Adapting Asynchronous Circuits to Operating Conditions by Logic Parameterisation

Andrey Mokhov, Danil Sokolov, Alex Yakovlev

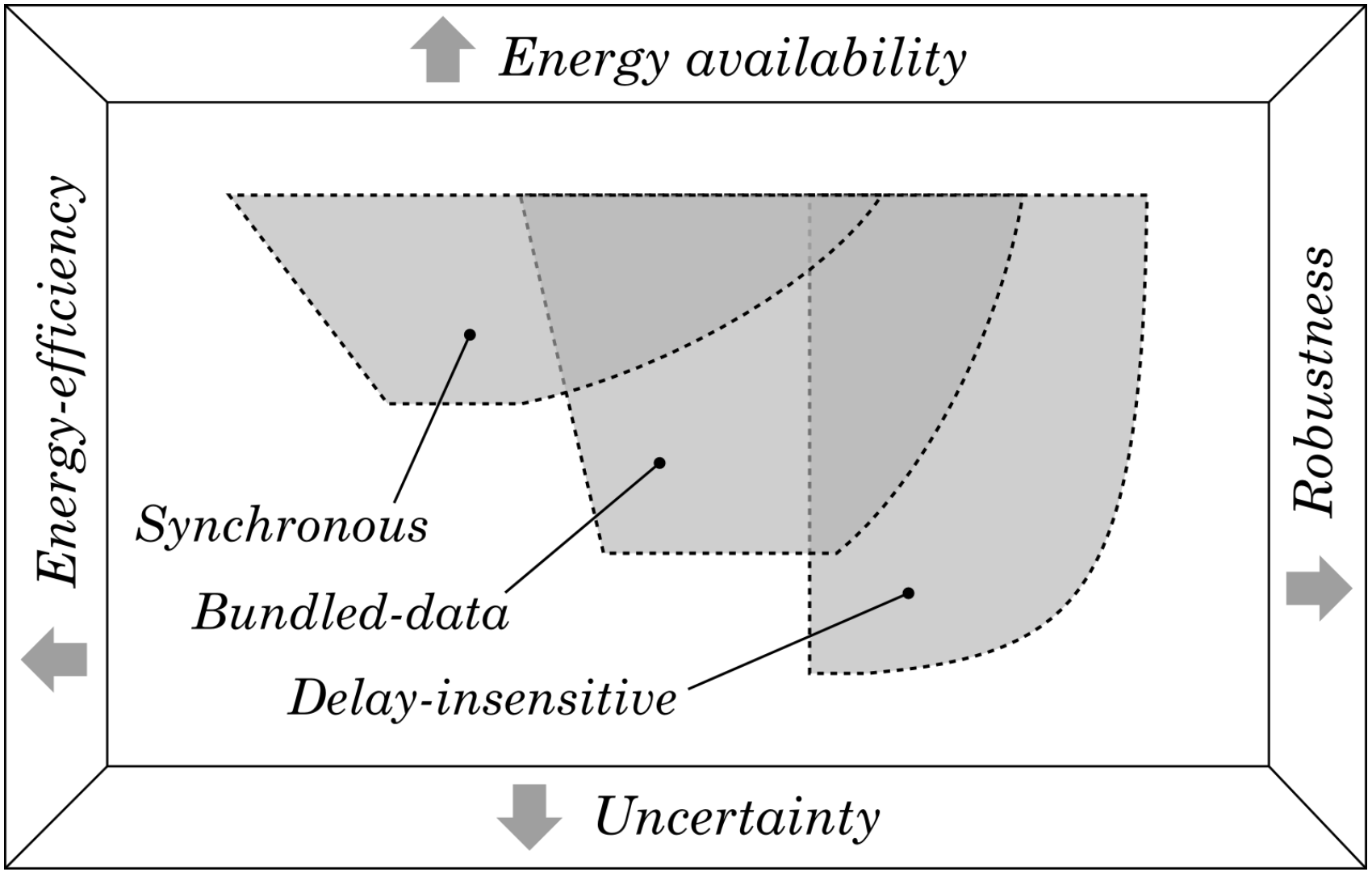
Newcastle University, UK

Designer's dilemma

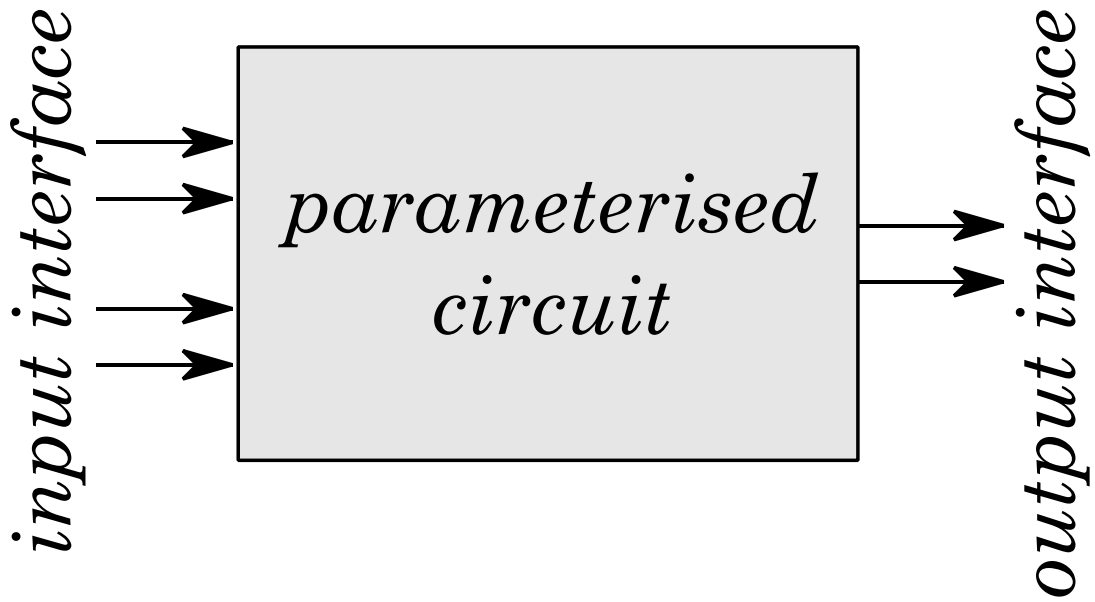
- Contradicting requirements:
  - ↑ performance, ↓ energy consumption, ↑ robustness
  - high performance → high energy consumption
  - low energy consumption → low robustness
  - high robustness → low performance

- Competing implementation styles:
  - Synchronous
  - Asynchronous
    - Delay Insensitive (DI)
    - Speed Independent (SI) or Quasi Delay Insensitive (QDI)
    - Bundled data, relative timing assumptions (TA)
    - …

Energy-efficiency v Robustness

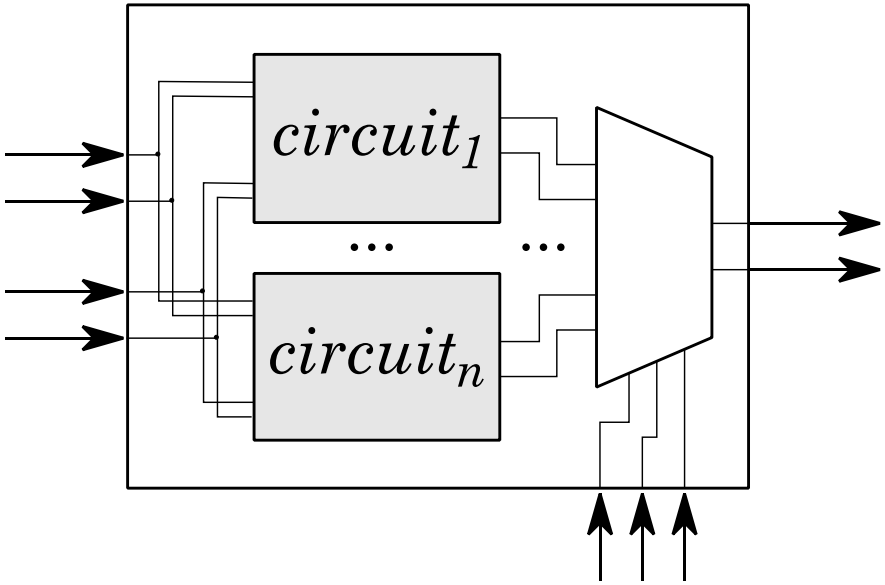


Parameterised Circuits



- Static parameters set at test/binning stages
- Dynamic parameters:
  - Power management controller
  - Maintenance mechanisms

Parameterised Circuits: Trivial Approach

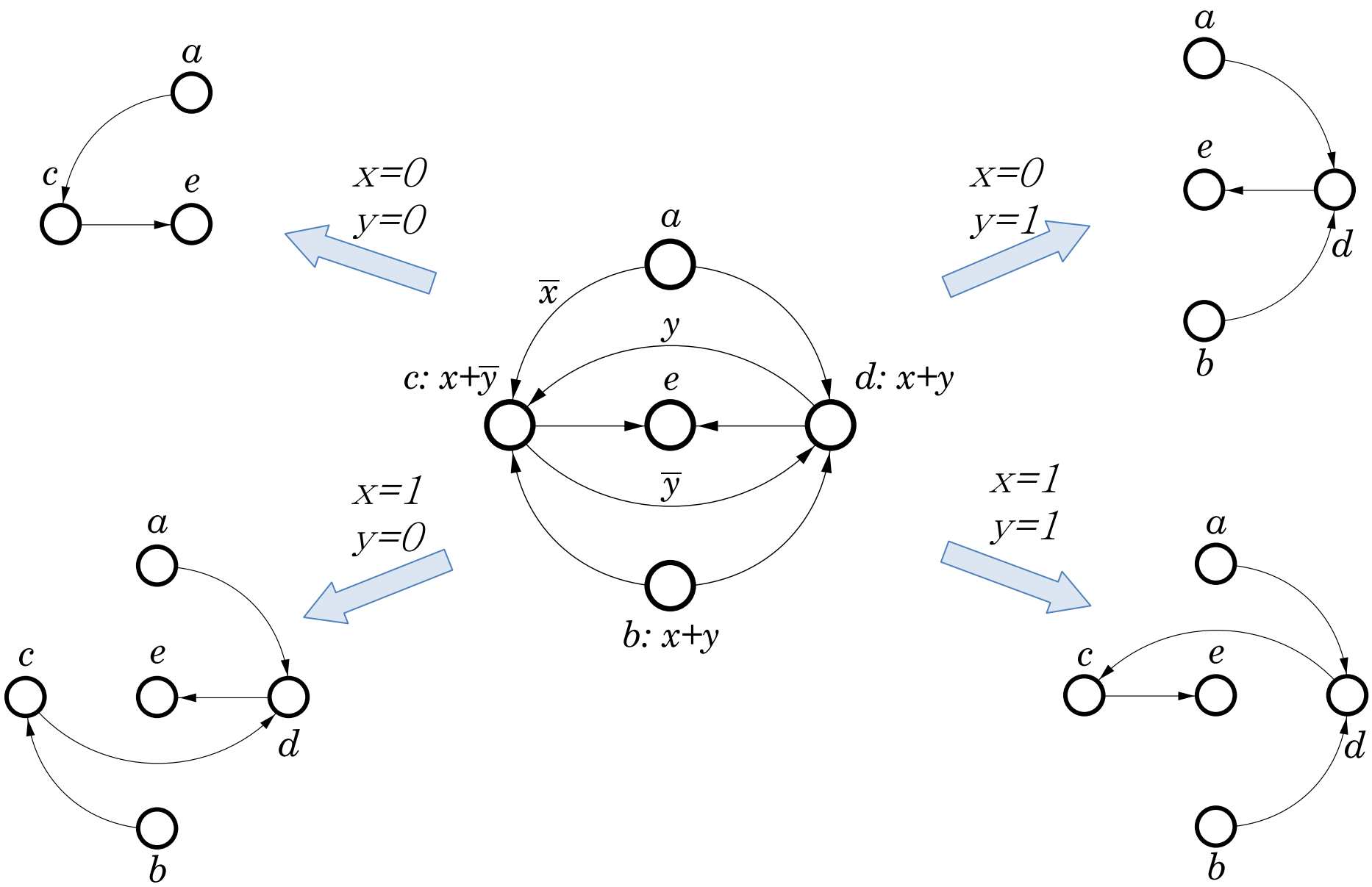


- Easy to design!
- Large overheads
- How can we do better?

