Beyond classical circuit design lecture 10

Clocked Design & Pipelining (Clocked & Asynchronous)

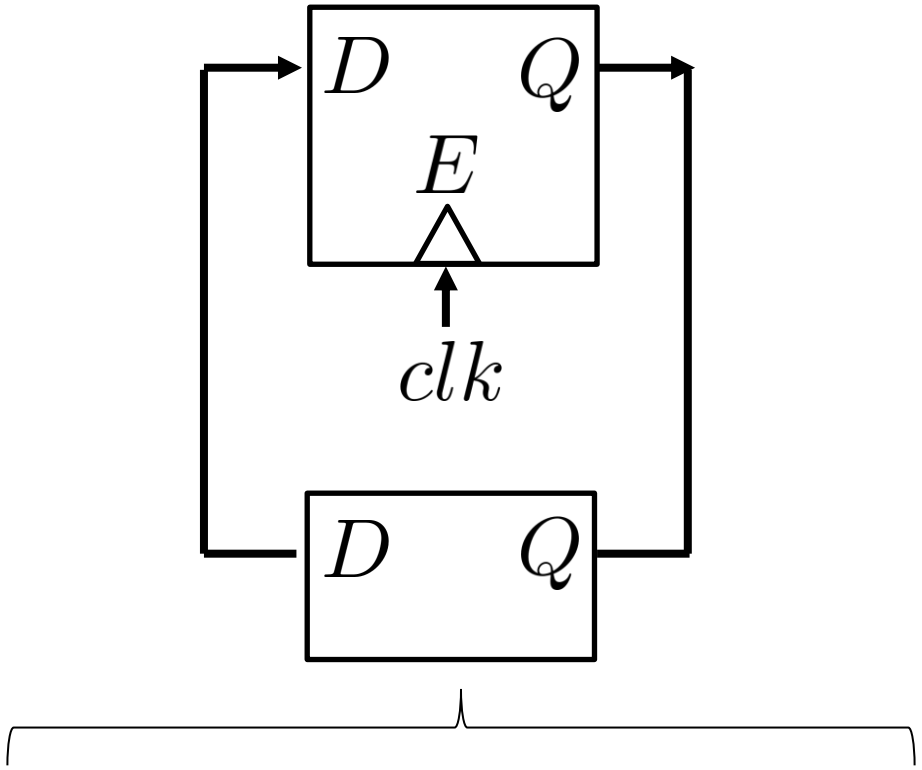
Further Reading

Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic: *Digital Integrated Circuits. A Design Perspective. 2nd edition.* Prentice Hall, 2003.

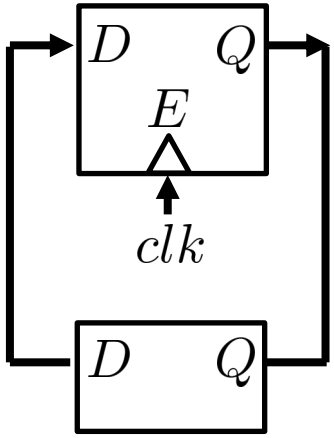
Ivan E. Sutherland: *Micropipelines.* Commun. ACM 32, 6 (June 1989), 720-738.

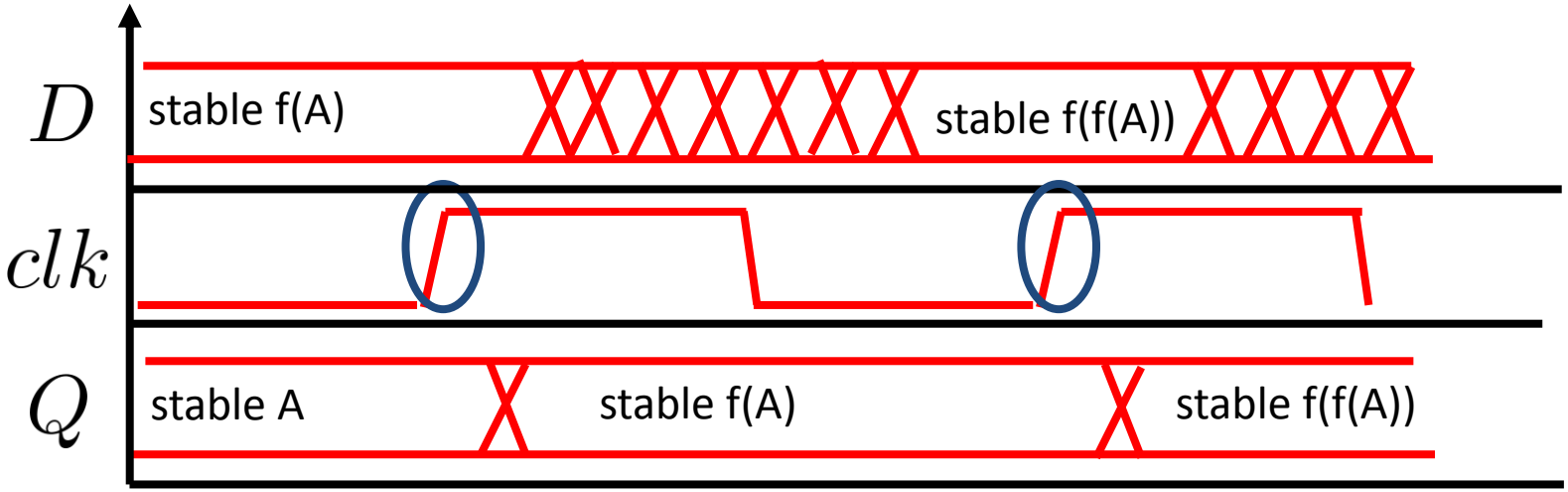
Steven M. Nowick, Montek Singh: *High-Performance Asynchronous Pipelines: An Overview.* Design & Test of Computers, IEEE , vol.28, no.5, pp.8,22, Sept.-Oct. 2011

Clocked Design

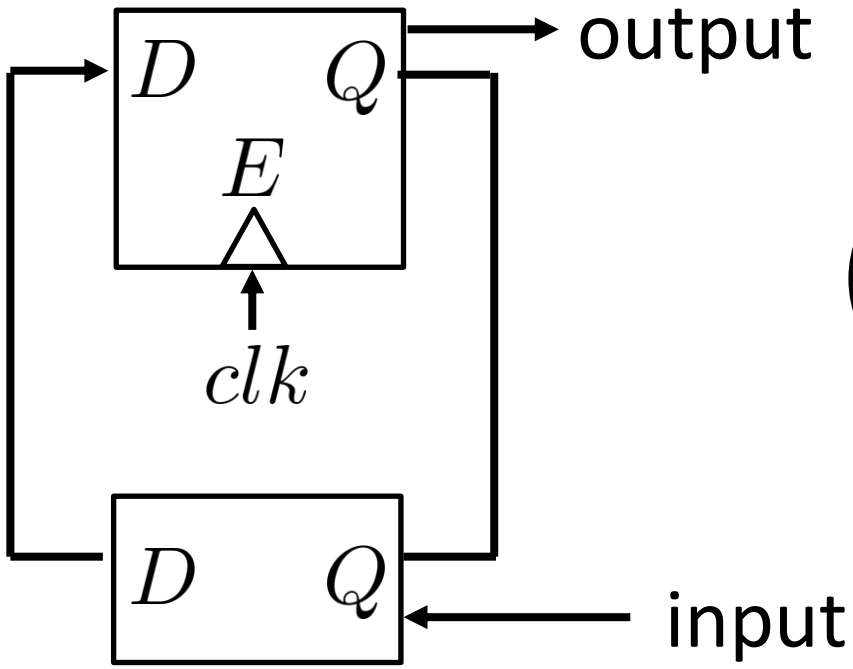


Clocked Design

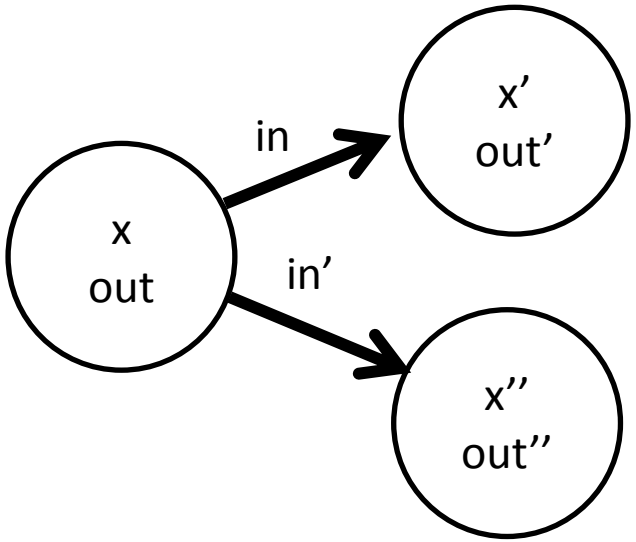




Clocked Design in Environment

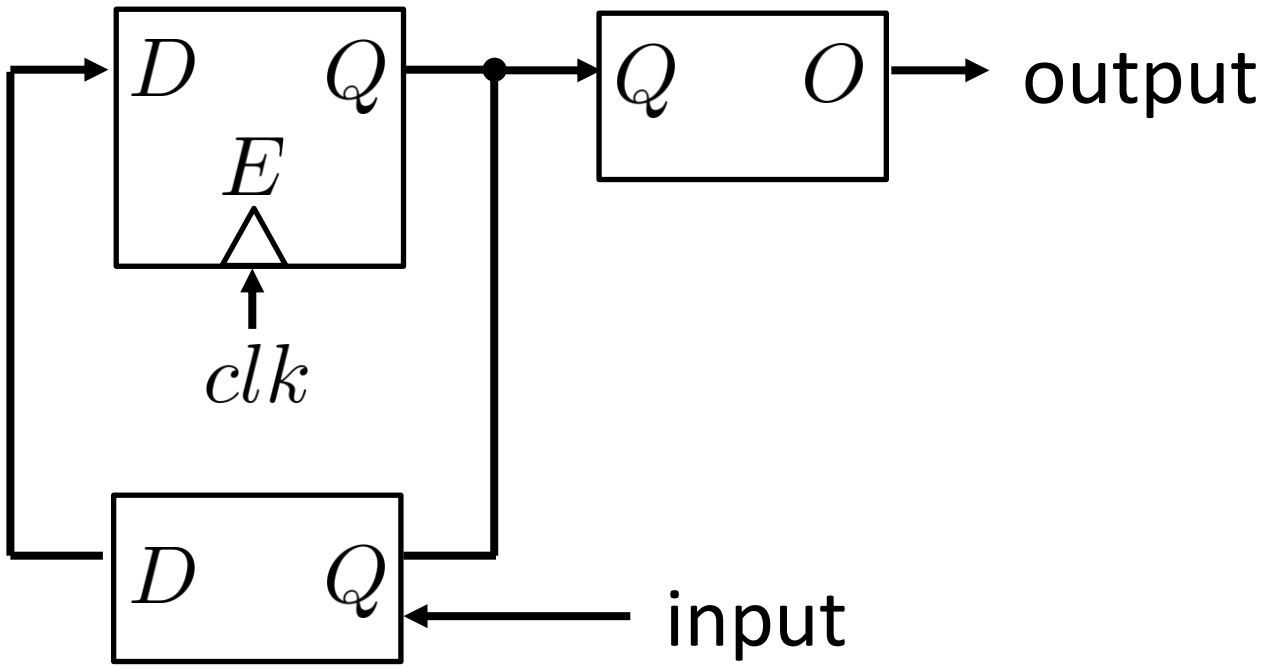


next state logic



Moore machine

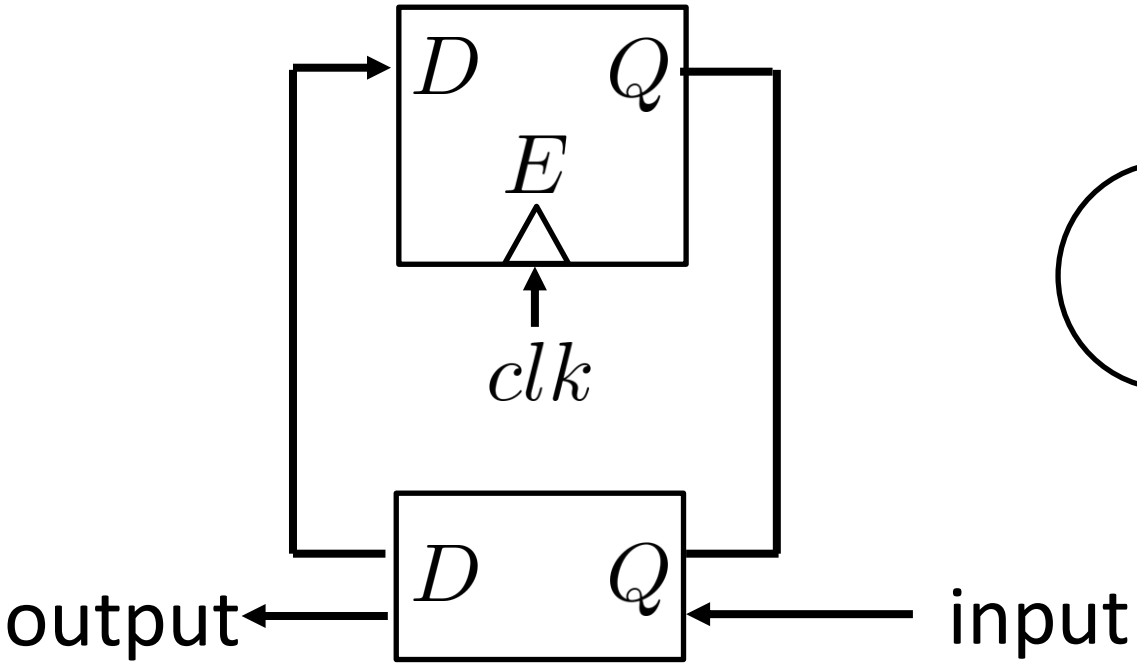
Variants



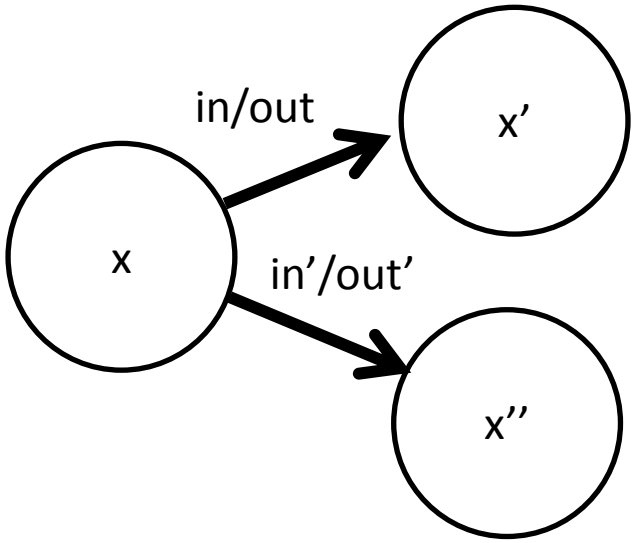
next state logic

Moore machine

Variants

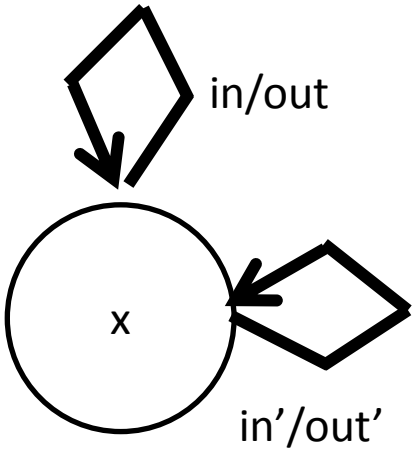


next state logic

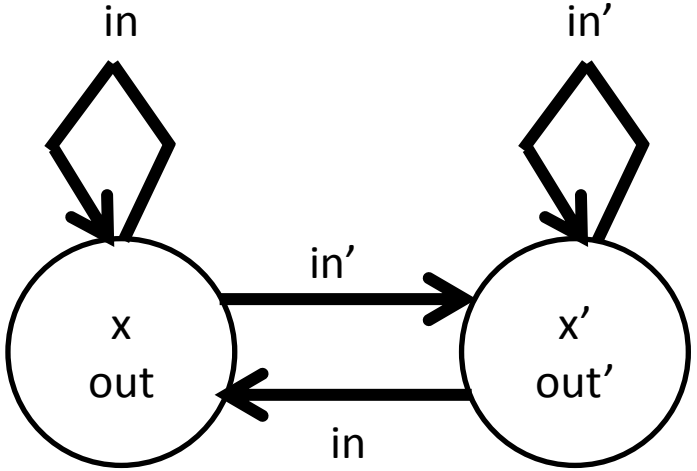


Mealy machine

Variants



versus



Mealy -> Moore: split states per output

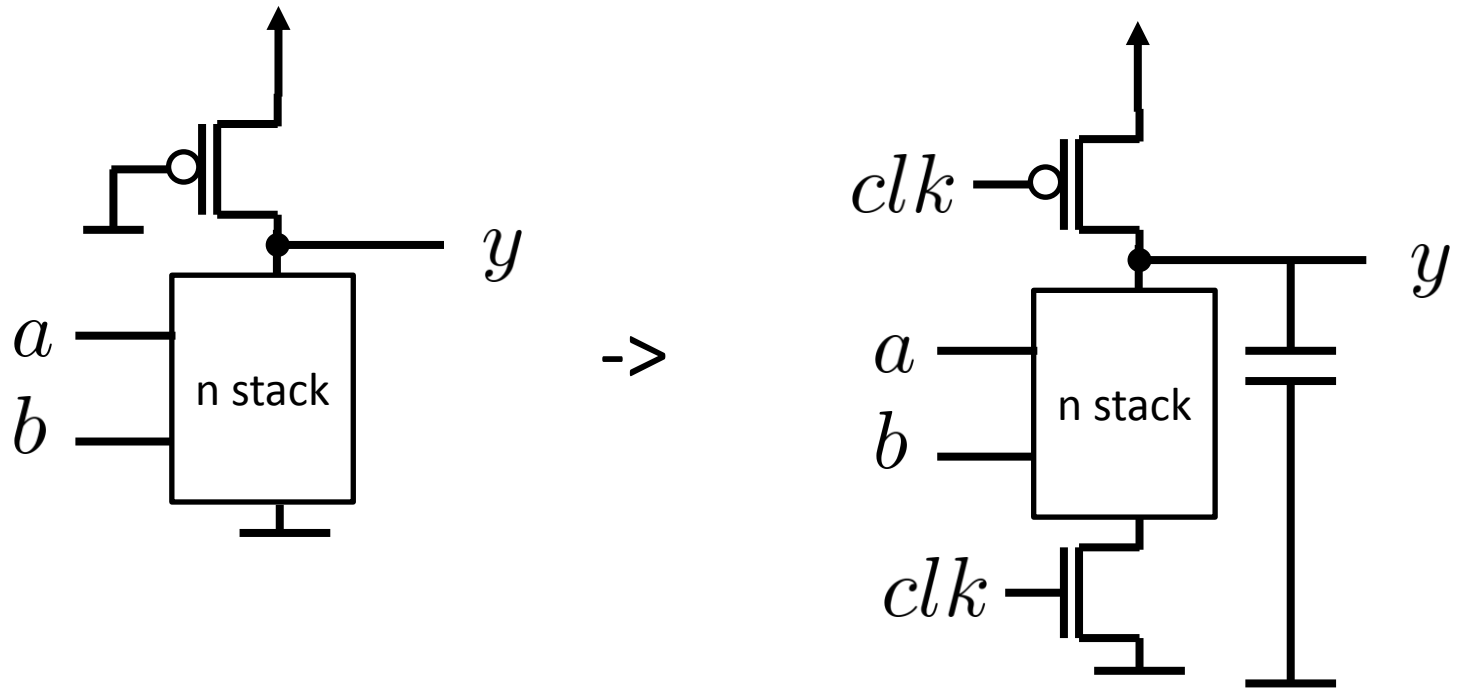
Beyond classical circuit design lecture 10.5

