MOUSETRAP: Designing High-Speed Asynchronous Digital Pipelines

Montek Singh

Univ. of North Carolina, Chapel Hill, NC
(formerly at Columbia University)

Steven M. Nowick

Columbia University, New York, NY

M. Singh and S.M. Nowick,
"MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines."
IEEE Trans. on VLSI Systems, vol. 15:6, pp. 684-698 (June 2007)
[also in ICCD 2001]

Contribution

New pipeline style that is:

- *asynchronous:* avoids problems of high-speed global clock
- *very high-speed:* FIFOs up to 2.10-2.38 GHz (0.18µ CMOS)
- *naturally elastic:* hold dynamically-variable # of data items
- *uses simple local timing constraints :* one-sided
- *robustly support variable-speed environments*
- *well-matched for fine-grain datapaths*

Outline

- MOUSETRAP: A New Asynchronous Pipeline
  - Basic FIFO
  - Pipeline with Logic Processing
  - Handling Forks and Joins
- Performance, Timing Analysis and Optimization
- Results
- Related Work
- Conclusions and Ongoing Work

MOUSETRAP Pipelines

Simple asynchronous implementation style, uses…

- *level-sensitive D-latches (not flipflops)*
- *simple stage controller:* 1 gate/pipeline stage
- *single-rail bundled data:* synchronous style logic blocks (1 wire/bit, with matched delay)

Target = static logic blocks

Goal: very fast cycle time

- simple inter-stage communication

MOUSETRAP Pipelines

Uses a *"capture-pass protocol"*

Latches …:

- normally transparent: *before* new data arrives
- become opaque: *after* data arrives (= "capture" data)

Control Signaling: *"transition-signaling" = 2-phase*

- simple req/ack protocol = only 2 events per handshake (not 4)
- *no* "return-to-zero"
- each transition (up/down) signals a distinct operation

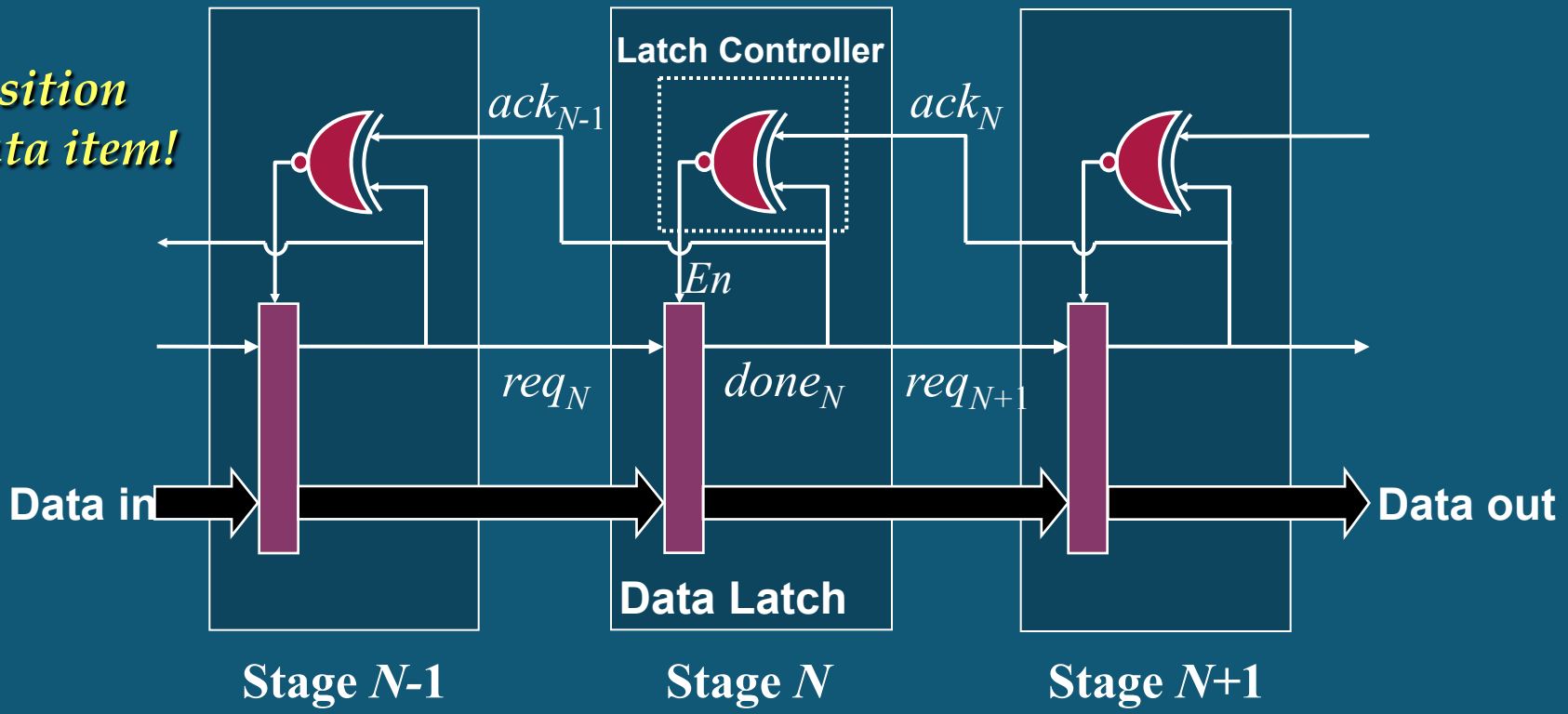
Data Encoding: *"single-rail bundled data"*

- Data: use normal single-rail data transmitted -- same as in synchronous!
  - glitches allowed on data wires
- Plus 1 added bit = "bundling signal" (i.e. 'req'):
  - sent just after data valid/stable: single transition (hazard-free) acts as a "strobe"

for more background see: S.M. Nowick and M. Singh, "High-Performance Asynchronous Pipelines: an Overview," IEEE Design & Test of Computers, v. 28:5, pp. 8-22 (Sept./Oct. 2011)

MOUSETRAP: A Basic FIFO

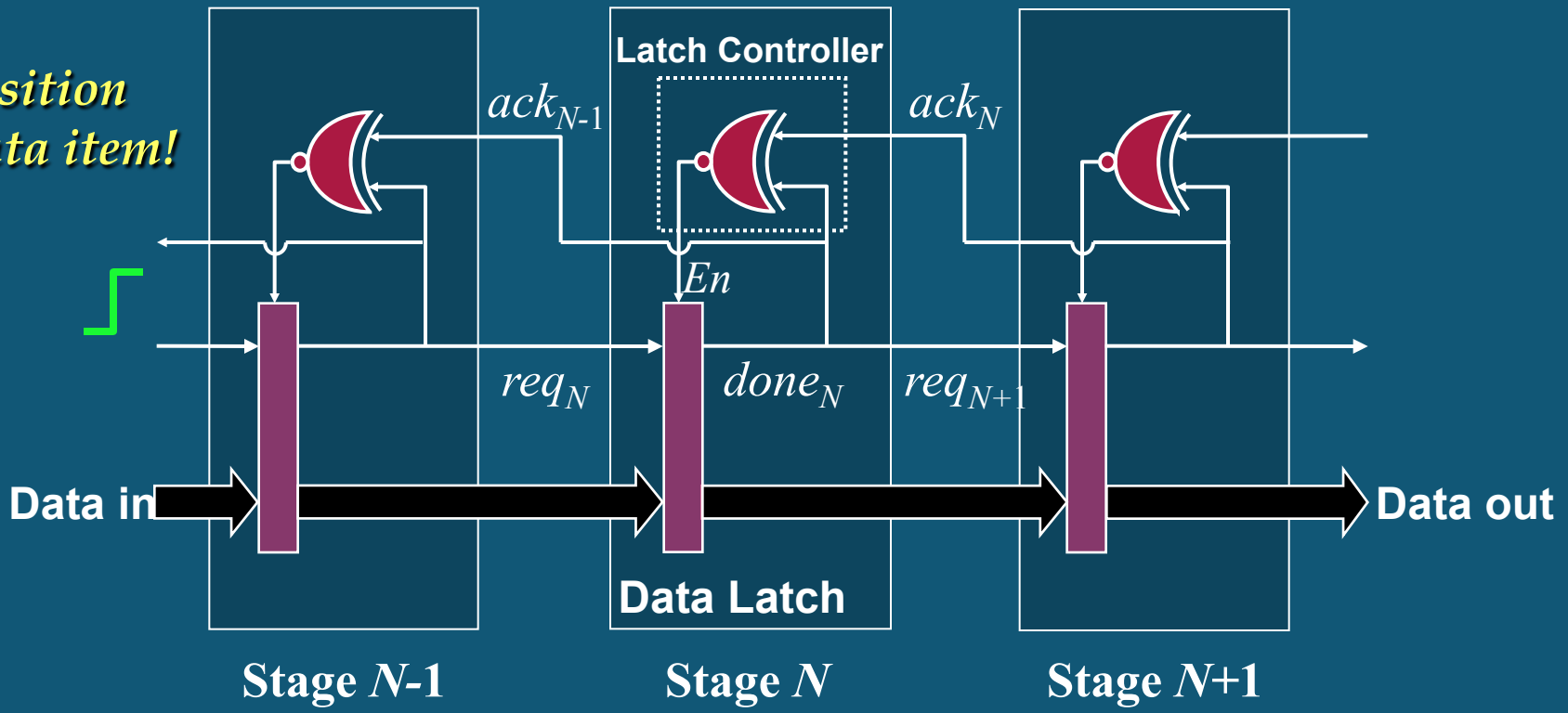
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

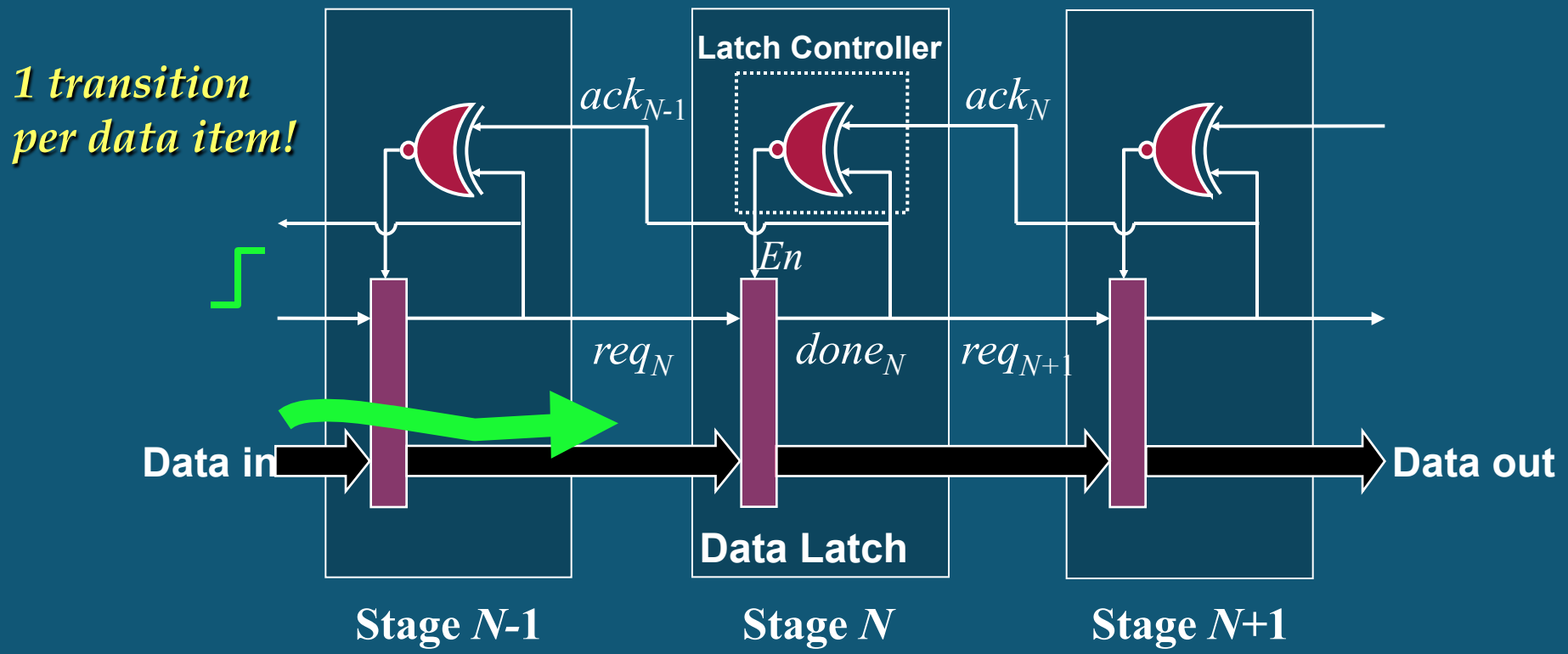
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

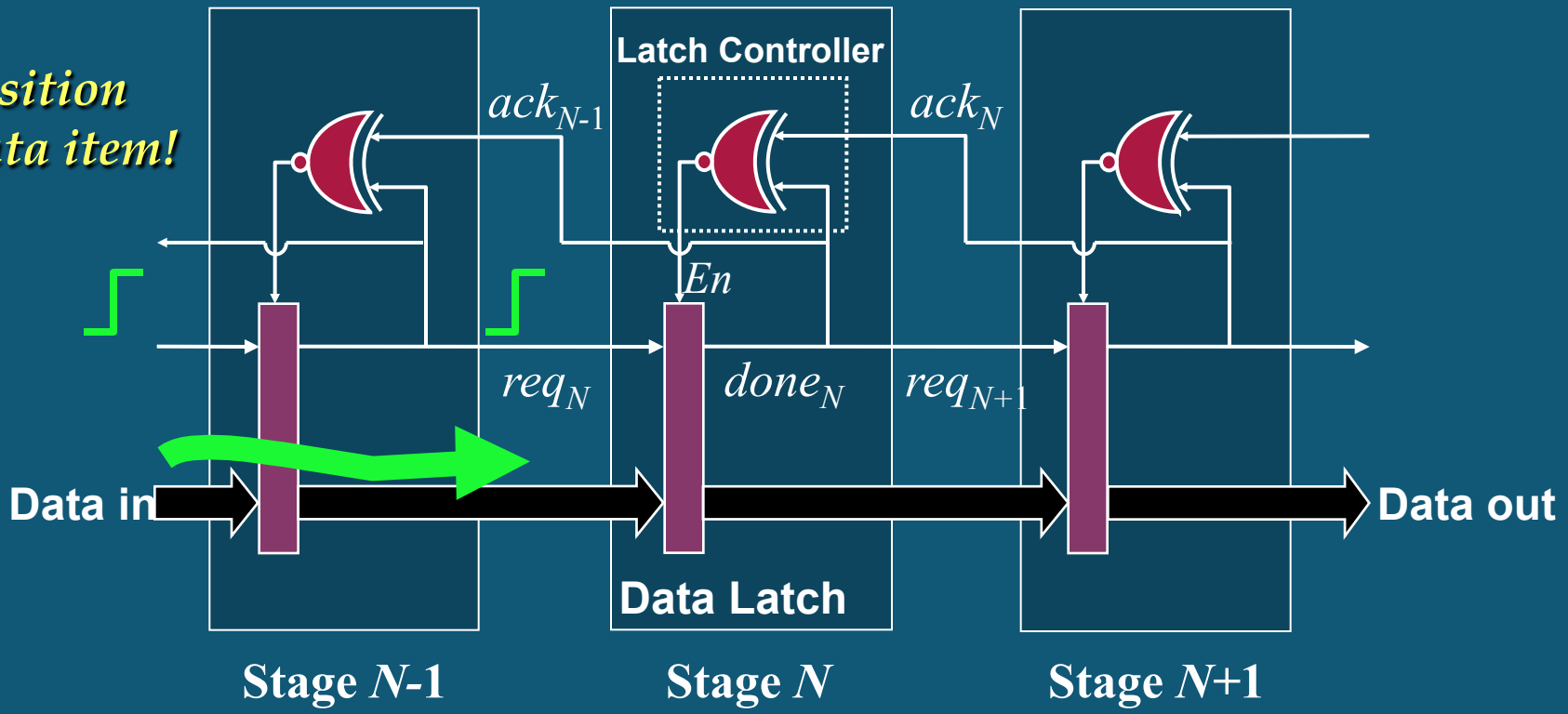
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

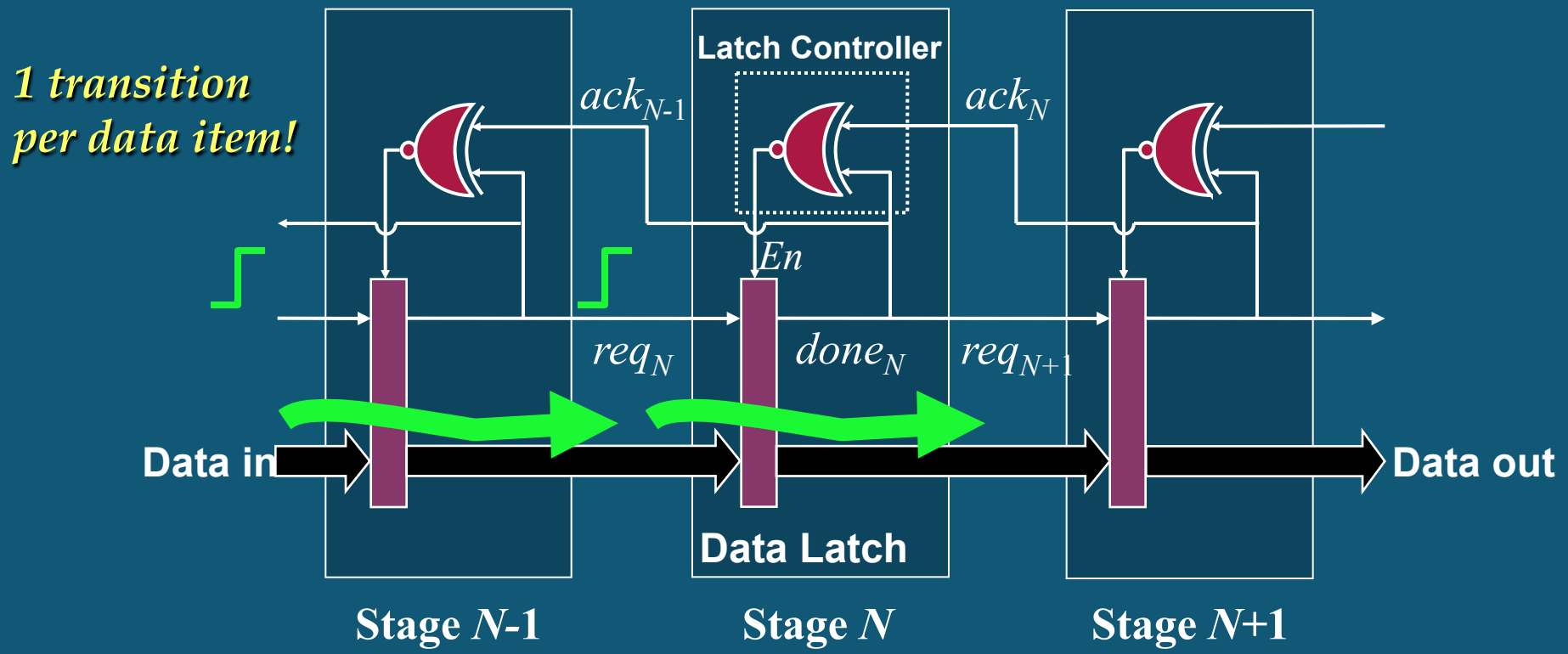
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

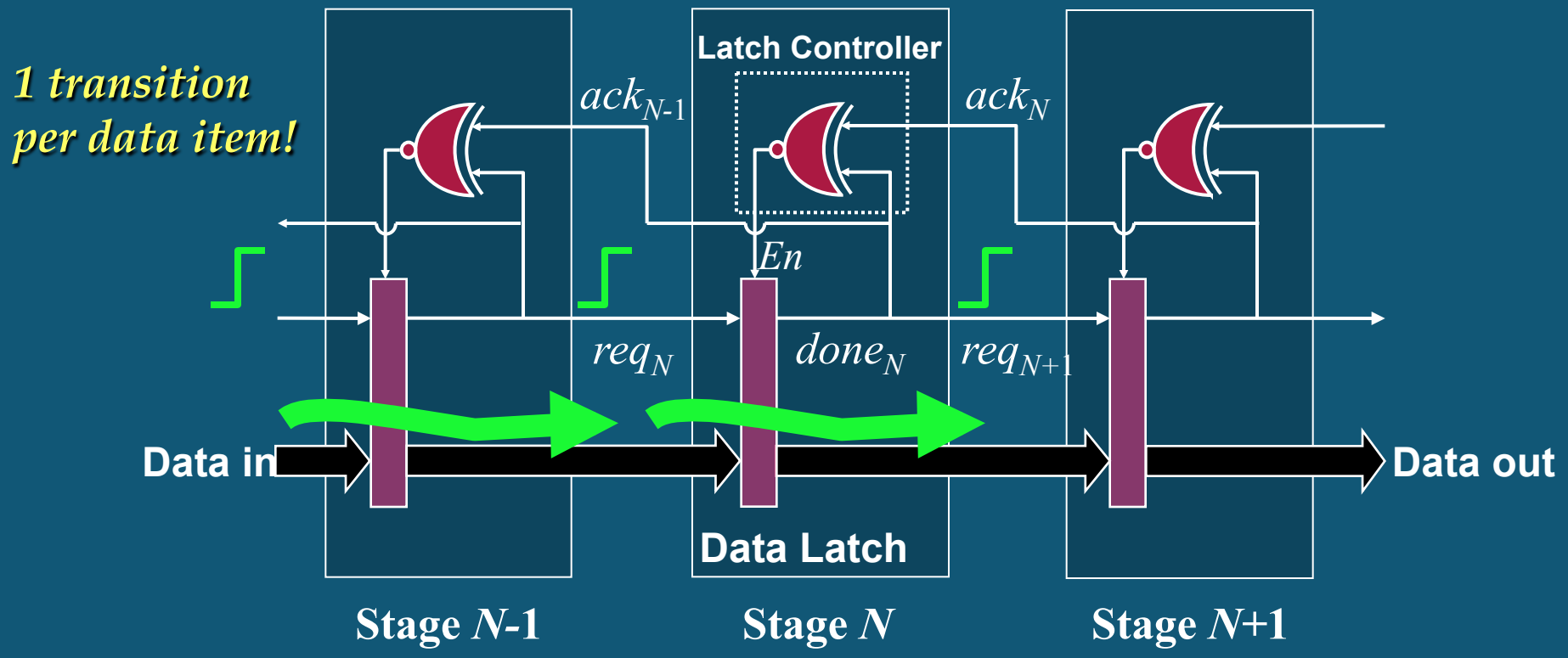
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

