Beyond classical chip design lecture 5

Communication

Further Reading

Alain J. Martin: *Synthesis of Asynchronous VLSI Circuits.* Tech report California Institute of Technology, 1991.

Alain J. Martin and Mika Nyström: *Asynchronous techniques for system-on-chip design.* Proceedings of the IEEE Volume 94, Issue 6:1089 - 1120, June 2006.

What we had...

- Communicating Hardware Processes as high-level spec.

- fundamental problem:

  Sequencing: a;b

  Multishot-sequencing: *[a;b]

... now look at communication

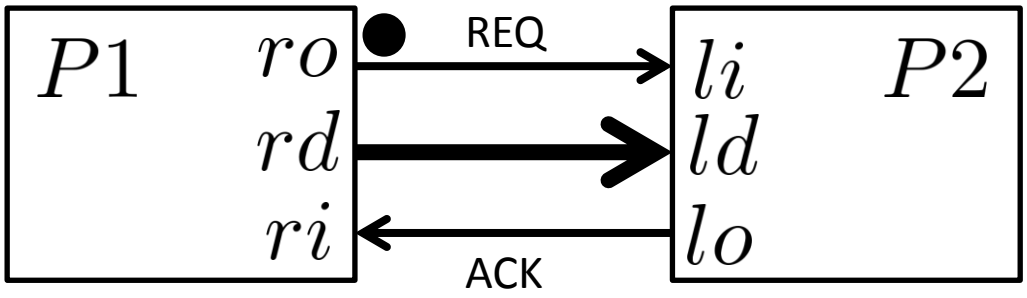
multishot-sequencing

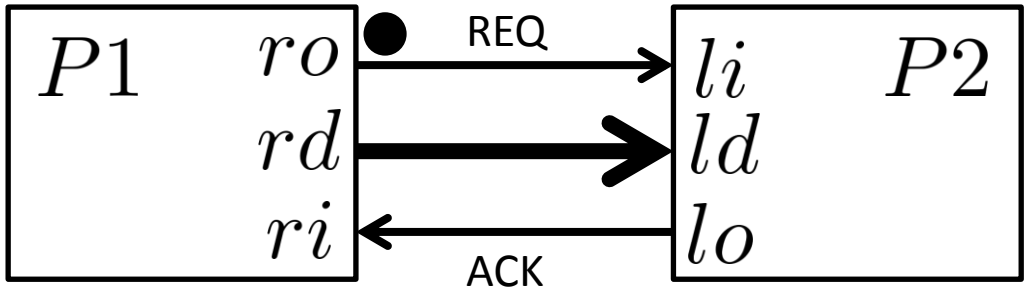
communication:

a <-> write(x)

b <-> y = read(x)

Bundled Data

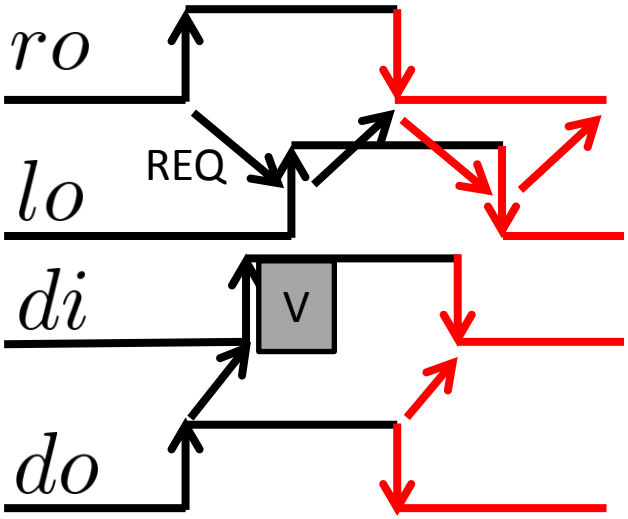


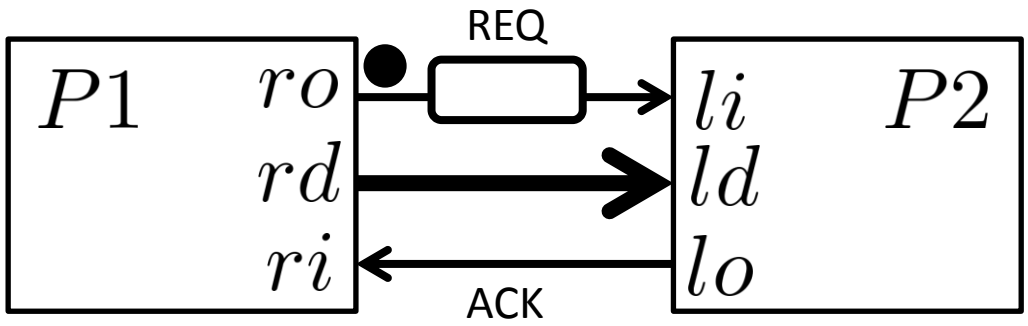


$$P1 : rd := x; ro\uparrow; [ri]; ro\downarrow; [\neg ri]$$

$$P2 : [li]; y := ld; lo\uparrow; [\neg li]; lo\downarrow$$

REQ ACK clREQ clACK

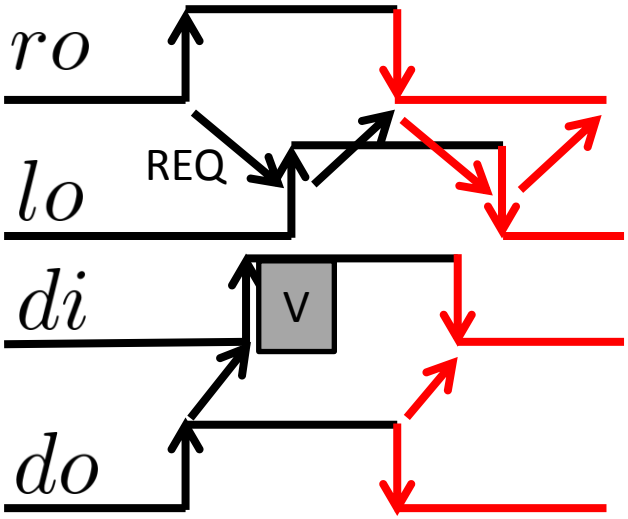




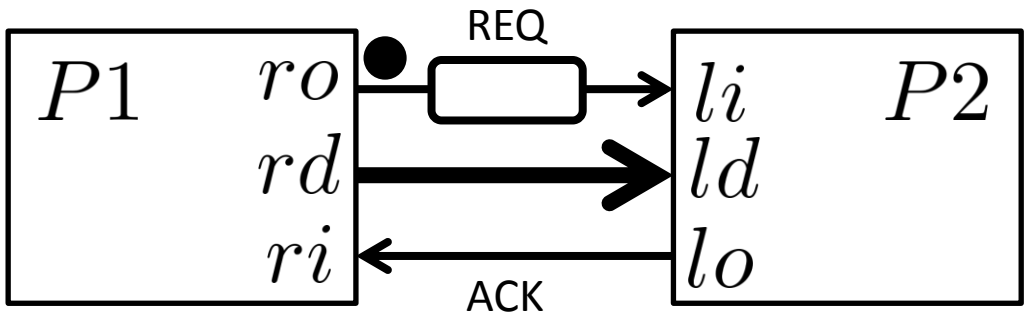
$$P1 : rd := x; ro\uparrow; [ri]; ro\downarrow; [\neg ri]$$

$$P2 : [li]; y := ld; lo\uparrow; [\neg li]; lo\downarrow$$

REQ ACK clREQ clACK



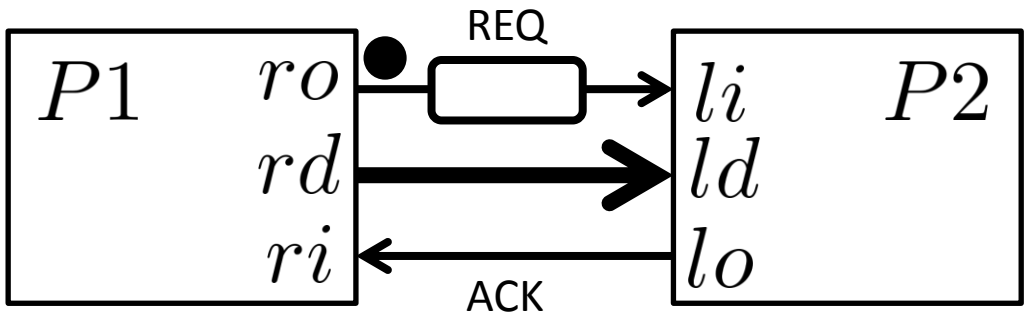
Bundled Data

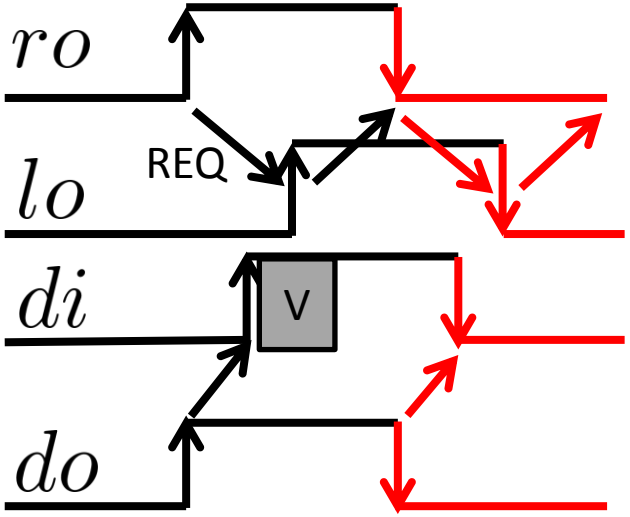


… requires one-sided constraint

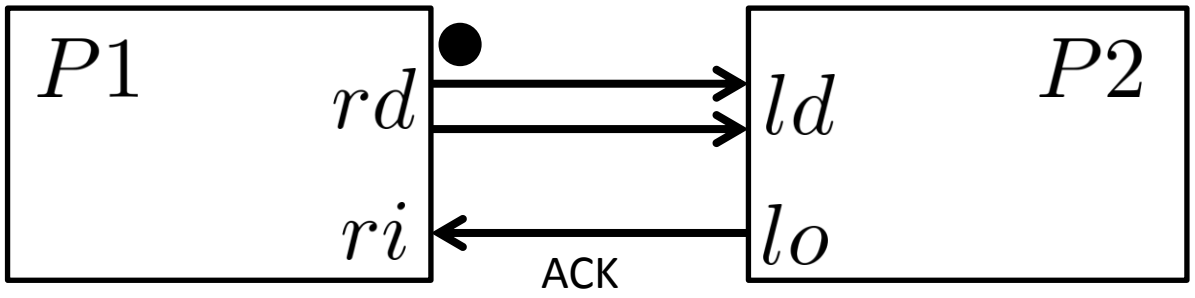
Can we do without?

