Beyond classical circuit design
lecture 10

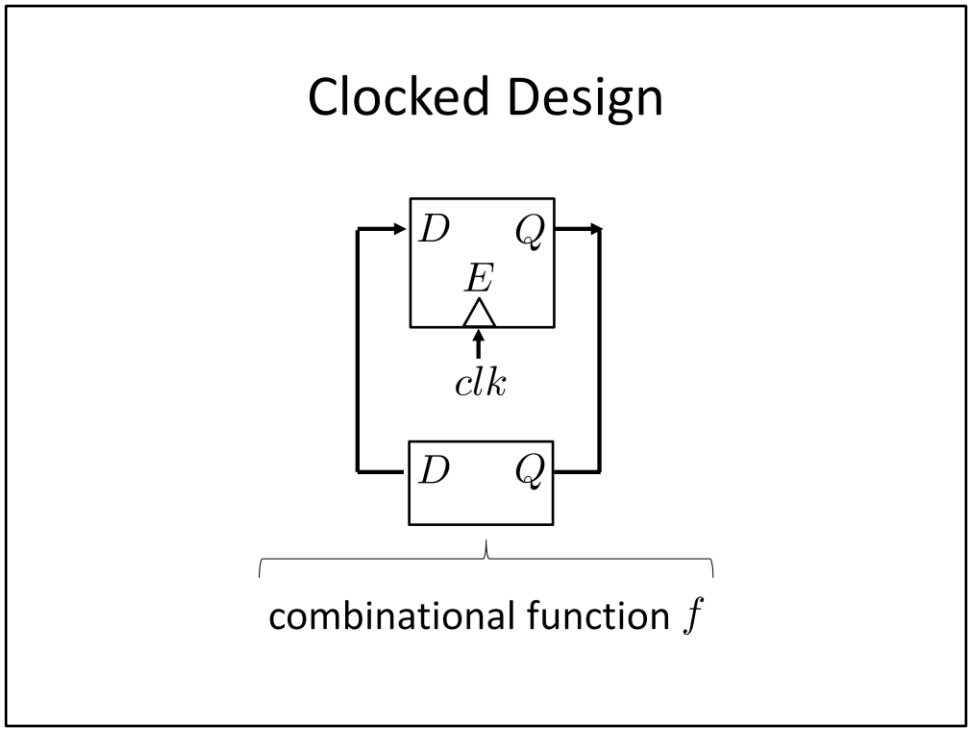
Clocked Design &
Pipelining (Clocked & Asynchronous)

Further Reading

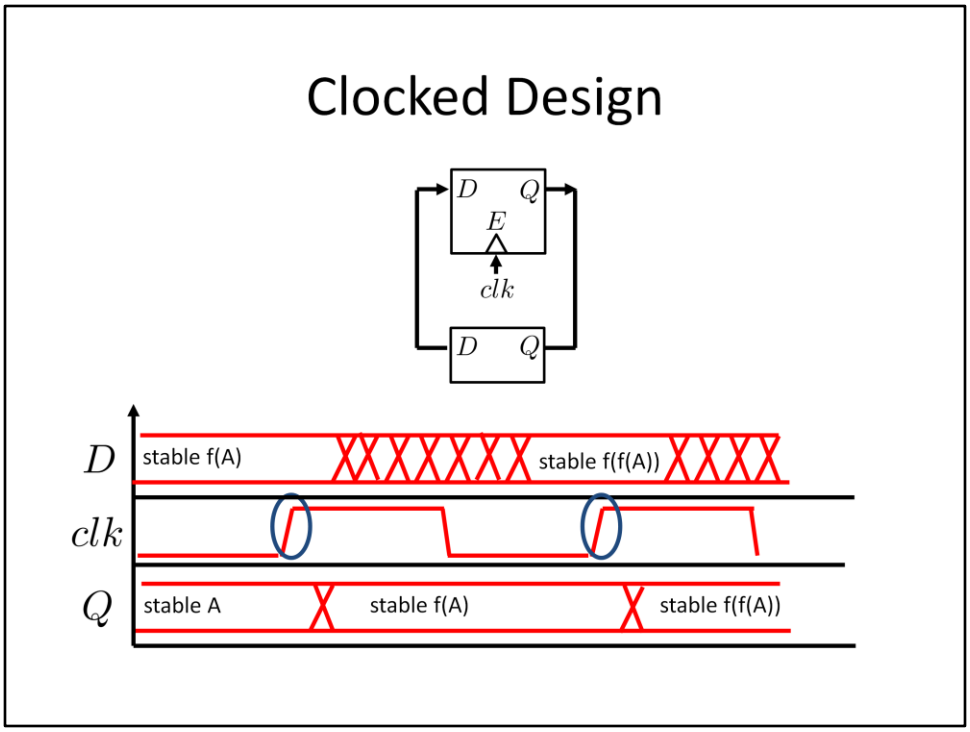
Jan M. Rabaey, Anantha Chandrakasan, Borivoje Nikolic: *Digital Integrated Circuits. A Design Perspective. 2nd edition.* Prentice Hall, 2003.

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



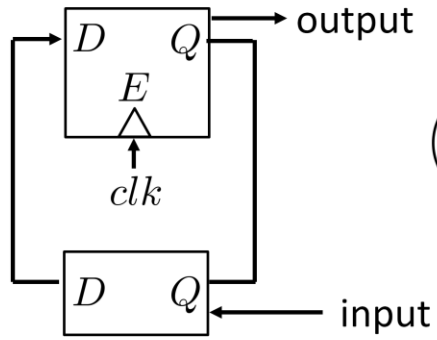
timed combinational gates with input Q and output D.
here: shown only 1-bit flip-flop. Of course, staggering to registers is possible.



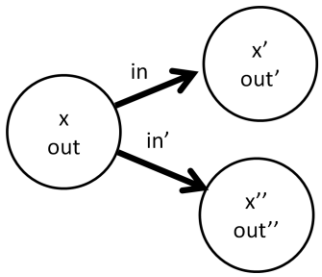
computing f(f(...(initial state)..))

timing constraints to ensure setup-hold conditions

Clocked Design in Environment

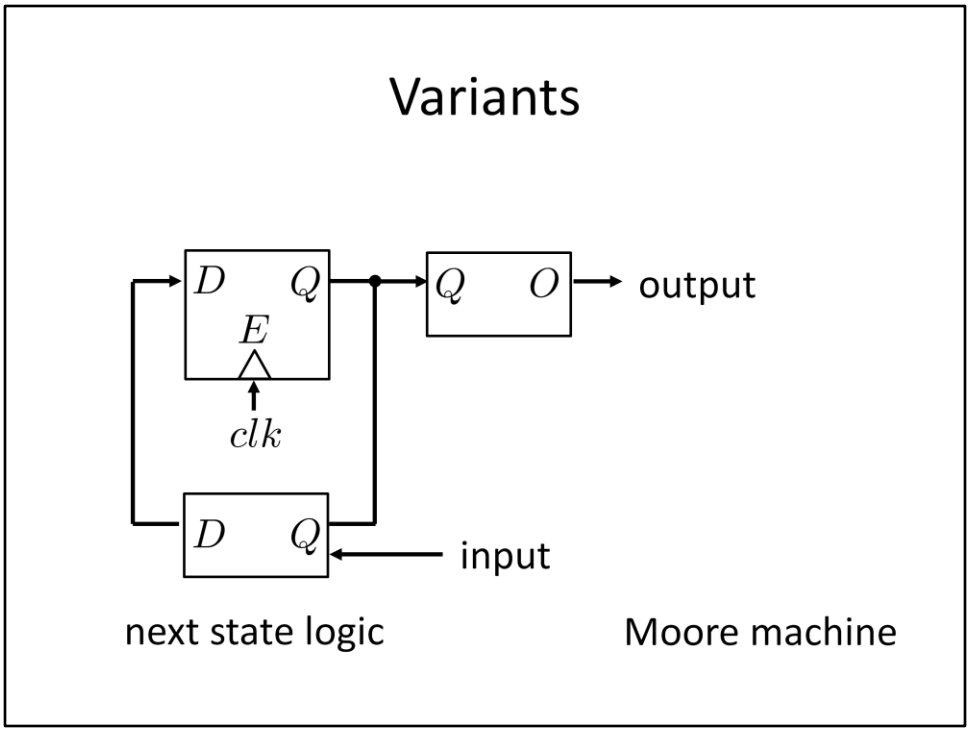


next state logic

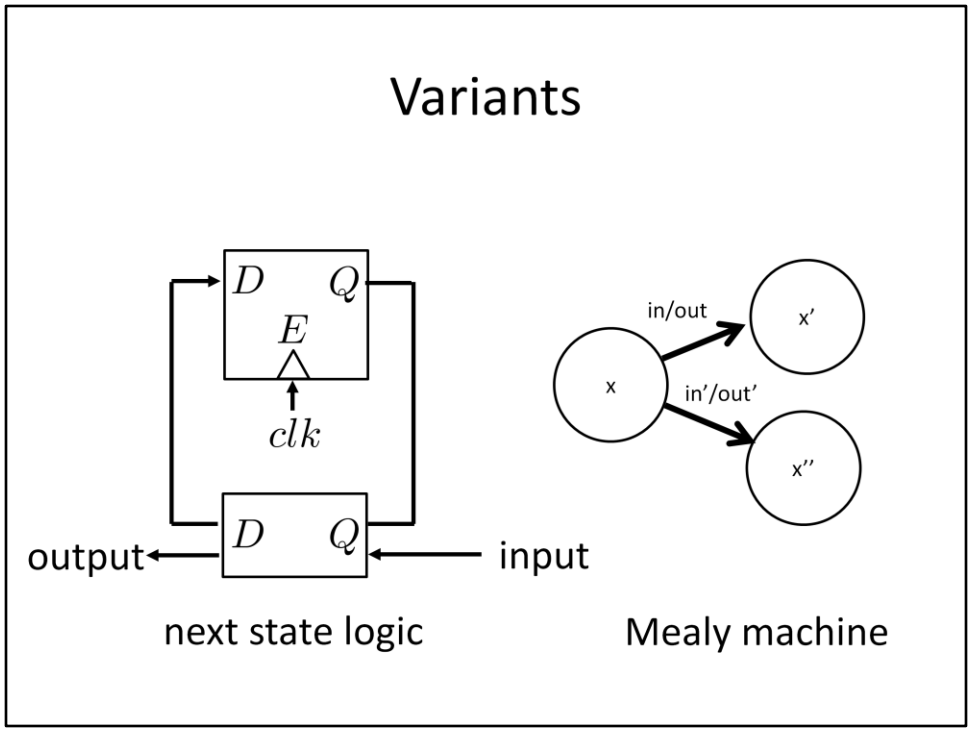


Moore machine

implementing state machines.
next state depends on current state & input.
We assume here that the input is hold stable by the environment.
The output is part of the state.

