ROUTING ?

Sorting?

Image Processing?

Sparse Matrices?

PÅRC

Heiko Schröder, 1998

Reconfigurable architectures

- FPGAs
- reconfigurable multibus
- reconfigurable networks (Transputers, PVM)
- **dynamically reconfigurable mesh**

Aim:

efficiency

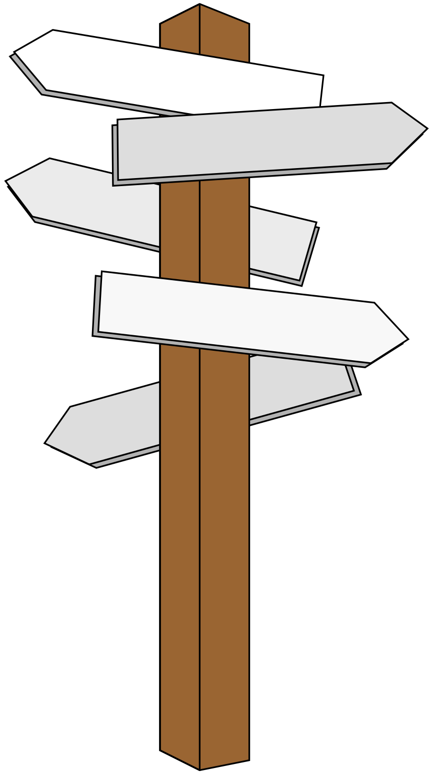
special purpose --> general purpose architectures

contents

1.) Motivation for the reconfigurable mesh

2.) Routing (and sorting):

- better than PRAM
- better than mesh

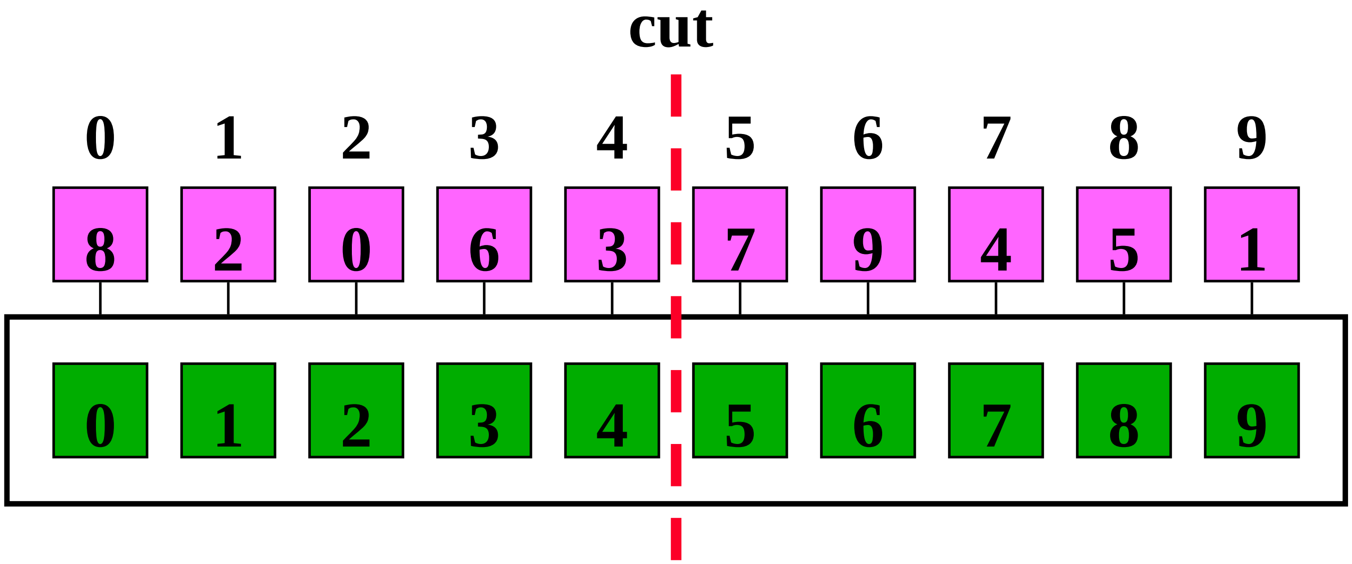


3.) Image processing

4.) Sparse matrix multiplication

5.) Bounded bus length

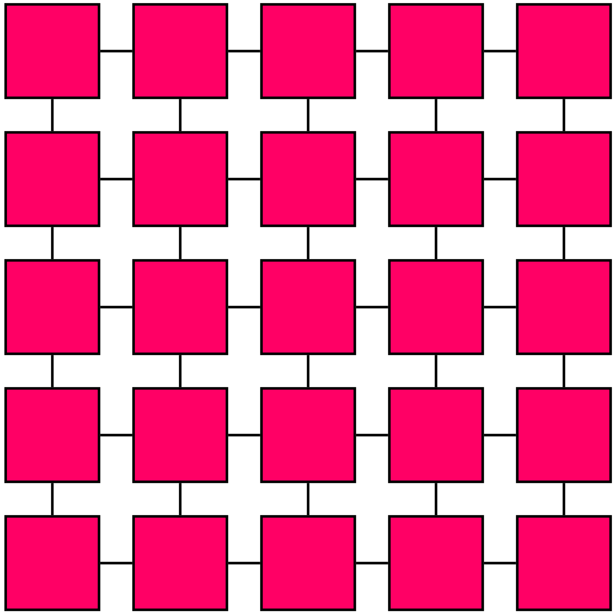
PRAM



diameter O(**1**) bisection width Ω(**n**)