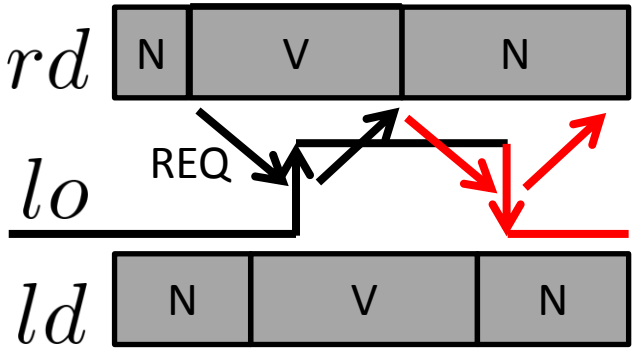
Bundled Data



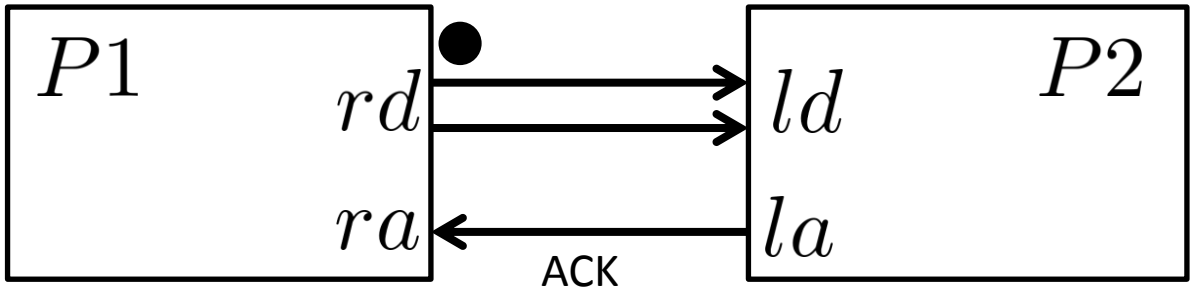


Delay-Insensitive Codes



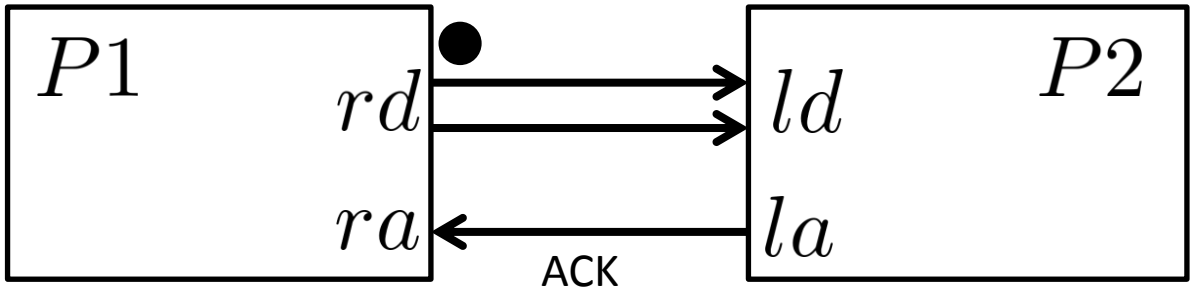


V(word), N(word)



$$P1 : rd := x; [ra]; rd := neutral; [\neg ra]$$

$$P2 : [v(ld)]; y := ld; la \uparrow; [n(ld)]; la \downarrow$$



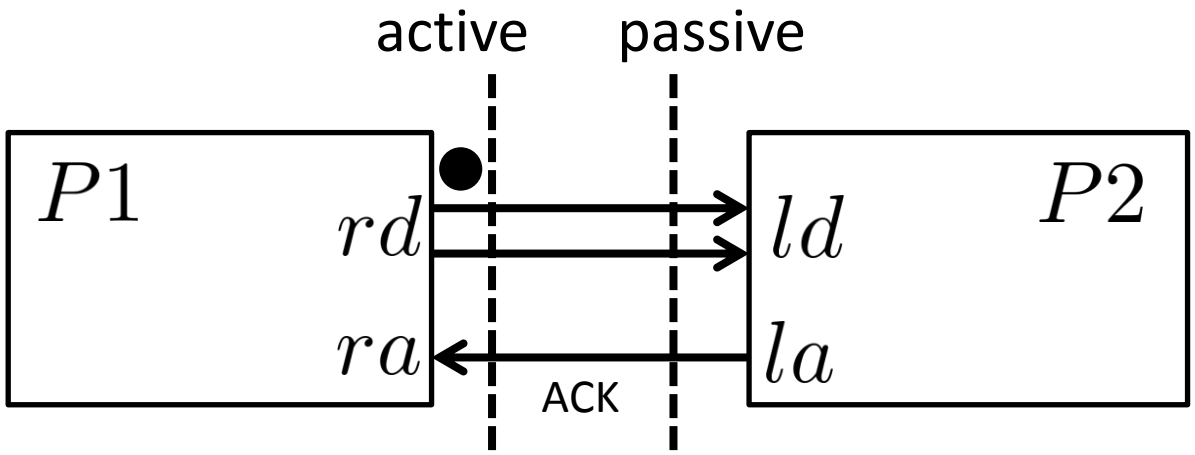
$$P1 : rd := x; [ra]; rd := neutral; [\neg ra]$$

$$P2 : [v(ld)]; y := ld; la \uparrow; [n(ld)]; la \downarrow$$

$$P1 : rd := x; [v(ra)]; rd := neutral; [n(ra)]$$

$$P2 : [v(ld)]; y := ld; la \uparrow; [n(ld)]; la \downarrow$$

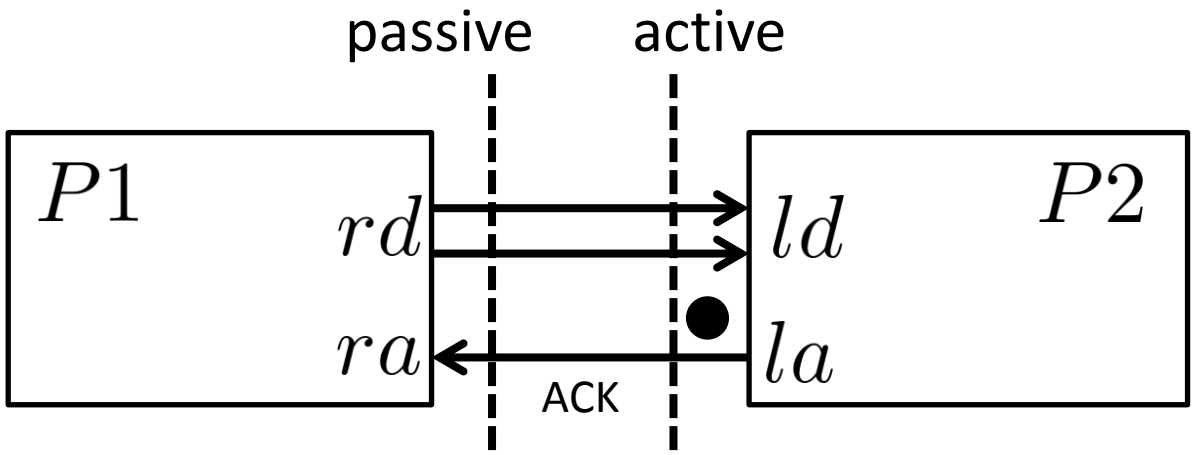
Active vs Passive



active $P1 : rd := x; [v(ra)]; rd := neutral; [n(ra)]$

passive $P2 : [v(ld)]; y := ld; la \uparrow; [n(ld)]; la \downarrow$

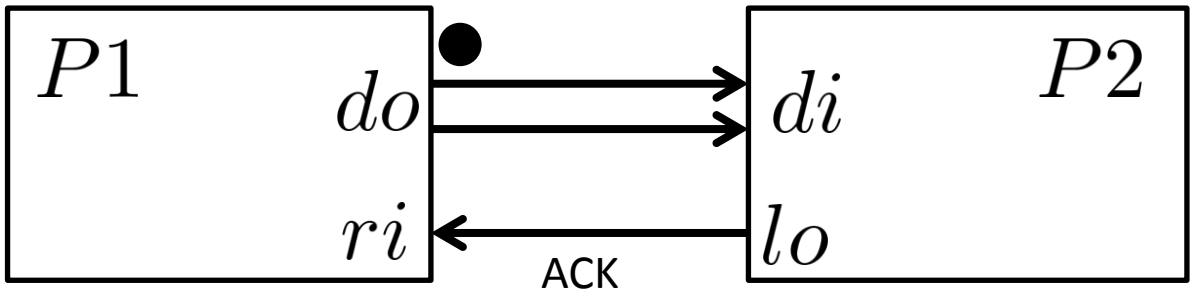
Active vs Passive



passive $P1 : [n(ra)]; rd := x; [v(ra)]; rd := neutral$

active $P2 : la \uparrow; [n(ld)]; la \downarrow; [v(ld)]; y := ld$

First code



2-rails

neutral: 0 0

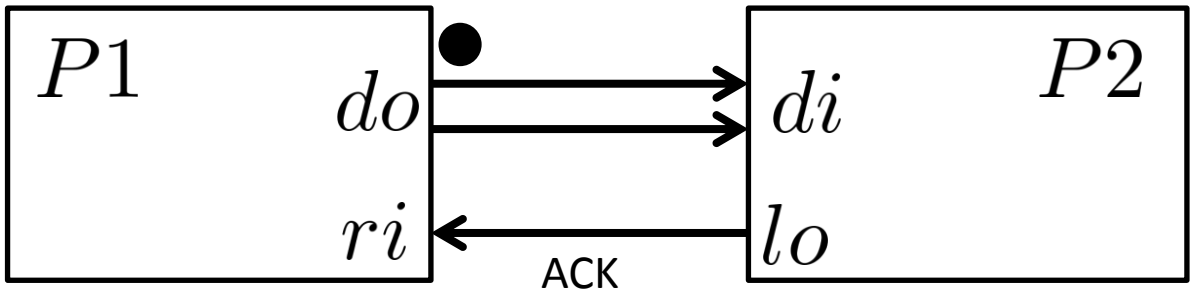
0: 1 0

1: 1 1

transition: 0 1

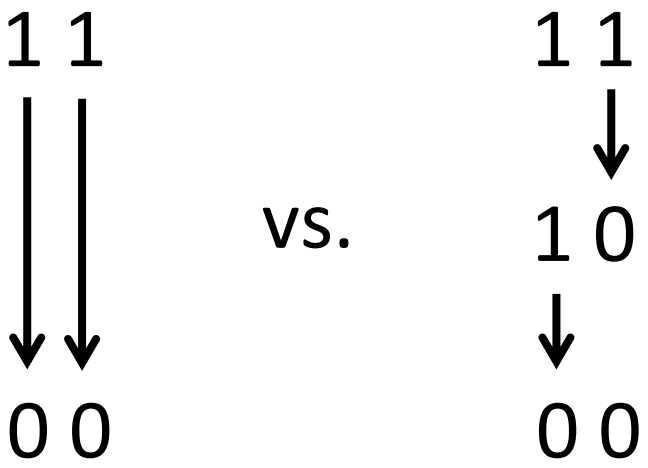
valid: is “0“ or “1“

First code... hmm

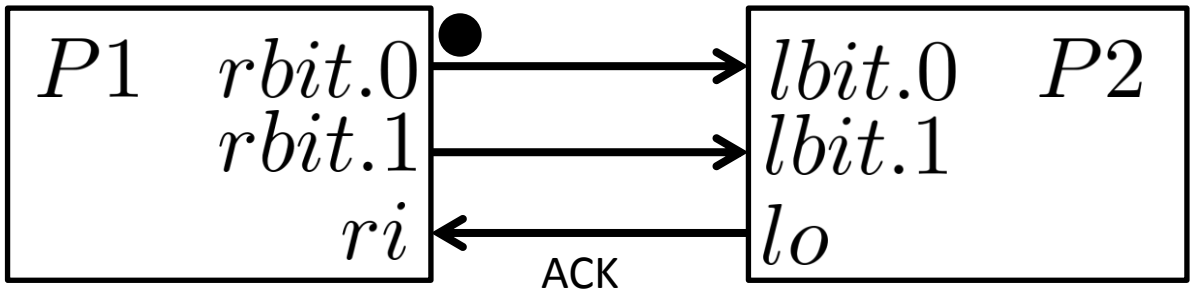


2-rails

neutral:	0 0
0:	1 0
1:	1 1
transition:	0 1



Dual-Rail



2-rails

neutral: 0 0

0: 1 0

1: 0 1

transition: 1 1

Dual-Rail (n bit)

bit0.0 bit0.1 bit1.0 bit1.1 …

2n rails for n bit

1-of-n / one-hot

n-rails for ld(n) bit

neutral:	0 0 0 0
0:	1 0 0 0
1:	0 1 0 0
2:	0 0 1 0
3:	0 0 0 1
transition:	else

Which one to choose...

… depends on

- efficient #wires
- fast & small n(), v() tests
- split & join

-> 1-of-4 splitting, dual-rail

Beyond classical circuit design
lecture 5.5

Circuit types

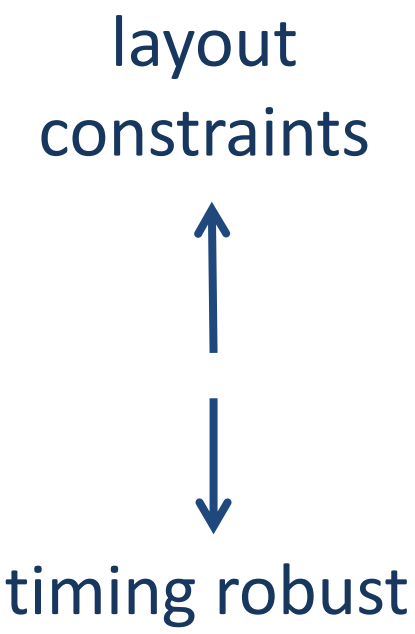
Further Reading

Keller, S.; Katelman, M.; Martin, A.J.: *A Necessary and Sufficient Timing Assumption for Speed-Independent Circuits.* Asynchronous Circuits and Systems (ASYNC'09). 15th IEEE Symposium on, pp. 65 - 76, 2009.