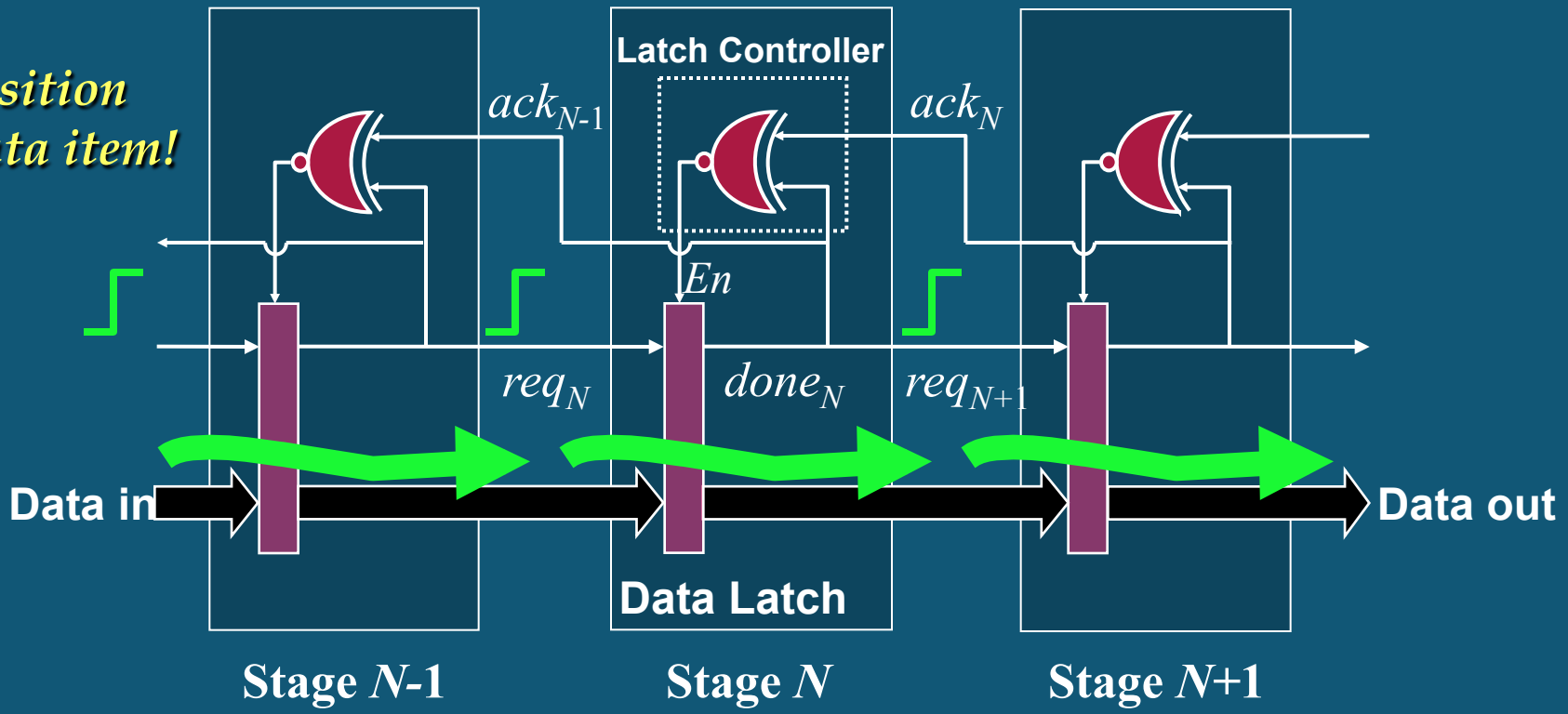
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

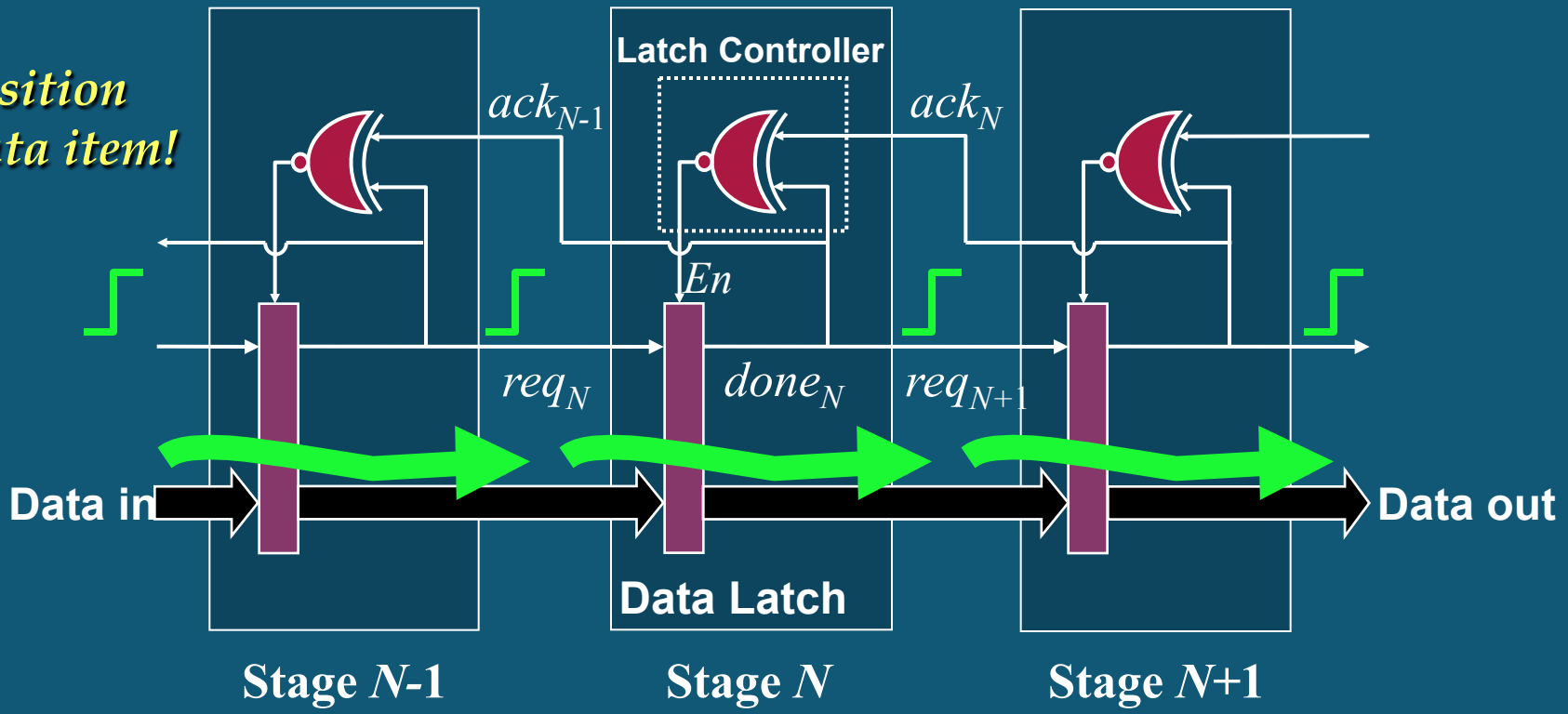
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

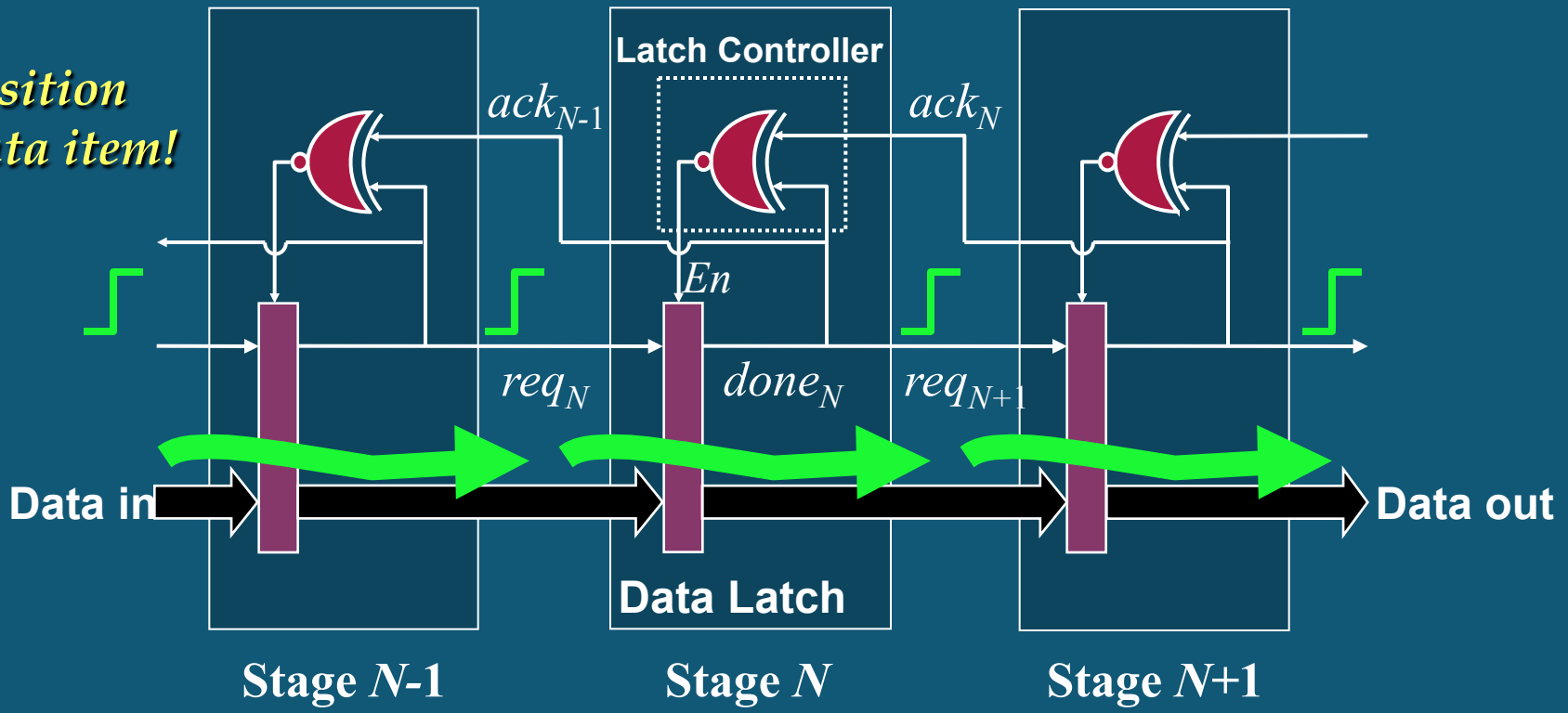
Stages communicate using *transition-signaling:*



1st data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

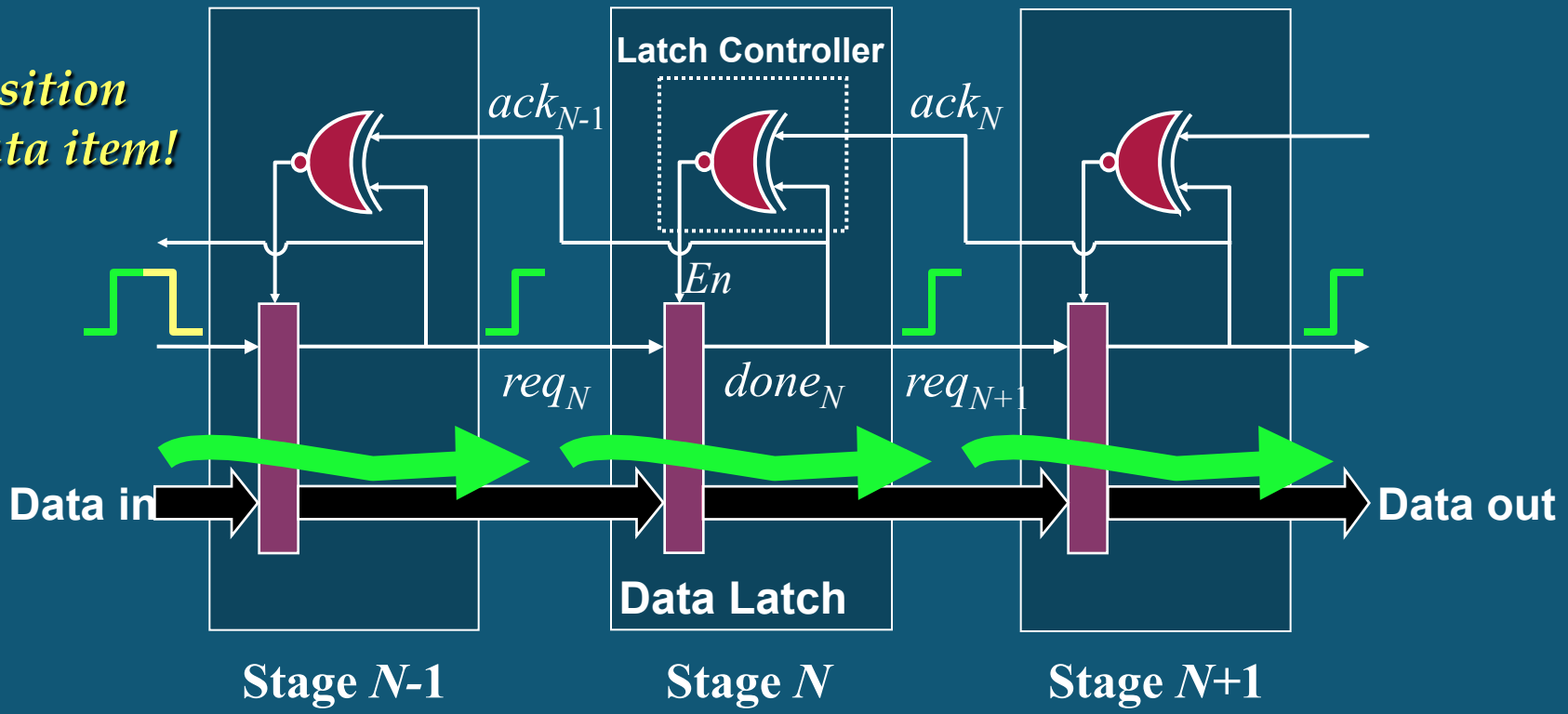
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

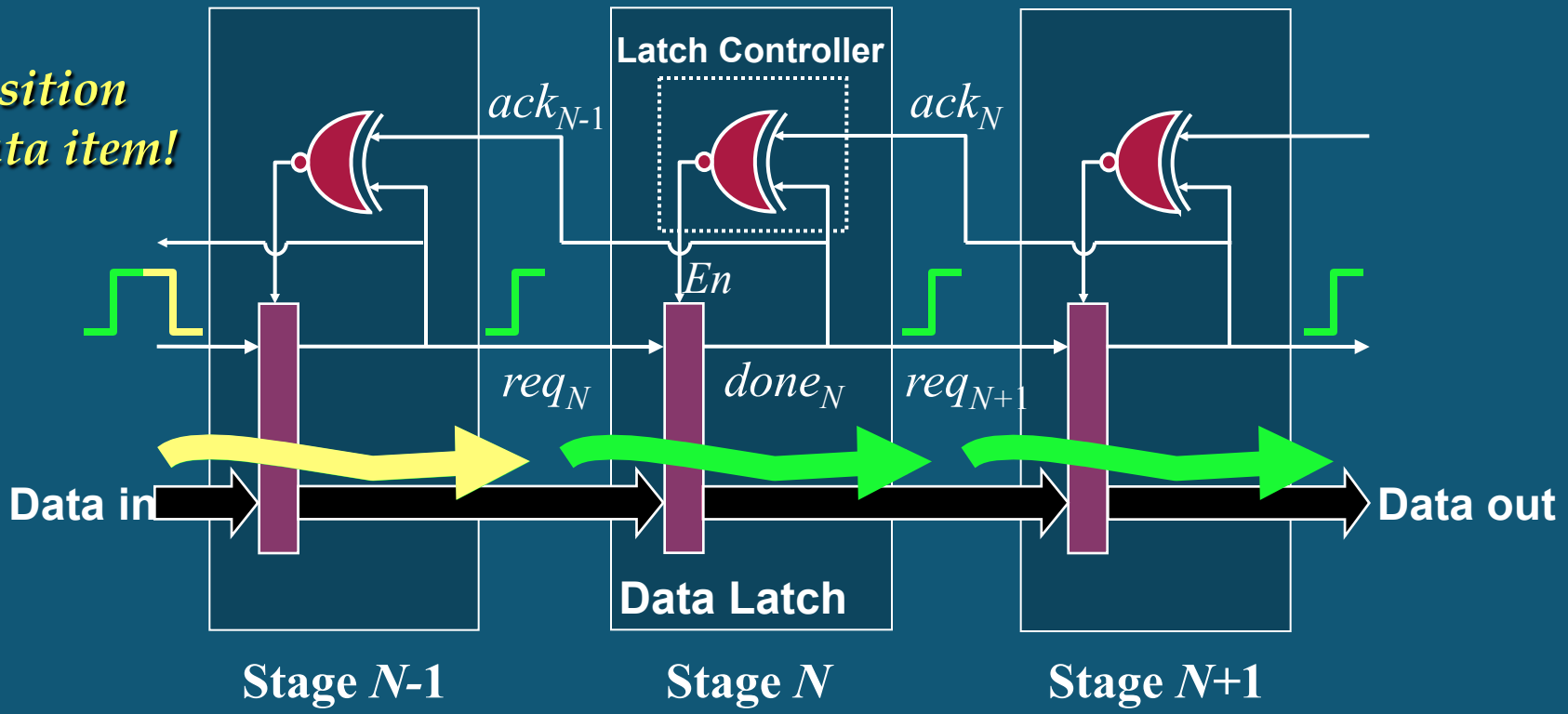
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

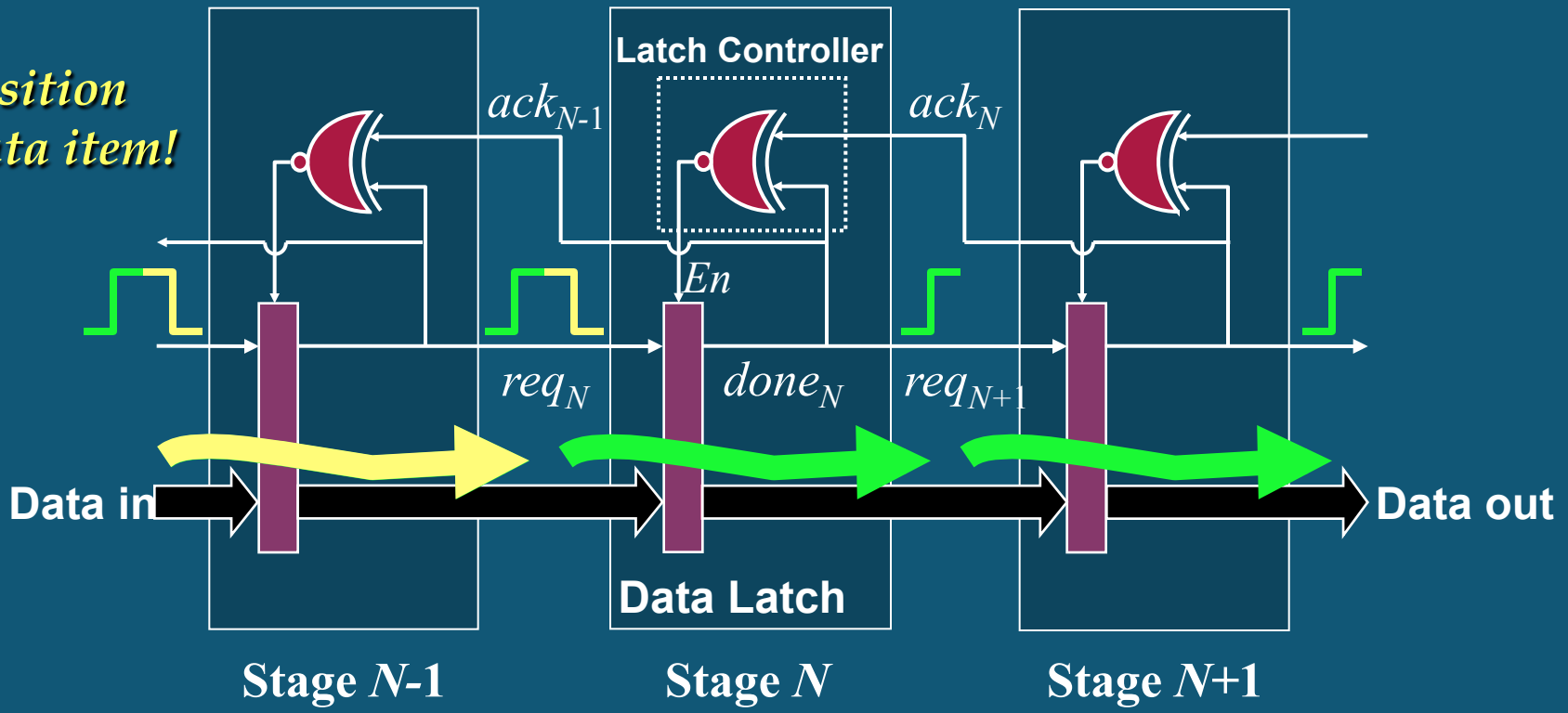
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

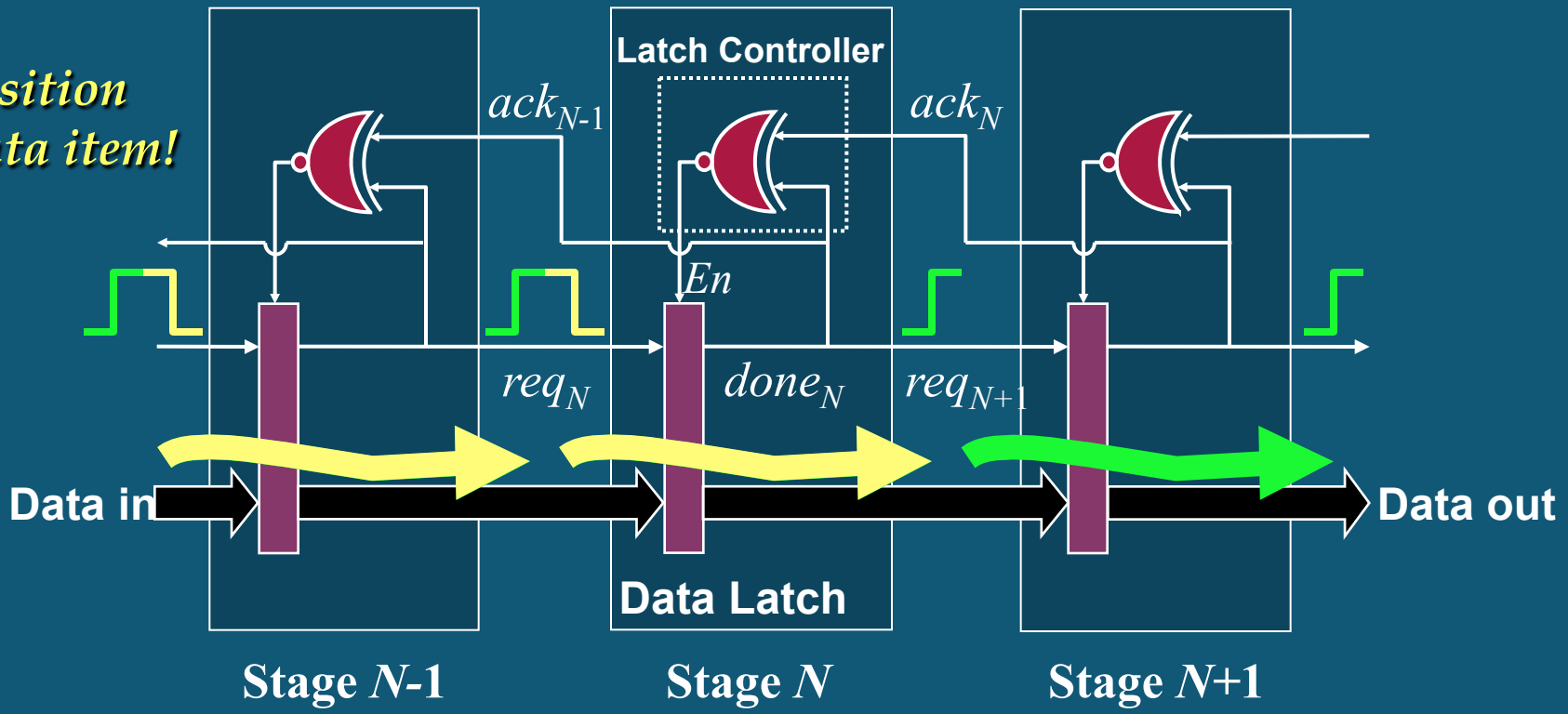
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

