Boolean Functions on disjoint Sets of Variables

A glimpse at complexity theory

Theoretical Computer Science asks

- what can be computed?
- what cannot be computed?
- how do you prove, that a function is not computable (recursion theory)
- what is the best (cheapest/fastest) way to compute a function
    - upper bounds: efficient algorithms
    - lower bounds: complexity theory

Theoretical Computer Science asks

- what can be computed?
- what cannot be computed?
- how do you prove, that a function is not computable (recursion theory)
- what is the best (cheapest/fastest) way to compute a function
    - upper bounds: efficient algorithms
    - lower bounds: complexity theory

here: circuit complexity

- how to find optimal circuit constructions with respect to
    - number of gates
    - depth

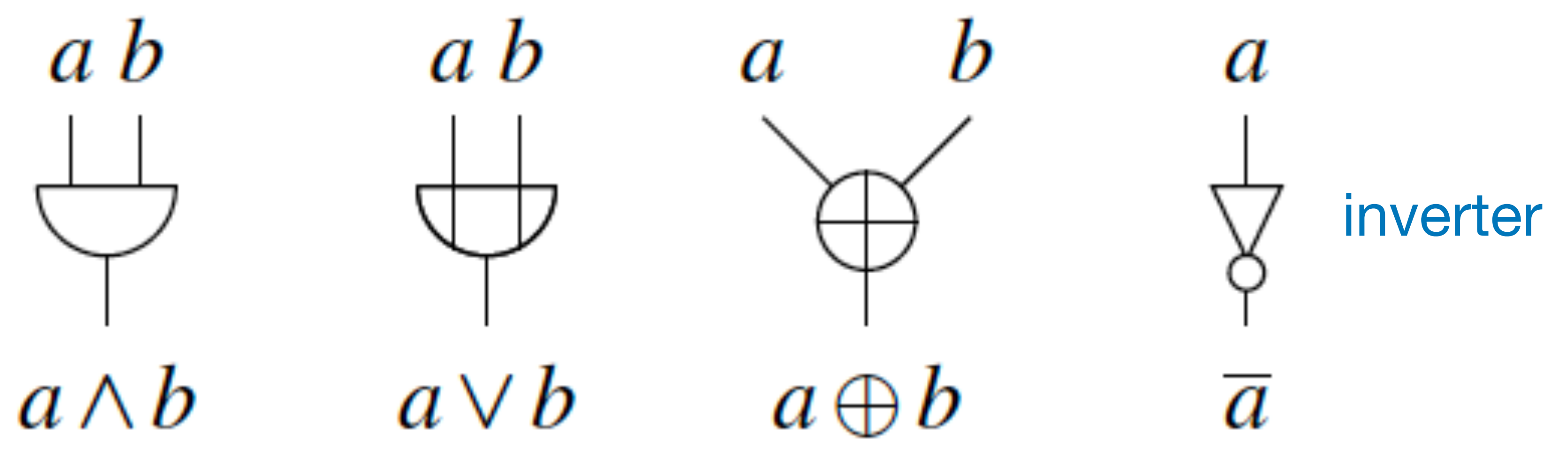
Theoretical Computer Science asks

- what can be computed?
- what cannot be computed?
- how do you prove, that a function is not computable (recursion theory)
- what is the best (cheapest/fastest) way to compute a function
  - upper bounds: efficient algorithms
  - lower bounds: complexity theory

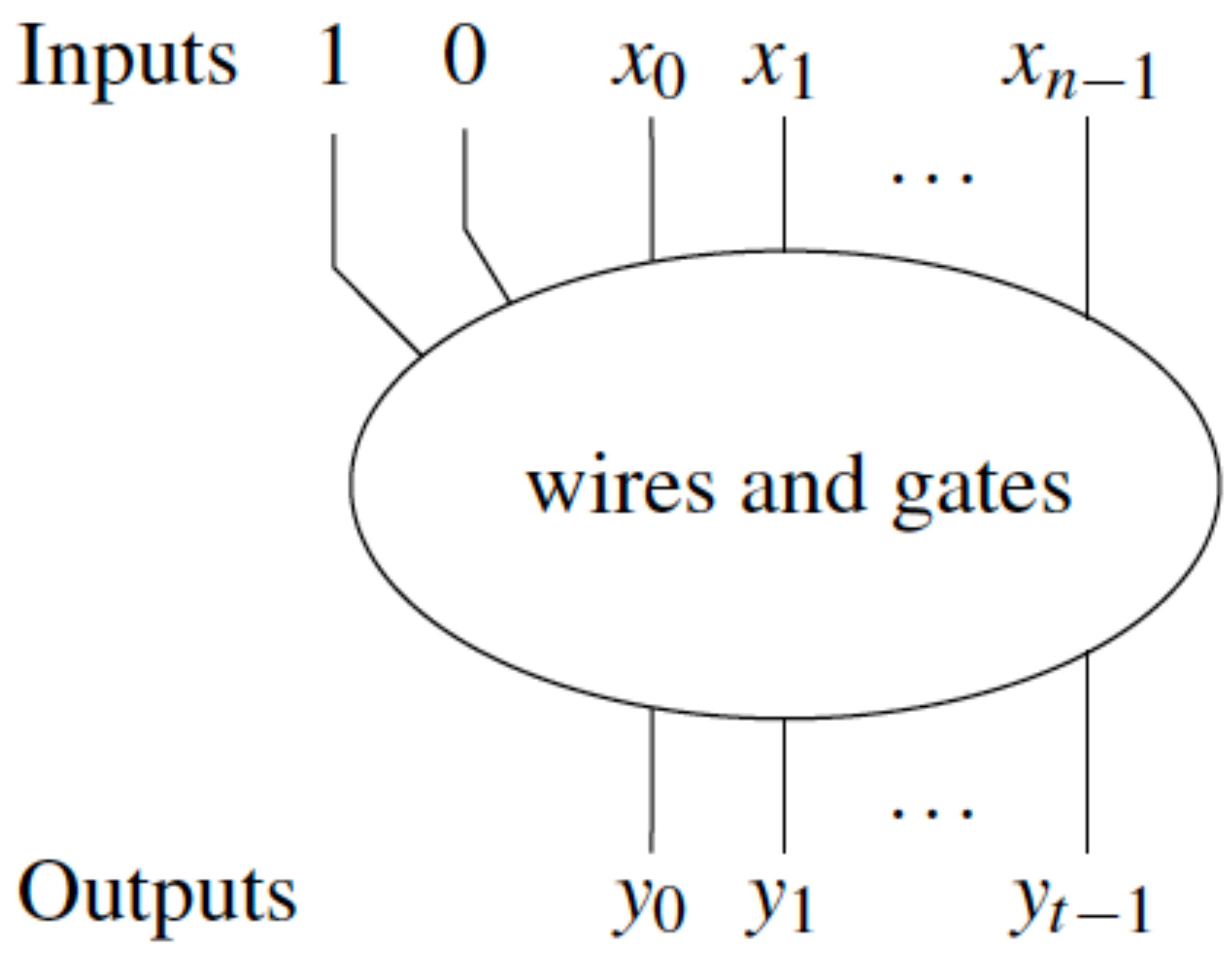
here: circuit complexity
- how to find optimal circuit constructions with respect to
  - number of gates
  - depth

the short answer
- as of today we can’t
- even the simplest problems have very mean solutions