Circuit types

- Synchronous

- Clockless
  - aggressively timed
  - …
  - Speed independent (SI)
  - Quasi-delay insensitive (QDI)
  - Delay insensitive (DI)



Circuit types

- Synchronous

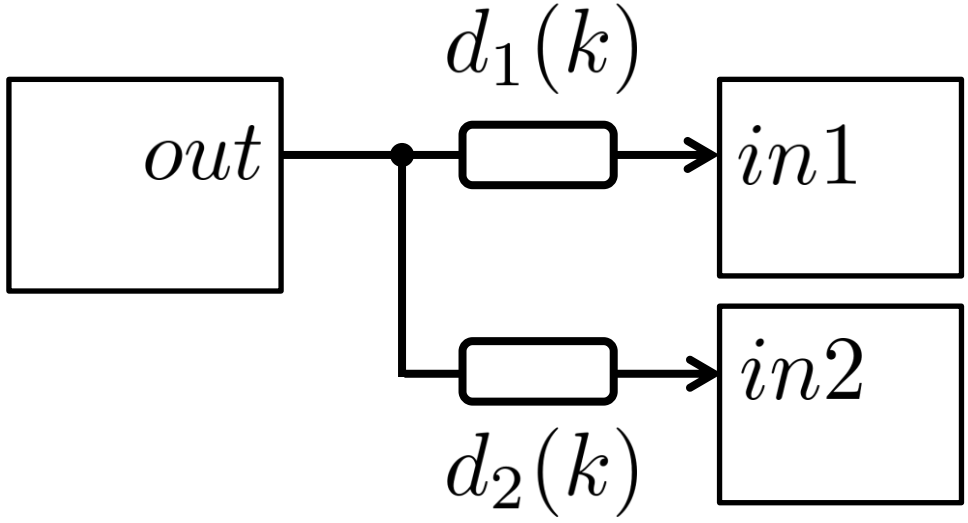
- Clockless
  - aggressively timed
  - …
  - Speed independent (SI)
  - Quasi-delay insensitive (QDI)
  - Delay insensitive (DI)

fast & small (?)

layout constraints

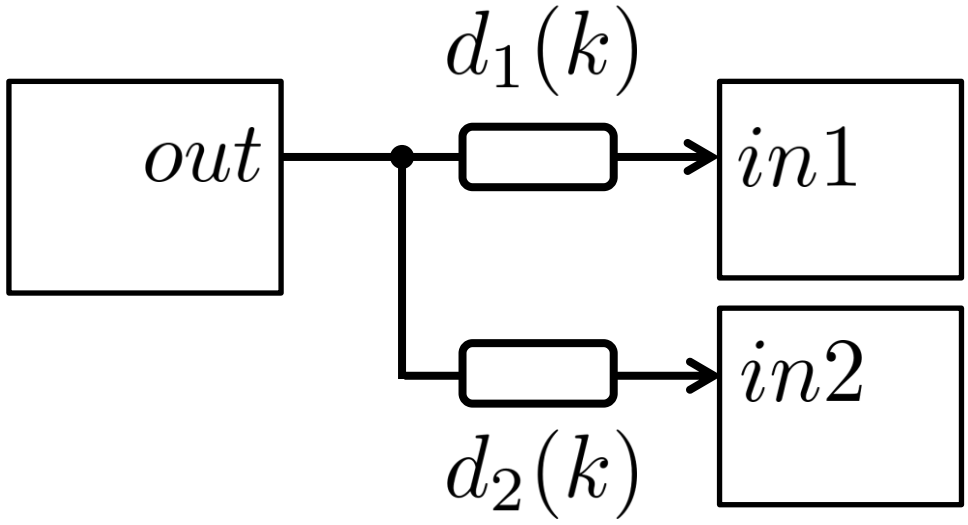
timing robust

Isochronic fork



$$\forall k : d_1(k) \approx d_2(k)$$

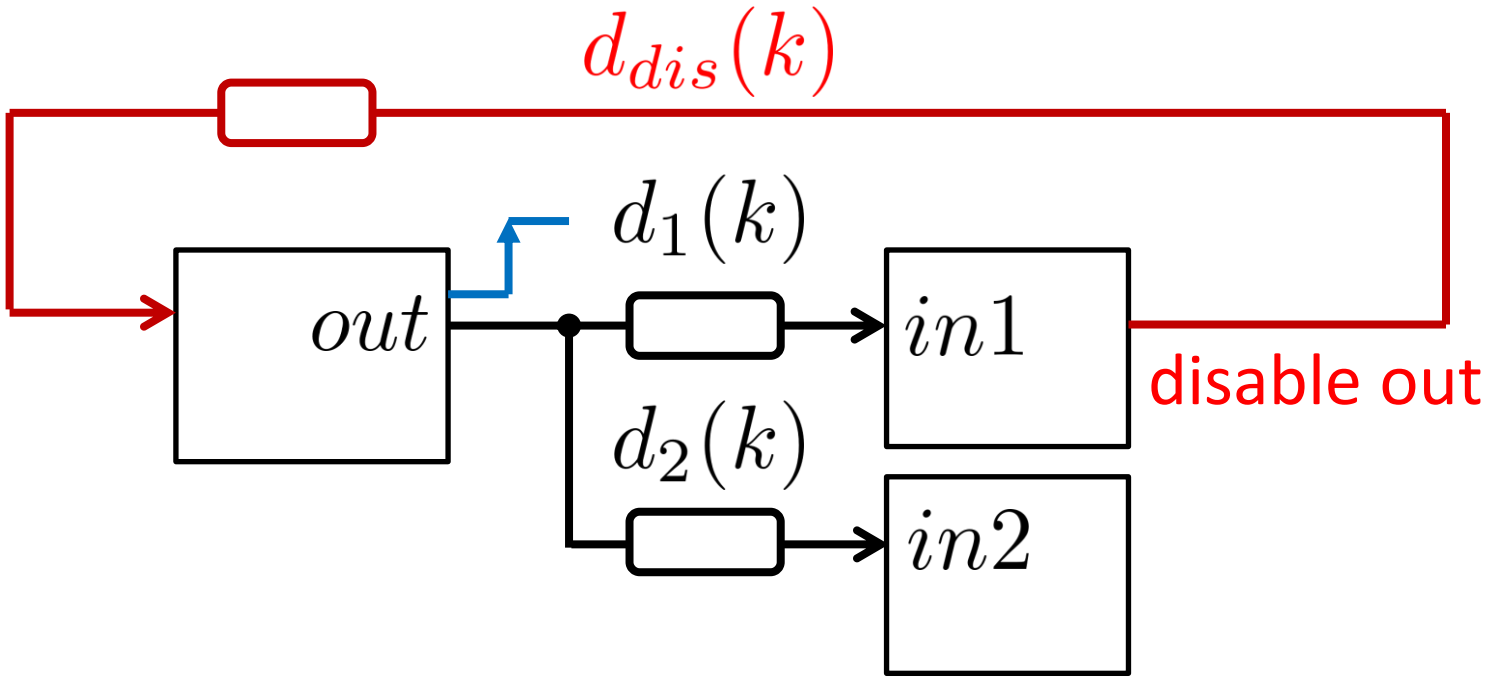
Isochronic fork



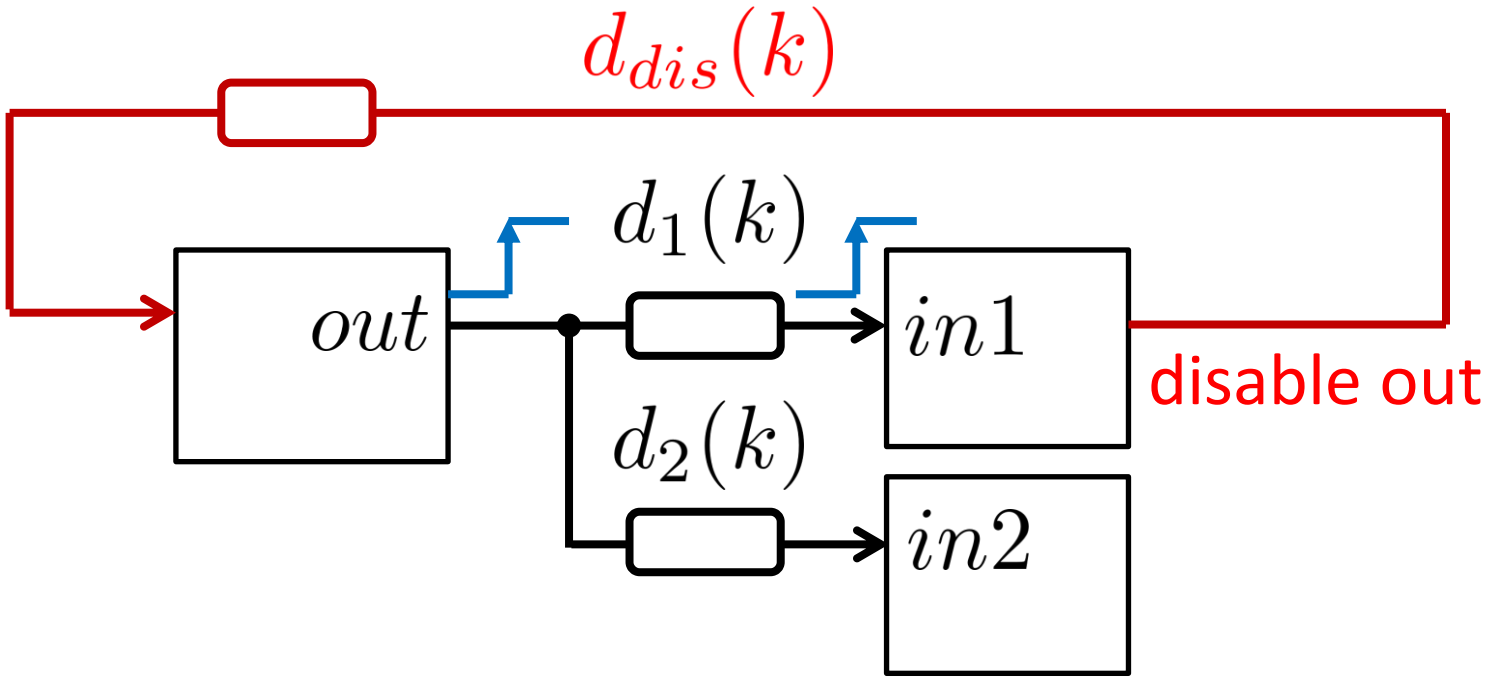
$\forall k : d_1(k) \approx d_2(k)$ ->

two-sided constraint [!]

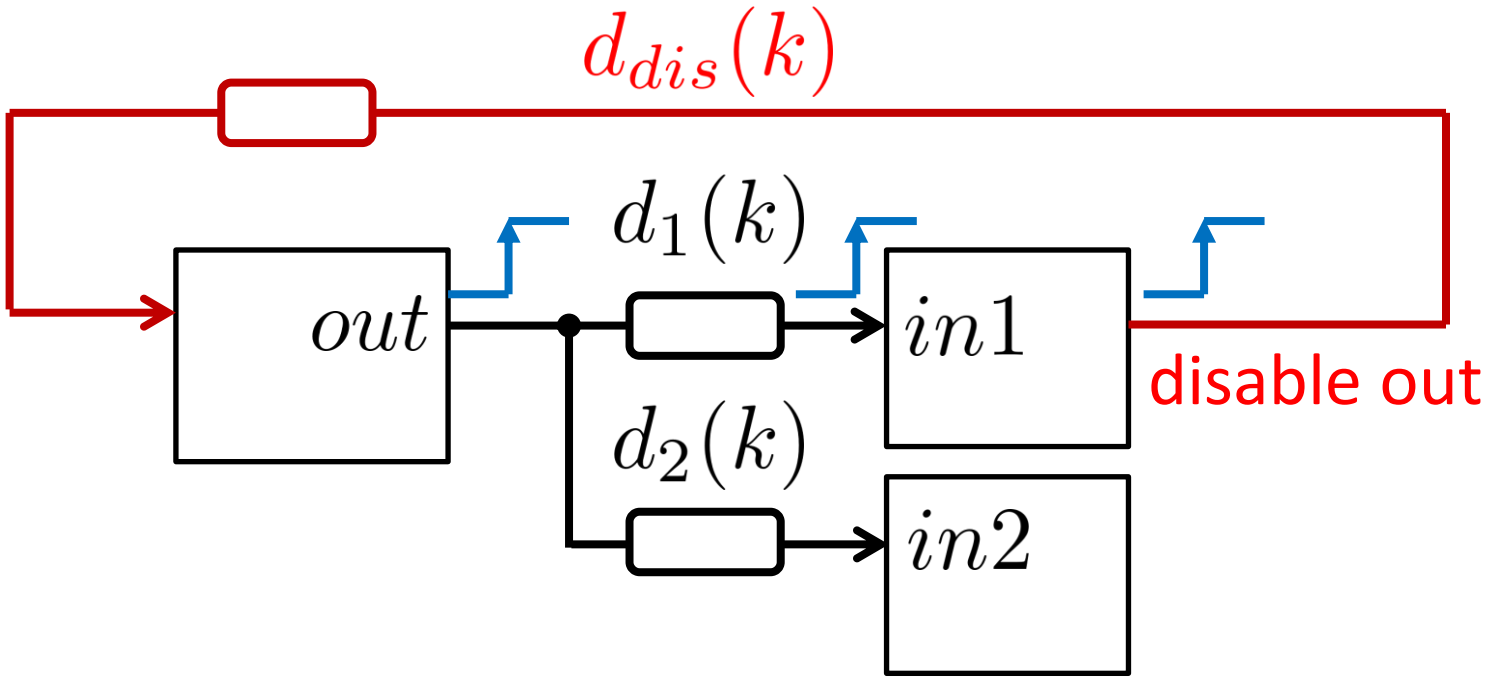
What we really want...



