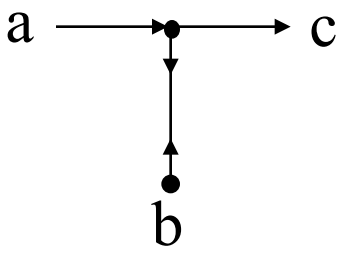
Exercise

- What does this circuit do?

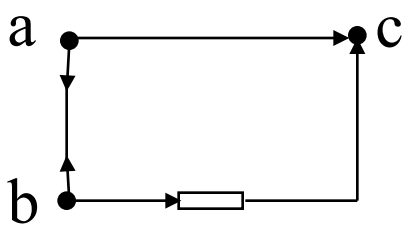


Exercise

- Design a circuit with 3 input nodes a, b, and c, and an output node d, with the following behavior:
  - A take from d succeeds only if there is
    - a value written to a, and
    - a value is written to b or c.

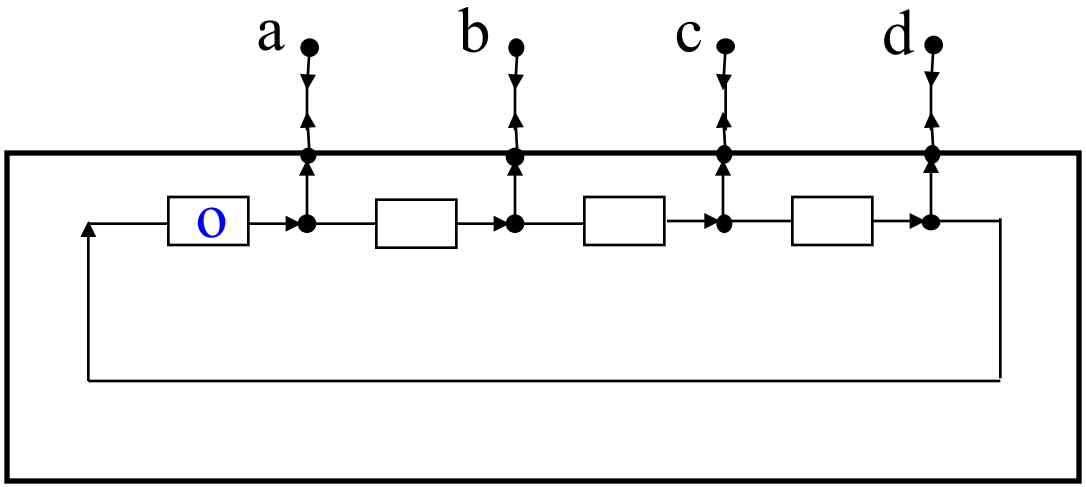
Exercise

- What does this circuit do?



Exercise

- What does this circuit do?



Exercise

- Design a circuit connecting two input nodes a and b to an output node c, with the following behavior:
  - The first take on c produces the first value written to a (regardless of the state of b).
  - The second take on c produces the first value written to b.
  - Successive takes on c produce successive values written to a and b, alternating between them.

Exercise

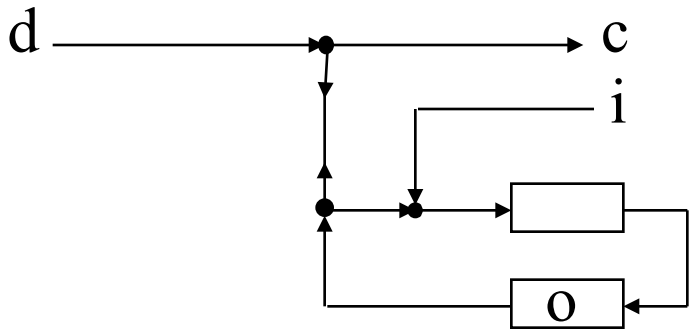
- Design a circuit connecting two input nodes a and b to an output node c, with the following behavior:
  - The first take on c produces the first value written to a.
  - The second take on c produces the second value written to a.
  - The third take on c produces the first value written to b.
  - Successive takes on c repeat the above sequence.

