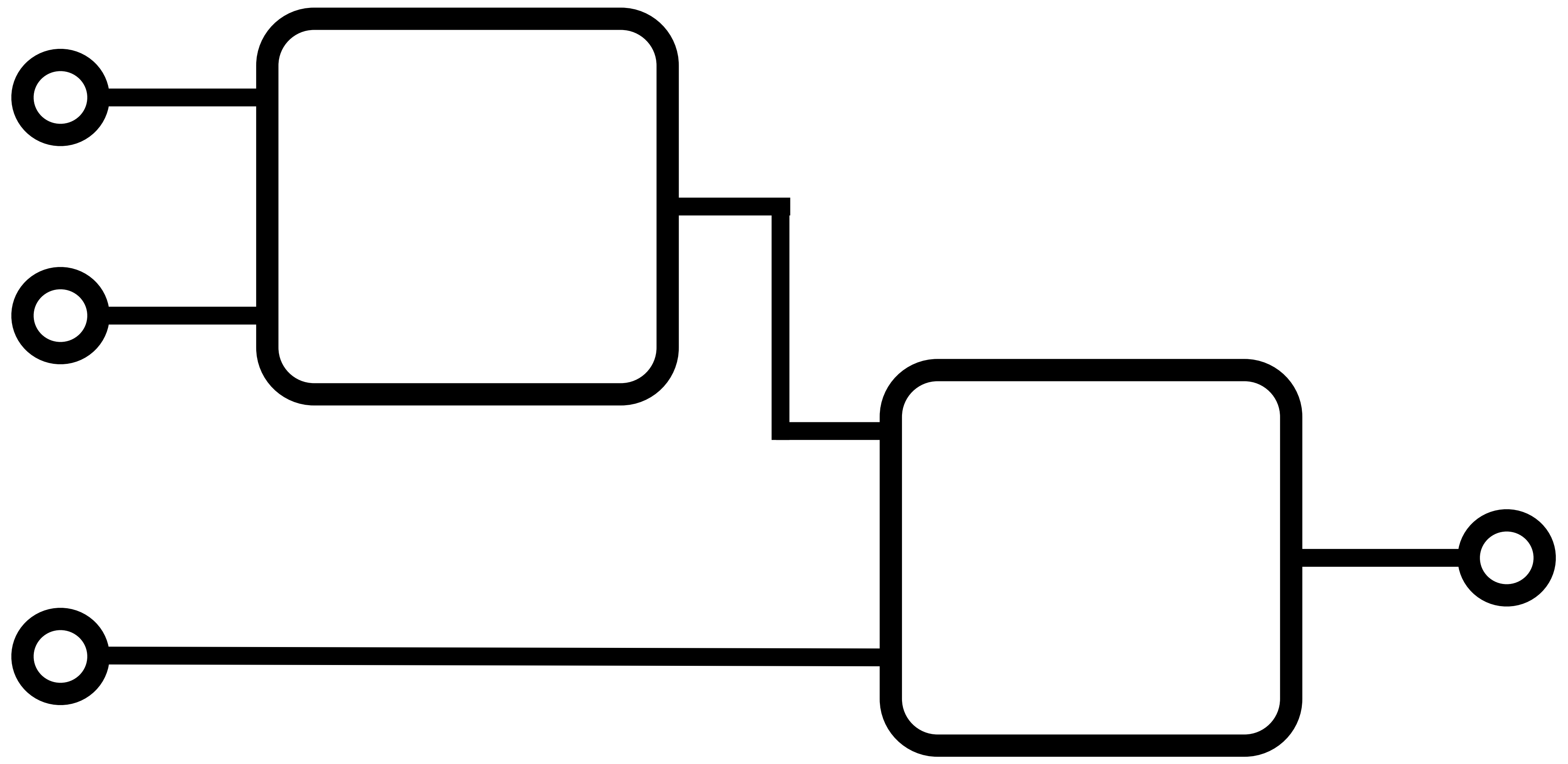
Tri-State Circuits

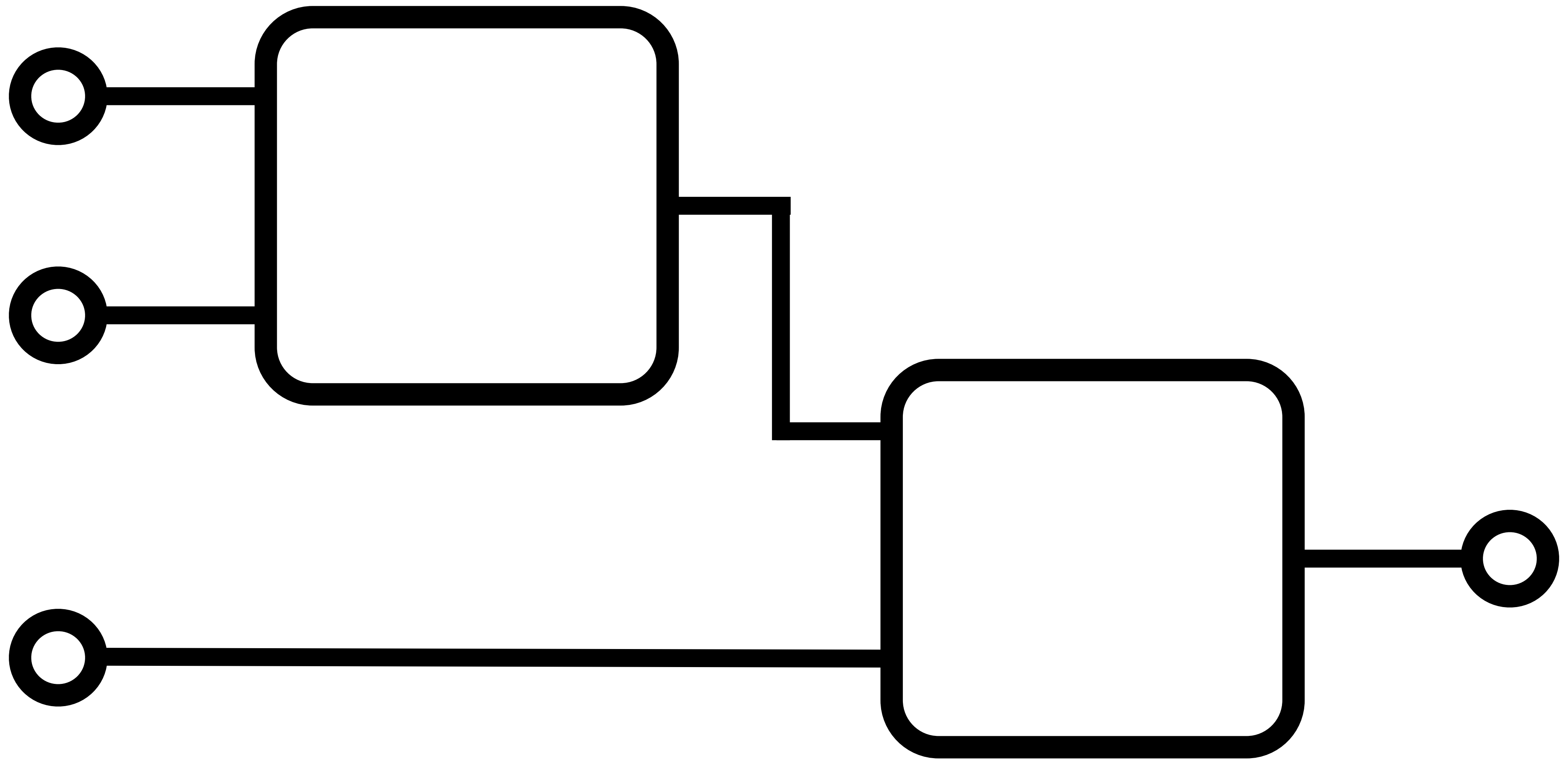
A Circuit Model that Captures RAM

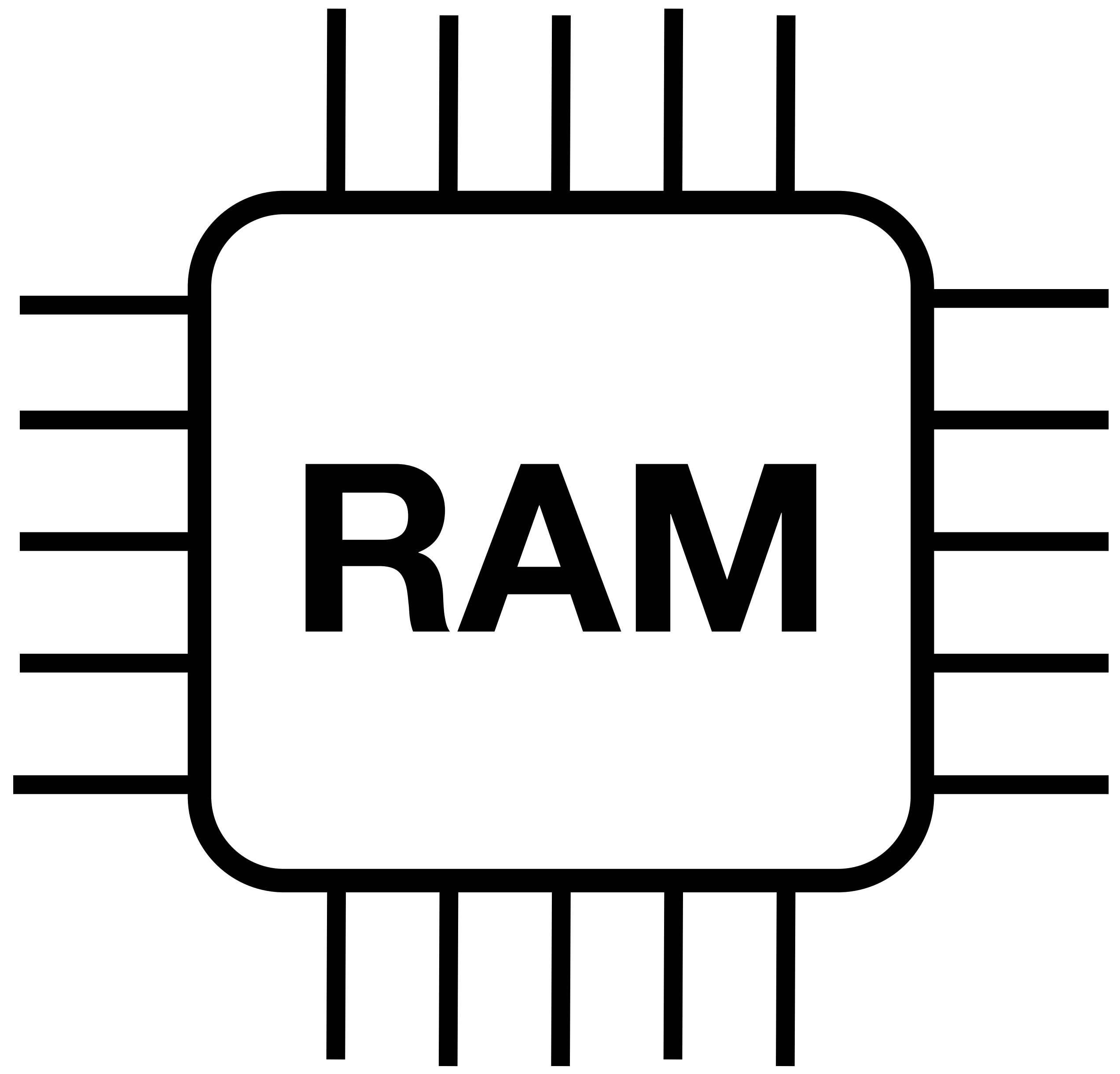
David Heath	**UIUC**
Vlad Kolesnikov	Georgia Tech
Rafi Ostrovsky	UCLA



Boolean Circuits

Convenient for cryptographic protocols and complexity theory



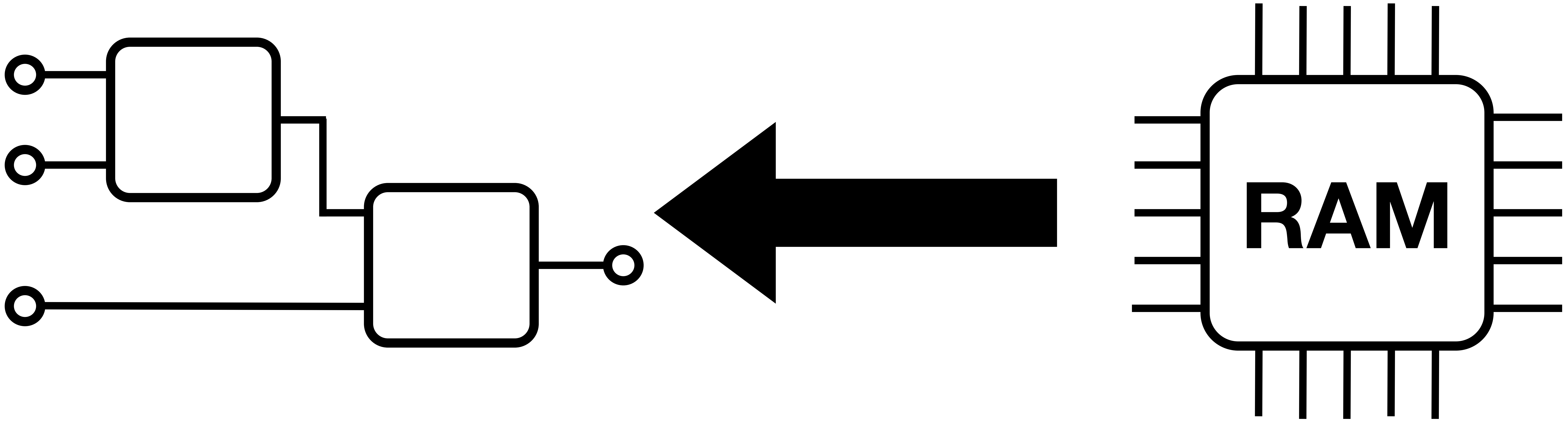


Boolean Circuits

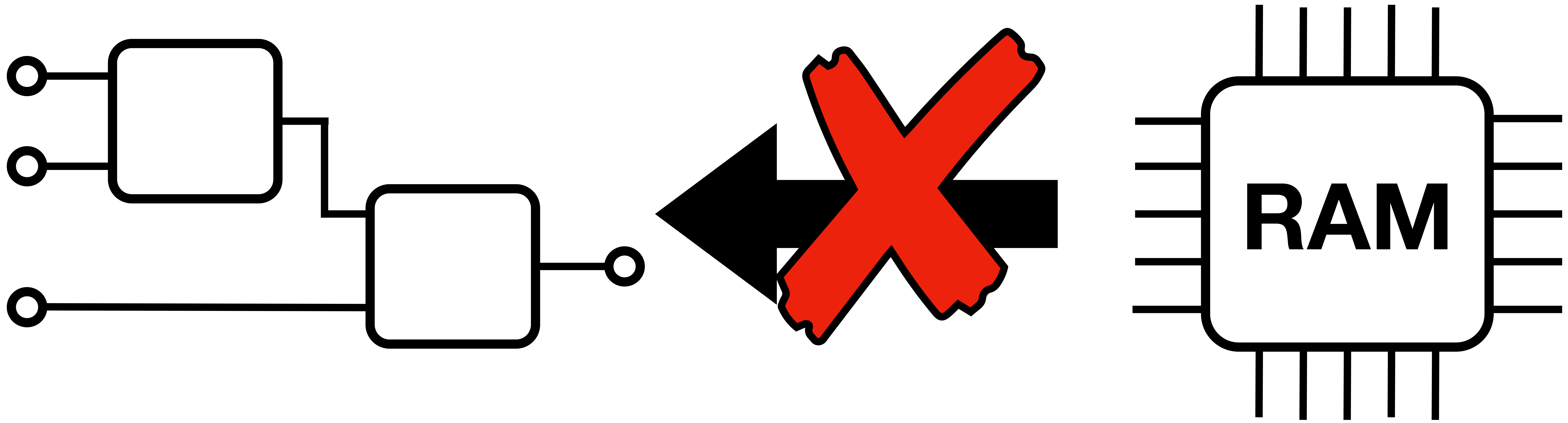
Convenient for cryptographic protocols and complexity theory