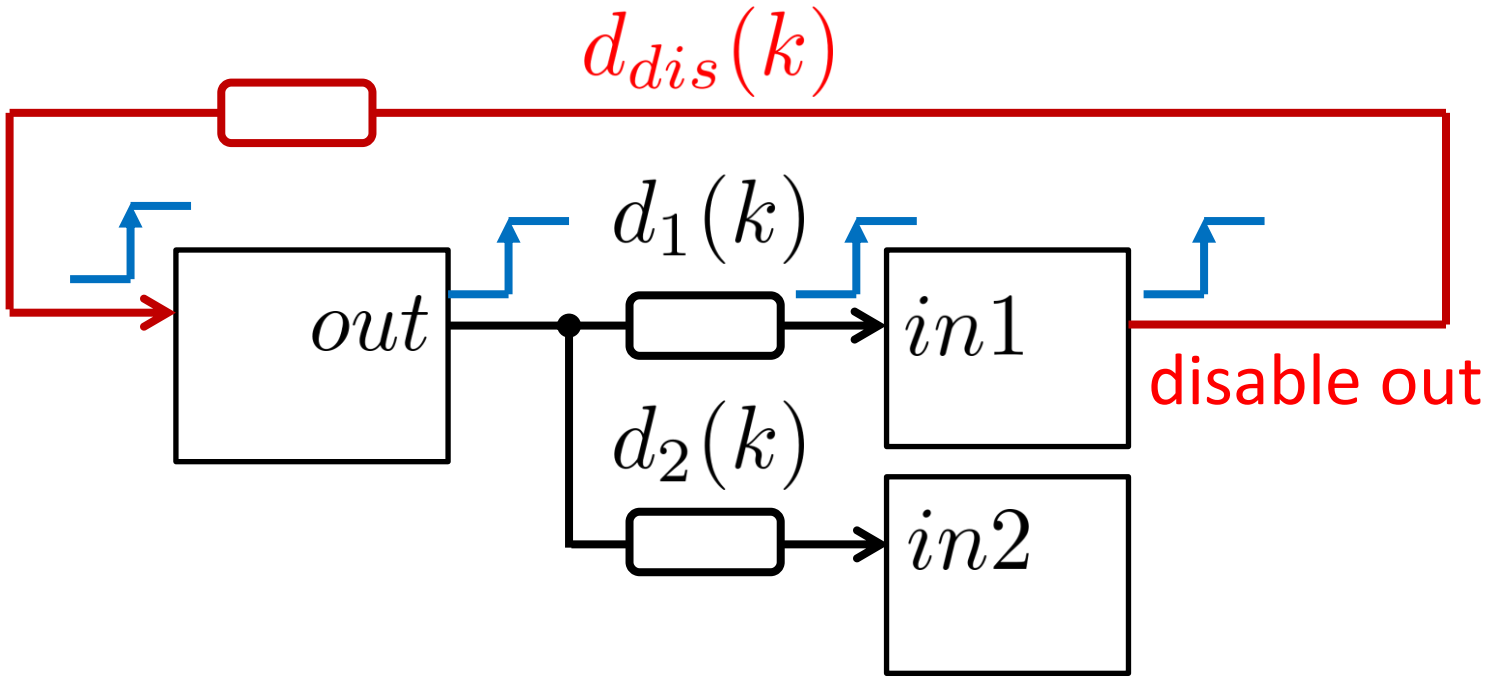
What we really want...



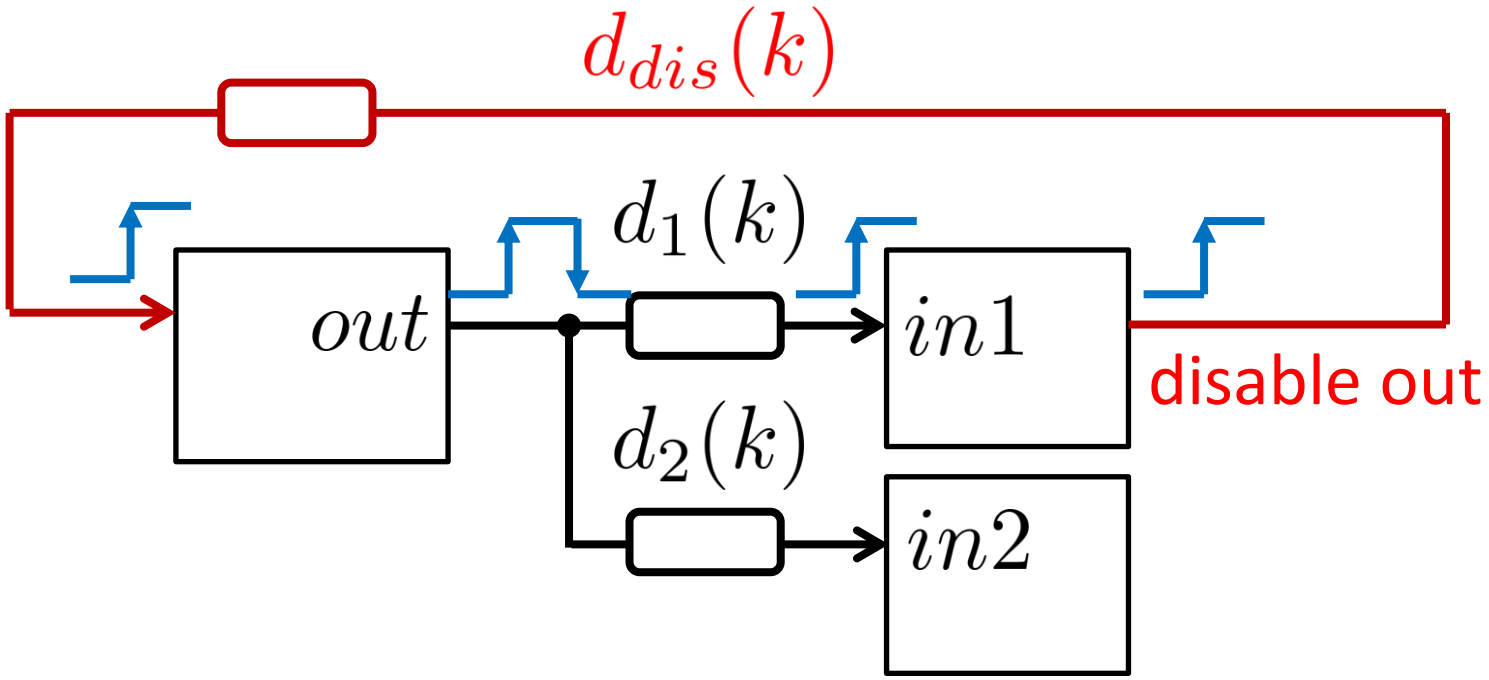
What we really want...



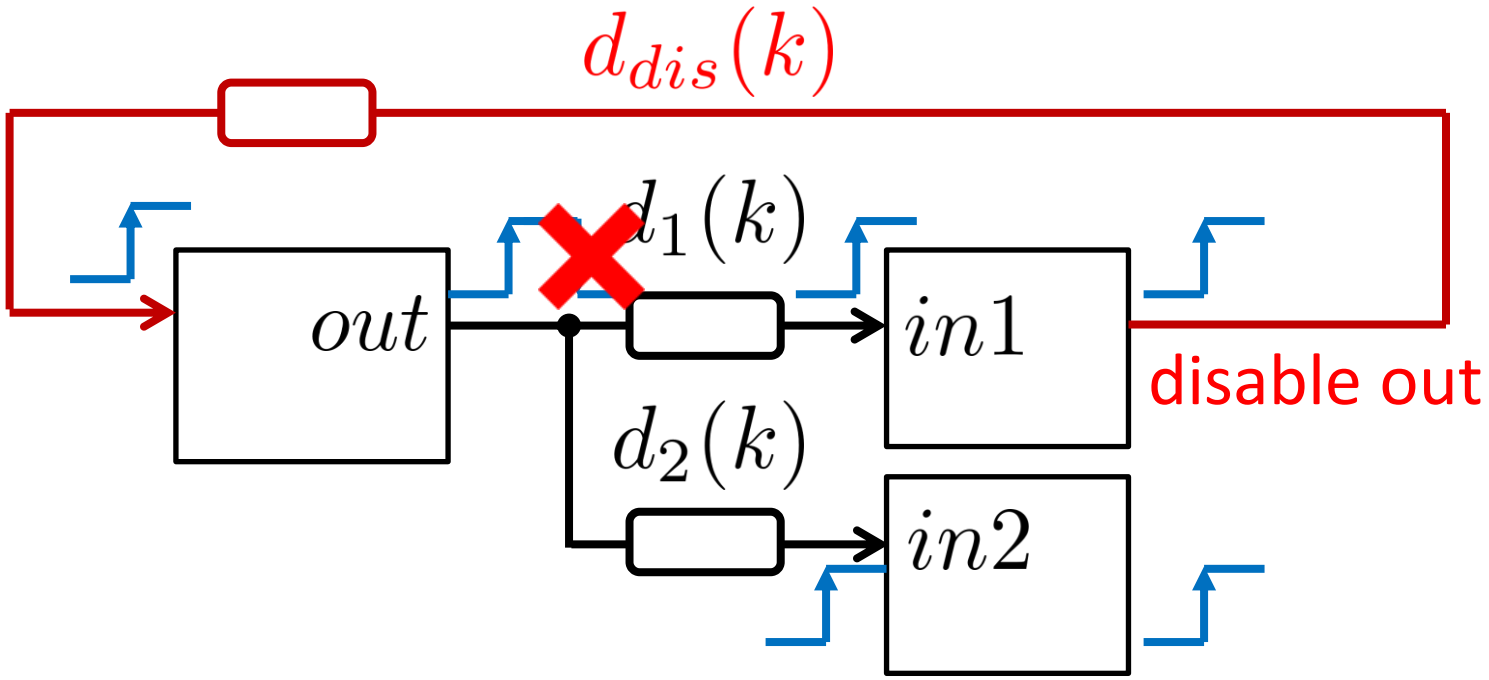
What we really want...



What we really want...