Dynamic CMOS

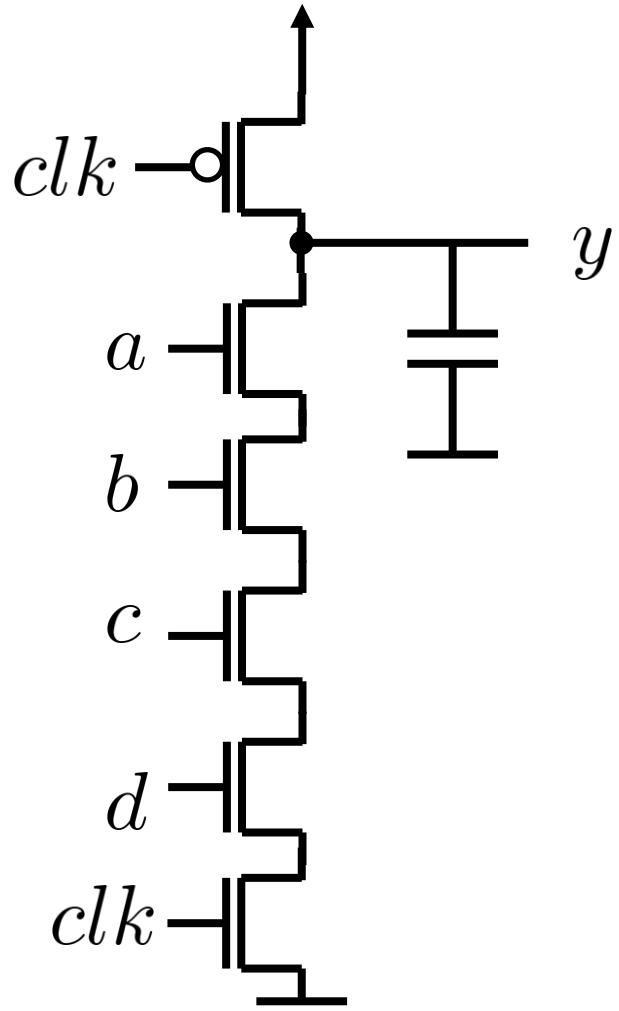
Dynamic CMOS

Like in pseudo nMOS: remove slow pMOS stack.

But:

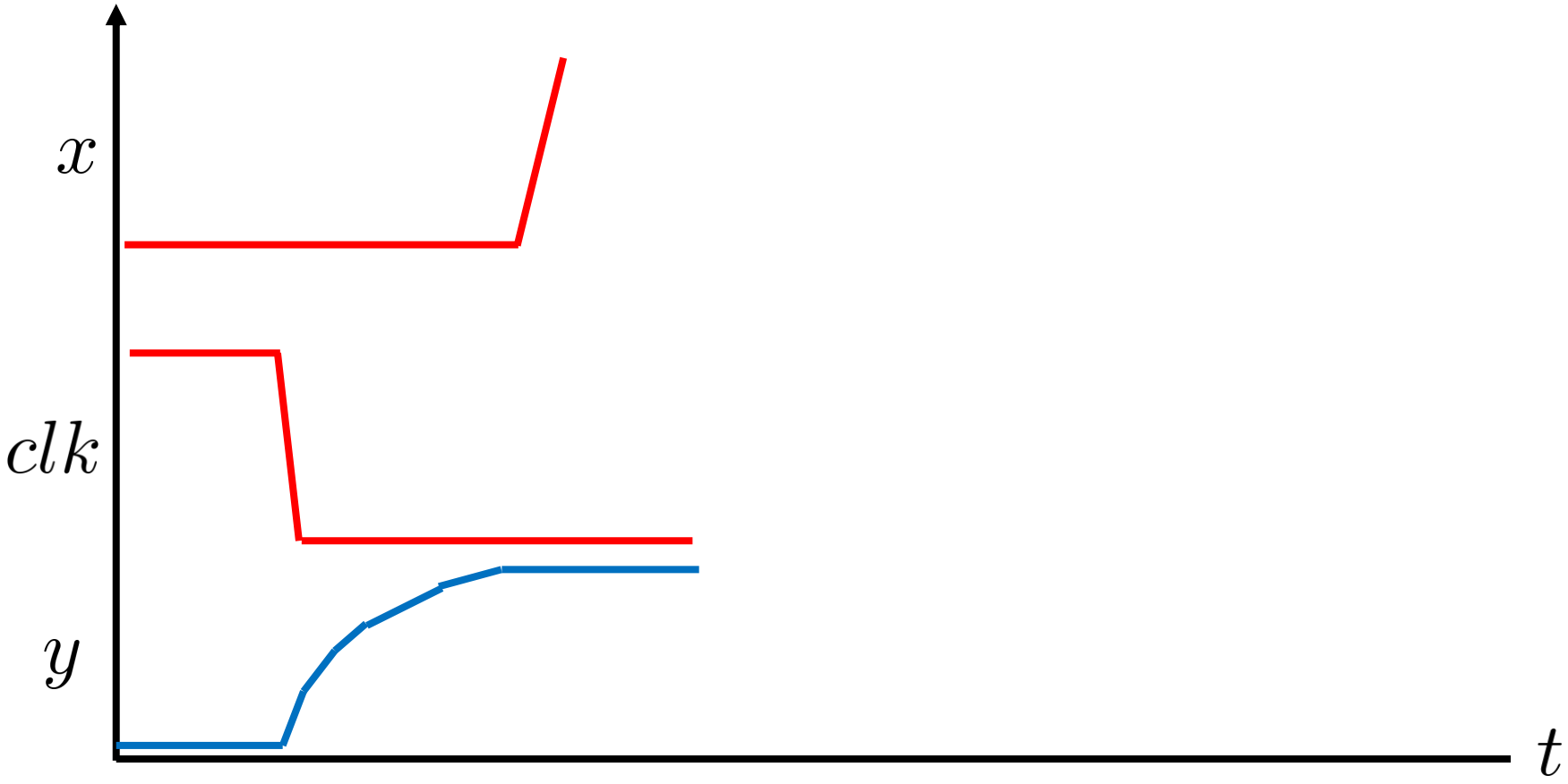


Dynamic CMOS 4NAND



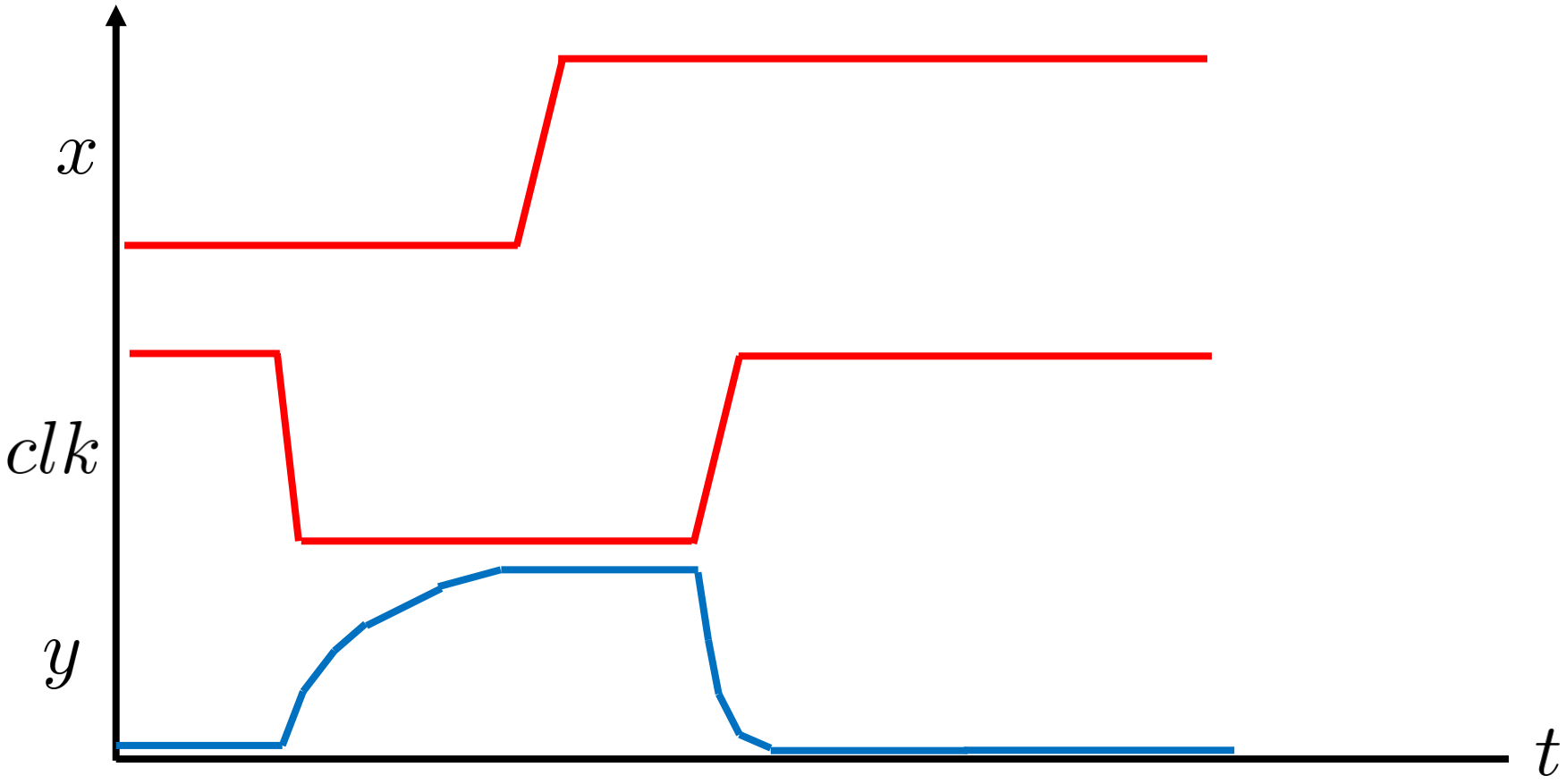
Dynamic CMOS INV

1. Precharge output & apply input (here 1)



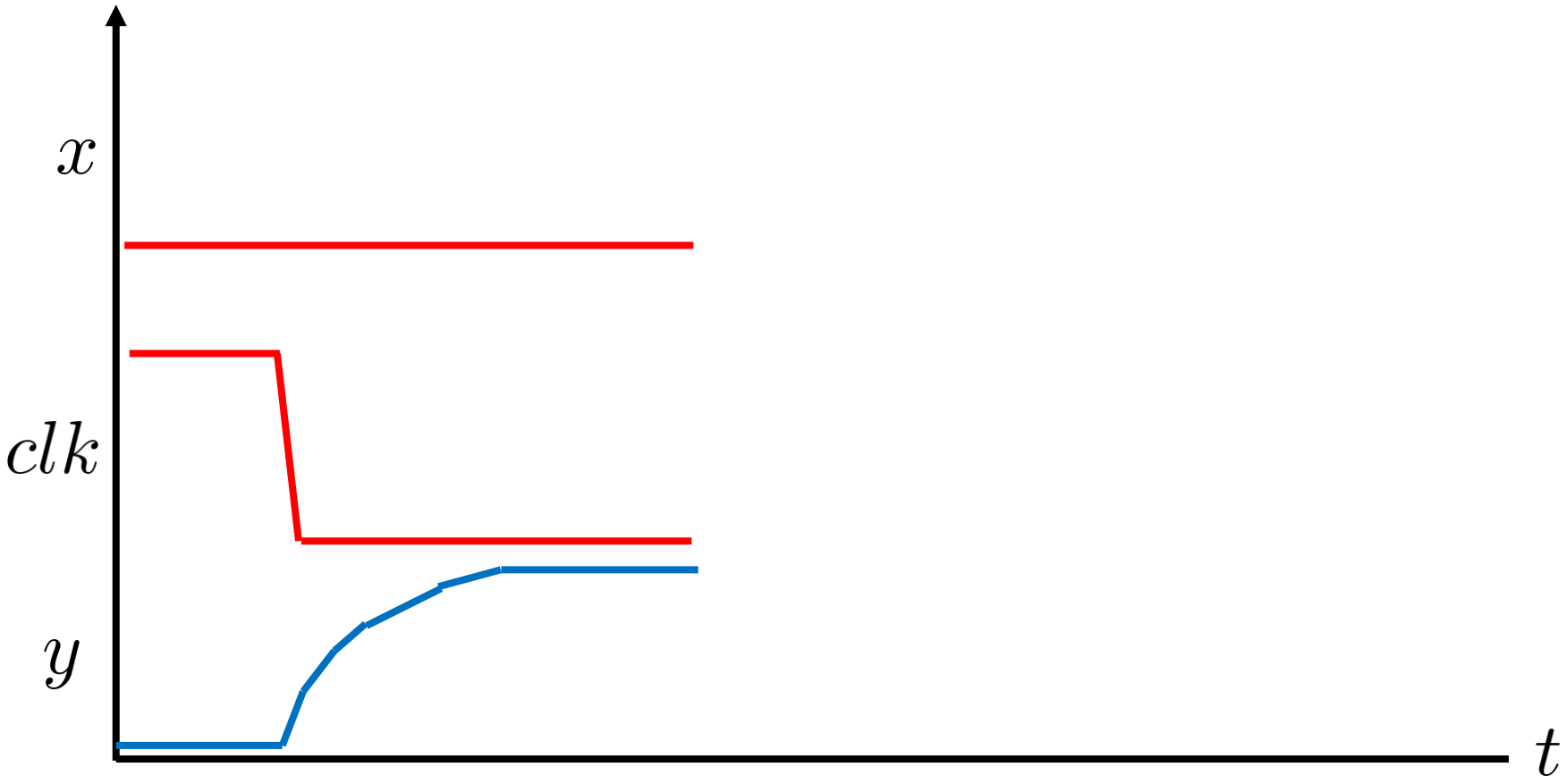
Dynamic CMOS INV

2. Evaluate at active clock transition



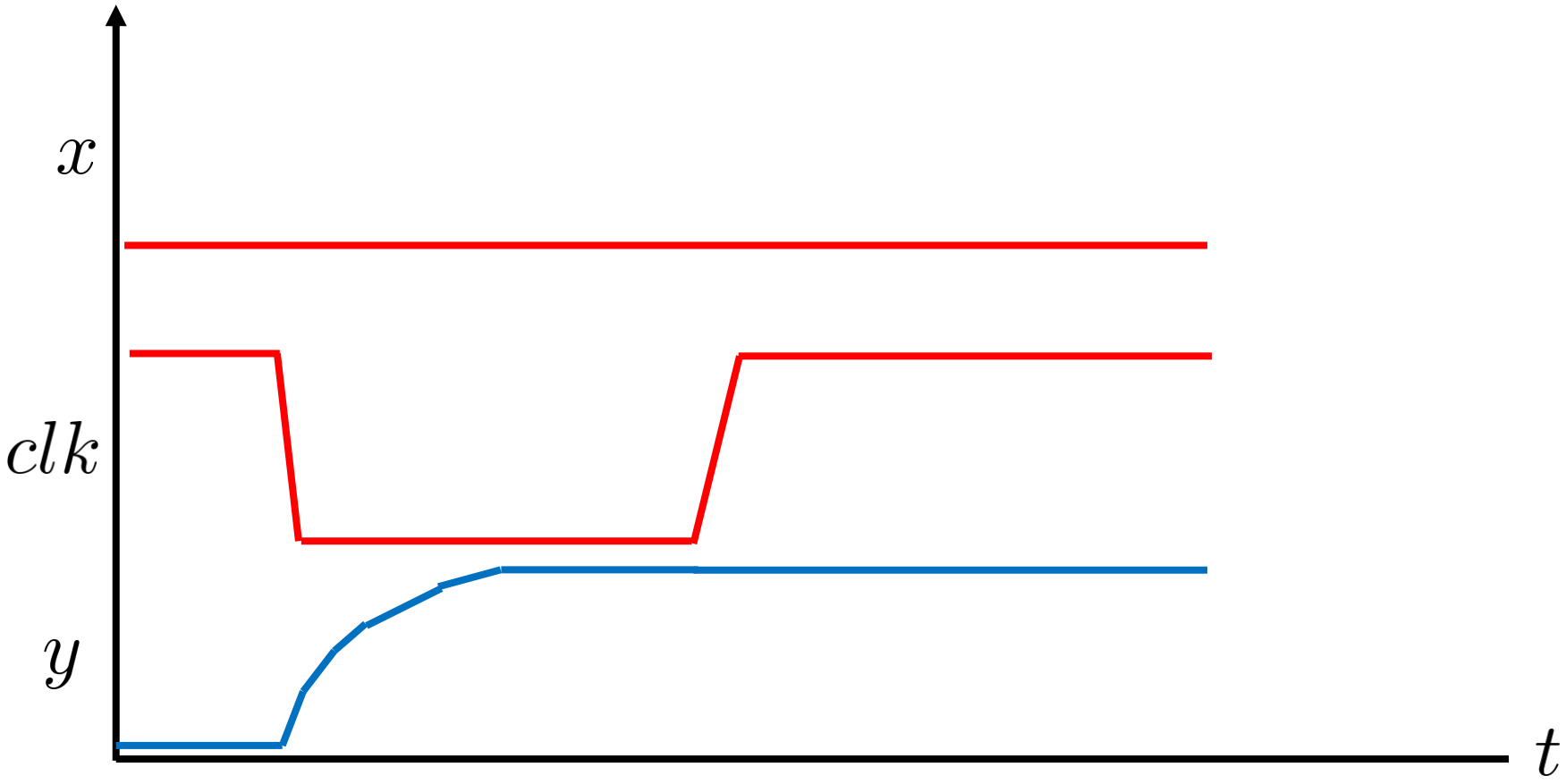
Dynamic CMOS INV

1. Precharge output & apply input (here 0)



Dynamic CMOS INV

2. Evaluate at active clock transition



Dynamic CMOS Properties

Glitches:

