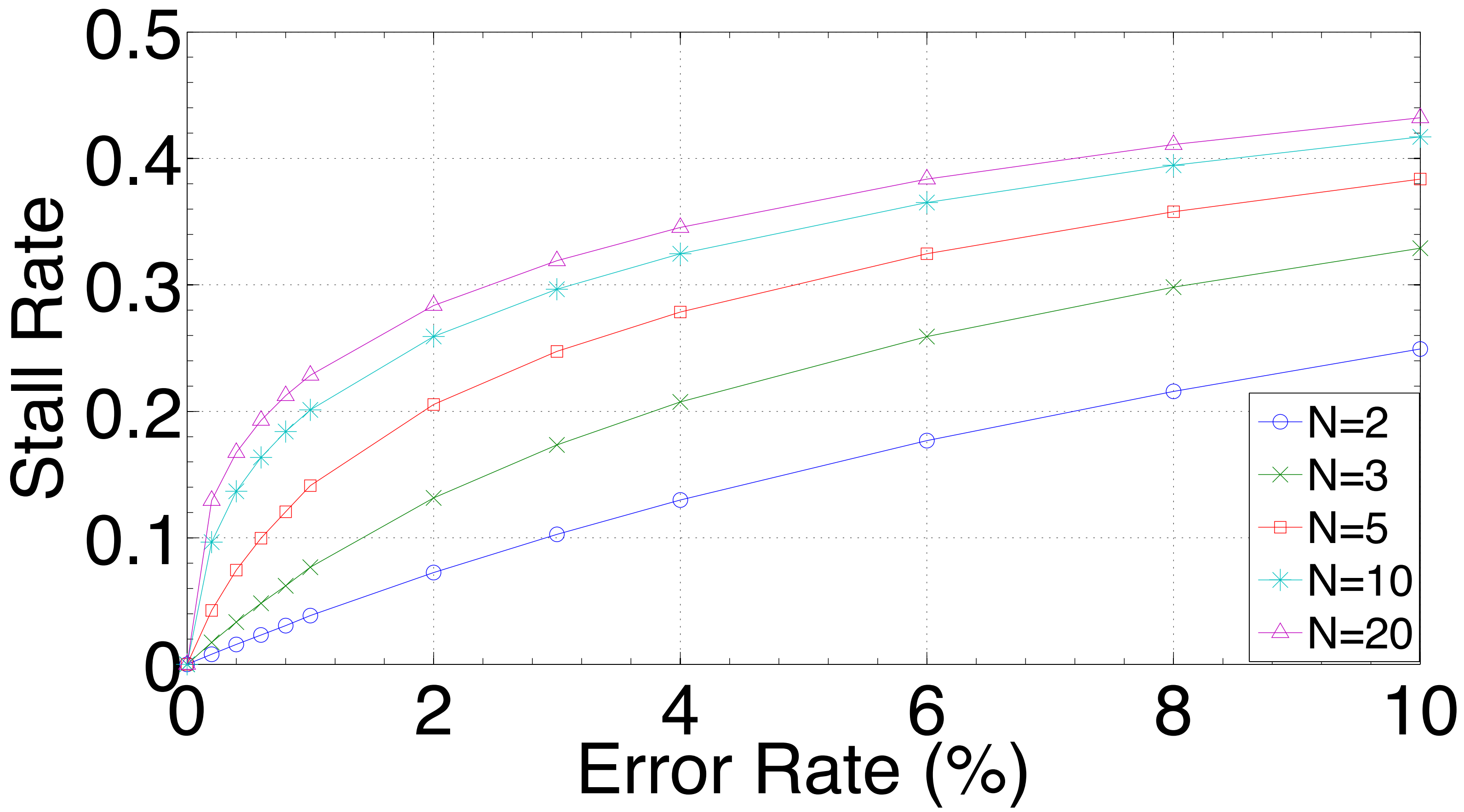
Might be scalable!!