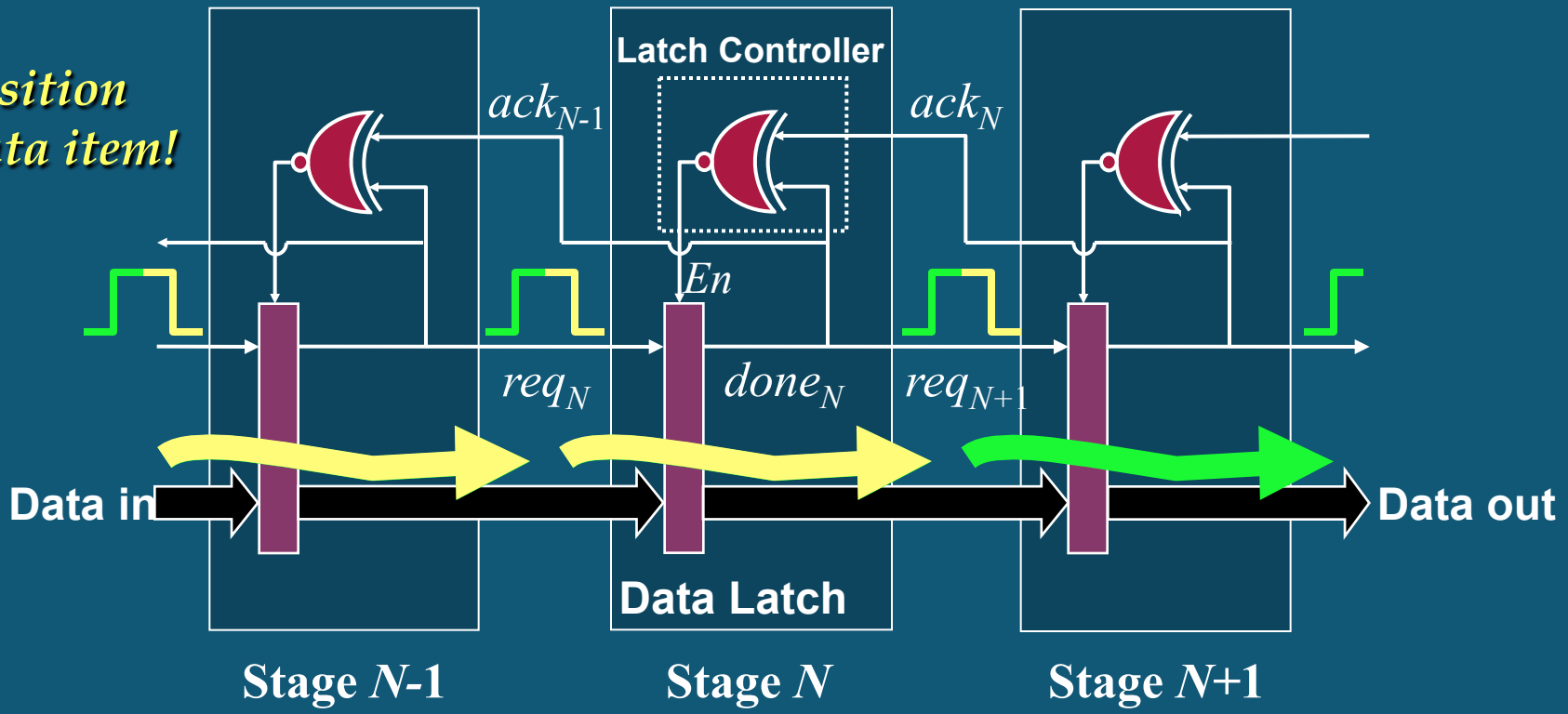
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

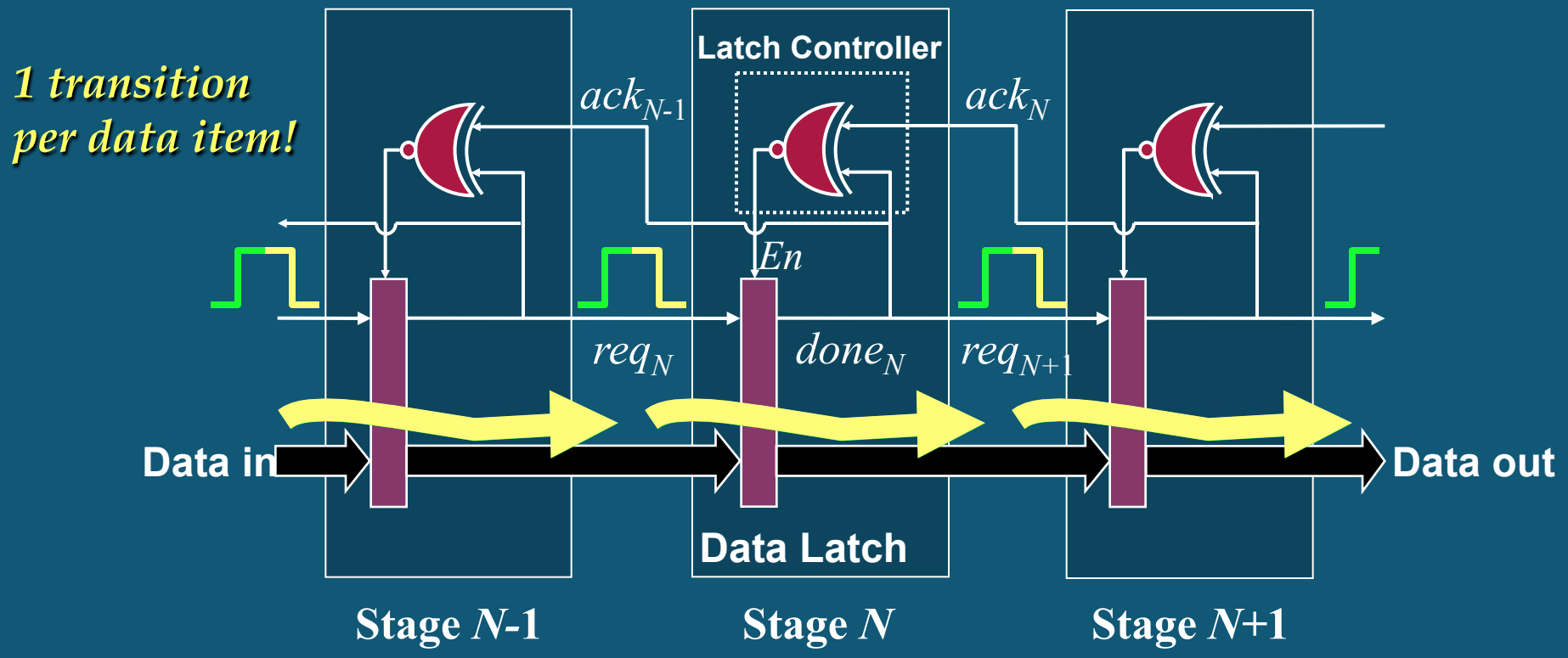
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

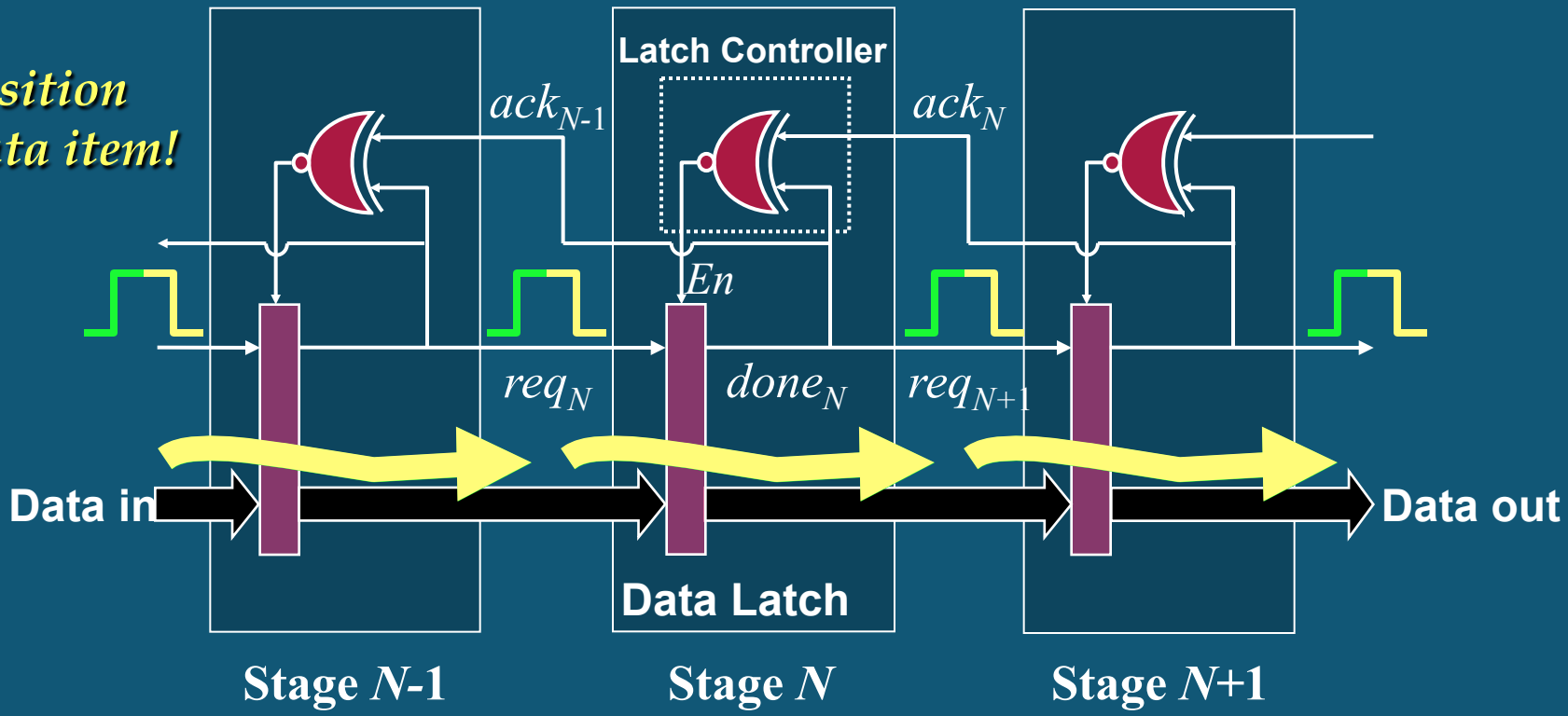
Stages communicate using *transition-signaling:*



2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO

Stages communicate using *transition-signaling:*

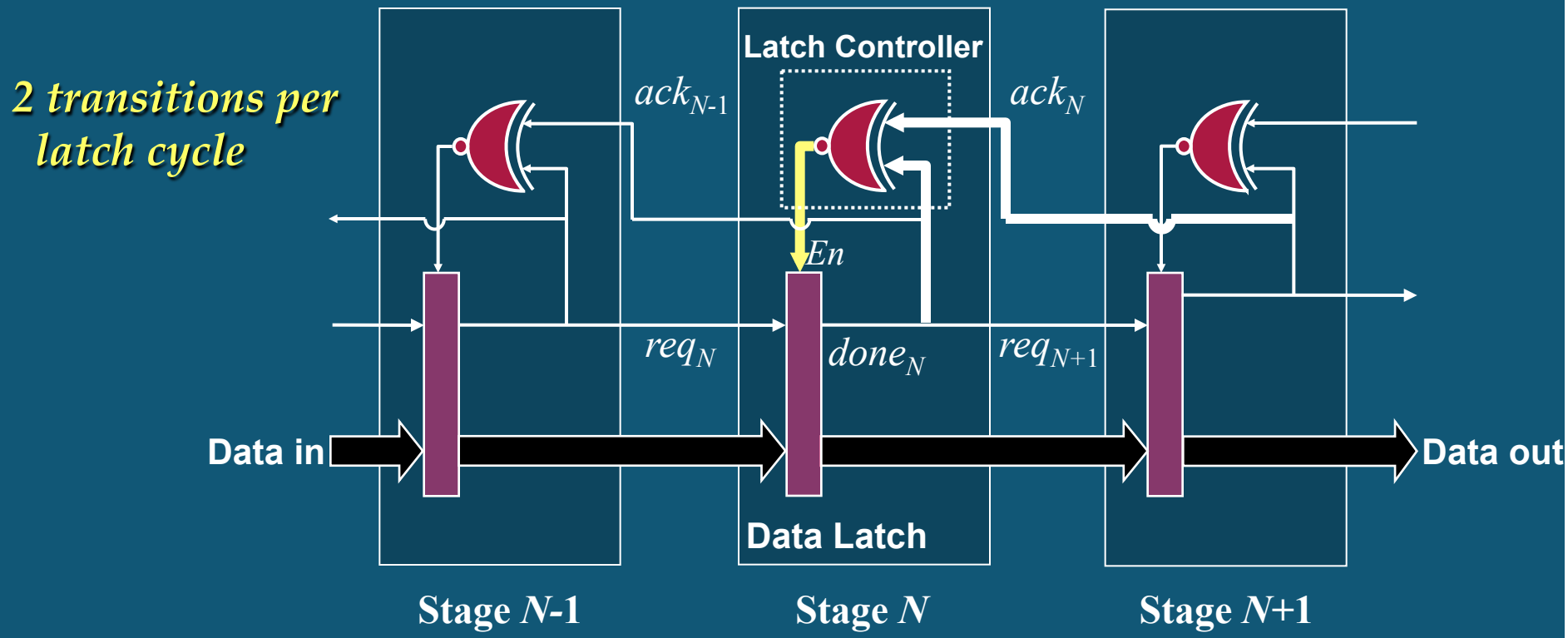


2nd data item flowing through the pipeline

MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

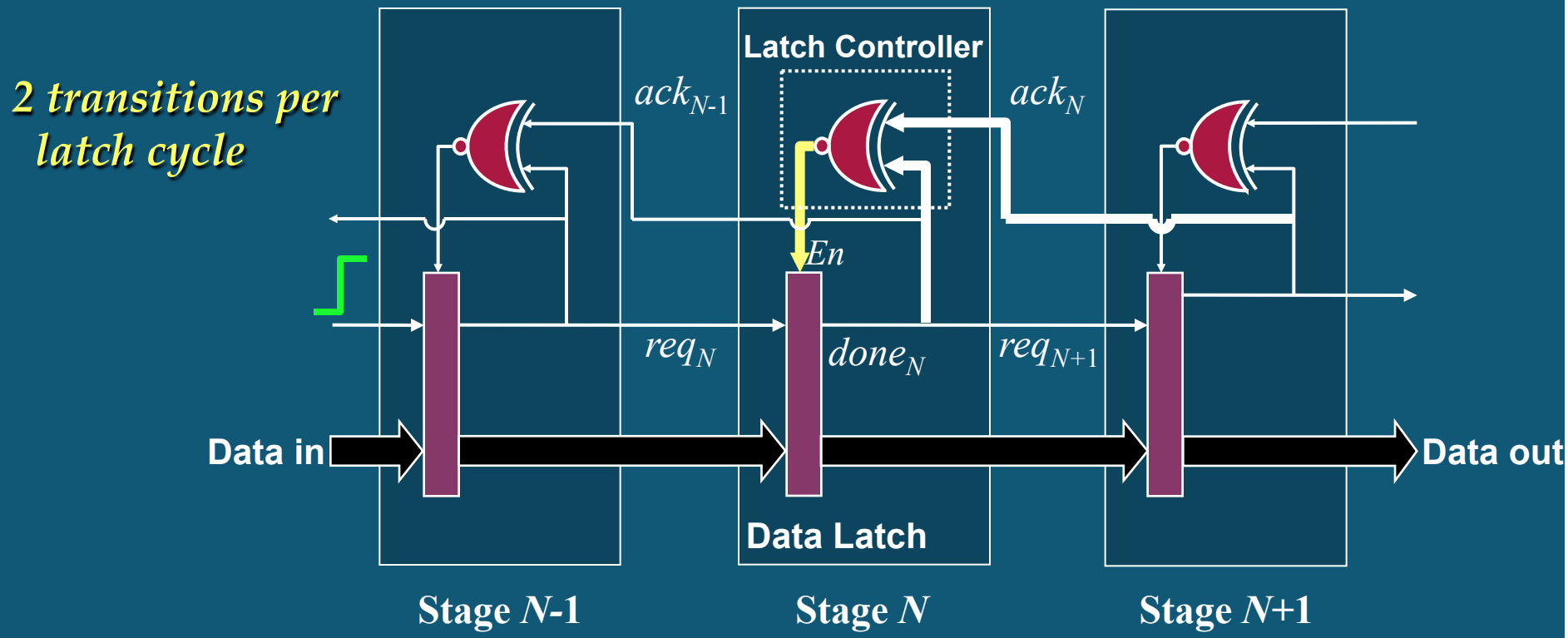
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

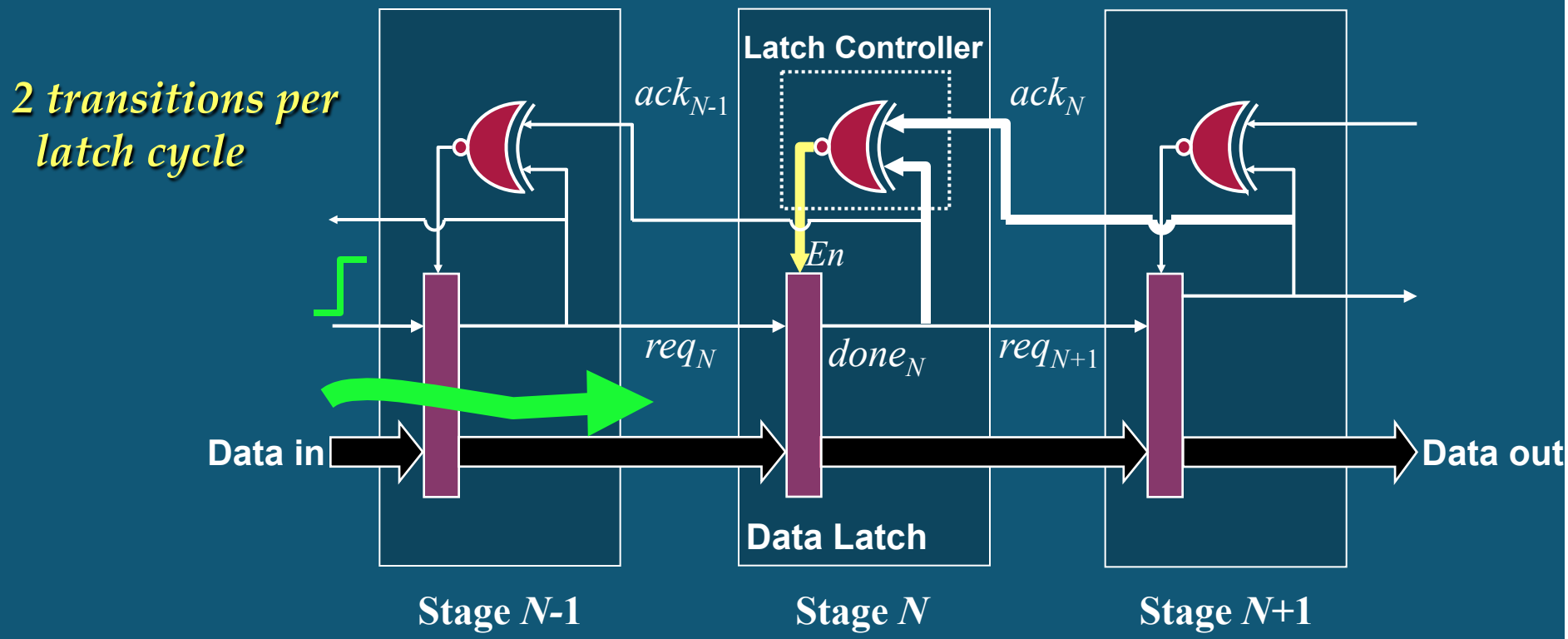
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

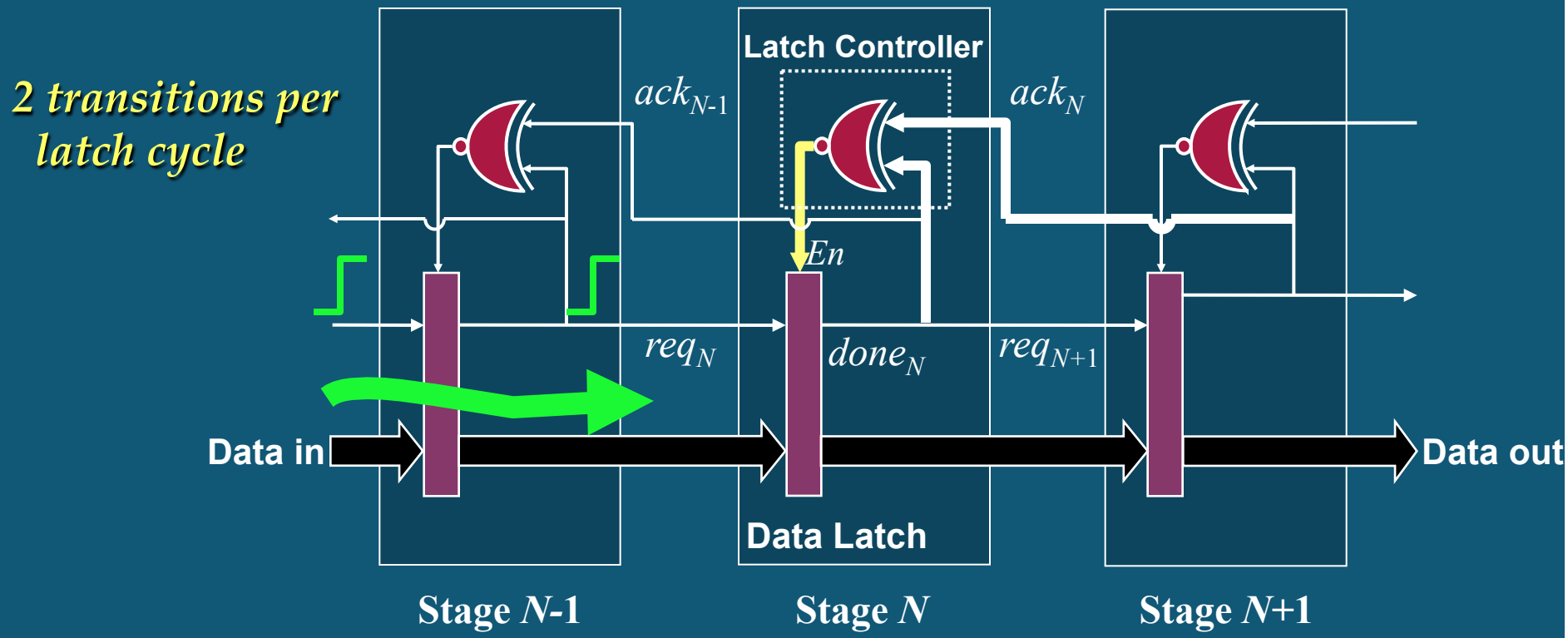
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