+ only transitions once from 1->0, or not at all

Speed:

+ very fast (only n stack & smaller load & 0 delay for 1 output)

Dynamic CMOS Properties

Power:

+ typically fewer transistors

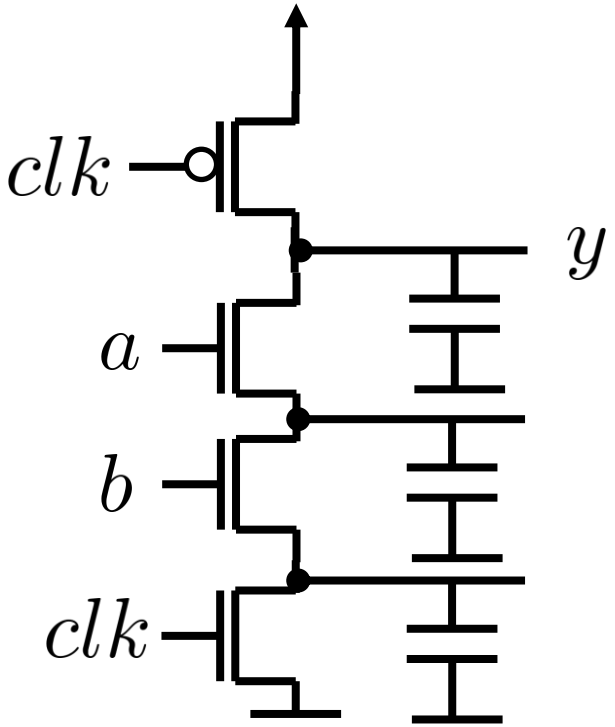
+ only one transistor driven (n stack) instead of 2 (n stack and p stack)

+ only transitions once from 1->0, or not at all

- potentially higher switching rate (!)