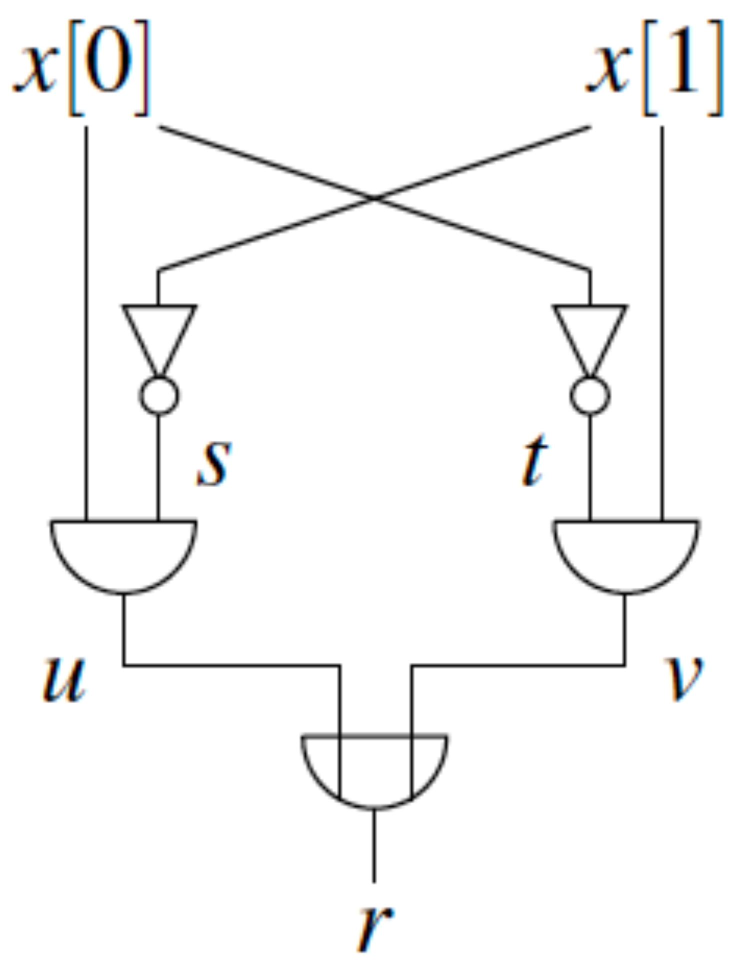
circuits as introduced in I2CA



(a) Symbols for gates in circuit schematics

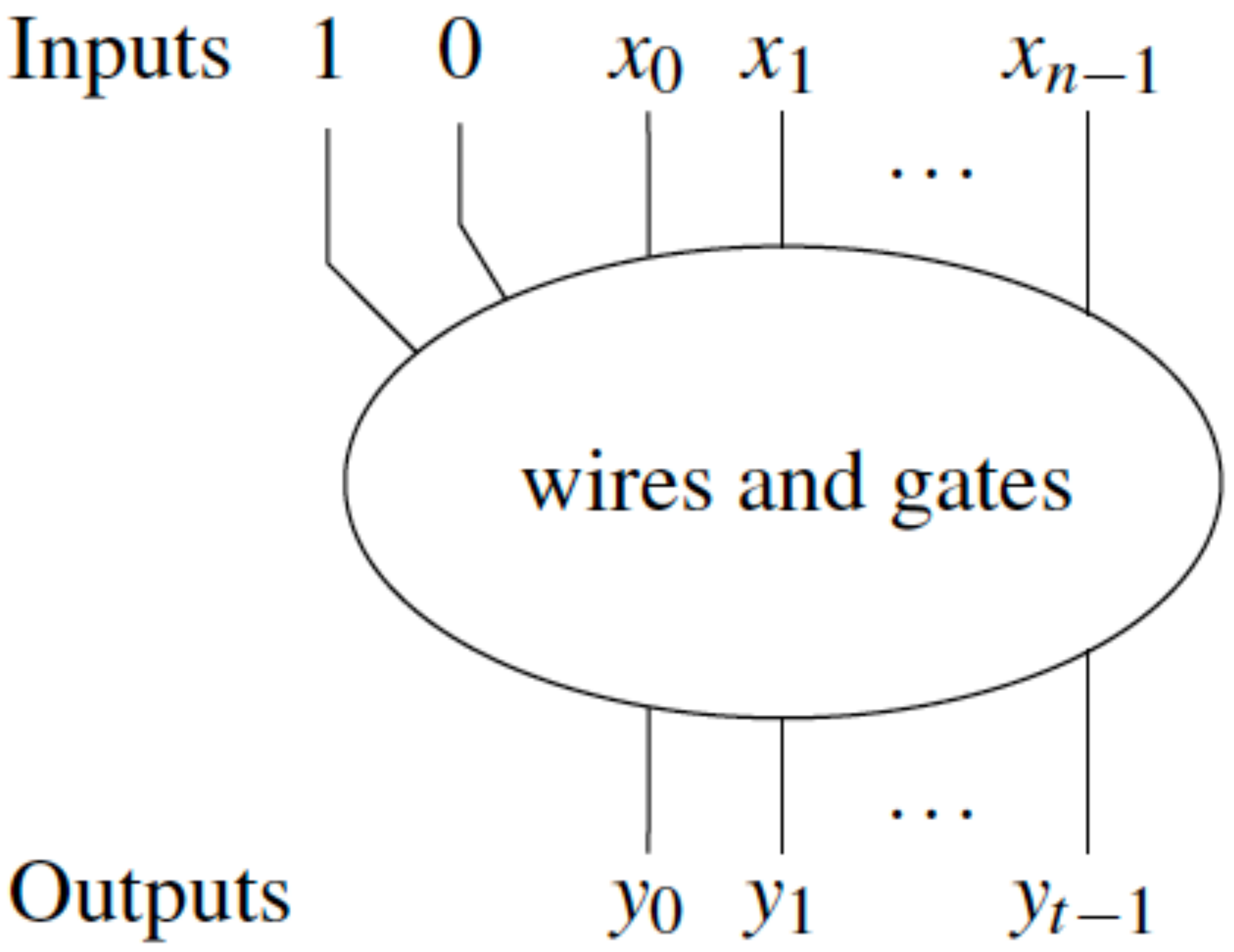


(b) Illustration of inputs and outputs of circuit C

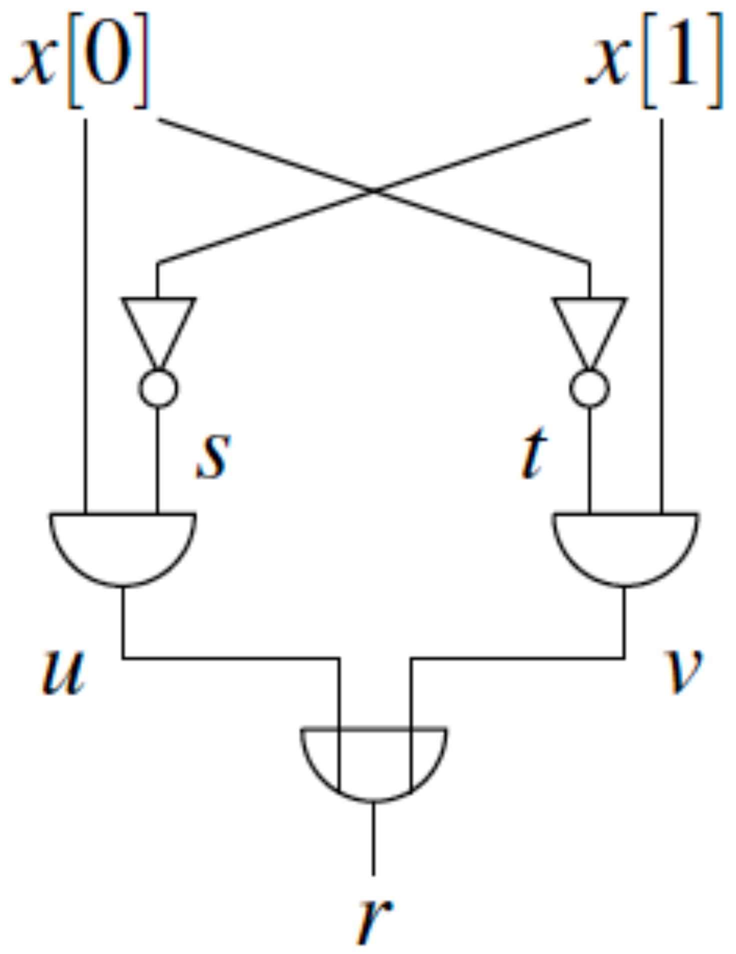


(c) Example of a circuit

Intuitively, a *circuit* C consists of a finite set H of gates, a sequence of input signals $x[n-1:0]$, a set N of wires that connect them, as well as a sequence of output signals $y[m-1:0]$ chosen from all signals of circuit C (as illustrated in Fig. 7(b)). Special inputs 0 and 1 are always available for use in a circuit. Formally we specify a circuit C by the following components:



(b) Illustration of inputs and outputs of circuit C

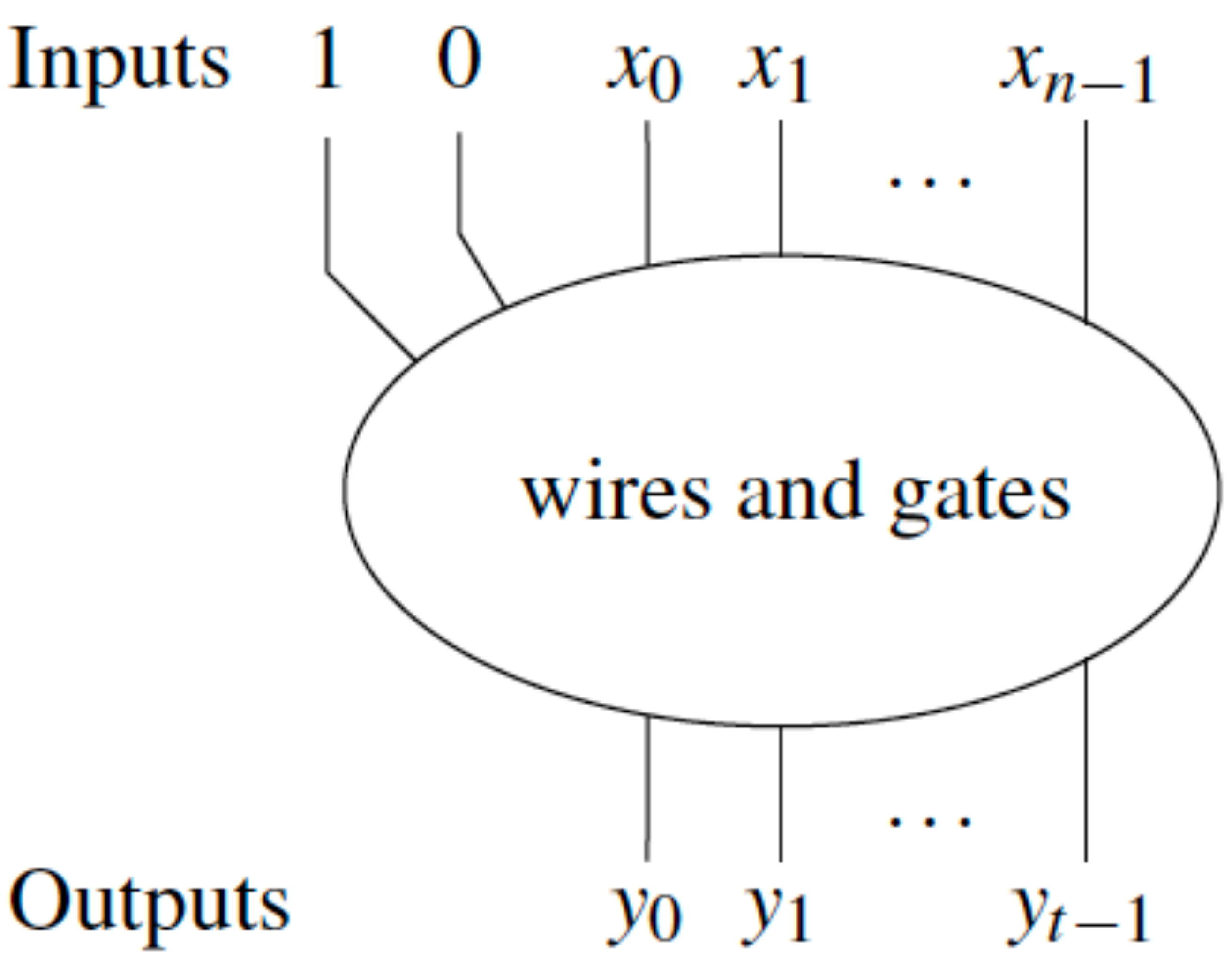


(c) Example of a circuit

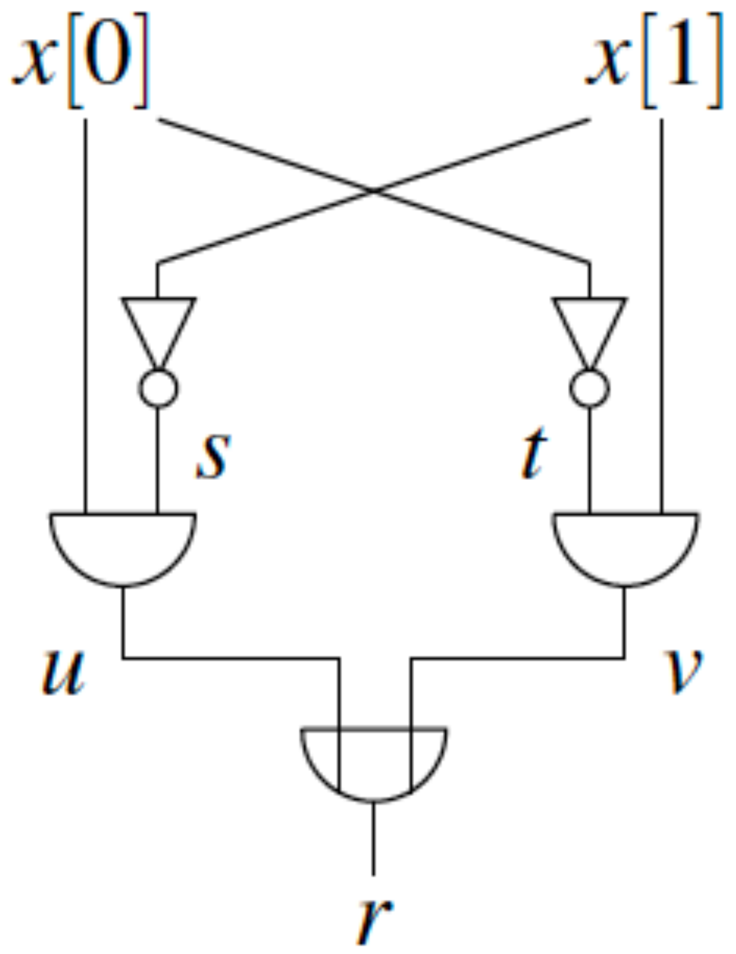
- a sequence of *inputs* $C.x[n-1:0]$. We define the corresponding set of inputs of the circuit as
$$In(C) = \{C.x[i] \mid i \in [0:n-1]\}.$$

- a set $C.H$ of *gates* which is disjoint from the set of inputs
$$C.H \cap In(C) = \emptyset.$$

  The *signals* of a circuit then are its inputs and its gates. Moreover the constant signals 0 and 1 are always available. We collect the signals of circuit C in the set
$$Sig(C) = In(C) \cup C.H \cup \{0,1\}.$$