Random Access Machines

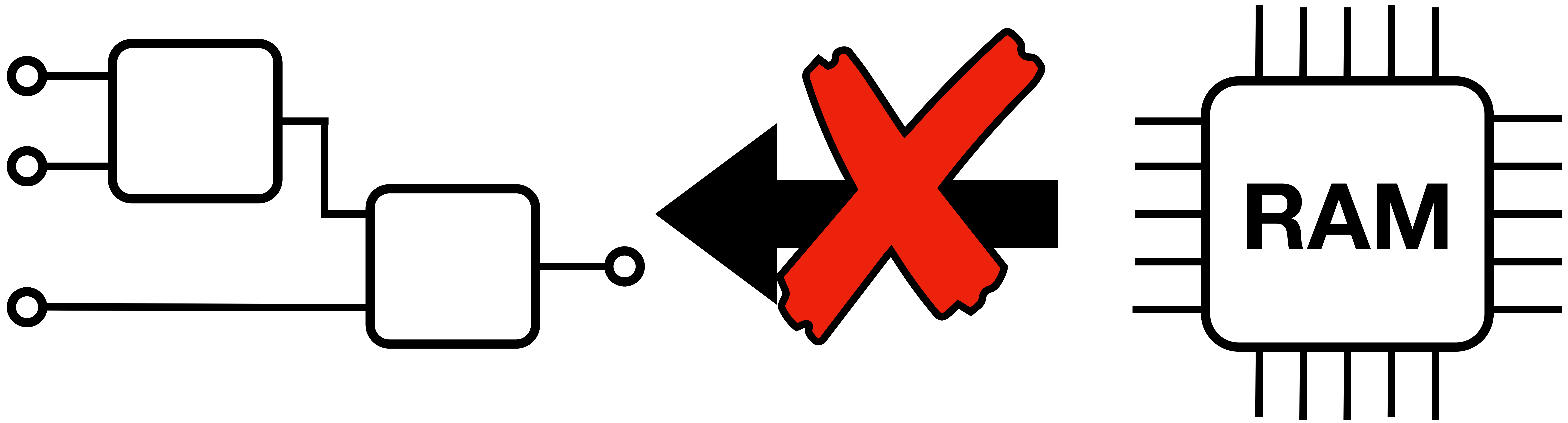
Convenient for algorithms and applications



efficient emulation would be convenient



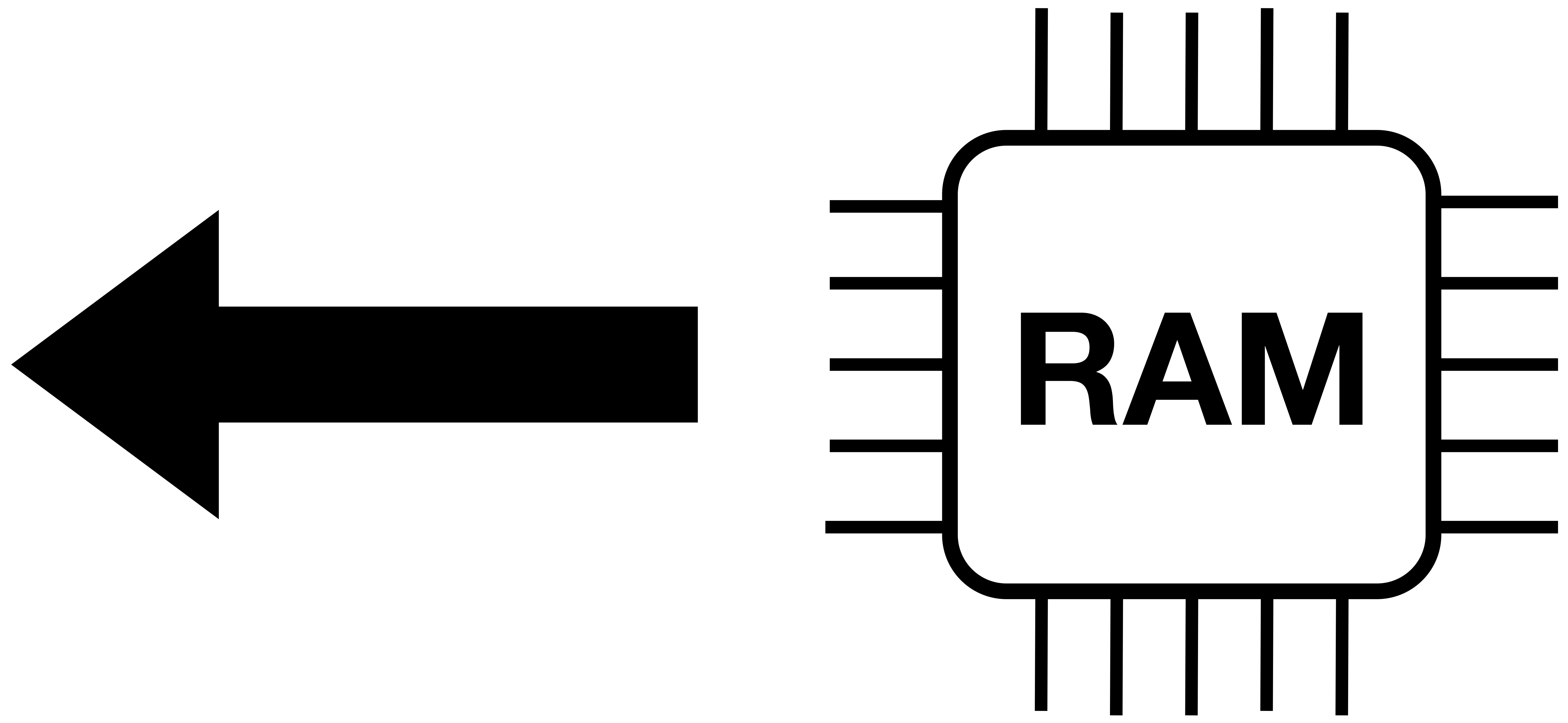
We do not have efficient *(i.e. quasilinear)* **Boolean circuits that emulate RAM**



We do not have efficient *(i.e. quasilinear)* Boolean circuits that emulate RAM

Key challenge: how can we emulate memory access?

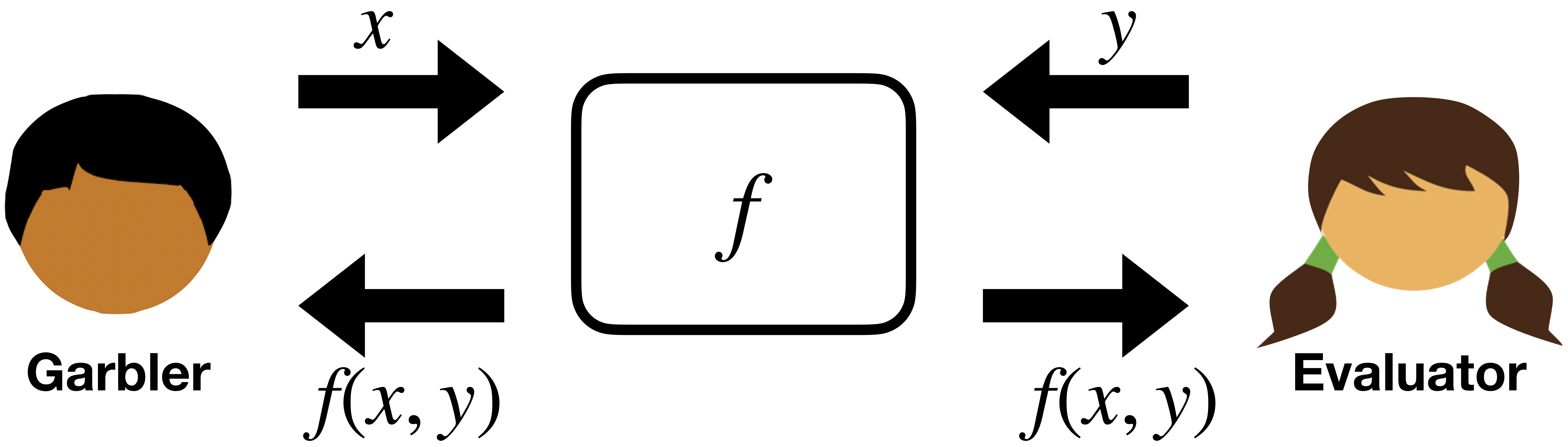
Tri-State Circuits



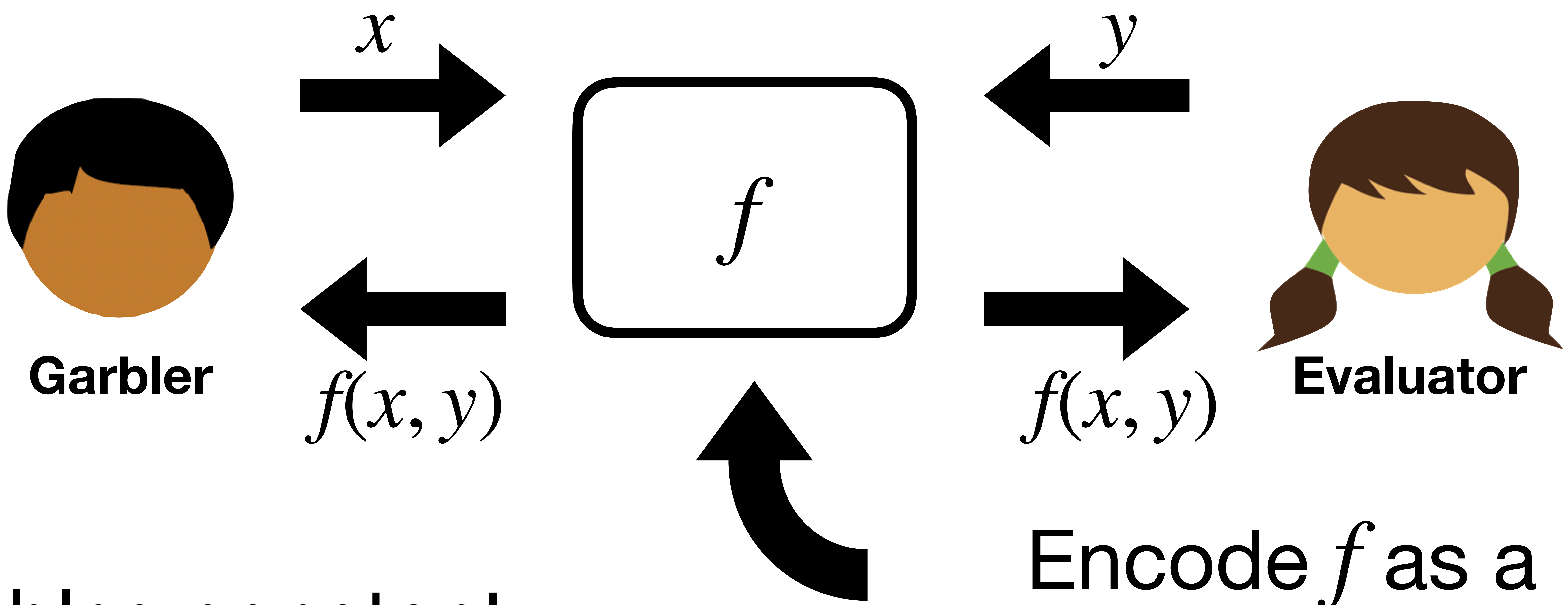
Tri-state circuits *can* efficiently emulate RAM

(i.e. with quasilinear overhead)

Our Application: Yao's Garbled Circuit



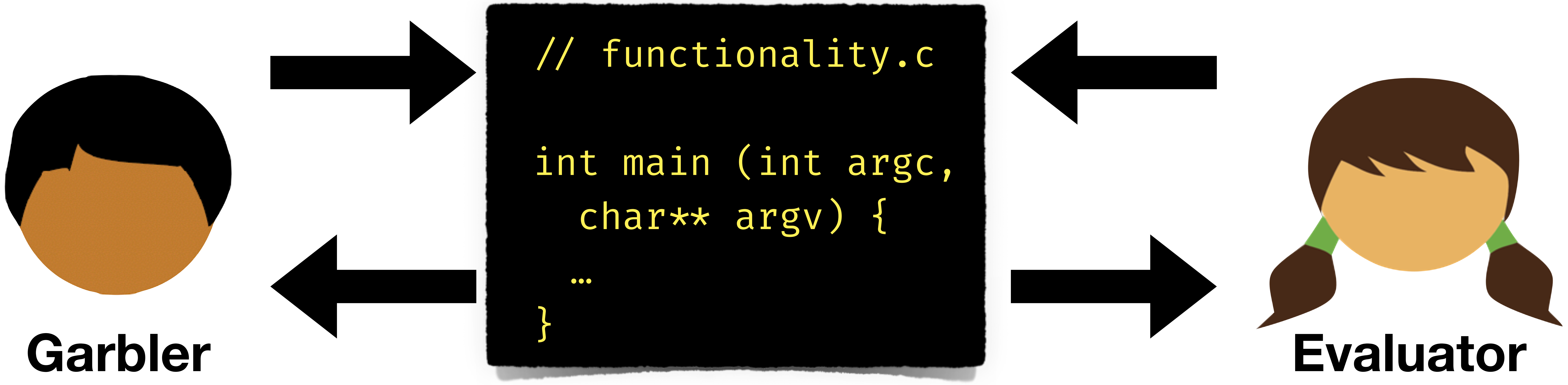
Our Application: **Yao's Garbled Circuit**



Enables constant round MPC protocols

Encode f as a Boolean circuit

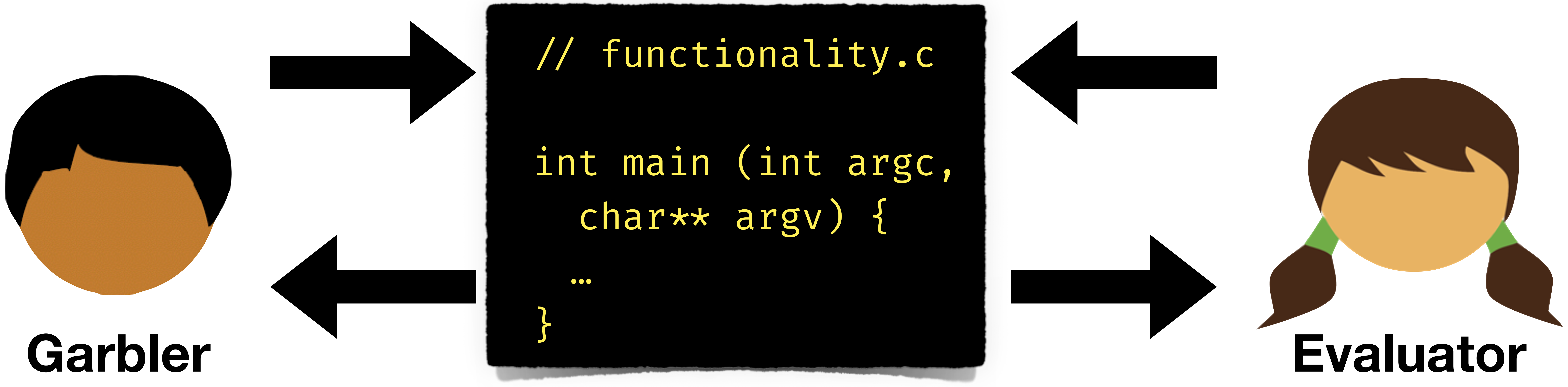
Garbled RAM



```
// functionality.c

int main (int argc,
  char** argv) {
   …
}
```

Garbled RAM



```
// functionality.c

int main (int argc,
  char** argv) {
  …
}
```

How to Garble RAM Programs*

Steve Lu† Rafail Ostrovsky‡

Garbled RAM Revisited
Part I

Craig Gentry* Shai Halevi* Mariana Raykova † Daniel Wichs‡

February 5, 2014

Garbled RAM From One-Way Functions

Sanjam Garg* Steve Lu† Rafail Ostrovsky‡ Alessandra Scafuro§

Practical Garbled RAM
GRAM with $O(\log^2 n)$ Overhead

David Heath[1], Vladimir Kolesnikov[2], and Rafail Ostrovsky[3]

NanoGRAM: Garbled RAM with $\widetilde{O}(\log N)$ Overhead

Andrew Park Wei-Kai Lin Elaine Shi*

Carnegie Mellon University

Abstract

We propose a new garbled RAM construction called NanoGRAM, which achieves an amortized cost of $\widetilde{O}(\lambda\cdot(W \log N + \log^3 N))$ bits per memory access, where λ is the security parameter, W is the block size, and N is the total number of blocks, and $\widetilde{O}(\cdot)$ hides poly log log factors. For sufficiently large blocks where $W = \Omega(\log^2 N)$, our scheme achieves $\widetilde{O}(\lambda \cdot W \log N)$ cost per memory access, where the dependence on N is optimal (barring poly log log factors), in terms of the evaluator's runtime. Our asymptotical performance matches even the *interactive* state-of-the-art (modulo poly log log factors), that is, running Circuit ORAM atop garbled circuit, and yet we remove the logarithmic number of interactions necessary in this baseline. Furthermore, we achieve asymptotical improvement over the recent work of Heath et al. Our scheme adopts the same assumptions as the mainstream literature on practical garbled circuits, i.e., circular correlation-robust hashes or a random oracle. We evaluate the concrete performance of NanoGRAM and compare it with a couple baselines that are asymptotically less efficient. We show that NanoGRAM starts to outperform the naïve linear-scan garbled RAM at a memory size of $N = 2^9$ and starts to outperform the recent construction of Heath et al. at $N = 2^{13}$.