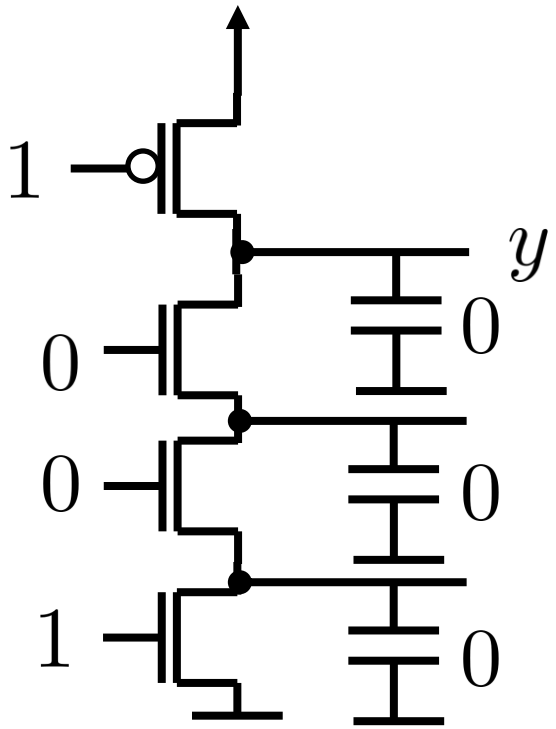
- clock transistor always switches

Charge Sharing

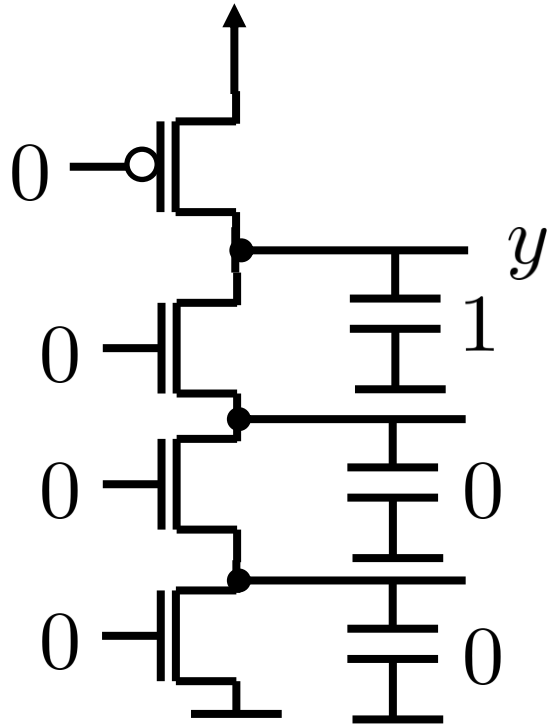


Charge Sharing

1. evaluate

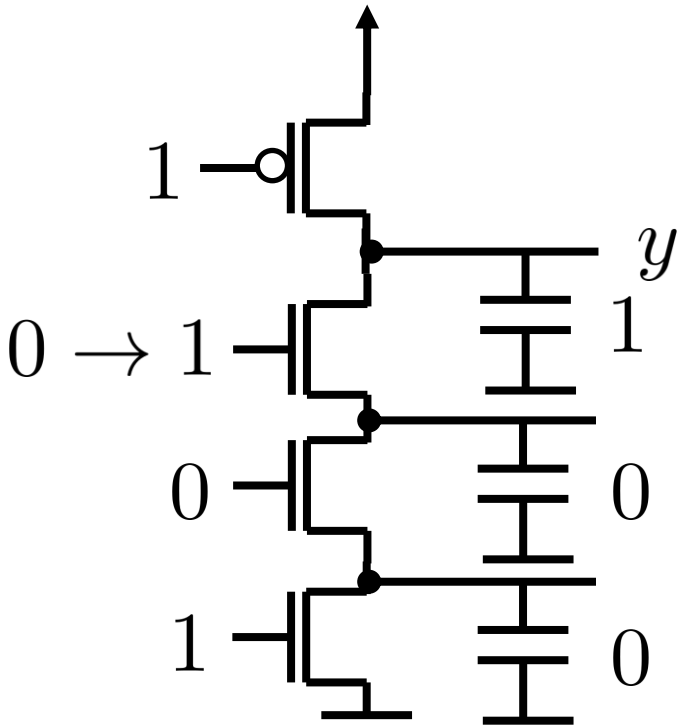


2. next charging

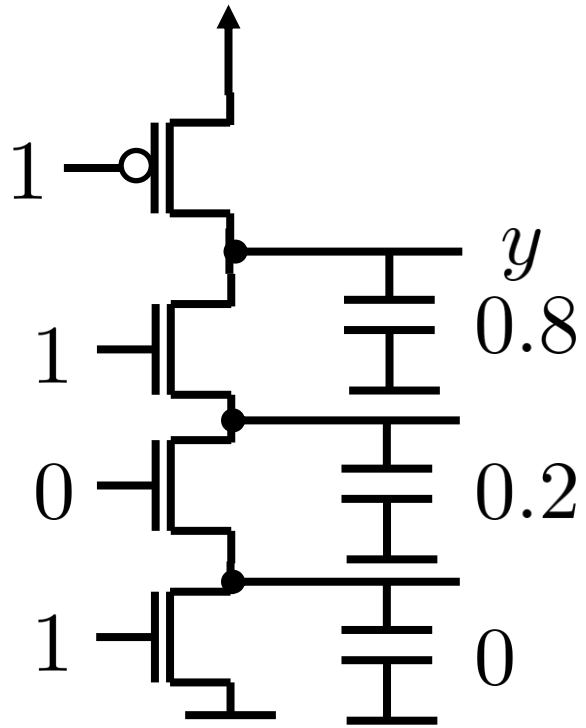


Charge Sharing

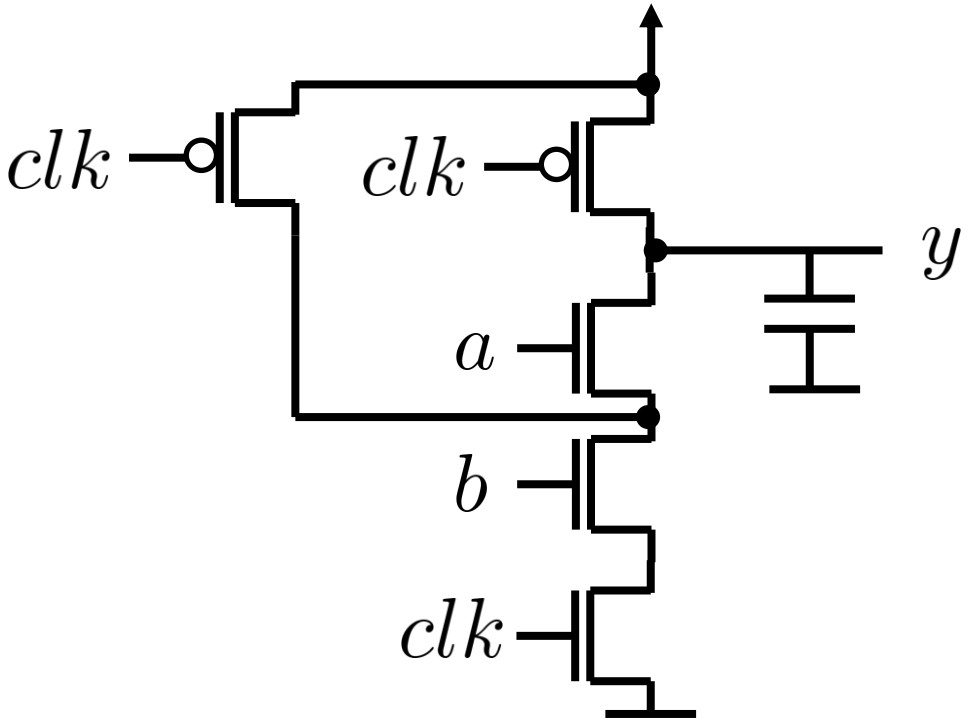
3. evaluate



3’. at end of evaluation

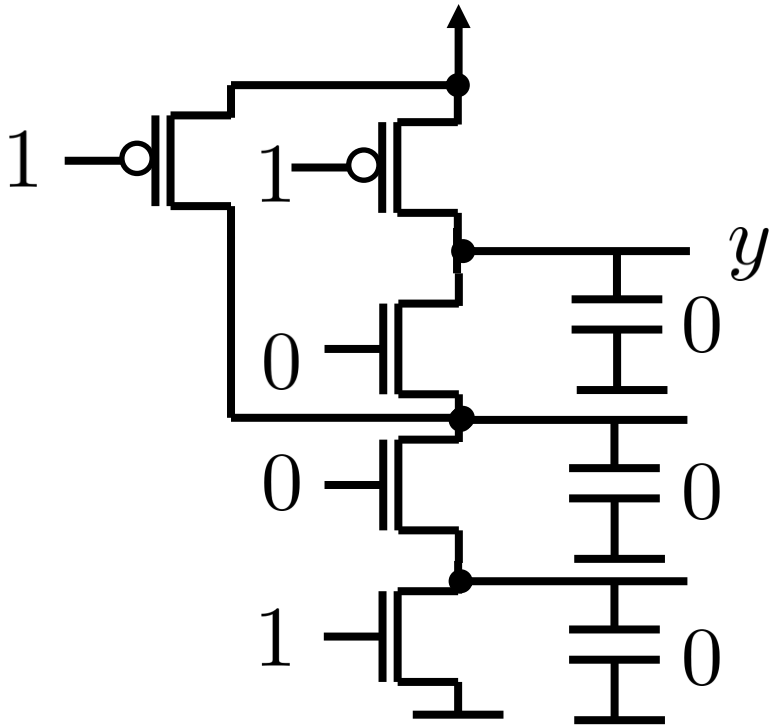


Charge Sharing Workaround

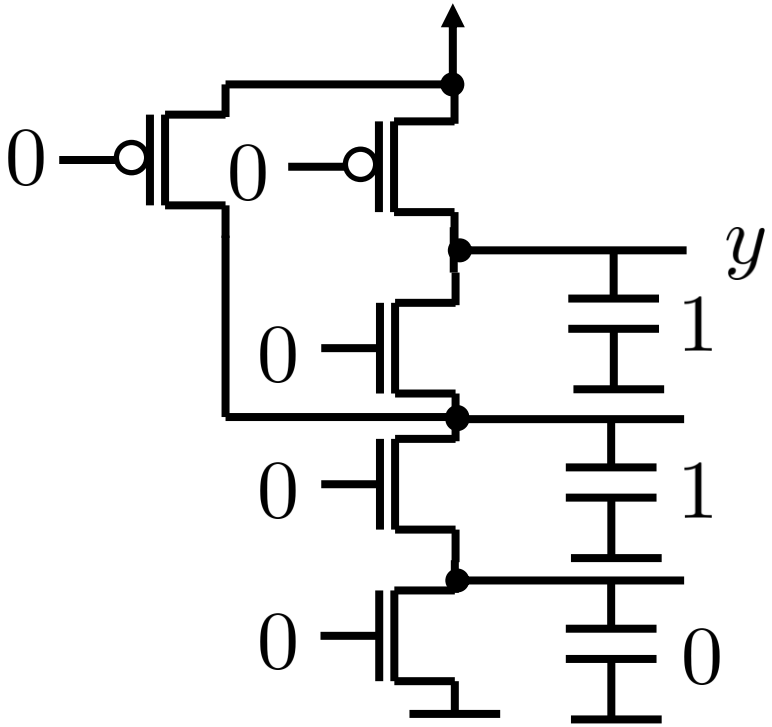


Charge Sharing Workaround

1. evaluate

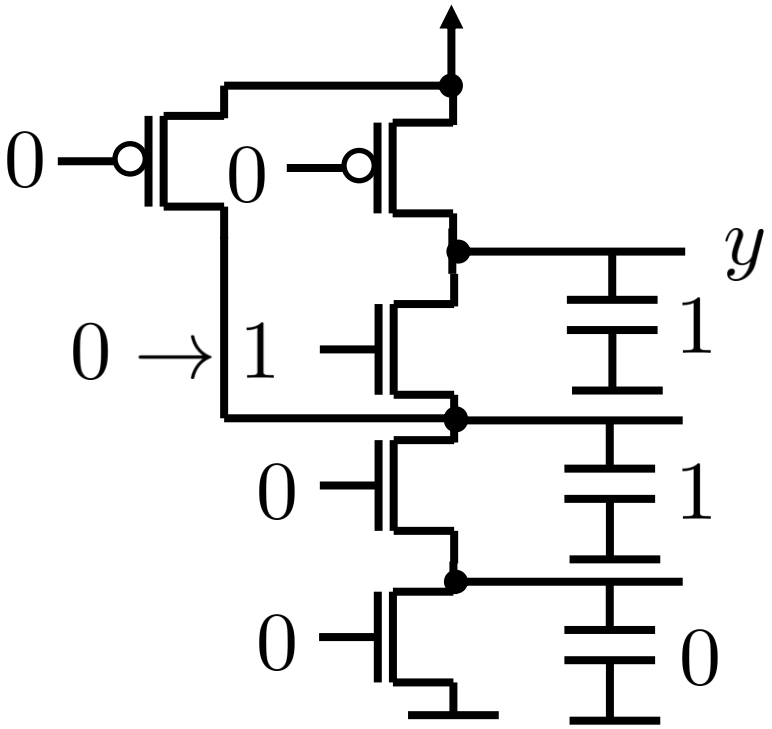


2. next charging

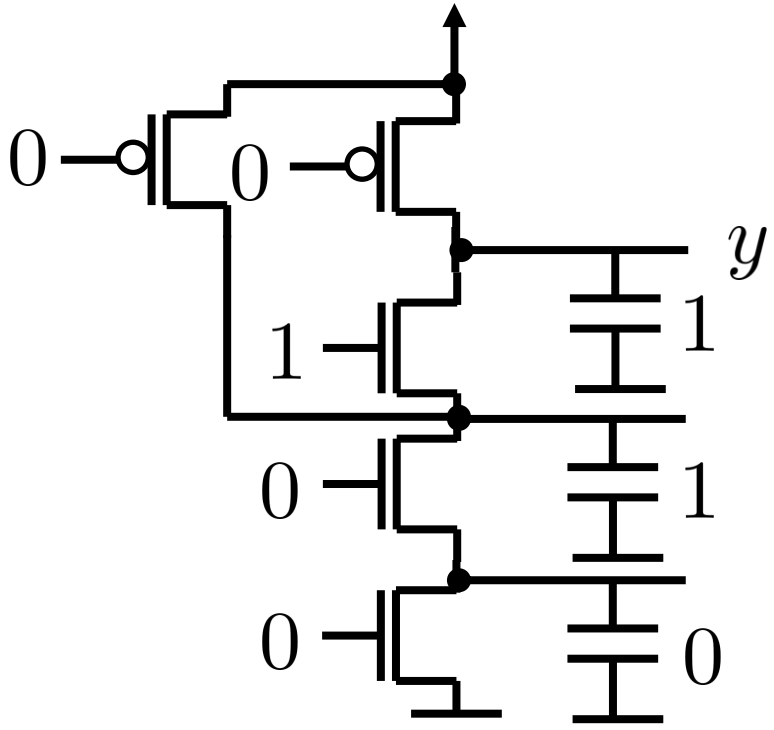


Charge Sharing Workaround

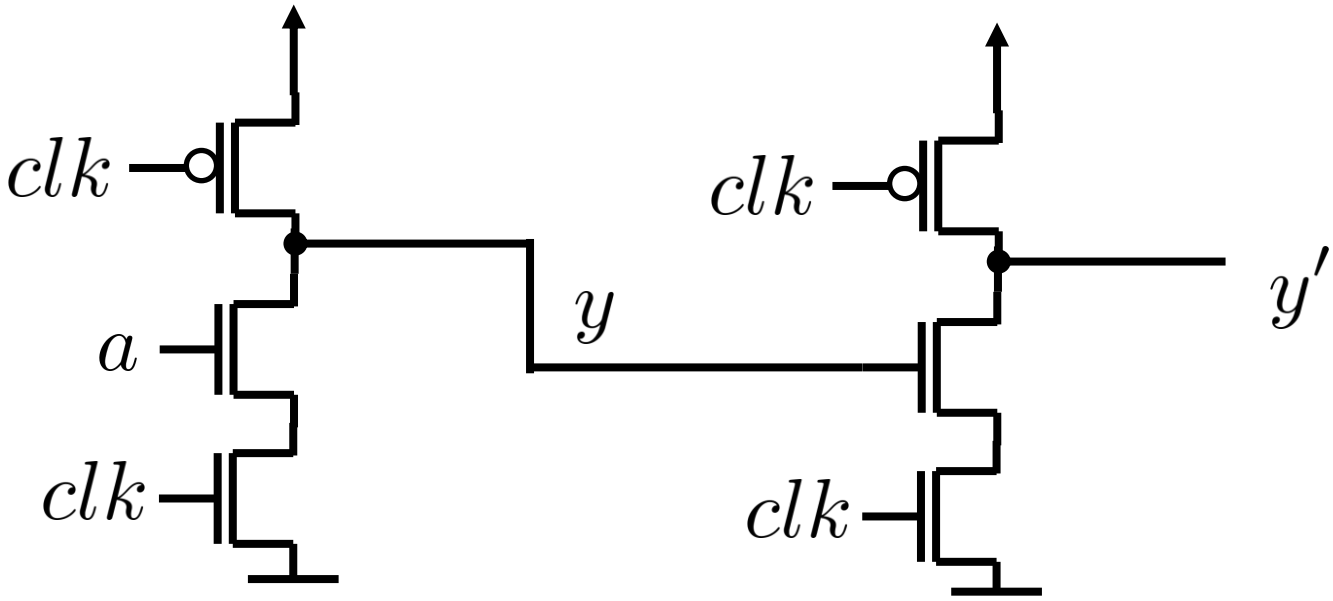
3. evaluate



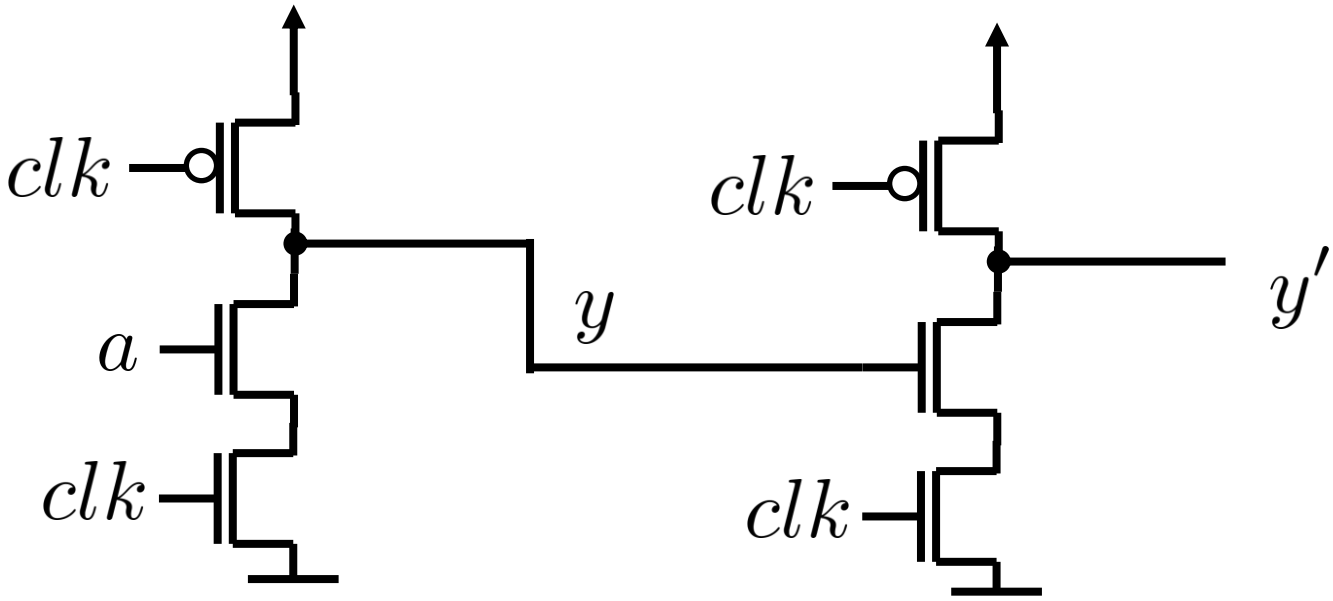
3’. at end of evaluation

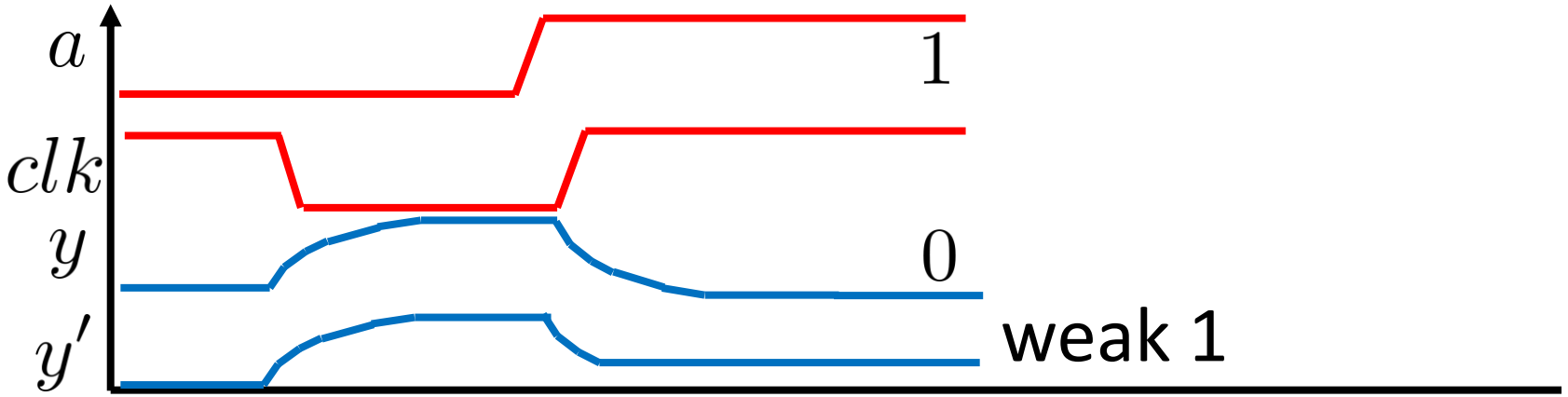


Cascading Gates



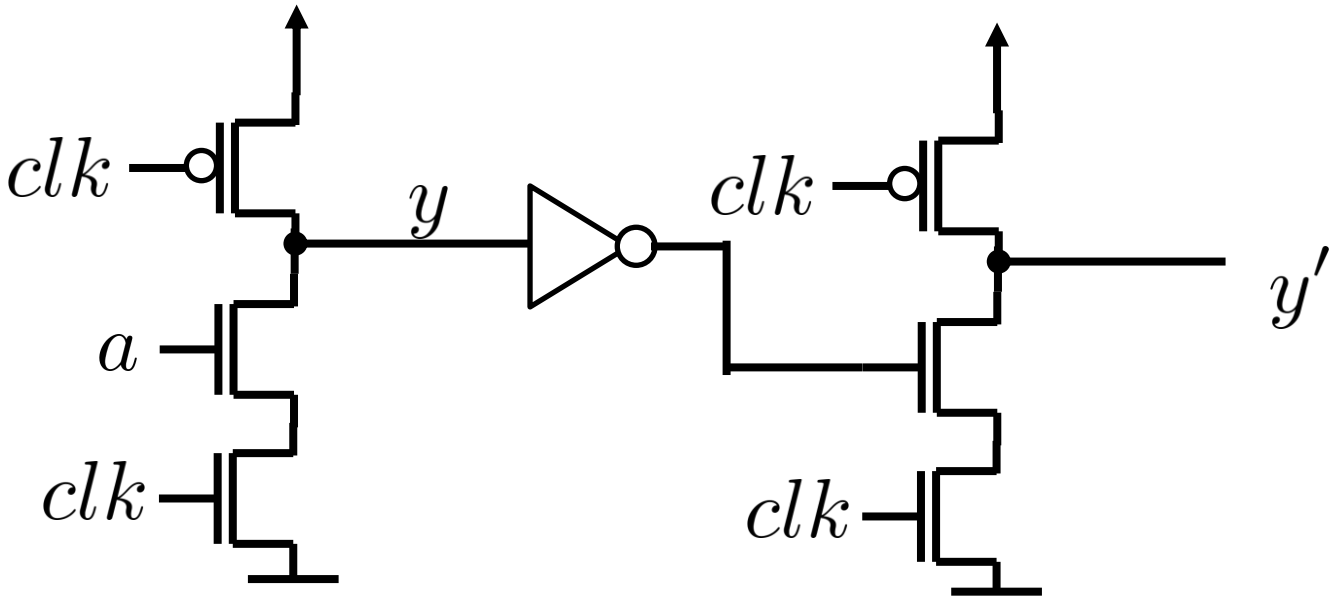
Cascading Gates

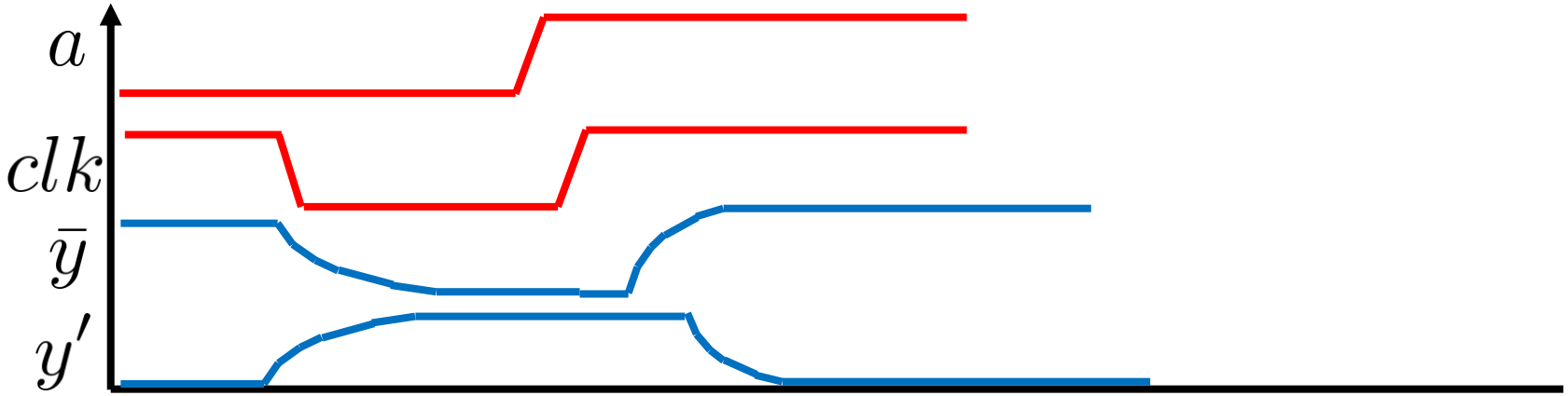




(possibly even less)!

Cascading Gates: Domino Logic





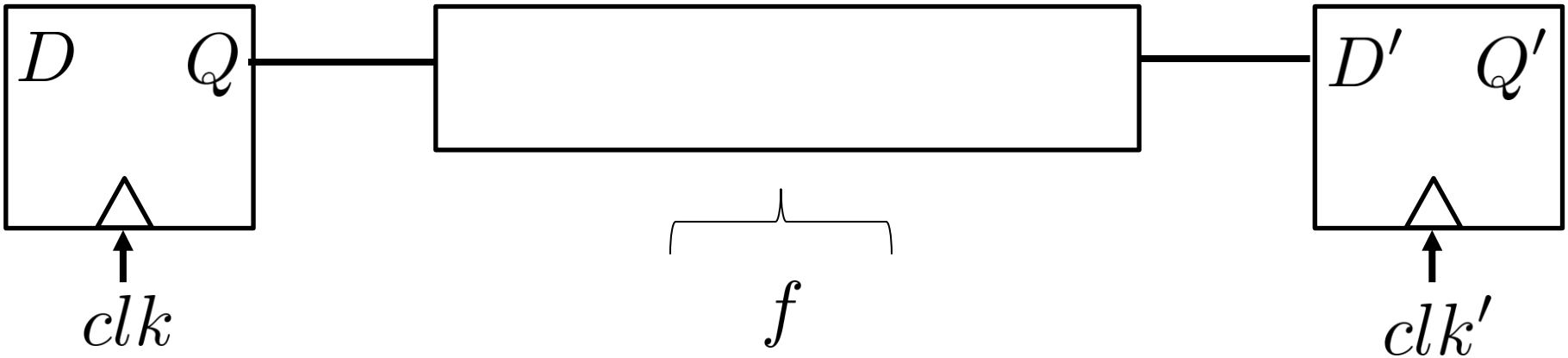
Beyond classical circuit design lecture 10.75

Pipelining

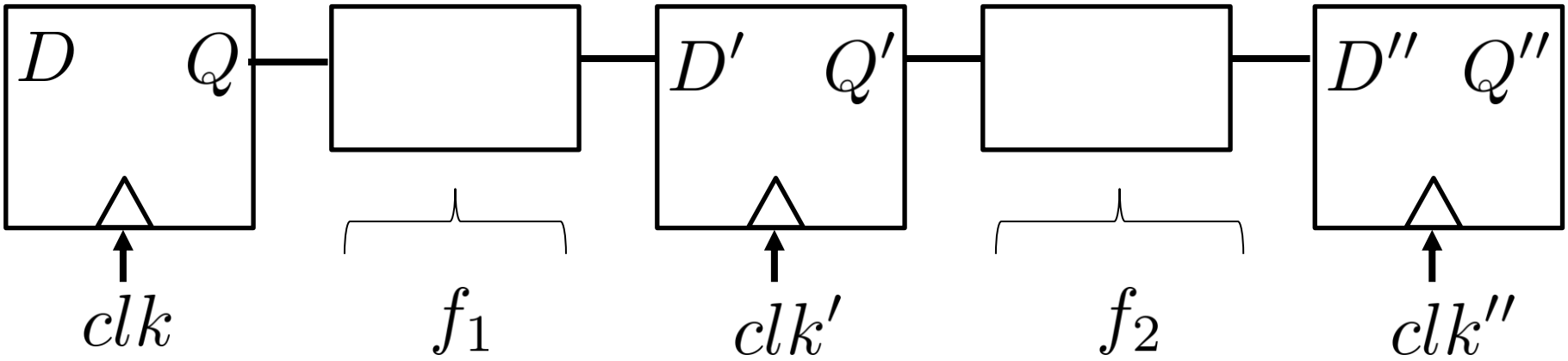
Pipelining

- In clocked/synchronous designs
- In clockless/asynchronous designs
  - static logic
  - dynamic logic