Parameterised Circuits: Better Approach

- Goal:
  - Combine circuit implementations **efficiently**
- Key observations:
  - Externally: the circuits have the same interface
  - Internally: the circuits behave **similarly**
- Solution: use a model that can capture functional similarities at the circuit level
  - Conditional Partial Order Graphs **almost** fit

Conditional Partial Order Graphs



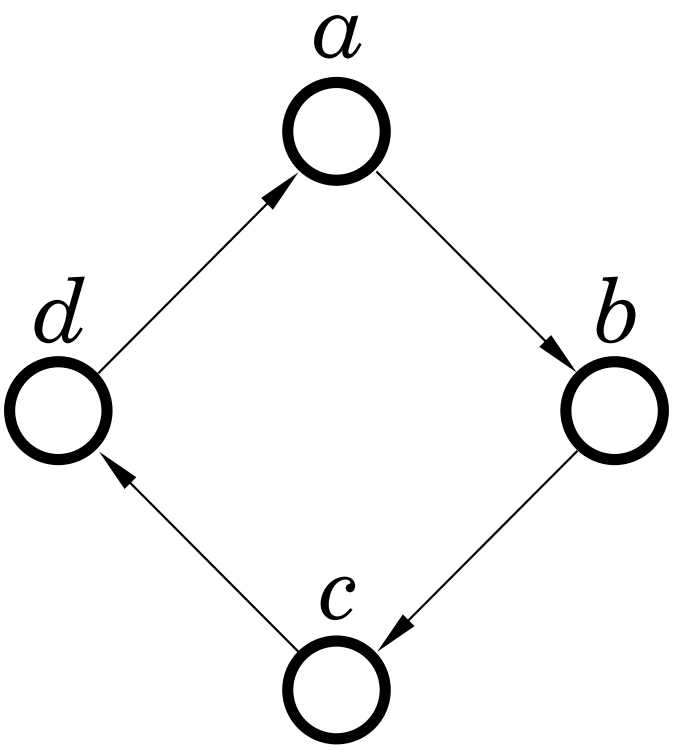
Conditional Partial Order Graphs

- Describe concurrency and causality
- Capture similarities in behaviours
- Represent families of **partial orders**
  - acyclic
  - not directly applicable to circuit specification

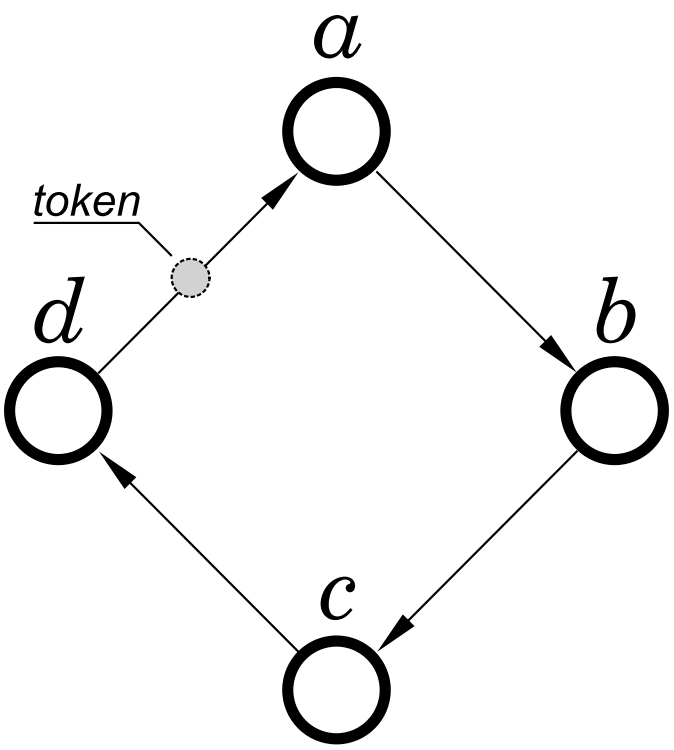
- Is it possible to specify cyclic behaviour using acyclic objects?

  –Yes!