Finally, as a by product, we also show the existence of a garbled RAM scheme assuming only

Garbled RAM constructions were monolithic

Incorporate gate garbling, algorithms, Oblivious RAM

Difficult to Improve

$$\begin{tikzcd}\text{Garbled TSCs} & \text{TSCs} \arrow[l] & \text{RAM} \arrow[l]\end{tikzcd}$$

$$\begin{tikzcd}\text{Garbled TSCs} & \text{TSCs} \arrow[l] & \text{RAM} \arrow[l]\end{tikzcd}$$

Detailed circuit design involving Oblivious RAM (ORAM [GO96])

$$\begin{tikzcd}
\text{Garbled TSCs} & \text{TSCs} \arrow[l] & \text{RAM} \arrow[l]
\end{tikzcd}$$

Basic encrypted truth tables are sufficient

Detailed circuit design involving Oblivious RAM (ORAM [GO96])

$$\begin{tikzcd}\text{Garbled TSCs} & \text{TSCs} \arrow[l] & \text{RAM} \arrow[l]\end{tikzcd}$$

Basic encrypted truth tables are sufficient

(See our paper for details)

Detailed circuit design involving Oblivious RAM (ORAM [GO96])

GRAM Improvements

	Model	Primitive	Size of Garbled Program (bits)
[PLS22] basic	Semi-honest	Random Oracle (CCRH)	$O(T \cdot \log^3 T \cdot \log^2 \log T \cdot \lambda)$
[PLS22] standard assumptions	Semi-honest	One Way Functions	$O(T \cdot \log^3 T \cdot \log^2 \log T \cdot \lambda^2)$
[PLS22] with cut & choose	Malicious	Random Oracle (CCRH)	$O(T \cdot \log^3 T \cdot \log^2 \log T \cdot \lambda \cdot \sigma)$

T - runtime of the RAM program, size of memory is $O(T)$

λ - computational security parameter

σ - statistical security parameter

All protocols are in OT hybrid model

GRAM Improvements

	Model	Primitive	Size of Garbled Program (bits)
[PLS22] basic	Semi-honest	Random Oracle (CCRH)	$O(T \cdot \log^3 T \cdot \log^2 \log T \cdot \lambda)$
[PLS22] standard assumptions	Semi-honest	One Way Functions	$O(T \cdot \log^3 T \cdot \log^2 \log T \cdot \lambda^2)$
[PLS22] with cut & choose	Malicious	Random Oracle (CCRH)	$O(T \cdot \log^3 T \cdot \log^2 \log T \cdot \lambda \cdot \sigma)$
Ours	Semi-honest	One Way Functions	$O(T \cdot \log^3 T \cdot \log \log T \cdot \lambda)$
Ours	Malicious	Random Oracle	$O(T \cdot \log^3 T \cdot \log \log T \cdot \lambda)$

T - runtime of the RAM program, size of memory is $O(T)$

λ - computational security parameter

σ - statistical security parameter

All protocols are in OT hybrid model

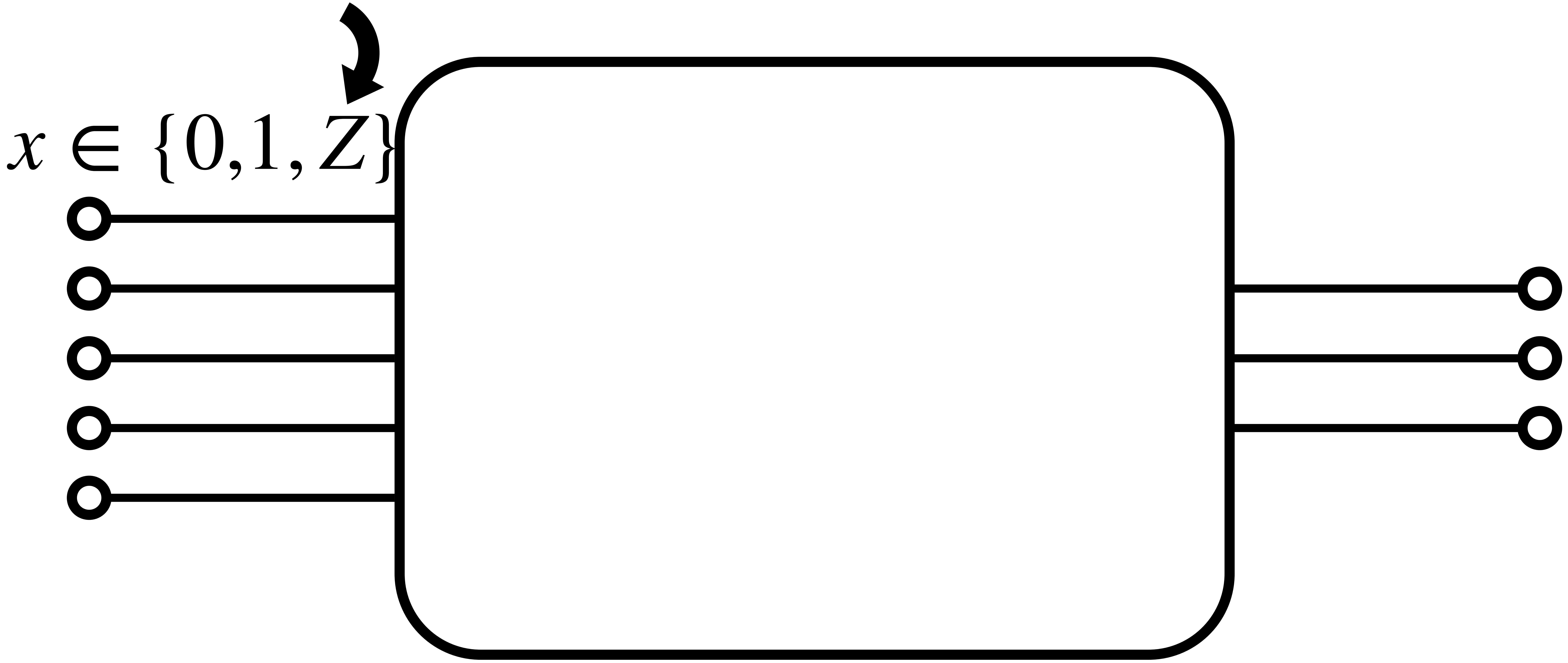
Tri-State Circuits

What is a tri-state circuit?

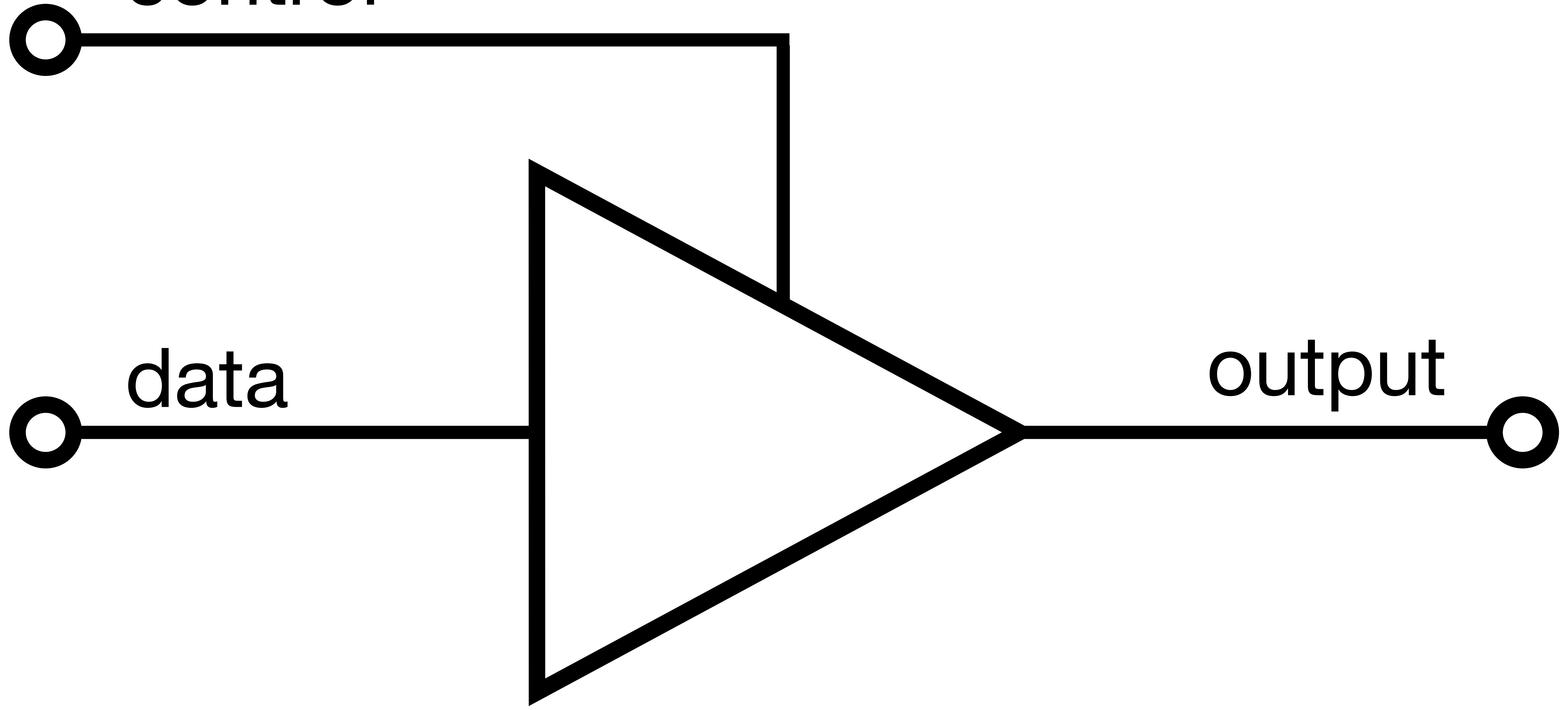
What is a tri-state circuit?

Nil: “this wire does not have a value”

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

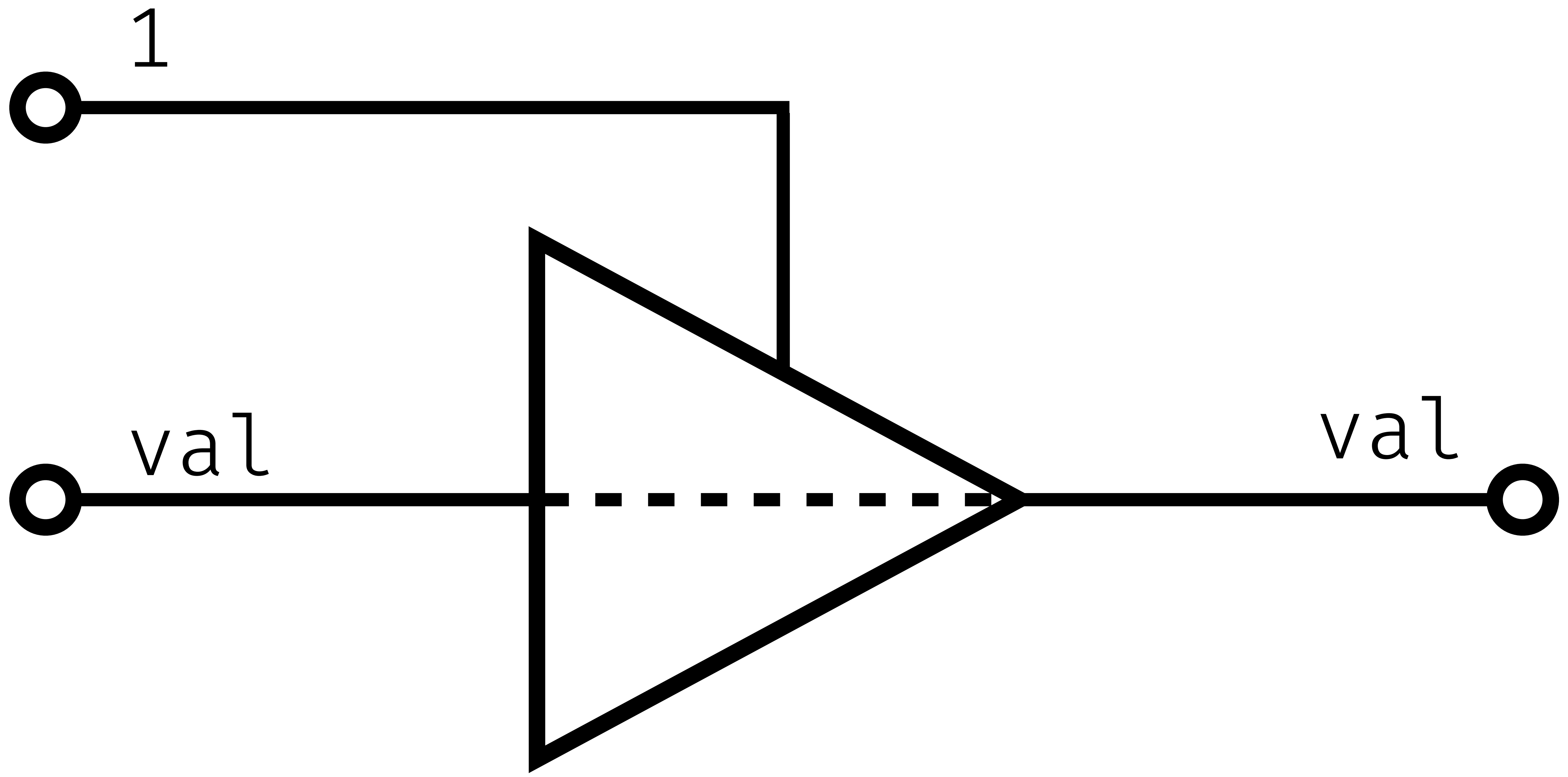


Buffer



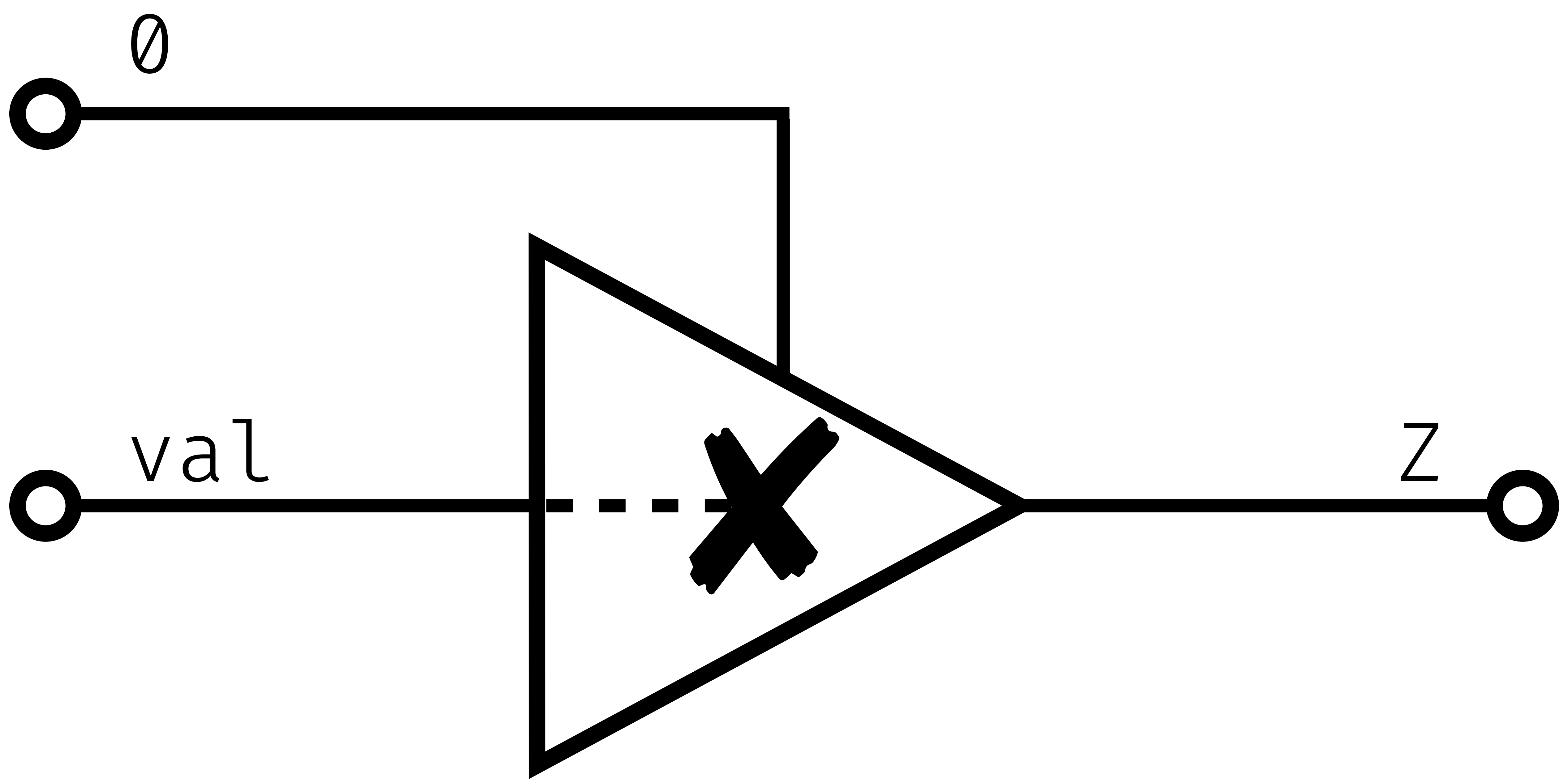
		control		
		0	1	Z
data	0	Z	0	Z
	1	Z	1	Z
	Z	Z	Z	Z