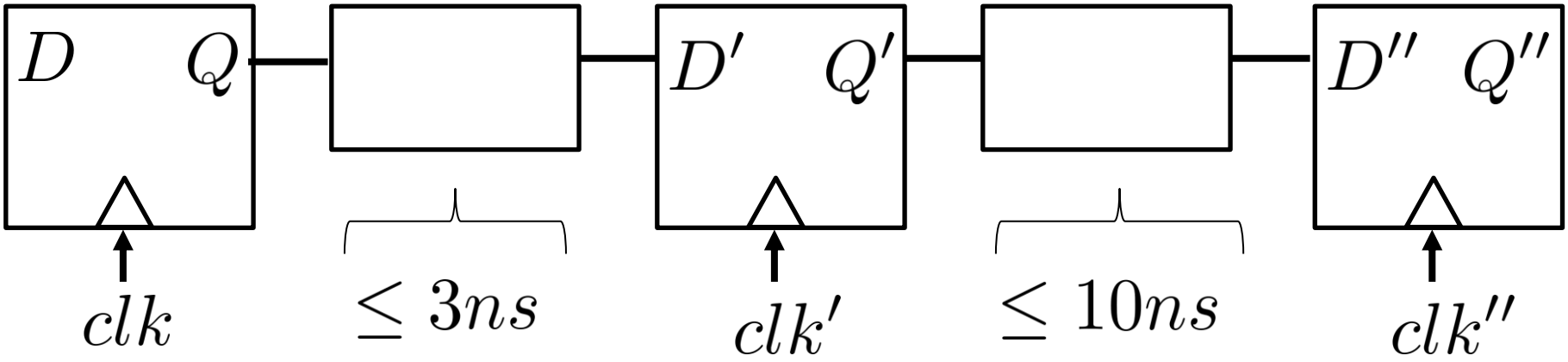
Synchronous Pipelining



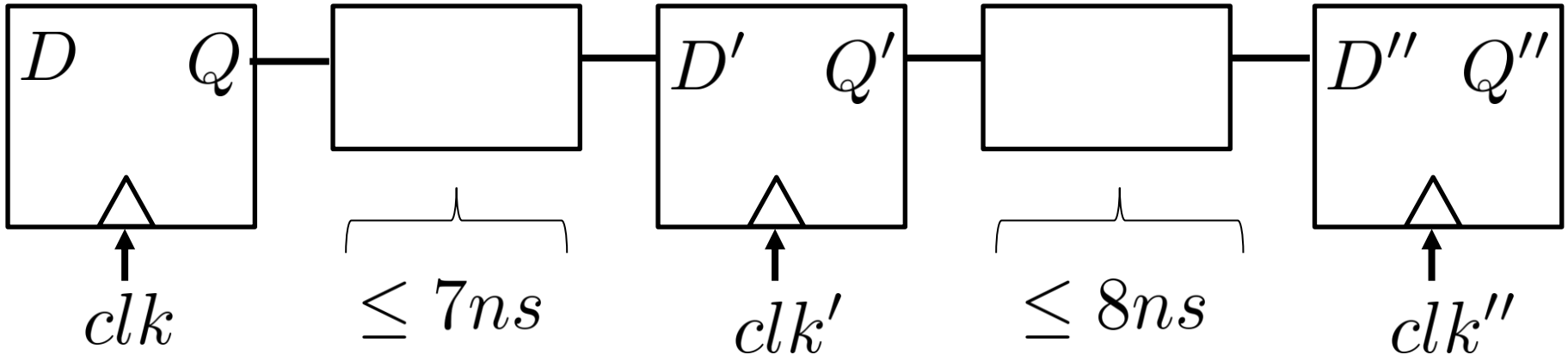
versus



Balancing



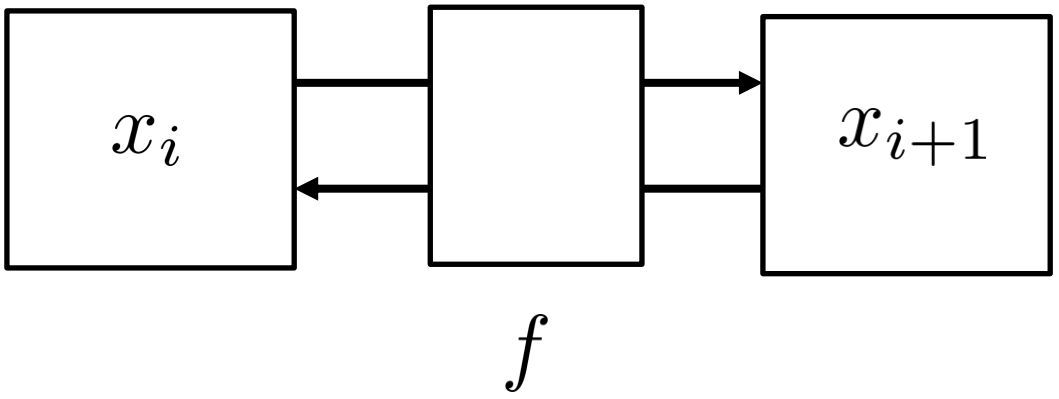
versus



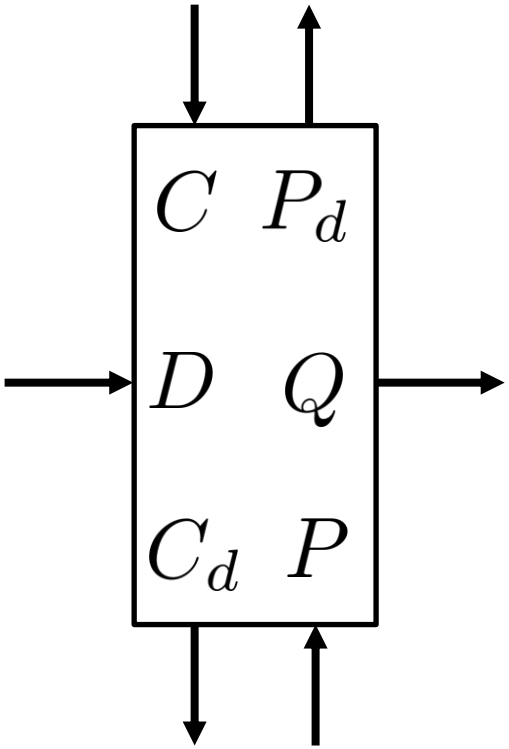
Clockless Pipelining

... with static logic

Local handshaking instead of global clock



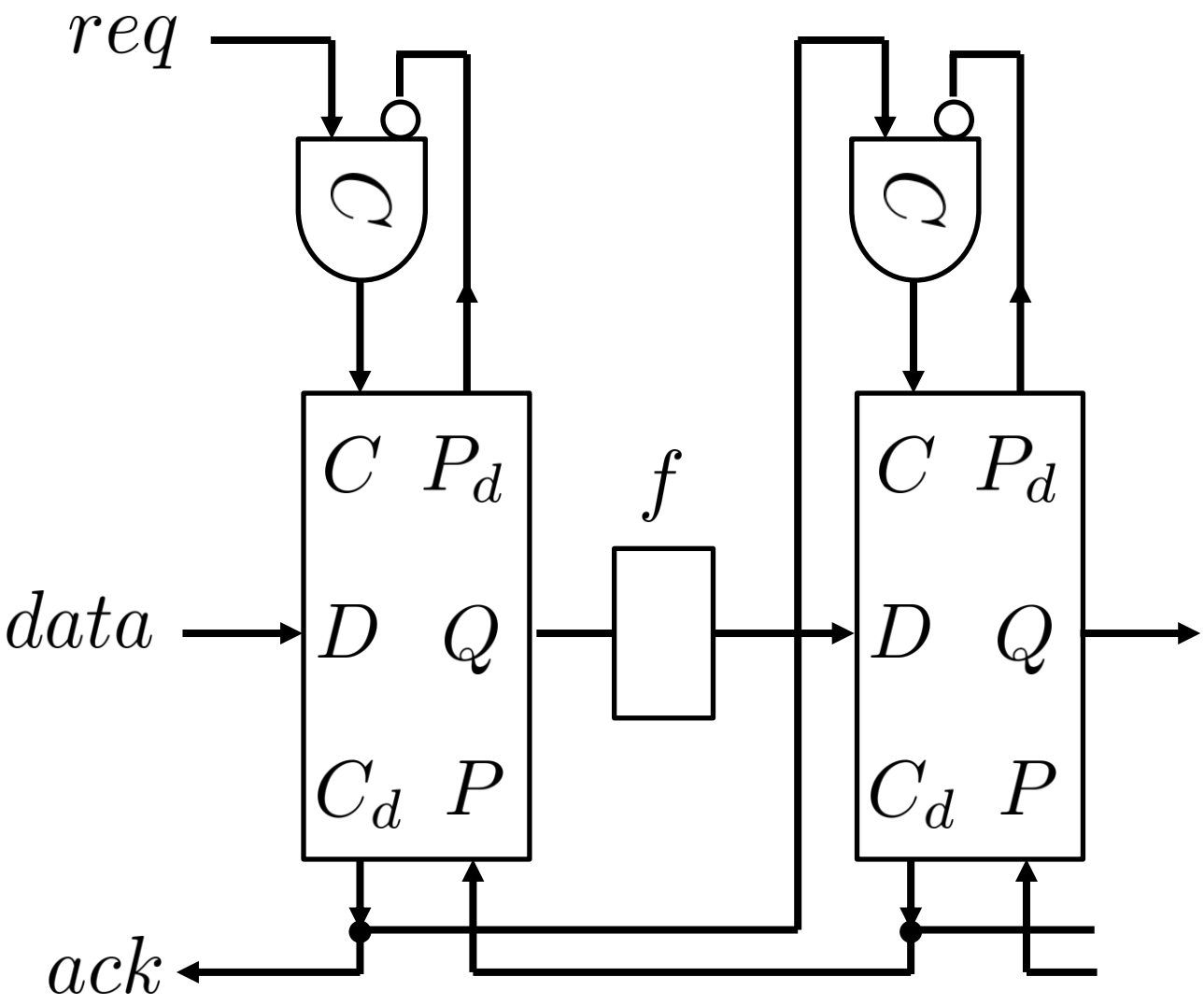
Capture-pass latch



$$C = P \rightarrow \text{ pass } D \text{ to } Q$$

$$C \neq P \rightarrow \text{ hold } Q$$

Sutherland's Micropipeline

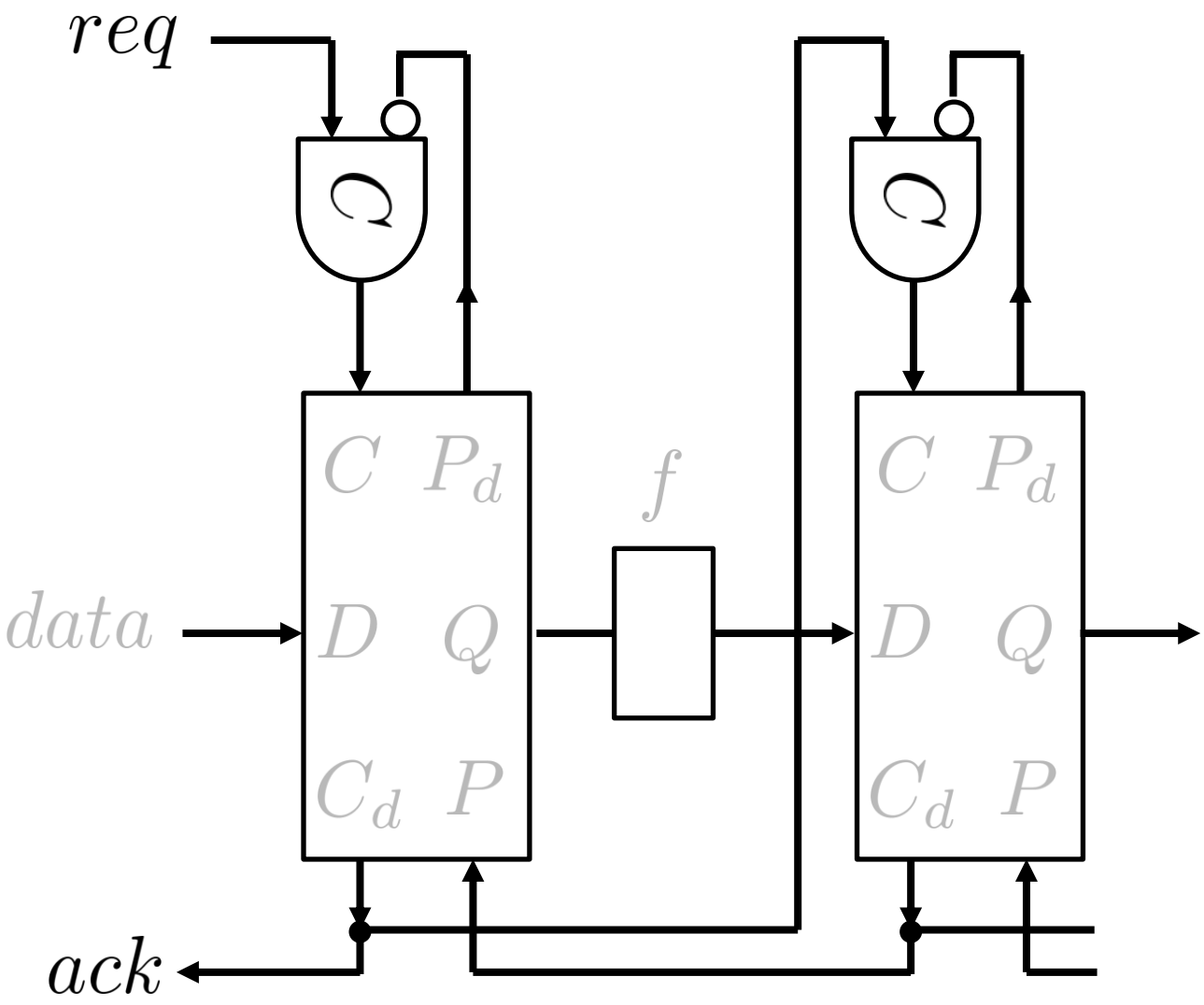


Sutherland's Micropipeline

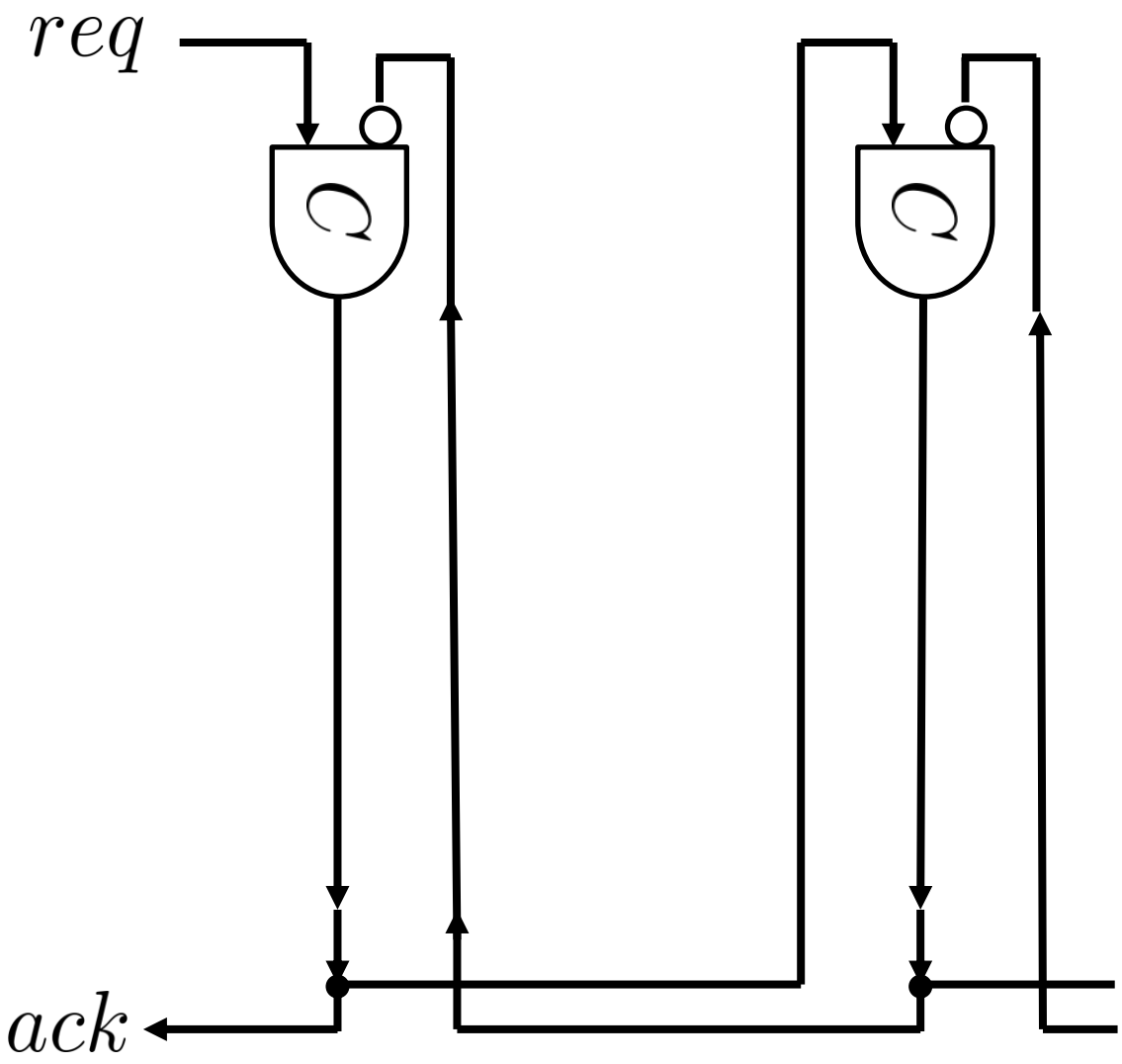
- 2-phase handshaking (here)
- Bundled data (here)
- Initially all latches transparent (= pass mode)

- Constraints:
  - delays from C to Cp and from P to Pd need to be long enough for latch hold times
  - bundled data constraint