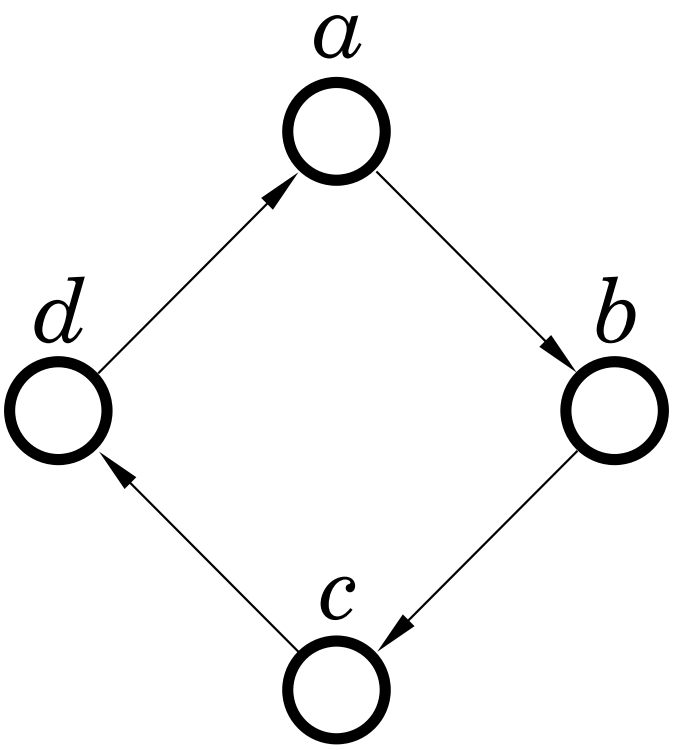
Describing cycles



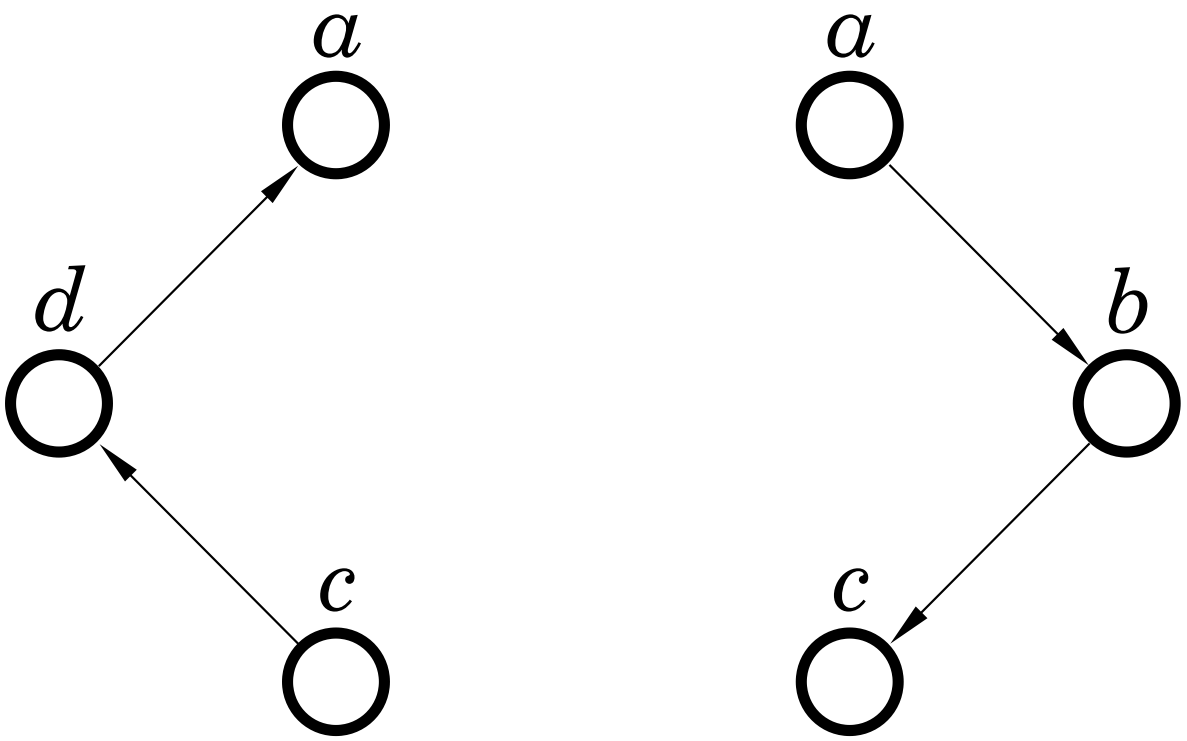
Describing cycles



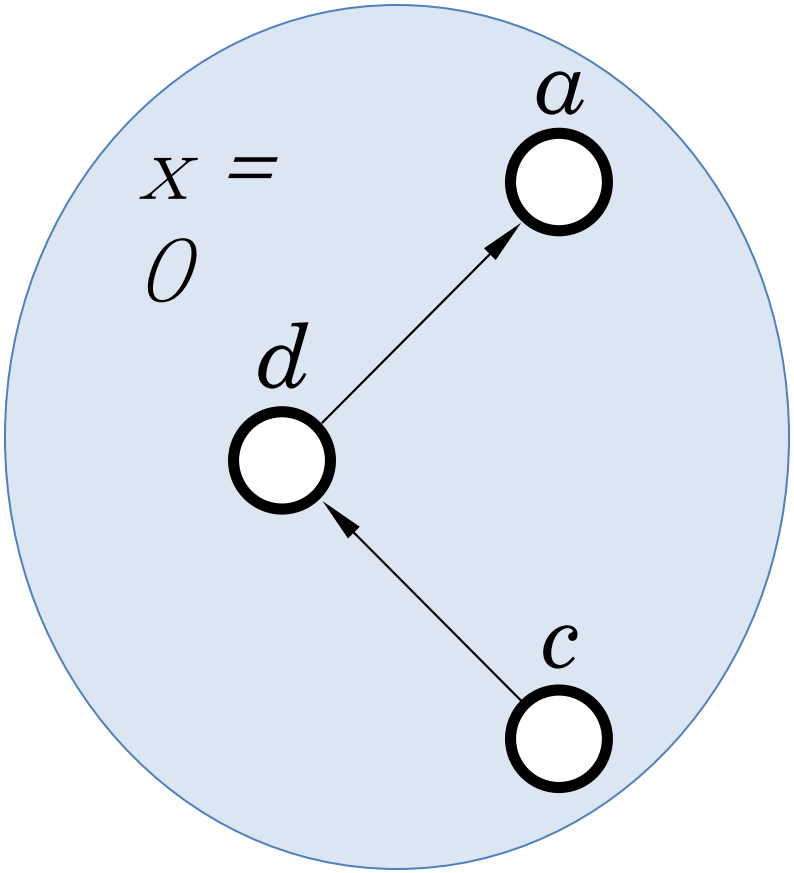
Describing cycles

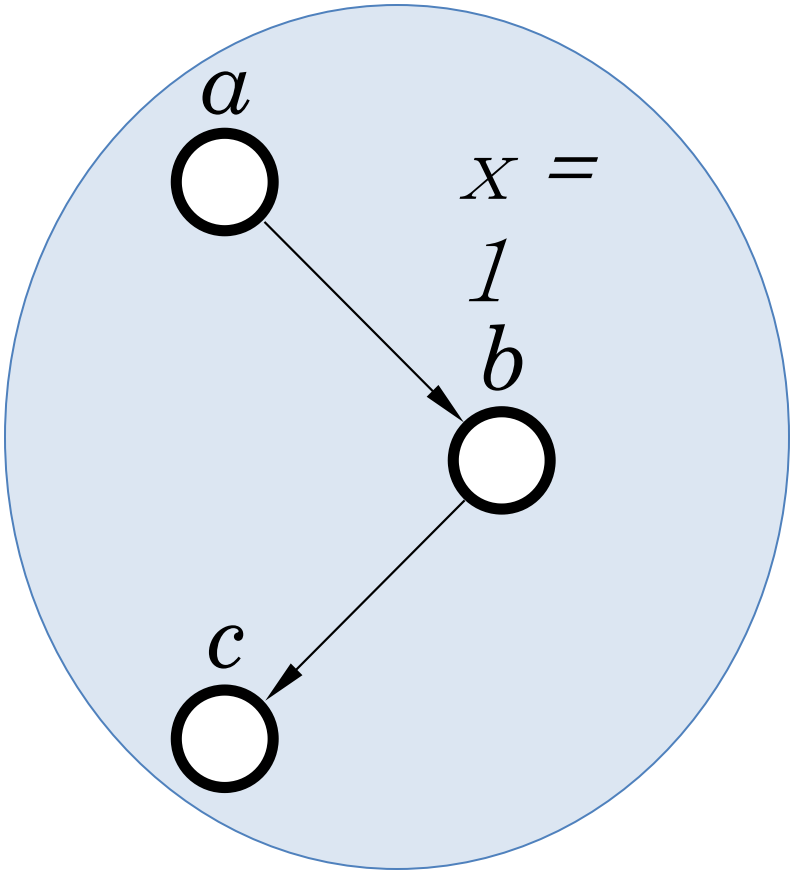


Describing cycles

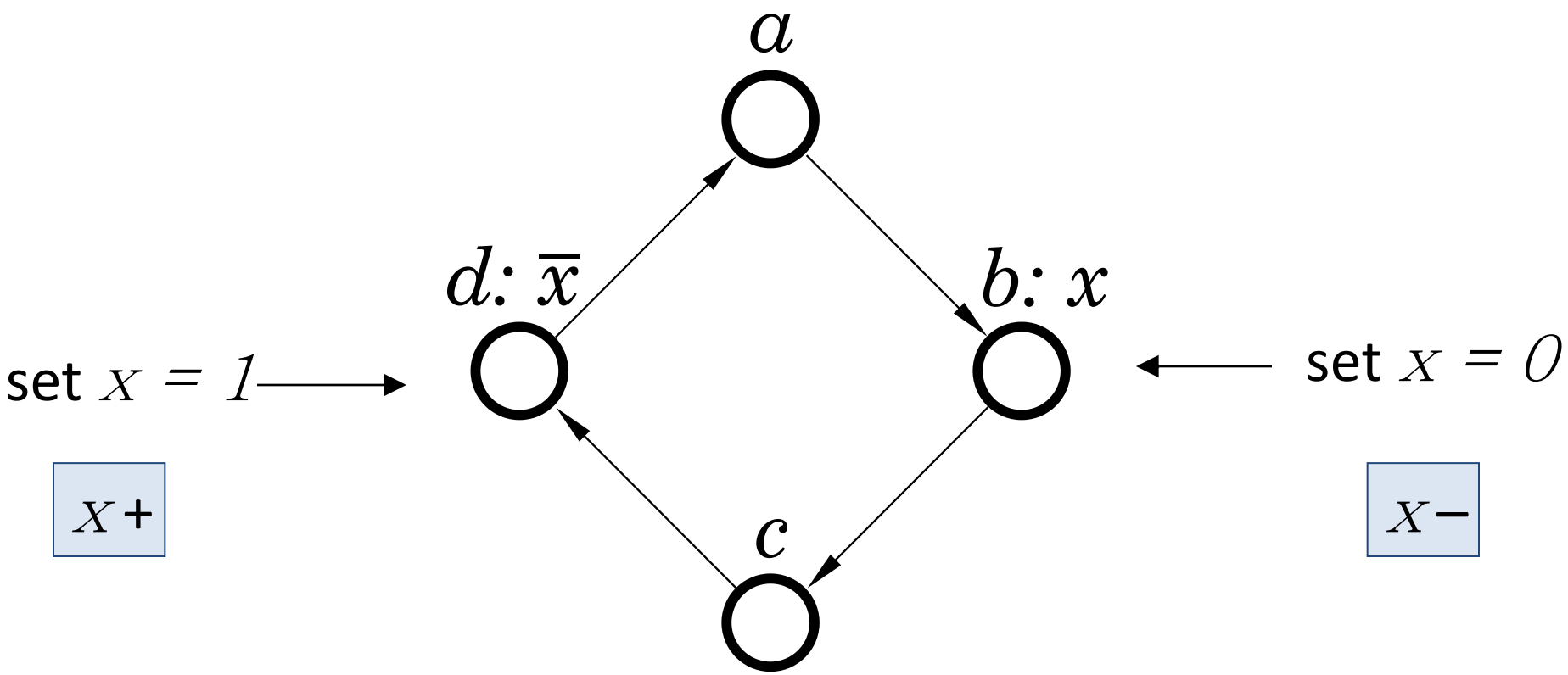


Describing cycles





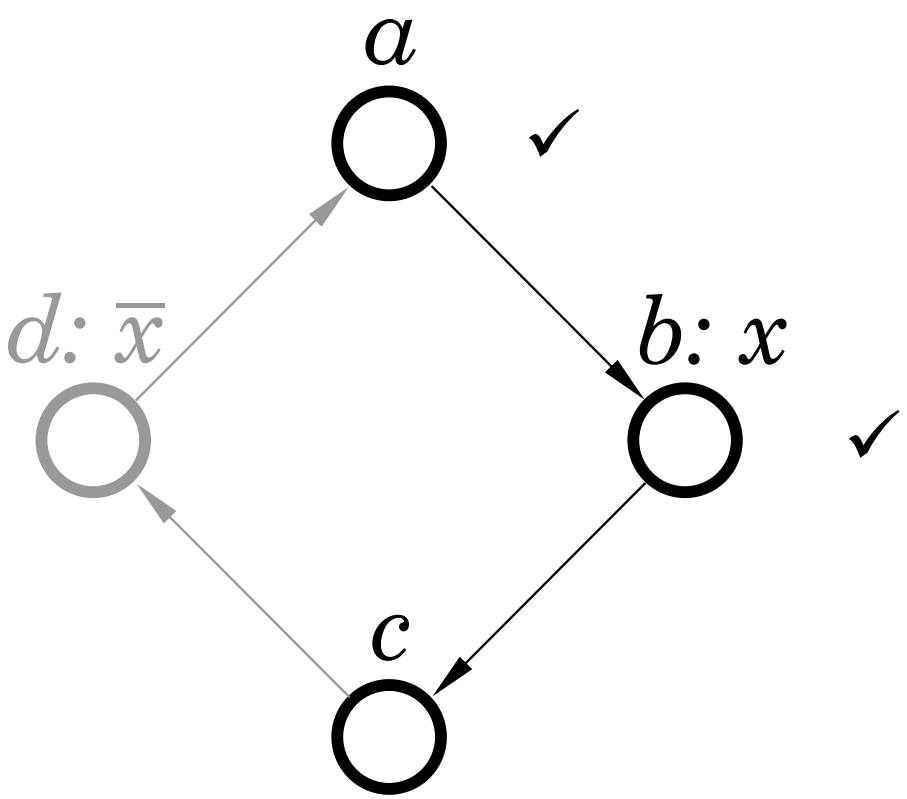
Describing cycles



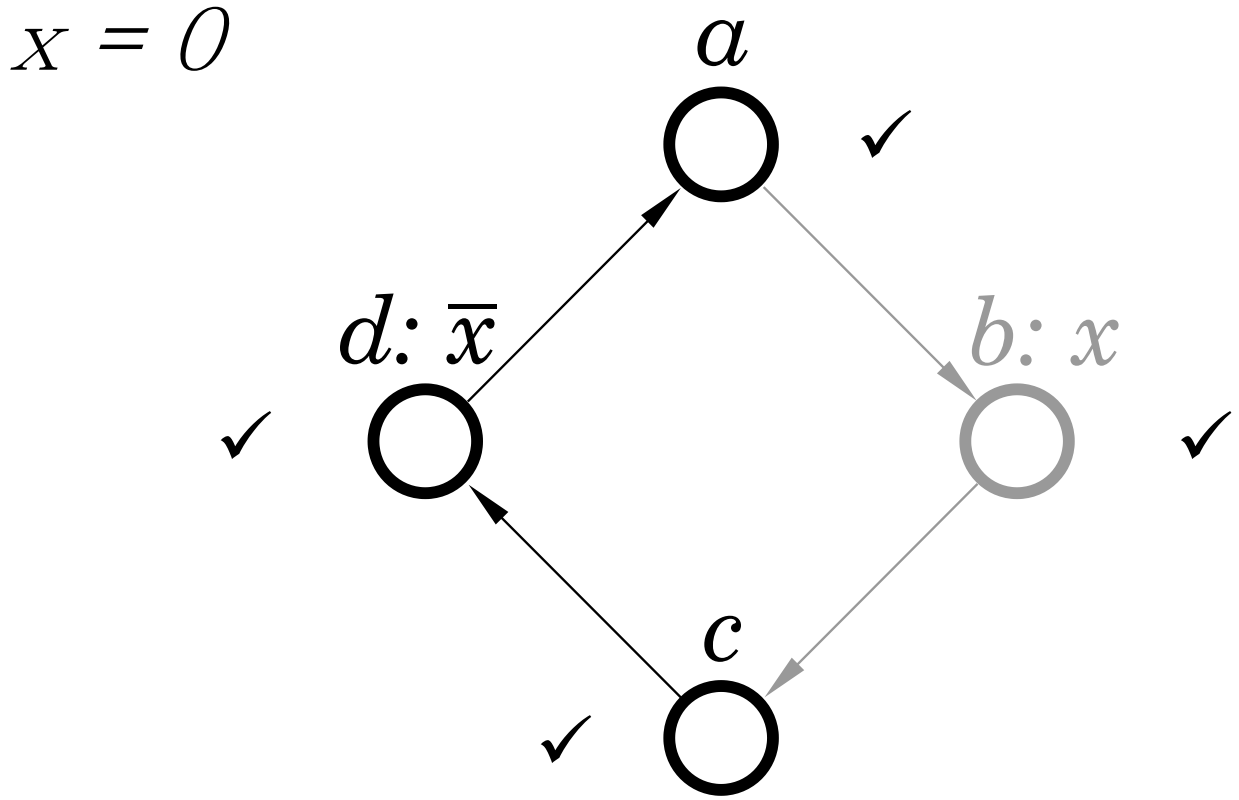
Initial state: $x = 1$