Manual experiment: # stalls vs # errors

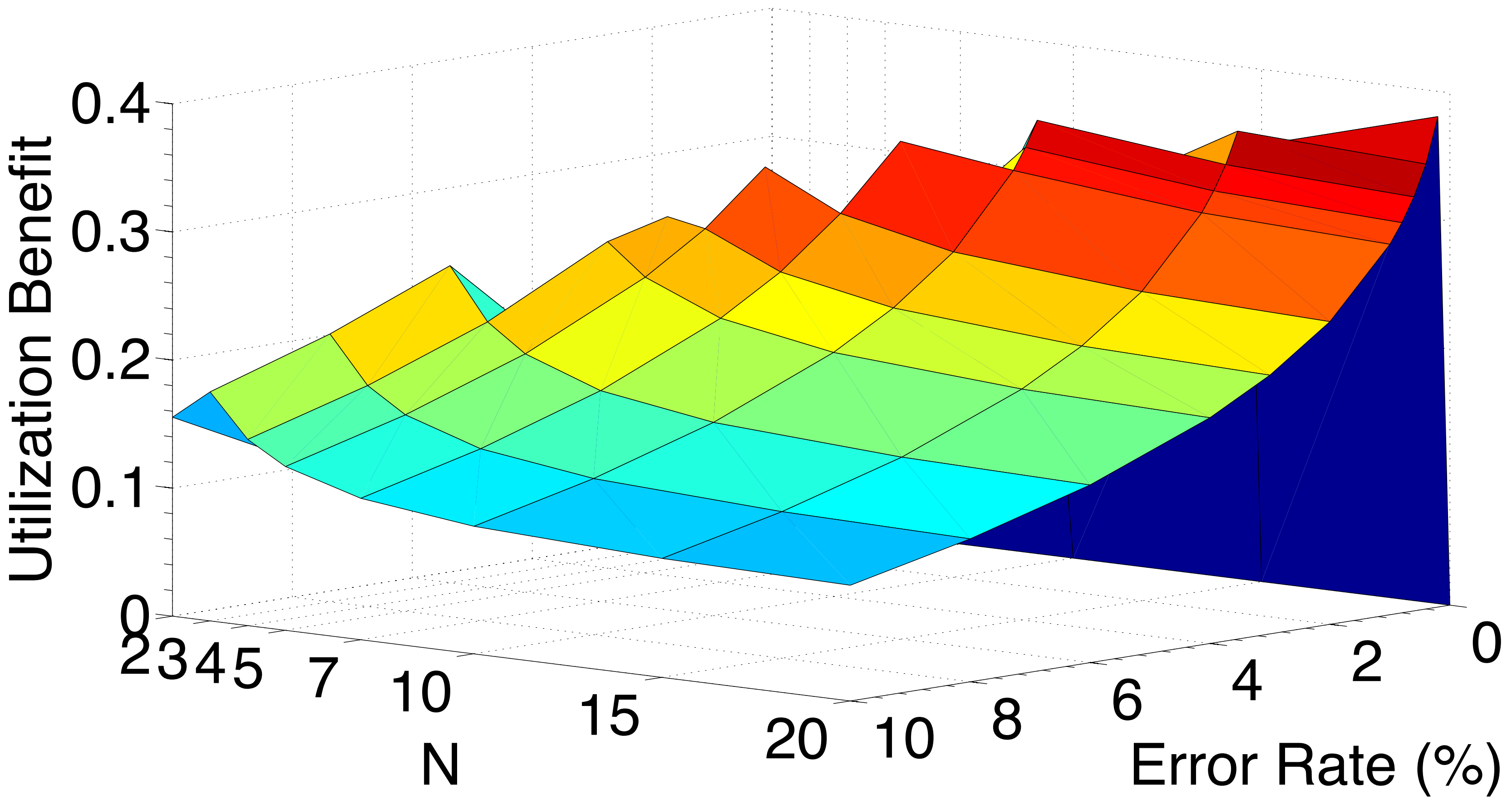


Monte Carlo Simulations…

Stalls vs Errors (N x N array)