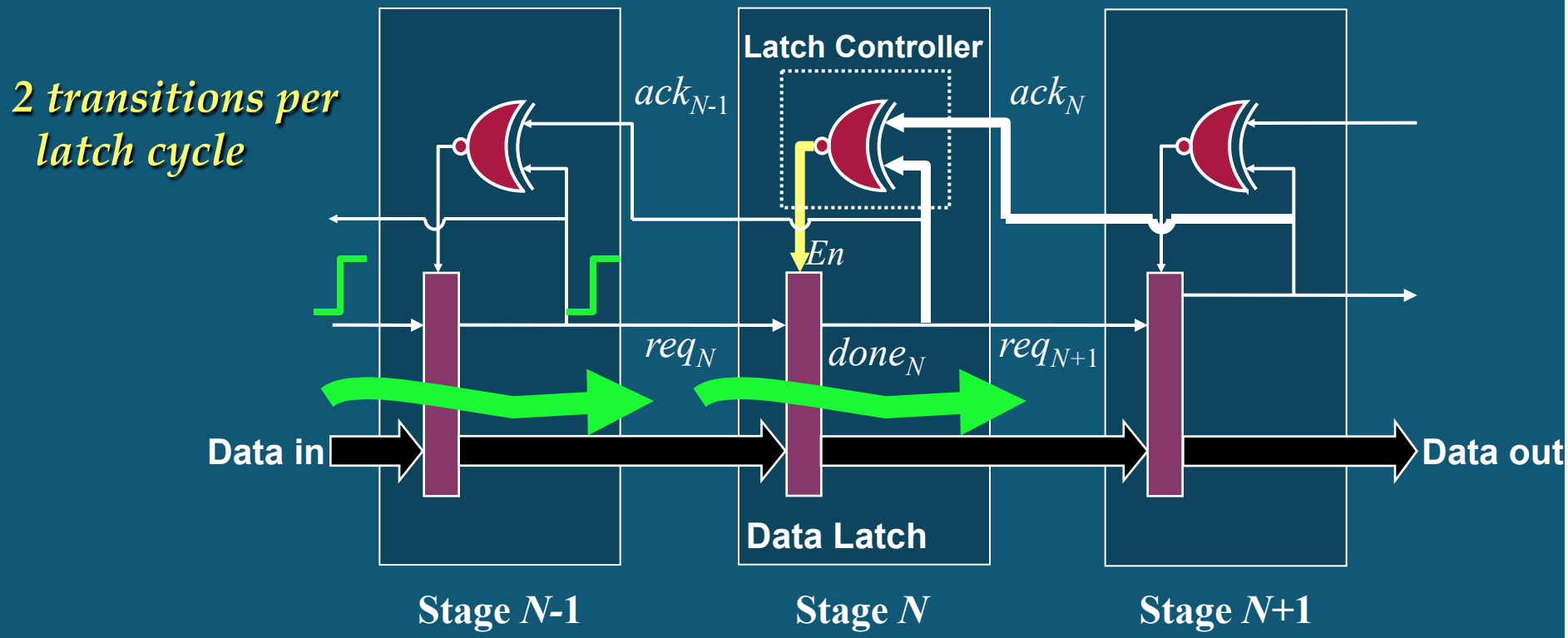
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

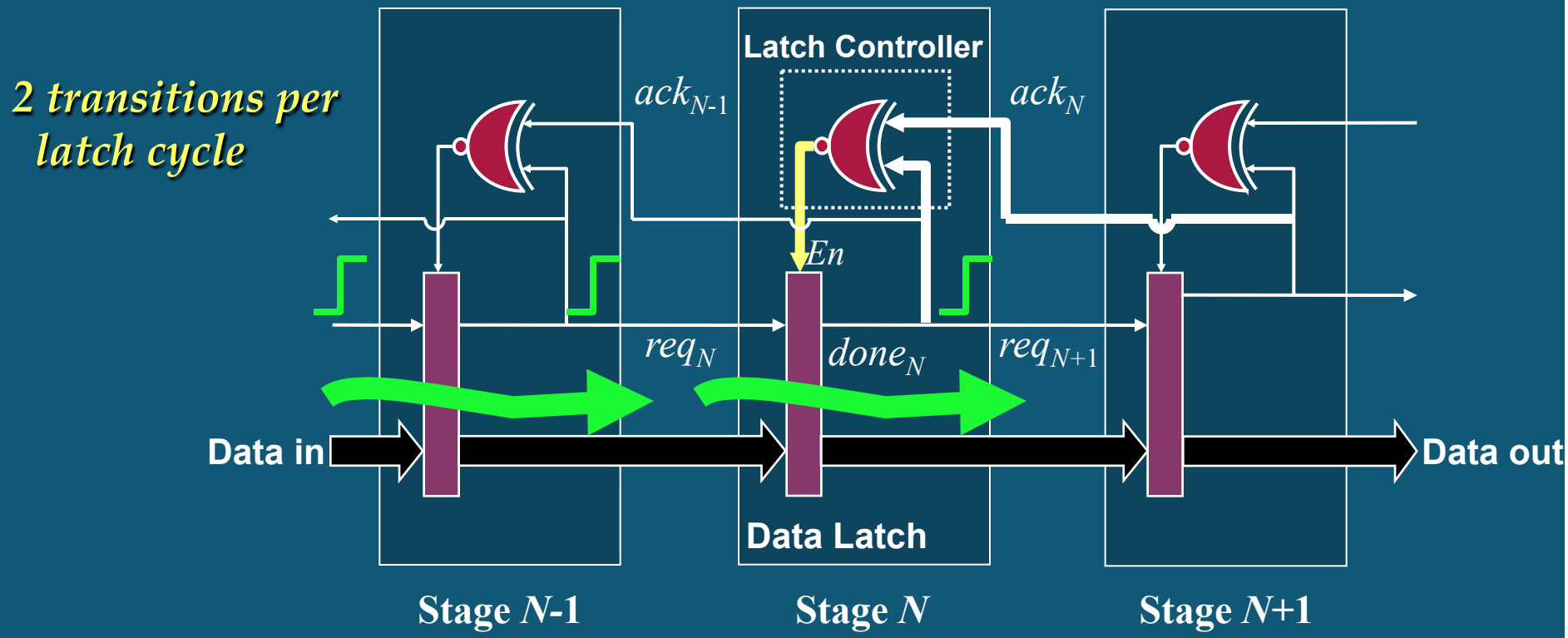
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

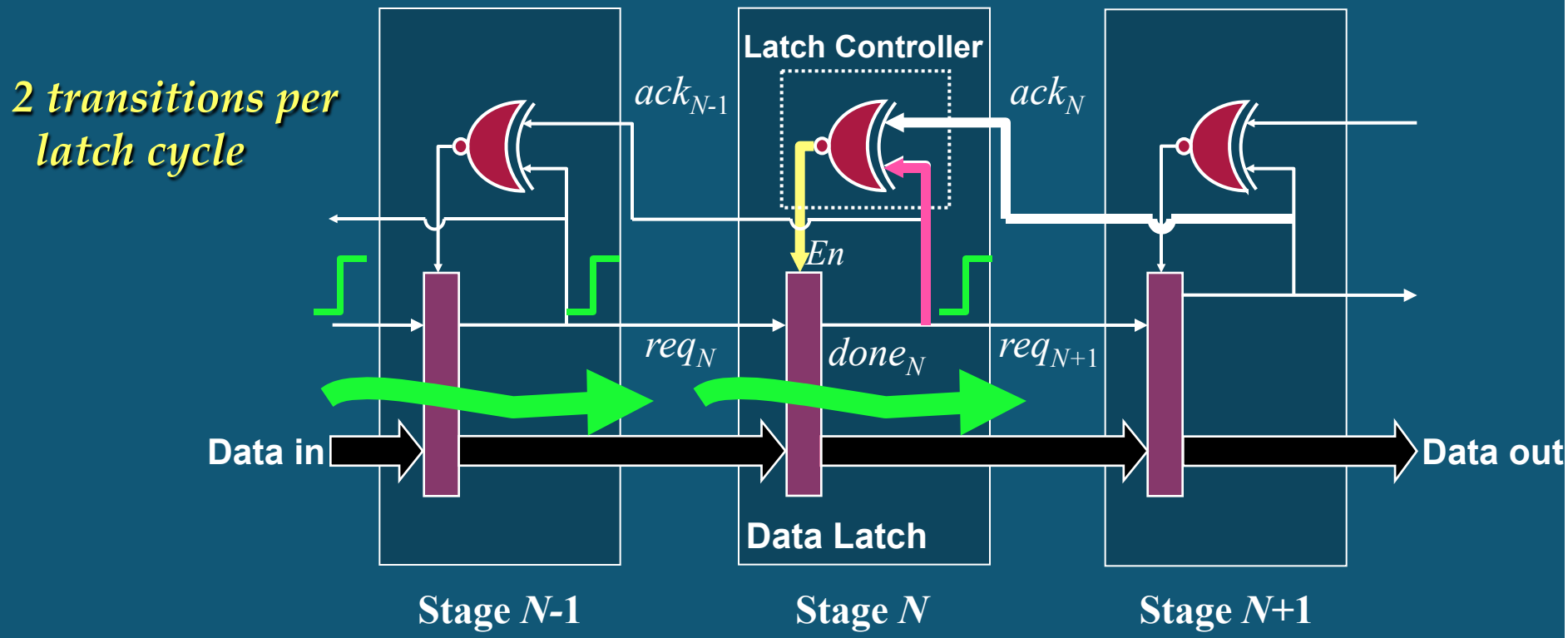
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

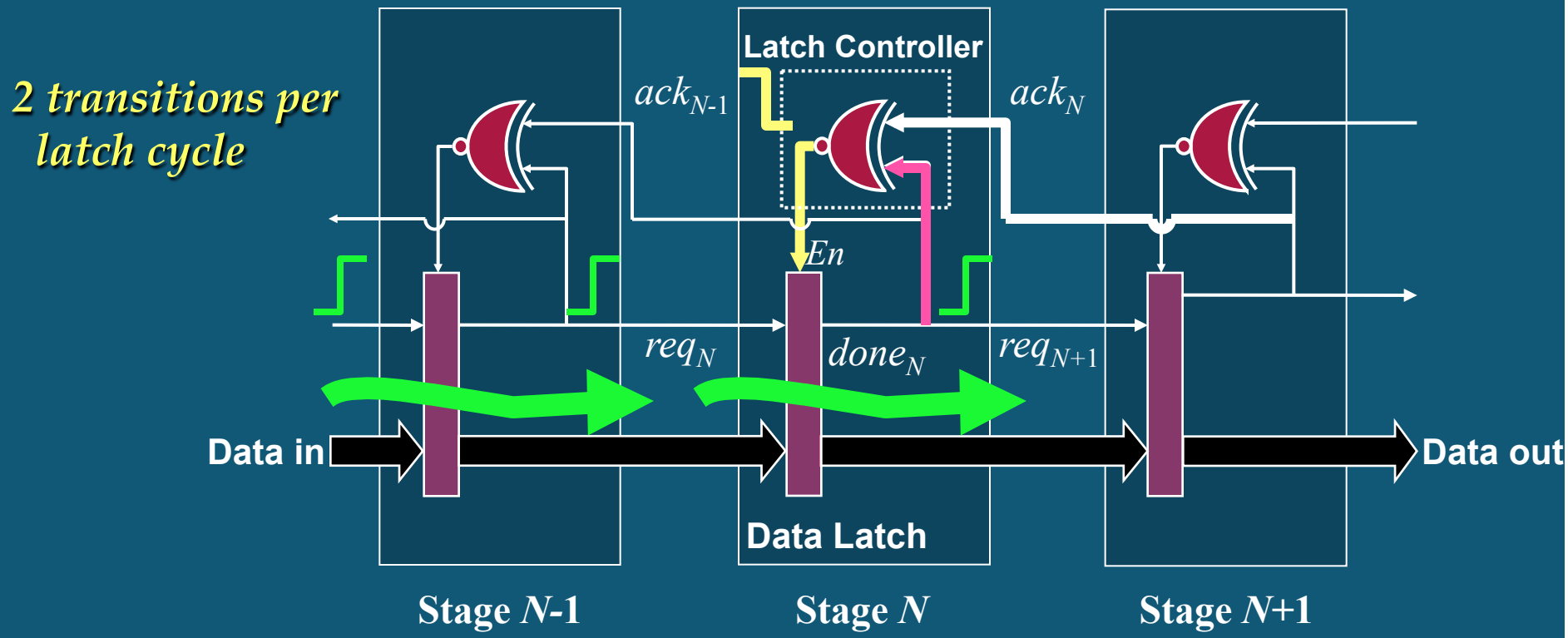
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

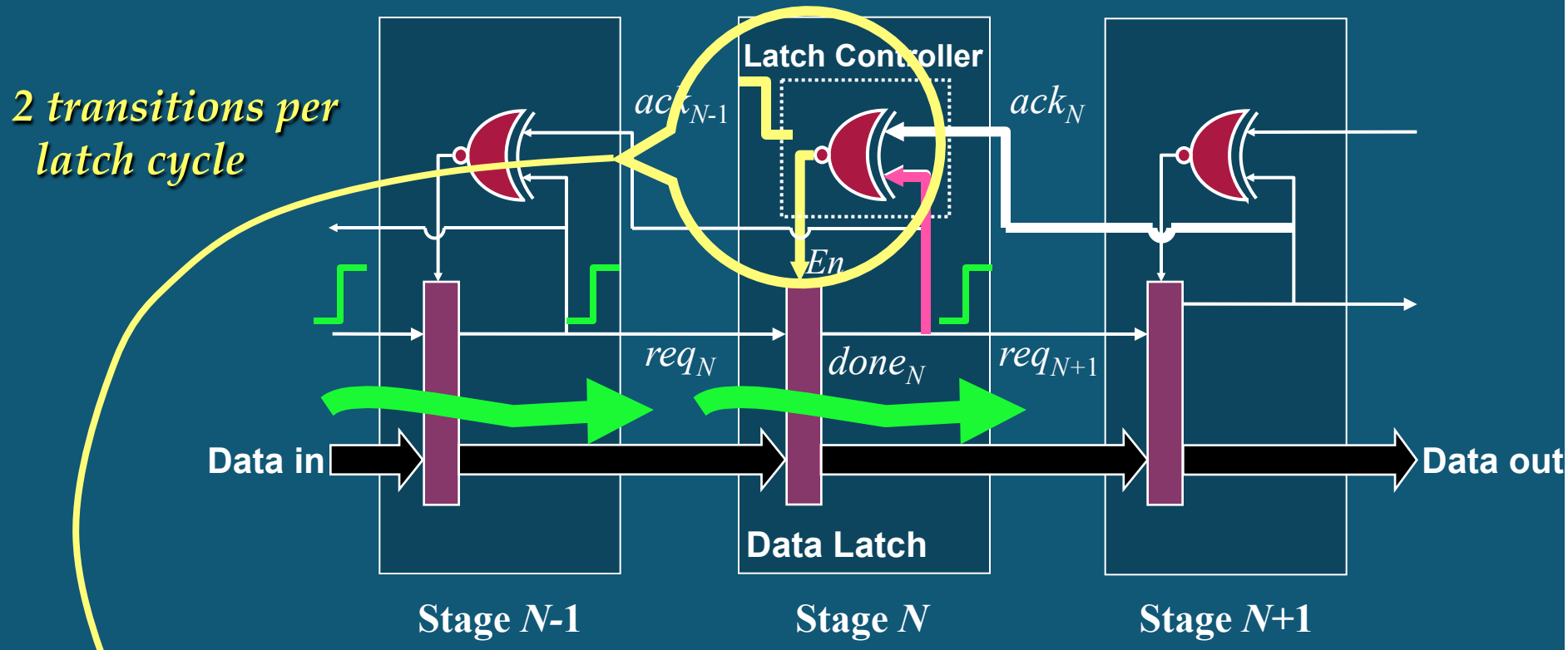
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

- 2 distinct transitions (up or down) ➔ *pulsed latch enable*

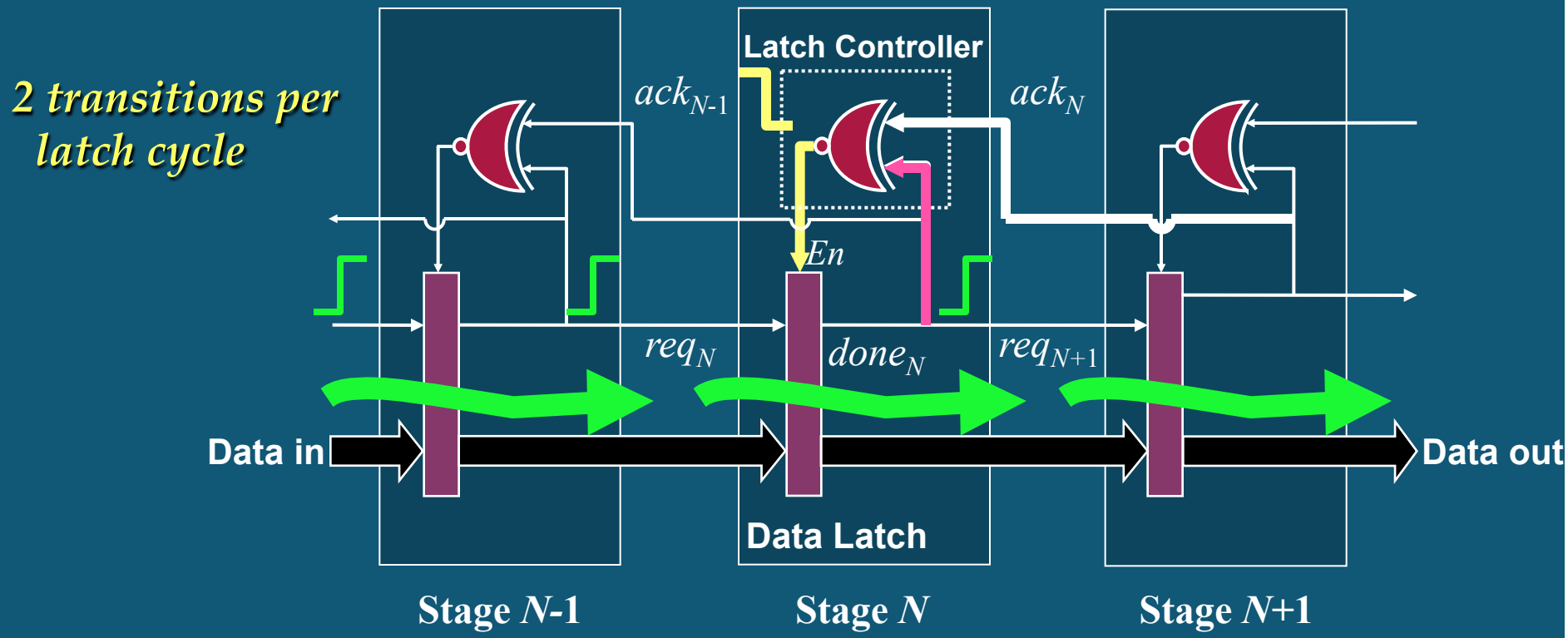


Latch is disabled when *current stage is "done"*

MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

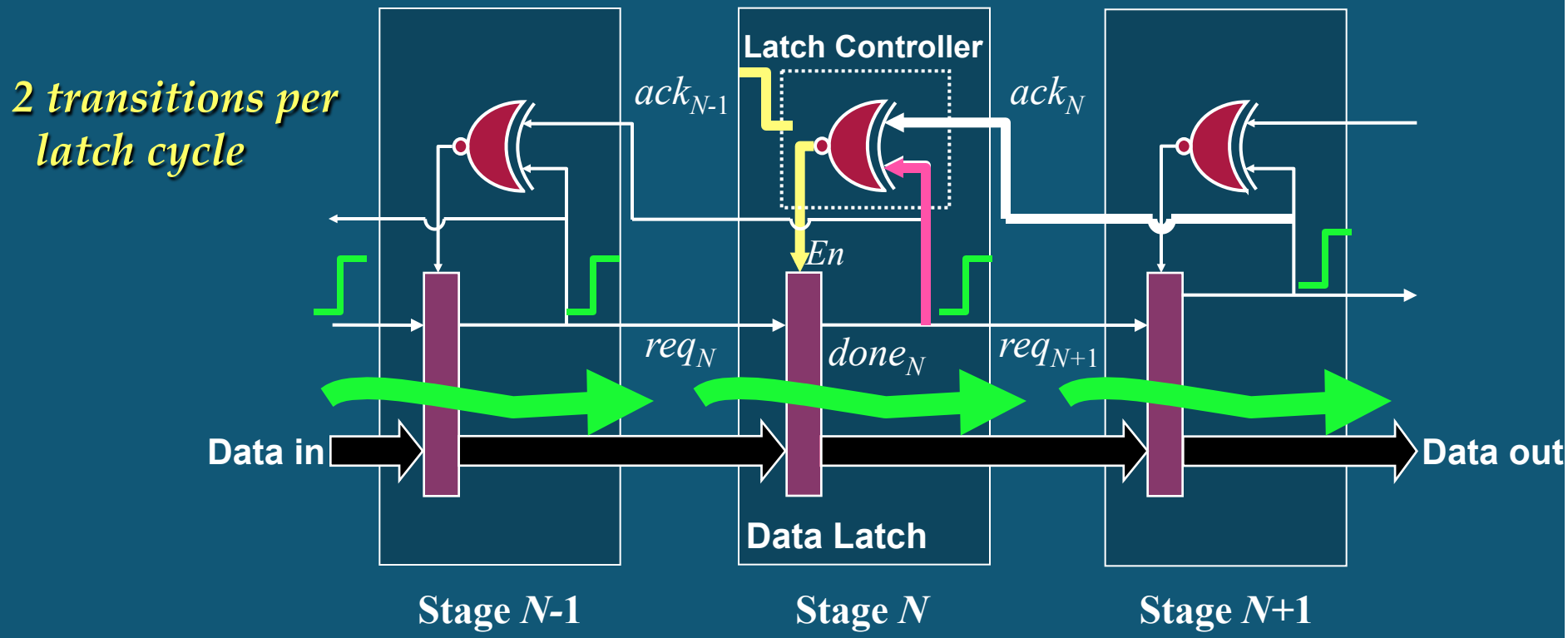
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

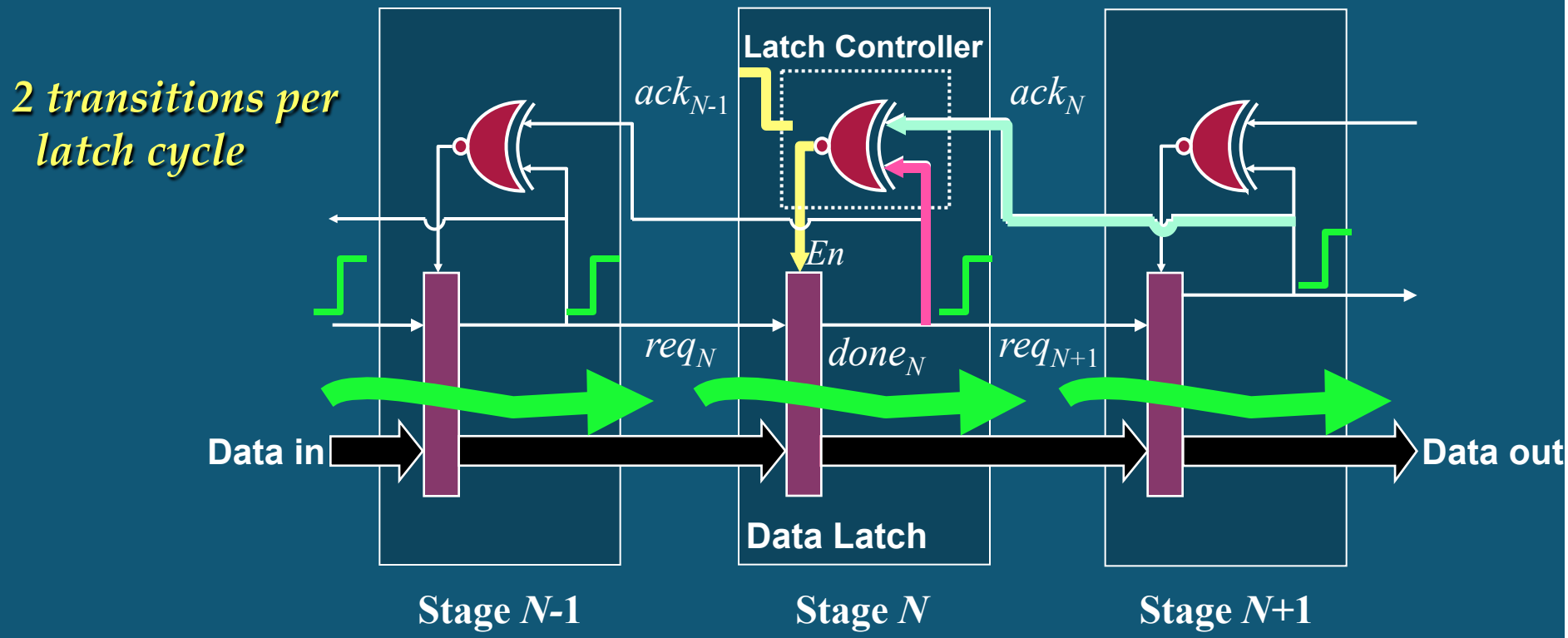
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