Buffer



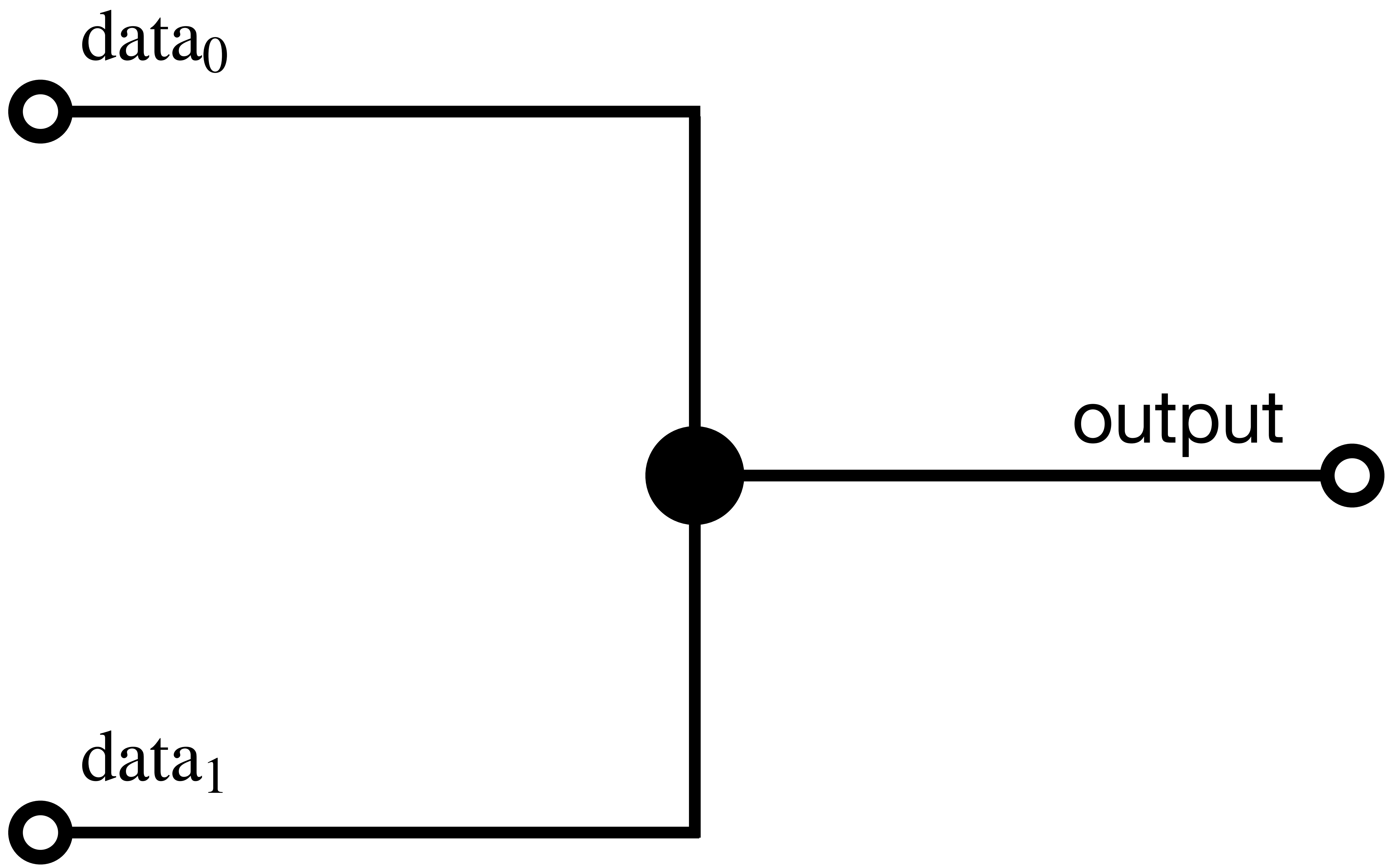
data \ control	0	1	Z
0	Z	0	Z
1	Z	1	Z
Z	Z	Z	Z

Buffer



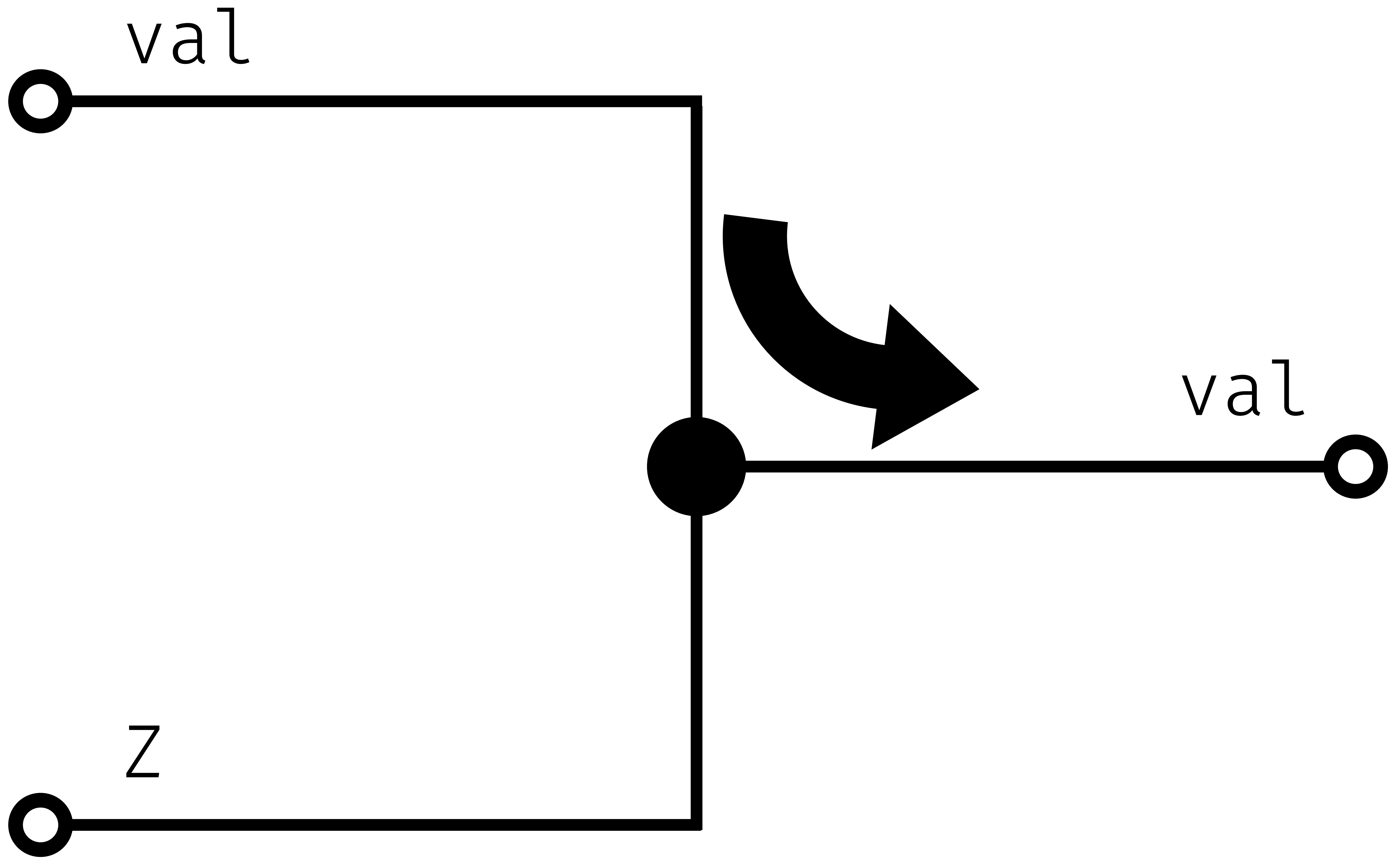
	control 0	control 1	control Z
data 0	Z	0	Z
data 1	Z	1	Z
data Z	Z	Z	Z

Join



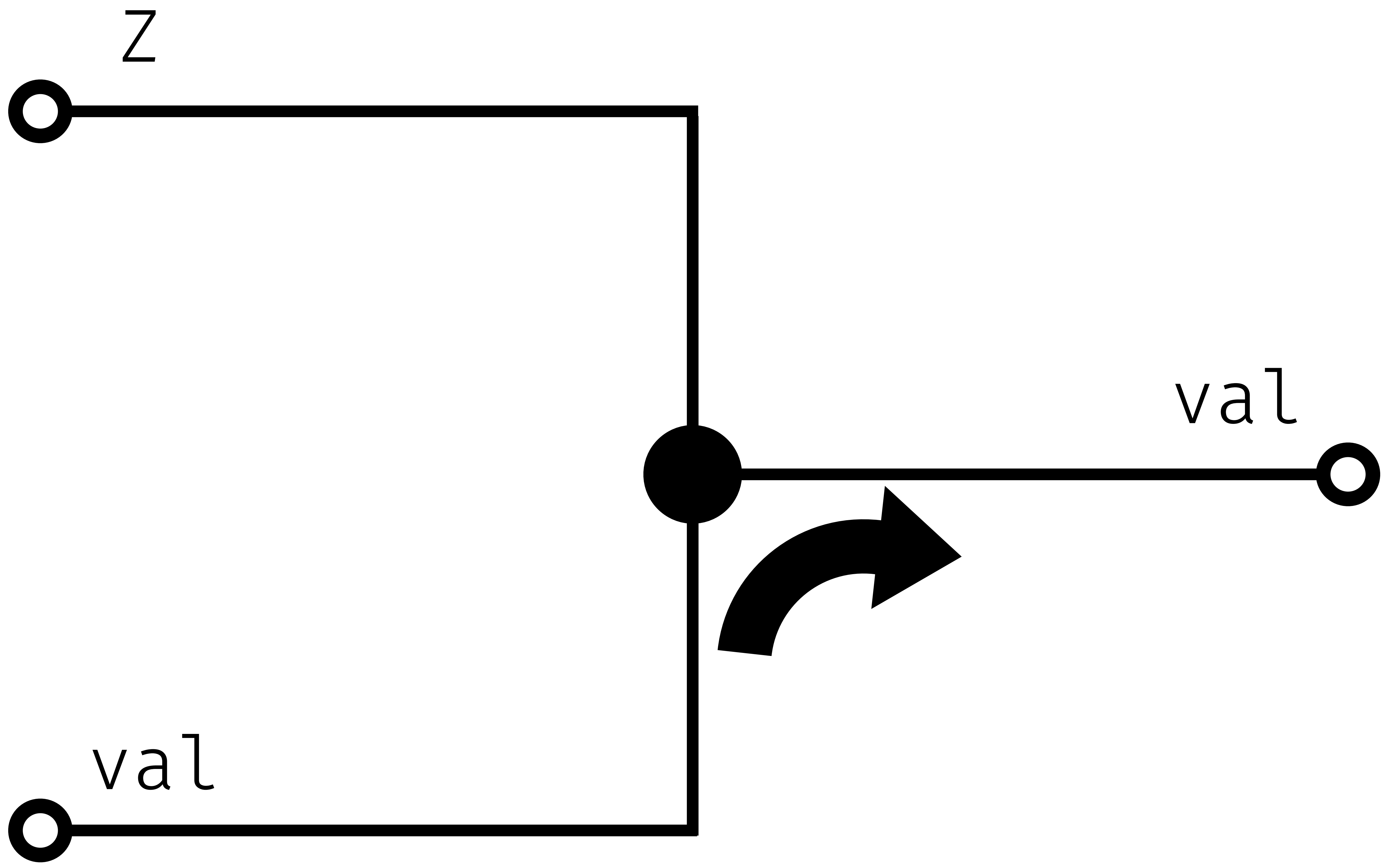
	data_1		
data_0	`0`	`1`	`Z`
`0`	`0`	$\perp$	`0`
`1`	$\perp$	`1`	`1`
`Z`	`0`	`1`	`Z`

Join



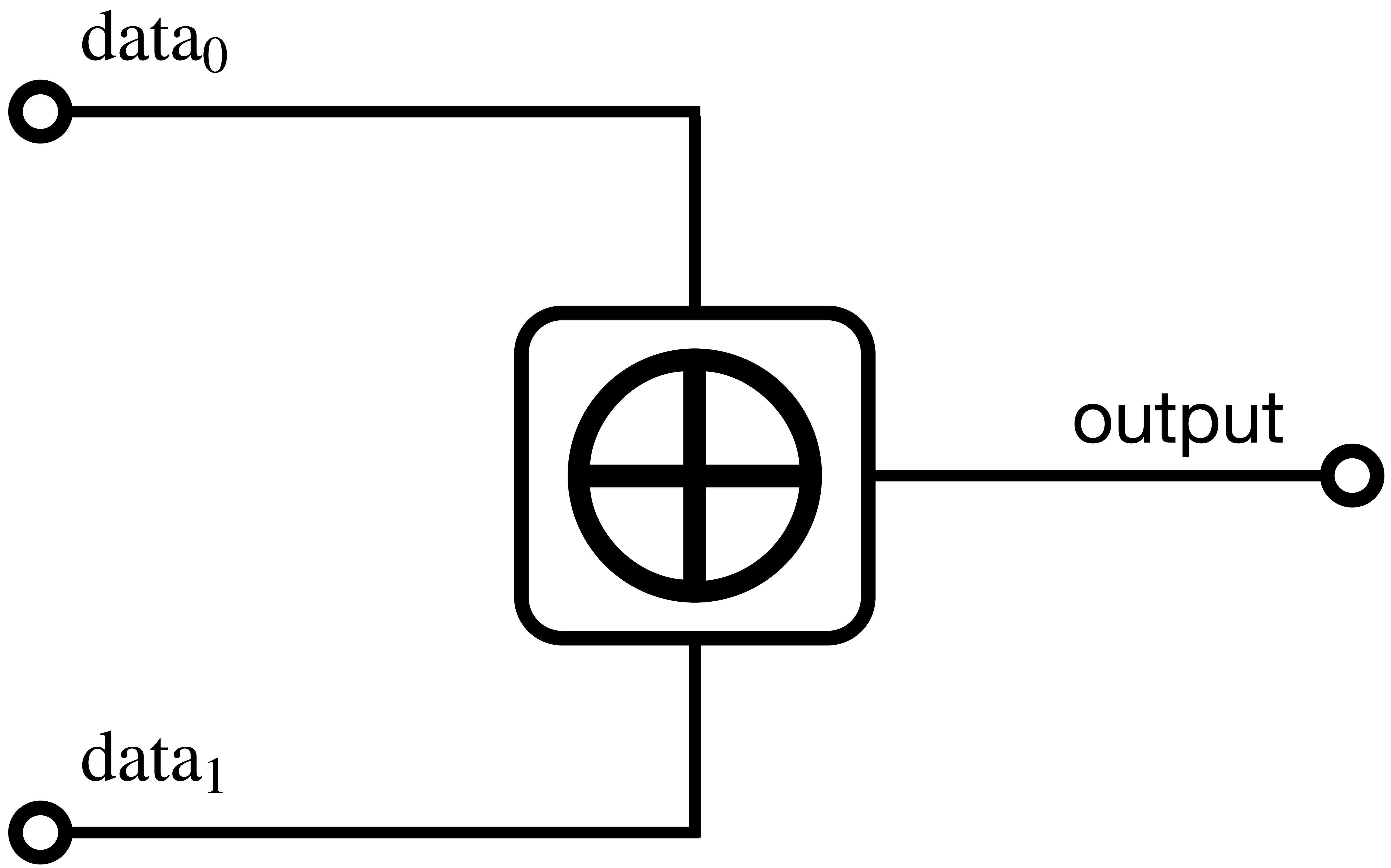
data_0 \ data_1	0	1	Z
0	0	⊥	0
1	⊥	1	1
Z	0	1	Z