EREW **CRCW**

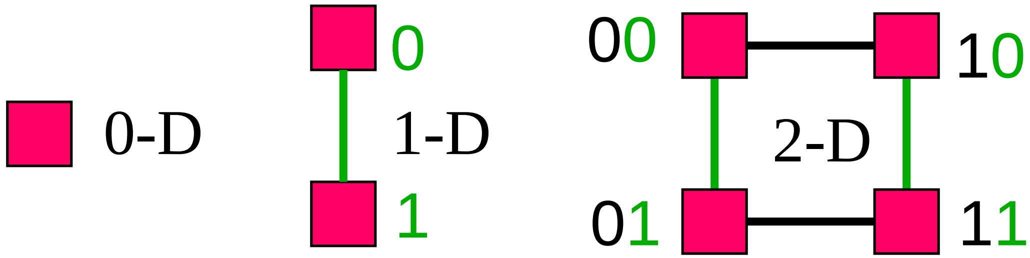
Mesh/Torus



2D mesh

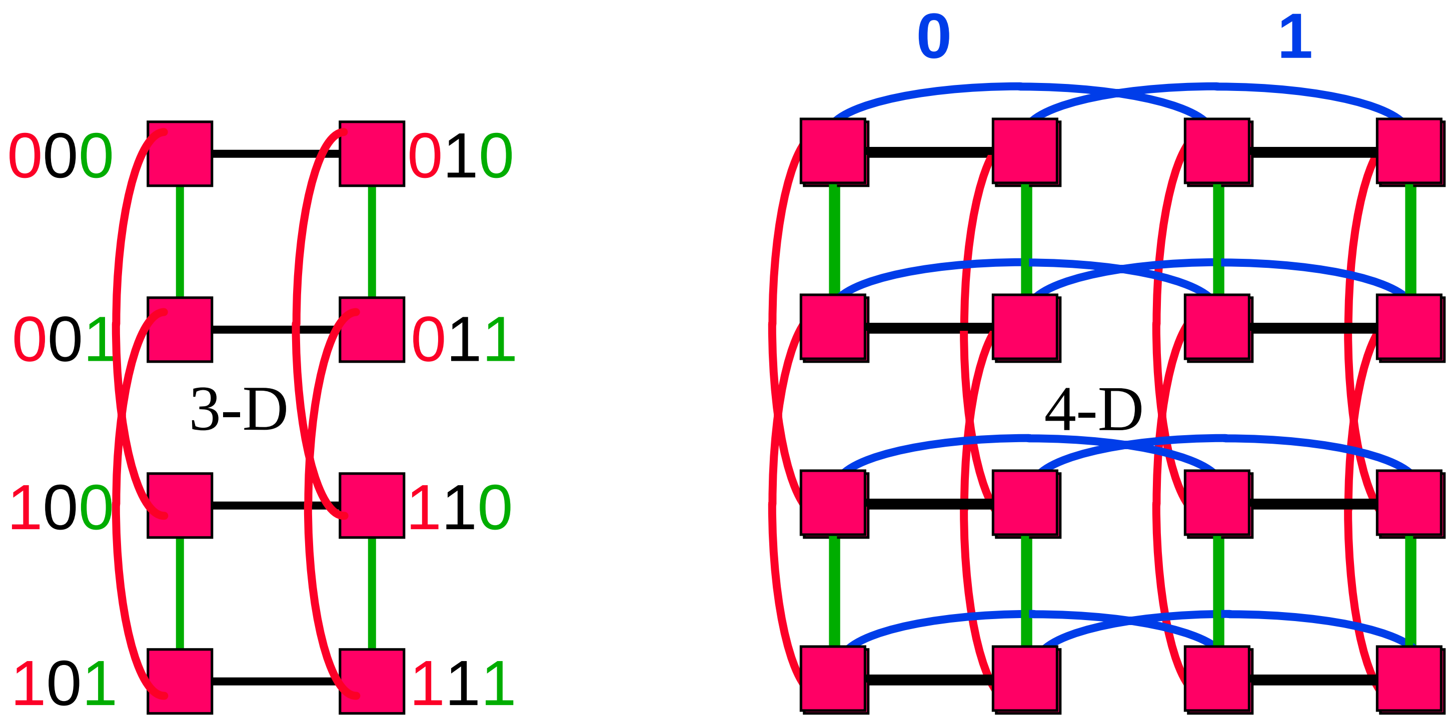
Diameter $\Theta(\sqrt{n})$
bisection width $\Theta(\sqrt{n})$