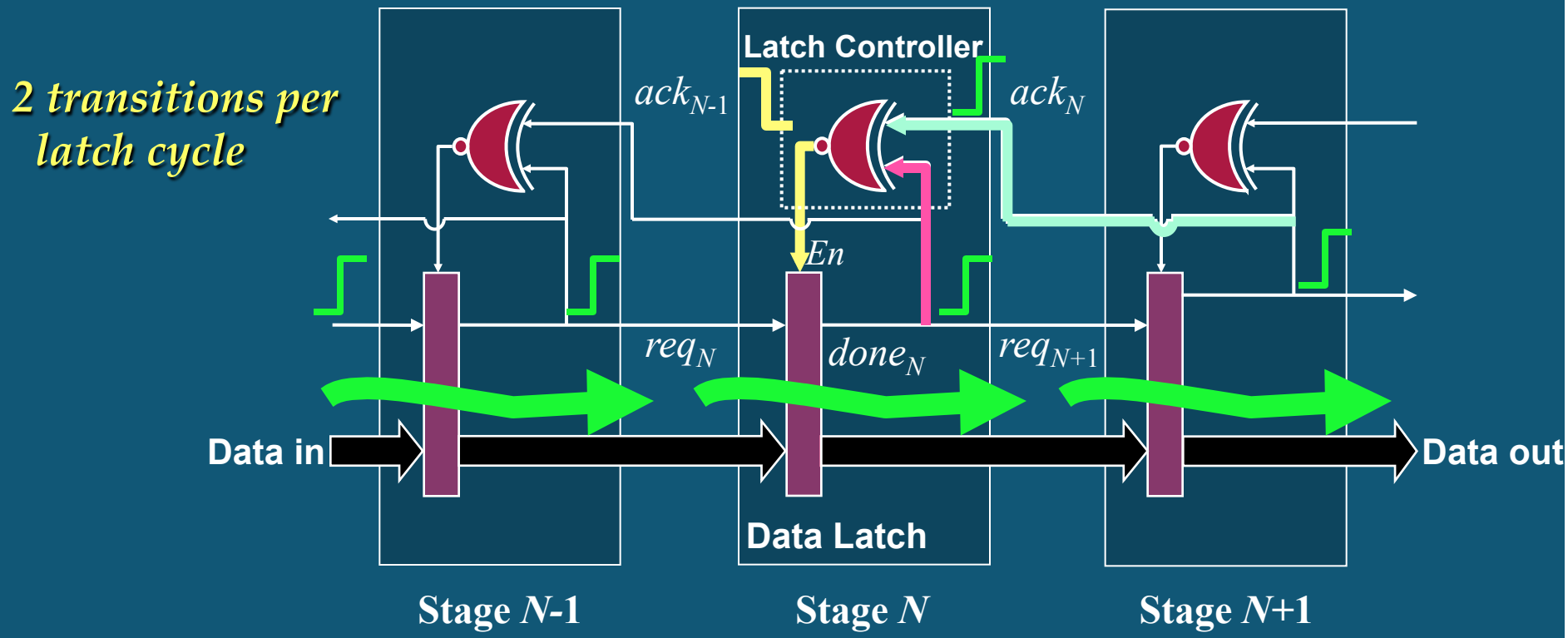
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

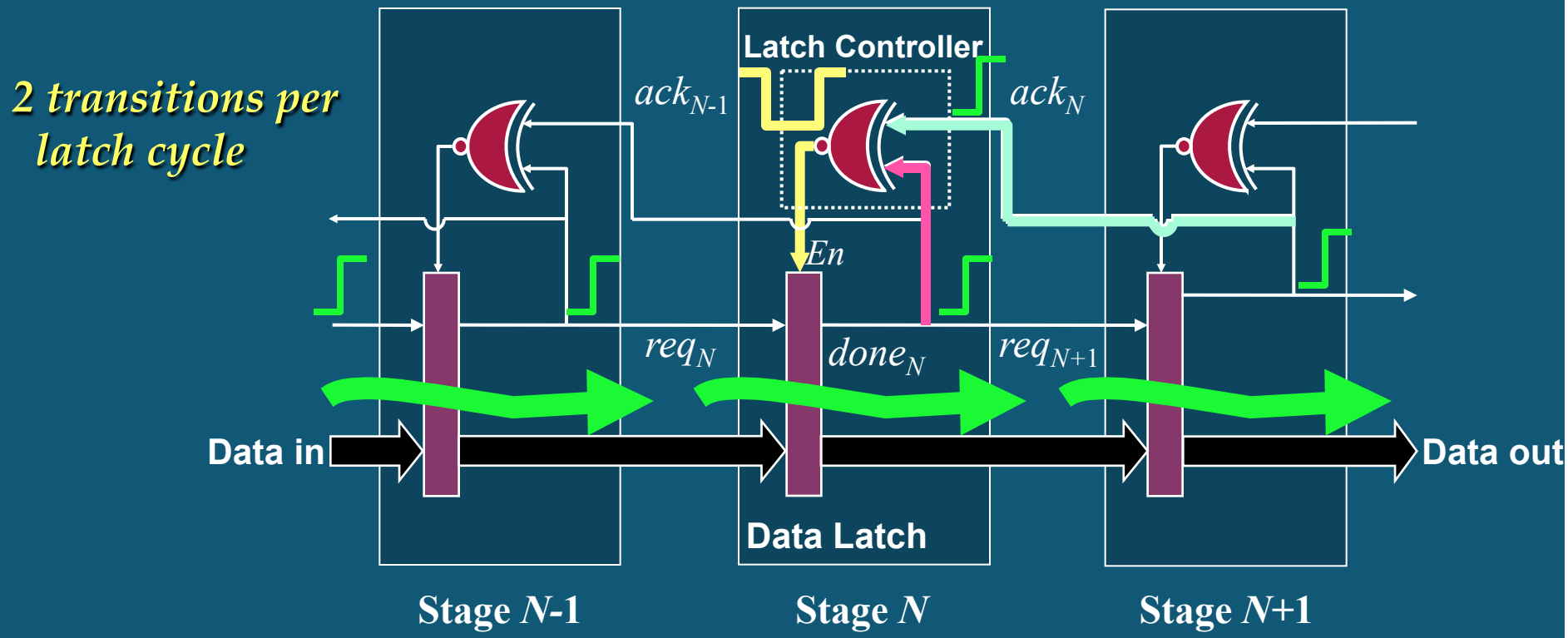
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

Latch controller (XNOR) acts as *"phase converter"*:

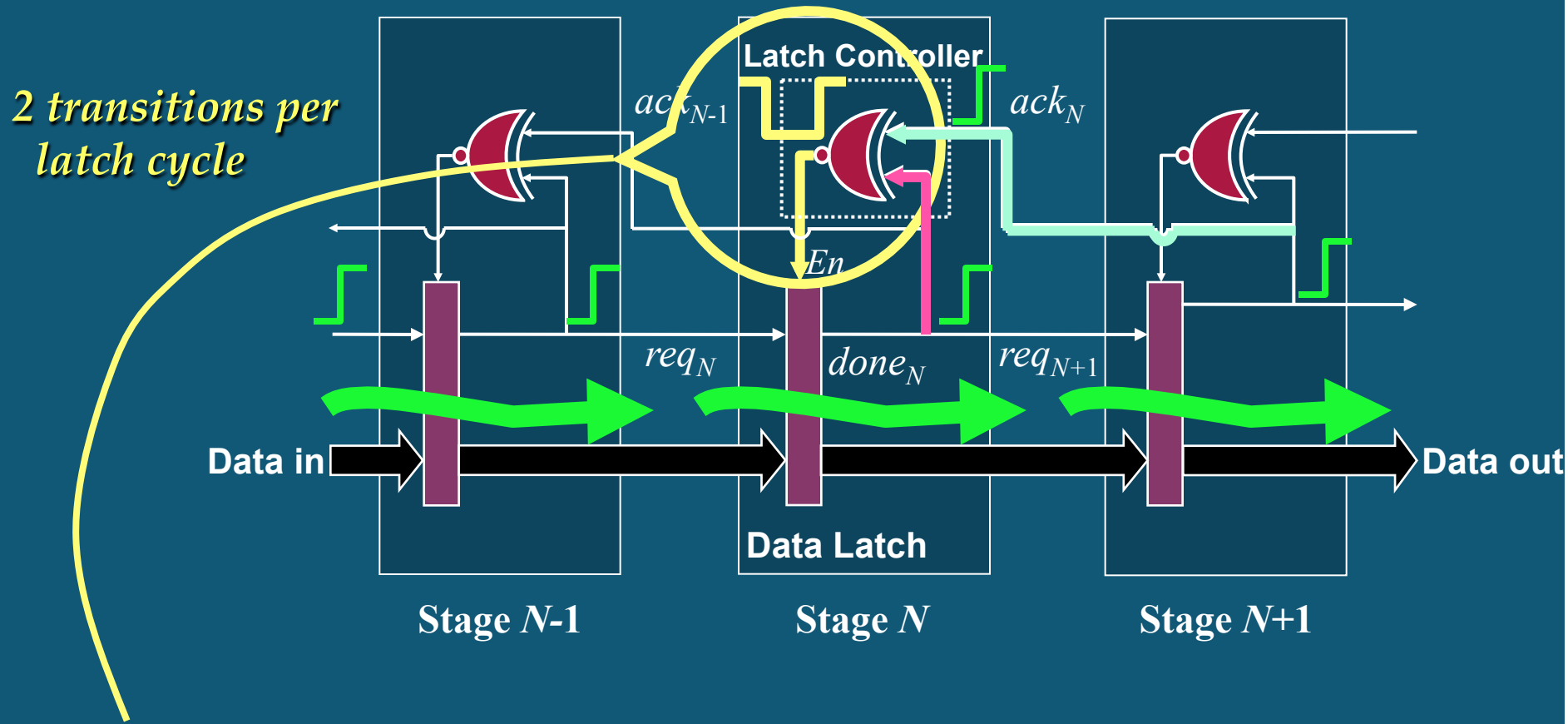
- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



MOUSETRAP: A Basic FIFO (contd.)

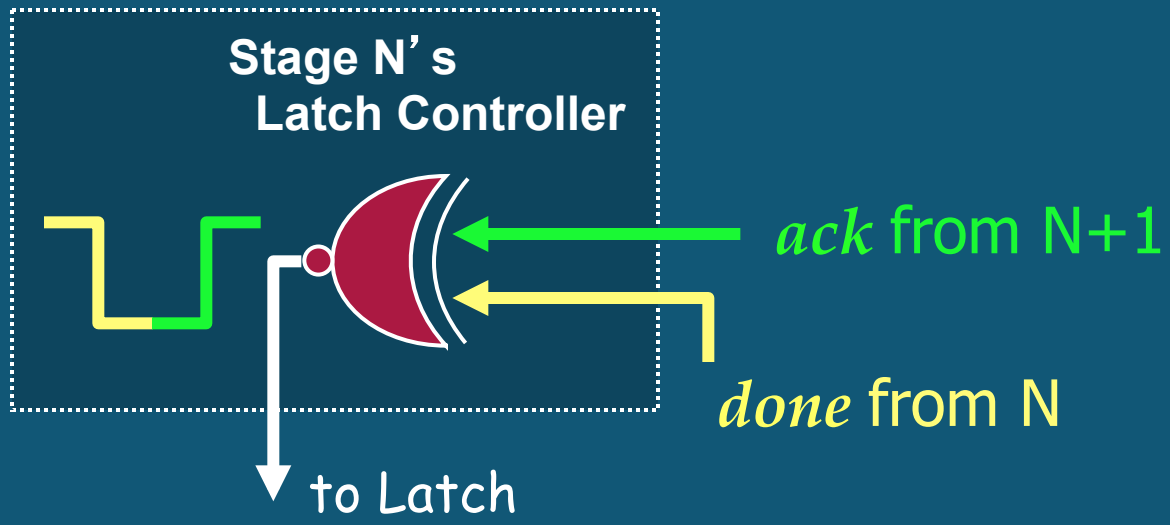
Latch controller (XNOR) acts as *"phase converter"*:

- 2 distinct transitions (up or down) ➔ *pulsed latch enable*



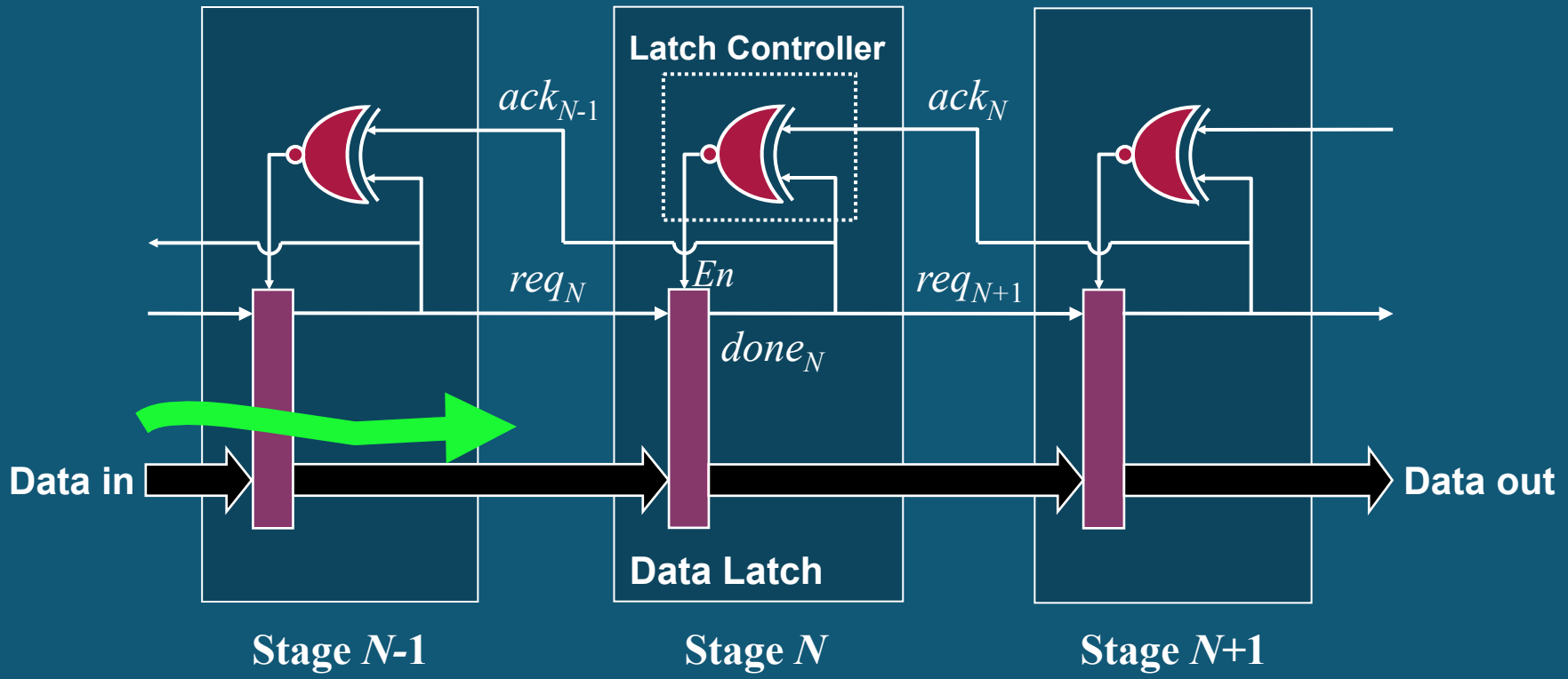
Latch is re-enabled when *next stage is "done"*

Detailed Controller Operation

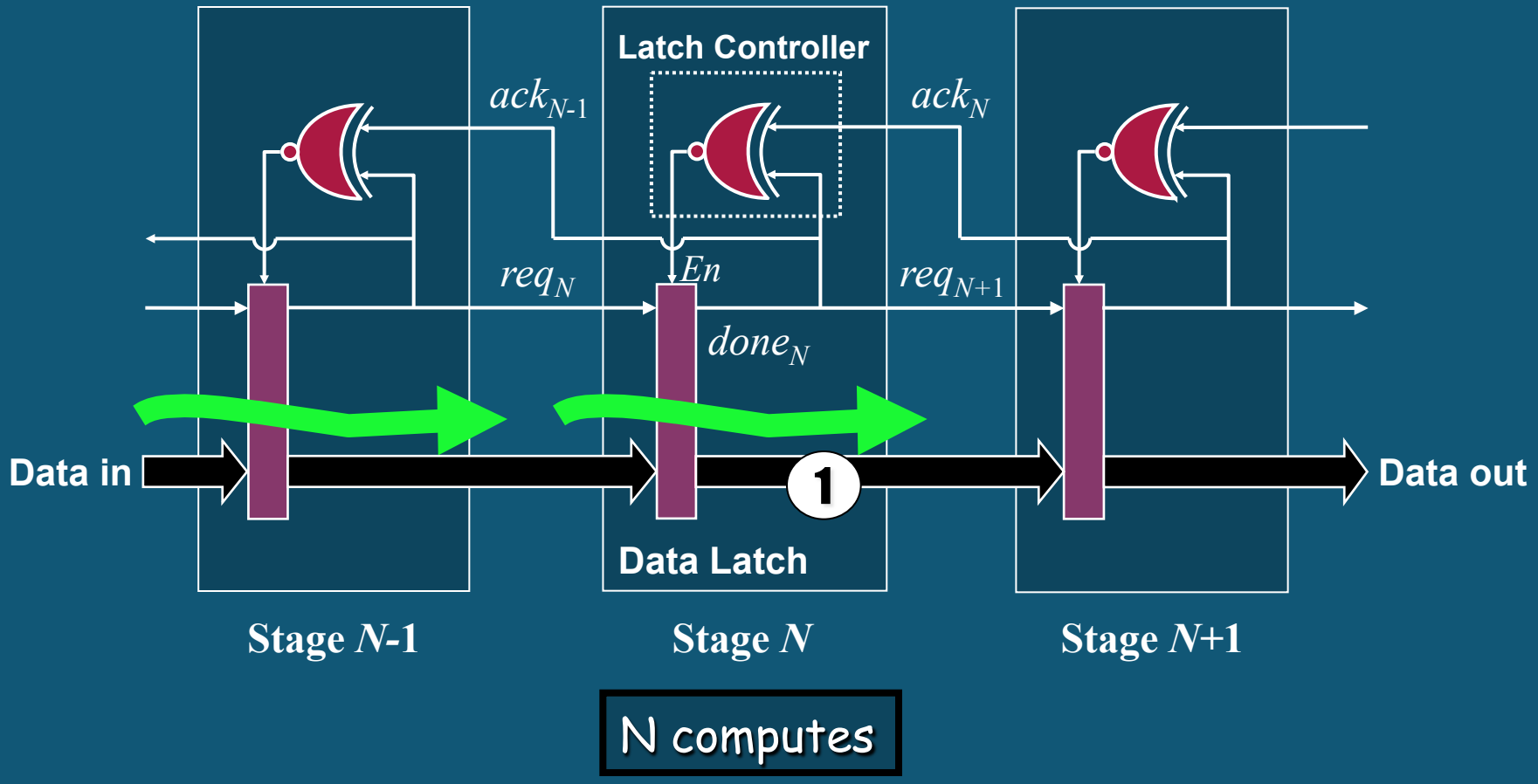


- One pulse per data item flowing through:
  - down transition: caused by *"done"* of N
  - up transition: caused by *"done"* of N+1
- *No minimum pulse width constraint!*
  - simply, down transition should start "early enough"
  - can be "negative width" (no pulse!)