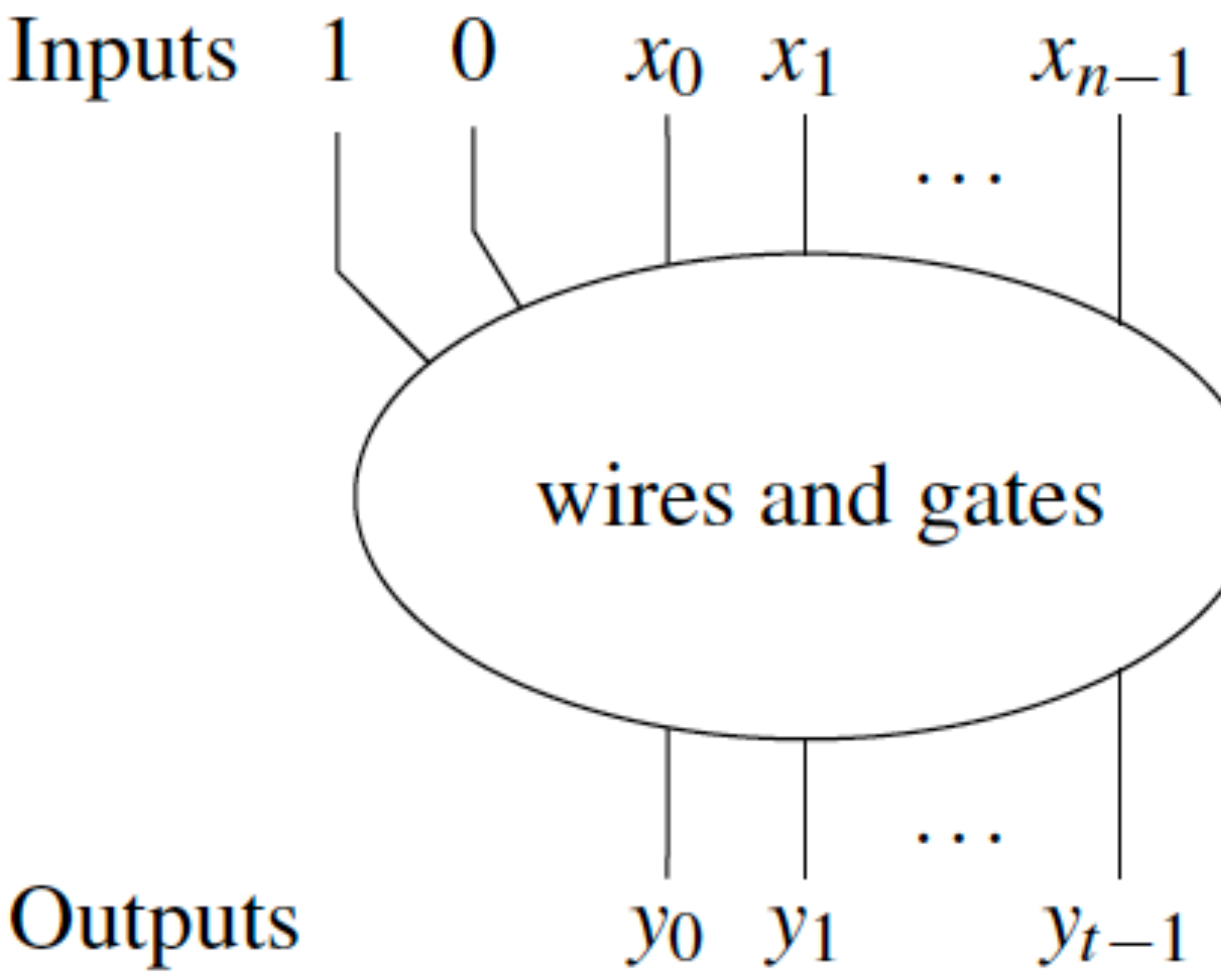


(b) Illustration of inputs and outputs of circuit C

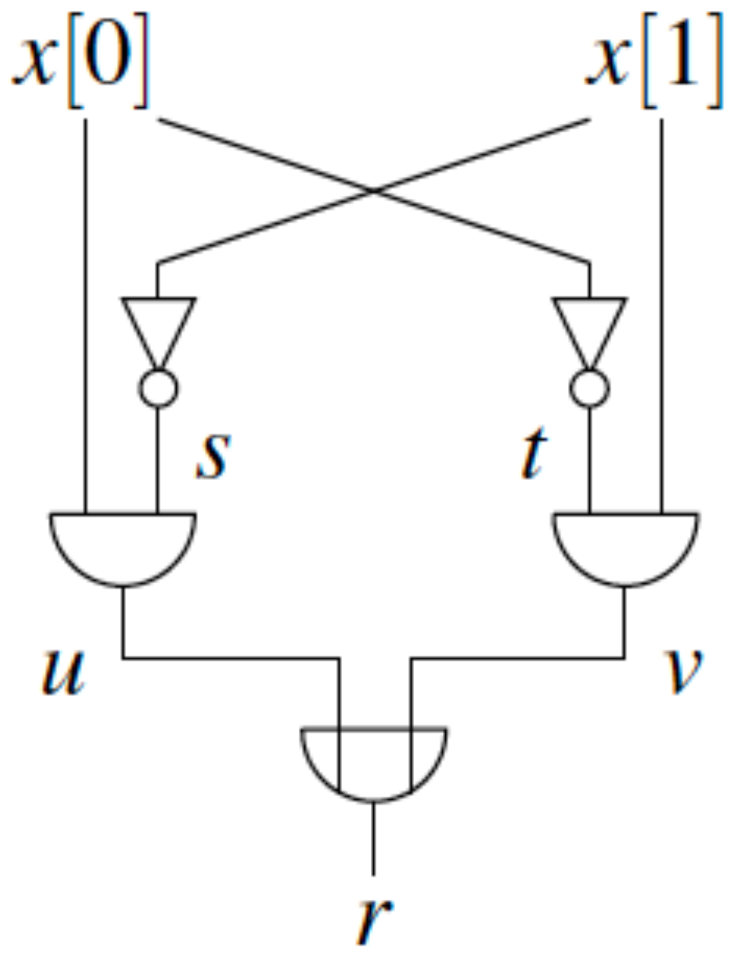


(c) Example of a circuit

- a sequence of *outputs* $C.y[m-1:0]$, which are taken from the signals of the circuit:
$$\forall i \in [0:m-1] : C.y[i] \in Sig(C).$$



(b) Illustration of inputs and outputs of circuit C



(c) Example of a circuit

- a labeling function
$$C.\ell : C.H \to \{\wedge, \vee, \oplus, \neg\}$$
specifying for each gate $g \in C.H$ its type $C.\ell(g)$. Thus, a gate g is a $\circ$-gate if $C.\ell(g) = \circ$.
- two functions
$$C.in1, C.in2 : C.H \to Sig(C)$$
specifying for each gate $g \in C.H$ the signals which provide its left input $C.in1(g)$ and its right input $C.in2(g)$. These functions specify the wires interconnecting the gates and inputs. For inverters the second input does not matter.

Realizing Switching Functions on Disjoint Sets of Variables

circuits S

specified by

- inputs $X[n-1:0]$
- set (here: better sequence) of gates H
- signals $Sig = H \cup \{X_{n-1}, \ldots X_0\}$. Signals of the circuit.
- labeling function $\ell : H \to \{\wedge, \vee, \oplus, \neg\}$. The type of gates.
- outputs $y[m-1:0]$. Without loss of generality the last m gates in H.
- wiring functions $in1, in2 : H \to H \cup \{X_{n-1}, \ldots X_0\}$. Left and right input of each gate.

Realizing Switching Functions on Disjoint Sets of Variables

circuits S

specified by

- inputs $X[n-1:0]$
- set (here: better sequence) of gates H
- signals $Sig = H \cup \{X_{n-1}, \ldots X_0\}$. Signals of the circuit.
- labeling function $\ell : H \to \{\wedge, \vee, \oplus, \neg\}$. The type of gates.
- outputs $y[m-1:0]$. Without loss of generality the last m gates in H.
- wiring functions $in1, in2 : H \to H \cup \{X_{n-1}, \ldots X_0\}$. Left and right input of each gate.

values of signals for an input:

For inputs $a \in \mathbb{B}^n$ and signals $s \in Sig$ we defined (by induction on the dept of s)

- $s(a) \in \mathbb{B}$,the value of s when the inputs have values $X(a) = a$.

Realizing Switching Functions on Disjoint Sets of Variables

circuits S

specified by

- inputs $X[n-1:0]$
- set (here: better sequence) of gates H
- signals $Sig = H \cup \{X_{n-1}, \ldots X_0\}$. Signals of the circuit.
- labeling function $\ell : H \rightarrow \{\wedge, \vee, \oplus, \neg\}$. The type of gates.
- outputs $y[m-1:0]$. Without loss of generality the last m gates in H.
- wiring functions $in1, in2 : H \rightarrow H \cup \{X_{n-1}, \ldots X_0\}$. Left and right input of each gate.

values of signals for an input:

For inputs $a \in \mathbb{B}^n$ and signals $s \in Sig$ we defined (by induction on the dept of s)

- $s(a) \in \mathbb{B}$,the value of s when the inputs have values $X(a) = a$.

switching function f_S computed by a circuit C.

$$f_S : \mathbb{B}^n \rightarrow \mathbb{B}^m \;,\; f_S(a) = (y_{m-1}(a), \ldots y_0(a)) \quad \text{for all } a \in \mathbb{B}^n$$

Realizing Switching Functions on Disjoint Sets of Variables

circuits S

specified by

- inputs $X[n-1:0]$
- set (here: better sequence) of gates H
- signals $Sig = H \cup \{X_{n-1}, \dots X_0\}$. Signals of the circuit.
- labeling function $\ell : H \to \{\wedge, \vee, \oplus, \neg\}$. The type of gates.
- outputs $y[m-1:0]$. Without loss of generality the last m gates in H.
- wiring functions $in1, in2 : H \to H \cup \{X_{n-1}, \dots X_0\}$. Left and right input of each gate.

values of signals for an input:

For inputs $a \in \mathbb{B}^n$ and signals $s \in Sig$ we defined (by induction on the dept of s)

- $s(a) \in \mathbb{B}$,the value of s when the inputs have values $X(a) = a$.

switching function f_S computed by a circuit C.

$$f_S : \mathbb{B}^n \to \mathbb{B}^m \ , \ f_S(a) = (y_{m-1}(a), \dots y_0(a)) \quad \text{for all } a \in \mathbb{B}^n$$

complexity measures

- cost $c(S) = H$: number of gates of S
- depth $d(S)$: depth of the underlying graph of S.

Cost of switching functions

For $f : \mathbb{B}^n \to \mathbb{B}^m$ we define

$$c(f) = \min\{c(S) \ : \ S \text{ computes } f\}$$

Cost of switching functions

For $f : \mathbb{B}^n \to \mathbb{B}^m$ we define

$$c(f) = \min\{c(S) \ : \ S \text{ computes } f\}$$

Lemma 1. *For all $f : \mathbb{B}^n \to \mathbb{B}$ holds*

$$c(f) = O(2^n)$$

Cost of switching functions

For $f : \mathbb{B}^n \to \mathbb{B}^m$ we define

$$c(f) = \min\{c(S) \ : \ S \text{ computes } f\}$$

Lemma 1. *For all* $f : \mathbb{B}^n \to \mathbb{B}$ *holds*

$$c(f) = O(2^n)$$

Proof.

- Realize f by an $n \times 1$- ROM

Cost of switching functions

For $f : \mathbb{B}^n \to \mathbb{B}^m$ we define

$$c(f) = \min\{c(S) \; : \; S \text{ computes } f\}$$

Lemma 1. *For all $f : \mathbb{B}^n \to \mathbb{B}$ holds*

$$c(f) = O(2^n)$$

Proof.