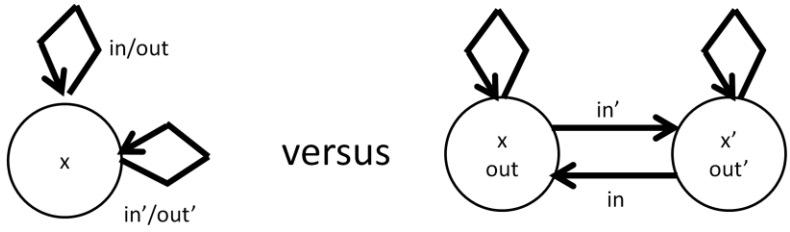


Sometimes more efficient to calculate output from state -> potentially smaller register



output directly depends on (current !) input -> potentially faster, but mind stability of inputs again.
variations with latched inputs and/or outputs exist, also.

Variants



Mealy -> Moore: split states per output

transitions to same state with different outputs possible

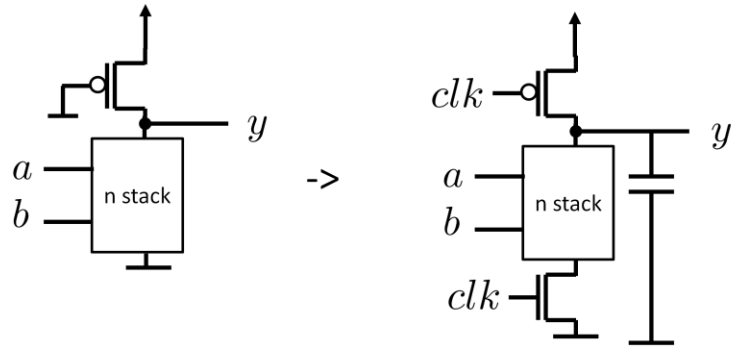
Beyond classical circuit design
lecture 10.5

Dynamic CMOS

Dynamic CMOS

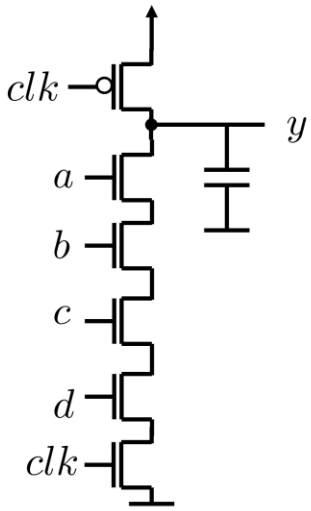
Like in pseudo nMOS: remove slow pMOS stack.

But:



precharge during clk = 0 and potentially decharge during clk = 1.

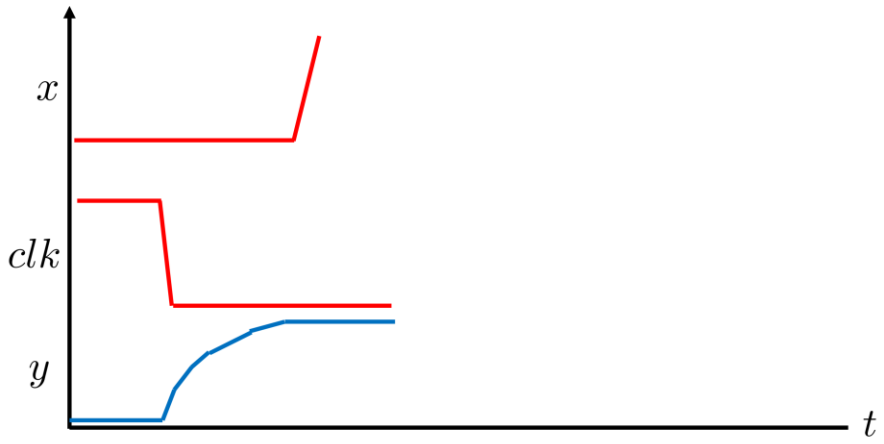
Dynamic CMOS 4NAND



mind charge sharing

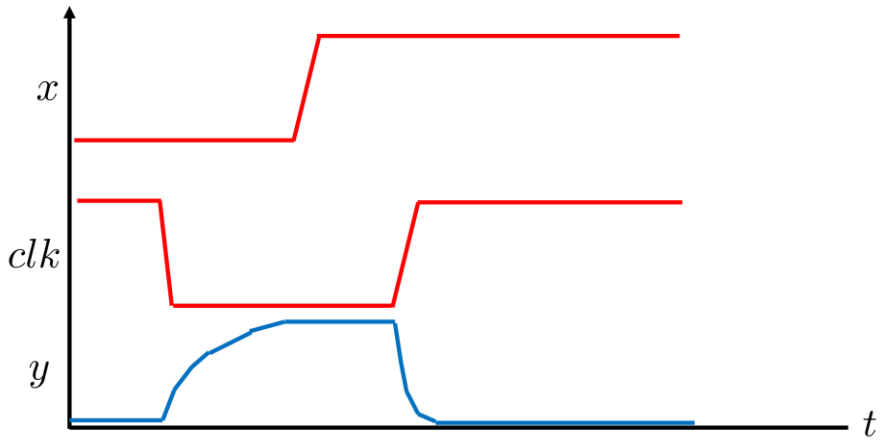
Dynamic CMOS INV

1. Precharge output & apply input (here 1)



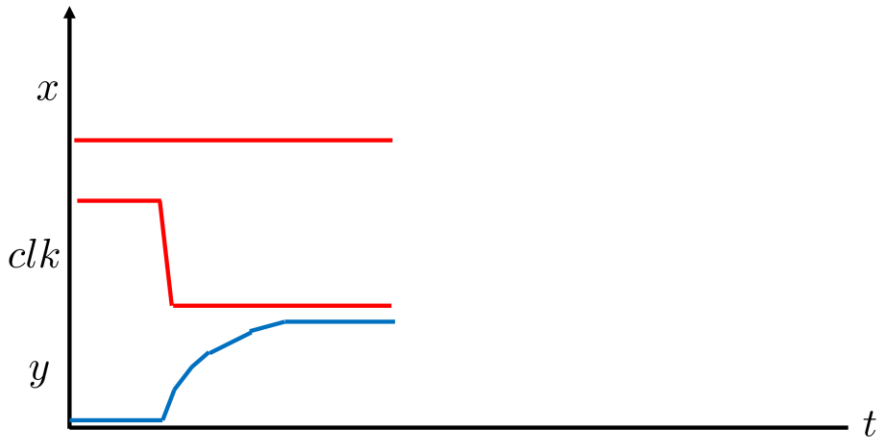
Dynamic CMOS INV

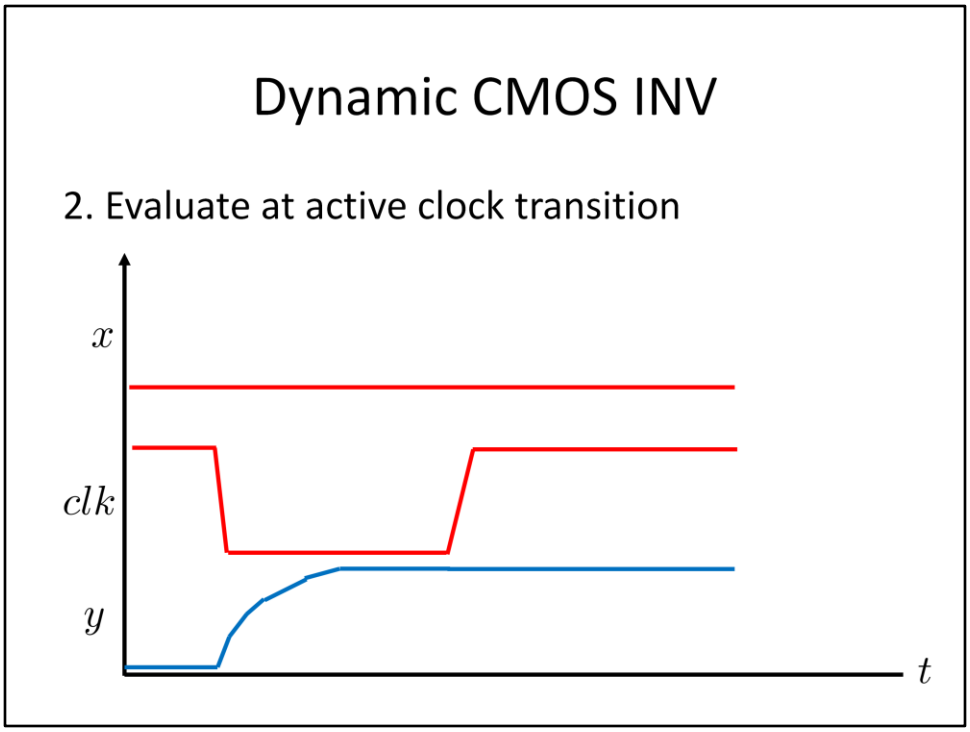
2. Evaluate at active clock transition



Dynamic CMOS INV

1. Precharge output & apply input (here 0)





gate can only make a 1->0 transition or remain at 1.