Hypercube



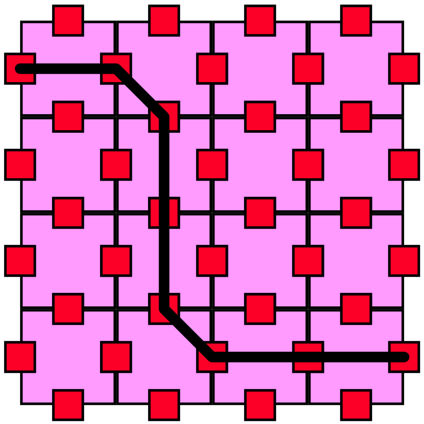
diameter $O(\log n)$

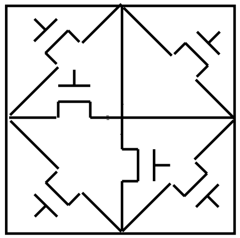
bisection width $\Theta(n)$



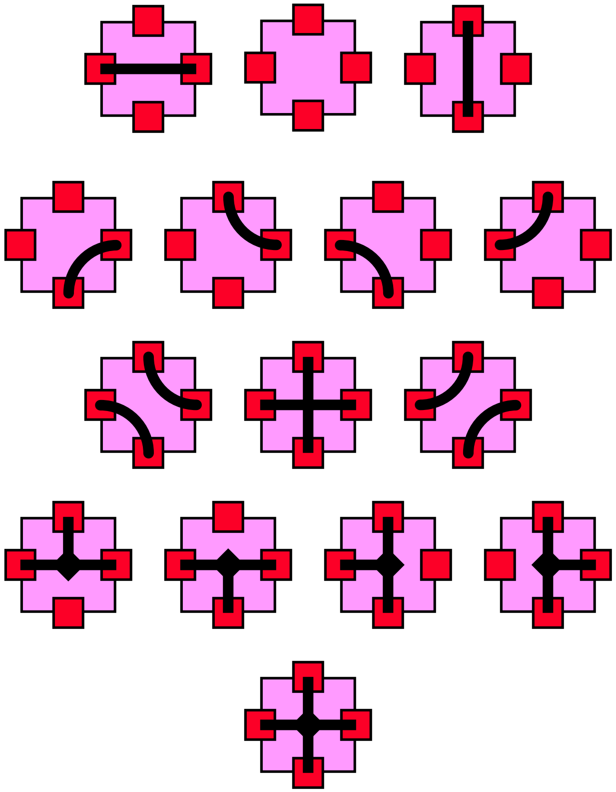
reconfigurable mesh

reconfigurable mesh = mesh + interior connections





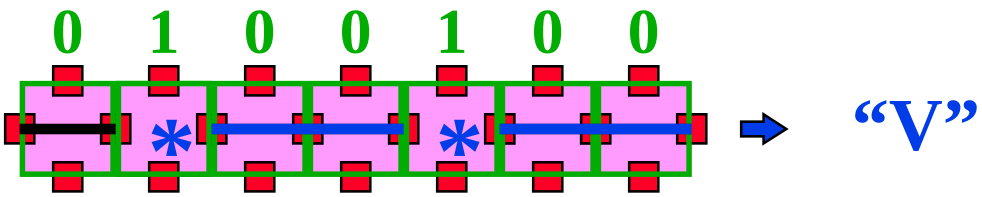
low cost



diameter 1 !!