MOUSETRAP: FIFO Cycle Time

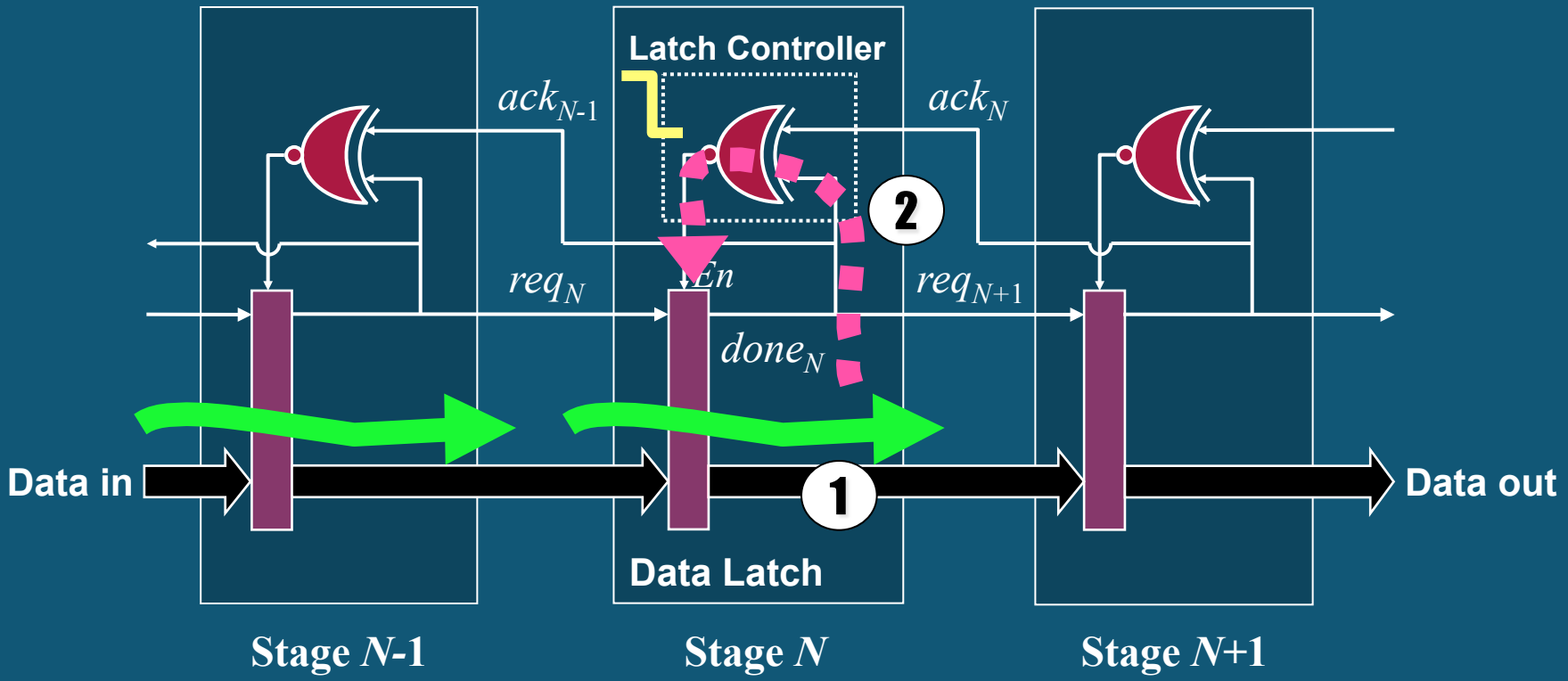


MOUSETRAP: FIFO Cycle Time



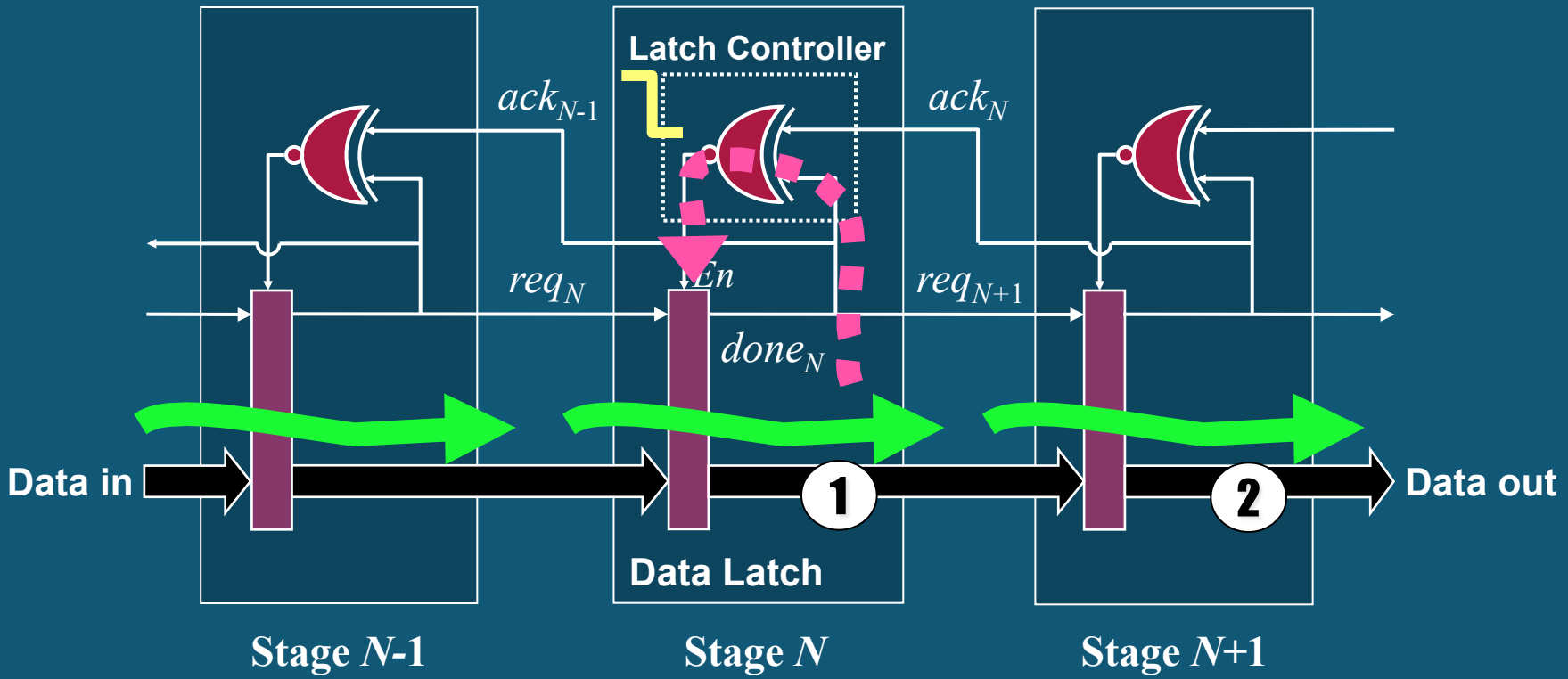
N computes

MOUSETRAP: FIFO Cycle Time



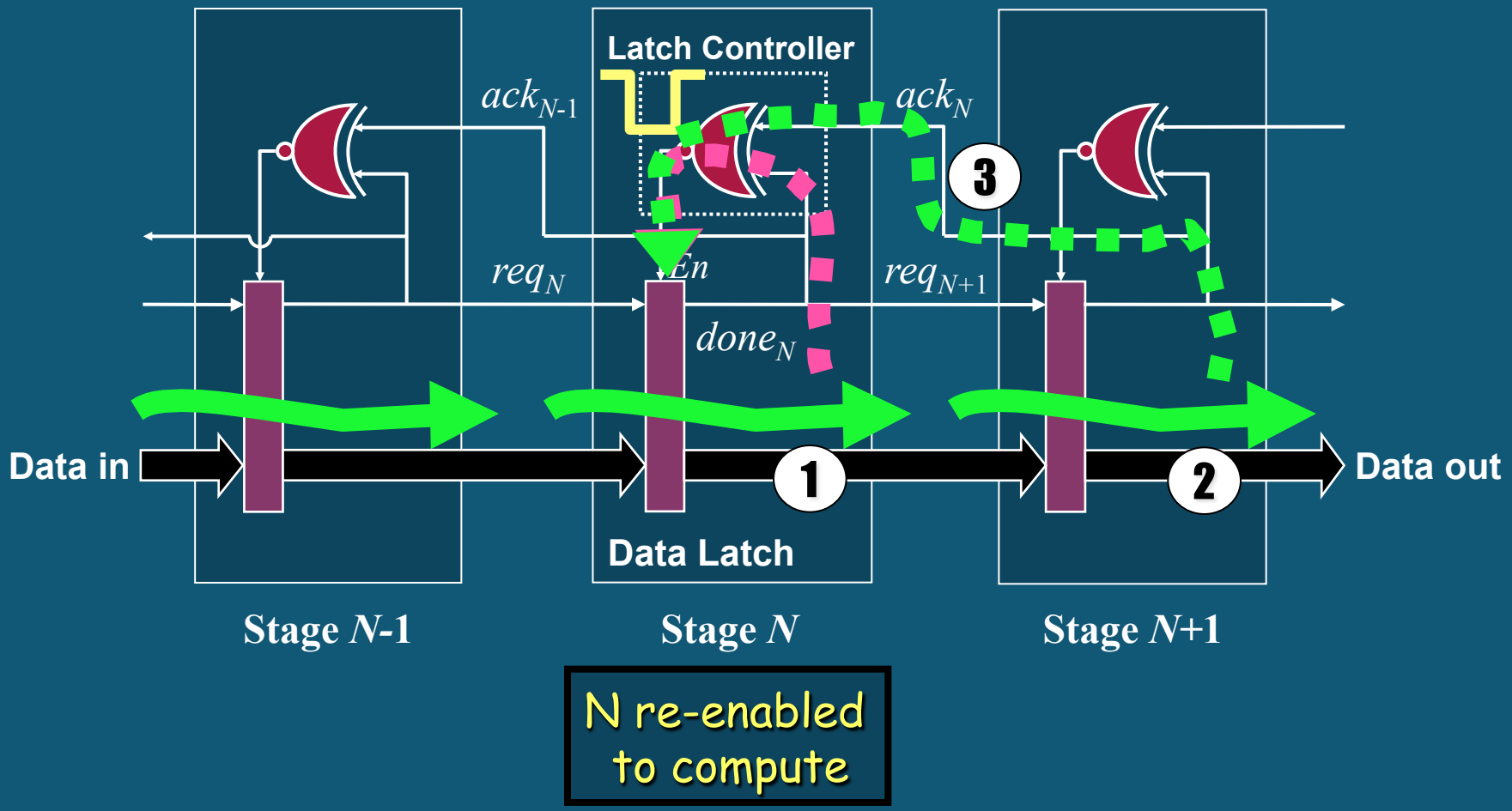
Fast self-loop:
N disables itself

MOUSETRAP: FIFO Cycle Time



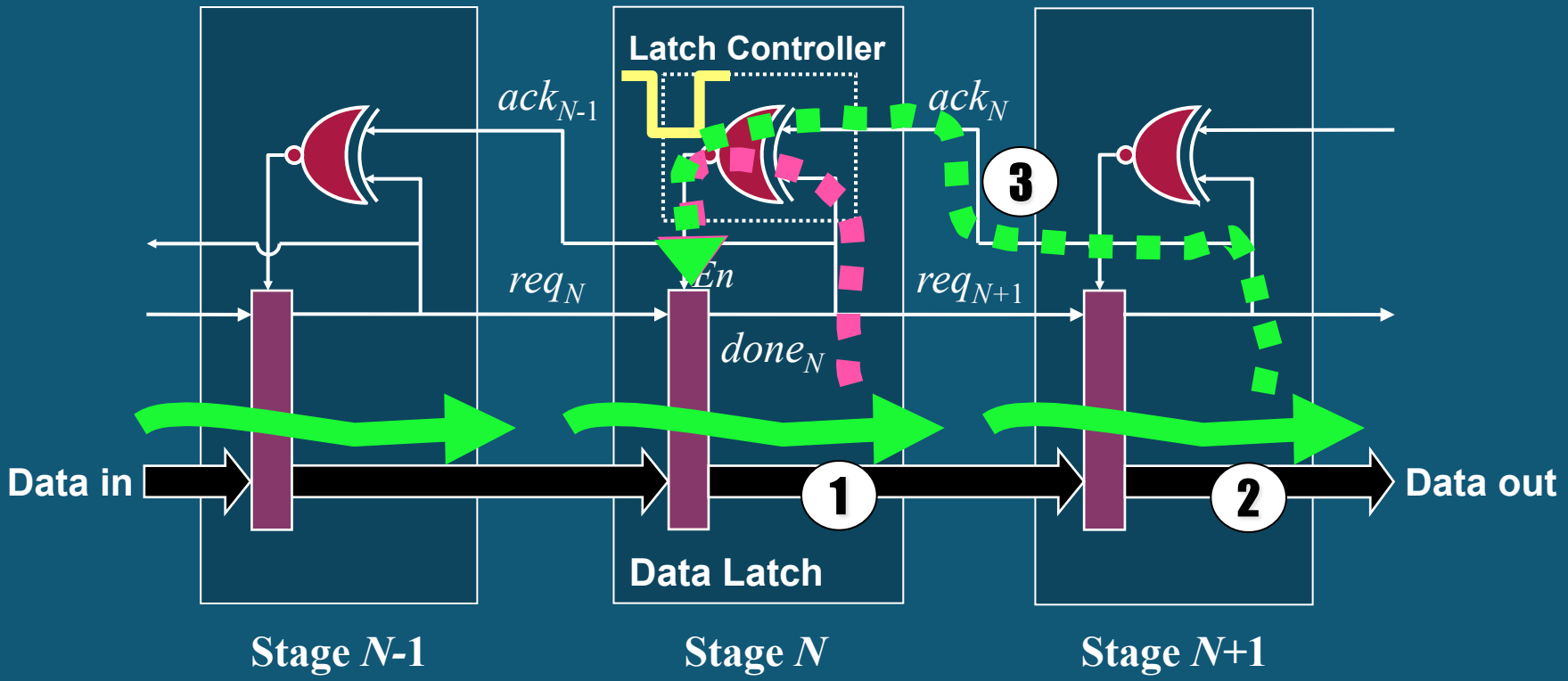
N+1 computes

MOUSETRAP: FIFO Cycle Time



N re-enabled to compute

MOUSETRAP: FIFO Cycle Time

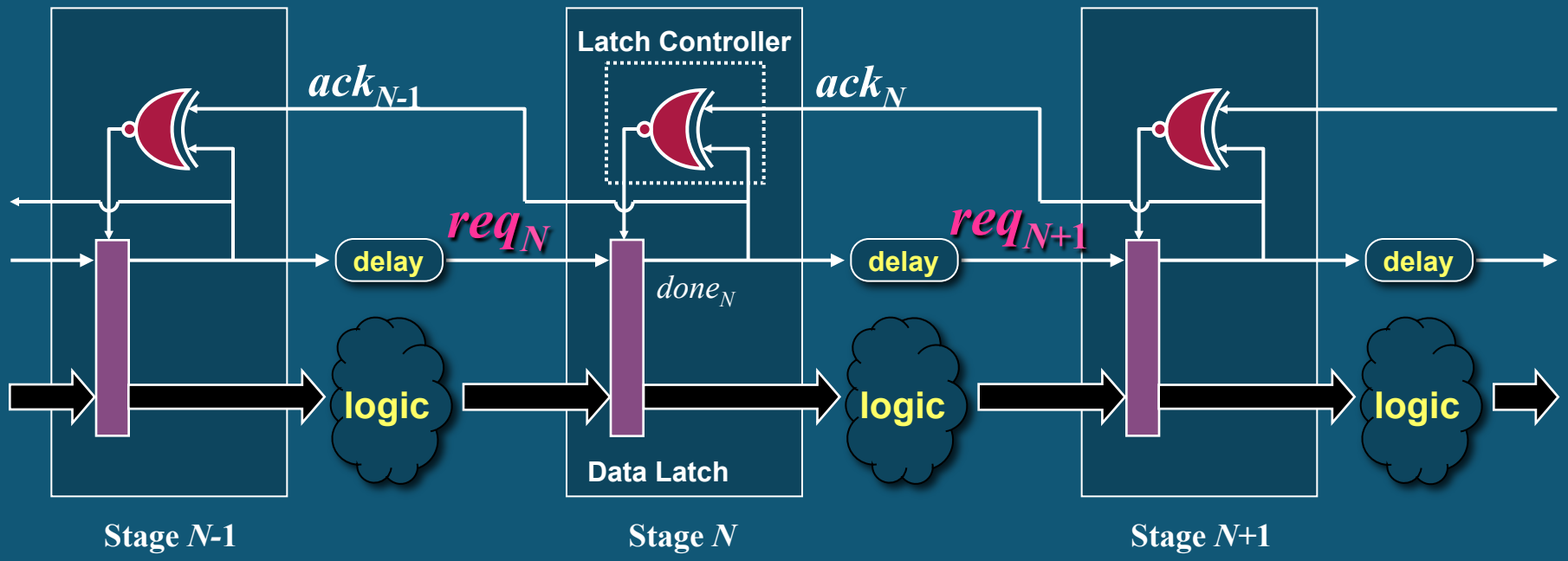


Cycle Time = 2 T_{LATCH} + T_{XNOR}

MOUSETRAP: Pipeline With Logic

Simple Extension to FIFO:

insert *logic block* + *matching delay* in each stage



Logic Blocks: can use standard single-rail (non-hazard-free)

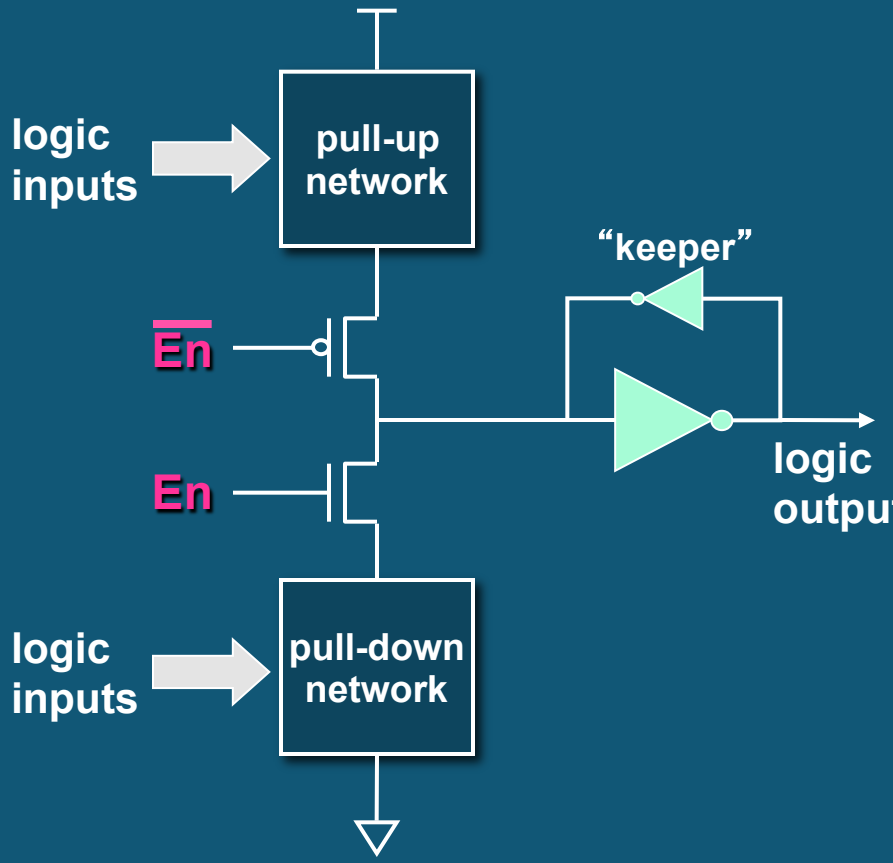
"Bundling" Requirement:

- each *"req"* must arrive *after* data inputs valid and stable

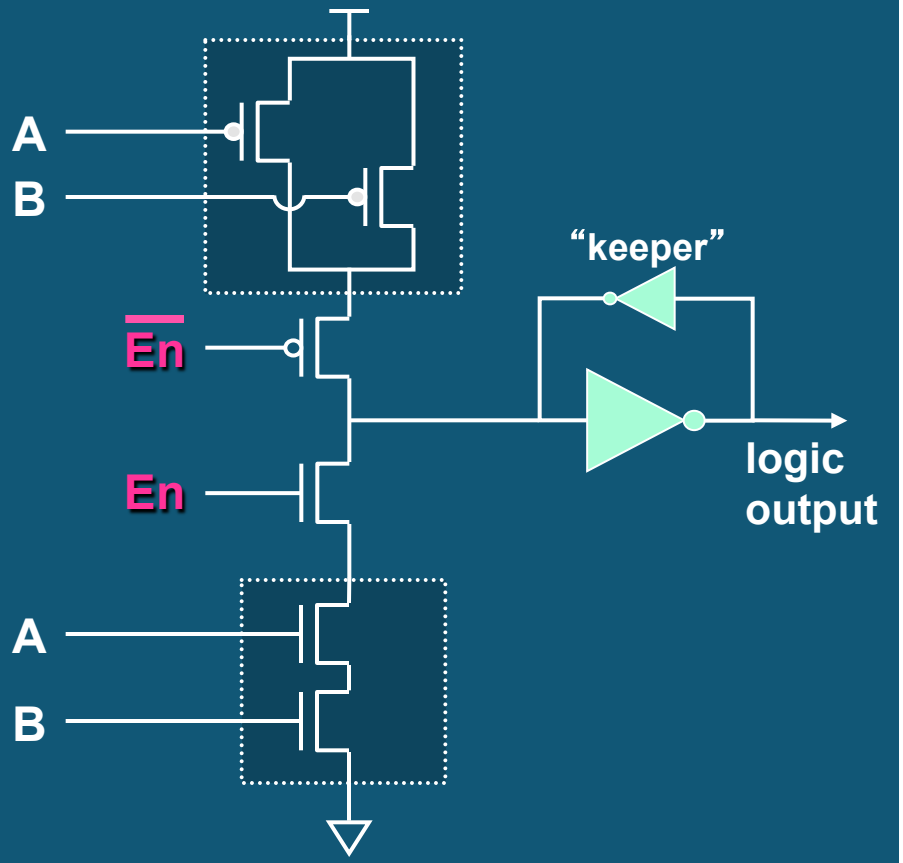
Special Case: Using "Clocked Logic"

Clocked-CMOS = C²MOS: eliminate explicit latches

- latch folded into logic itself

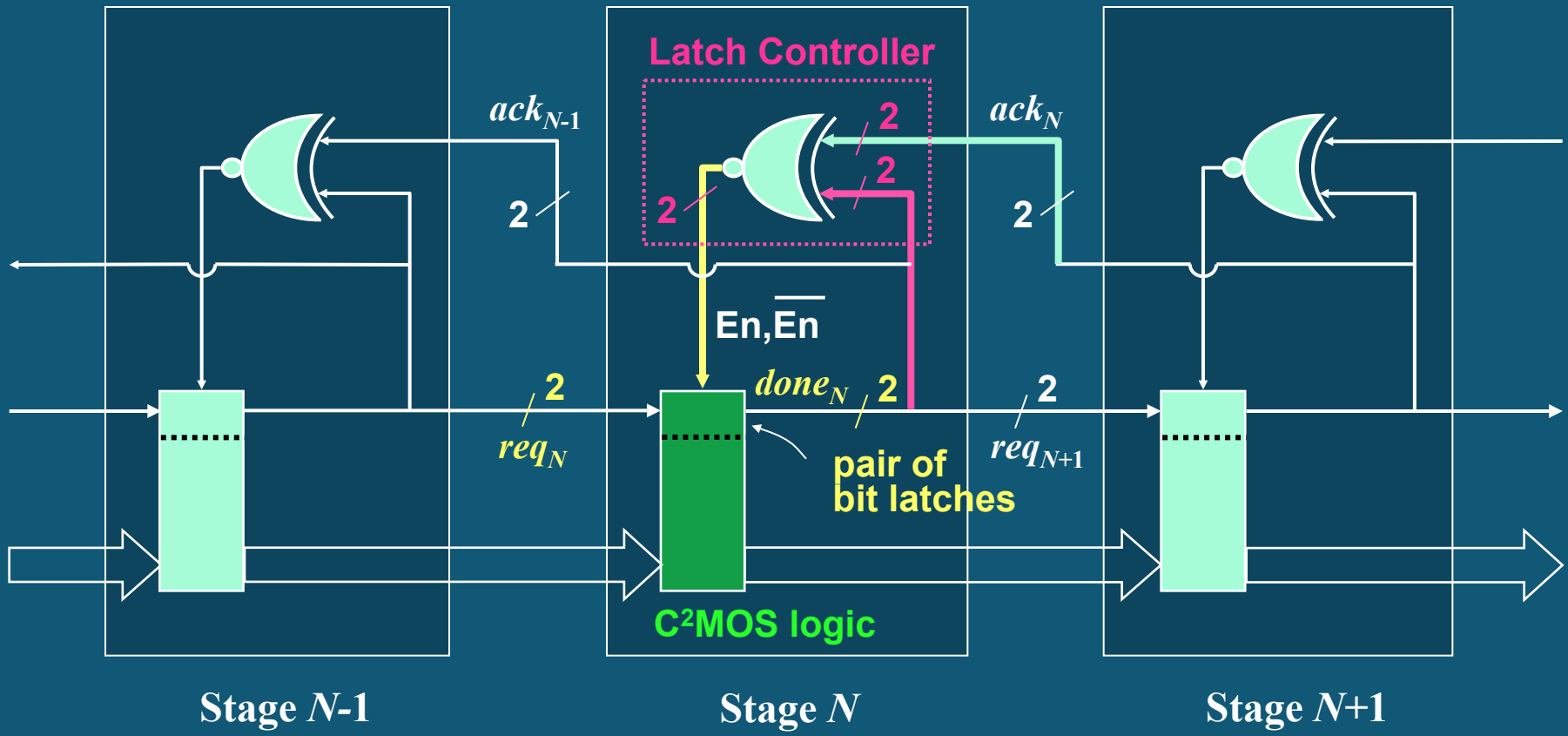


A General C²MOS gate



C²MOS AND-gate

Gate-Level MOUSETRAP: with C²MOS

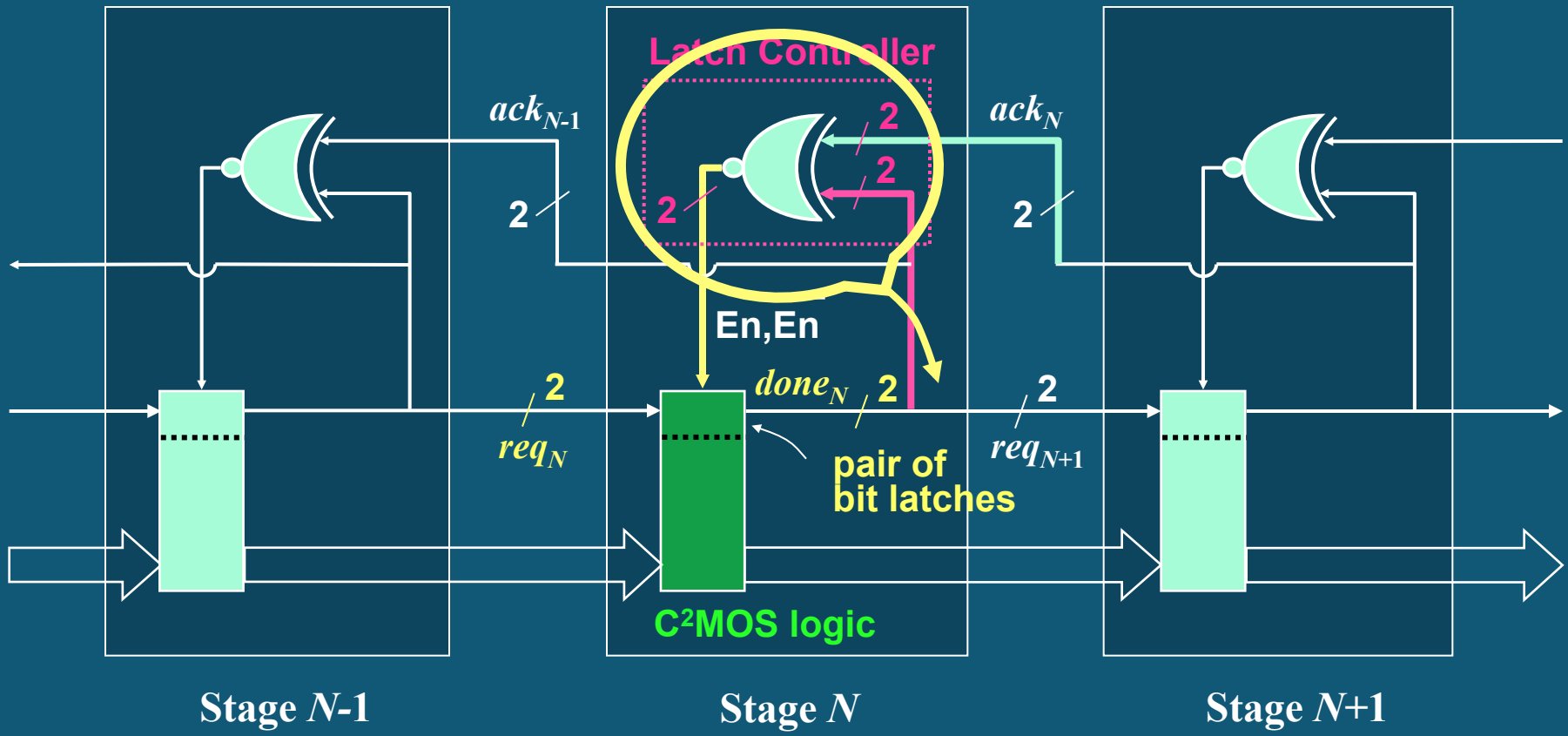


Use C²MOS: eliminate explicit latches

New Control Optimization = "Dual-Rail XNOR"