Dynamic CMOS Properties

Glitches:

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

Speed:

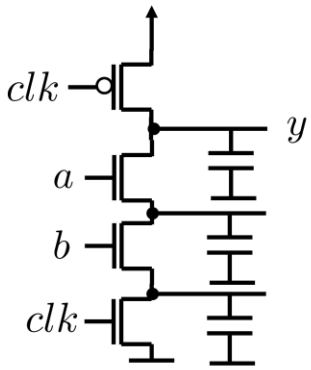
+ very fast (only n stack & smaller load & 0 delay for 1 output)

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

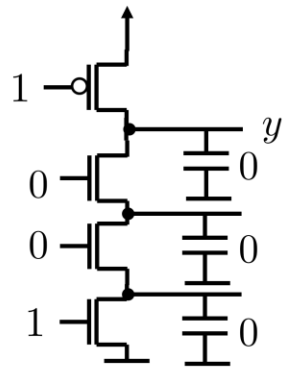
Charge Sharing



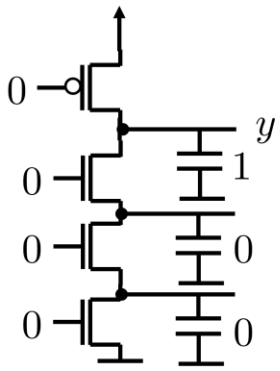
parasitic capacitance at intermediate points -> these can charge instead of the main (output) capacitance -> “charge sharing”

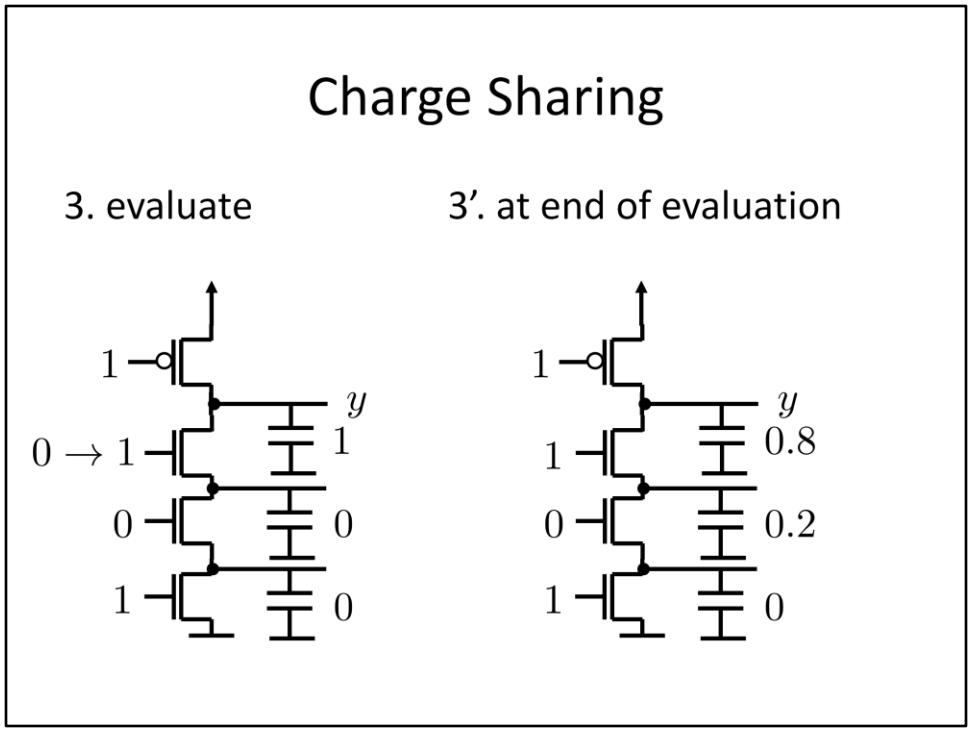
Charge Sharing

1. evaluate



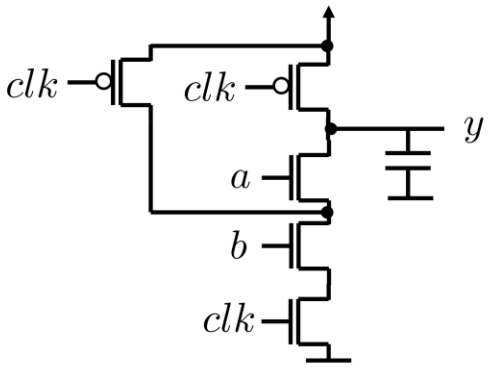
2. next charging





current from output capacitance to parasitic capacitance below
at end of evaluation phase: only a weak 1 at output! -> this can cause problems from higher current in next gate to wrong values in next gate.

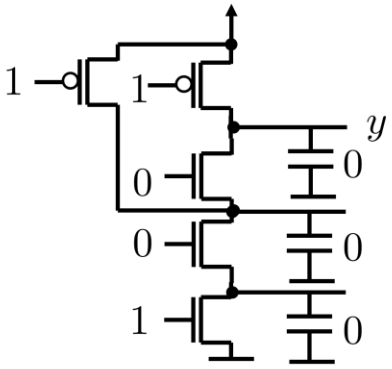
Charge Sharing Workaround



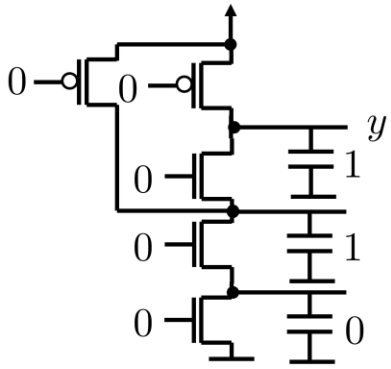
we also load the parasitic capacitance in between.