- Realize f by an $n \times 1$- ROM

- basically the same:

$$f(x[n-1:0] = x_{n-1} \wedge f(1, x[n-2:0]) \; \vee /x_{n-1} \wedge f(0, x[n-2:0])$$

Let $F(n)$ be an upper bound for realizing any function $f : \mathbb{B}^n \to \mathbb{B}$, i.e.

$$c(f) \leq F(n) \text{ for all } f : \mathbb{B}^n \to \mathbb{B}$$

We get

$$\begin{aligned} F(n) &\leq 4 + 2F(n-1) \\ F(1) &= 1 \end{aligned}$$

Cost of switching functions

For $f : \mathbb{B}^n \to \mathbb{B}^m$ we define

$$c(f) = \min\{c(S) \ : \ S \text{ computes } f\}$$

Lemma 1. *For all* $f : \mathbb{B}^n \to \mathbb{B}$ *holds*

$$c(f) = O(2^n)$$

Proof. • Realize f by an $n \times 1$- ROM

- basically the same:

$$f(x[n-1:0] = x_{n-1} \wedge f(1, x[n-2:0]) \ \vee /x_{n-1} \wedge f(0, x[n-2:0])$$

Let $F(n)$ be an upper bound for realizing any function $f : \mathbb{B}^n \to \mathbb{B}$, i.e.

$$c(f) \leq F(n) \text{ for all } f : \mathbb{B}^n \to \mathbb{B}$$

We get

$$\begin{aligned} F(n) &\leq 4 + 2F(n-1) \\ F(1) &= 1 \end{aligned}$$

Functions on disjoint sets of variables

Let $f : \mathbb{B}^n \to \mathbb{B}^r$ and $g : \mathbb{B}^m \to \mathbb{B}^s$ be switching functions. Define

$$f \times g : B^{n+m} \to \mathbb{B}^{r+s}$$

$$f \times g(a,b) = (f(a), g(b))$$

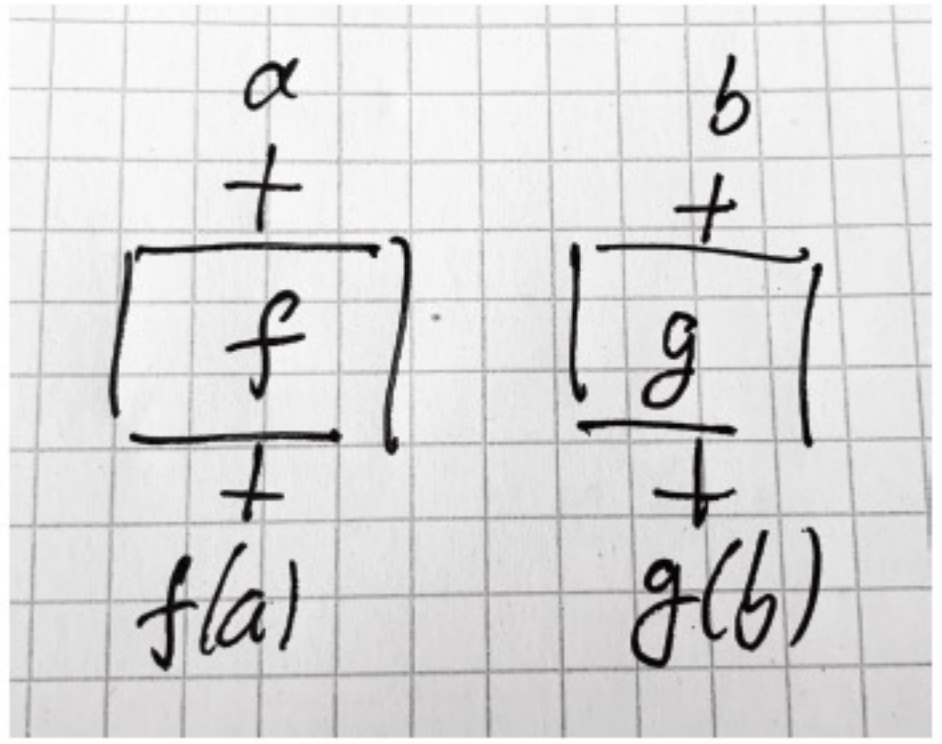


Figure 1: The obvious way to realize $f \times g$ is to build and optimize circuits for f and g separately.

Functions on disjoint sets of variables

Let $f : \mathbb{B}^n \to \mathbb{B}^r$ and $g : \mathbb{B}^m \to \mathbb{B}^s$ be switching functions. Define

$$f \times g : B^{n+m} \to \mathbb{B}^{r+s}$$

$$f \times g(a,b) = (f(a), g(b))$$

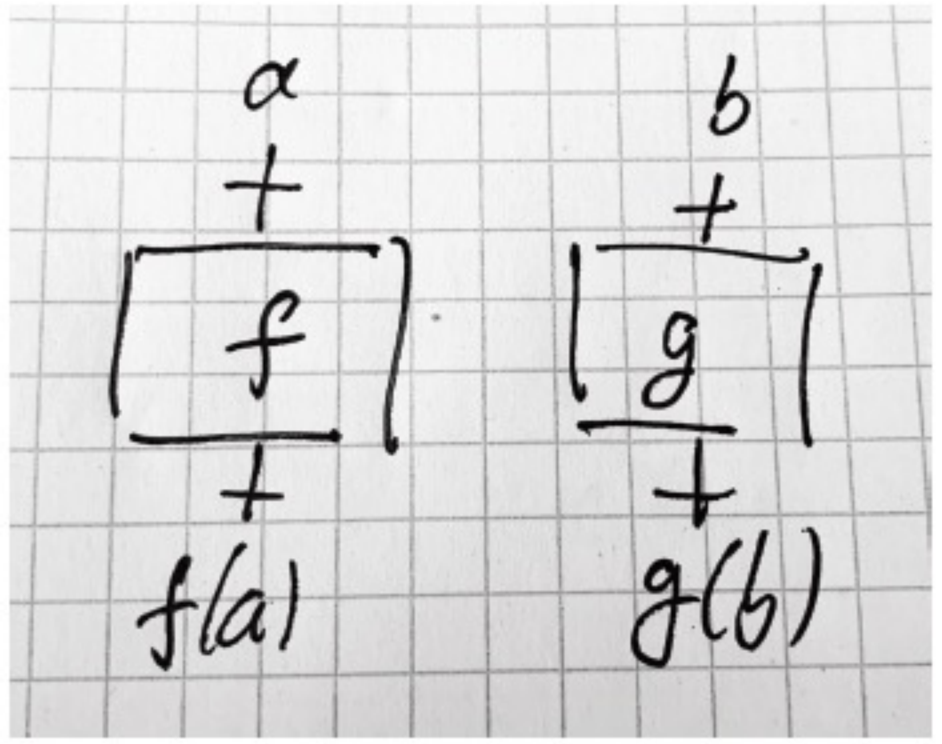


Figure 1: The obvious way to realize $f \times g$ is to build and optimize circuits for f and g separately.

plausible conjecture

$$c(f \times g) = c(f) + c(g)$$

i.e. this is the best one can do

Functions on disjoint sets of variables

Let $f : \mathbb{B}^n \rightarrow \mathbb{B}^r$ and $g : \mathbb{B}^m \rightarrow \mathbb{B}^s$ be switching functions. Define

$$f \times g : B^{n+m} \rightarrow \mathbb{B}^{r+s}$$

$$f \times g(a,b) = (f(a), g(b))$$

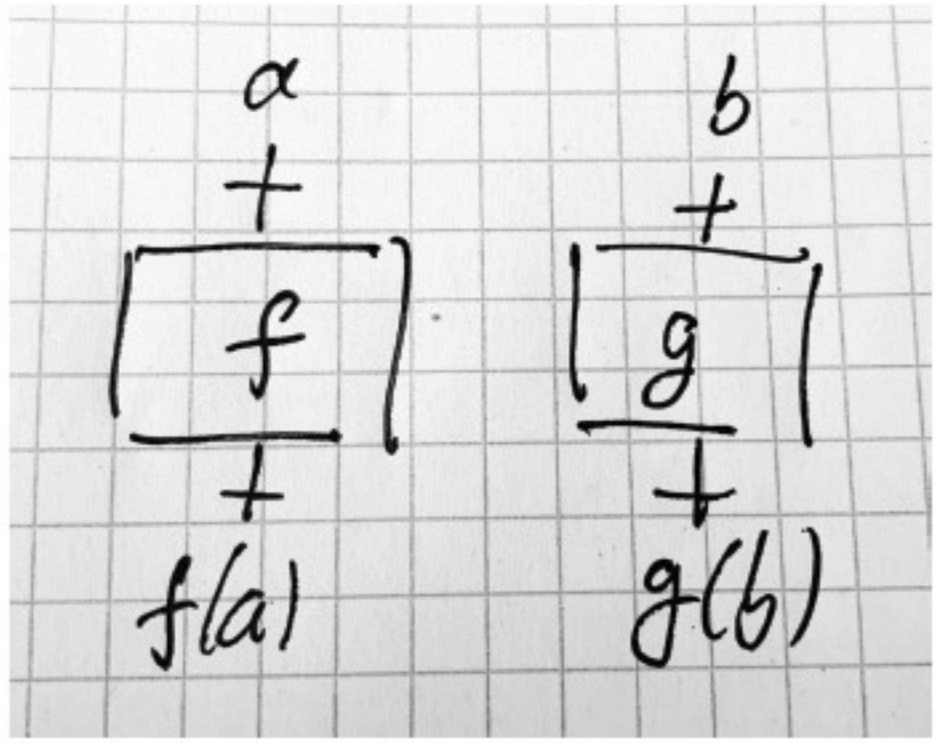


Figure 1: The obvious way to realize $f \times g$ is to build and optimize circuits for f and g separately.

plausible conjecture

$$c(f \times g) = c(f) + c(g)$$

i.e. this is the best one can do

'obviously true', but not really true

Existence of very expensive switching functions

Lemma 2. *There are switching functions* $f : \mathbb{B}^n \to B$ *such that* $c(f) = \Omega(2^n/n)$.