15 positions

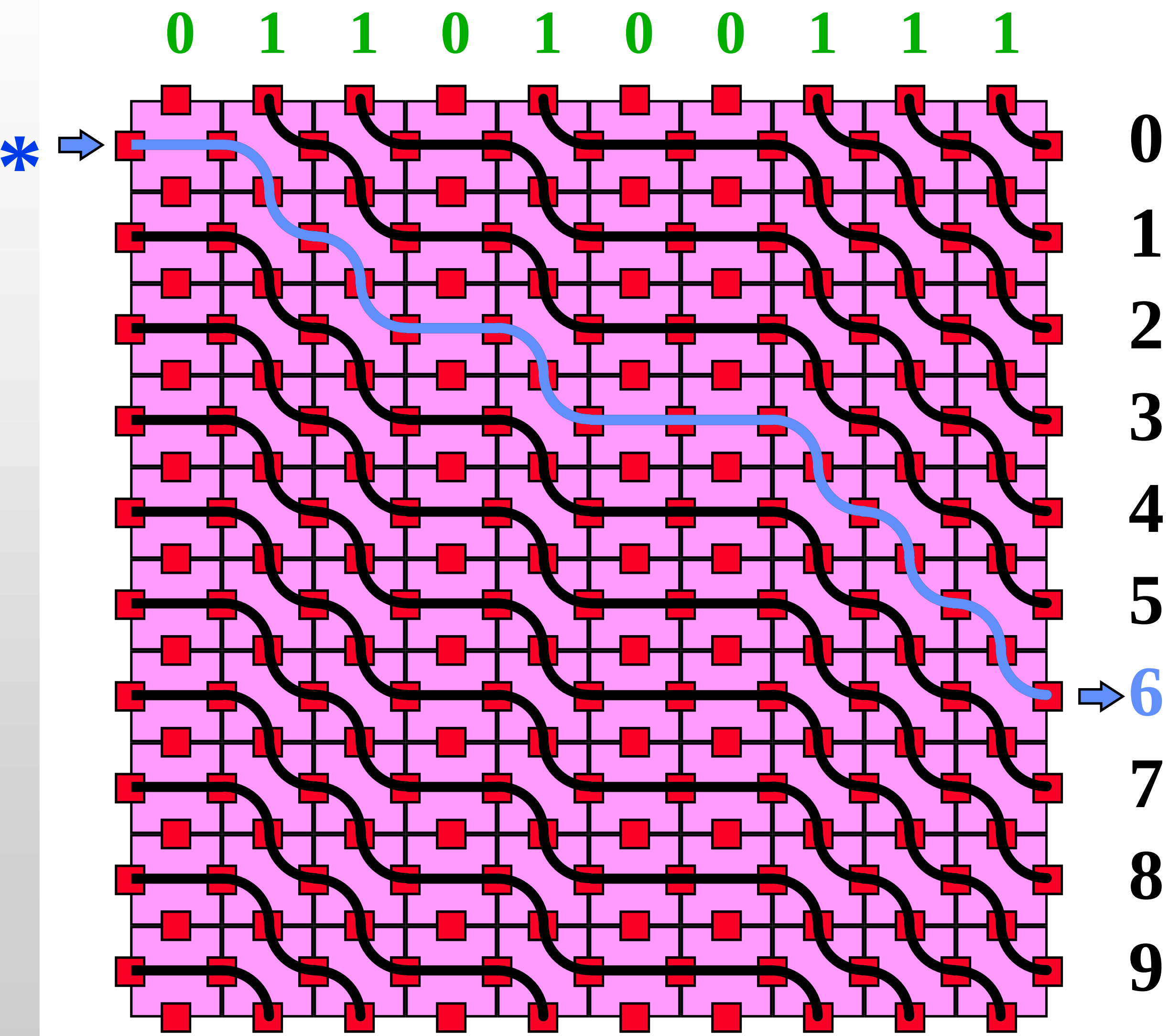
global OR



Time: O(1) on RM

-- Ω(log n) on EREW-PRAM

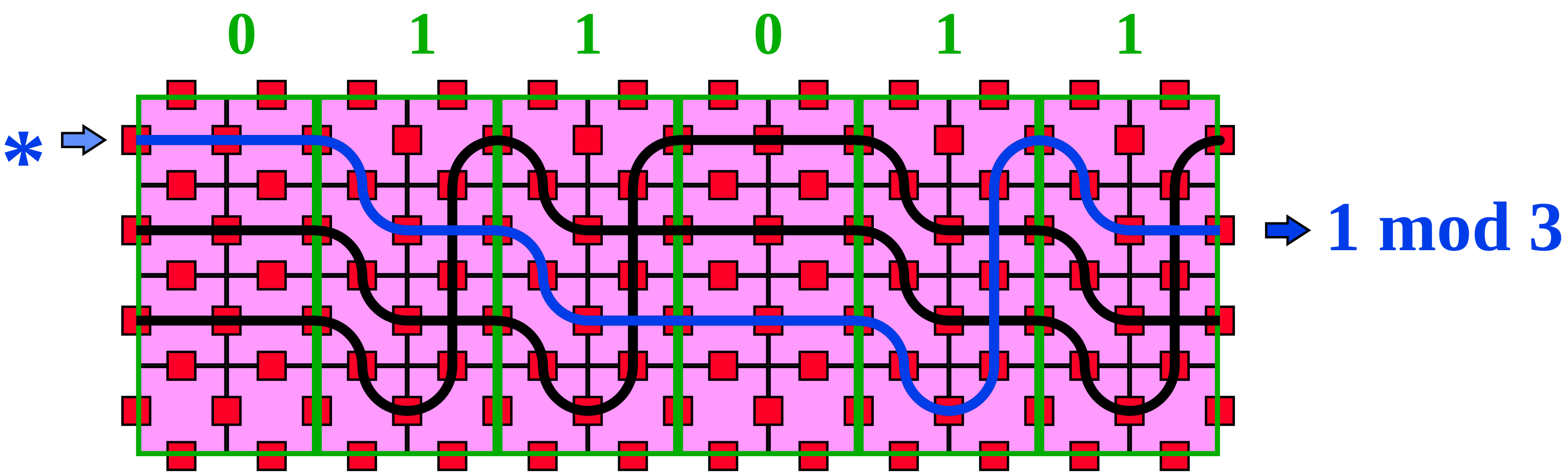
Prefix sum



Time : O(1)
Area: Ω(nxn)

Fast but expensive

Modulo 3 counter

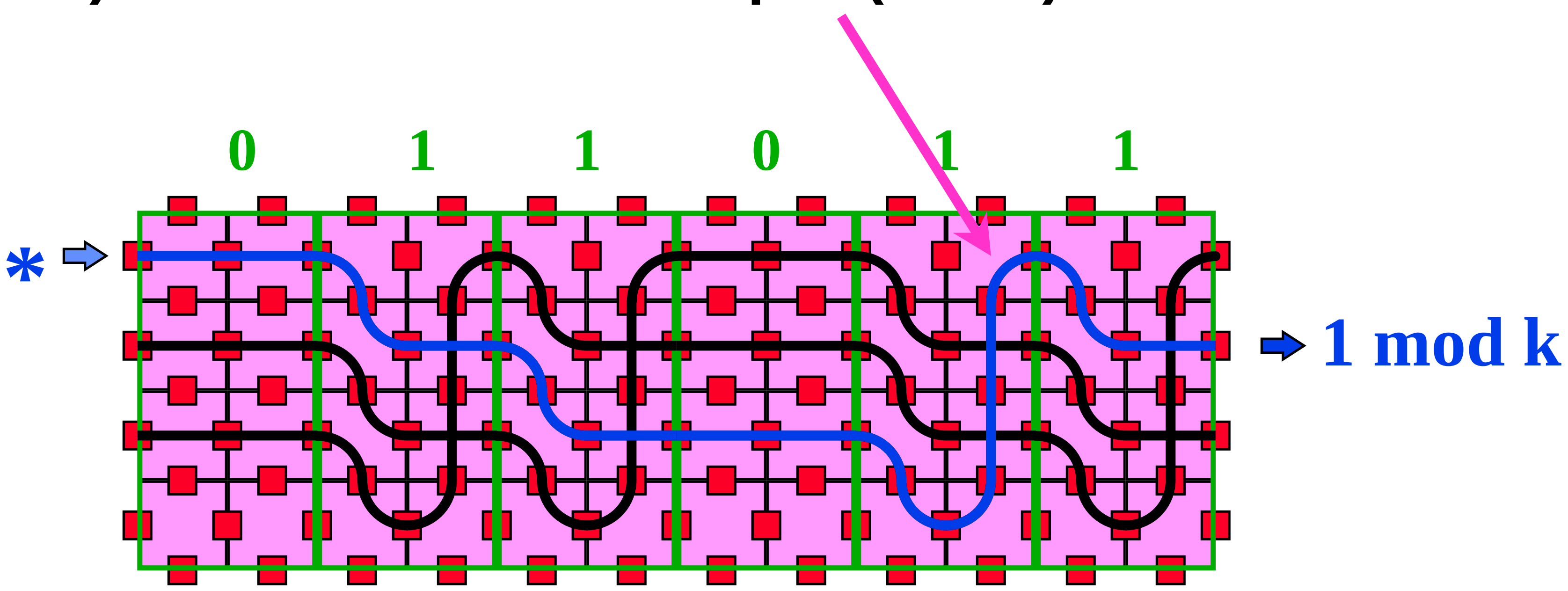


Time: O(1) on RM

Ω(log n / log log n) on CRCW-PRAM

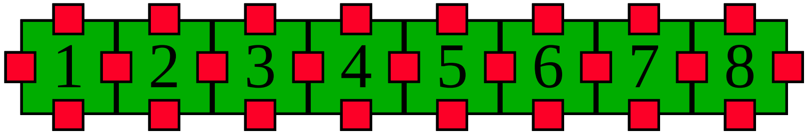
modulo k^2 counter (ranking)

- 2 digit numbers to the basis of k represent all numbers smaller than k^2.
- 1.) determine x mod k (=lsd)
- 2.) count number of “wraps” (=msd).