Charge Sharing Workaround

1. evaluate

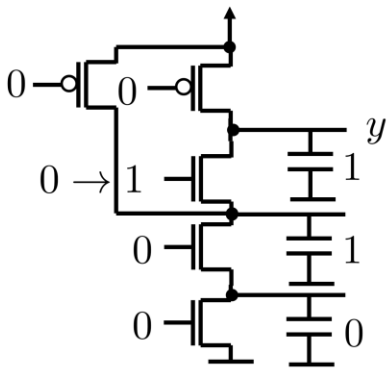


2. next charging

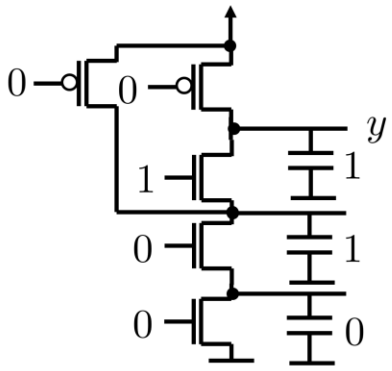


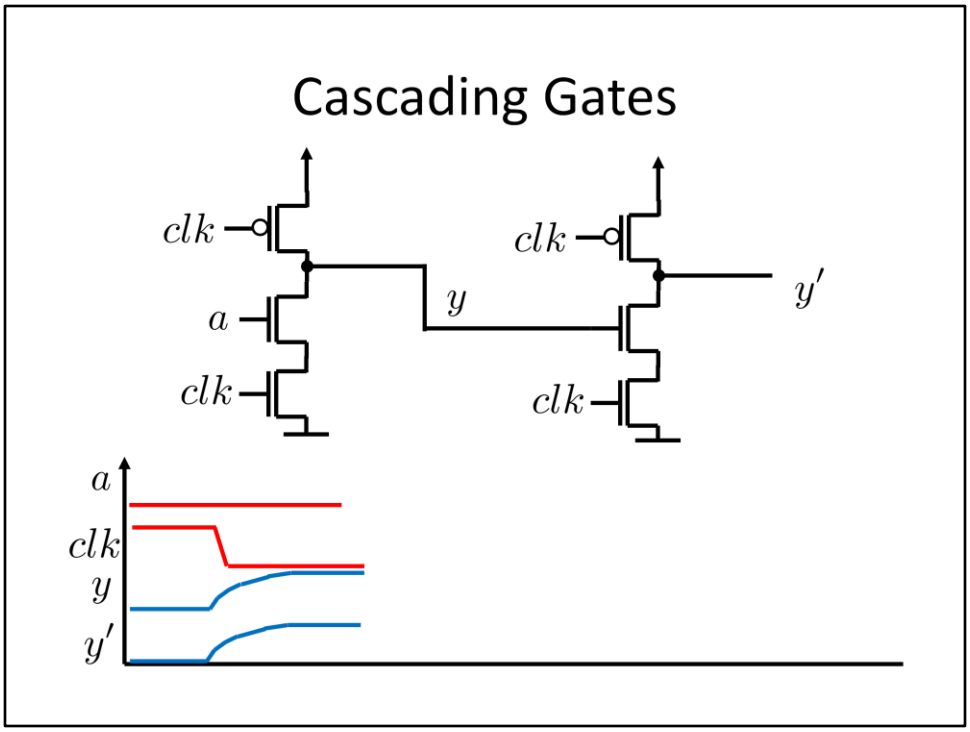
Charge Sharing Workaround

3. evaluate

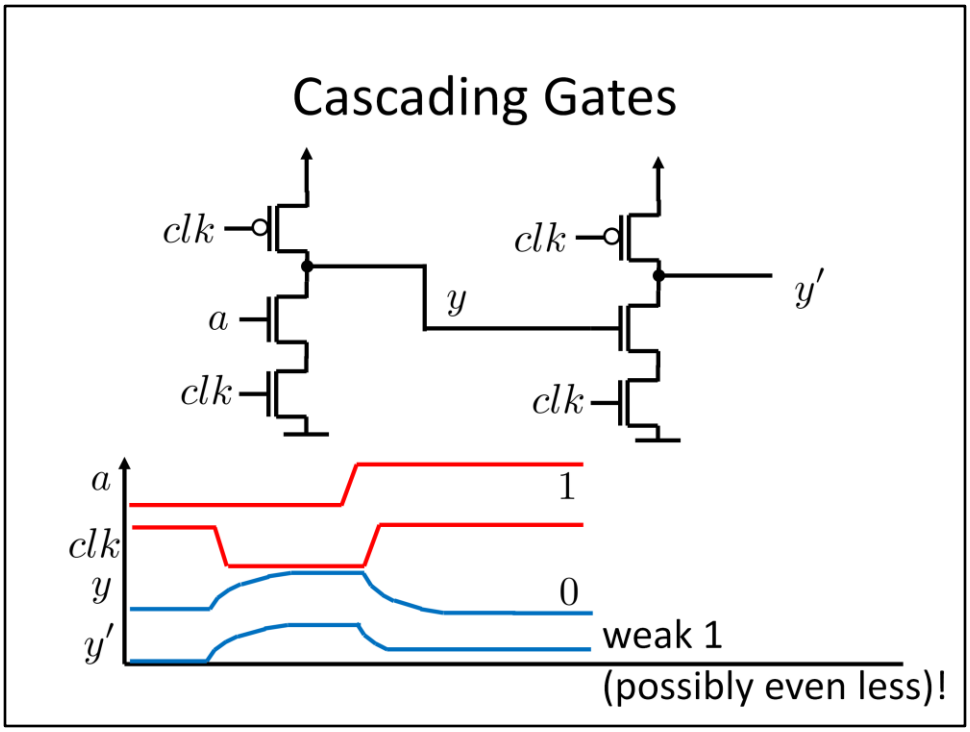


3'. at end of evaluation

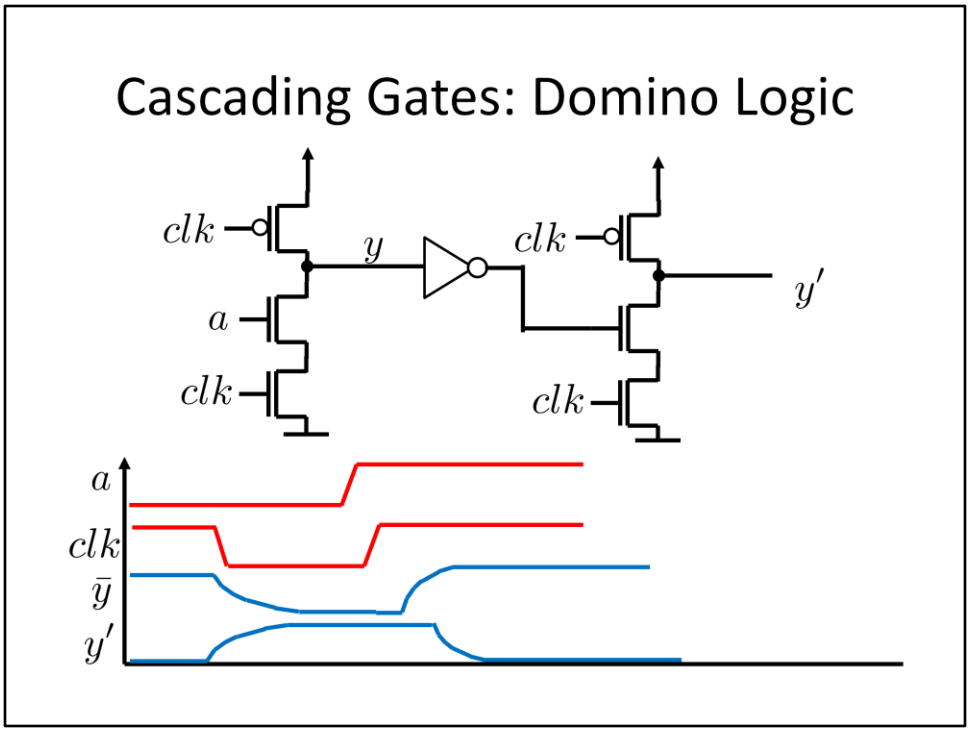




precharging both y and y’ during the negative clock phase.



y' discharges until y has reached threshold voltage of nMOS. The problem is that both evaluate their input when the active clock edge comes,
but the output of the first INV needs time to travel to the input of the next INV.
-> 1. We could delay the clock of the second INV (which is not a good solution in presence of lots of dependencies)
-> 2. Or, we make sure that all outputs and thus also next gate inputs are initially 0 and only in case they are evaluated to 1, they get 1.
-> invert them



in the figure: the INV in the middle is a standard CMOS inverter.

during precharge -> all next gate inputs are charged to 0 -> no problem.

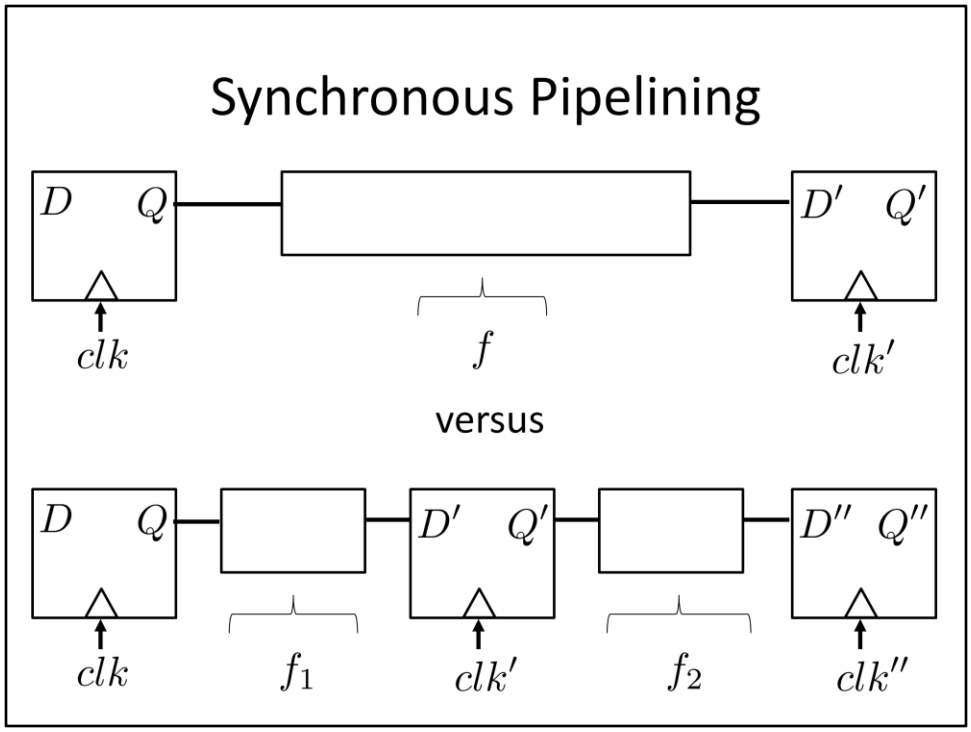
But mind: this is now not an identity gate anymore since we invert in-between.
Domino-logic makes only allows to directly implement non-inverting gates.
If we need negated outputs, we need to negate after the domino circuit with a standard CMOS inverter.

Beyond classical circuit design
lecture 10.75

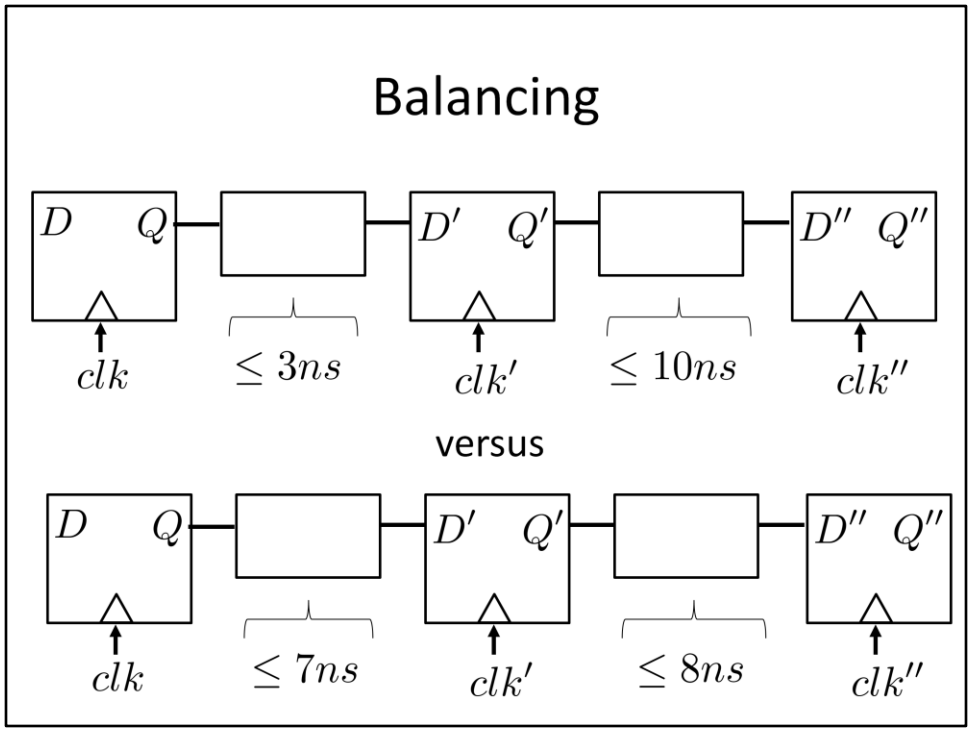
Pipelining

Pipelining

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



pipelining: split up computation into intermediate steps.
Here f(x) = f_2(f_1(x)).



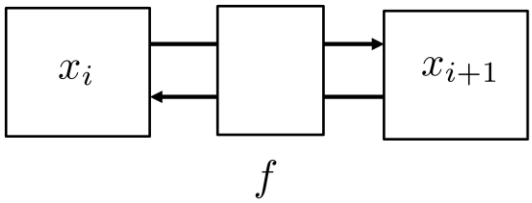
balancing/pipelining if
- some computations have high latency, others not -> split up into equal parts and get smaller balanced latency
- reusing hardware components

-clock skew = maximum difference in time, corresponding active transitions occur at any two clk ports. Ideally 0 skew. Challenges due
to layout and delay variations.