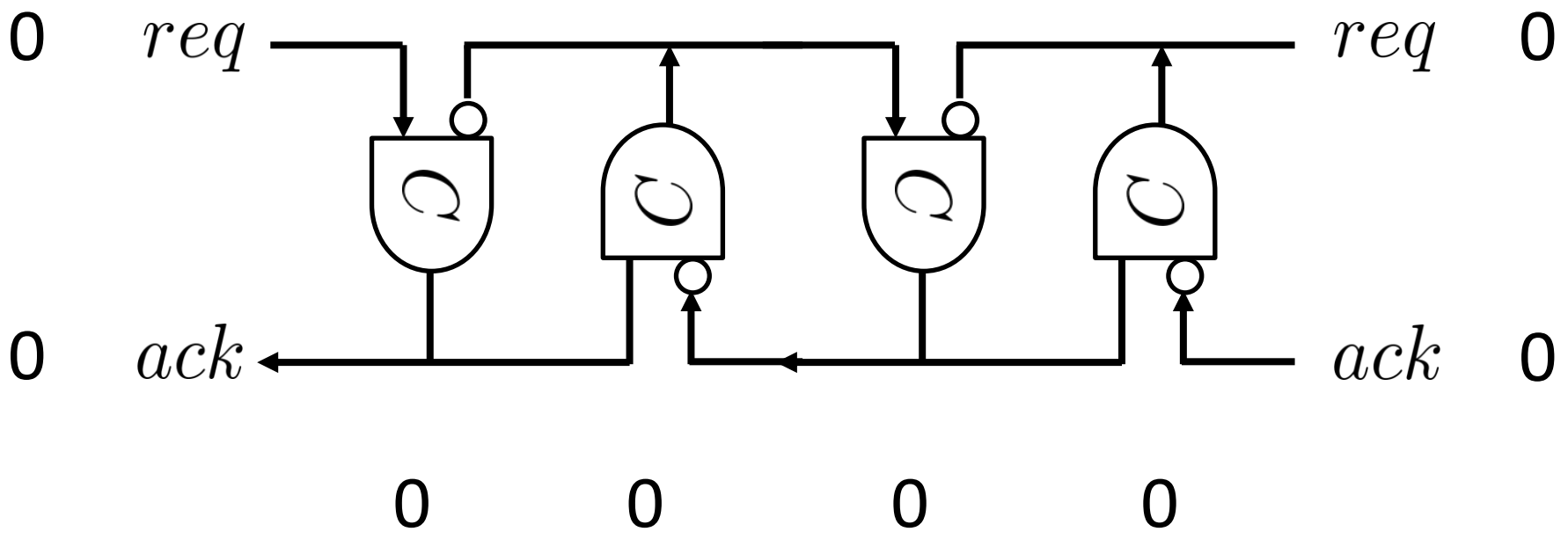
Control structure



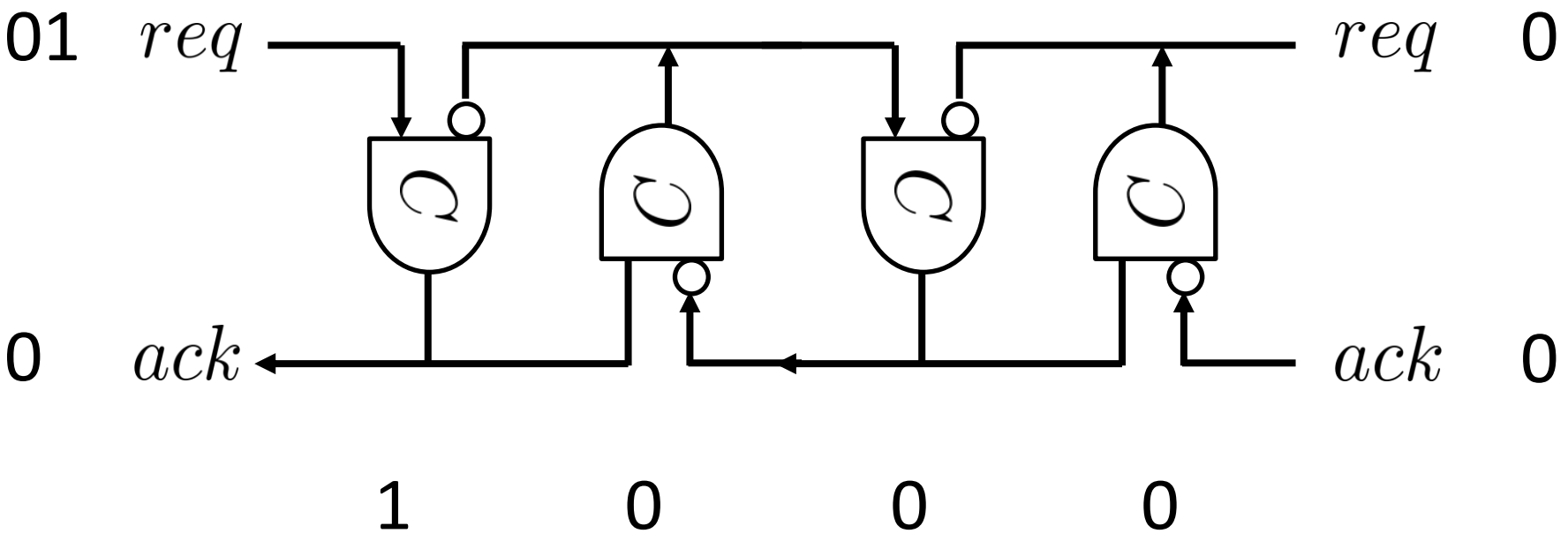
Control structure



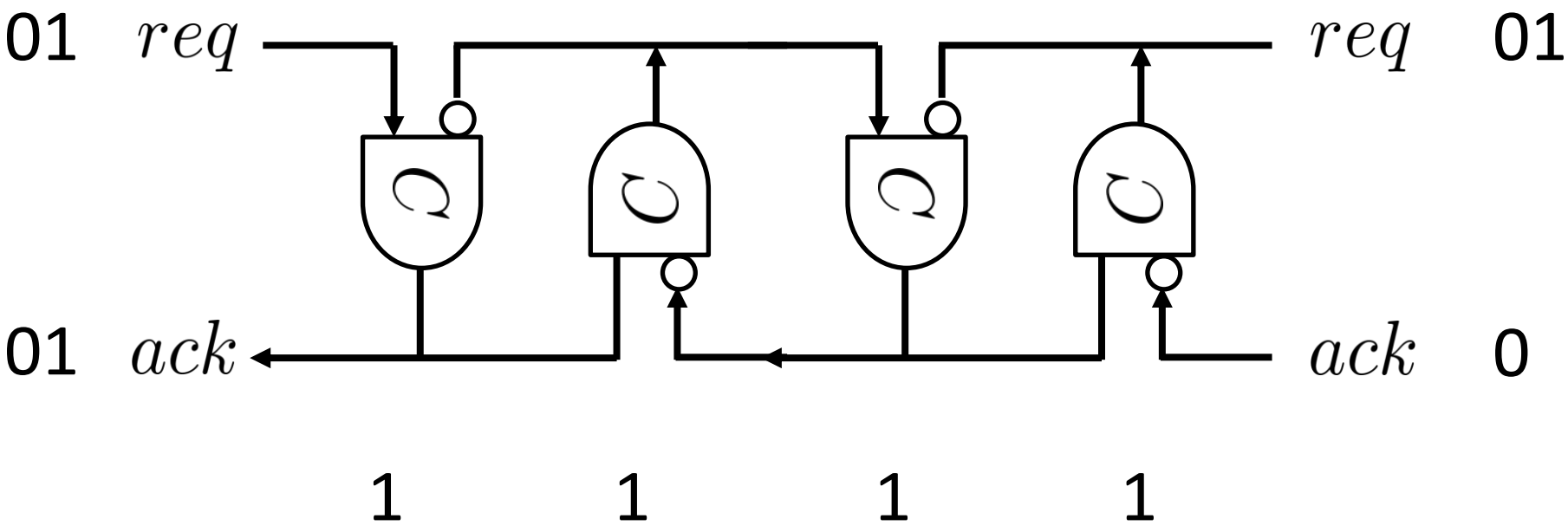
Control structure



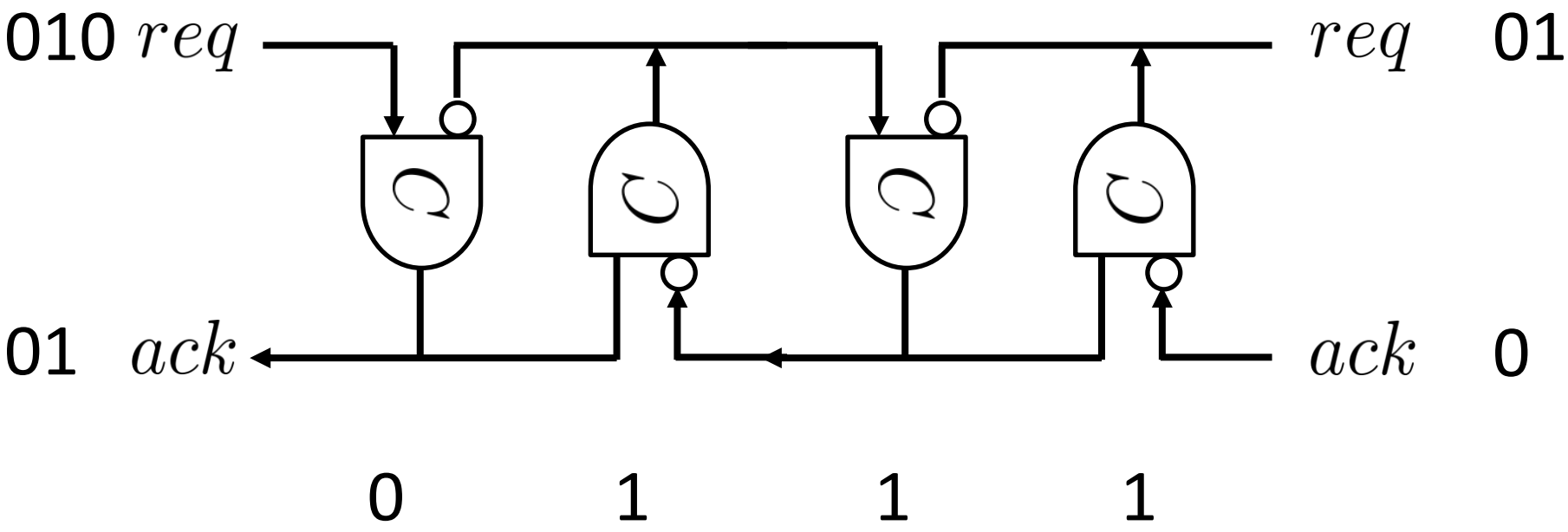
Control structure



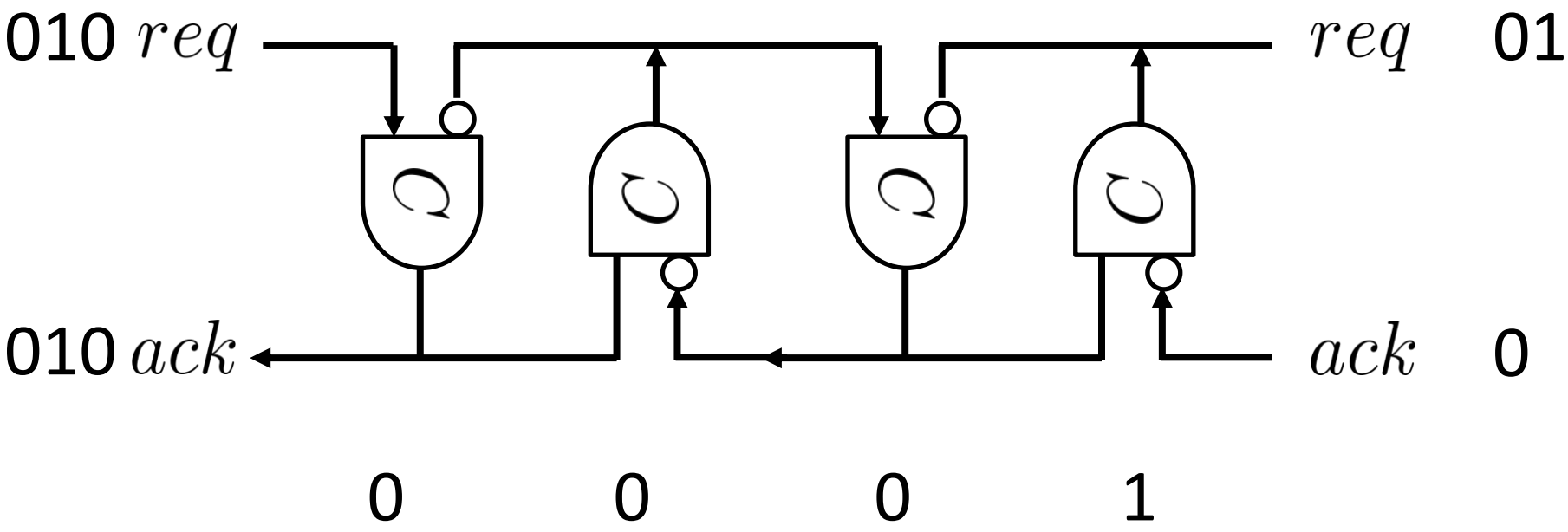
Control structure



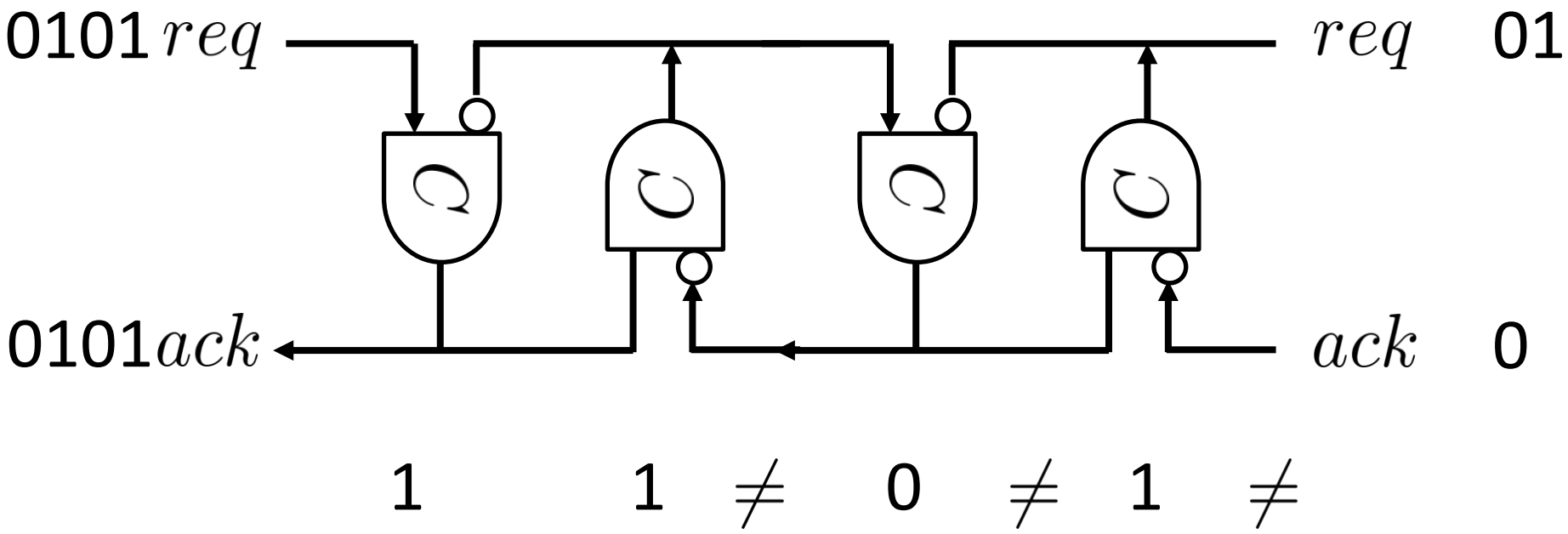
Control structure



Control structure

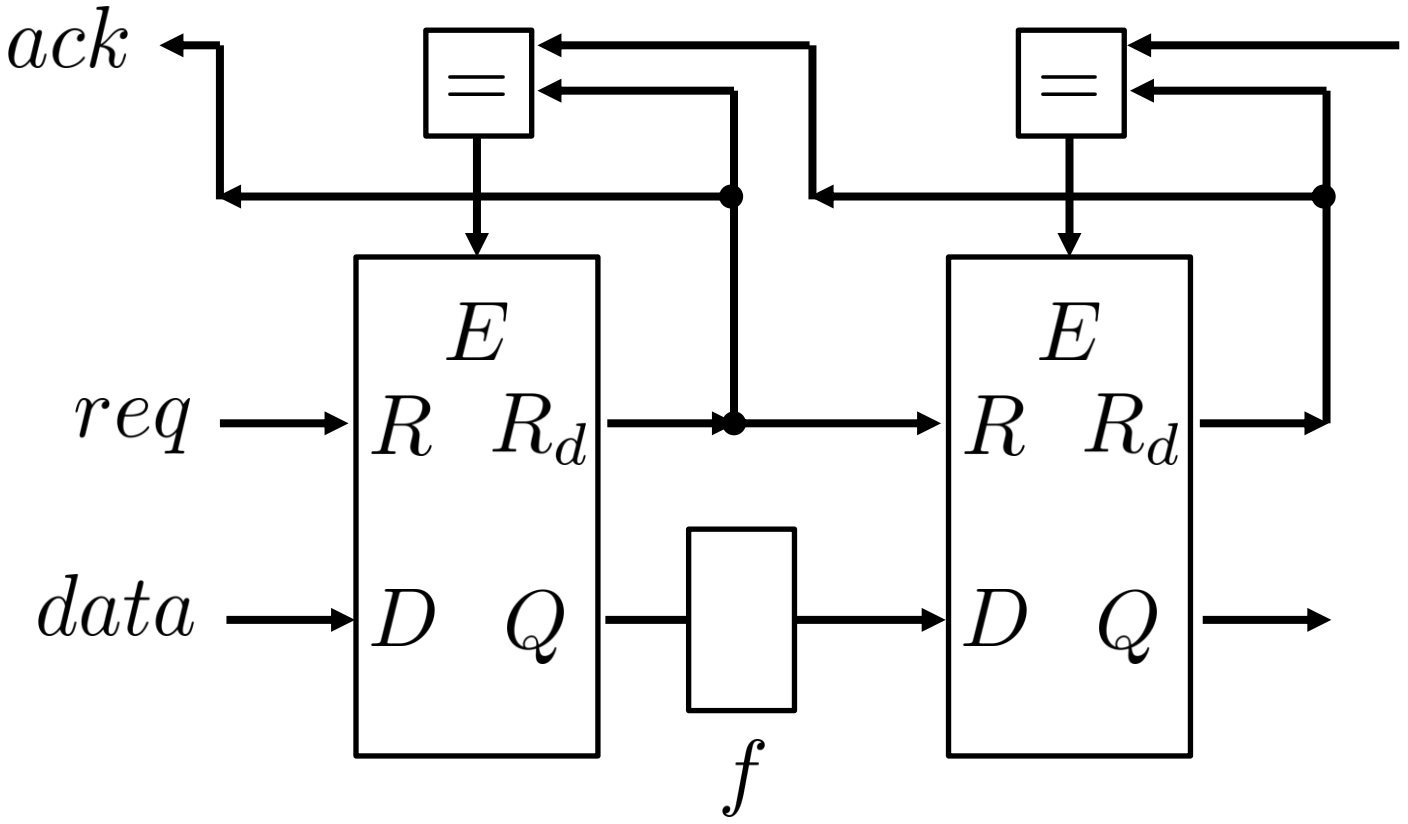


Control structure



3 different consecutive C-states -> 3 “items” stored

Mousetrap pipeline



Mousetrap pipeline

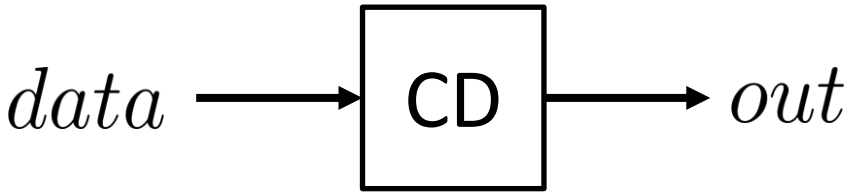
- 2-phase handshaking
- Bundled data
- Initially all latches transparent (= pass mode)

- Constraints:
  - ack delay needs to be long enough for latch hold time
  - bundled data constraint