Recovered Utilization (N x N)



Experimental Results…

Experimental Results

- Build Processor Array on FPGA…
  - 2 x 2 array in silicon (running)
  - 3 x 3 array in simulation (verification)

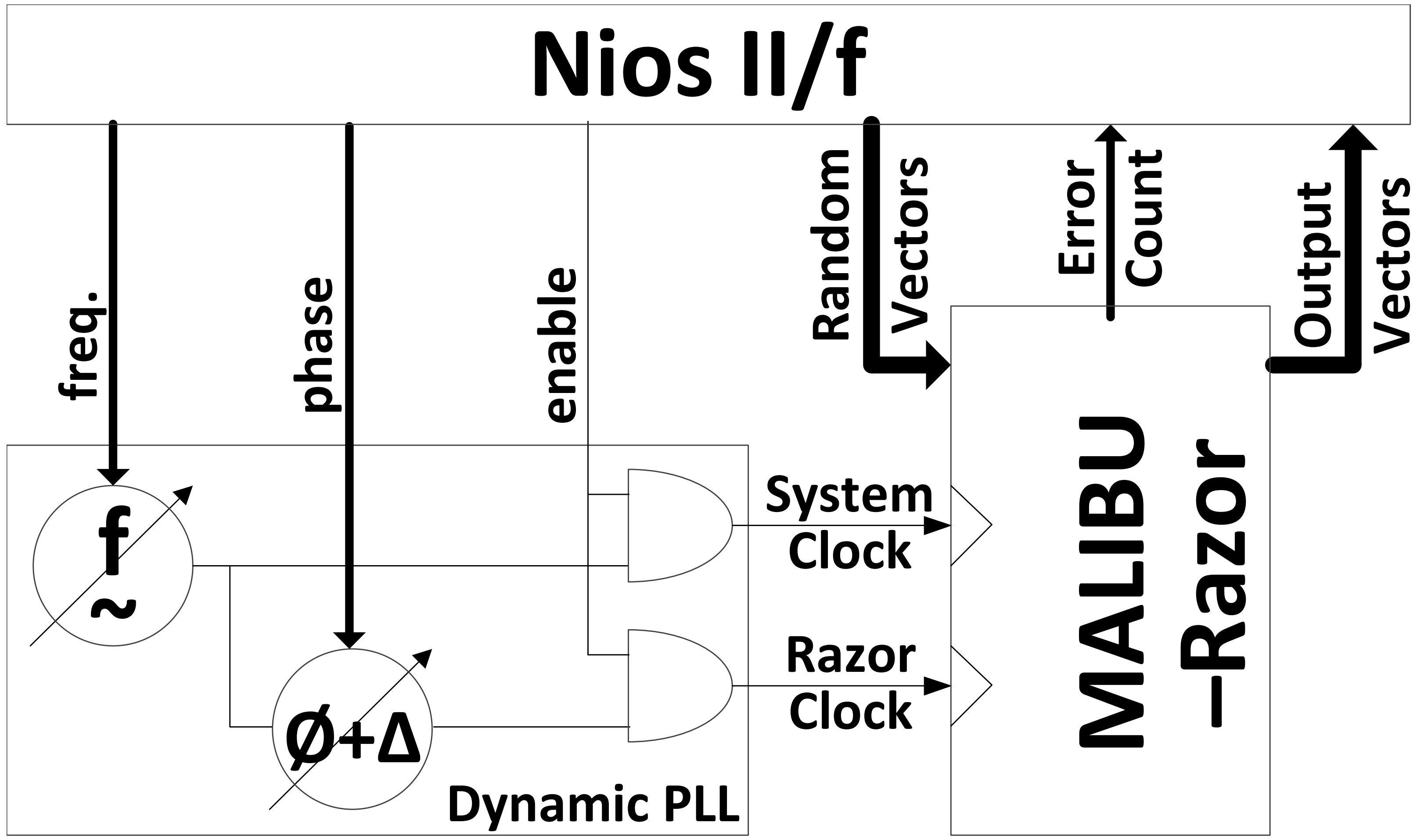
- Static critical path always through ALU + communication channels
  - Typically multipilier, but only if used (!)
  - Depends upon values being multiplied