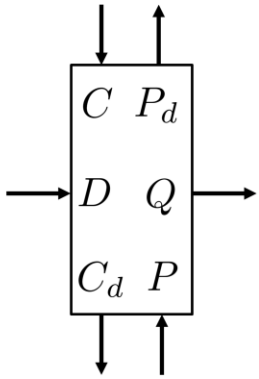
Clockless Pipelining

... with static logic

Local handshaking instead of global clock

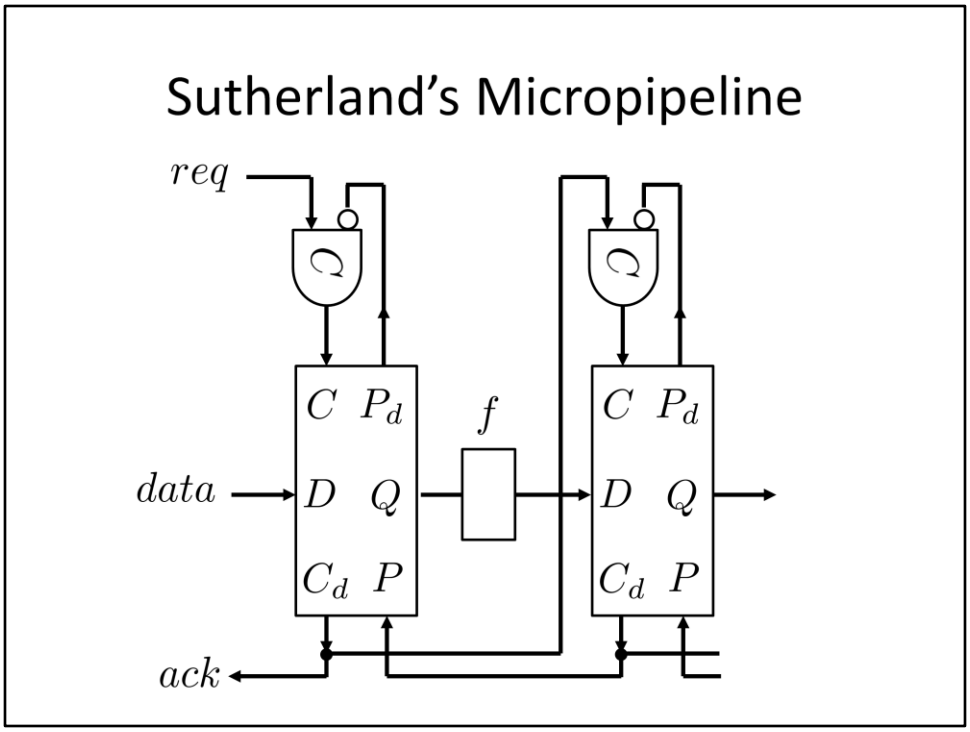


Capture-pass latch



$C = P \rightarrow$ pass D to Q

$C \neq P \rightarrow$ hold Q



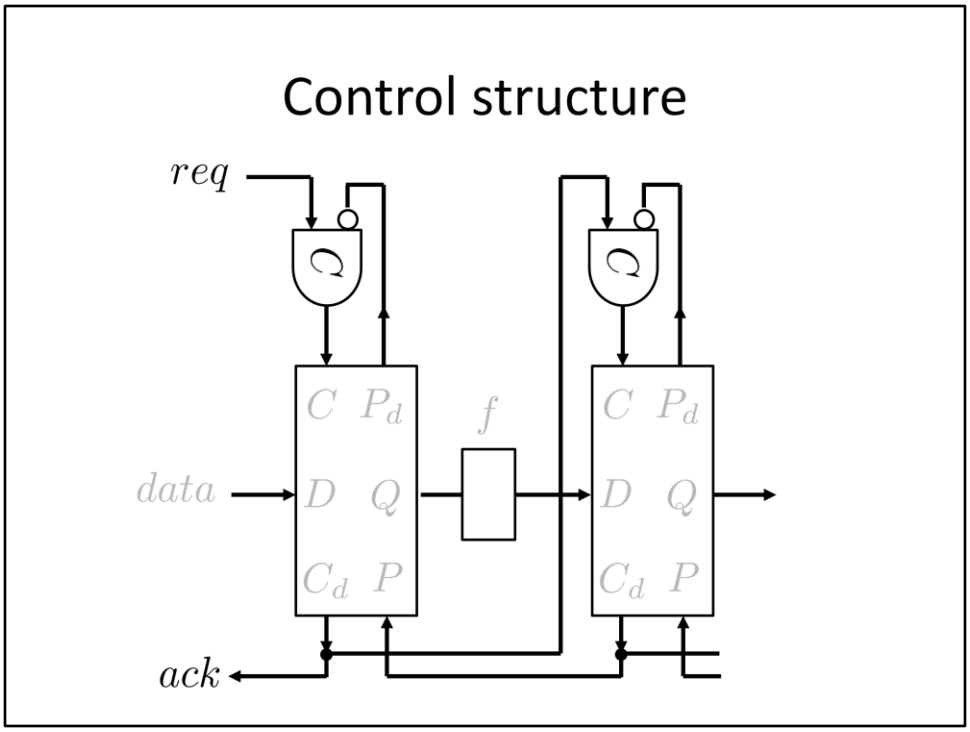
initially all C-element states 0.
C_d (= capture done) is a delayed C (= capture).
P_d (= pass done) is a delayed P (= pass)

intially P = C = P_d = C_d = 0.
loop:
when C makes a 0-1 transition -> captures data on D and holds it on Q.
when P makes a 0-1 transition -> pass data from D to Q.
when C makes a 1-0 transition -> captures data on D and holds it on Q.
when P makes a 1-0 transition -> pass data from D to Q.
end loop.