Exercise

- Design a circuit connecting one input node a and one output node b, with the following behavior:
  - First, the first write on a succeeds. The actual value written is irrelevant and is lost.
  - Second, the first take on b succeeds. The actual value taken is irrelevant and can be any arbitrary value.
  - The circuit repeats the above sequence.

Exercise

- What does this circuit do?

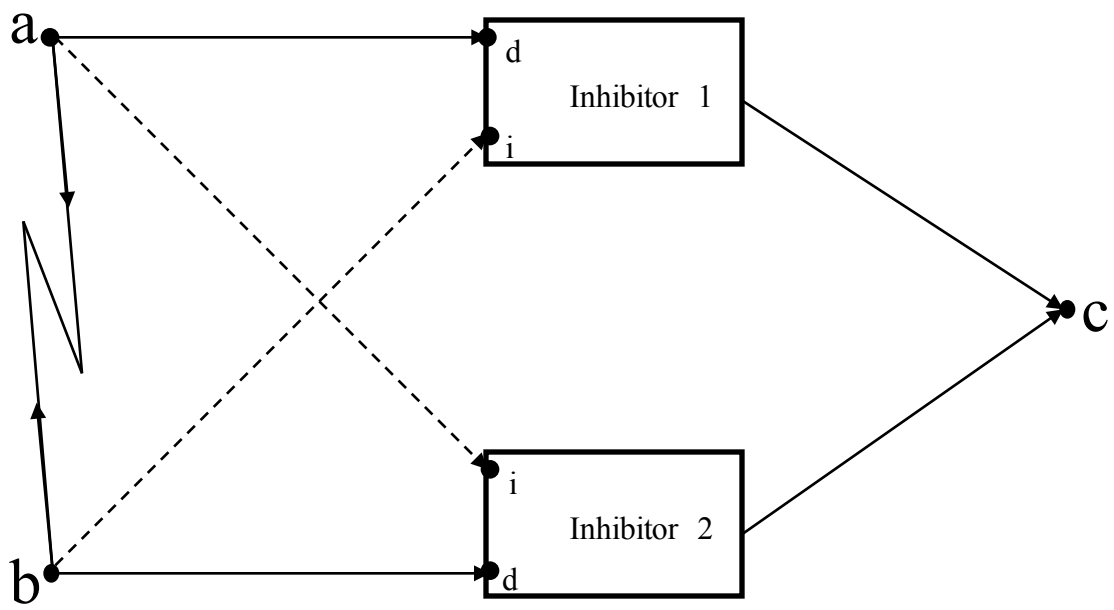


Exercise

- Design an AsyncDrain.
- An AsyncDrain is an asynchronous-drain, which is the dual of a SyncDrain channel.
- An AsyncDrain has two source channel-ends.
- A single write to either of its two ends always succeeds and the written value is lost.
- Two writes on its both ends can succeed in any non-deterministic order, but *never simultaneously*.

Exercise

- What does this circuit do?



Exercise

- Design a circuit to realize the behavior of the fork component in the dining philosophers problem:
  - The fork has two input nodes, *t* (for take) and *f* (for free).
  - The fork has two states, free and taken.
  - Initially the fork is free and accepts a write on its *t* node only. This write changes its state to taken.
  - In its taken state, the fork accepts a write on its *f* node only. This write changes its state to free.
  - All values written to *f* and *t* nodes are lost.

Exercise

- Design an exclusive router circuit connecting one input node a to two output nodes b and c, with the following behavior:
  - Every value written to a is consumed by a take on either b or c, but not both.
  - If there is a write on a and a take on either b or c (but not both), the value is transferred from the write to the take.
  - If there is a write on a and a take on each of b and c, then the circuit transfers the value from a to only one of the two takes, chosen non-deterministically, and the other take remains pending.

Exercise

- Design an inclusive router circuit connecting one input node a to two output nodes b and c, with the following behavior:
  - Every value written to a is consumed by a take on b or c, or both.
  - If there is a write on a and a take on either b or c (but not both), the value is transferred from the write to the take.
  - If there is a write on a and a take on each of b and c, then the circuit transfers the value from a to both takes.