- eliminate 2 inverters from critical path

Gate-Level MOUSETRAP: with C²MOS

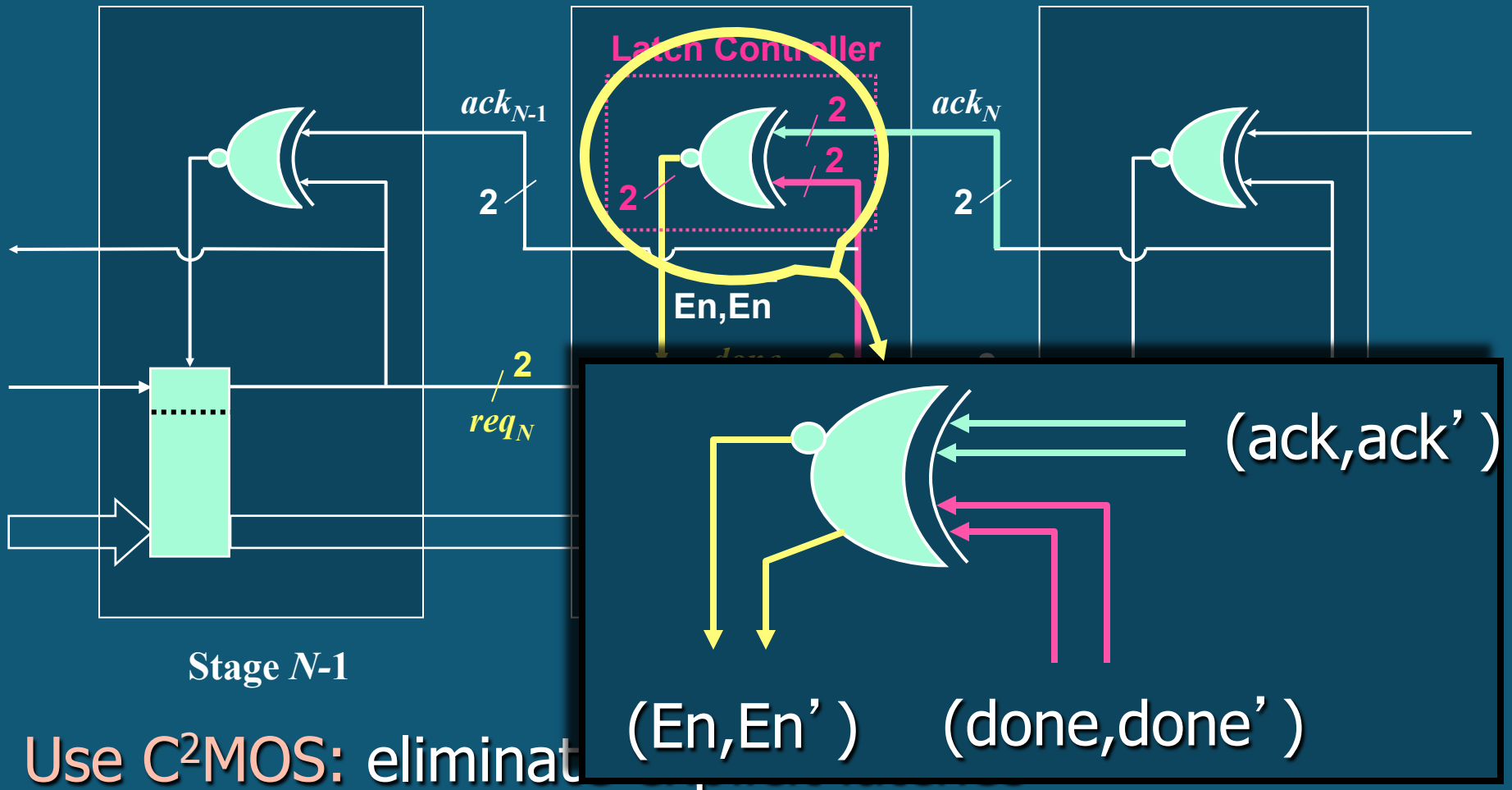


Use C²MOS: eliminate explicit latches

New Control Optimization = "Dual-Rail XNOR"

- eliminate 2 inverters from critical path

Gate-Level MOUSETRAP: with C²MOS



Use C²MOS: eliminat

New Control Optimization = "Dual-Rail XNOR"

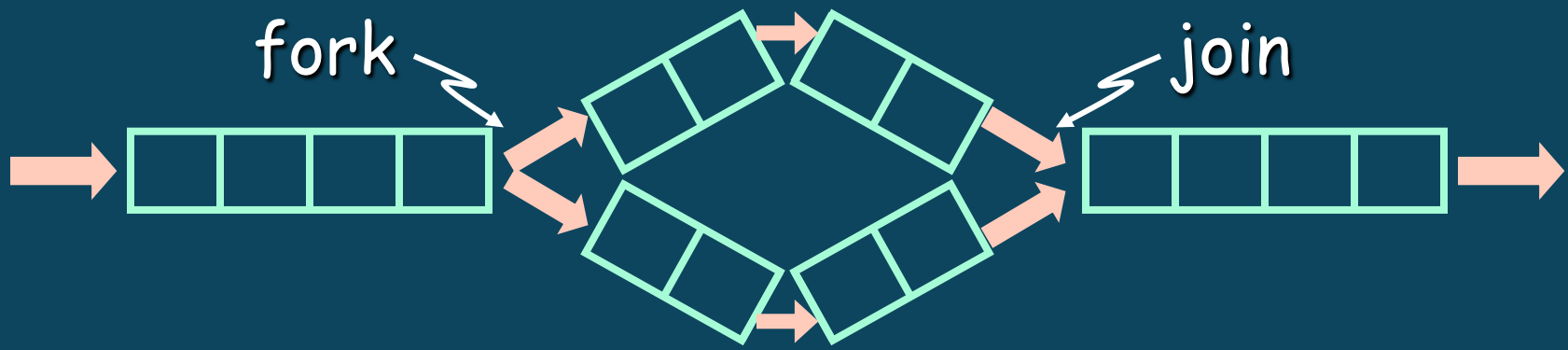
- eliminate 2 inverters from critical path

Complex Pipelining: Forks & Joins

Problems with Linear Pipelining:

- handles limited applications; real systems are more complex

Non-Linear Pipelining: has forks/joins

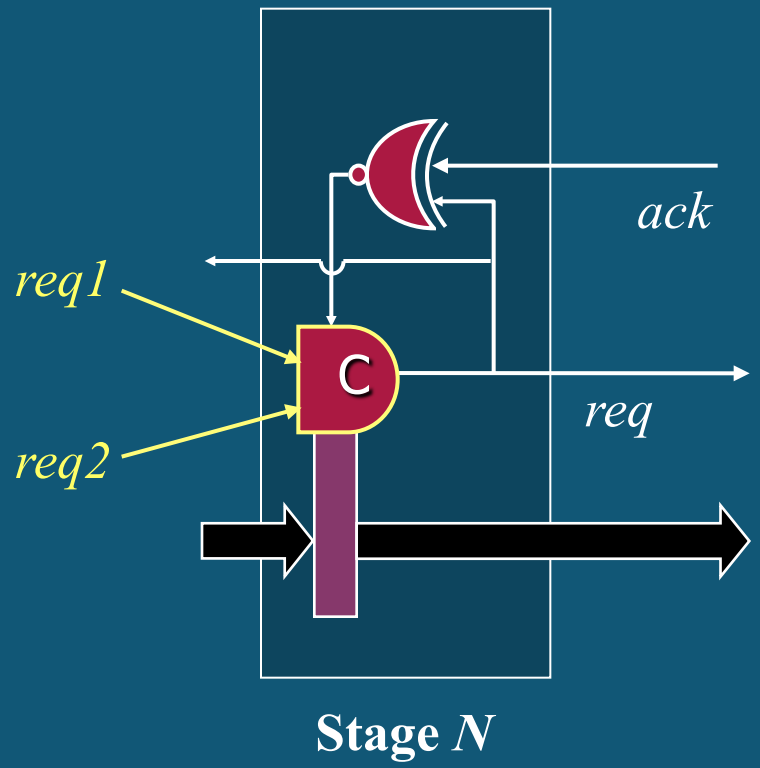


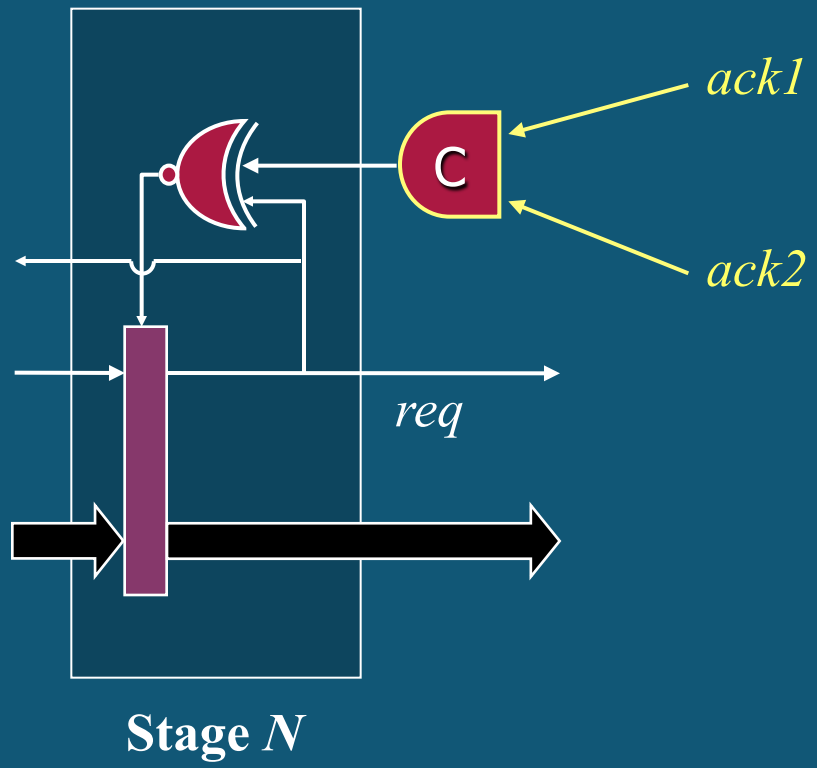
Contribution: introduce efficient circuit structures

- Forks: distribute data + control to multiple destinations
- Joins: merge data + control from multiple sources

➡ *Enabling technology for building complex async systems*

Forks and Joins: Implementation





Join: merge multiple requests

Fork: merge multiple acknowledges

Performance, Timing and Optzn.

MOUSETRAP with Logic:

Stage Latency = $T_{LATCH} + T_{LOGIC}$

Cycle Time = $2 ?T_{LATCH} + T_{XNOR} + T_{LOGIC}$

MOUSETRAP Using C²MOS Gates:

Stage Latency = T_{C^2MOS}

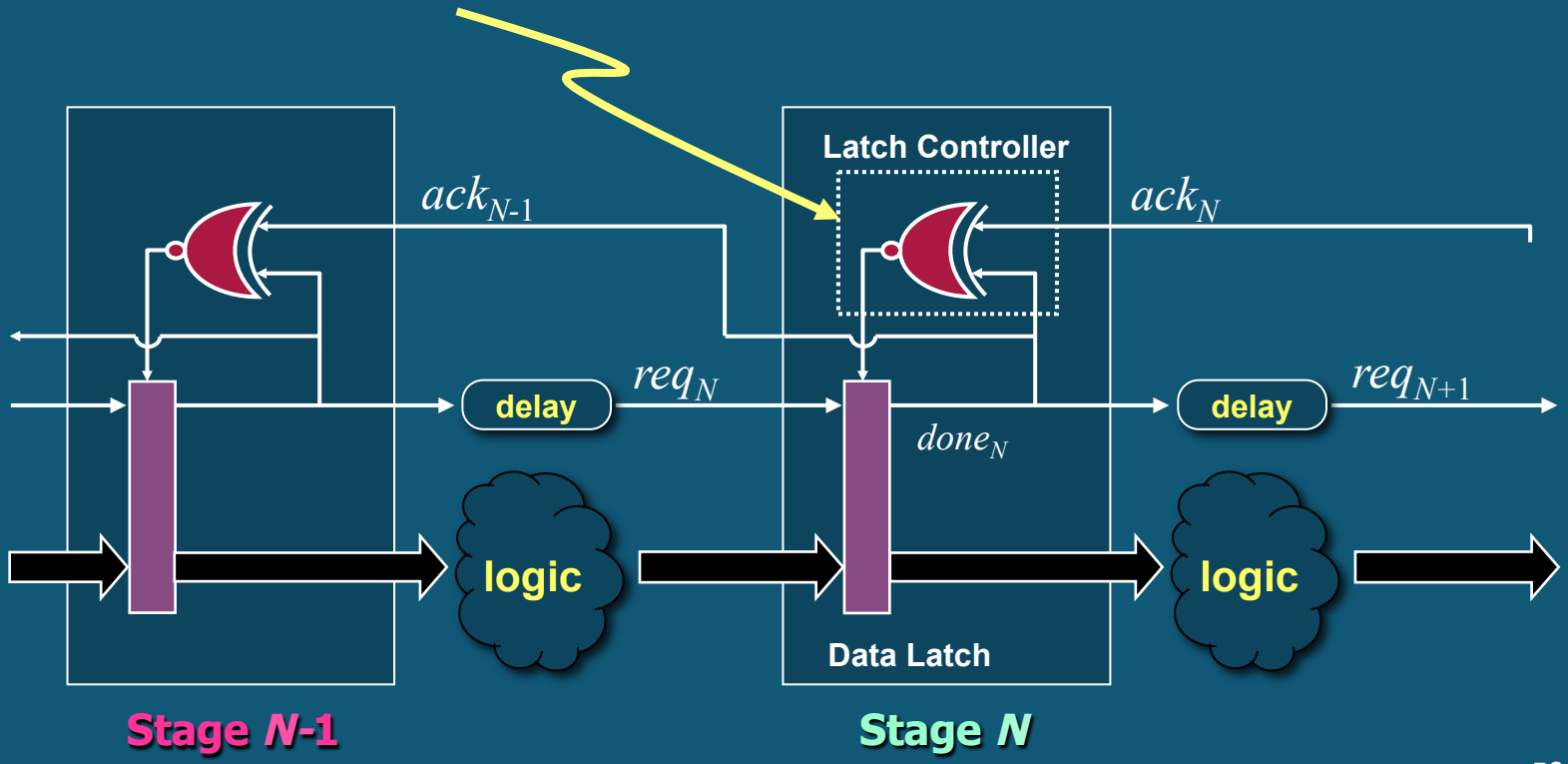
Cycle Time = $2 ?T_{C^2MOS} + T_{XNOR}$

Timing Analysis

Main Timing Constraint: avoid "data overrun"

Data must be safely "captured" by Stage N before new inputs arrive from Stage N-1

- Simple 1-sided timing constraint: fast latch disable
- Stage N's "self-loop" faster than entire path through previous stage

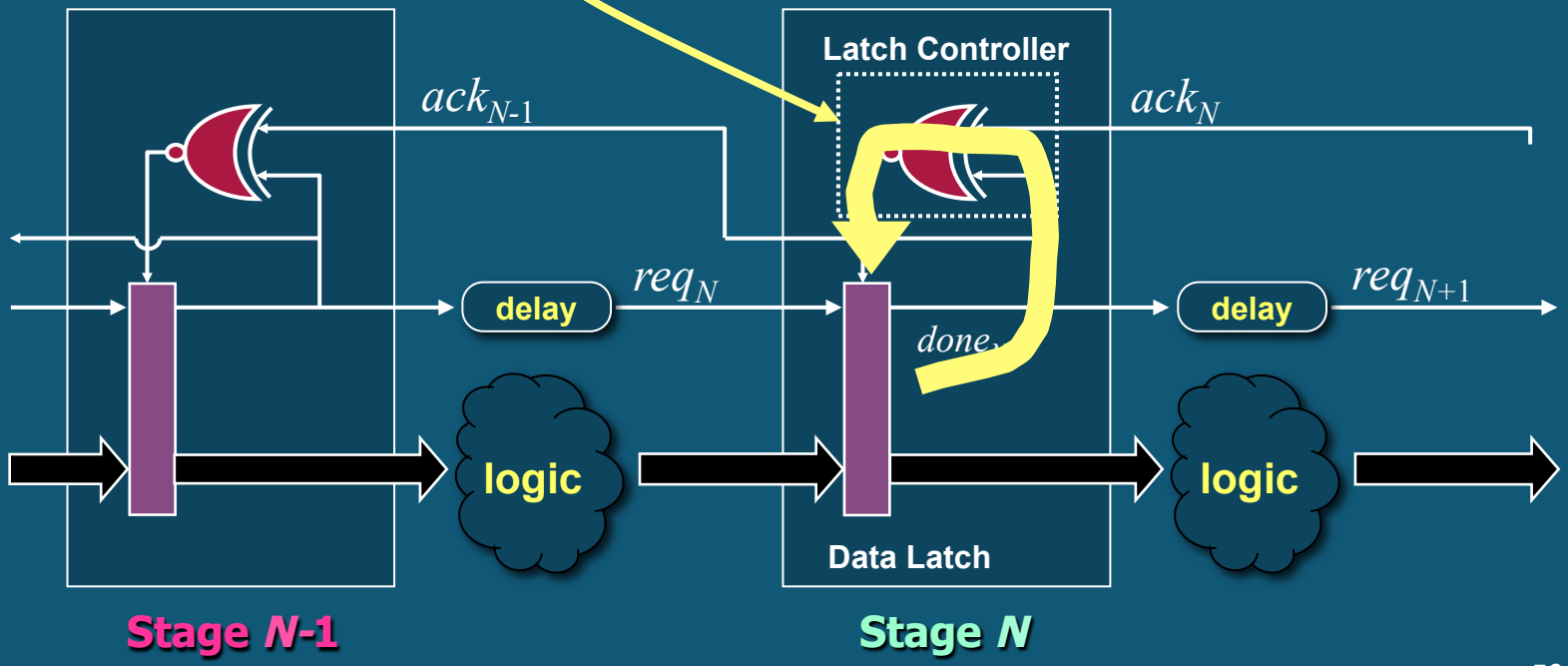


Timing Analysis

Main Timing Constraint: avoid "data overrun"

Data must be safely "captured" by Stage N before new inputs arrive from Stage N-1

- simple 1-sided timing constraint: fast latch disable
- Stage N's "self-loop" faster than entire path through previous stage

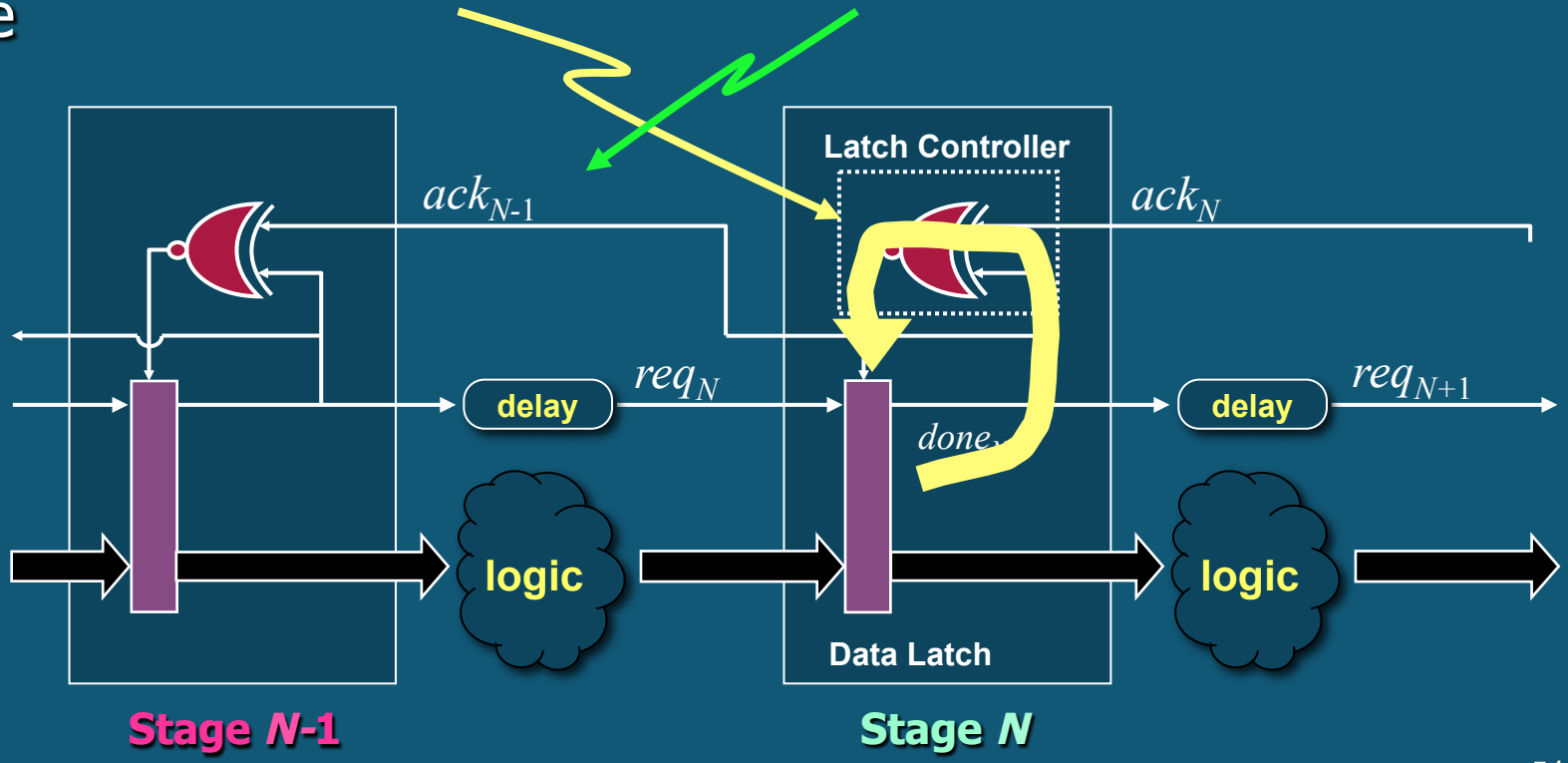


Timing Analysis

Main Timing Constraint: avoid "data overrun"