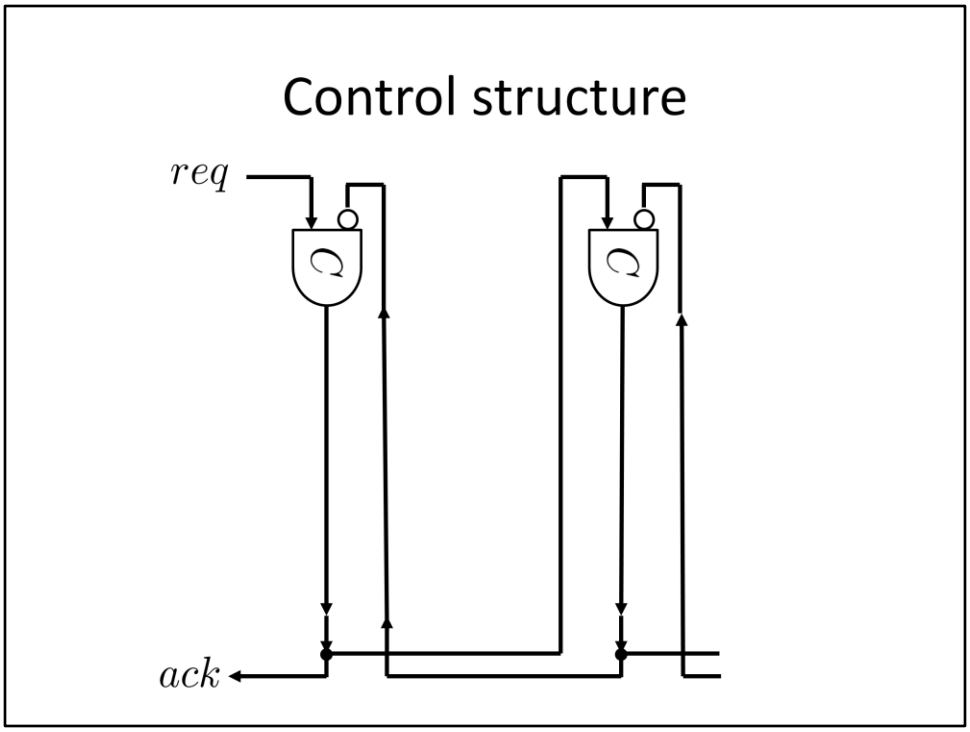
Sutherland's Micropipeline

- 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

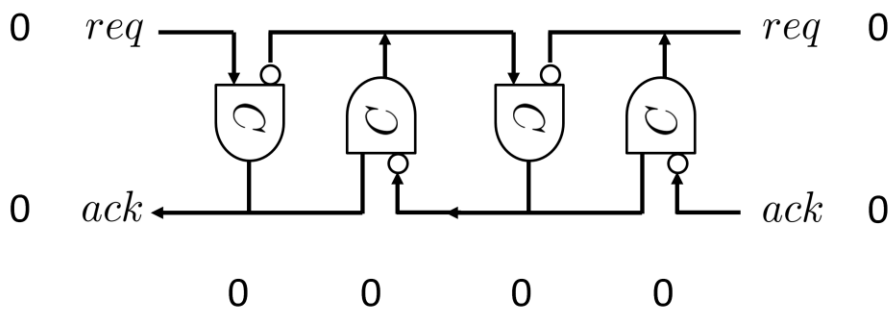


only the control structure

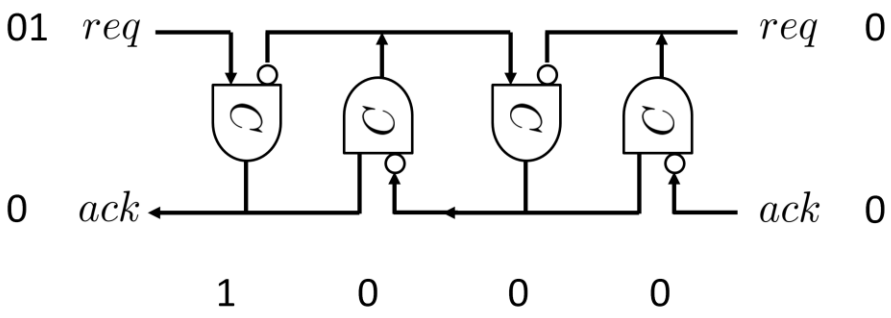


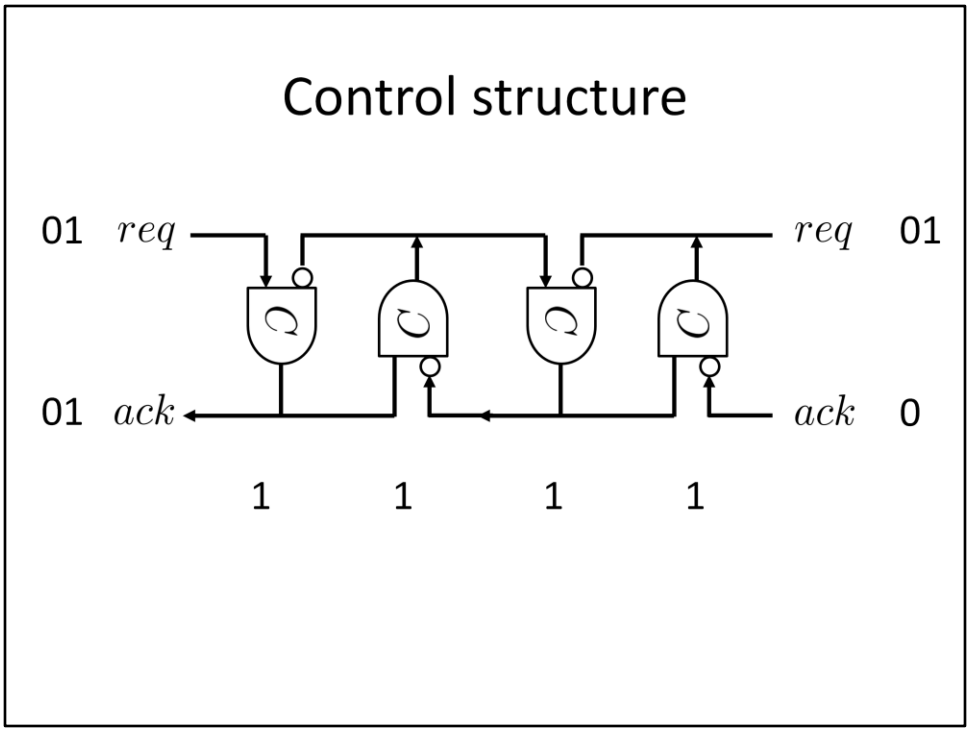
only the control structure

Control structure



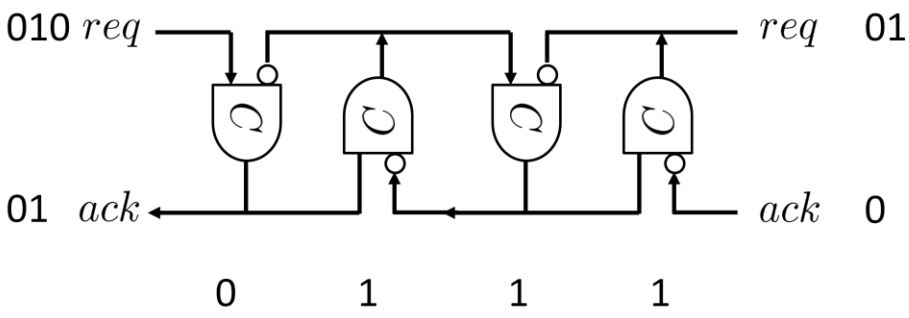
Control structure



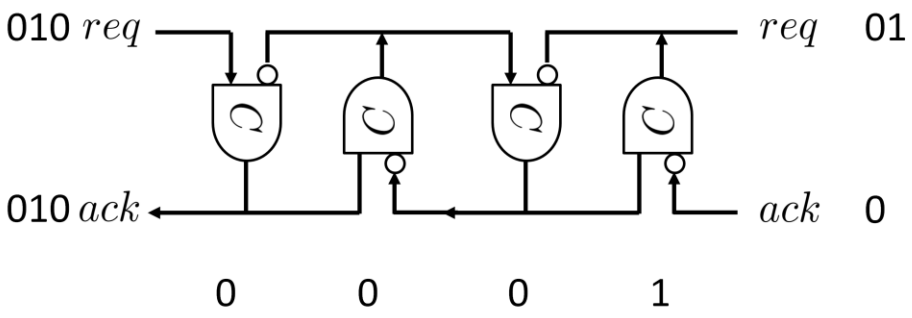


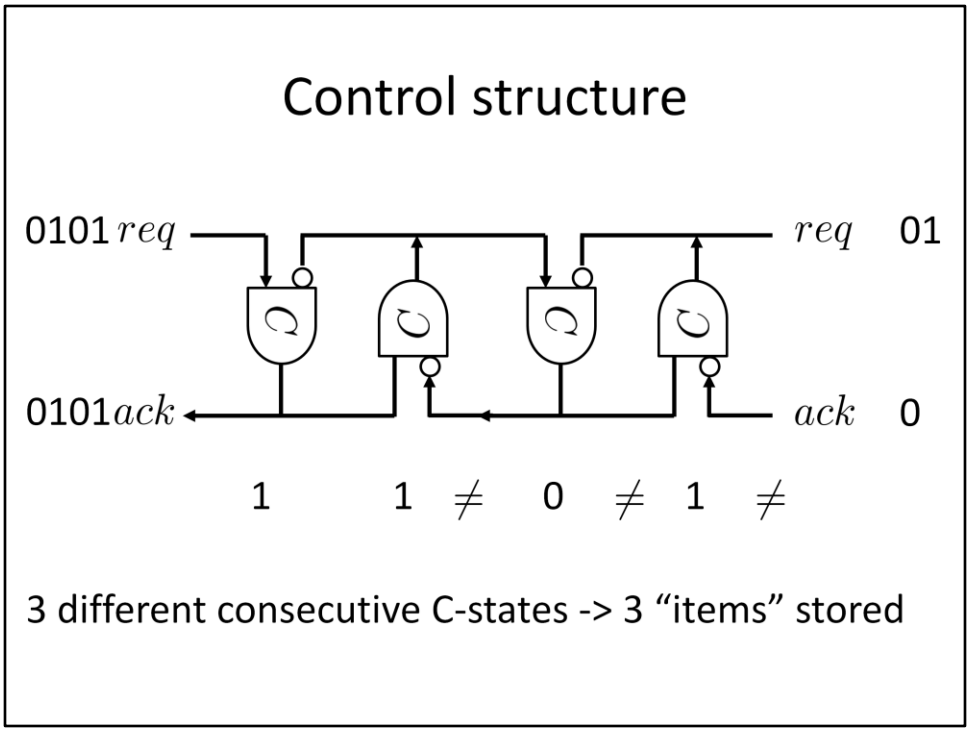
assume we do not ack at right interface. -> ack right remains 0.

Control structure



Control structure

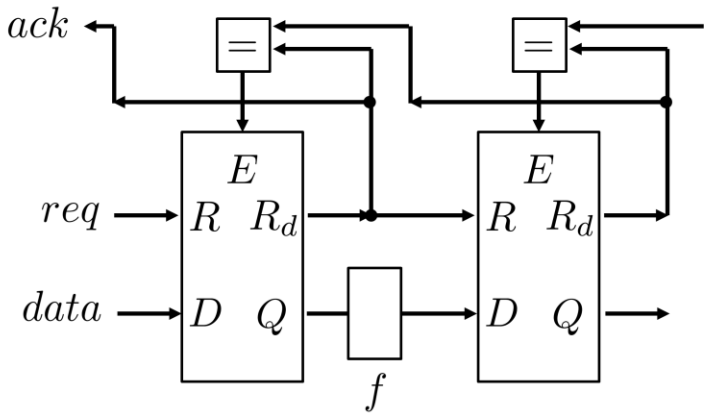




+ one more req...

remember: different P(ass) and C(apture) -> latch is in capture mode.
-> 3 consecutive different C-element states -> 3 latches that would sit in between would be in capture mode.

Mousetrap pipeline



latches here with positive enable.
"=" gate is implemented by XNOR

Mousetrap pipeline

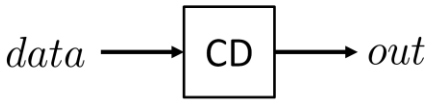
- 2-phase handshaking
- Bundled data
- Initially all latches transparent (= pass mode)