- Overclock system
  - F_{max} depends upon multiplier use, data values

Place & Route Results

- Area analysis for processor
  - 2,958 ALMs + 304 Regs (baseline)
  - 3,082 ALMs + 517 Regs (with Razor)
- Static timing analysis for array
  - 90 MHz (baseline)
  - 88 MHz (with Razor)
- Overhead is very low (4% ALMs)

System under Test



Methodology (baseline)

- Run once: circuit at low speed
  - Record correct output vectors
- For increasingly higher clock speeds
  - Run circuit with input test vectors
  - Fail on first error
- Remember highest clock speed

Results (baseline)

Benchmark	Static Timing (CAD)	Overclocked (1st error)
Random	90 MHz	135 MHz
Mean	90 MHz	121 MHz
Wang	90 MHz	131 MHz
PR	90 MHz	136 MHz
average	90 MHz	130.4 MHz

- Processor arrays can be overclocked
  - Amount depends on application + data + chip
- But is it safe?
  - Our “test jig” tested results offline to find errors

Methodology (Razor)

- Run once: circuit at low speed
  - Record correct output vectors
- For increasingly higher clock speeds
  - For increasingly higher shadow FF delay
    - Run circuit
    - Record # errors, # corrected errors, # stalls
- Remember highest throughput

Results (baseline)

Benchmark	Static Timing (CAD)	Overclocked (runs past 1st error)	Stall Rate
Random	88 MHz	163 MHz	5.0%
Mean	88 MHz	144 MHz	1.3%
Wang	88 MHz	147 MHz	0.7%
PR	88 MHz	145 MHz	1.7%
average	88 MHz	149.4 MHz	2.0%

- Processor arrays can be overclocked
  - Even higher rates past 1st error
  - Errors require stalls to correct, lowers thru-put
  - Stop increasing Fmax after thru-put peaks

Results (new Razor)

Benchmark	Static Timing (CAD)	Overclocked (runs past 1st error)	Stall Rate
Random	88 MHz	163 MHz	5.0%
Mean	88 MHz	144 MHz	1.3%
Wang	88 MHz	147 MHz	0.7%
PR	88 MHz	145 MHz	1.7%
average	88 MHz	149.4 MHz	2.0%

- But is it safe?
  - Safe! Razor detects and corrects errors
  - Our “test jig” tested results offline to verify the errors were corrected

Results (Comparison)

Benchmark	Baseline STA (safe)	Razor-Corrected Effective Throughput	Speedup
Random	90 MHz	155 MHz	1.72 x
Mean	90 MHz	142 MHz	1.58 x
Wang	90 MHz	146 MHz	1.62 x
PR	90 MHz	143 MHz	1.59 x
average	90 MHz	146.5 MHz	1.63 x

- Processor arrays can be safely overclocked
  - 63% higher throughput
- Low area cost (+4% ALMs)

Observations / Notes

- Time-multiplexed CGRAs/FPGAs can also benefit
  - Just reserve 1-2 clock cycles in the time-mux schedule for error recovery
- Loosely coupled processor arrays can be overclocked locally
  - Just add Razor to each processor
  - No need to propagate stall signals; automatically done through data presence indicators

Summary / Conclusions

- Processor arrays can be safely overclocked
  - Even with very tightly scheduled communication
- Processor arrays are scalable
  - Errors produce stall wavefronts
  - Several wavefronts merge into a single stall cycle
- Throughput increased 63% on average
  - Speedup depends upon benchmark, data values