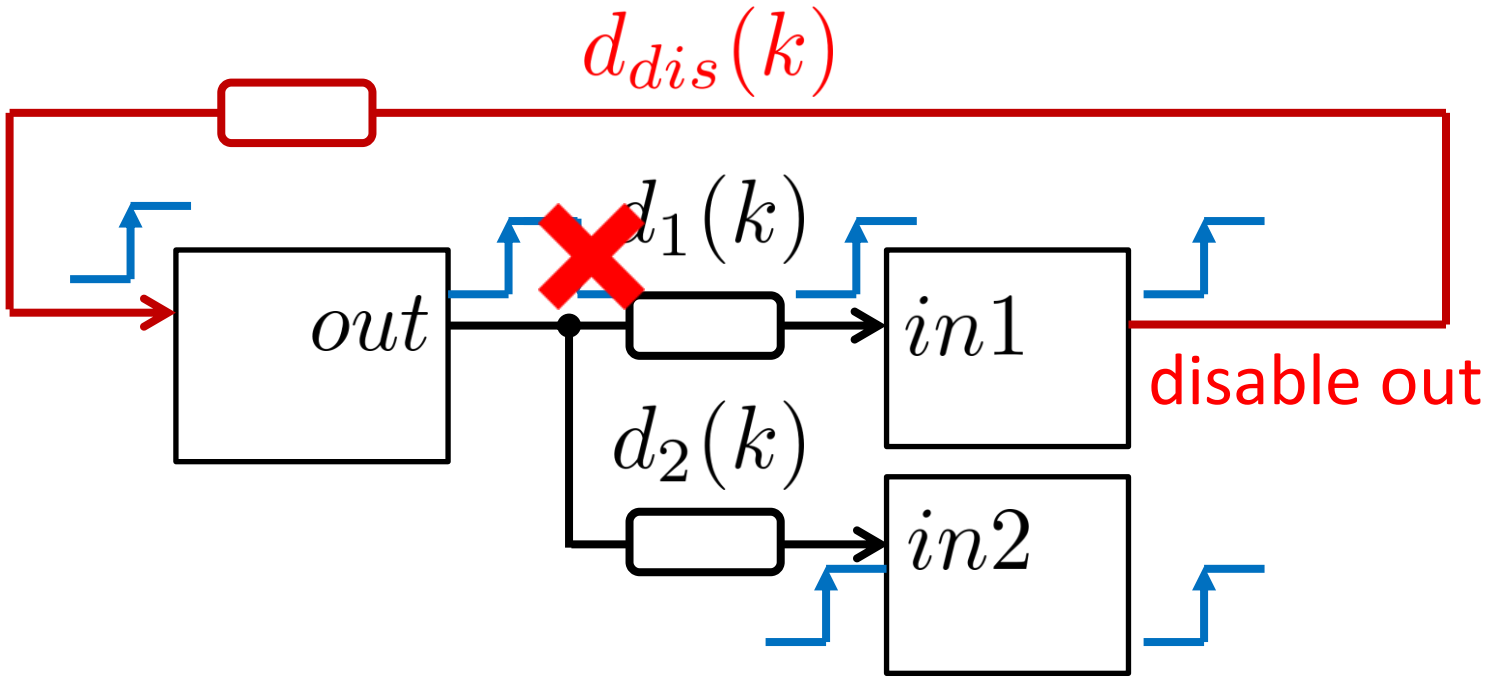


What we really want...



$$\forall k : d_2(k) \leq d_1(k) + d_{dis}(k)$$

What we really want...



adversarial path condition

-> one-sided constraint

Circuit types

- Synchronous
- Clockless
  - aggressively timed (beyond isochronic condition)
  - …
  - Speed independent (SI: all isochronic)
  - Quasi-delay insensitive (QDI: some isochronic)
  - Delay insensitive (DI)

Examples

… soon from synthesis.

Beyond classical circuit design lecture 5.75

Synthesis

Synthesis

In: CHP

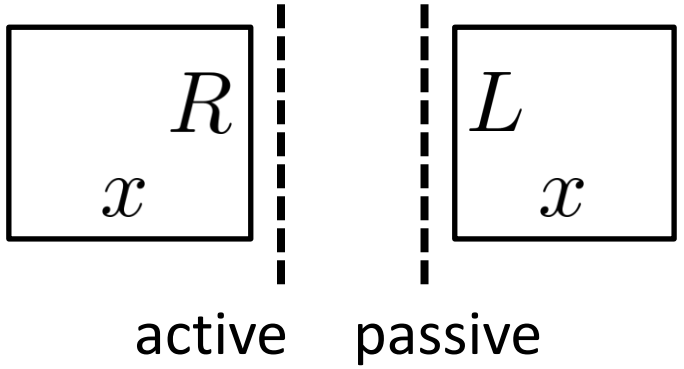
Out: Circuit (= Production rules + constraints)

Example: 1-bit channel

Send $x \in \{0, 1\}$

active send,

passive receive



CHP:

sender: $Ps : *[\ldots R!x \ldots]$

receiver: $Pr : *[\ldots L?x \ldots]$

Example: 1-bit channel

Choice: dual-rail encoding.

$$R!x \quad rd := x; [v(ra)]; rd := neutral; [n(ra)]$$

$$[x \to rd.1 \uparrow || \bar{x} \to rd.0 \uparrow]; [ra];$$
$$(rd.1 \downarrow || rd.0 \downarrow); [r\bar{a}]$$

Example: 1-bit channel

$$L?x \qquad [v(ld)]; y := ld; la\uparrow; [n(ld)]; la\downarrow$$

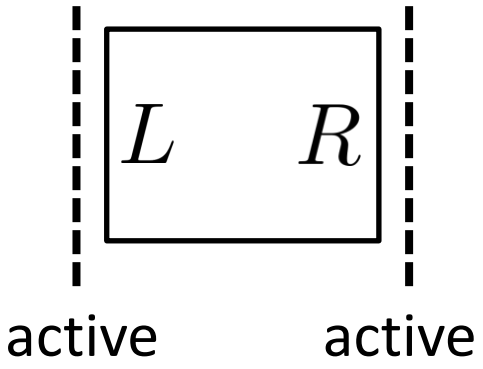
$$[ld.1 \vee ld.0]; [ld.1 \rightarrow x\uparrow \,||\, ld.0 \rightarrow x\downarrow]; la\uparrow;$$
$$[\neg(ld.1 \vee ld.0)]; la\downarrow$$

$$[ld.1 \rightarrow x\uparrow \,||\, ld.0 \rightarrow x\downarrow]; la\uparrow;$$
$$[\neg(ld.1 \vee ld.0)]; la\downarrow$$

Example: active-active sequencer

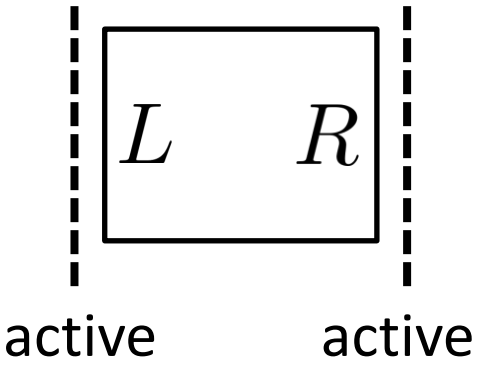
two active interfaces L, R.

CHP: $P1 : *[L; R]$



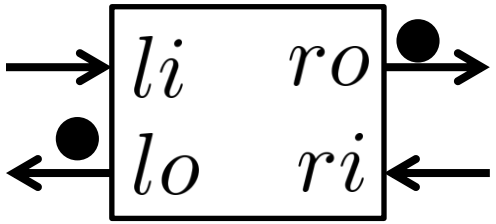
Example: active-active sequencer

two active interfaces L, R.



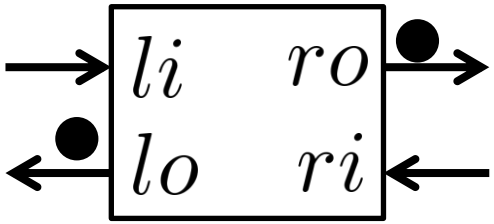
CHP: $P1 : *[L; R]$

Choice: 4-ph hs



$$*[lo\uparrow; [li]; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; ro\downarrow; [\bar{ri}]]$$

For lo…



CHP: $*[lo\uparrow; [li]; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; ro\downarrow; [\bar{ri}]]$

$lo = 0$

by circuit

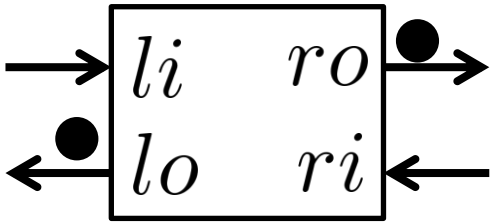
$ro = 0$

$ri = 0$

by environment

$li = 0$

For lo...



CHP: $*[lo\uparrow; [li]; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; ro\downarrow; [\bar{ri}]]$

$lo = 0$

$ro = 0$

by circuit

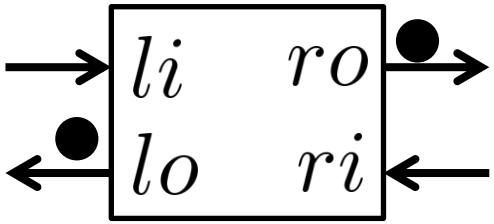
$ri = 0$

$li = 0$

by environment

PR: $\bar{lo} \wedge r\bar{o} \wedge \bar{ri} \wedge \bar{li} \rightarrow lo\uparrow$

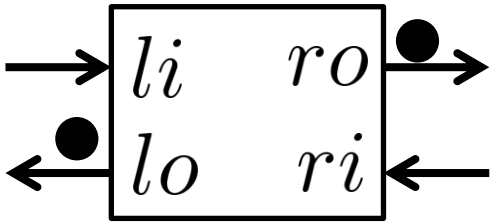
For lo…



CHP: $*[lo\uparrow; [li]; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; ro\downarrow; [\bar{ri}]]$

$$\frac{lo = 0 \quad ro = 0}{ri = 0 \quad li = 0} \qquad \frac{lo = 0 \quad ro = 0}{ri = 0 \quad li = 0}$$

PR: $\bar{lo} \wedge \bar{ro} \wedge \bar{ri} \wedge \bar{li} \rightarrow lo\uparrow$ …. does not work!



CHP: $*[lo\uparrow; [li]; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; ro\downarrow; [\bar{ri}]]$

$x\uparrow$ $x\downarrow$

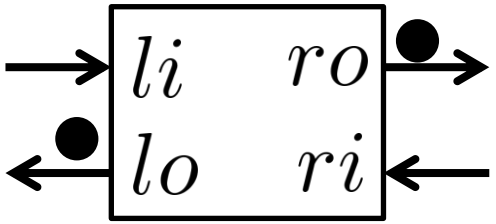
$lo = 0$ $lo = 0$

$ro = 0$ $ro = 0$

$x = 0$ $x = 1$

$ri = 0$ $ri = 0$

$li = 0$ $li = 0$



CHP: $*[lo\uparrow; [li]; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; ro\downarrow; [\bar{ri}]]$

$x\uparrow$ $x\downarrow$

$lo = 0$ $lo = 0$

$ro = 0$ $ro = 0$

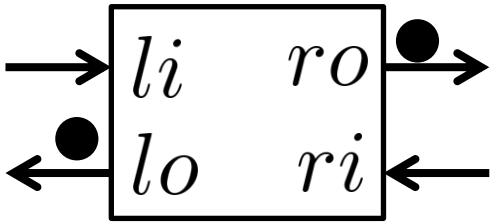
$x = 0$ $x = 1$

$ri = 0$ $ri = 0$

$li = 0$ $li = 0$

PR: $\bar{x} \wedge \bar{lo} \wedge \bar{ro} \wedge \bar{ri} \wedge \bar{li} \rightarrow lo\uparrow$

For lo…

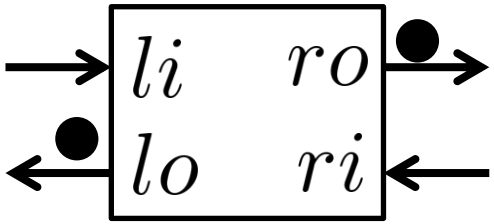


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

PR: $\bar{x} \wedge \bar{lo} \wedge \bar{ro} \wedge \bar{ri} \wedge \bar{li} \rightarrow lo\uparrow$

$\bar{x} \wedge \bar{ri} \rightarrow lo\uparrow$

For lo…



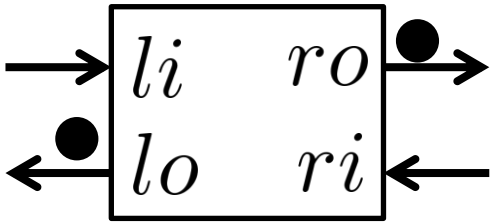
CHP: $*[lo\uparrow; [li]; x\uparrow, lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

PR: $\bar{x} \wedge \bar{lo} \wedge \bar{ro} \wedge \bar{ri} \wedge \bar{li} \rightarrow lo\uparrow$

$\bar{x} \wedge \bar{ri} \rightarrow lo\uparrow$

$x \vee ri \rightarrow lo\downarrow$

For lo…

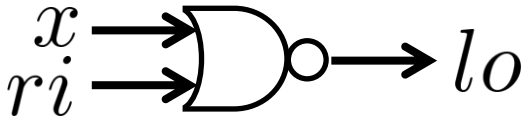


CHP: $*[lo\uparrow; [li]; x\uparrow, lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

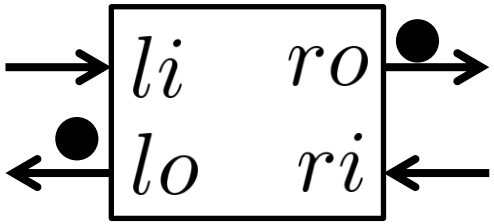
PR: $\bar{x} \wedge \bar{lo} \wedge \bar{ro} \wedge \bar{ri} \wedge \bar{li} \rightarrow lo\uparrow$

$\bar{x} \wedge \bar{ri} \rightarrow lo\uparrow$

$x \vee ri \rightarrow lo\downarrow$



For ro...