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

Clockless Pipelining

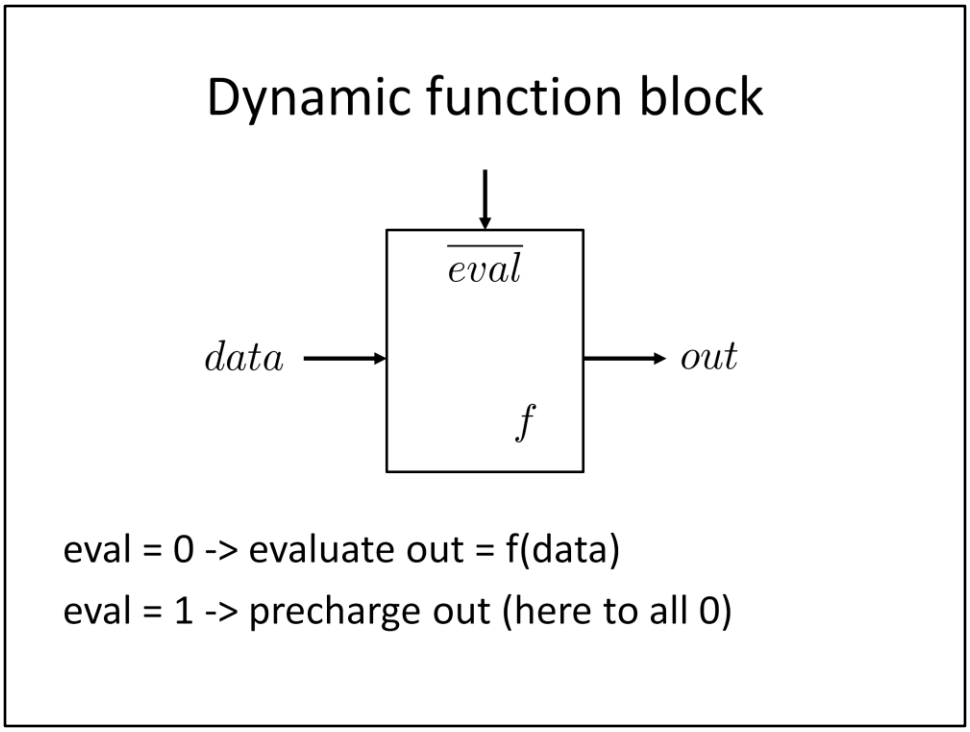
... with dynamic logic

Completion dedection

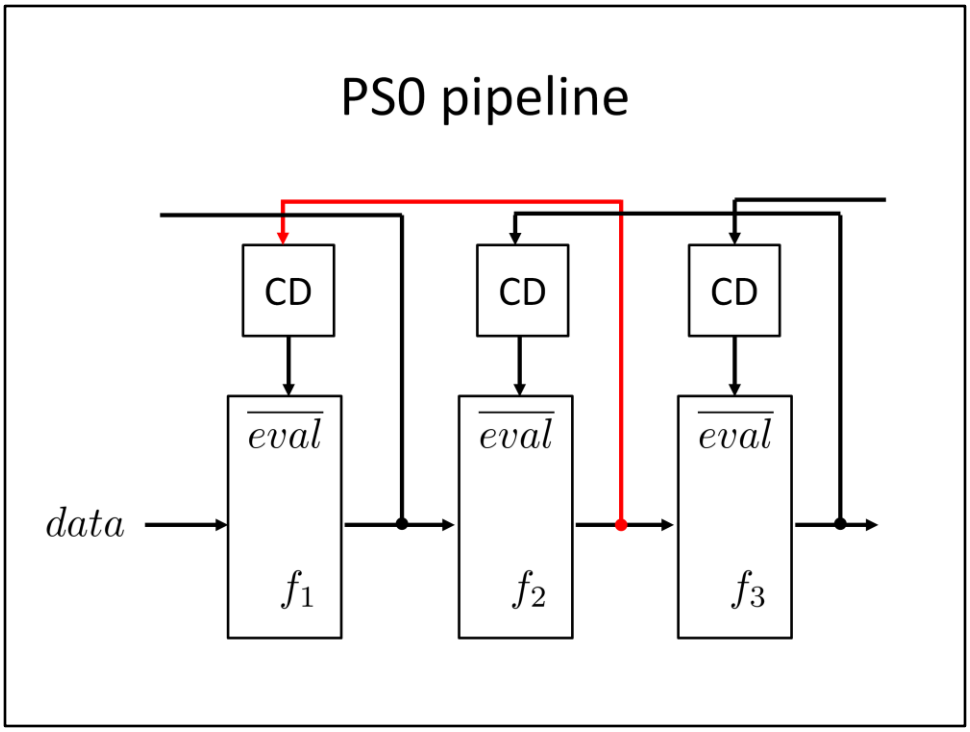


$$*[v(data); out\uparrow; n(data); out\downarrow]$$

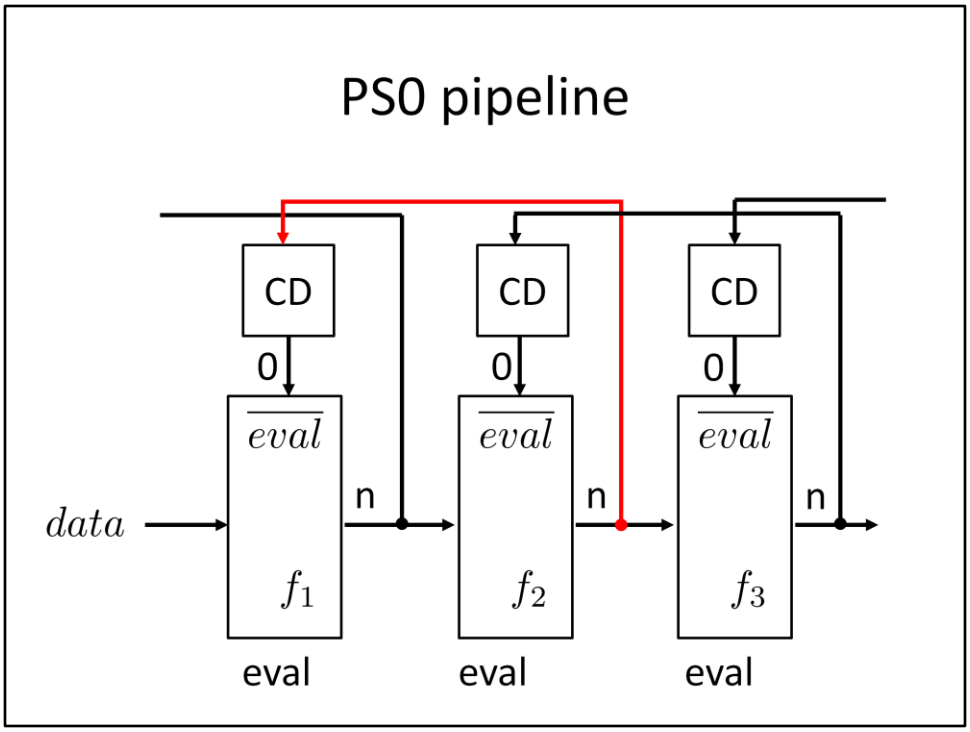
v, n... valid/neutral.
data... possibly several bit, requires proper coding of data, e.g., dual-rail encoding (see lecture on encodings)
out... single bit.



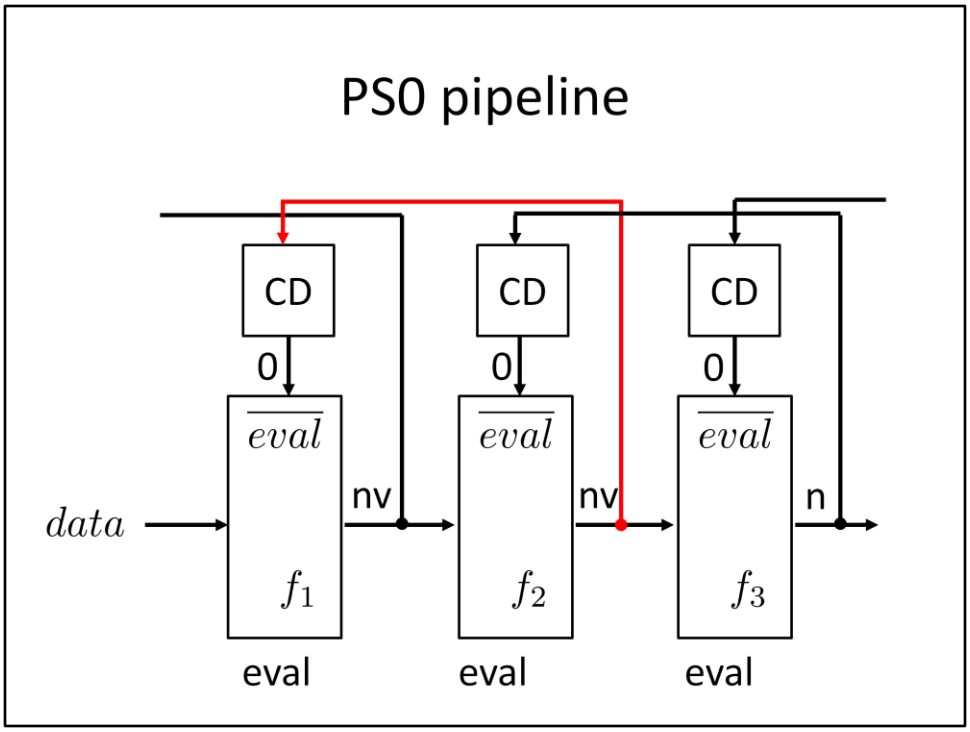
data and out can be several bit each.
Both data and out need to be in a Delay-insensitive code (here: dual-rail encoding).
dynamic function with precharge & evaluate phase.

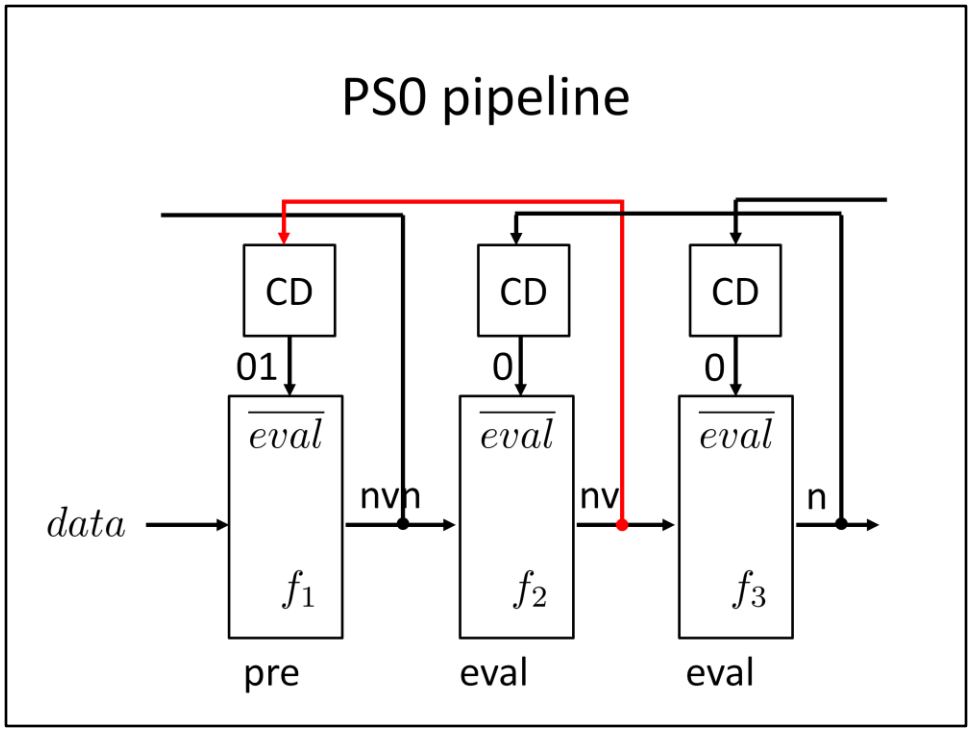


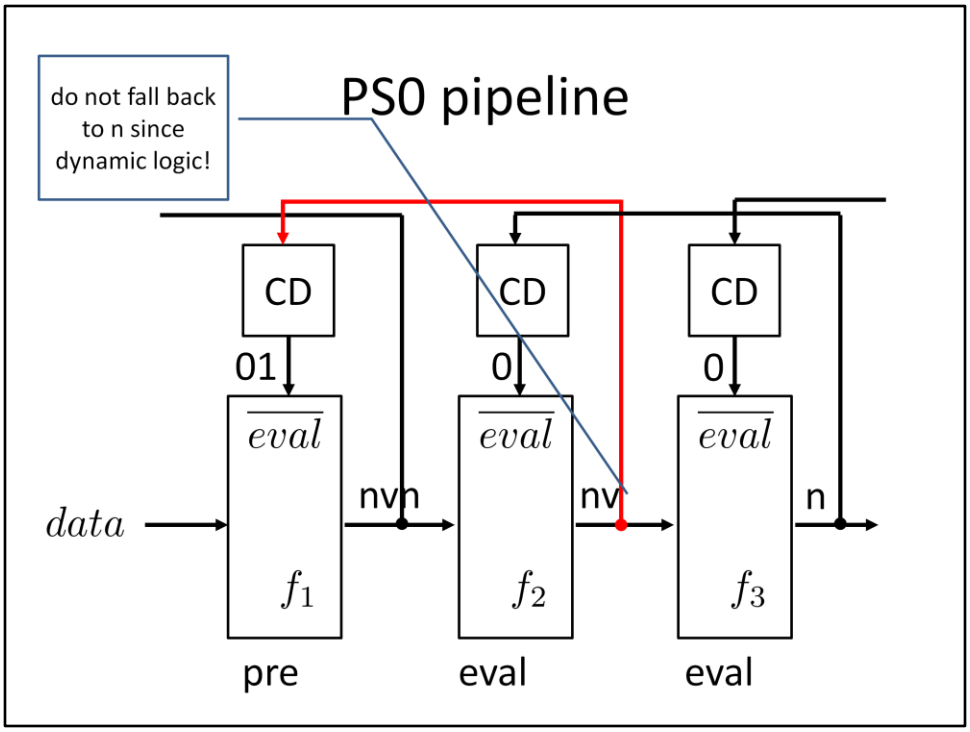
by Williams and Horowitz
see Wiliams: Self-timed rings and their application to division. PhD thesis, 1991.