--> modulo k^2 counting in 2 steps on a k x k^2 array

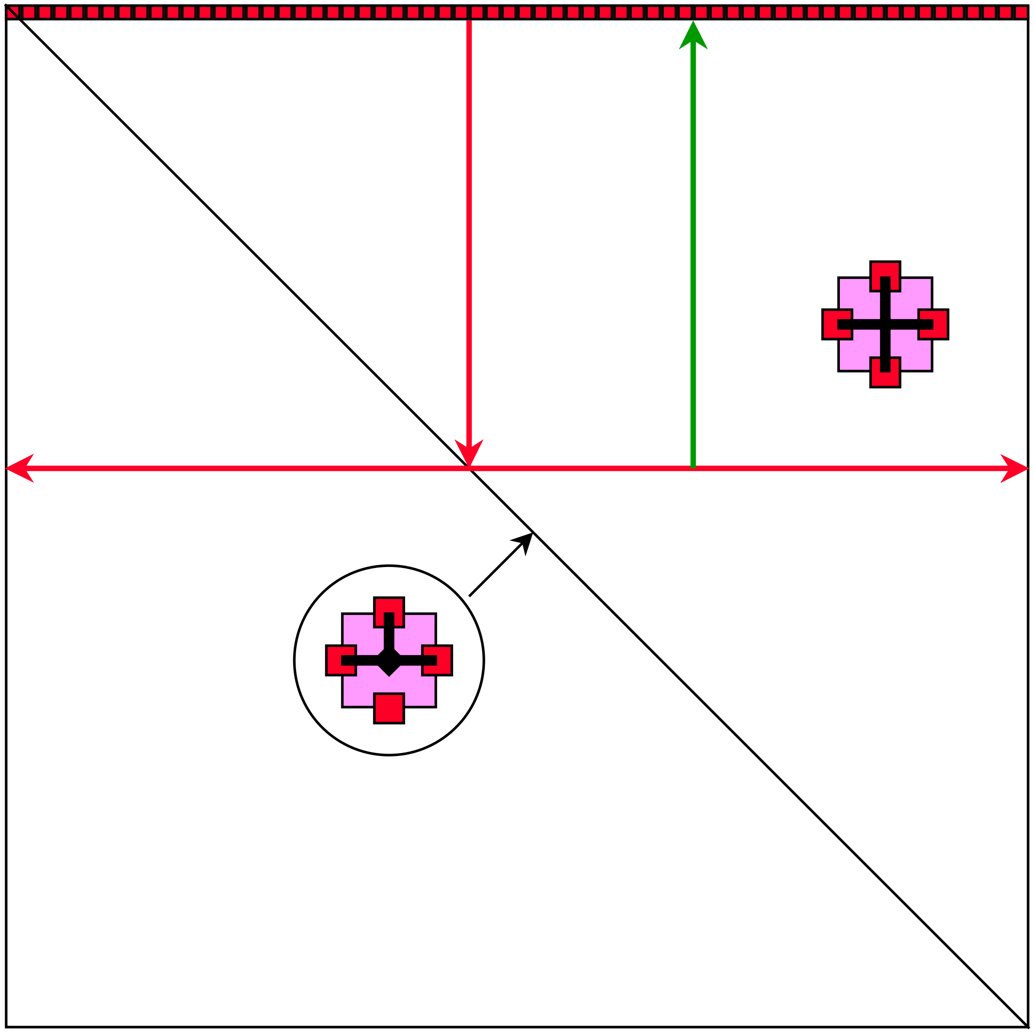
enumeration / prefix sum



time: O(log n)