Describing cycles

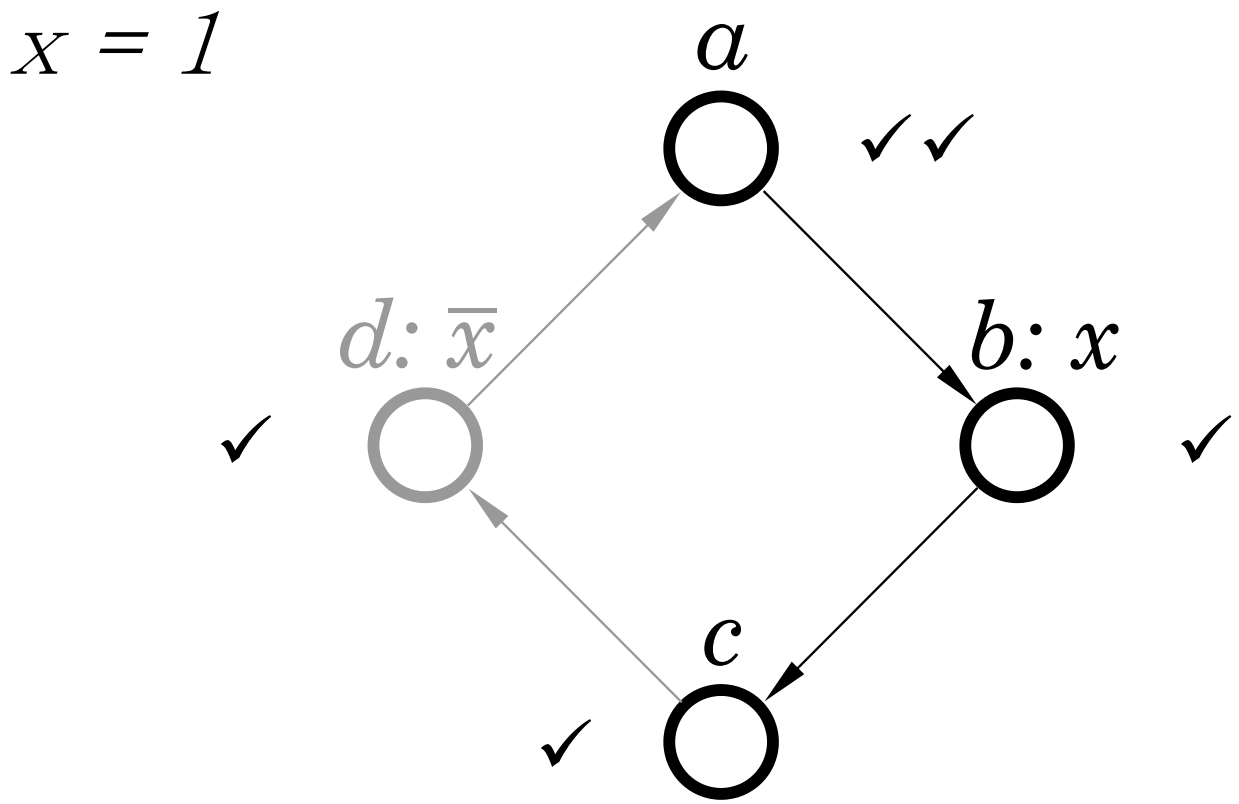
$x = 1$



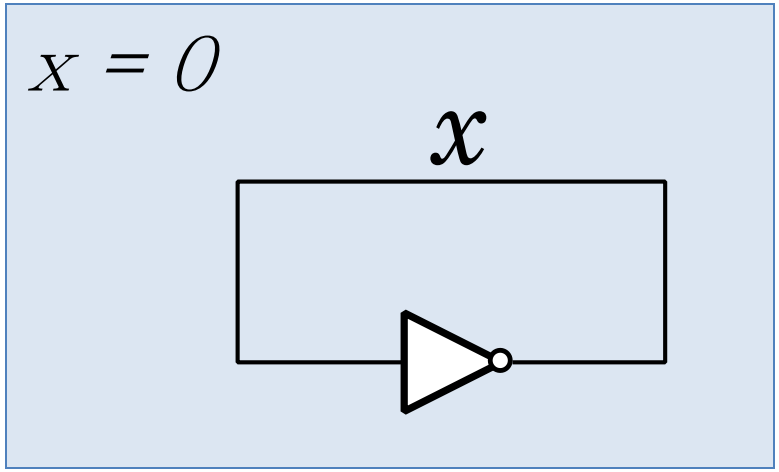
Describing cycles

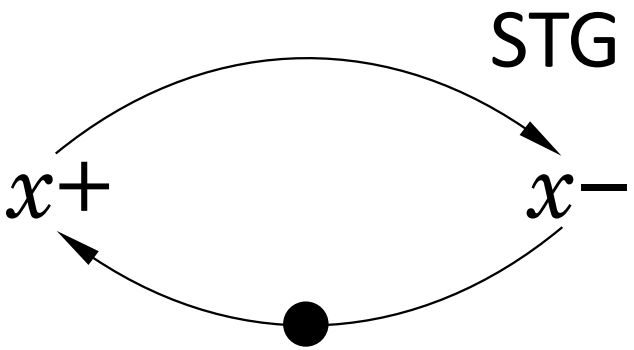


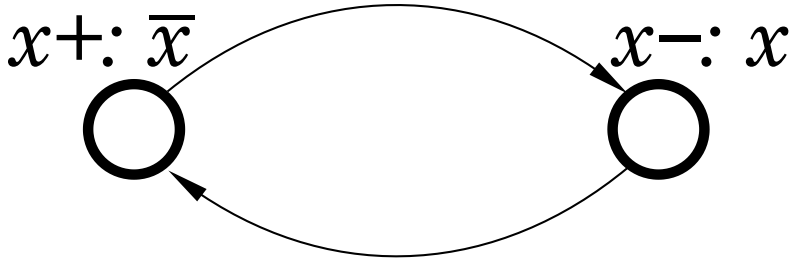
Describing cycles

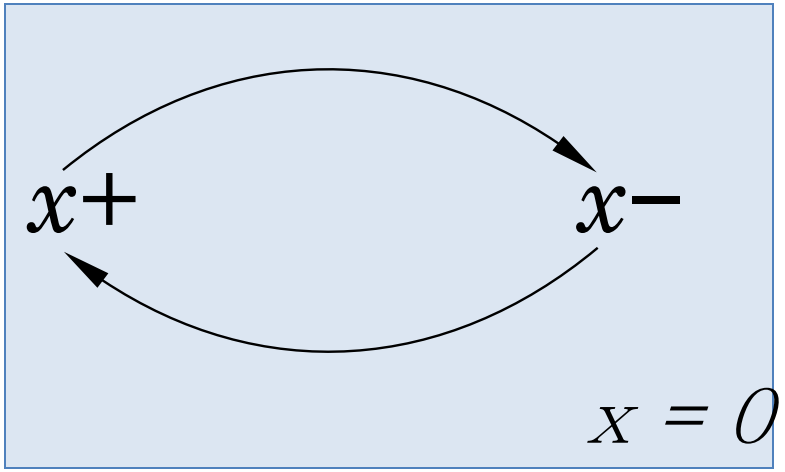


Describing circuits: oscillator



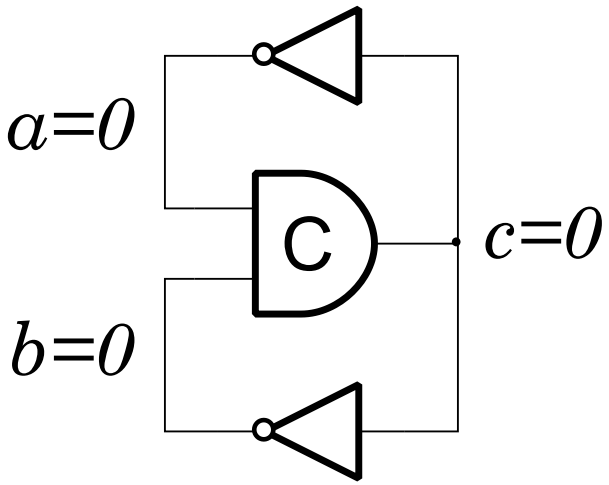


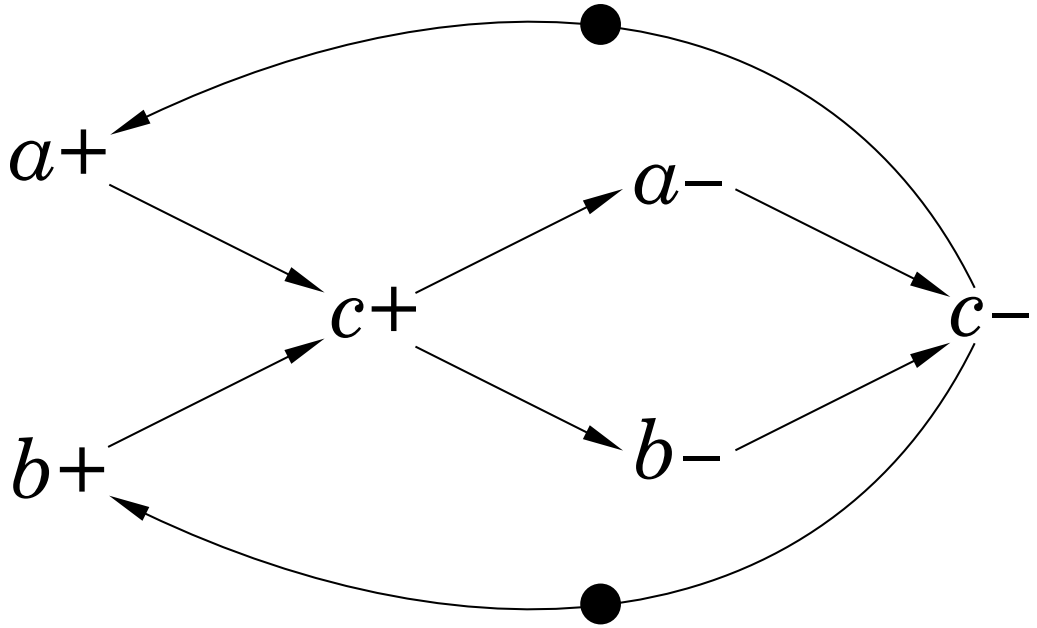




Conditional Signal Graphs (CSGs)