Join



		data$_1$		
		0	1	Z
data$_0$	0	0	⊥	0
	1	⊥	1	1
	Z	0	1	Z

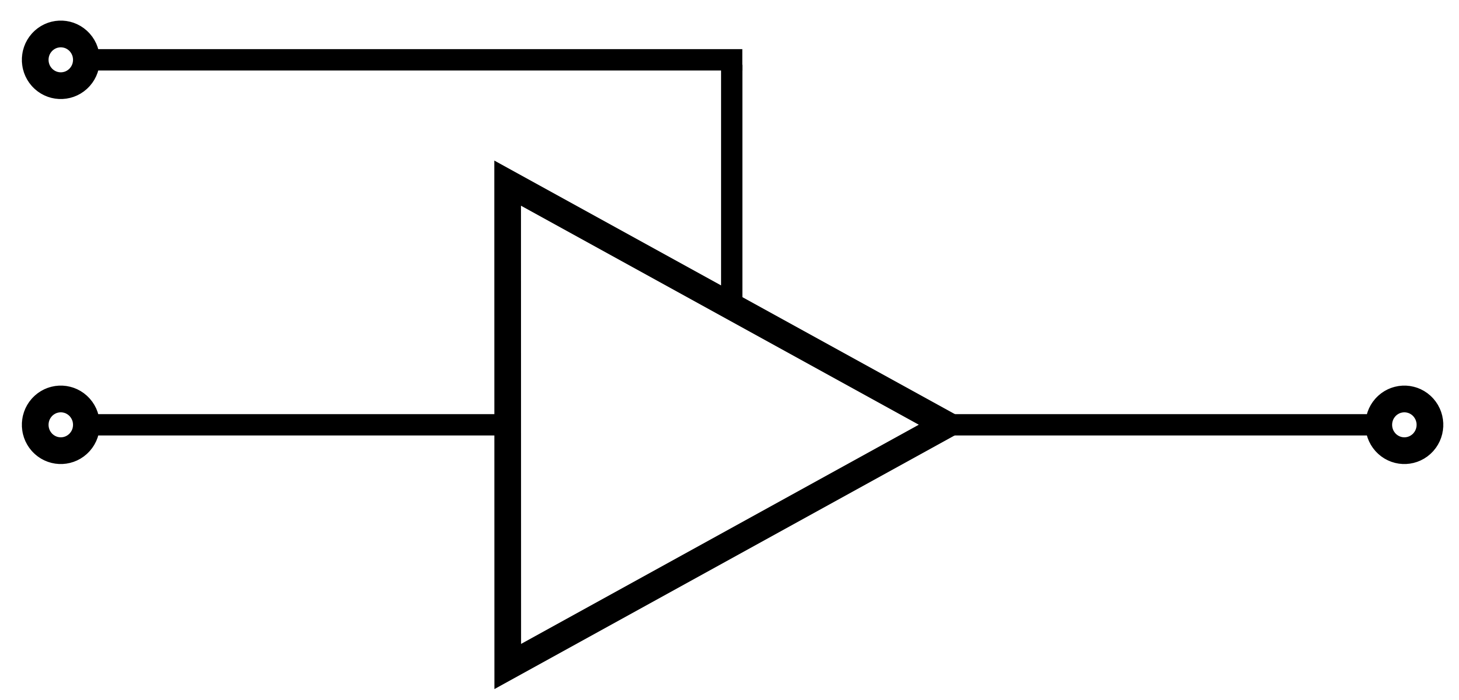
XOR

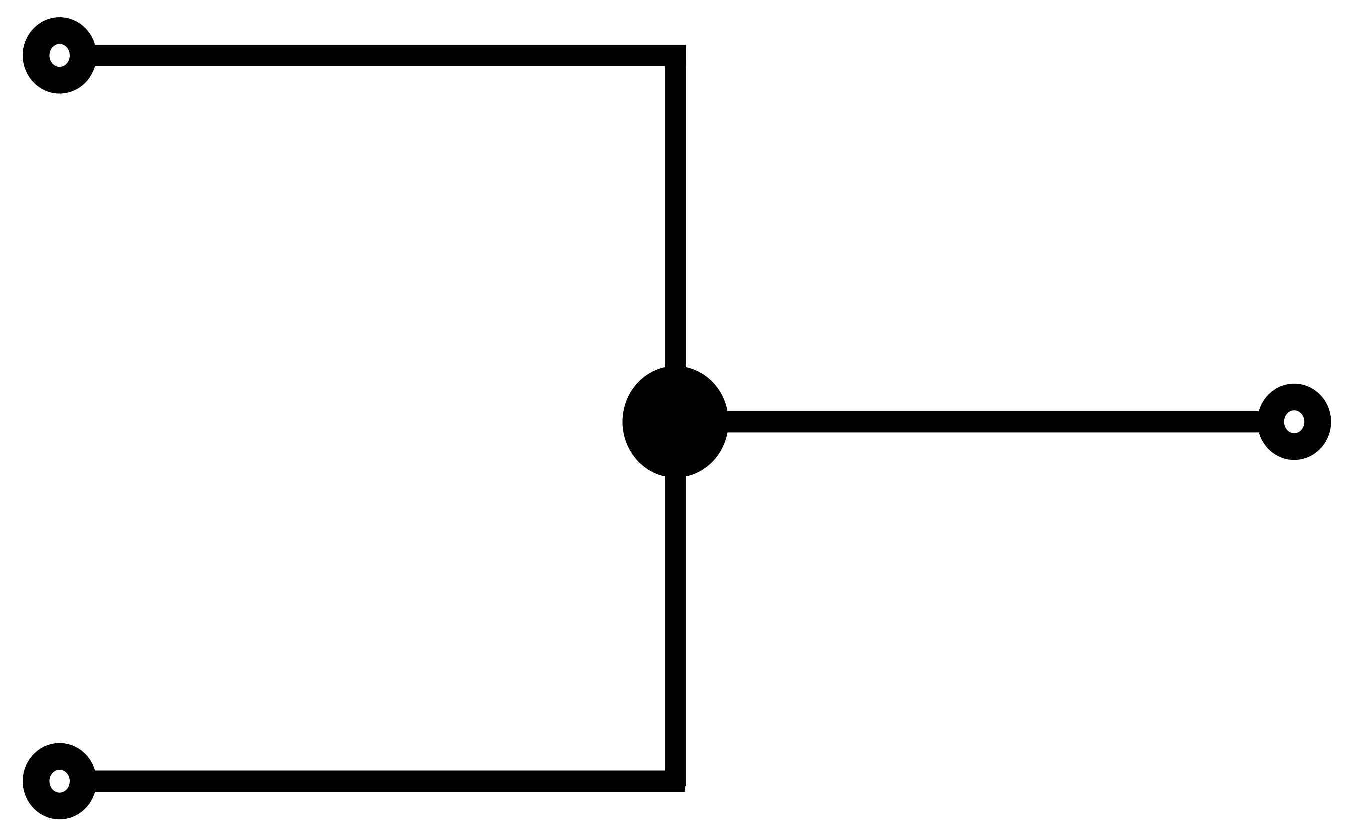


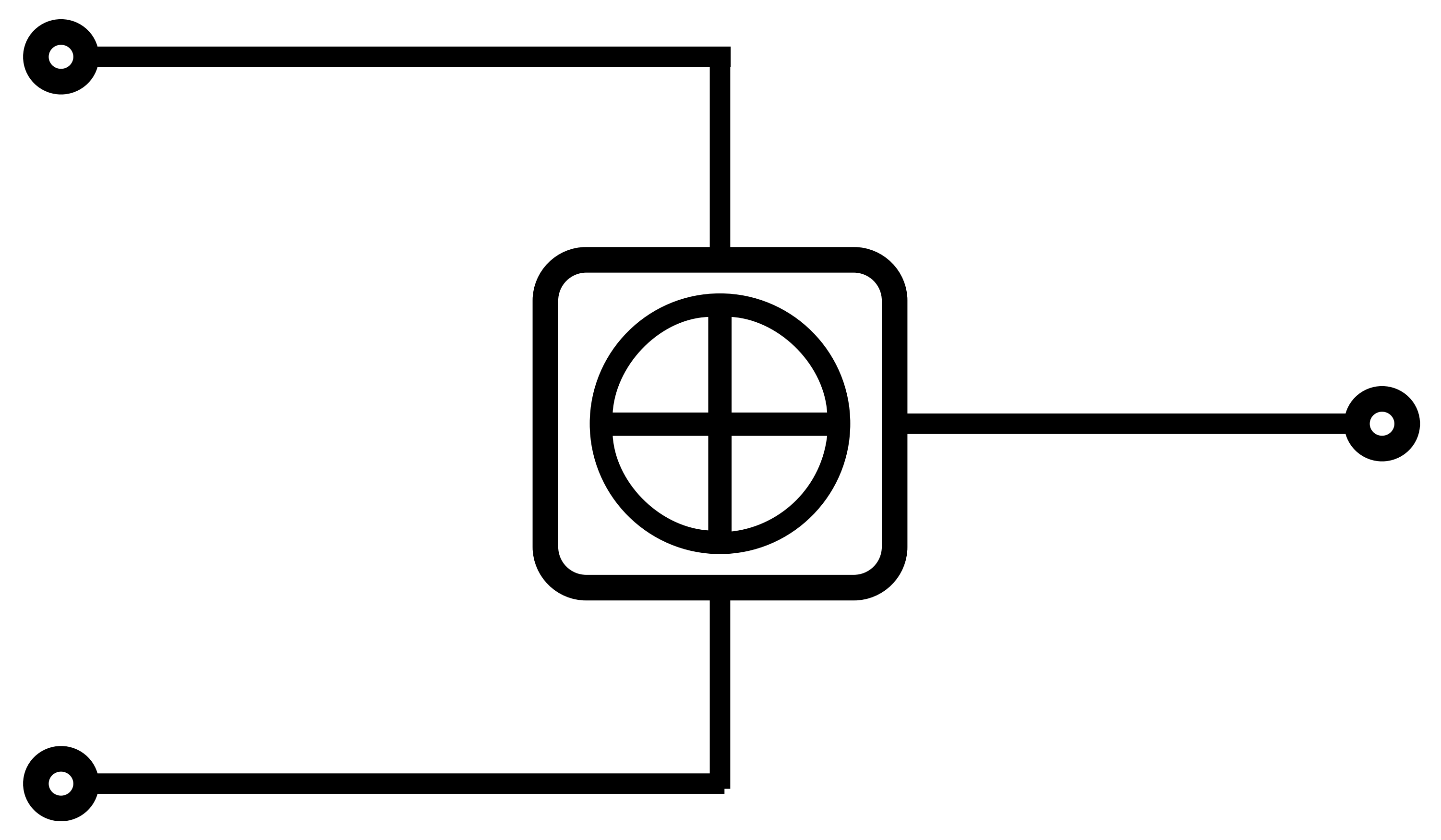
data_1 across the columns, data_0 down the rows:

	0	1	Z
0	0	1	Z
1	1	0	Z
Z	Z	Z	Z

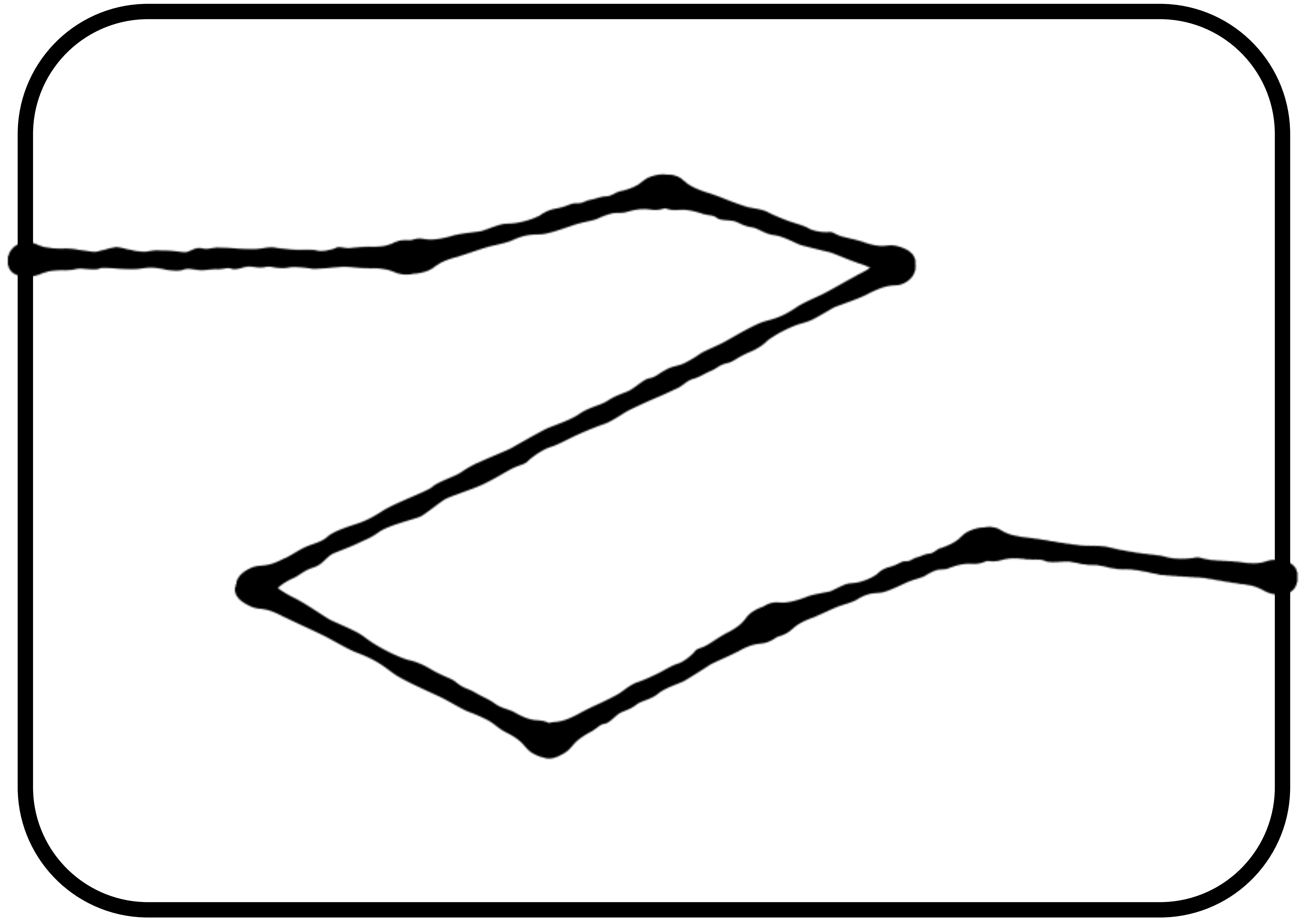
Definition. *A tri-state circuit is a circuit composed from buffers, joins, and XORs.* ***Tri-state circuits allow cycles in their circuit graph.***