wire efficiency ! -- (compared with tree)
1/2 number of processors

permutation routing - 2 steps

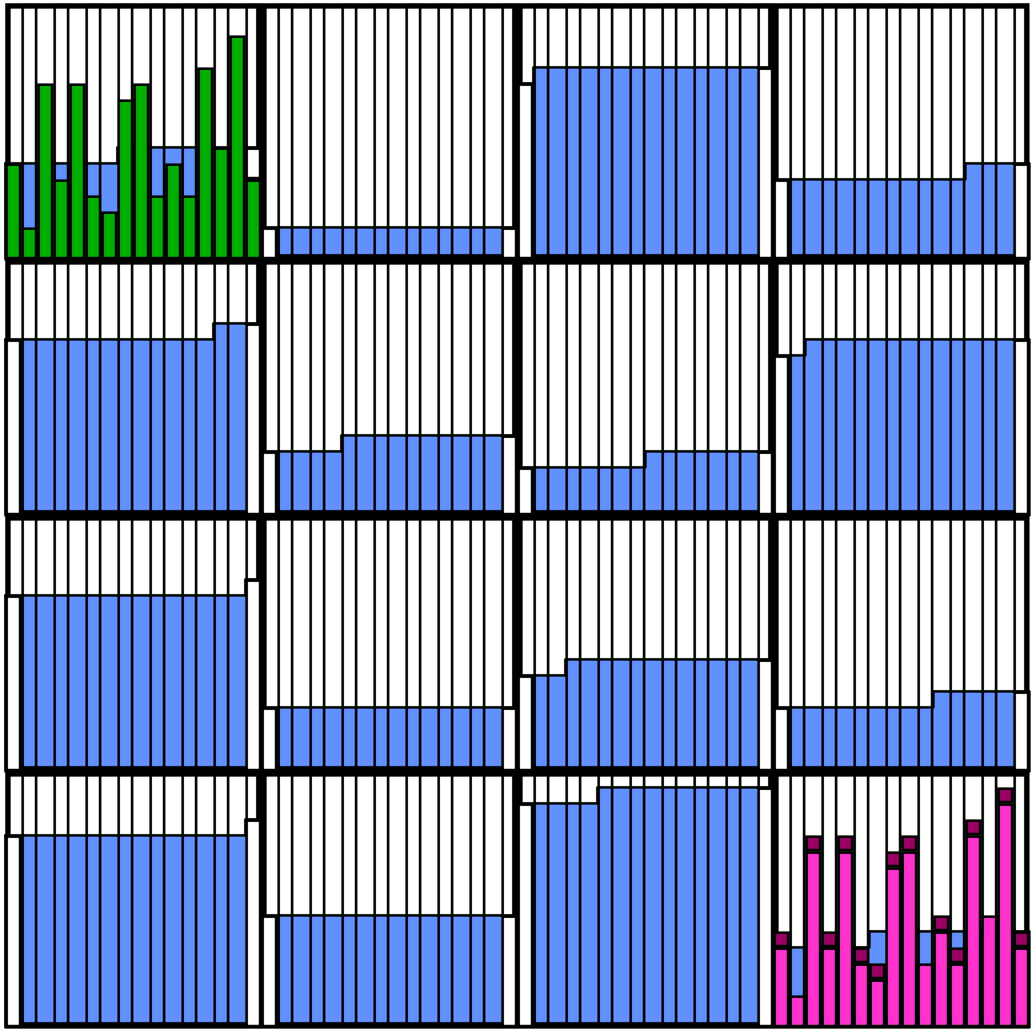


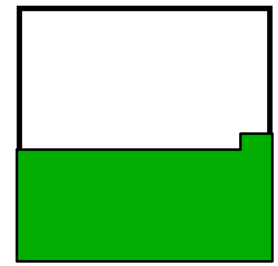
n x n

2 steps !!!

PÅRC

Kunde's all-to-all mapping





Sorting:

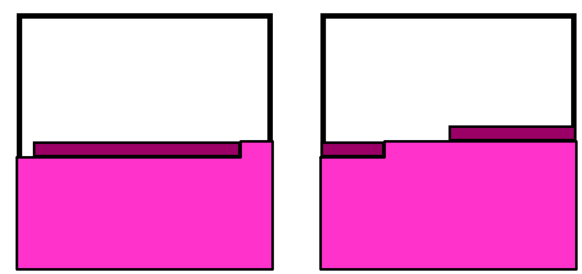
sort blocks

all-to-all (columns)

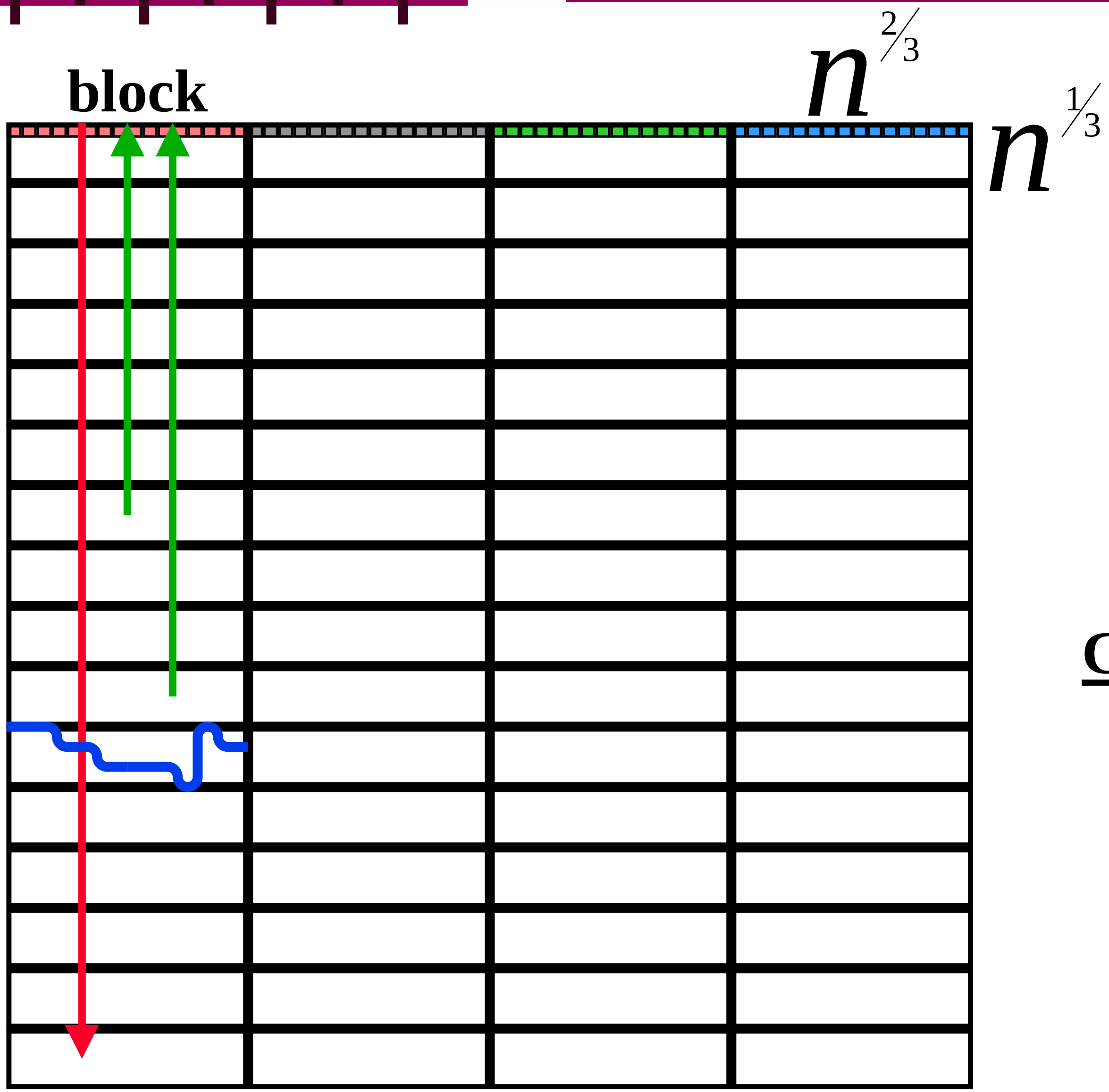
sort blocks

all-to-all (rows)

o-e-sort blocks



sorting in constant time



Sort blocks:

broadcast (1)

rank (2)