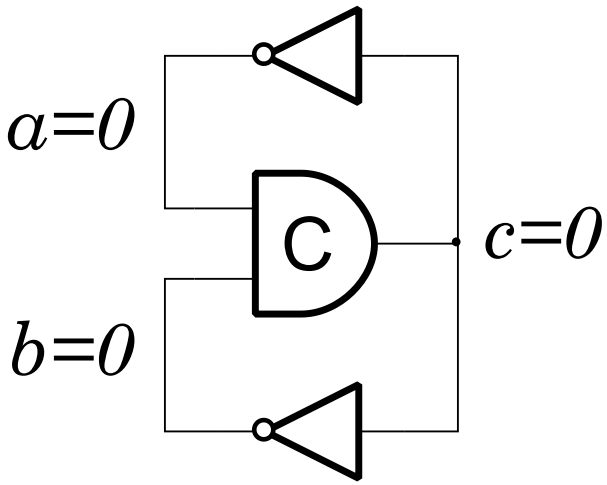
Describing circuits: C-element

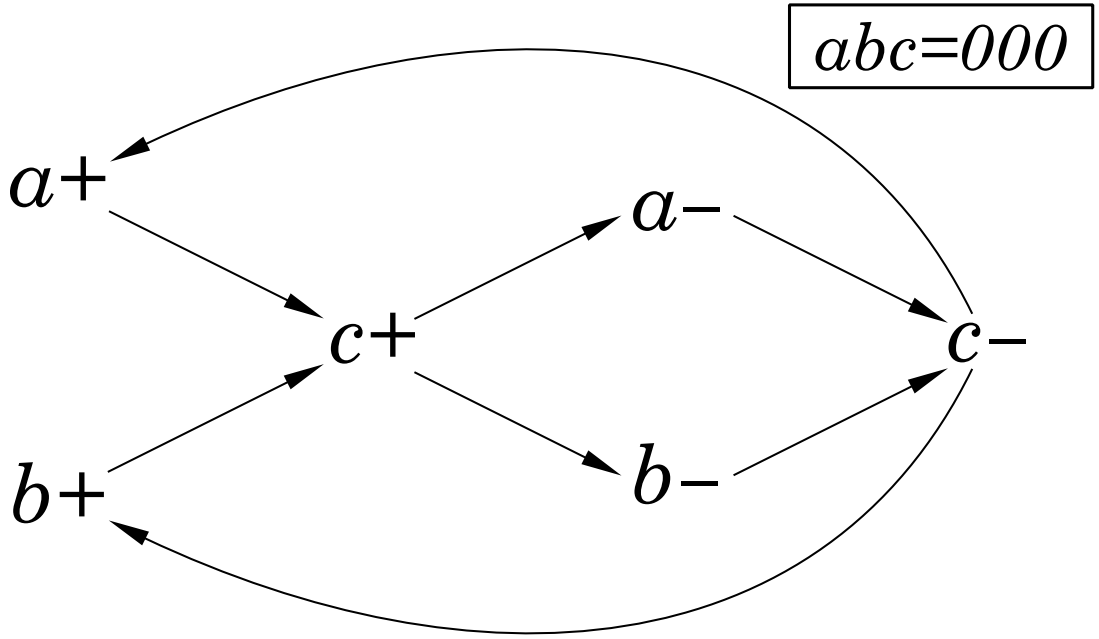




STG

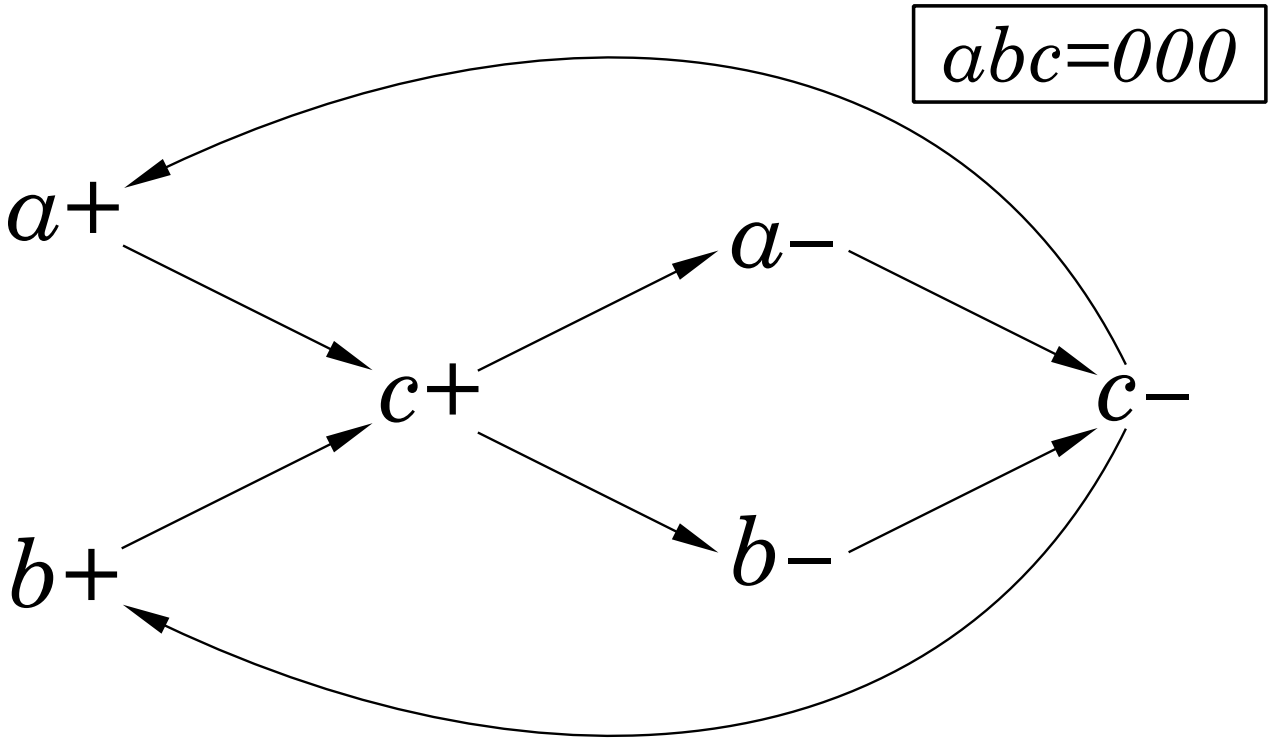
Describing circuits: C-element



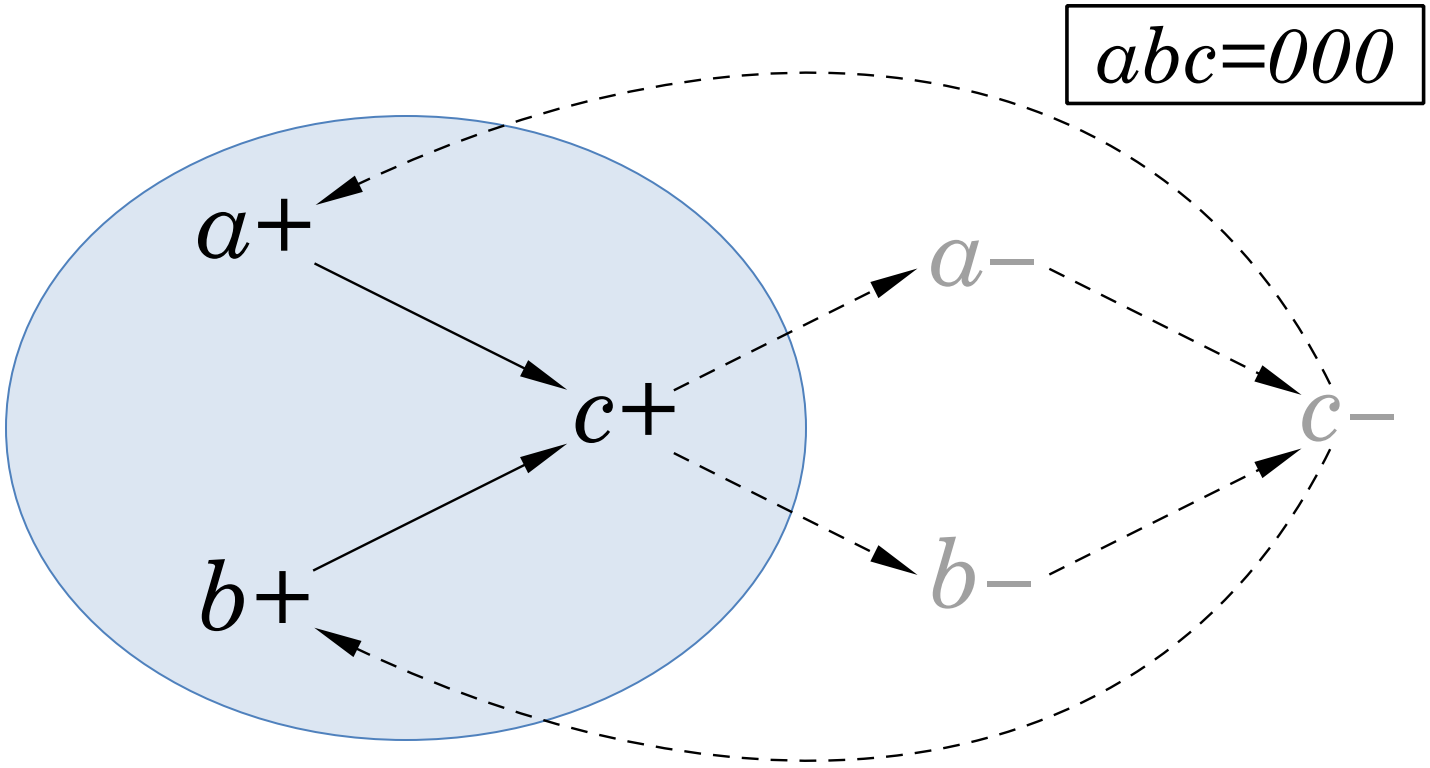


CSG

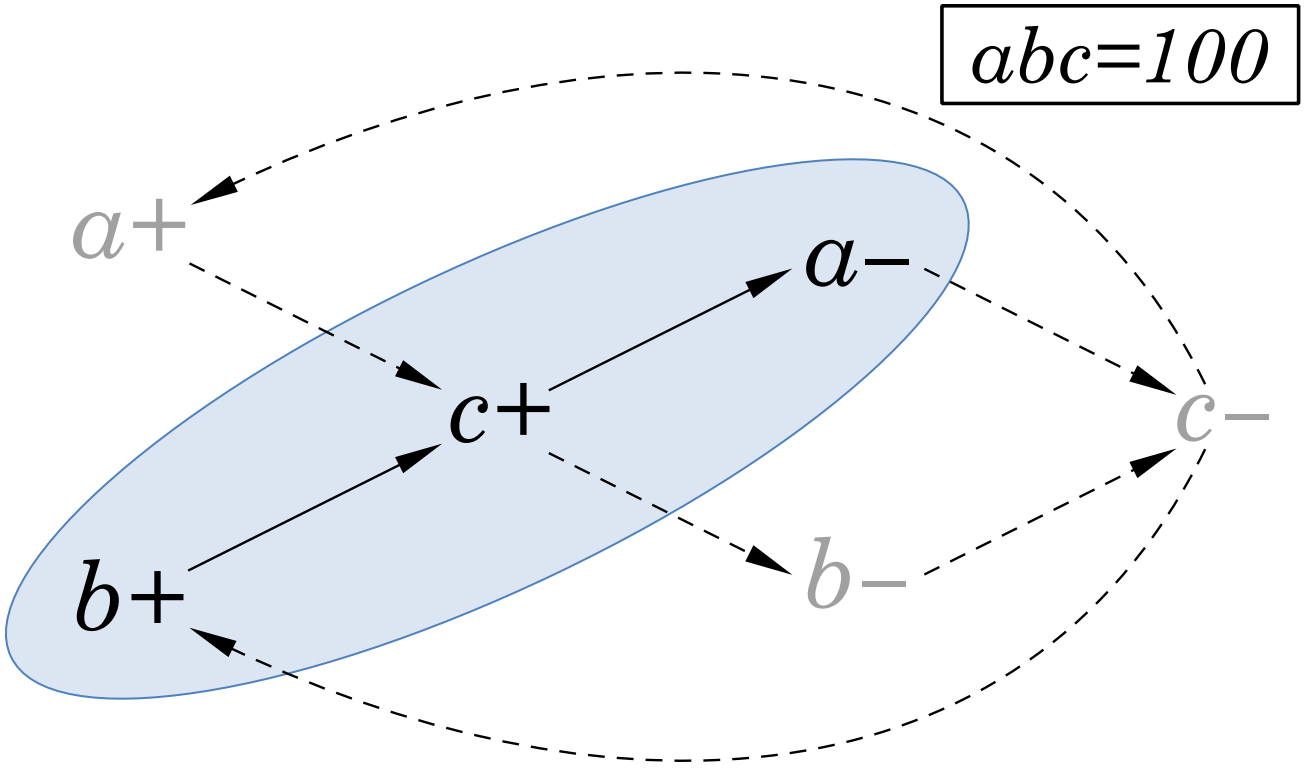
Describing circuits: C-element



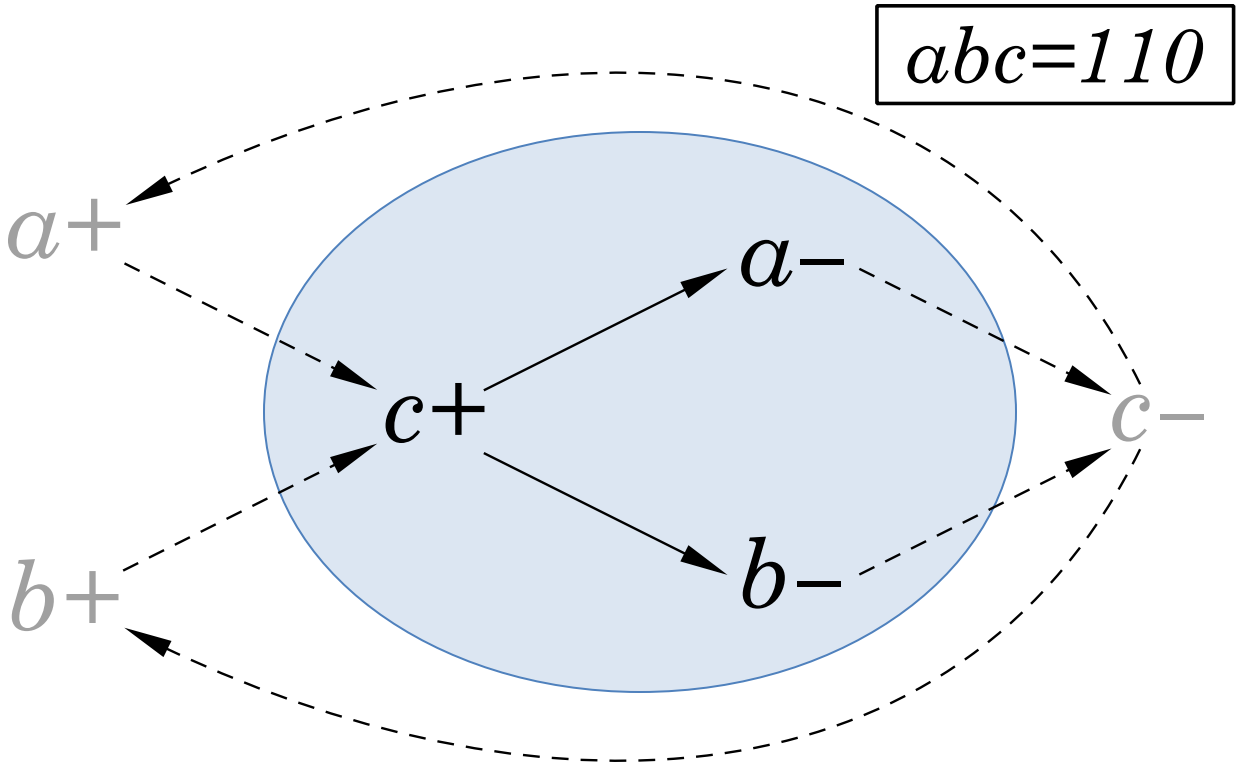
Describing circuits: C-element



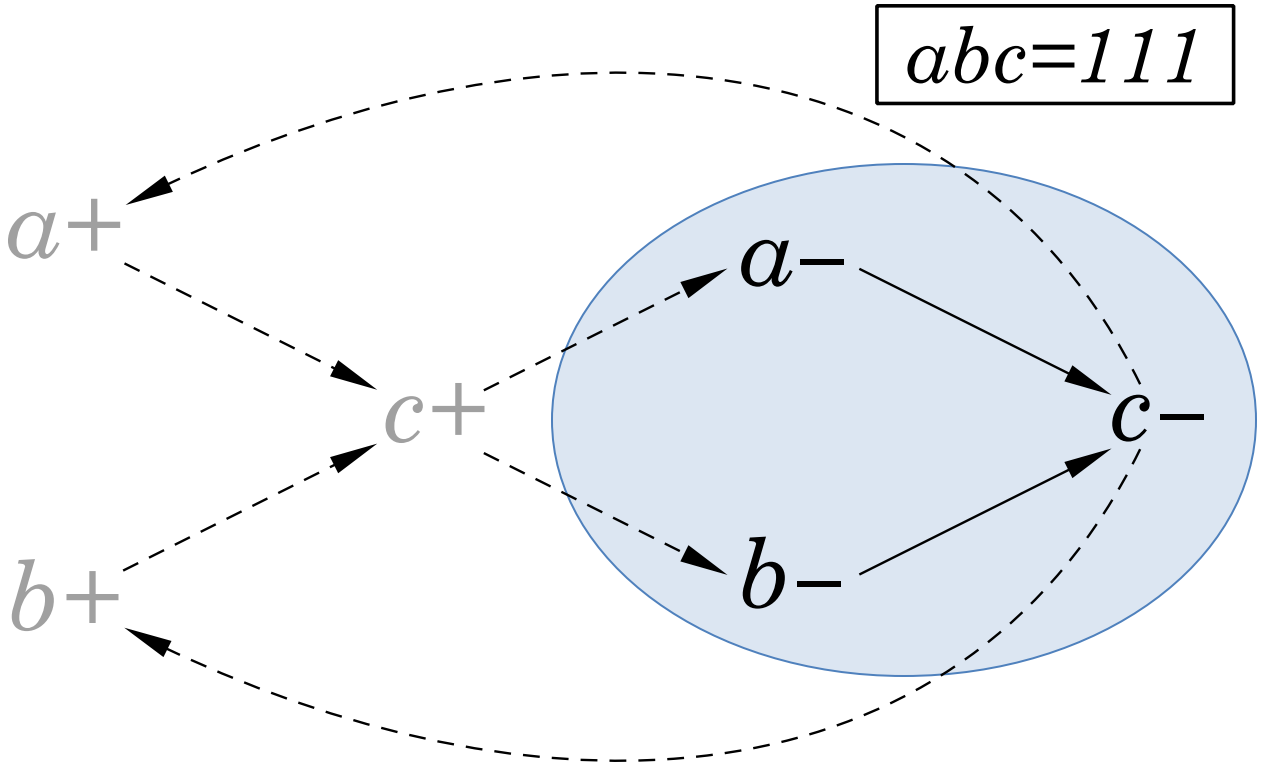
Describing circuits: C-element



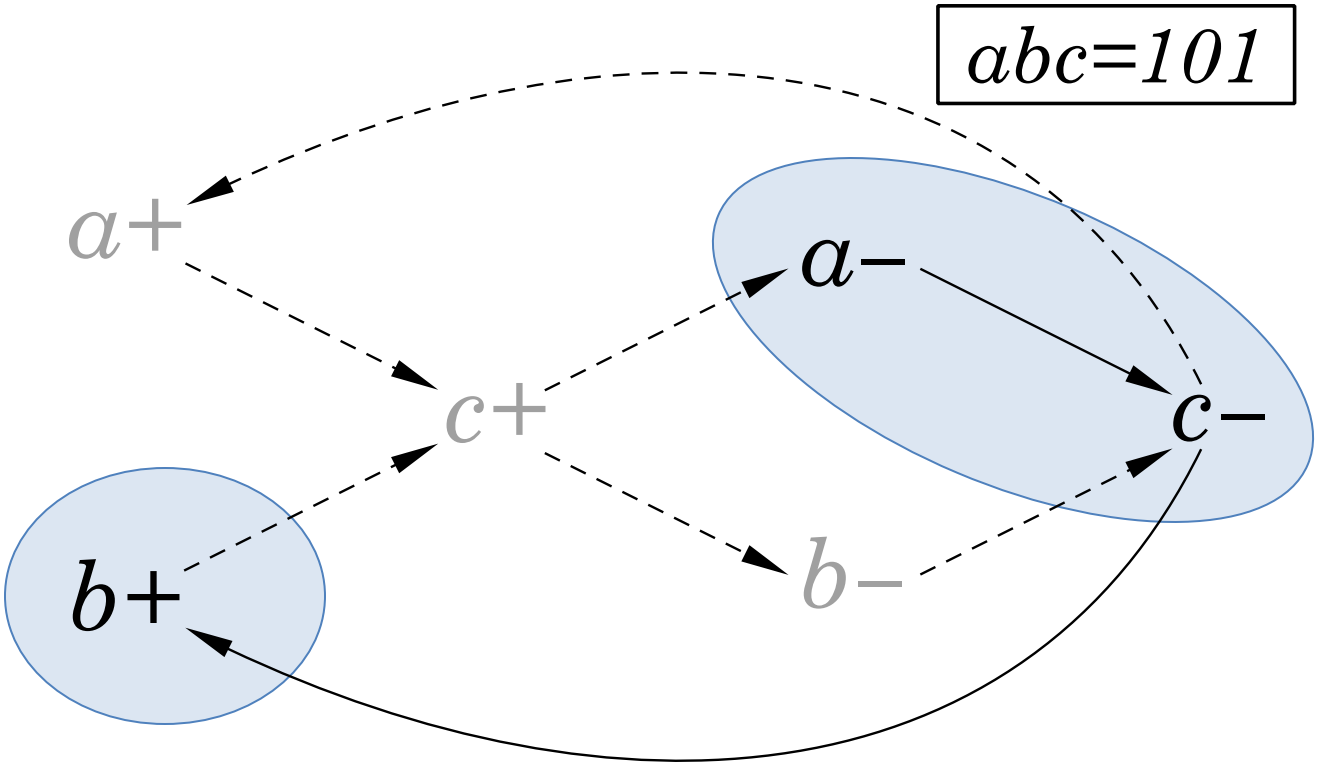
Describing circuits: C-element