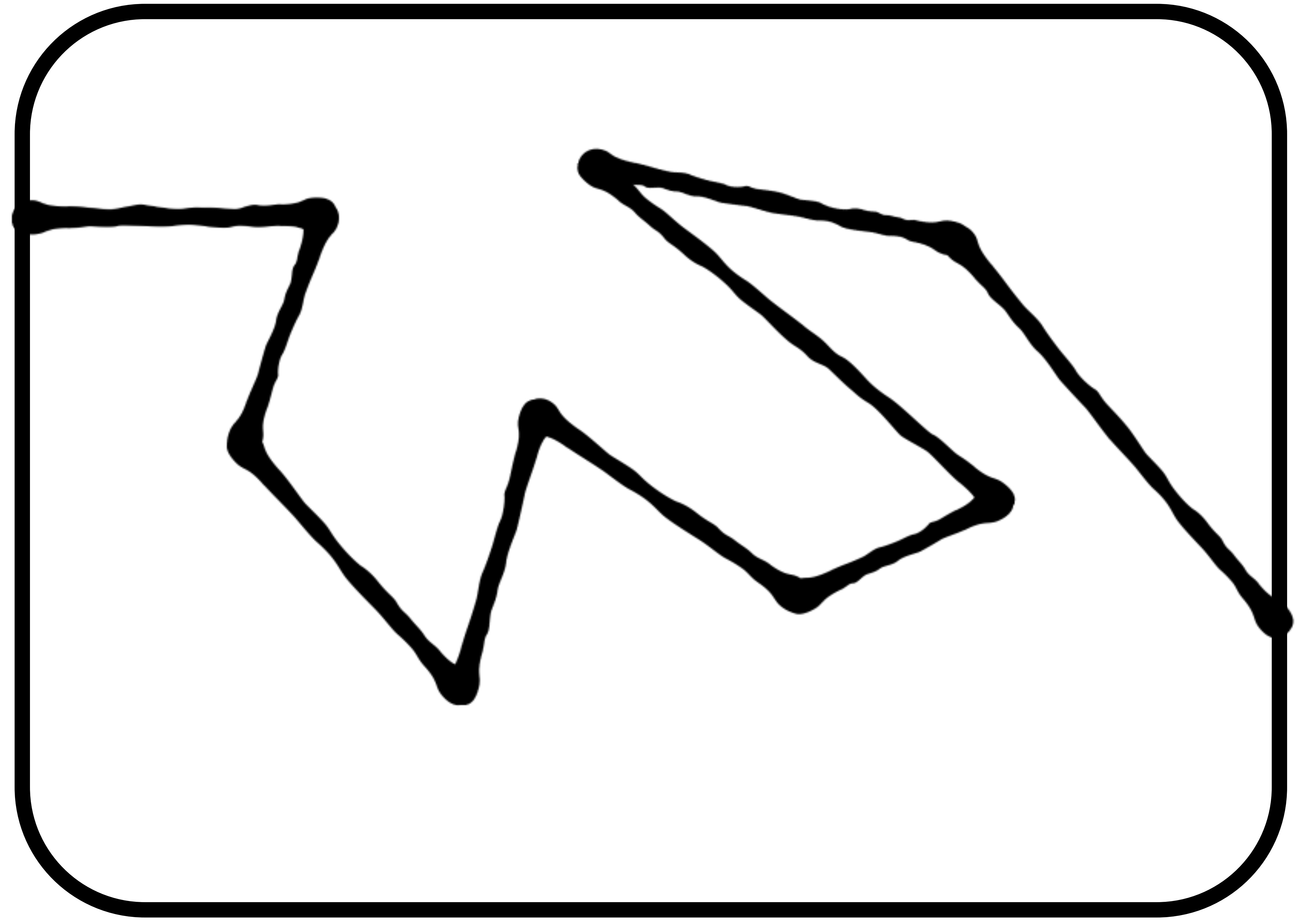






Order of gate execution *depends on the input*



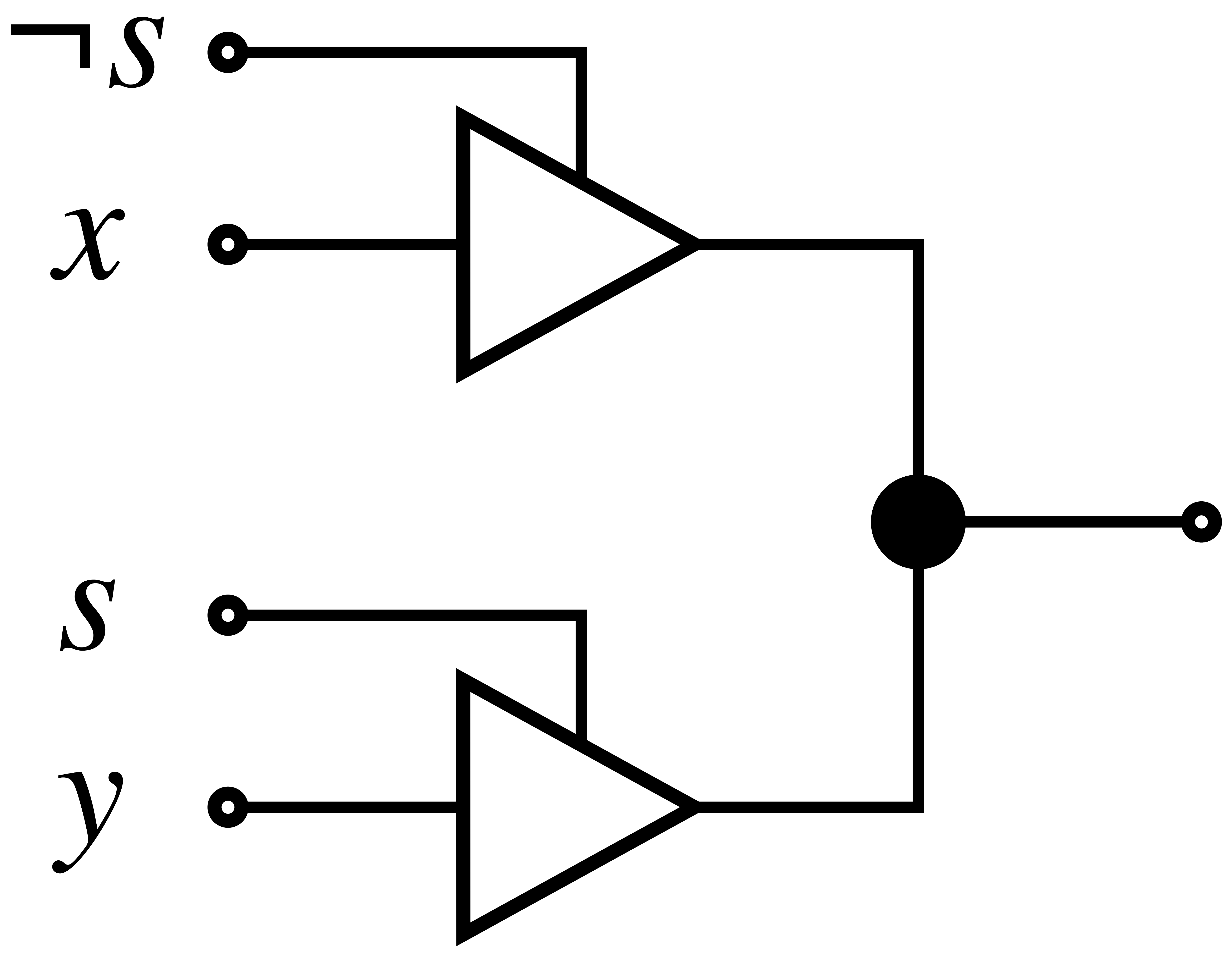


Primitive form of control flow

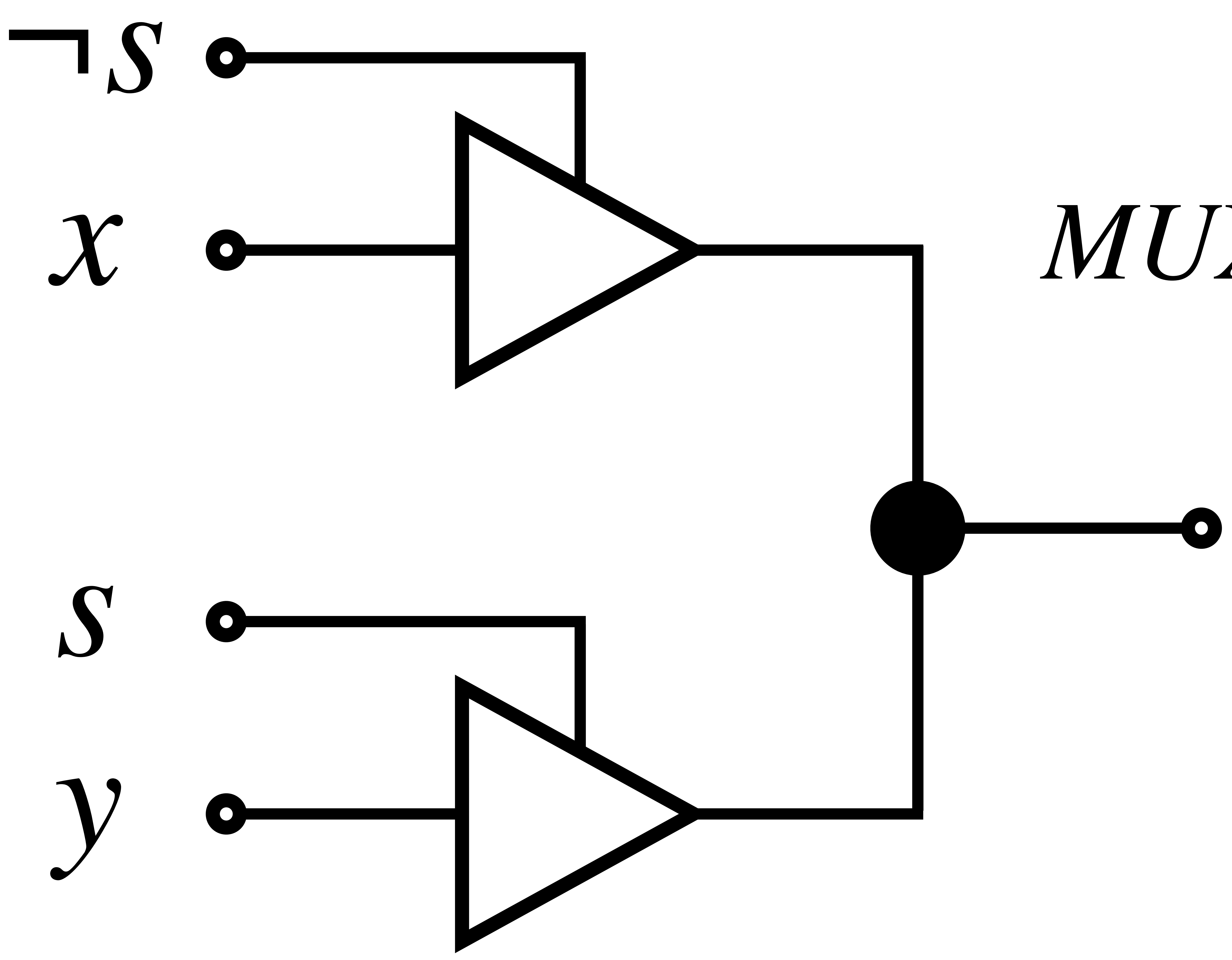
Enough control to efficiently implement random access memory

How to emulate RAM with TSCs

MUX

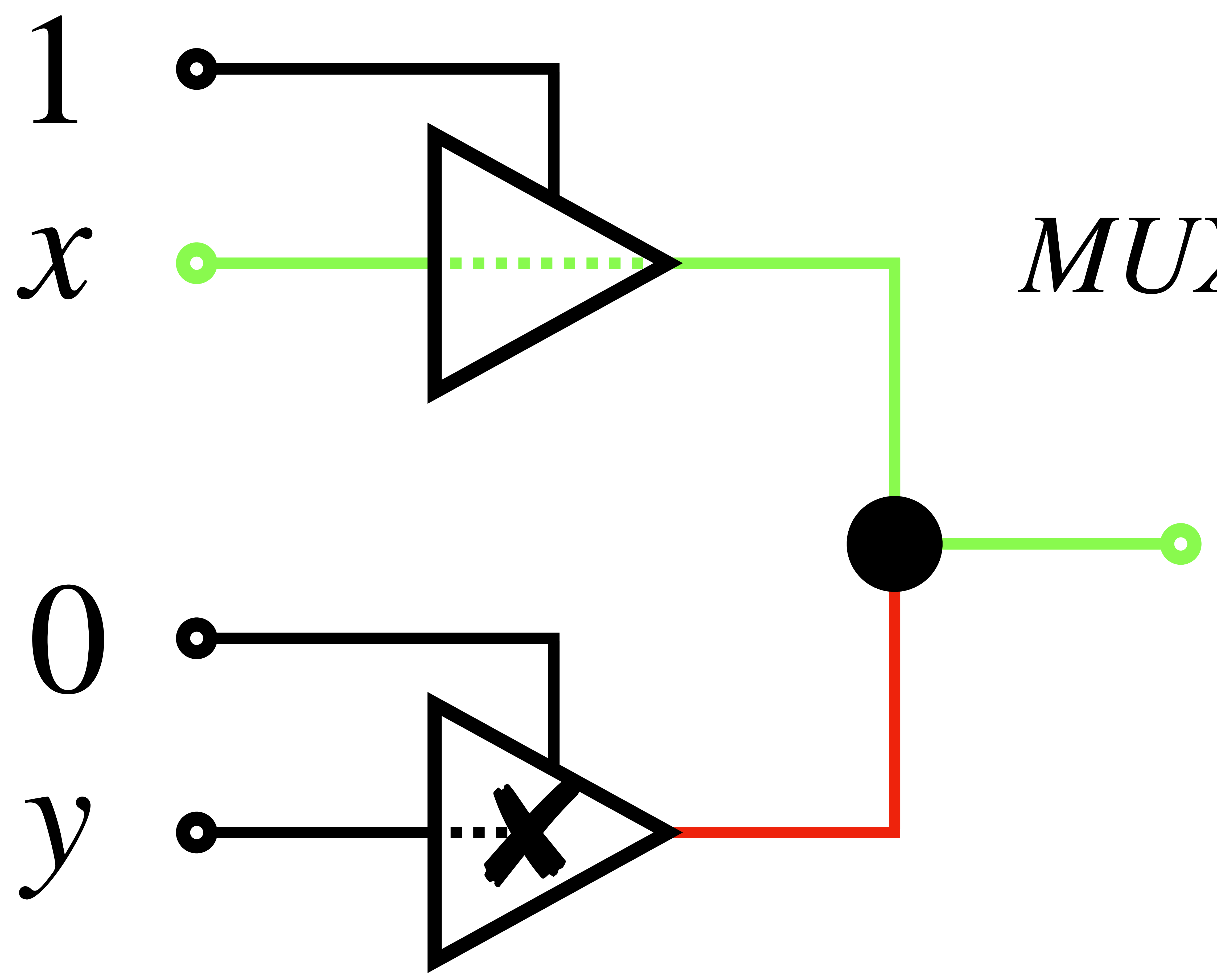


MUX



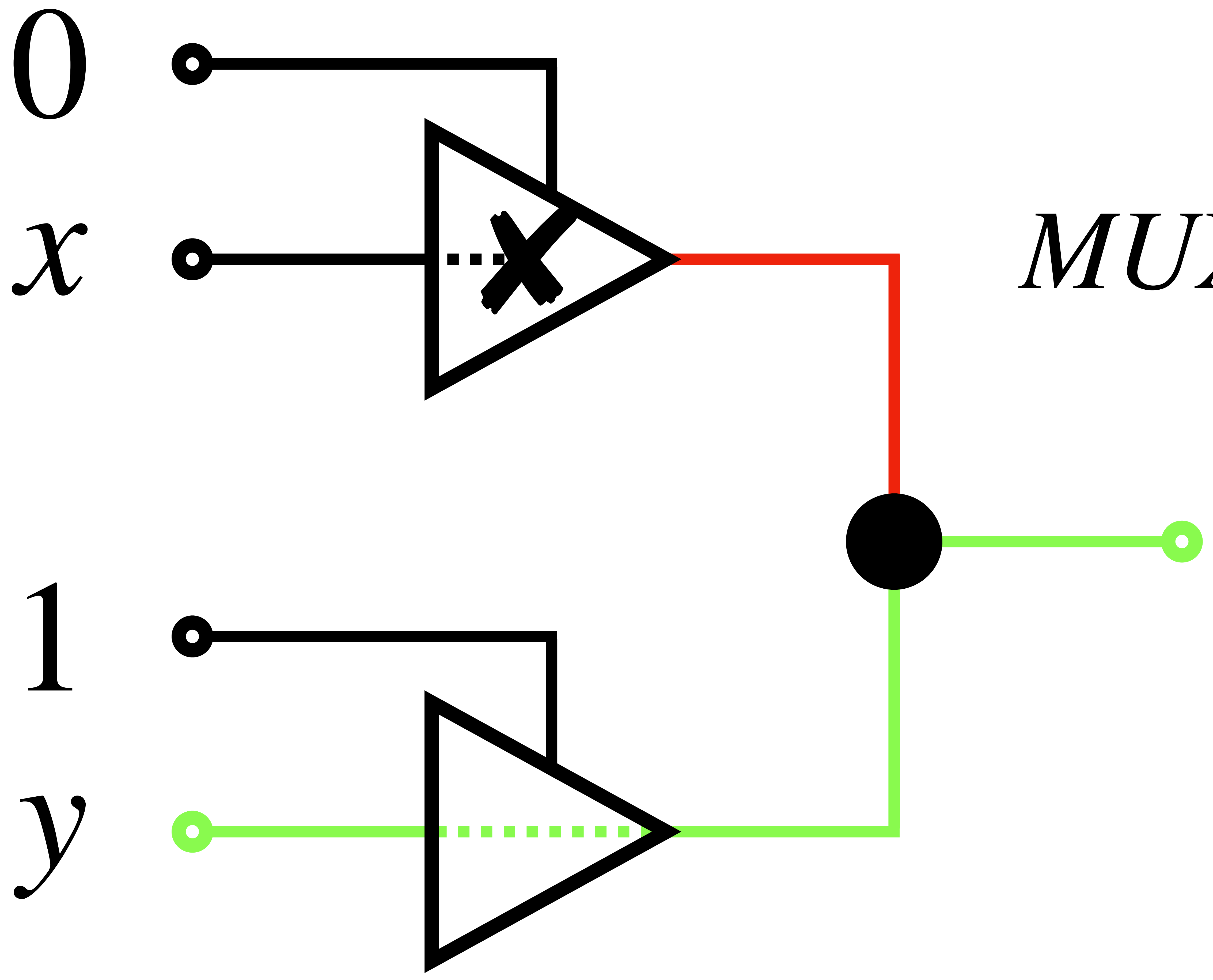
$$MUX(s,x,y) = \begin{cases} x & if\ s = 0 \\ y & if\ s = 1 \\ Z & if\ s = Z \end{cases}$$