Exercise

- Design a circuit that behaves as an *overflow lossy FIFO1* channel:
  - Like a FIFO1 channel, it has a buffer with the capacity of 1.
  - The buffer is initially empty.
  - It behaves like a normal FIFO1 channel, except that a write that finds the buffer full always succeeds immediately and its value is lost.

Exercise

- Design a circuit that behaves as an *shift lossy FIFO1* channel:
  - Like a FIFO1 channel, it has a buffer with the capacity of 1.
  - The buffer is initially empty.
  - It behaves like a normal FIFO1 channel, except that a write that finds the buffer full causes the channel to lose its current buffer contents, and then succeeds, placing its value into the buffer.

Exercise

- An incoming stream feeds the real-time value of a given stock. Design a circuit that accepts this feed and provides a single output node with the following behavior:
  - Every take on this output node produces the latest available stock value.

Exercise

- Design a circuit to connect a clock, a thermometer, and a display together to yield a time-temperature display system.
- The clock produces a stream of data, one every *c* seconds, each representing the current time.
- The thermometer produces a stream of data, one every *t* seconds, each representing the current temperature.
- The display consumes a stream of data and displays each item for *d* seconds.
- We know that $d>c$ and $d>t$.
- The display must alternately show the latest value produced by the clock, followed by the latest value produced by the thermometer.

Exercise

- Design a circuit to repeat a given constant value. This circuit provides a single output node. Every take on this output node obtains the same constant value.

Exercise

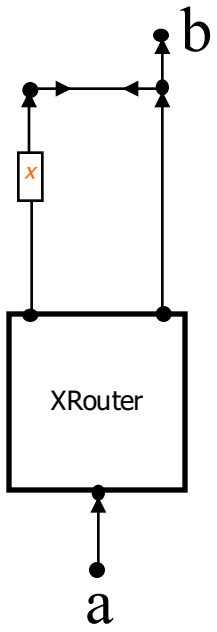
- Design a circuit to replace every input value with a given constant value.
- This circuit has one input and one output nodes.
- Every take on this output node obtains the same constant value.
- A take on the output node succeeds only together with a write on the input node.
- The values written to the input node are lost.

Exercise

- Design a circuit to produce the Fibonacci series using a an adder component.
- The adder component has two input and one output ports:
  - it takes a pair of values, one from each of its two input ports, adds them, and produces the result through its output port.
- To start off the series, use two FIFO1 channels whose buffers are initialized to contain the values 0 and 1, respectively.

Exercise

- What does this circuit do?



Exercise

- Design a valve circuit with the following three nodes:
  - *a*: an input node for normal data input.
  - *b*: an output node for normal data output.
  - *c*: an input node for control data.
- The valve has two states: open and closed.
- Initially, the valve is open.
- In its open state, the valve allows the stream of data flow from *a* to *b*.
- When the valve consumes an input from its control node, *c*, it changes its state to closed.
- In its closed state, the valve stops the flow of data from *a* to *b*.
- The actual values consumed through *c* are irrelevant and are lost.
- The valve must not take a value from *a* unless it can synchronously dispose of it through *b*.
- The data streams at the nodes *a* and *b* must be identical:
  - No data lost
  - No new data produced
  - Same order preserved

Exercise

- Design a valve circuit with the following four nodes:
  - *a*: an input node for normal data input.
  - *b*: an output node for normal data output.
  - *c*: an input node for control data to close the valve.
  - *o*: an input node for control data to open the valve.
- The valve has two states: open and closed.
- Initially, the valve is open.
- In its open state, the valve allows the stream of data flow from *a* to *b*.
- In its open state, the valve consumes and ignores any data on its open control node, o. Opening an open valve is a no-operation.
- The first input from its close control node, *c*, closes the valve.
- In its closed state, the valve consumes and ignores any data on its close control node, c. Closing a closed valve is a no-operation.
- In its closed state, the valve stops the flow of data from *a* to *b*.
- The actual values consumed through *c* and *o* are irrelevant and are lost.
- The valve must not take a value from *a* unless it can synchronously dispose of it through *b*.
- The data streams at the nodes *a* and *b* must be identical:
  - No data lost
  - No new data produced
  - Same order preserved

Exercise

- What does this circuit do?