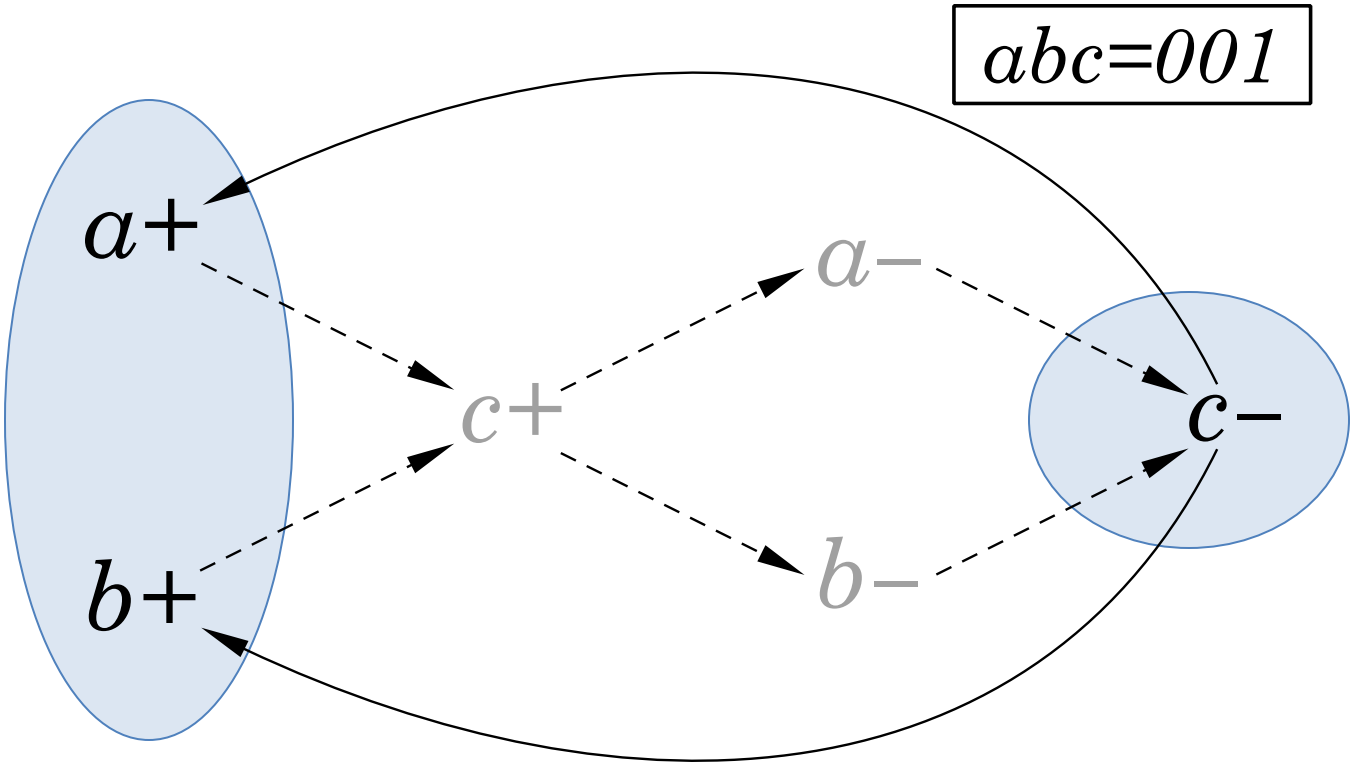
Describing circuits: C-element



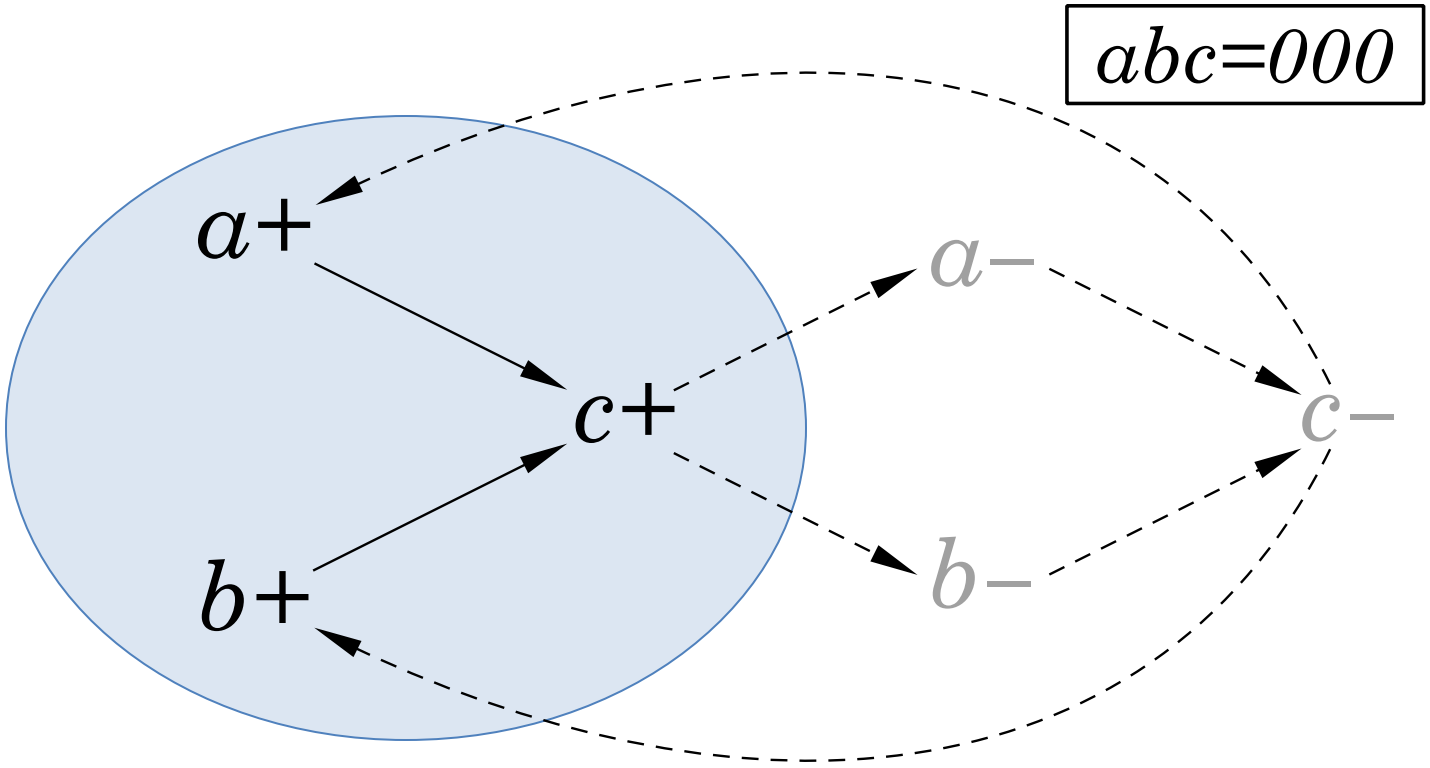
Describing circuits: C-element



Describing circuits: C-element

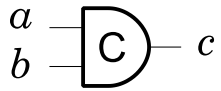


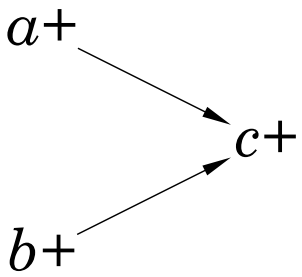
Describing circuits: C-element

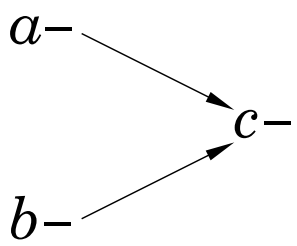


Circuit composition

C-element

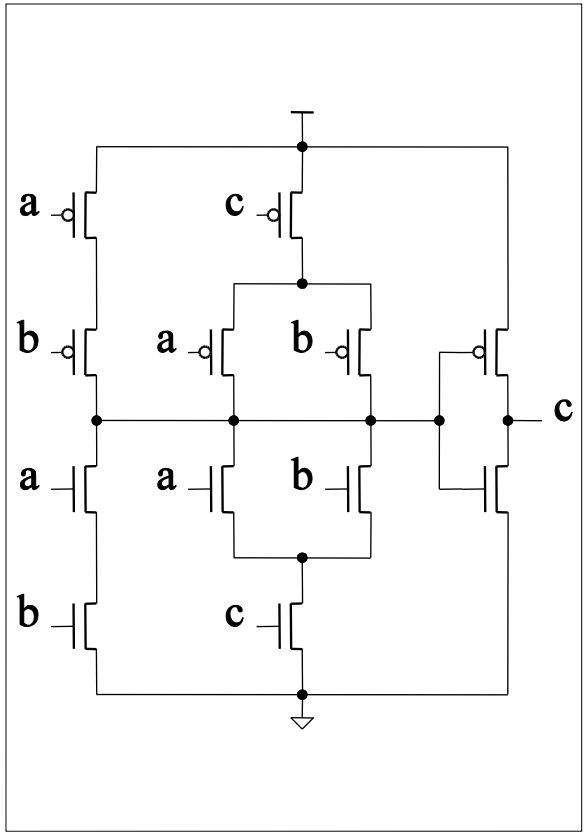


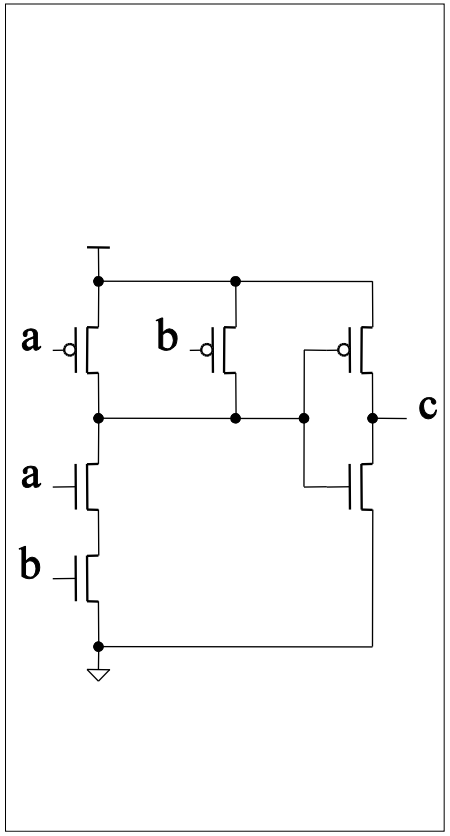


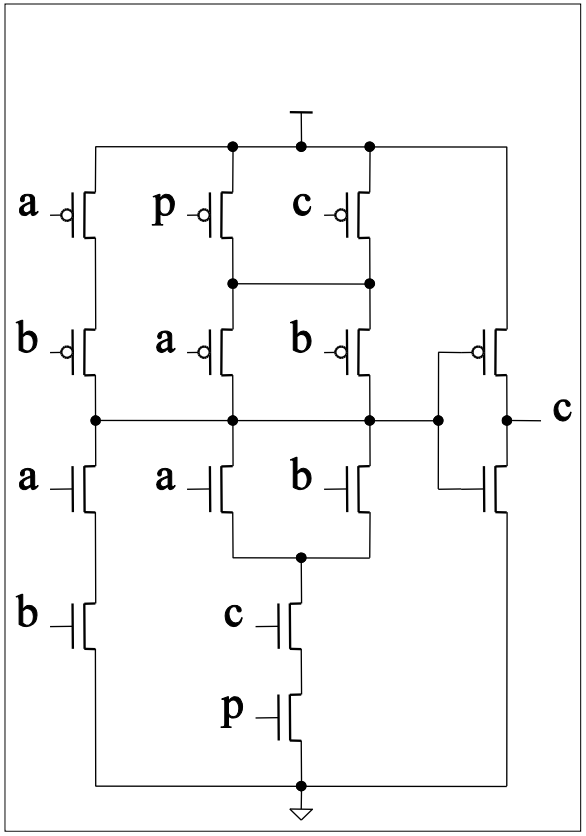


$$c = ab + c(a + b)$$

Circuit composition







C-element:

↑ max(a↑, b↑)