Data must be safely "captured" by Stage N before new inputs arrive from Stage N-1

- simple 1-sided timing constraint: fast latch disable
- Stage N's "self-loop" faster than entire path through previous stage

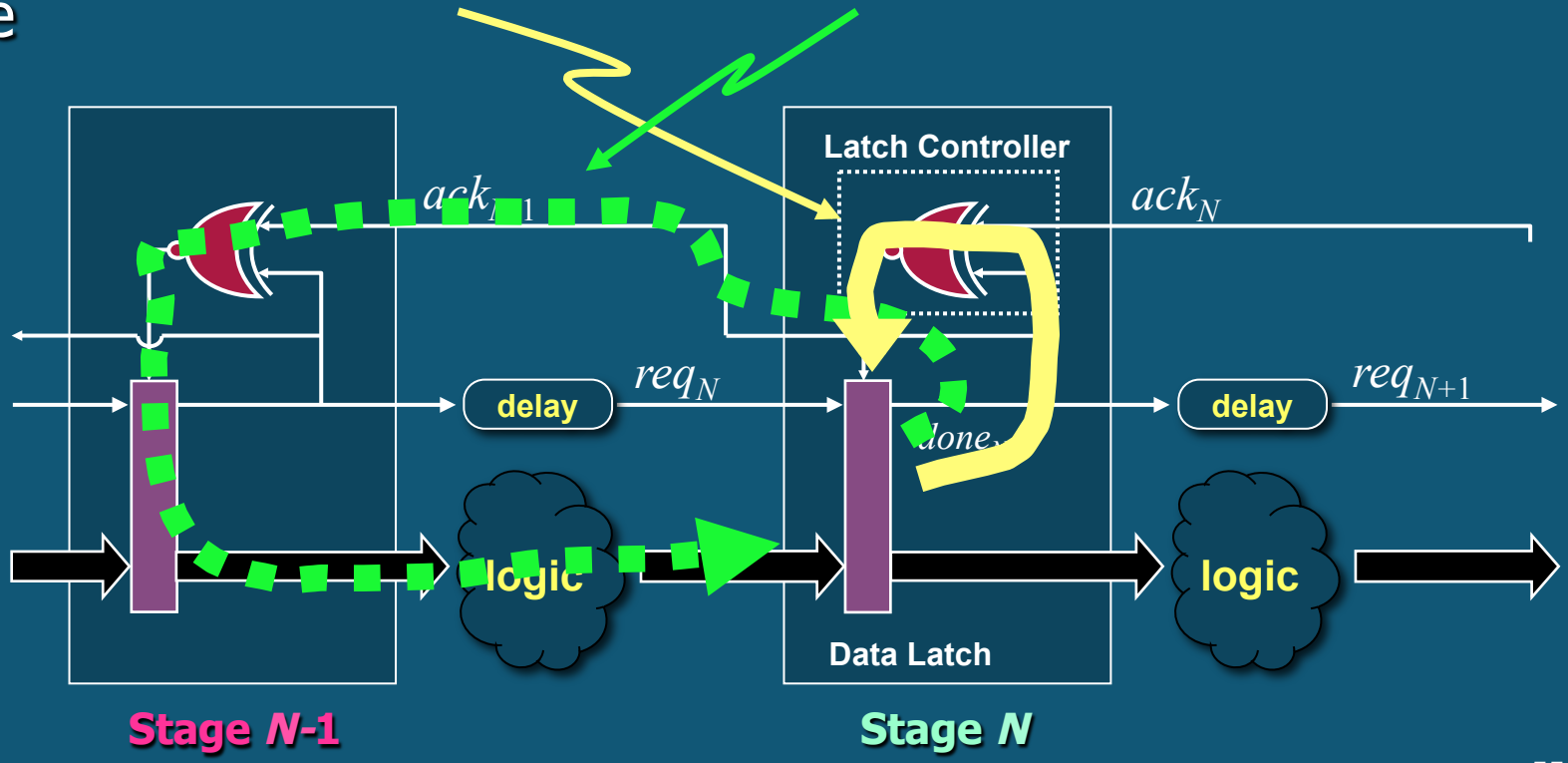


Timing Analysis

Main Timing Constraint: avoid "data overrun"

Data must be safely "captured" by Stage N before new inputs arrive from Stage N-1

- simple 1-sided timing constraint: fast latch disable
- Stage N's "self-loop" faster than entire path through previous stage



Timing Optzn: Reducing Cycle Time

Analytical Cycle Time = 2 x T_{latch} + T_{logic} + T_{XNOR}+

Goal: shorten T_{XNOR}+ (in steady-state operation)

Steady-state = no undue pipeline congestion

Observation:

- XNOR switches twice per data item: T_{XNOR}- and T_{XNOR}+
- *only 2nd (up) transition critical* for performance: T_{XNOR}+

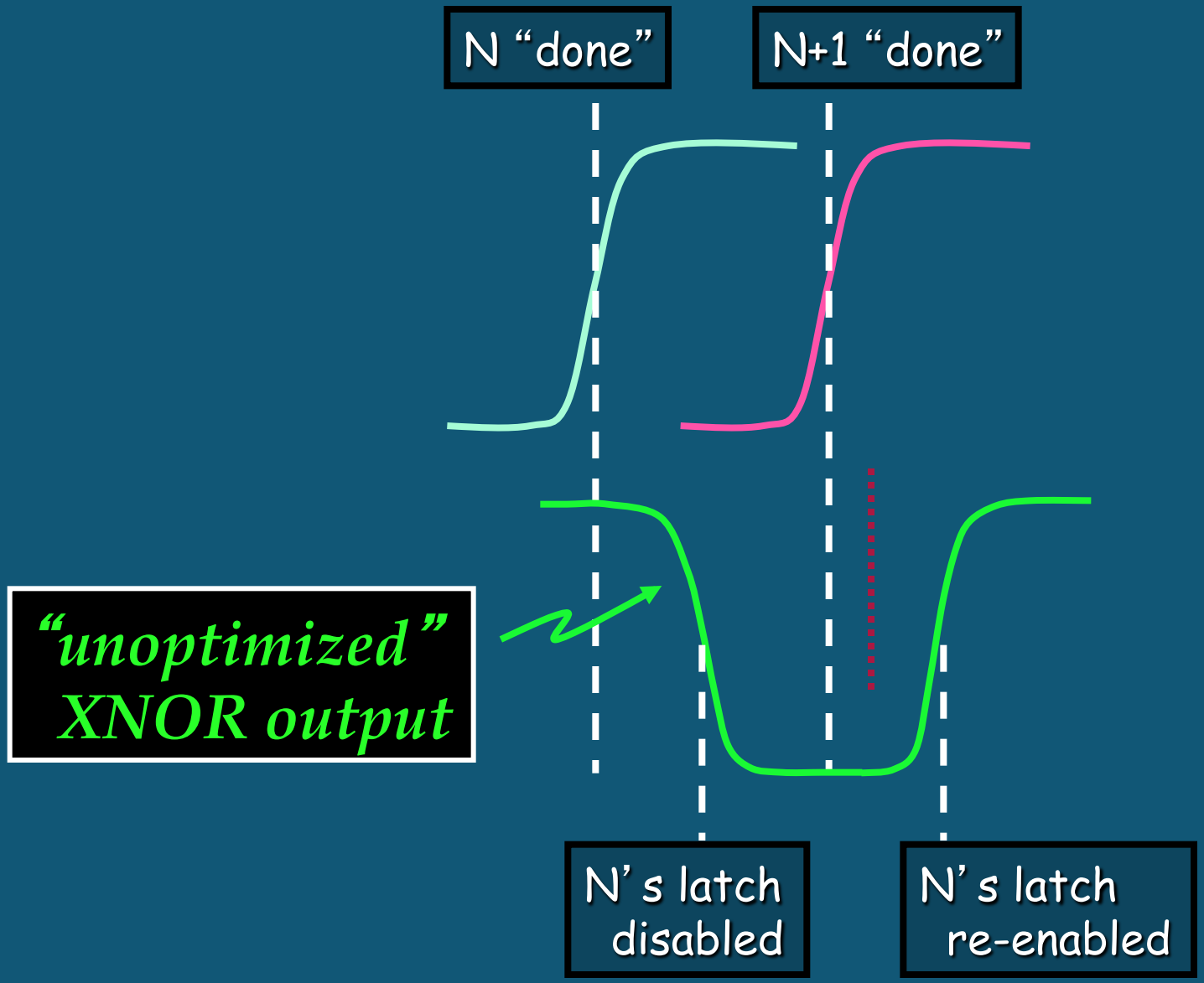
Solution: ***reduce XNOR output swing***

- degrade "slew" for start of pulse
- allows quick pulse completion: faster rise time

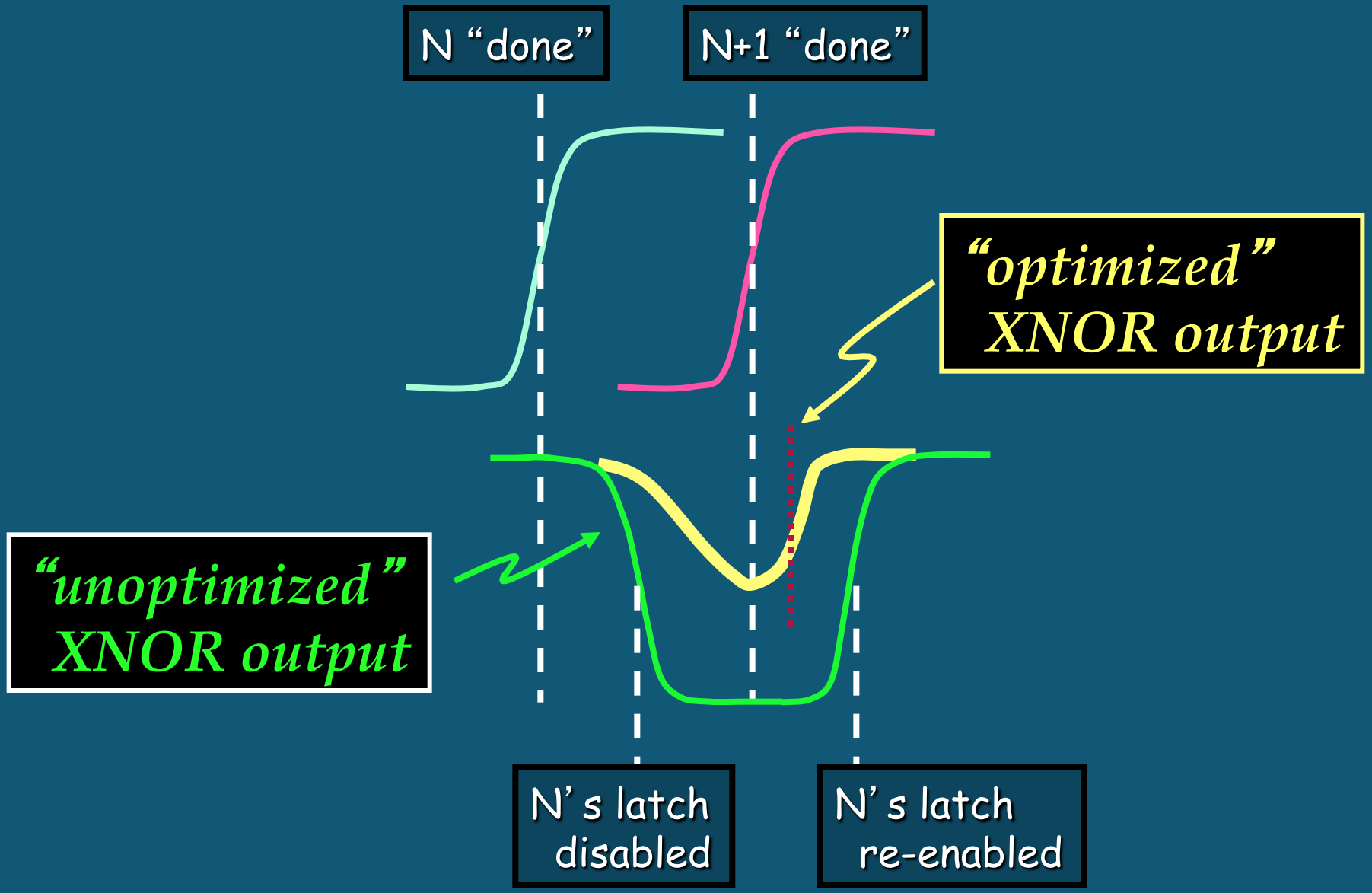
Still safe when congested: pulse starts on time

- pulse maintained until congestion clears

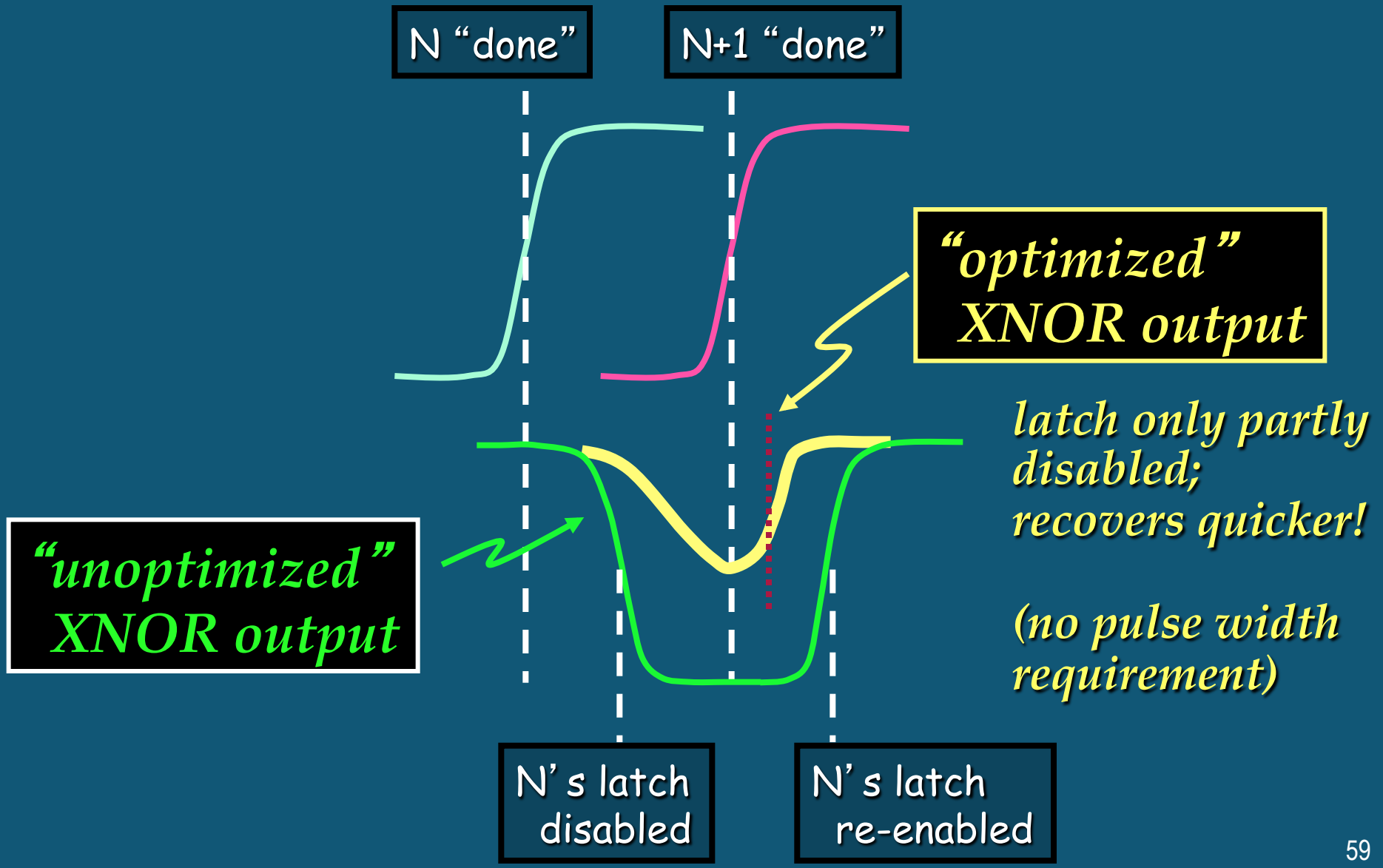
Timing Optzn (contd.)



Timing Optzn (contd.)



Timing Optzn (contd.)



Timing Issues: Handling Wide Datapaths

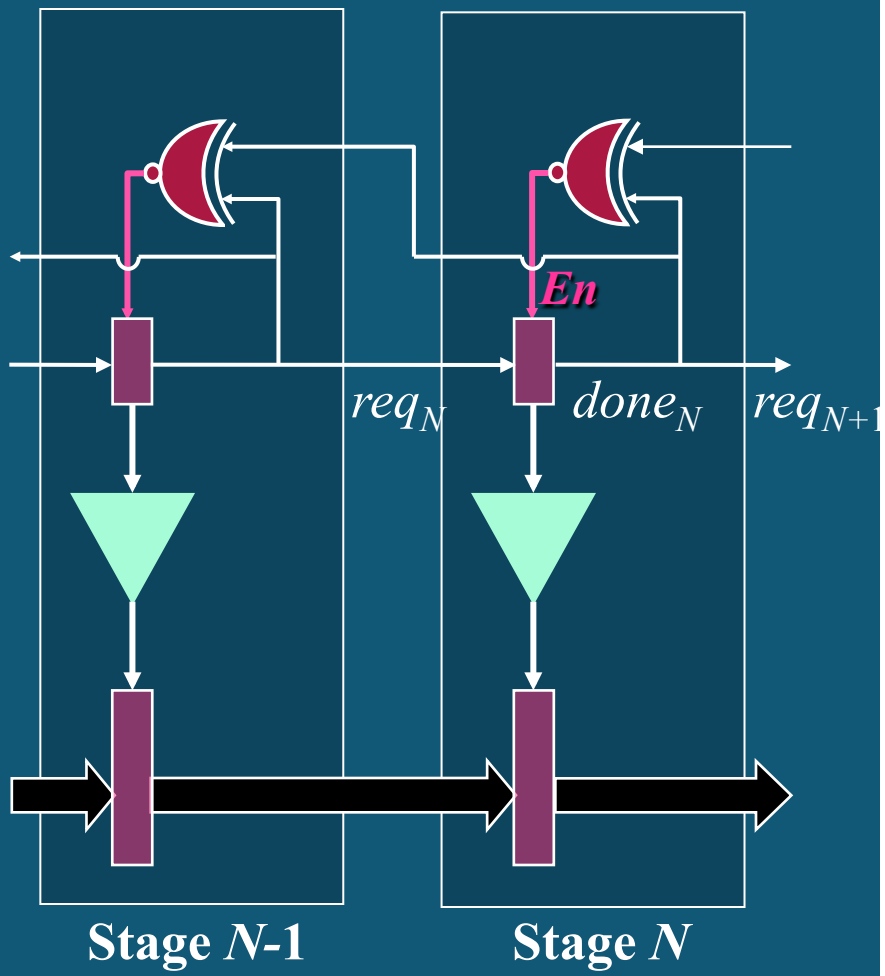
Buffers inserted to amplify latch signals *(En)*:

Reducing Impact of Buffers:

- control uses ***unbuffered*** signals
  - ➡ *buffer delay off of critical path!*
- datapath ***skewed*** w.r.t. control

Timing assumption:

buffer delays roughly equal



Experimental Results

Post-Layout Simulations of FIFO's:

- 0.18u TSMC, 1.8V, 300K temp., normal process corner

Basic MOUSETRAP: 2.10 GigaOps/sec

Optimized MOUSETRAP *(waveform shaping):* 2.38 GigaOps/sec.

Related Work: Synchronous Pipelines

- A number of advanced high-speed approaches:
  - Wave-pipelining (Wong ’93, Hauck ’99)
  - Clock-delayed domino (Yee/Sechen ’96)
  - C2MOS (Borah et al. ’97)
  - Wave-steered YADDs (Mukherjee ’99)
  - Skew-tolerant/self-resetting domino (Harris/Horowitz’97)
  - Clock-skew scheduling (Kourtev/Friedman ’99)

- Drawbacks:
  - lack elasticity/difficult to verify
  - require careful delay-management
  - still require high-speed clock distribution

Related Work: Asynchronous Pipelines

- Sutherland (Sun '89): micropipelines
  - elegant but expensive (and slow) 2-phase pipelines
- Sun Labs (Molnar, Sutherland et al. '97-01):
  - GasP: fast design (1.5 GHz in 0.35μ), but drawbacks…:
    - fine-grain transistor sizing to achieve delay equalization
    - need extensive post-layout simulation to verify complex timing constraints
- IBM's IPCMOS (Schuster et al. ISSCC'00):
  - currently fabricated in 0.18μ: 3.3 GHz, but drawbacks…:
    - very complex timing requirements + circuit structures
    - must avoid short-circuit scenarios

Related Work: Asynchronous Pipelines (cont.)

- Basic Dynamic Pipelines (Williams '86, Martin '97):
  - benefits: no explicit latches, low latency
  - drawbacks: poor cycle time *("throughput-limited")*
- Advanced Dynamic Pipelines (Singh/Nowick '00):
  - 2 styles: "lookahead pipelines" (LP), "high-capacity pipelines" (HC)
  - high throughput: FIFOs – 3.5 GHz (in 0.18μ)
  - IBM interest: fabricated 4 experimental designs (0.18μ)

Comparison with Wave Pipelining

Two Scenarios:

- Steady State:
  - both MOUSETRAP and wave pipelines act like transparent "flow through" combinational pipelines
- Congestion:
  - right environment stalls: each MOUSETRAP stage safely captures data
  - internal right stage slow: MOUSETRAP stages to left safely capture data

➡ *congestion properly handled in MOUSETRAP*

Conclusion: MOUSETRAP has potential of…

- speed of wave pipelining
- greater robustness and flexibility

Related Protocols

Day/Woods (’97), and Charlie Boxes (’00)

Similarities: all use…

- transition signaling for handshakes
- phase conversion for latch signals

Differences: MOUSETRAP has…

- higher throughput
- ability to handle *fork/join* datapaths
- more aggressive timing, less insensitivity to delays

Conclusions and Future Work

Introduced new asynchronous pipeline style:

- Targets static logic implementation
- Simple latches and control:
  - transparent latches, or C^2MOS gates
  - single gate control = 1 XNOR gate/stage
- Highly concurrent event-driven protocol
- High throughputs obtained:
  - 2.10-2.38 Gops/sec in 0.18μ
  - comparable to wave pipelines; yet more robust/less design effort
- Correctly handle *forks* and *joins* in datapaths
- Timing constrains: local, 1-sided, easily met

Ongoing Work:

- Applications: complex chips (Boeing applications)
- CAD tools: integrating into Philips-based experimental flow (DARPA "CLASS" project)