Clockless Pipelining

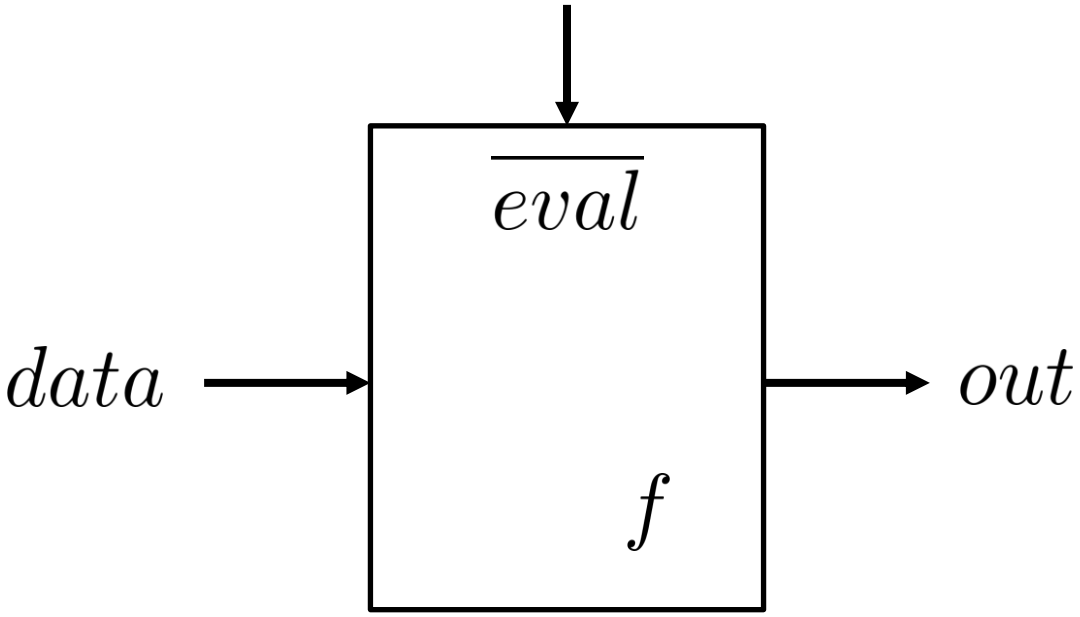
... with dynamic logic

Completion dedection



$$*[v(data); out\uparrow; n(data); out\downarrow]$$

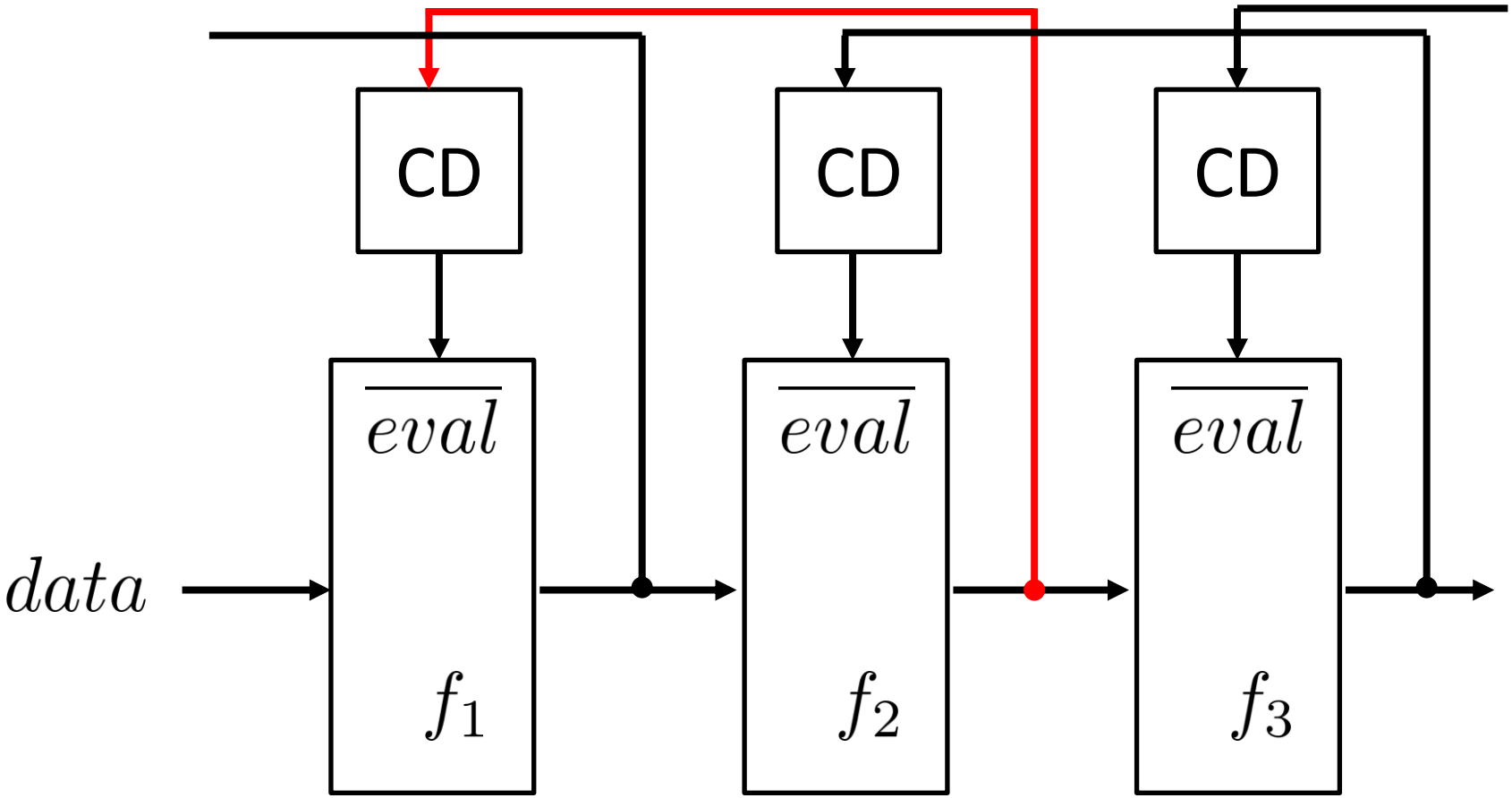
Dynamic function block



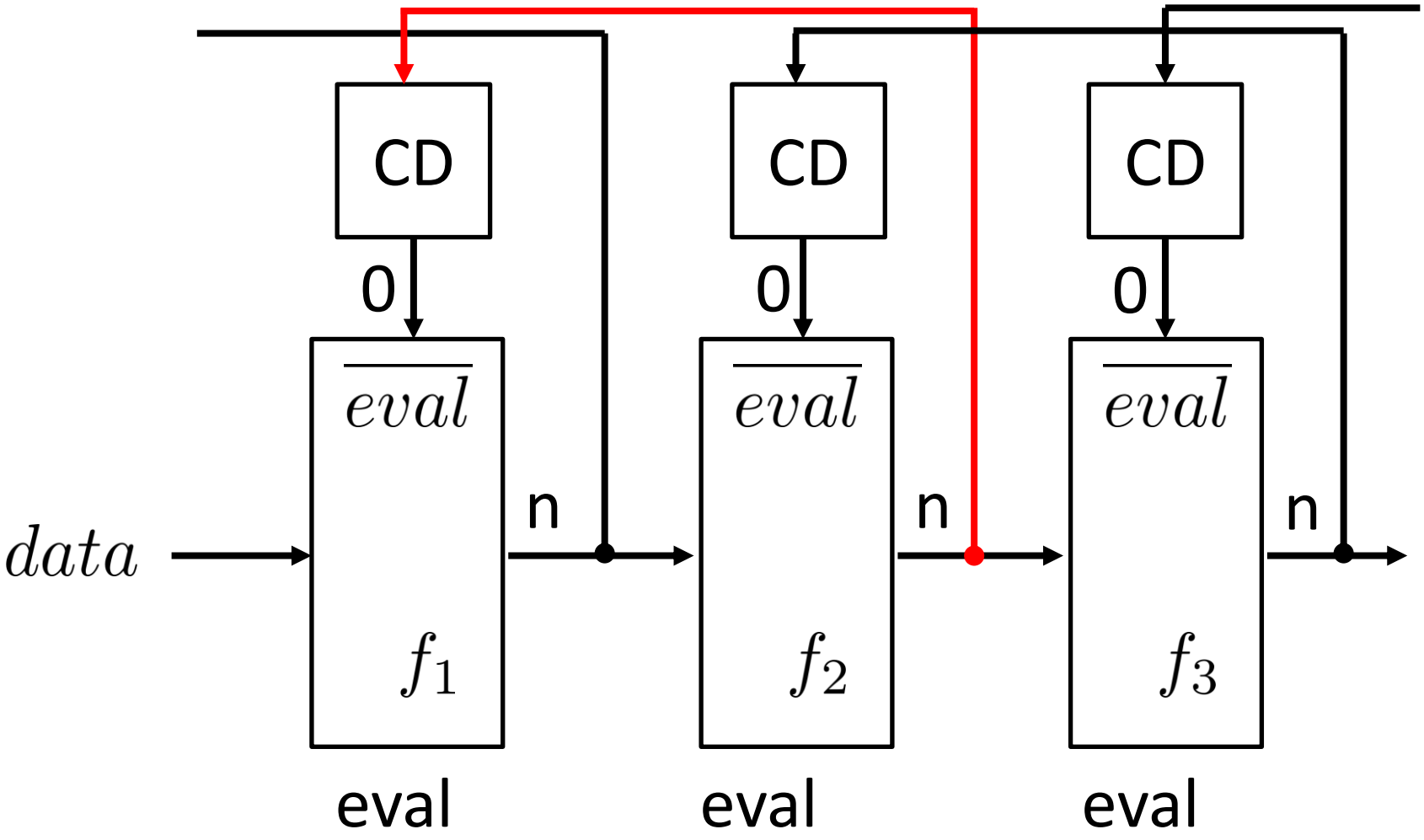
eval = 0 -> evaluate out = f(data)

eval = 1 -> precharge out (here to all 0)