↓ max(a↓, b↓)

AND-gate:

↑ max(a↑, b↑)

↓ min(a↓, b↓)

C/AND-element:

can be chosen in run-time!

Circuit composition

- Algebraic operations with CSGs:
  - Addition (overlay): $G_1 + G_2$
  - Scalar multiplication (encoding): $f \cdot G$
- Composition of C-element and AND-gate:

$$G = p \cdot G_C + \overline{p} \cdot G_{AND}$$

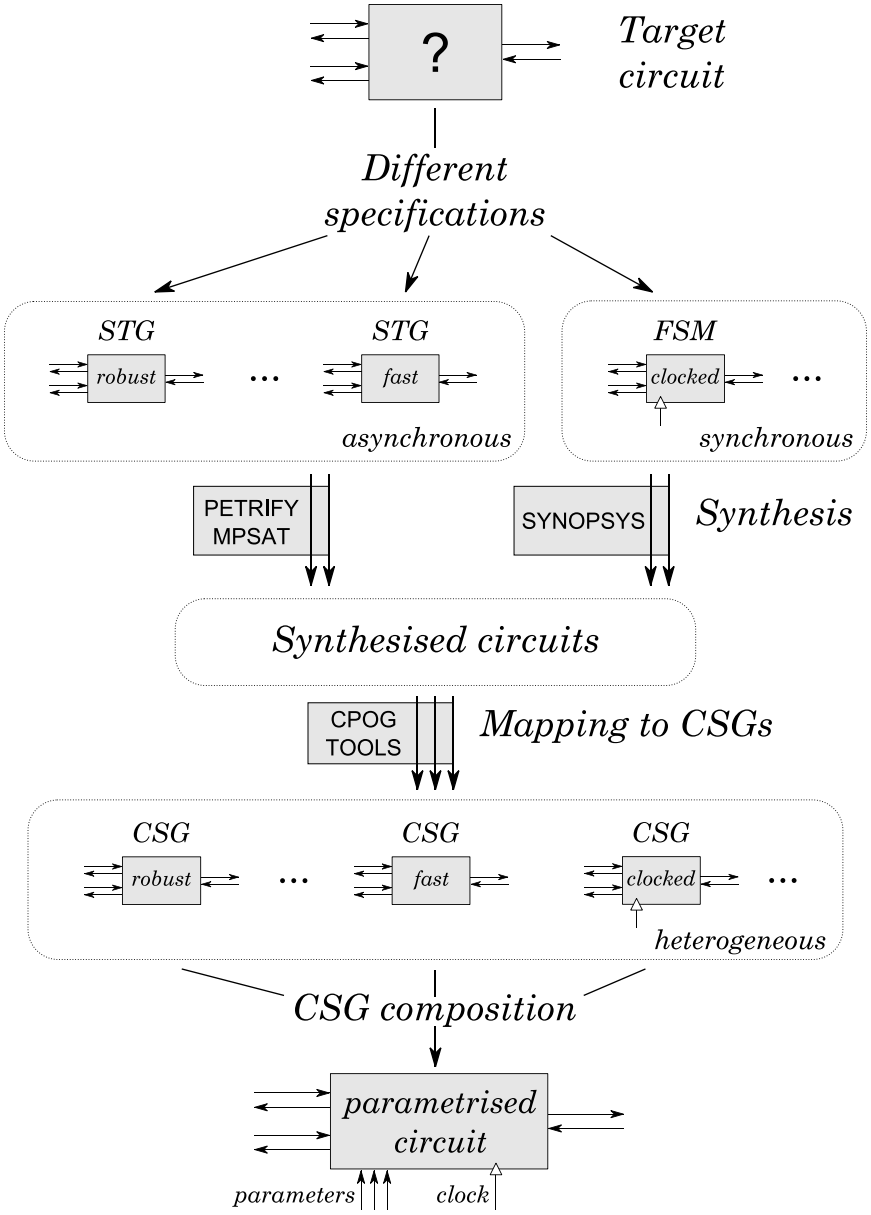
- General composition of *n* circuits:

$$G = f_1 \cdot G_1 + f_2 \cdot G_2 + \cdots + f_n \cdot G_n$$

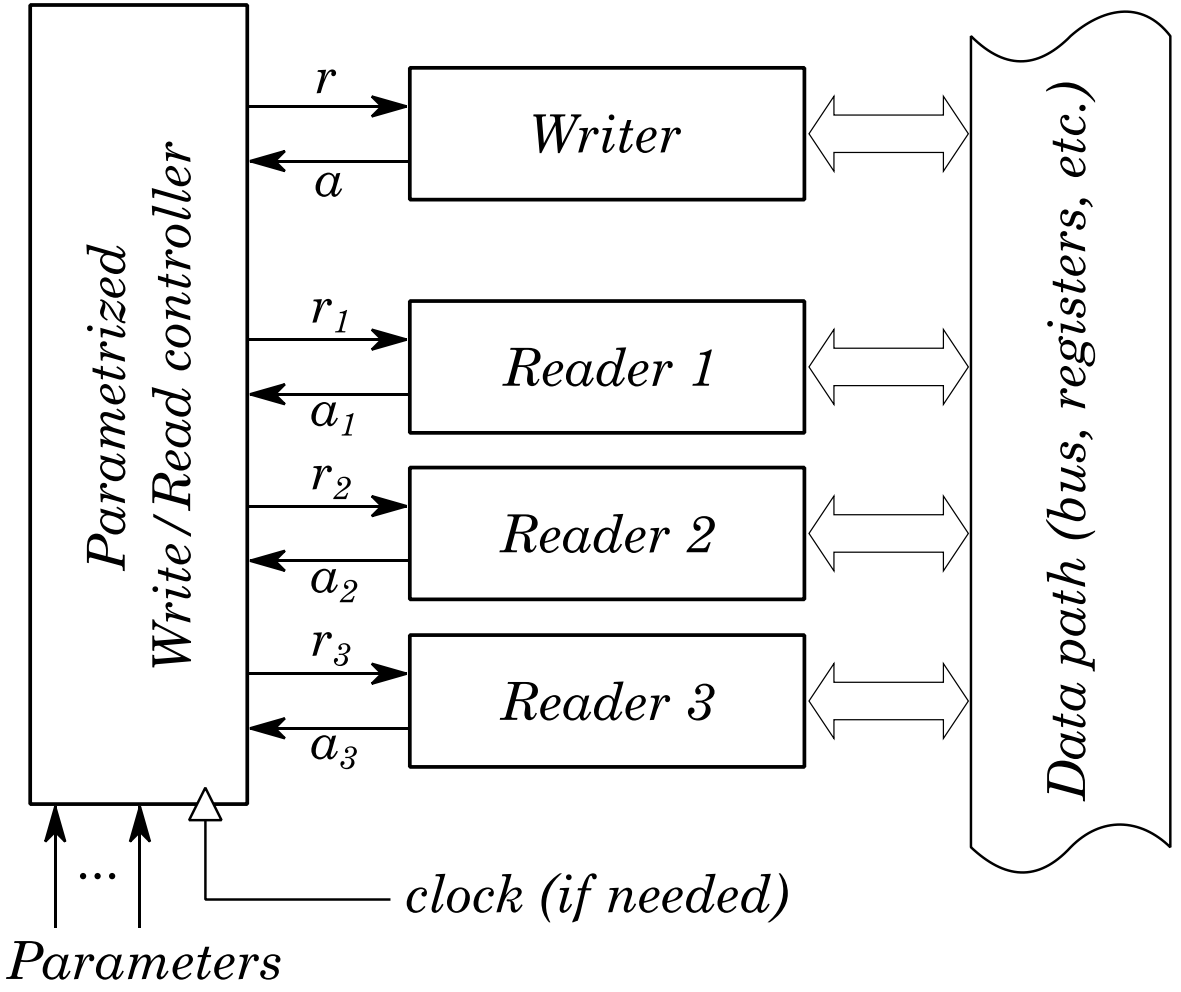
- Fast structural operation (if encoding is given)