Existence of very expensive switching functions

Lemma 2. *There are switching functions $f : \mathbb{B}^n \to B$ such that $c(f) = \Omega(2^n/n)$.*

By a counting argument:

- there are very many switching functions

  $$|\{f \ : \ f : \mathbb{B}^n \to \mathbb{B}\}| = 2^{2^n}$$

  because each f is specified by the sequence $s_f \in \mathbb{B}^{2^n}$ of function values in lexicographic order

Existence of very expensive switching functions

Lemma 2. *There are switching functions $f : \mathbb{B}^n \to B$ such that $c(f) = \Omega(2^n/n)$.*

By a counting argument:

- there are very many switching functions

  $$|\{f \ : \ f : \mathbb{B}^n \to \mathbb{B}\}| = 2^{2^n}$$

  because each f is specified by the sequence $s_f \in \mathbb{B}^{2^n}$ of function values in lexicographic order

- there are not many circuits with n inputs and c gates. Let $N(c,n)$ be the number of such circuits then

  $$N(c,n) \leq 4^c \cdot (n+c)^c \cdot (n+c)^c$$

  number of ways to label gates and wire left and right input.

Existence of very expensive switching functions

Lemma 2. *There are switching functions $f : \mathbb{B}^n \to B$ such that $c(f) = \Omega(2^n/n)$.*

By a counting argument:

- there are very many switching functions

$$|\{f \ : \ f : \mathbb{B}^n \to \mathbb{B}\}| = 2^{2^n}$$

  because each f is specified by the sequence $s_f \in \mathbb{B}^{2^n}$ of function values in lexicographic order

- there are not many circuits with n inputs and c gates. Let $N(c,n)$ be the number of such circuits then

$$N(c,n) \leq 4^c \cdot (n+c)^c \cdot (n+c)^c$$

  number of ways to label gates and wire left and right input.

- we need

$$4^c \cdot (n+c)^{2c} \geq N(c,n) \geq 2^{2^n}$$

$$2c(1+\log(n+c)) \geq \log(N(c,n)) \geq 2^n$$

Existence of very expensive switching functions

Lemma 2. *There are switching functions $f : \mathbb{B}^n \to B$ such that $c(f) = \Omega(2^n/n)$.*

By a counting argument:

- there are very many switching functions

$$|\{f \ : \ f : \mathbb{B}^n \to \mathbb{B}\}| = 2^{2^n}$$

  because each f is specified by the sequence $s_f \in \mathbb{B}^{2^n}$ of function values in lexicographic order

- there are not many circuits with n inputs and c gates. Let $N(c,n)$ be the number of such circuits then

$$N(c,n) \leq 4^c \cdot (n+c)^c \cdot (n+c)^c$$

  number of ways to label gates and wire left and right input.

- we need

$$4^c \cdot (n+c)^{2c} \geq N(c,n) \geq 2^{2^n}$$

$$2c(1+\log(n+c)) \geq \log(N(c,n)) \geq 2^n$$

- claim: $c > 2^n/(4n)$ for large n

  otherwise

$$\begin{aligned}
\log(N(c,n)) &\leq \frac{2^n}{2n}(1+\log(2^n/(4n)+n) \\
&\leq \frac{2^n}{2n}(1+\log(2^n/2)) \\
&= 2^n/2
\end{aligned}$$

Realizing an expensive function k times

For $f : \mathbb{B}^n \to \mathbb{B}$ define

$$f^k : \mathbb{B}^n \to \mathbb{B}^k$$

$$f^k(a_1, \ldots, a_n) = (f(a_1), \ldots f(a_n)) \quad \text{for } a_i \in \mathbb{B}^n$$

Realizing an expensive function k times

For $f : \mathbb{B}^n \to \mathbb{B}$ define

$$f^k : \mathbb{B}^n \to \mathbb{B}^k$$

$$f^k(a_1, \ldots, a_n) = (f(a_1), \ldots f(a_n)) \quad \text{for } a_i \in \mathbb{B}^n$$

- If conjecture holds and f expensive then

$$c(f^k) = k \cdot c(f) \geq k \cdot 2^n/(4n)$$

-

$$c(f) = O(2^n)$$

comparators for packets with payload

- inputs $(x,a),(y,b)$ with keys (addresses) $x,y \in \mathbb{B}^u$ and payloads $a,b \in \mathbb{B}^\mu$
- outputs: inputs sorted by keys

$$(x < y)?((x,a),(y,b)) : (y,b),(x,a)$$

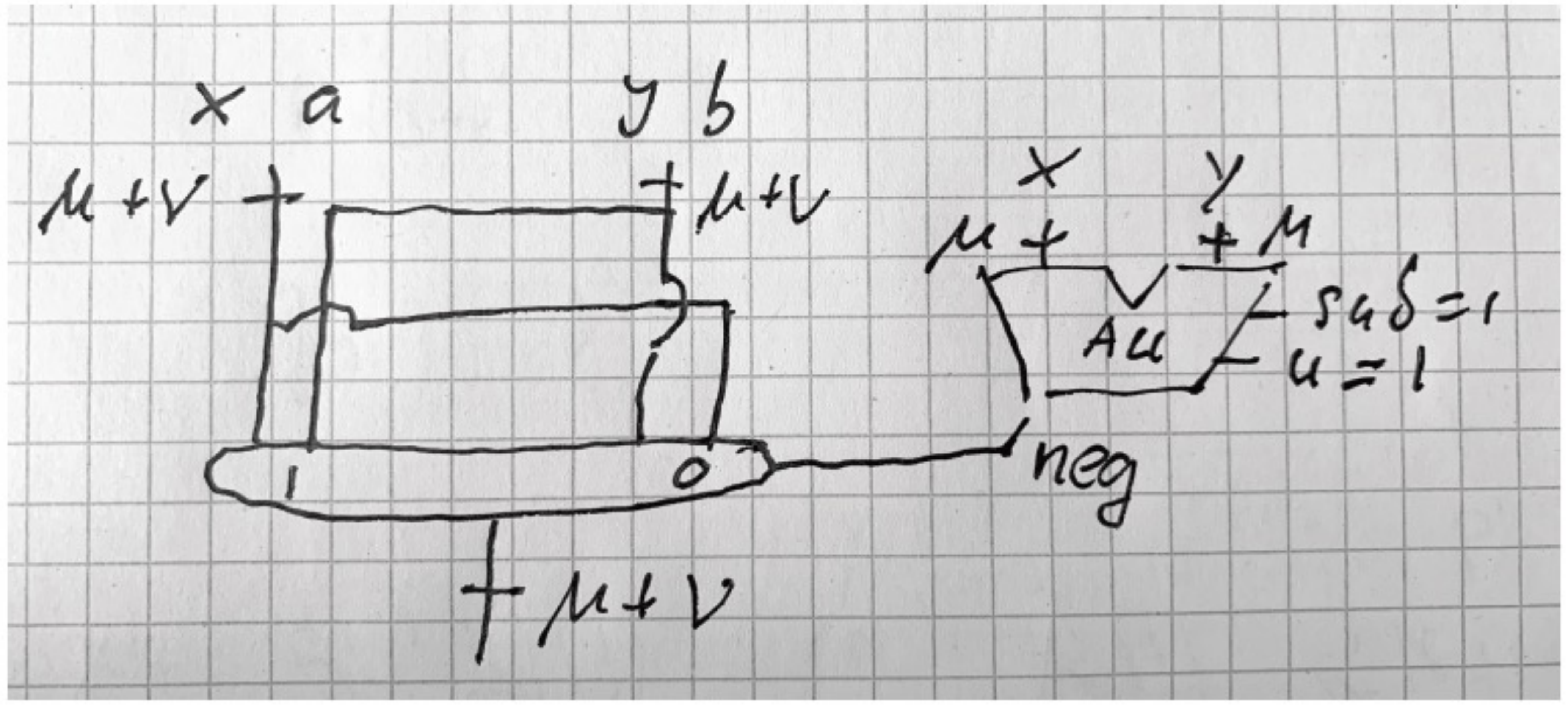


Figure 2: Circuit for a packet-comparator with ν bit keys x,y and ν bit payloads a,b