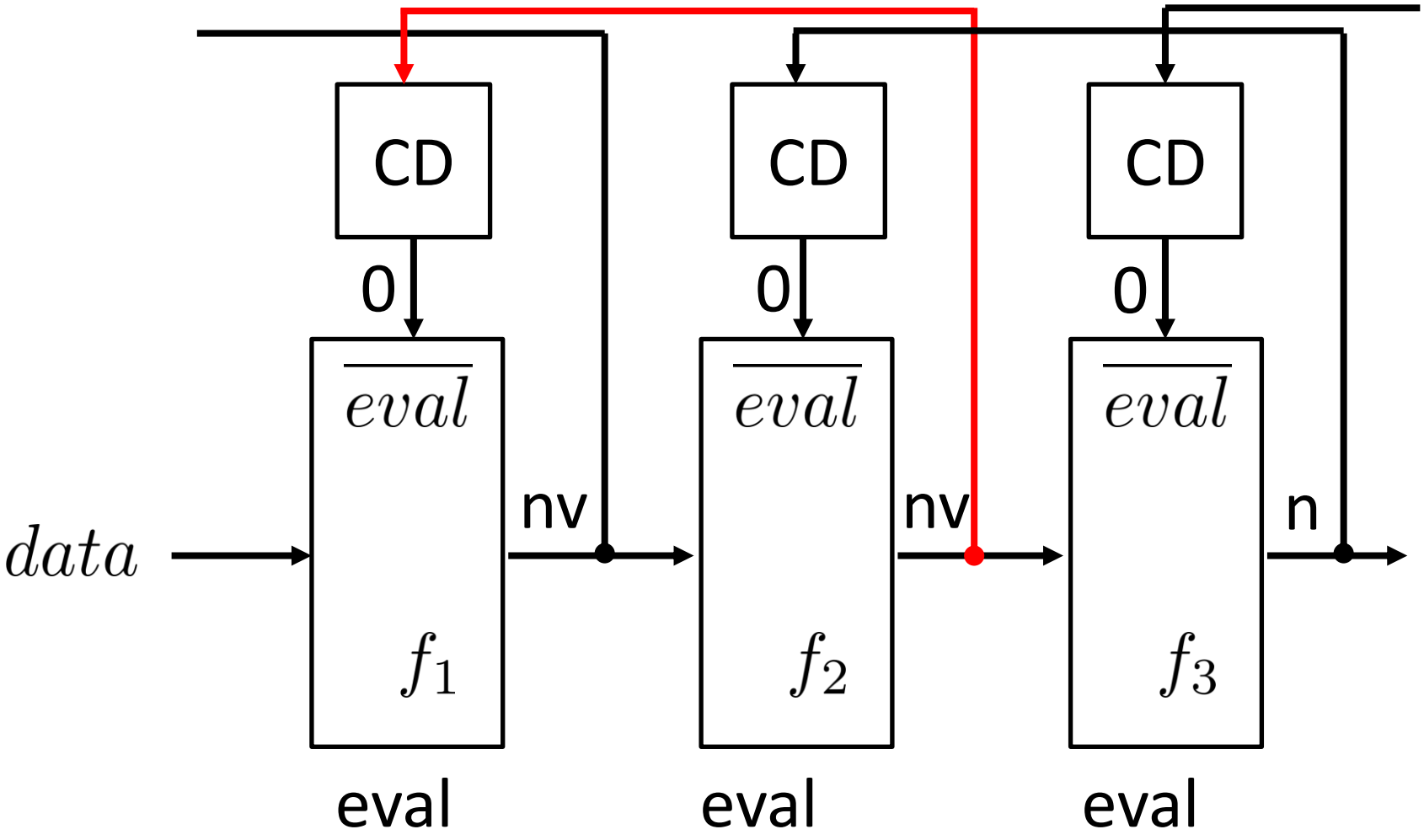
PS0 pipeline



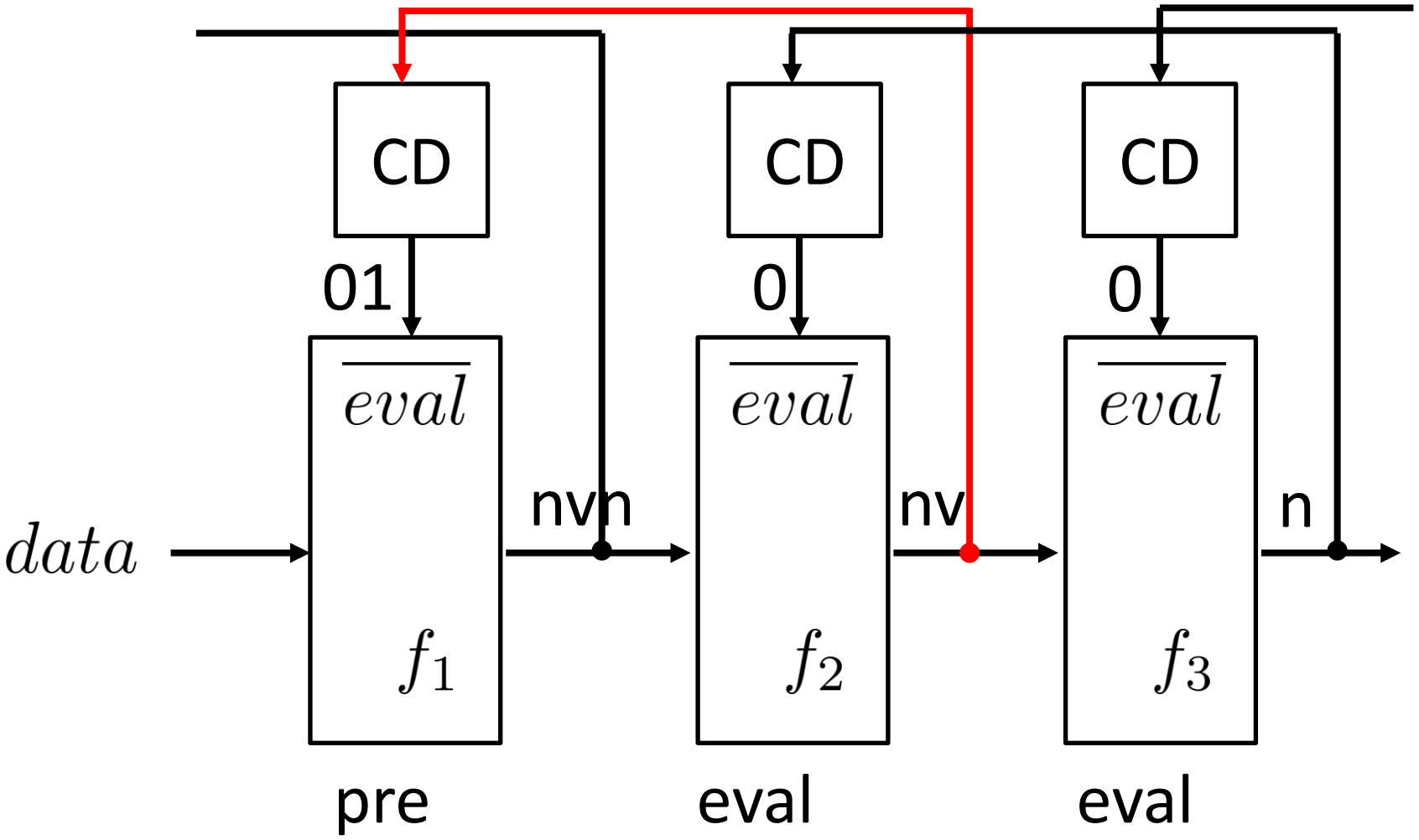
PS0 pipeline



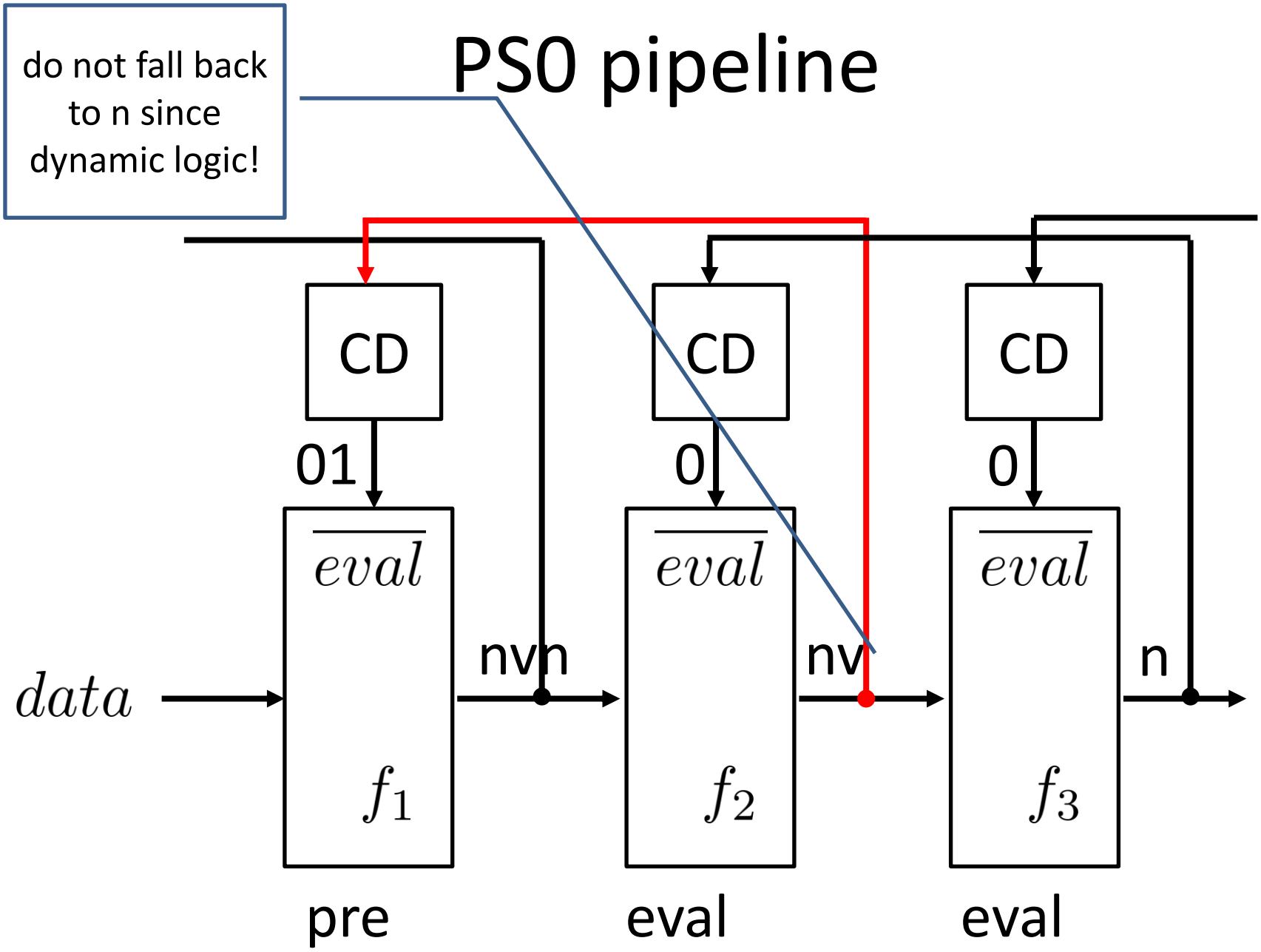
PS0 pipeline



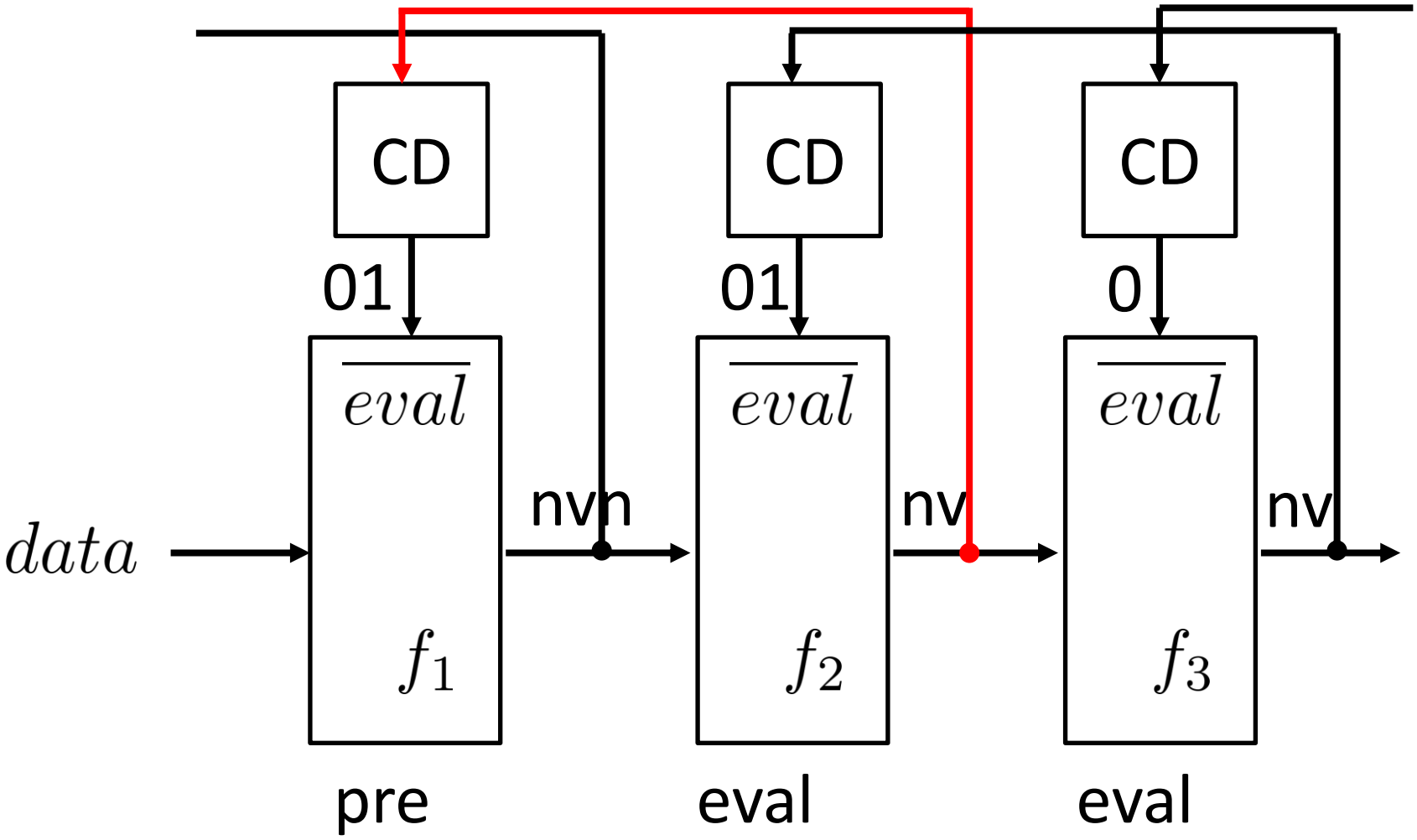
PS0 pipeline



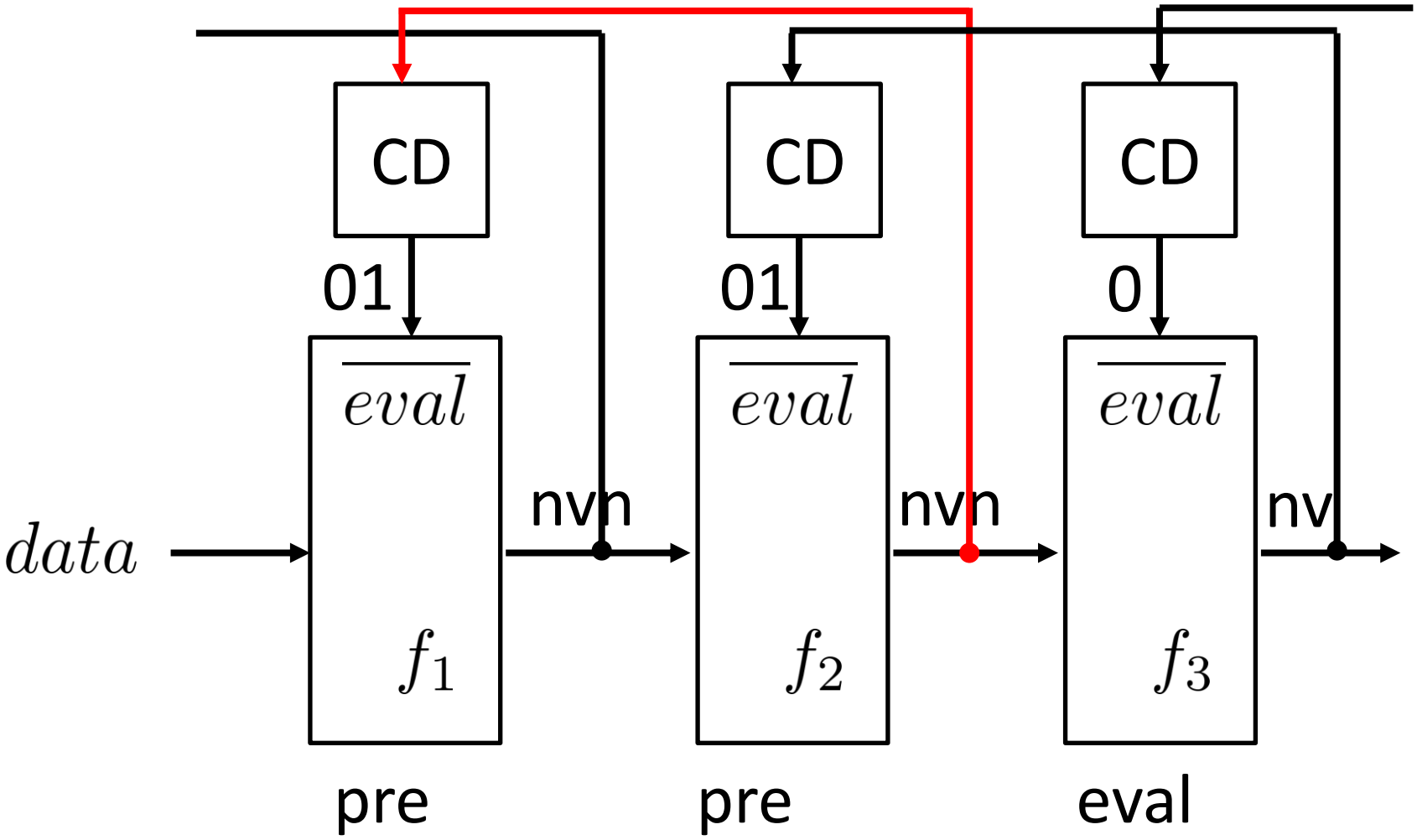
PS0 pipeline



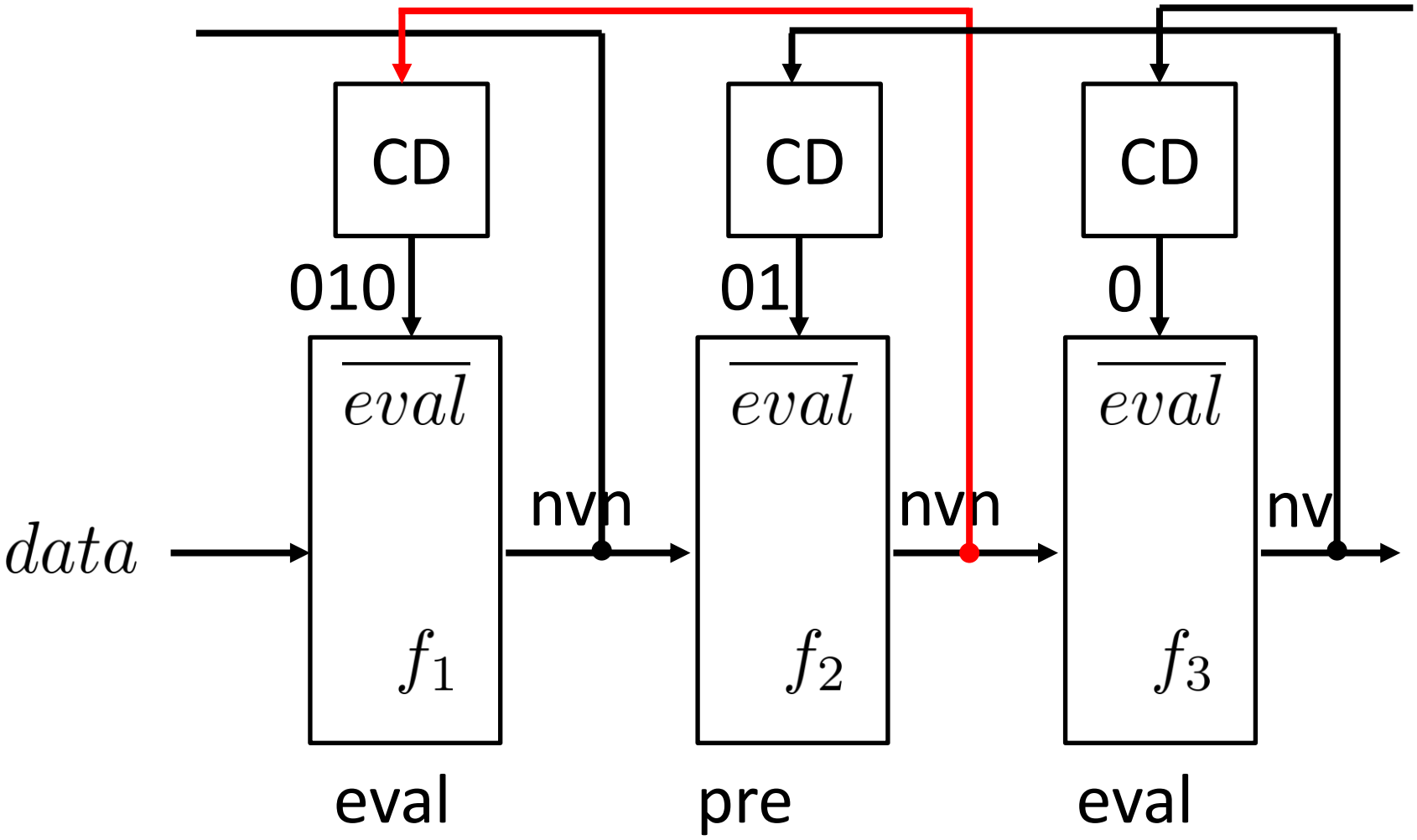
PS0 pipeline



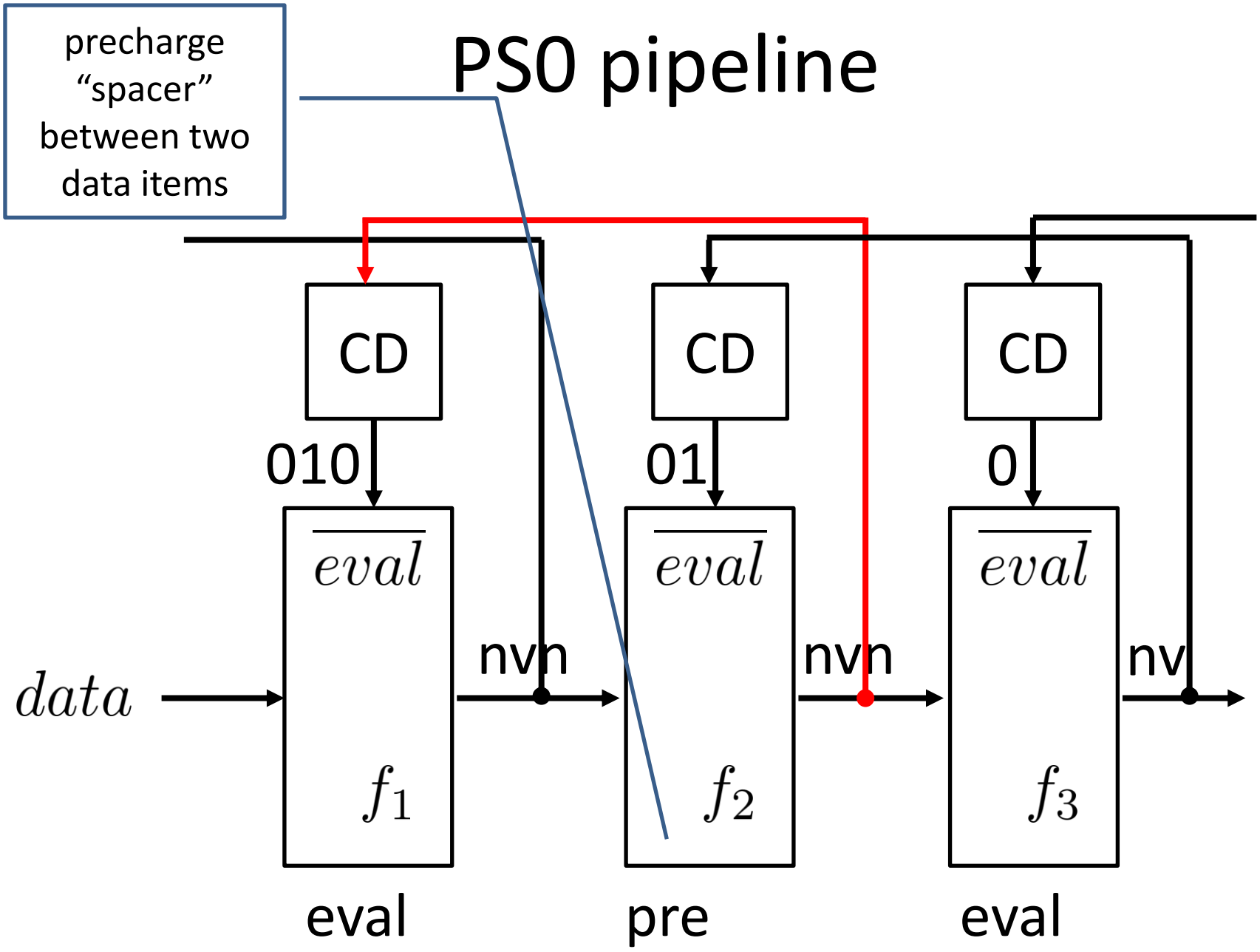
PS0 pipeline



PS0 pipeline



PS0 pipeline



PS0 pipeline

- DI encoded data (here)
- Initially all latches transparent (= pass mode)

- Constraint:
  - CD delay needs to be long enough for latch hold time

PS0 pipeline

+ pros of dynamic logic

- handshake delays between 3 stages for "freeing" storage again after data wave
- spacers between data items

  -> can use only half the pipeline

Clockless pipelines

+ balancing not necessary (although can improve)
+ automatic “clock gating” (low energy)
+ very fast initial wave

Other static/dynamic pipeline designs exist that improve shown designs.