MUX



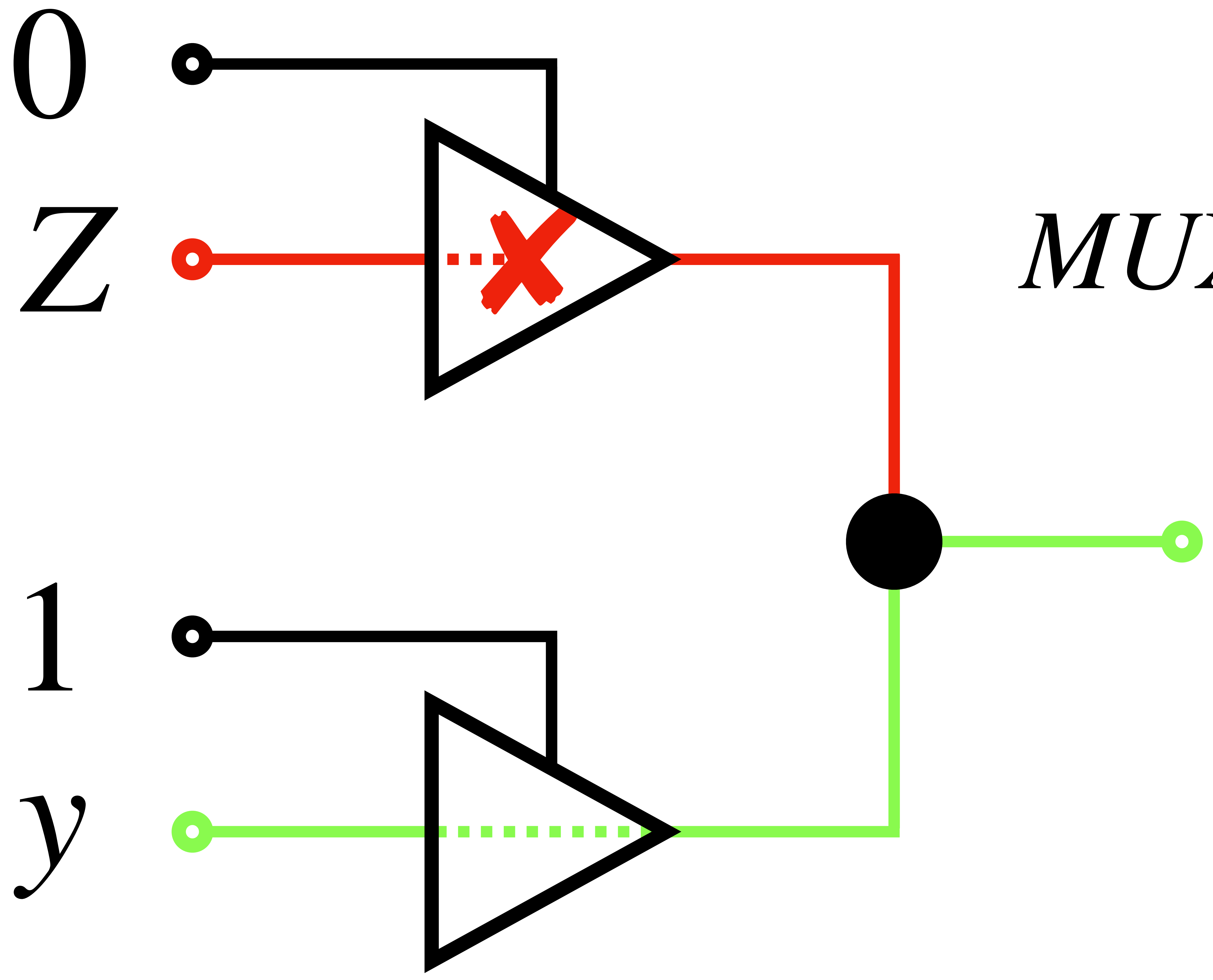
$$MUX(s,x,y) = \begin{cases} x & \text{if } s = 0 \\ y & \text{if } s = 1 \\ Z & \text{if } s = Z \end{cases}$$

MUX



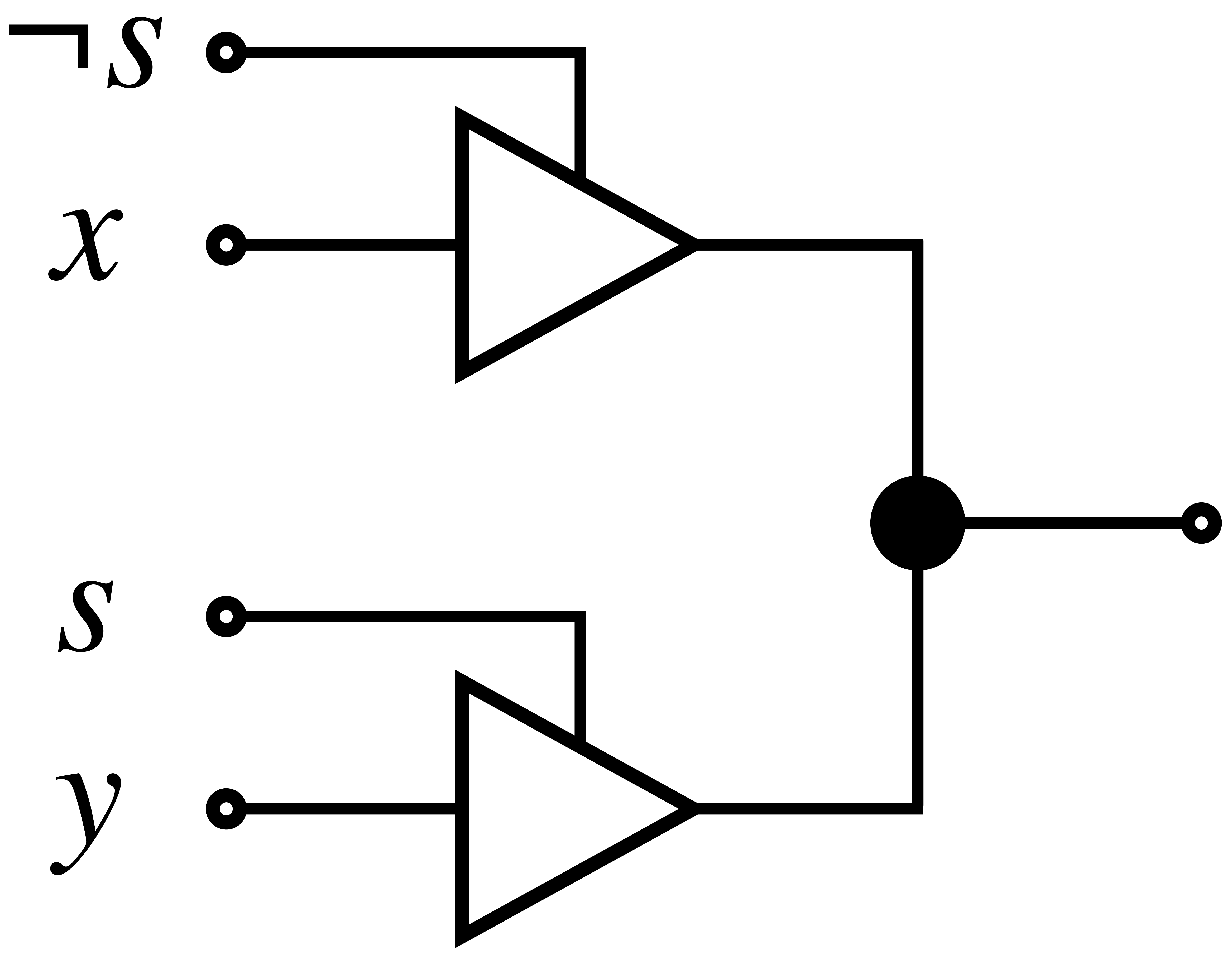
$$MUX(s, x, y) = \begin{cases} x & if\ s = 0 \\ y & if\ s = 1 \\ Z & if\ s = Z \end{cases}$$

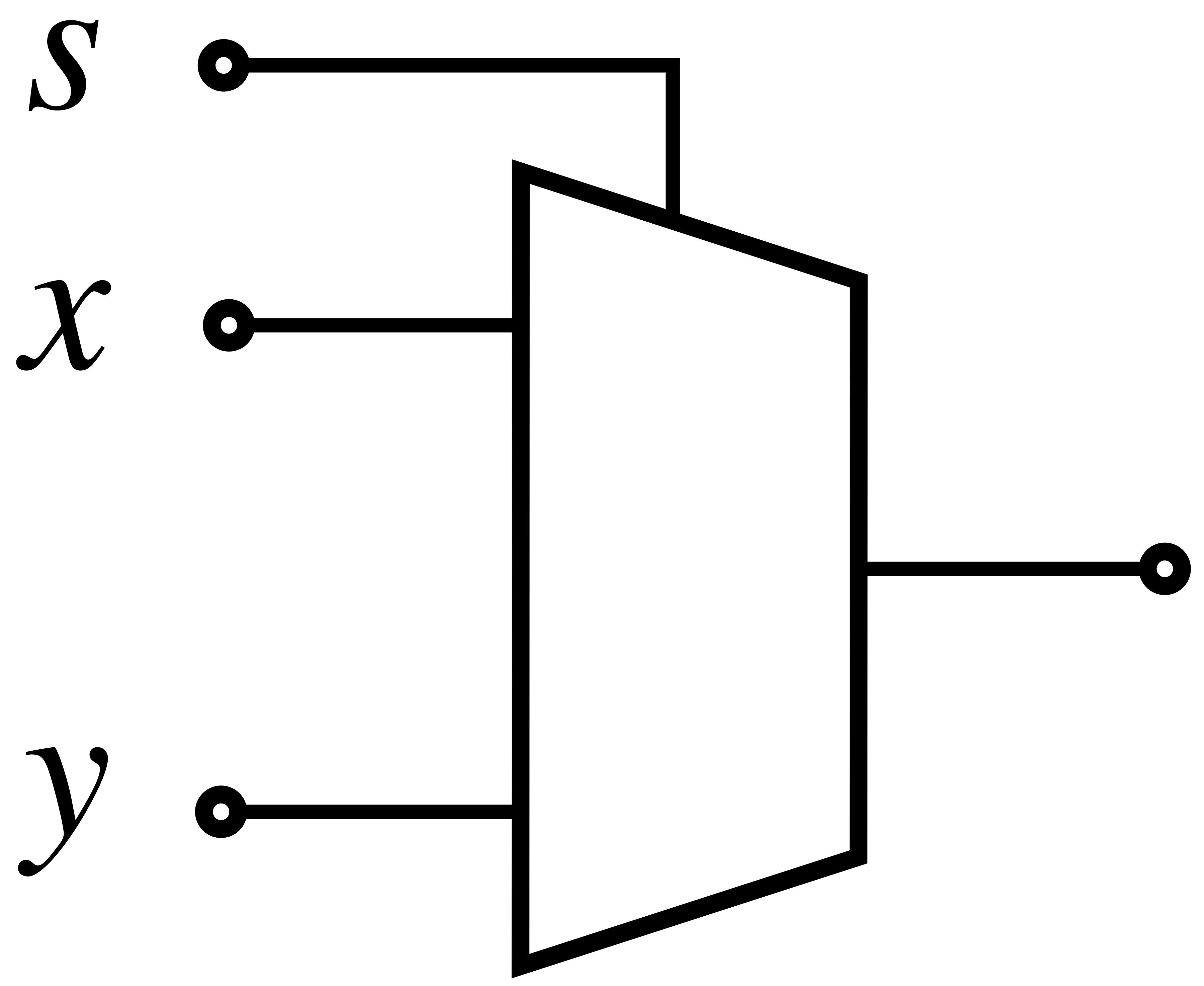
MUX



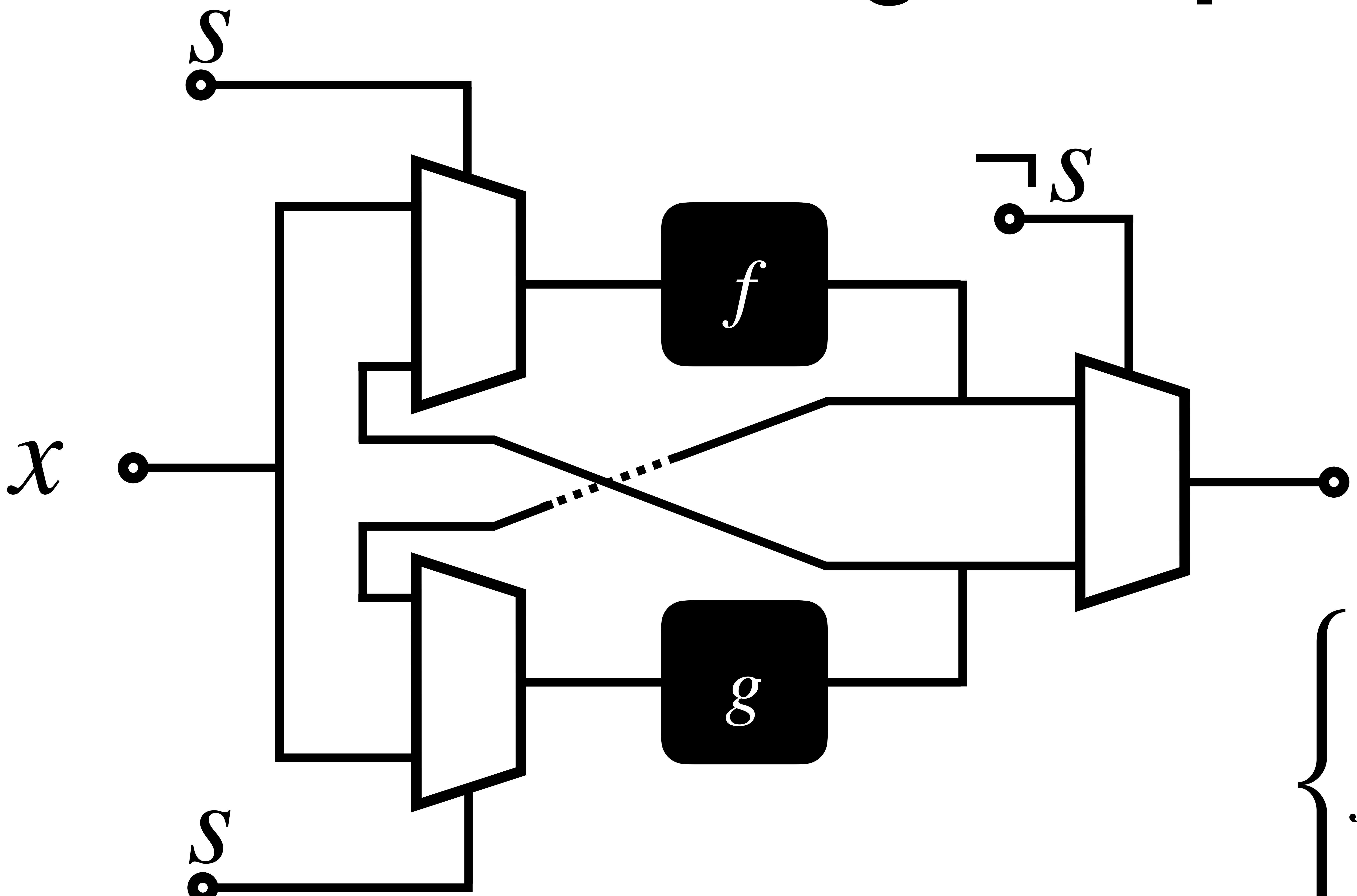
$$MUX(s, x, y) = \begin{cases} x & if\ s = 0 \\ y & if\ s = 1 \\ Z & if\ s = Z \end{cases}$$