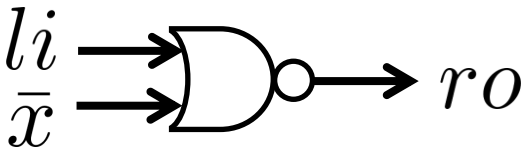


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

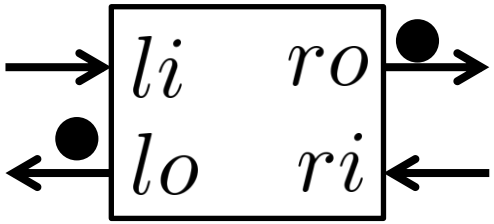
PR:

$$x \wedge \bar{li} \rightarrow ro\uparrow$$

$$\bar{x} \wedge li \rightarrow ro\downarrow$$

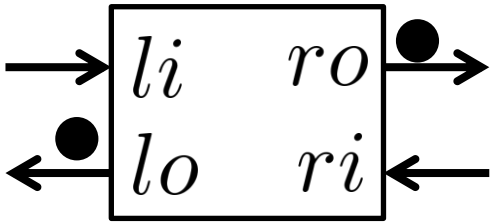


For x...



CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

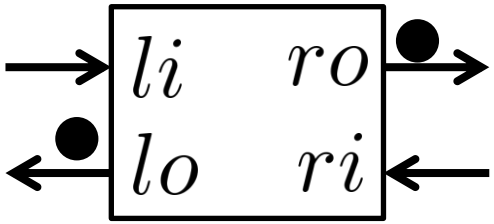
For x…



CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

… What gate is this?

For x...

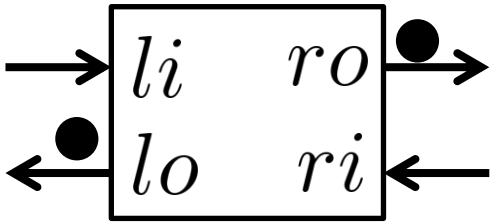


CHP: $*[lo\uparrow;[li];x\uparrow;lo\downarrow;[\bar{li}];ro\uparrow;[ri];x\downarrow;ro\downarrow;[\bar{ri}]]$

PR: $\bar{ri} \wedge li \rightarrow x\uparrow$

$ri \wedge \bar{li} \rightarrow x\downarrow$

For x...

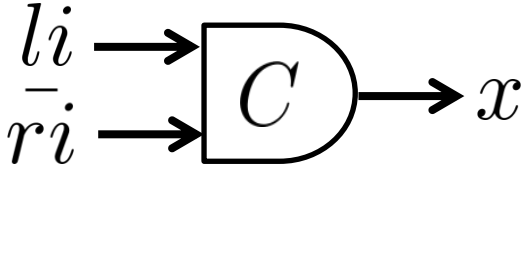


CHP: $*[lo\uparrow;[li];x\uparrow;lo\downarrow;[\bar{li}];ro\uparrow;[ri];x\downarrow;ro\downarrow;[\bar{ri}]]$

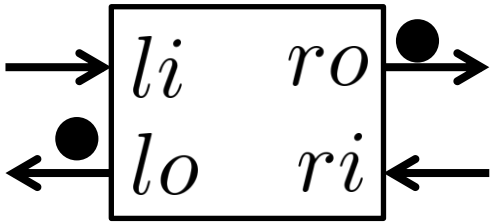
PR:

$$\bar{ri} \wedge li \rightarrow x\uparrow$$

$$ri \wedge \bar{li} \rightarrow x\downarrow$$

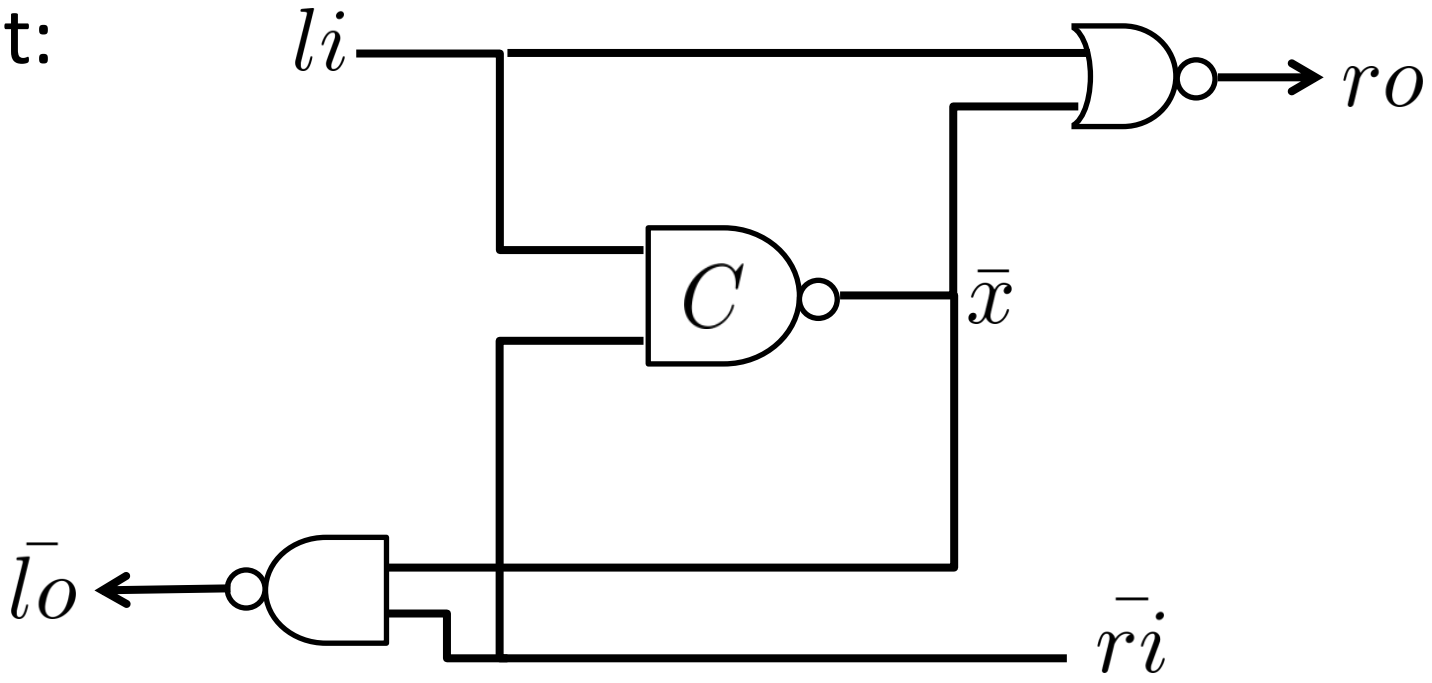


Putting it together

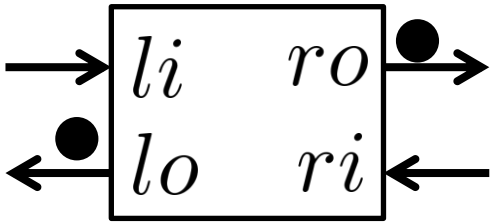


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

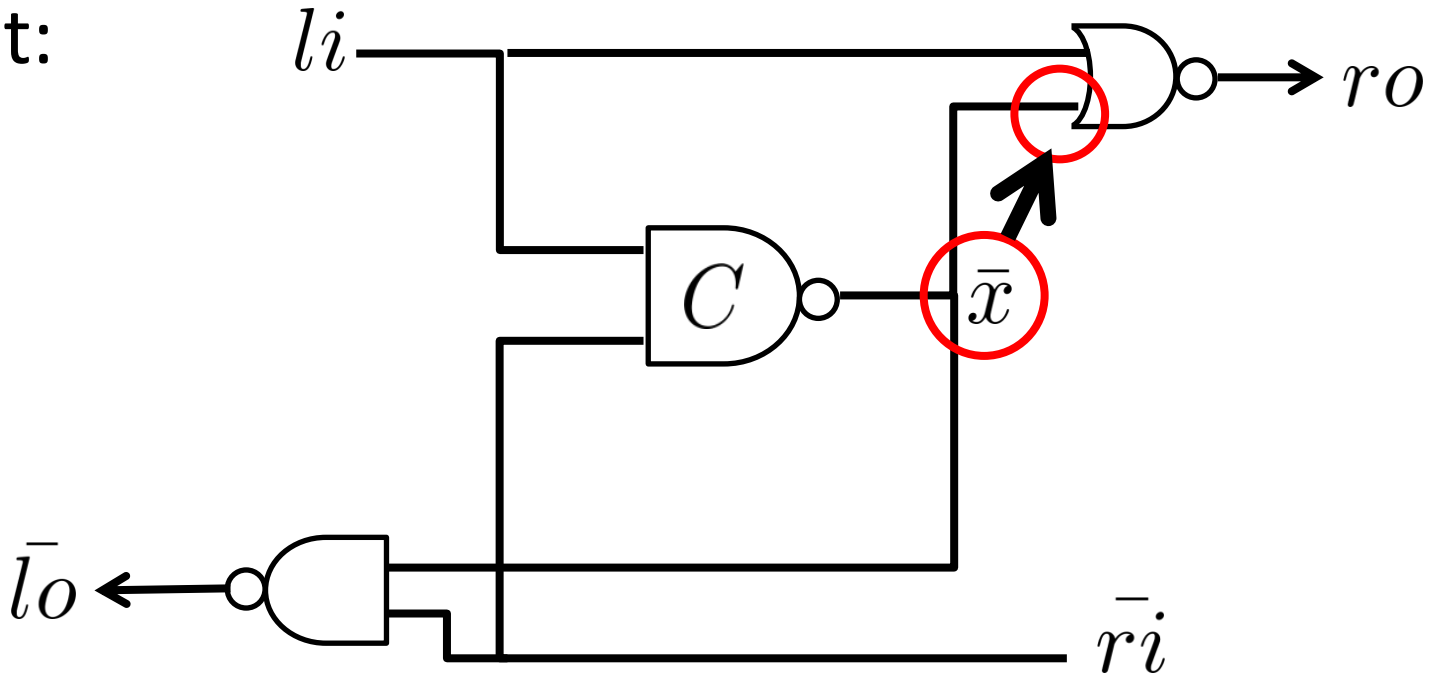


Mind wires...