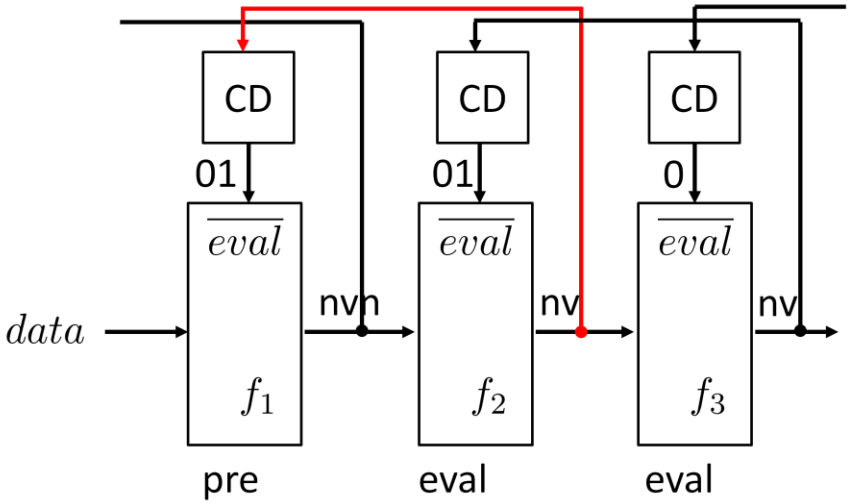
initially:
all data neutral (= all 0)
-> CDs output 0.
-> all dynamic function blocks in evaluate mode.

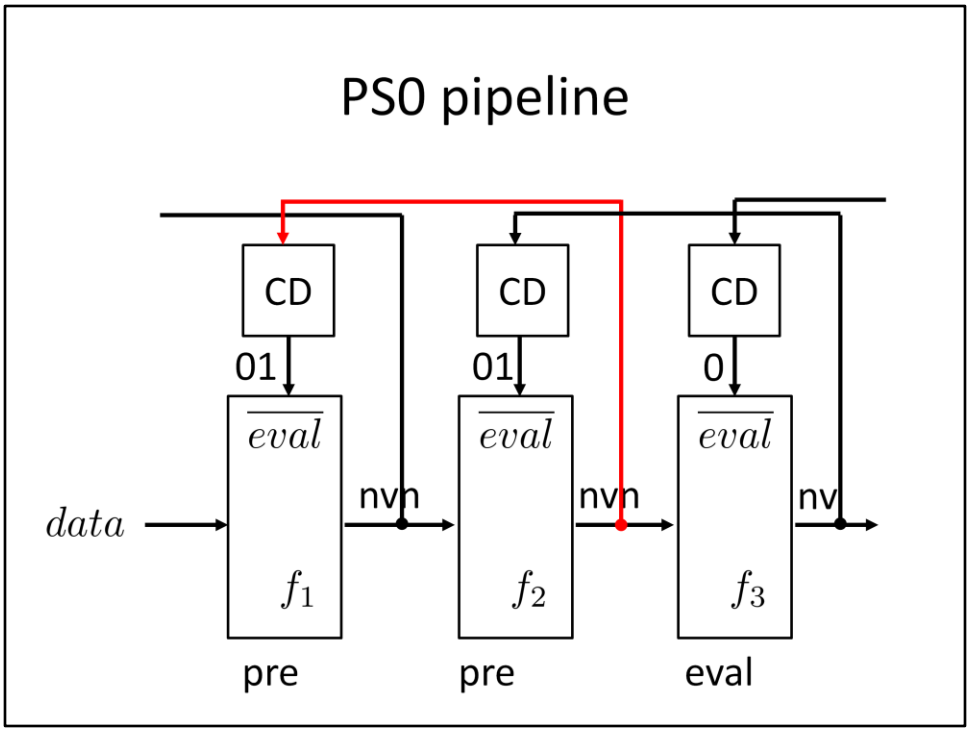

PS0 pipeline
CD
CD
CD
0
0
0
$\overline{eval}$
$\overline{eval}$
$\overline{eval}$
nv
nv
n
$data$
f_1
f_2
f_3
eval
eval
eval


PS0 pipeline
CD
CD
CD
01
0
0
$\overline{eval}$
$\overline{eval}$
$\overline{eval}$
$data$
nvn
nv
n
f_1
f_2
f_3
pre
eval
eval

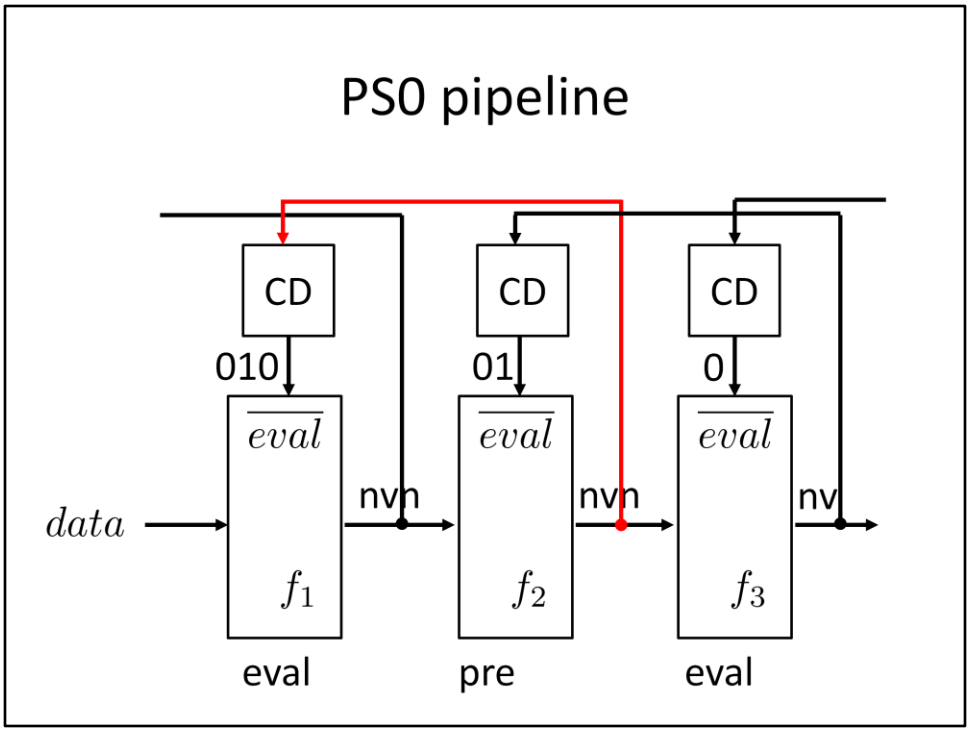

do not fall back to n since dynamic logic!
PS0 pipeline
CD
CD
CD
01
0
0
$\overline{eval}$
$\overline{eval}$
$\overline{eval}$
data
nvn
nv
n
f_1
f_2
f_3
pre
eval
eval

PS0 pipeline

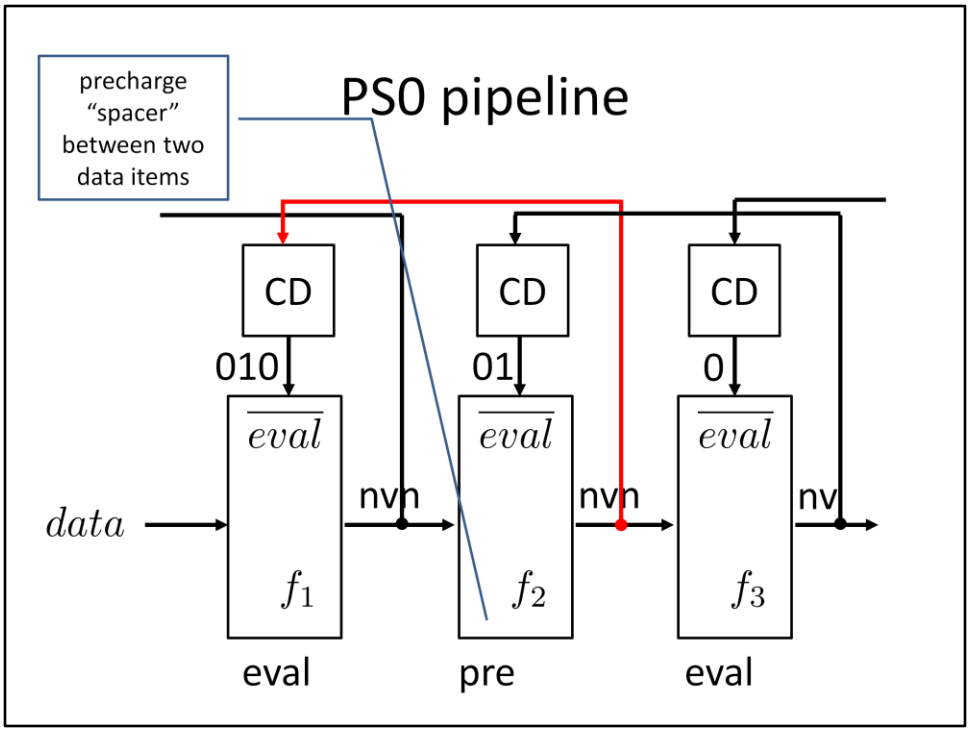




now fall back to n(eutral) by precharging



trigger first stage to evaluate again.


precharge
“spacer”
between two
data items
PS0 pipeline
CD
CD
CD
010
01
0
$\overline{eval}$
$\overline{eval}$
$\overline{eval}$
nvn
nvn
nv
$data$
f_1
f_2
f_3
eval
pre
eval

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.