comparators for packets with payload

- inputs $(x,a),(y,b)$ with keys (addresses) $x,y \in \mathbb{B}^u$ and payloads $a,b \in \mathbb{B}^\mu$
- outputs: inputs sorted by keys

$$(x<y)?((x,a),(y,b)):(y,b),(x,a)$$

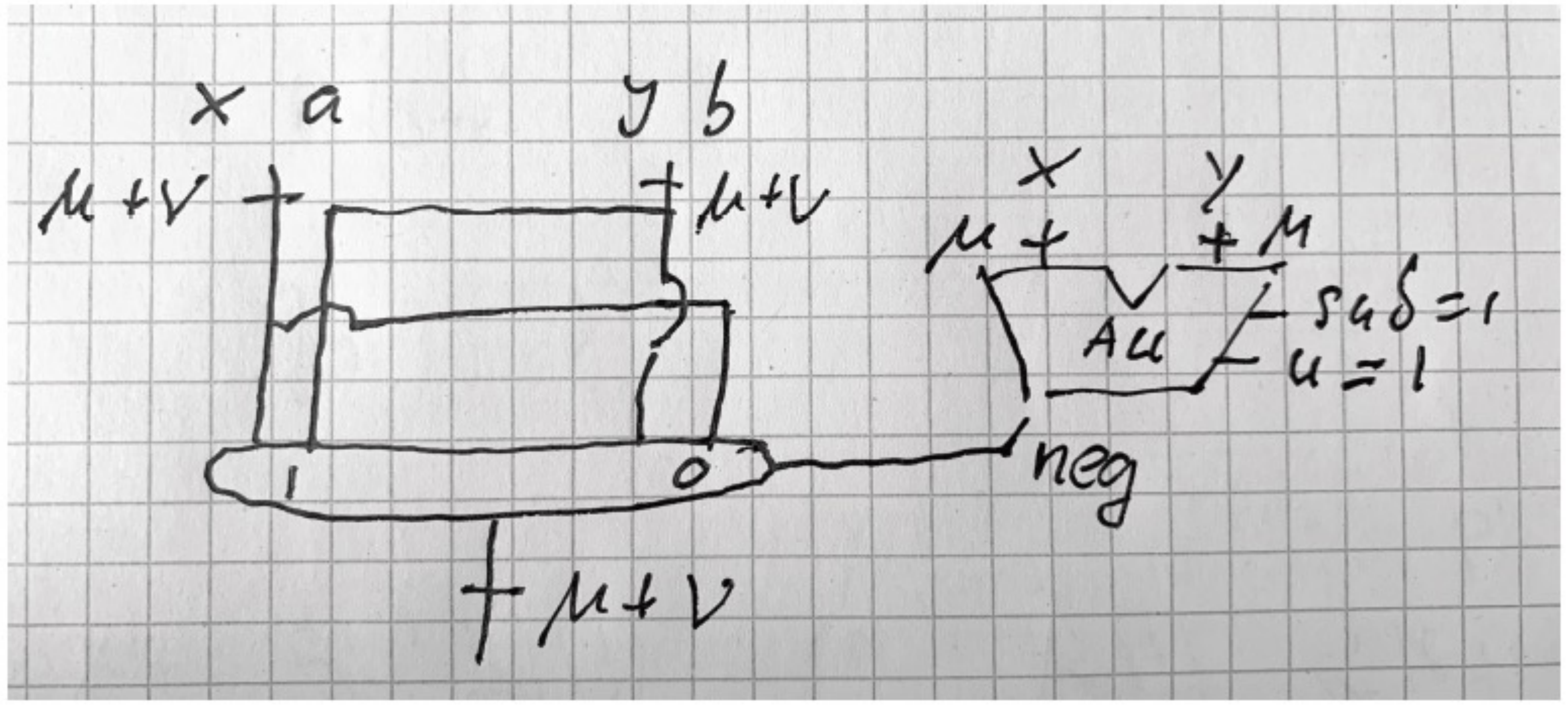


Figure 2: Circuit for a packet-comparator with ν bit keys x,y and ν bit payloads a,b

Cost: $O(\nu+\mu)$

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).

$$\nu + \mu = n + \log k$$

  Sort by a_i's. Indices i are return destinations.

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).
$$\nu + \mu = n + \log k$$
Sort by a_i's. Indices i are return destinations.

- result (in column i):
$$(b_i, \pi(i)) \text{ with } b_i = a_{\pi(i)}$$
π is the permutation necessary to 'send letters' back to a_i's.

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).

$$\nu + \mu = n + \log k$$

  Sort by a_i's. Indices i are return destinations.

- result (in column i):

$$(b_i, \pi(i)) \text{ with } b_i = a_{\pi(i)}$$

  π is the permutation necessary to 'send letters' back to a_i's.

- the b_i are sorted with duplicates forming blocks. Find leaders of blocks:

$$c_i = \begin{cases} 0b_i & i = 1 \vee b_{i-1} \neq b_i \quad \text{(leader of block)} \\ 1b_i & \text{otherwise (duplicate)} \end{cases}$$

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).
$$\nu + \mu = n + \log k$$
Sort by a_i's. Indices i are return destinations.

- result (in column i):
$$(b_i, \pi(i)) \text{ with } b_i = a_{\pi(i)}$$
π is the permutation necessary to 'send letters' back to a_i's.

- the b_i are sorted with duplicates forming blocks. Find leaders of blocks:
$$c_i = \begin{cases} 0b_i & i = 1 \vee b_{i-1} \neq b_i \quad \text{(leader of block)} \\ 1b_i & \text{otherwise (duplicate)} \end{cases}$$

- sort (c_i, i) by c_i's
$$\nu + \mu = n + \log k + 1$$

- results
$$(d_i, \rho(i)) \text{ with } d_i = b_{\rho(i)}$$
ρ is the permutation necessary to 'send letters' back to b_i's.
Leaders occupy first L columns where
$$L \leq 2^n \text{ as } d_i \in 0 \circ \mathbb{B}^n$$

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).

$$\nu + \mu = n + \log k$$

Sort by a_i's. Indices i are return destinations.

- result (in column i):

$$(b_i, \pi(i)) \text{ with } b_i = a_{\pi(i)}$$

π is the permutation necessary to 'send letters' back to a_i's.

- the b_i are sorted with duplicates forming blocks. Find leaders of blocks:

$$c_i = \begin{cases} 0b_i & i = 1 \vee b_{i-1} \neq b_i \quad \text{(leader of block)} \\ 1b_i & \text{otherwise (duplicate)} \end{cases}$$

- sort (c_i, i) by c_i's

$$\nu + \mu = n + \log k + 1$$

- results

$$(d_i, \rho(i)) \text{ with } d_i = b_{\rho(i)}$$

ρ is the permutation necessary to 'send letters' back to b_i's.
Leaders occupy first L columns where

$$L \leq 2^n \text{ as } d_i \in 0 \circ \mathbb{B}^n$$

- use 2^n circuits for f to compute

$$e_i = f(d_i[n-1:0])$$

For leaders we have

$$e_i = f(b_{\rho(i)})$$

Cost

$$O(2^n \cdot 2^n) = O(2^{2n})$$

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).
$$\nu + \mu = n + \log k$$
Sort by a_i's. Indices i are return destinations.

- result (in column i):
$$(b_i, \pi(i)) \text{ with } b_i = a_{\pi(i)}$$
π is the permutation necessary to 'send letters' back to a_i's.

- the b_i are sorted with duplicates forming blocks. Find leaders of blocks:
$$c_i = \begin{cases} 0b_i & i = 1 \vee b_{i-1} \neq b_i \quad \text{(leader of block)} \\ 1b_i & \text{otherwise (duplicate)} \end{cases}$$

- sort (c_i, i) by c_i's
$$\nu + \mu = n + \log k + 1$$

- results
$$(d_i, \rho(i)) \text{ with } d_i = b_{\rho(i)}$$
ρ is the permutation necessary to 'send letters' back to b_i's.
Leaders occupy first L columns where
$$L \leq 2^n \text{ as } d_i \in 0 \circ \mathbb{B}^n$$

- use 2^n circuits for f to compute
$$e_i = f(d_i[n-1:0])$$
For leaders we have
$$e_i = f(b_{\rho(i)})$$
Cost
$$O(2^n \cdot 2^n) = O(2^{2n})$$

- send e_i back to sender: sort
$$(e_i, \rho(i))$$
by second component
$$\nu + \mu = 1 + \log k$$

Using sorting networks as post office

goal: cheap circuit for f^k

Use $k > 2^n$. Bitonic sorting network with columns $i \in [1:k]$. What happens in column $i \in \mathbb{B}^{\log k}$ (coded binary)

- input (a_i, i).
$$\nu + \mu = n + \log k$$
Sort by a_i's. Indices i are return destinations.