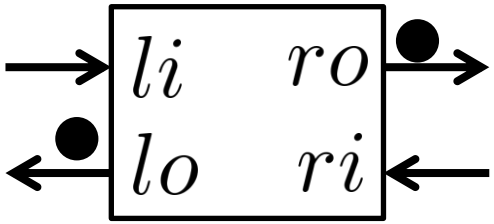


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

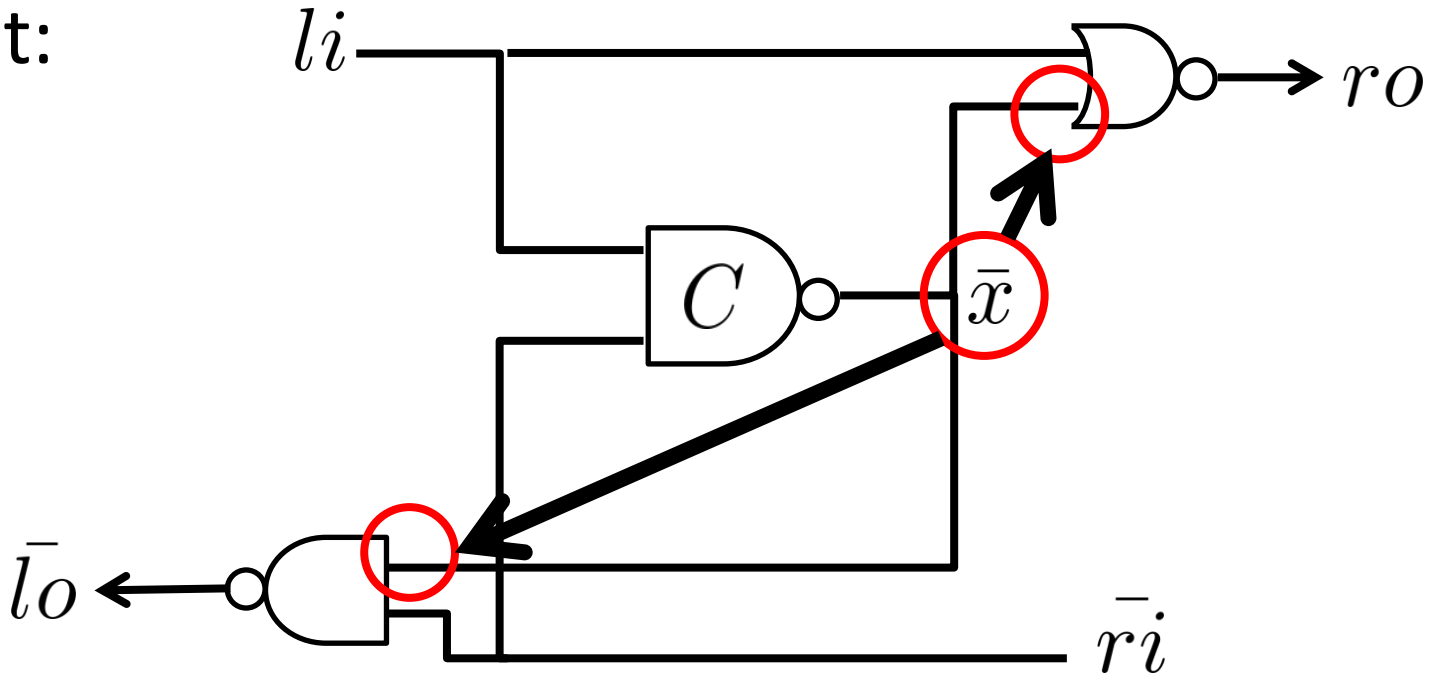


Mind forks...

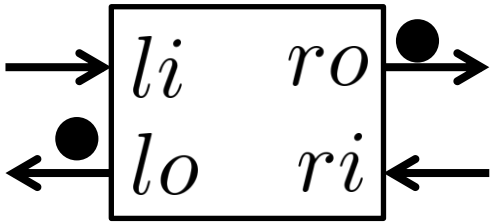


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

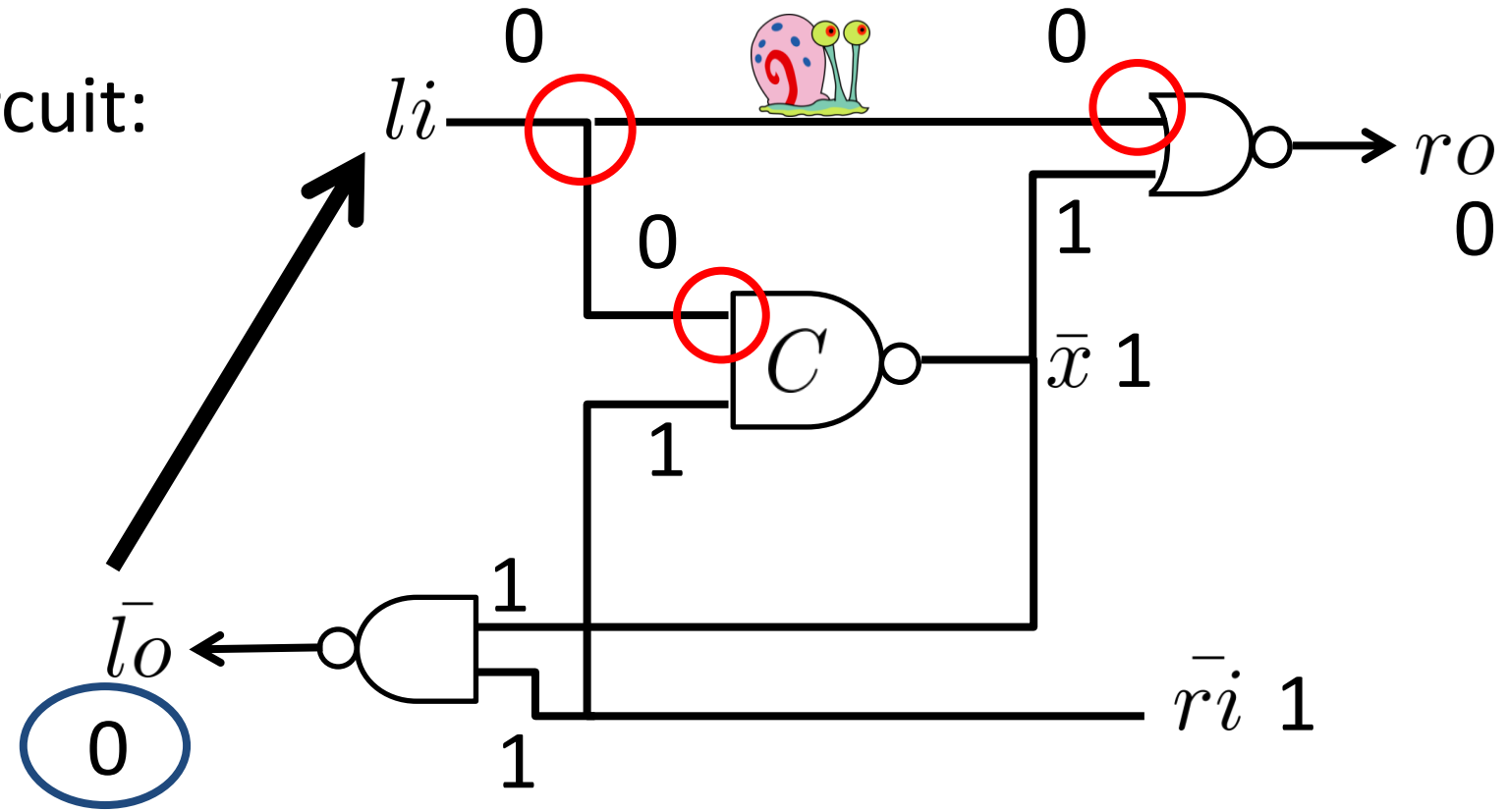


Checking fork "li" (1)

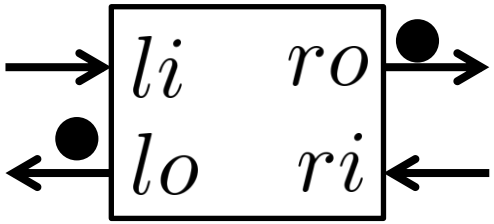


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

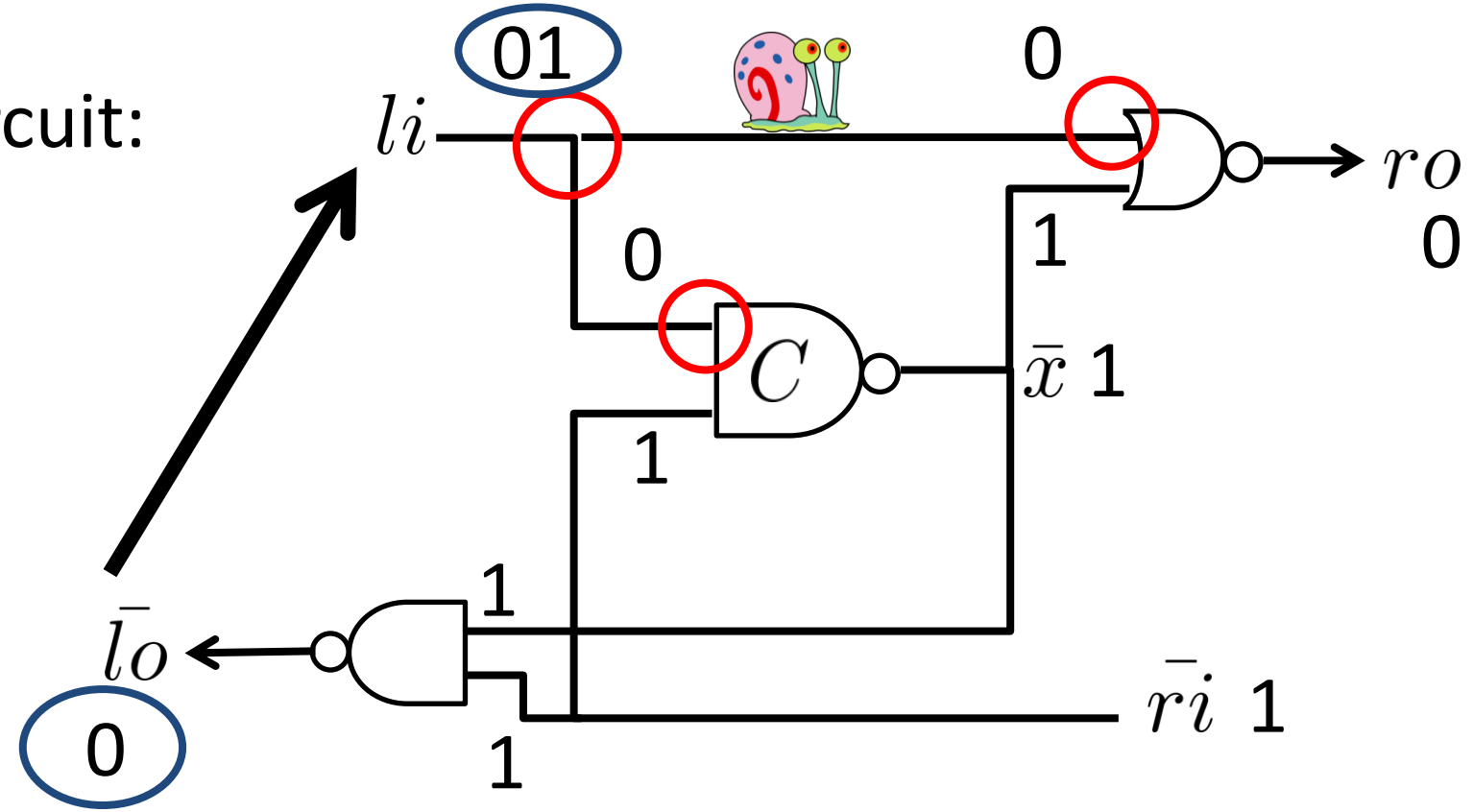


Checking fork "li" (1)