MUX



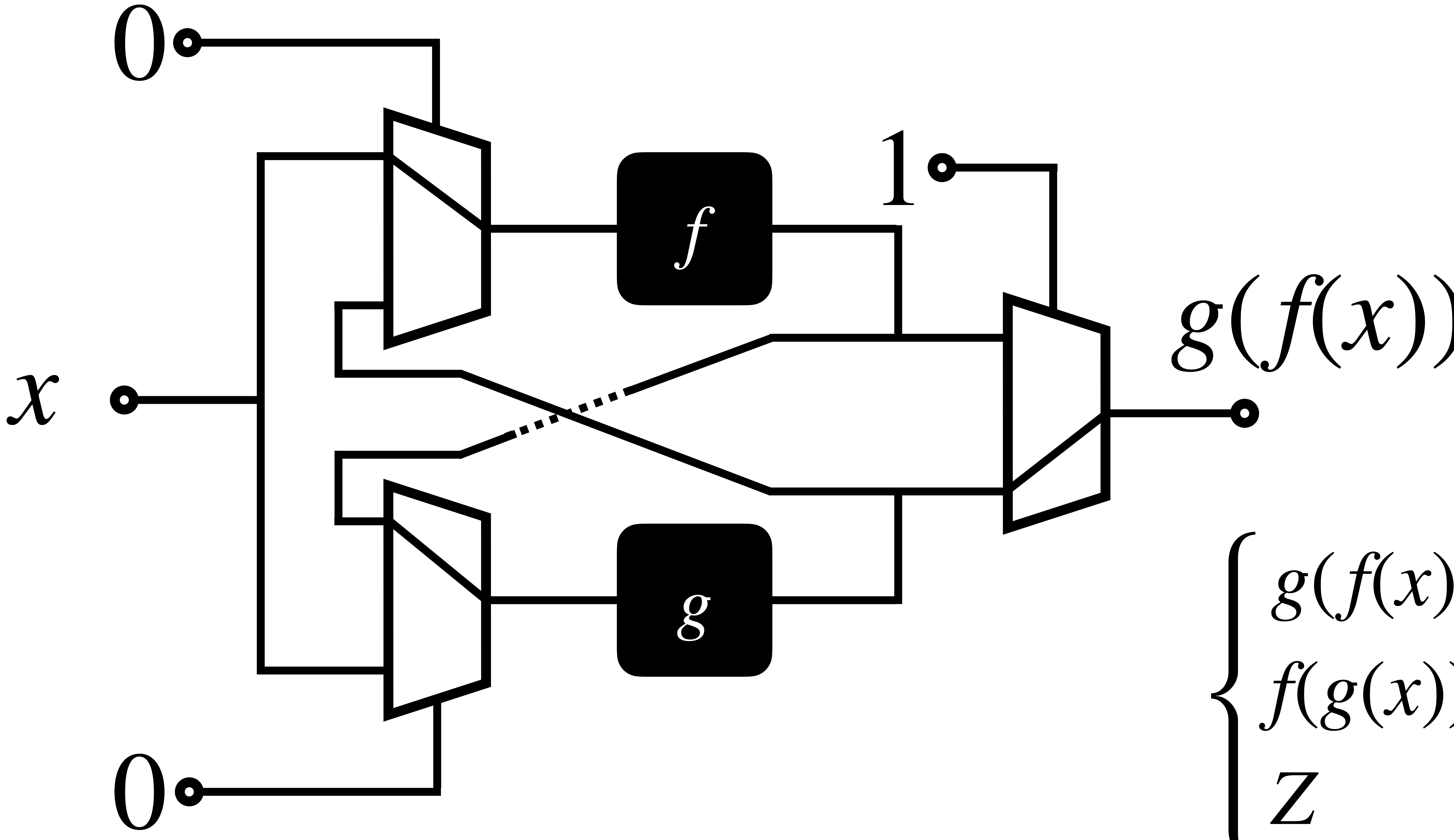


Reordering Components



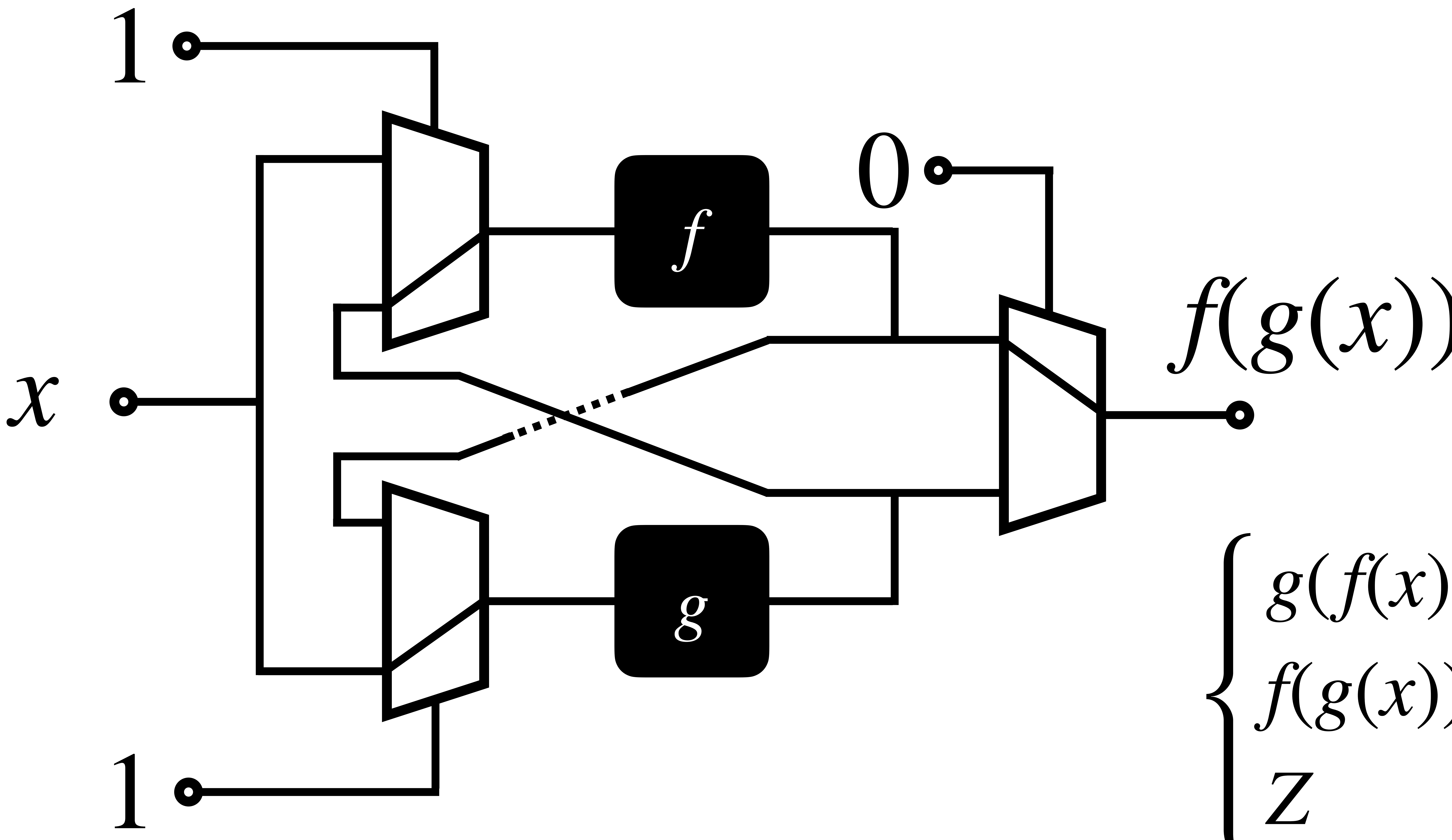
$$\begin{cases} g(f(x)) & if\ s = 0 \\ f(g(x)) & if\ s = 1 \\ Z & if\ s = Z \end{cases}$$

Reordering Components



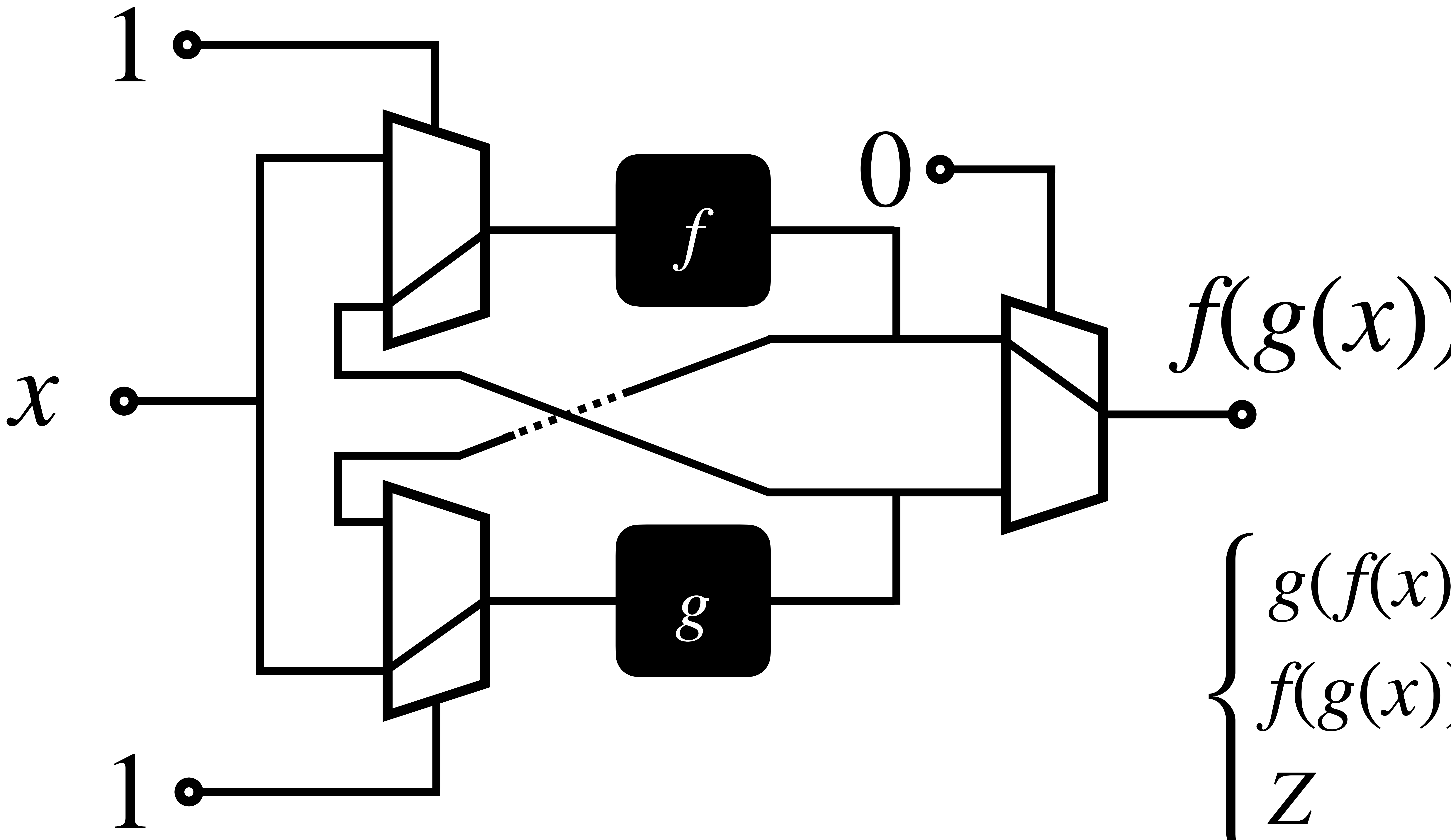
$$\begin{cases} g(f(x)) & if\ s = 0 \\ f(g(x)) & if\ s = 1 \\ Z & if\ s = Z \end{cases}$$

Reordering Components

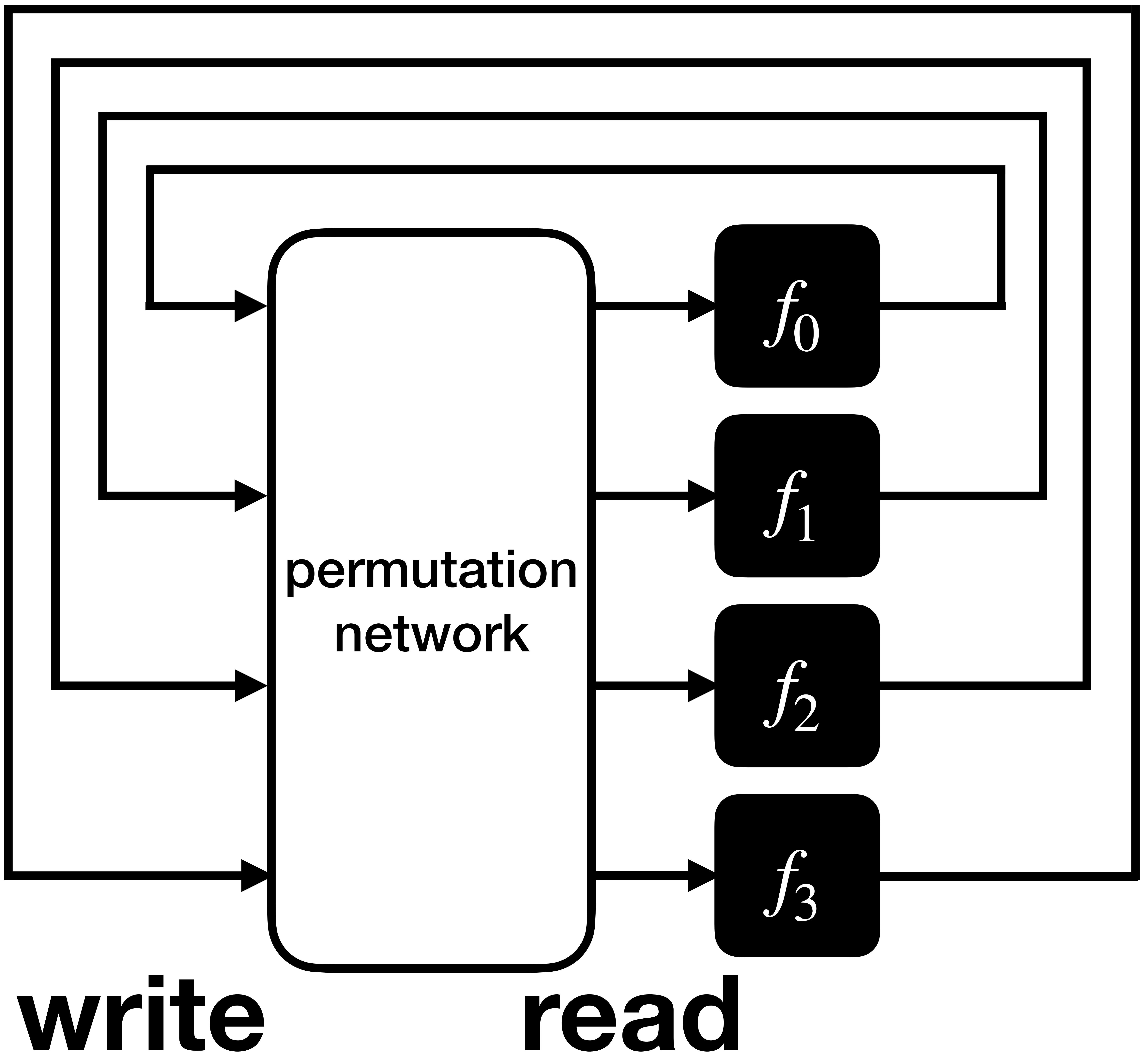


$$\begin{cases} g(f(x)) & if\ s = 0 \\ f(g(x)) & if\ s = 1 \\ Z & if\ s = Z \end{cases}$$

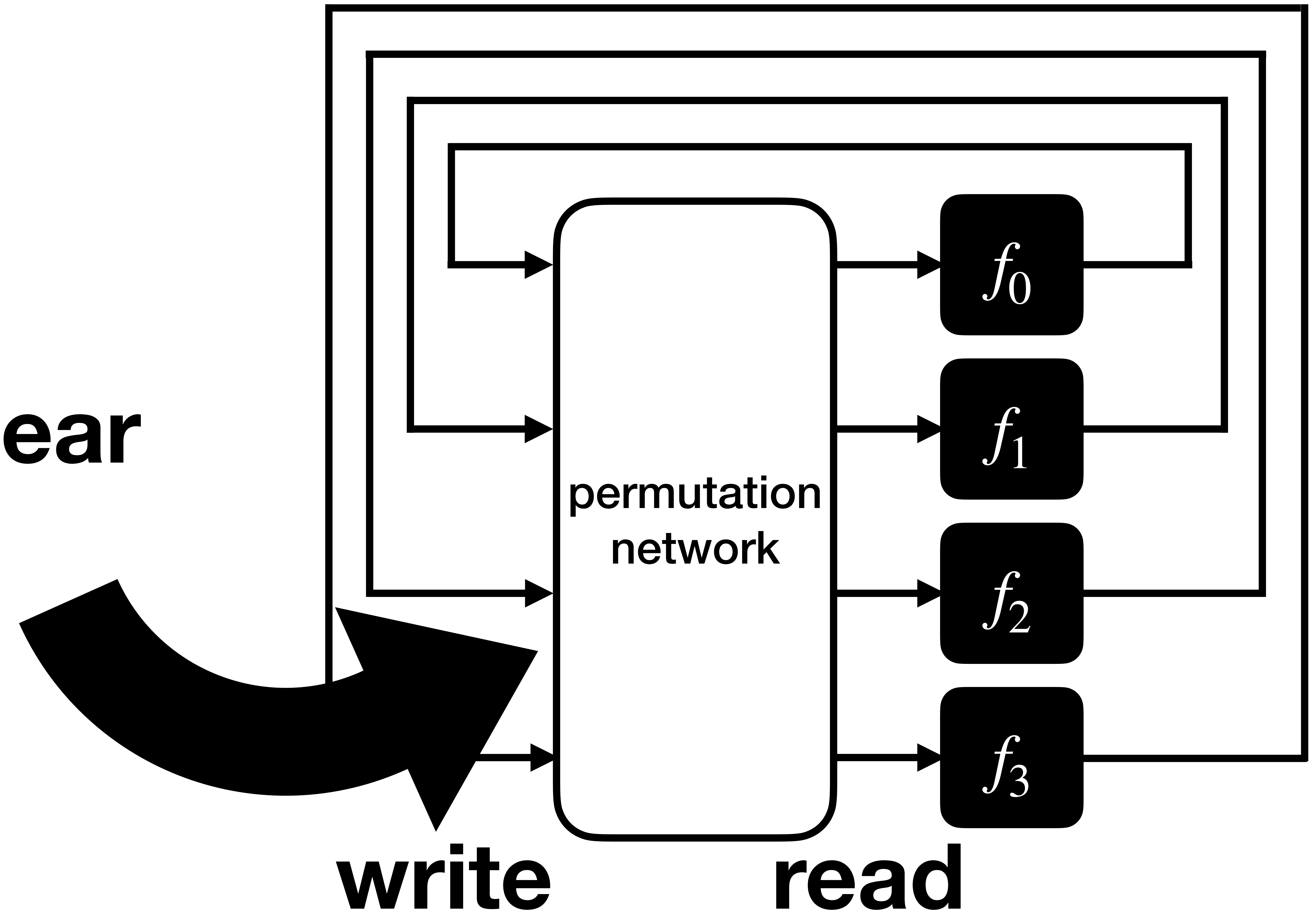
Reordering Components



$$\begin{cases} g(f(x)) & \textit{if } s = 0 \\ f(g(x)) & \textit{if } s = 1 \\ Z & \textit{if } s = Z \end{cases}$$


permutation network
f_0
f_1
f_2
f_3
write
read

Quasilinear Size!



See our paper for

Oblivious TSC definition

Full RAM construction

Semi-honest garbling of TSCs

Malicious garbling of TSCs

Contributions

Tri-State circuits (TSCs): adds lightweight "control flow" to Boolean circuits

A quasilinear TSC that emulates RAM

State of the art improvements to GRAM