- result (in column i):
$$(b_i, \pi(i)) \text{ with } b_i = a_{\pi(i)}$$
π is the permutation necessary to 'send letters' back to a_i's.

- the b_i are sorted with duplicates forming blocks. Find leaders of blocks:
$$c_i = \begin{cases} 0b_i & i = 1 \vee b_{i-1} \neq b_i \quad \text{(leader of block)} \\ 1b_i & \text{otherwise (duplicate)} \end{cases}$$

- sort (c_i, i) by c_i's
$$\nu + \mu = n + \log k + 1$$

- results
$$(d_i, \rho(i)) \text{ with } d_i = b_{\rho(i)}$$
ρ is the permutation necessary to 'send letters' back to b_i's.
Leaders occupy first L columns where
$$L \leq 2^n \text{ as } d_i \in 0 \circ \mathbb{B}^n$$

- use 2^n circuits for f to compute
$$e_i = f(d_i[n-1:0])$$
For leaders we have
$$e_i = f(b_{\rho(i)})$$
Cost
$$O(2^n \cdot 2^n) = O(2^{2n})$$

- send e_i back to sender: sort
$$(e_i, \rho(i))$$
by second component
$$\nu + \mu = 1 + \log k$$

- result in row $\rho(i)$ is
$$(e_i, \rho(i))$$
For leaders this is
$$(e_i, \rho(i)) = (f(b_{\rho(i)}), \rho(i))$$
Let $j = \rho(i)$ the columns from where a leader was sent to column i. Then we have in column j
$$(f(b_j), j)$$

Using sorting networks as post office

goal: cheap circuit for f^k

- use 2^n circuits for f to compute

$$e_i = f(d_i[n-1:0])$$

  For leaders we have

$$e_i = f(b_{\rho(i)})$$

  Cost

$$O(2^n \cdot 2^n) = O(2^{2n})$$

- send e_i back to sender: sort

$$(e_i, \rho(i))$$

  by second component

$$\nu + \mu = 1 + \log k$$

- result in row $\rho(i)$ is

$$(e_i, \rho(i))$$

  For leaders this is

$$(e_i, \rho(i)) = (f(b_{\rho(i)}), \rho(i))$$

  Let $j = \rho(i)$ the columns from where a leader was sent to column i. Then we have in column j

$$(f(b_j), j)$$

Using sorting networks as post office

goal: cheap circuit for f^k

- use 2^n circuits for f to compute

$$e_i = f(d_i[n-1:0])$$

  For leaders we have

$$e_i = f(b_{\rho(i)})$$

  Cost

$$O(2^n \cdot 2^n) = O(2^{2n})$$

- send e_i back to sender: sort

$$(e_i, \rho(i))$$

  by second component

$$\nu + \mu = 1 + \log k$$

- result in row $\rho(i)$ is

$$(e_i, \rho(i))$$

  For leaders this is

$$(e_i, \rho(i)) = (f(b_{\rho(i)}), \rho(i))$$

  Let $j = \rho(i)$ the columns from where a leader was sent to column i. Then we have in column j

$$(f(b_j), j)$$

- copy function values of leaders into their duplicates

$$h_i = \begin{cases} f(b_i) & i = 0 \lor b_{i-1} \neq b_i \quad \text{(leader)} \\ h_{i-1} & \text{otherwise} \end{cases}$$

  and we have in *all* columns i

$$(f(b_i), i) = (f(a_{\pi(i)}), i)$$

Using sorting networks as post office

goal: cheap circuit for f^k

- use 2^n circuits for f to compute

$$e_i = f(d_i[n-1:0])$$

For leaders we have

$$e_i = f(b_{\rho(i)})$$

Cost

$$O(2^n \cdot 2^n) = O(2^{2n})$$

- send e_i back to sender: sort

$$(e_i, \rho(i))$$

by second component

$$\nu + \mu = 1 + \log k$$

- result in row $\rho(i)$ is

$$(e_i, \rho(i))$$

For leaders this is

$$(e_i, \rho(i)) = (f(b_{\rho(i)}), \rho(i))$$

Let $j = \rho(i)$ the columns from where a leader was sent to column i. Then we have in column j

$$(f(b_j), j)$$

- copy function values of leaders into their duplicates

$$h_i = \begin{cases} f(b_i) & i = 0 \vee b_{i-1} \neq b_i \quad \text{(leader)} \\ h_{i-1} & \text{otherwise} \end{cases}$$

and we have in *all* columns i

$$(f(b_i), i) = (f(a_{\pi(i)}), i)$$

- sort $(f(a_{\pi(i)}), \pi(i))$ by second components. The $\pi(i)$ were computed in columns i before. This transports to columns $\pi(i)$ packet

$$(f(a_{\pi(i)}), \pi(i))$$

Resp to column each column $j = \pi(i)$ packet

$$(f(a_j), j)$$

Using sorting networks as post office

goal: cheap circuit for f^k

- use 2^n circuits for f to compute

$$e_i = f(d_i[n-1:0])$$

For leaders we have

$$e_i = f(b_{\rho(i)})$$

Cost

$$O(2^n \cdot 2^n) = O(2^{2n})$$

- send e_i back to sender: sort

$$(e_i, \rho(i))$$

by second component

$$\nu + \mu = 1 + \log k$$

- result in row $\rho(i)$ is

$$(e_i, \rho(i))$$

For leaders this is

$$(e_i, \rho(i)) = (f(b_{\rho(i)}), \rho(i))$$

Let $j = \rho(i)$ the columns from where a leader was sent to column i. Then we have in column j

$$(f(b_j), j)$$

- copy function values of leaders into their duplicates

$$h_i = \begin{cases} f(b_i) & i = 0 \vee b_{i-1} \neq b_i \quad \text{(leader)} \\ h_{i-1} & \text{otherwise} \end{cases}$$

and we have in *all* columns i

$$(f(b_i), i) = (f(a_{\pi(i)}), i)$$

- sort $(f(a_{\pi(i)}), \pi(i))$ by second components. The $\pi(i)$ were computed in columns i before. This transports to columns $\pi(i)$ packet

$$(f(a_{\pi(i)}), \pi(i))$$

Resp to column each column $j = \pi(i)$ packet

$$(f(a_j), j)$$

- Output $(f(a_1), \ldots, f(a_k))$

cost

- each comparator: $O(n + \log k)$
- number of comparators in 4 sorting networks: $O(k(\log k)^2)$
- computation of b_i's: $O(k \cdot n))$
- circuits for f: $O(2^{2n})$
- computation of h_i's: $O(k)$ as leading bits of c_i's are known

cost

- each comparator: $O(n + \log k)$
- number of comparators in 4 sorting networks: $O(k(\log k)^2)$
- computation of b_i's: $O(k \cdot n))$
- circuits for f: $O(2^{2n})$
- computation of h_i's: $O(k)$ as leading bits of c_i's are known

Choose
$$k = (2^n)^2 = 2^{2n}$$

cost

- each comparator: $O(n + \log k)$
- number of comparators in 4 sorting networks: $O(k(\log k)^2)$
- computation of b_i's: $O(k \cdot n))$
- circuits for f: $O(2^{2n})$
- computation of h_i's: $O(k)$ as leading bits of c_i's are known

Choose
$$k = (2^n)^2 = 2^{2n}$$

then

- each comparator: $O(n)$
- number of comparators in 4 sorting networks: $O(n^2 \cdot 2^{2n})$
- computation of b_i's: $O(n \cdot 2^{2n}))$
- circuits for f: $O(2^{2n})$
- computation of h_i's: $O(2^{2n})$ as leading bits of c_i's are known

cost

- each comparator: $O(n + \log k)$
- number of comparators in 4 sorting networks: $O(k(\log k)^2)$
- computation of b_i's: $O(k \cdot n))$
- circuits for f: $O(2^{2n})$
- computation of h_i's: $O(k)$ as leading bits of c_i's are known

Choose

$$k = (2^n)^2 = 2^{2n}$$

then

- each comparator: $O(n)$
- number of comparators in 4 sorting networks: $O(n^2 \cdot 2^{2n})$
- computation of b_i's: $O(n \cdot 2^{2n}))$
- circuits for f: $O(2^{2n})$
- computation of h_i's: $O(2^{2n})$ as leading bits of c_i's are known

Total cost

$$c(f^k) = O(n^3 \cdot 2^{2n})$$

If conjecture would be true we would have

$$c(f^k) = k \cdot c(f) = \Omega(2^{2n} \cdot 2^n/n) = \Omega(2^{3n}/n)$$