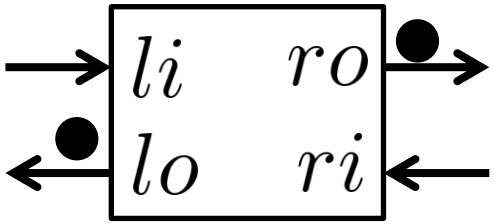


CHP: $*[lo\uparrow [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

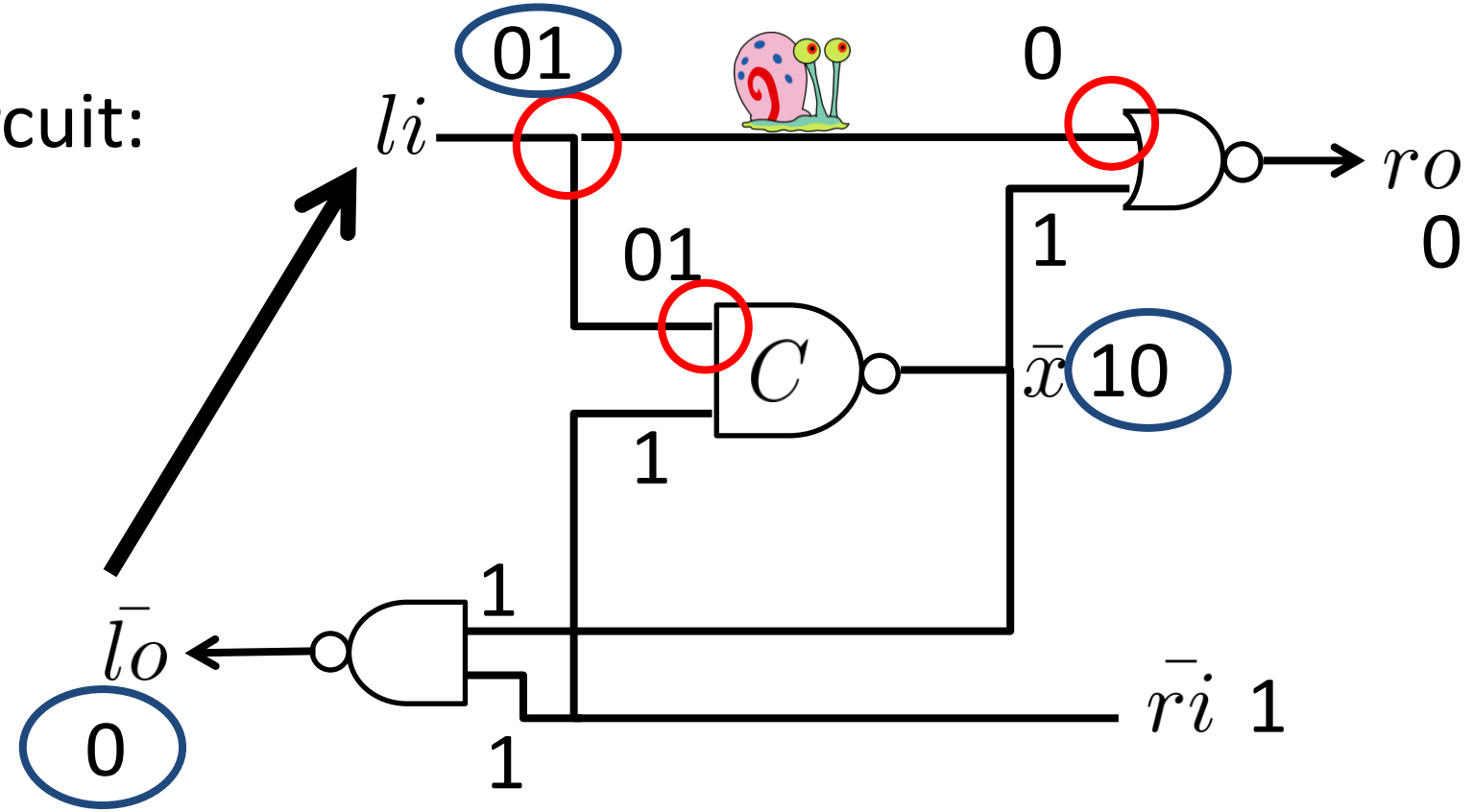


Checking fork "li" (1)

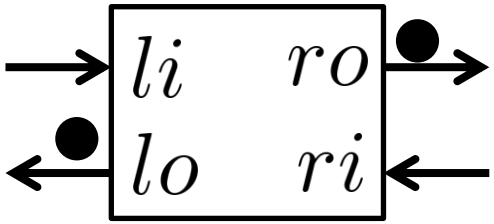


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

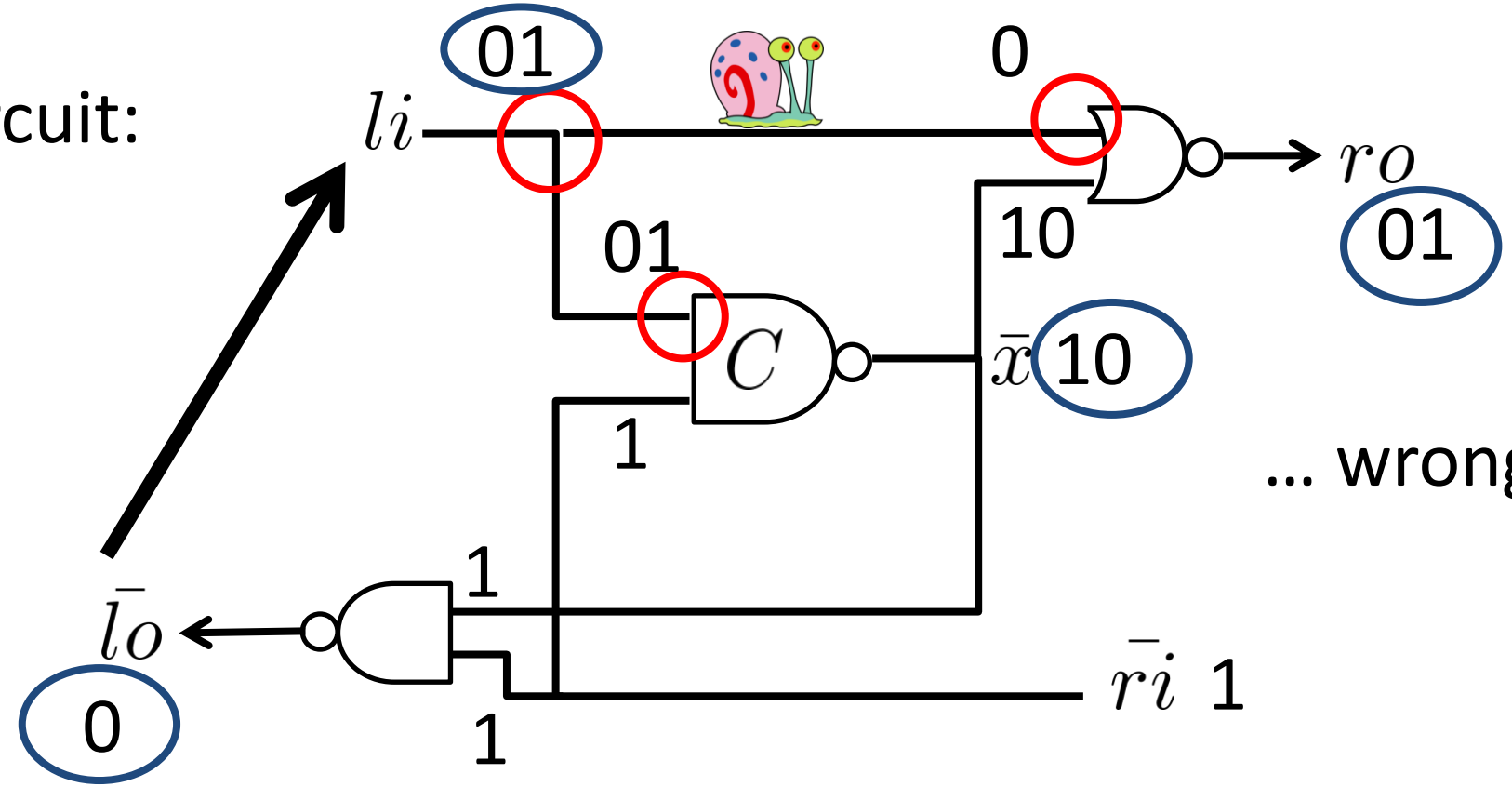


Checking fork "li" (1)

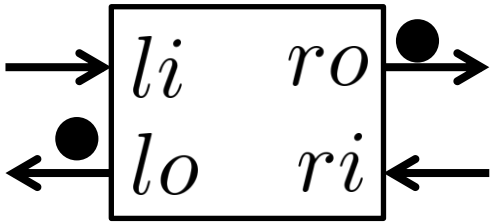


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:

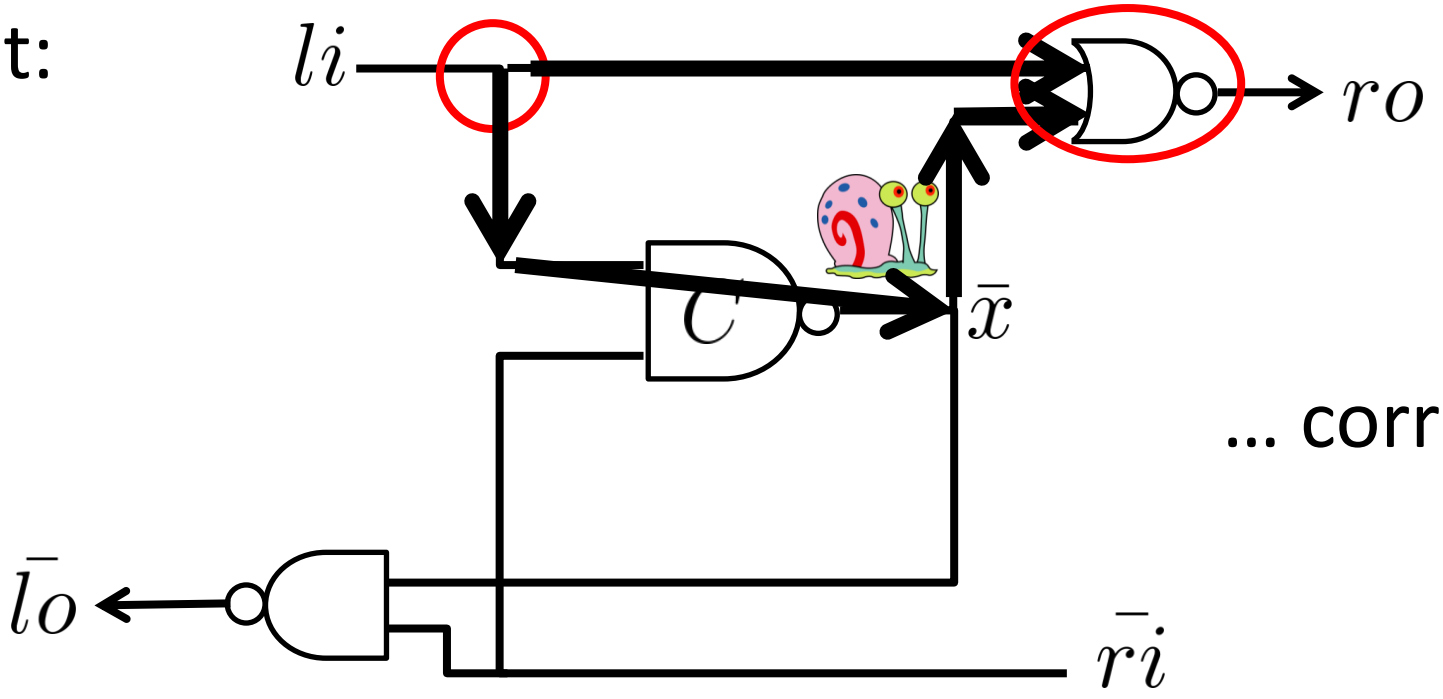


+ one-sided Constraint



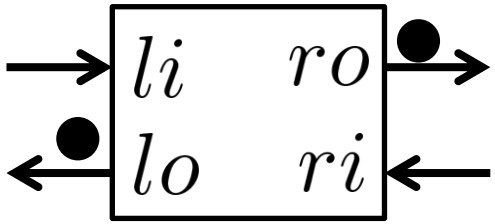
CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Circuit:



… correct.

+ one-sided Constraint



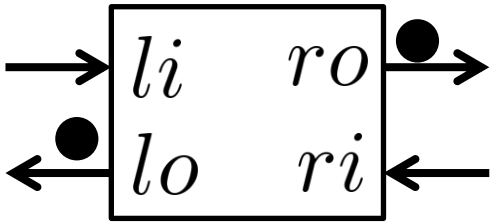
CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

Proving correct:

e.g. by induction.

base case: start with initial states and prove ordering of events from there for the first loop

+ one-sided Constraint



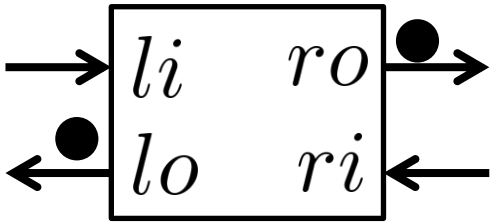
CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

proof:

examine ordering in time "<".

1) **lo-up < [li]:** guaranteed by environment: Initially li = 0. Can be set to li = 1 only by environment. Environment does this only after lo = 1.

+ one-sided Constraint

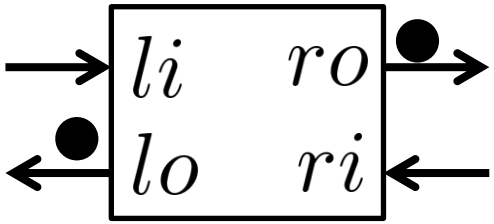


CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

proof:

2) **[li] < x-up:** x = 1 can happen only after both C-Element inputs are 1. This can happen only after li = 1 for the first time.

+ one-sided Constraint



CHP: $*[lo\uparrow; [li]; x\uparrow; lo\downarrow; [\bar{li}]; ro\uparrow; [ri]; x\downarrow; ro\downarrow; [\bar{ri}]]$

proof:

3) **x-up < lo-down:** this can happen only if one of the NAND inputs becomes 0 for the first time.

We first show that lo-down cannot happen because of the input connected to not ri becoming 0: .....

[hw]