Design flow

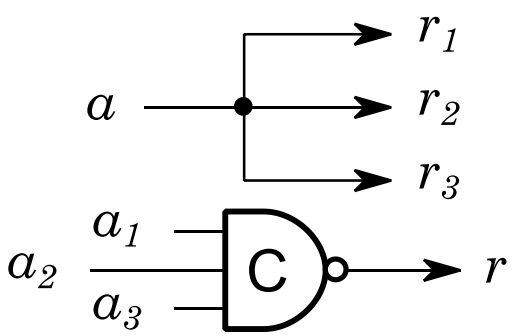
- Heterogeneous models and implementation styles
- Not computationally expensive:
  - composition is fast
  - exact encoding is slow but there are fast approximate methods



Example: Read/Write controller

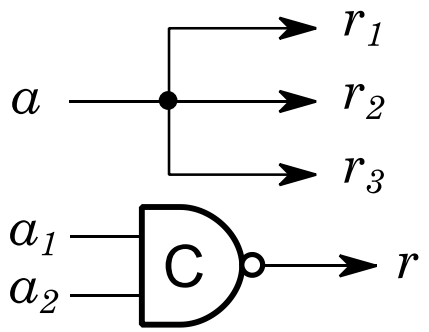


Example: possible implementations



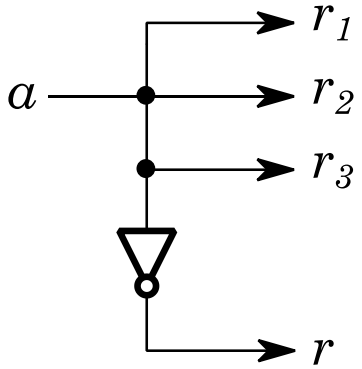
Delay Insensitive (DI)

$T = d + max\{d_1, d_2, d_3\} + C_3$



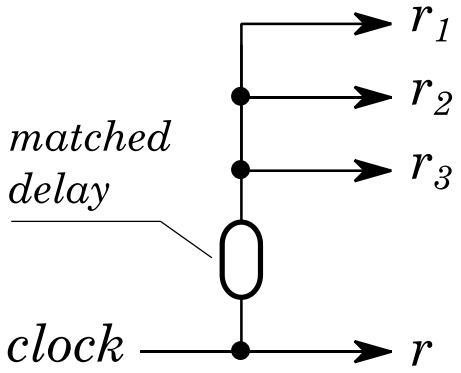
Partial Acknowledgement (PA)

$T = d + max\{d_1, d_2\} + C_2$



Timing Assumptions (TA)

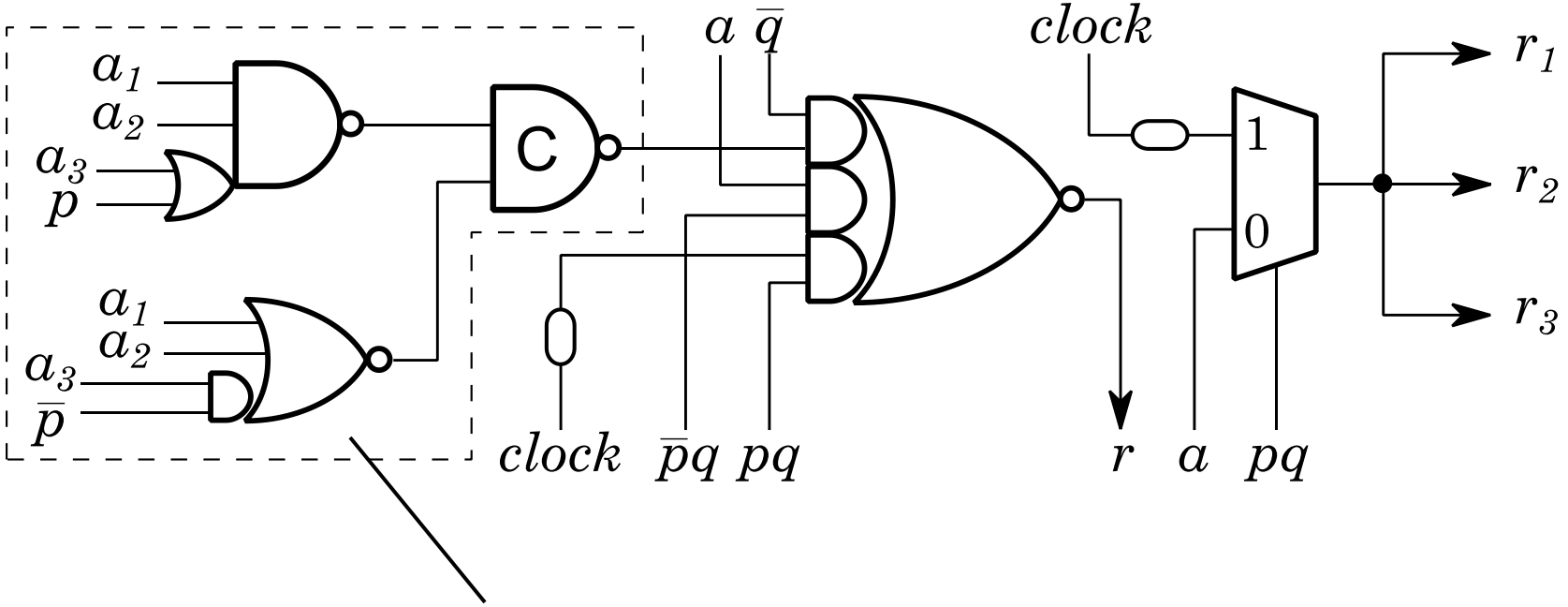
$T = d + I$



Synchronous (CL)

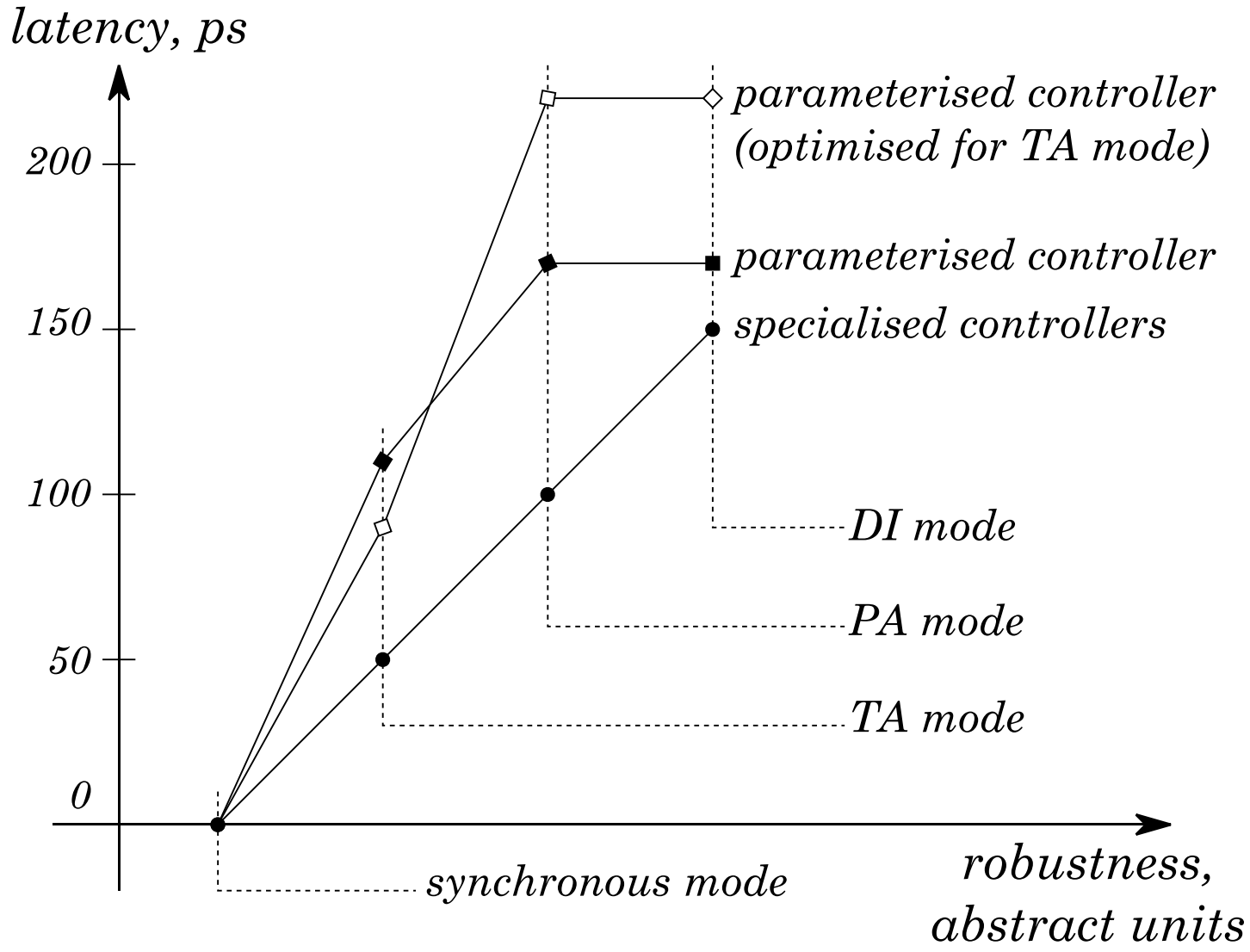
$T = t_{clock}$

Example: parameterised controller



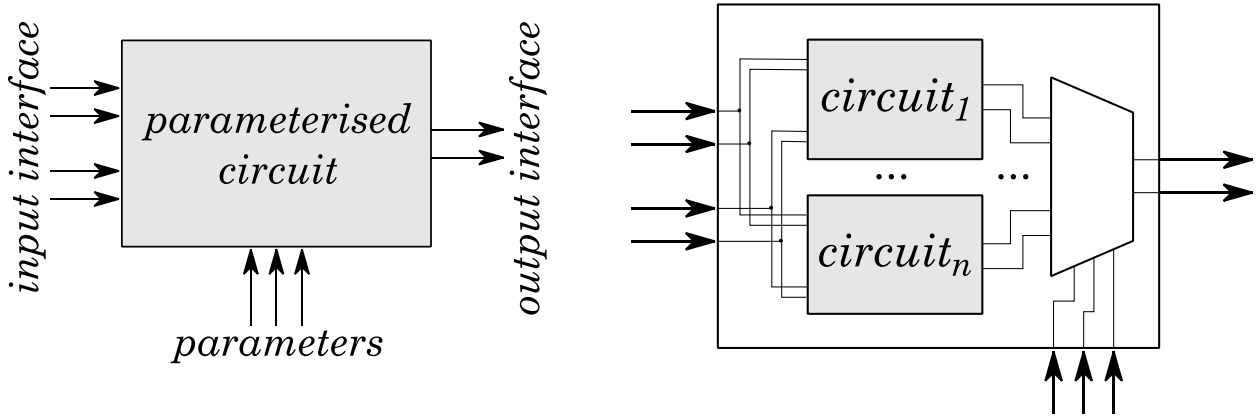
Can be switched off by power-gating in TA/CL modes

Example: simulation results



Cost of Parameterisation

- Trivial approach:
  - Latency overhead: 120-182%, **122%** on average
  - Energy overhead: 151-330%, **201%** on average



- Our approach:
  - Latency overhead: 0-40%, **19%** on average
  - Energy overhead: 0-98%, **23%** on average

- Latency/energy **savings** at the system level!

Conclusions & Future work

- *“Don’t oppose different implementation styles, take advantage of their benefits!”* (reviewer X)
- New model for circuit-level description and composition
- Cost of parameterisation is not prohibitive

- Future work:
  - Tool prototyping
  - CSC signals awareness
  - Power-gating mechanisms

Questions?