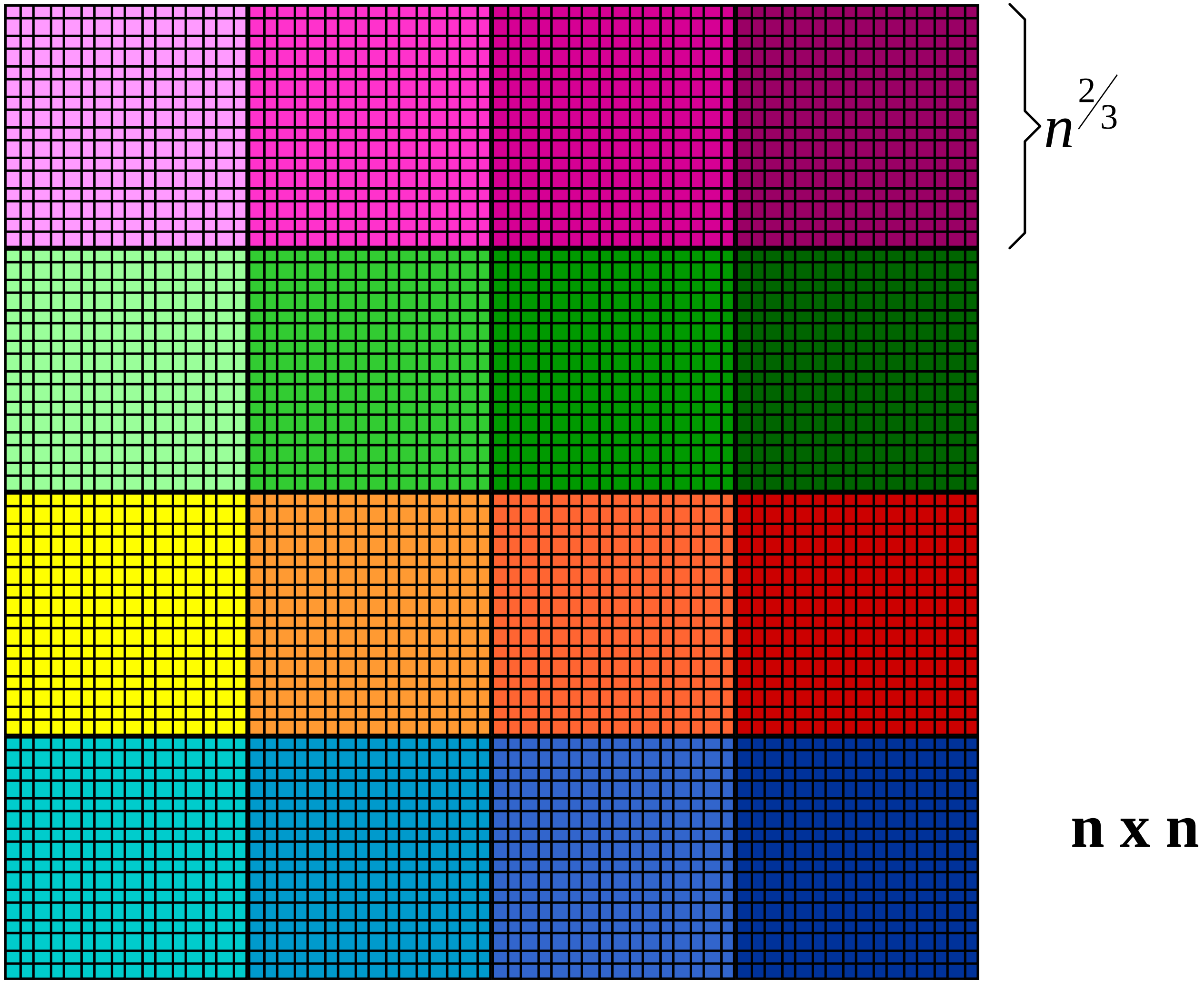
broadcast (1)

Complete sort:

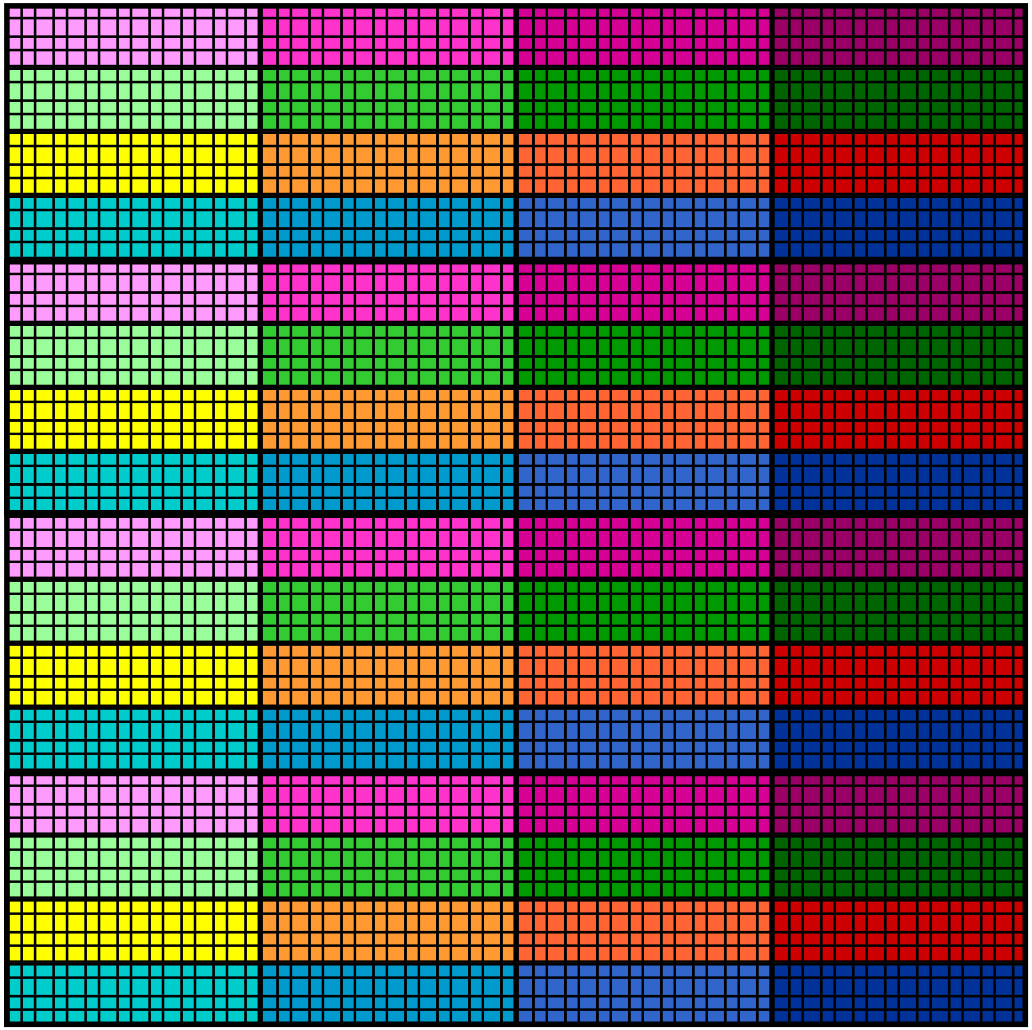
sort blocks
all-to-all (2)
sort blocks
all-to-all (2)
o-e-sort blocks

- better than PRAM --- but useless!!!

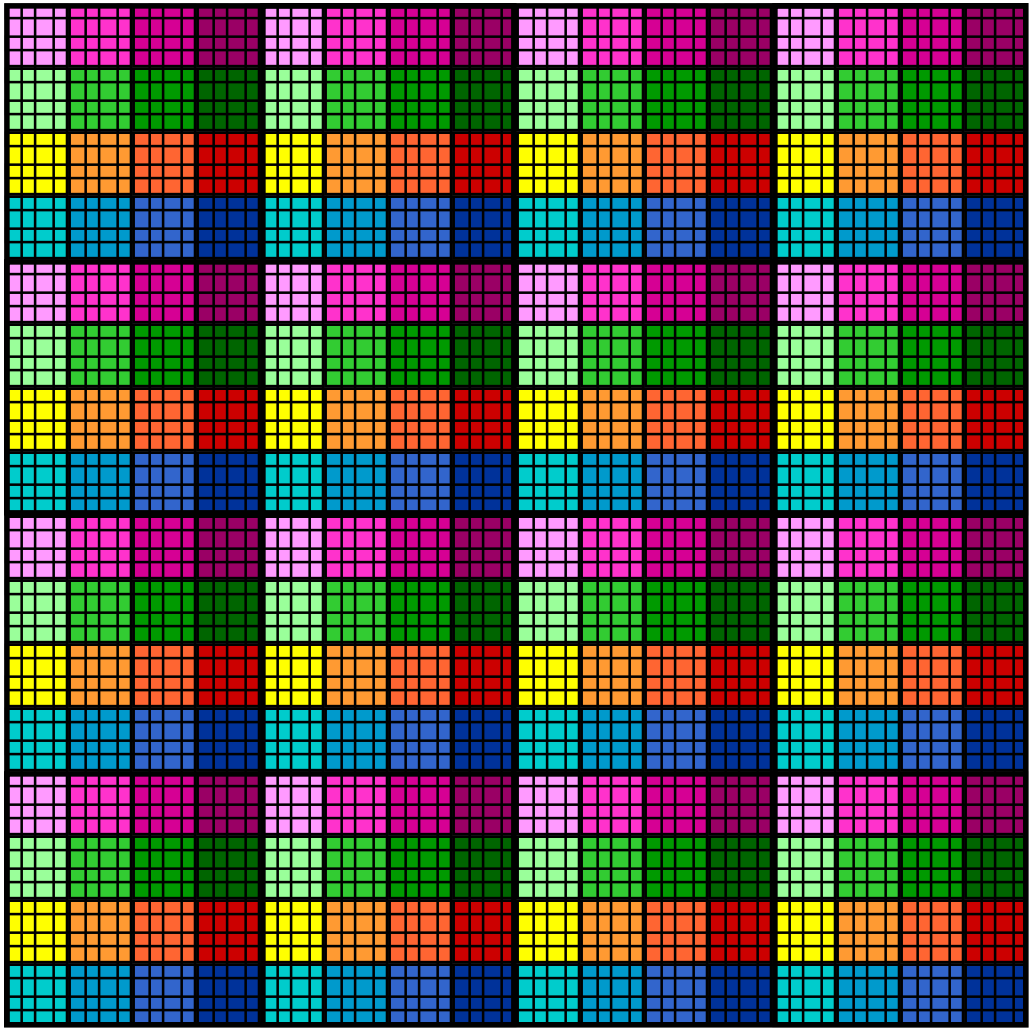
Kunde's all-to-all mapping



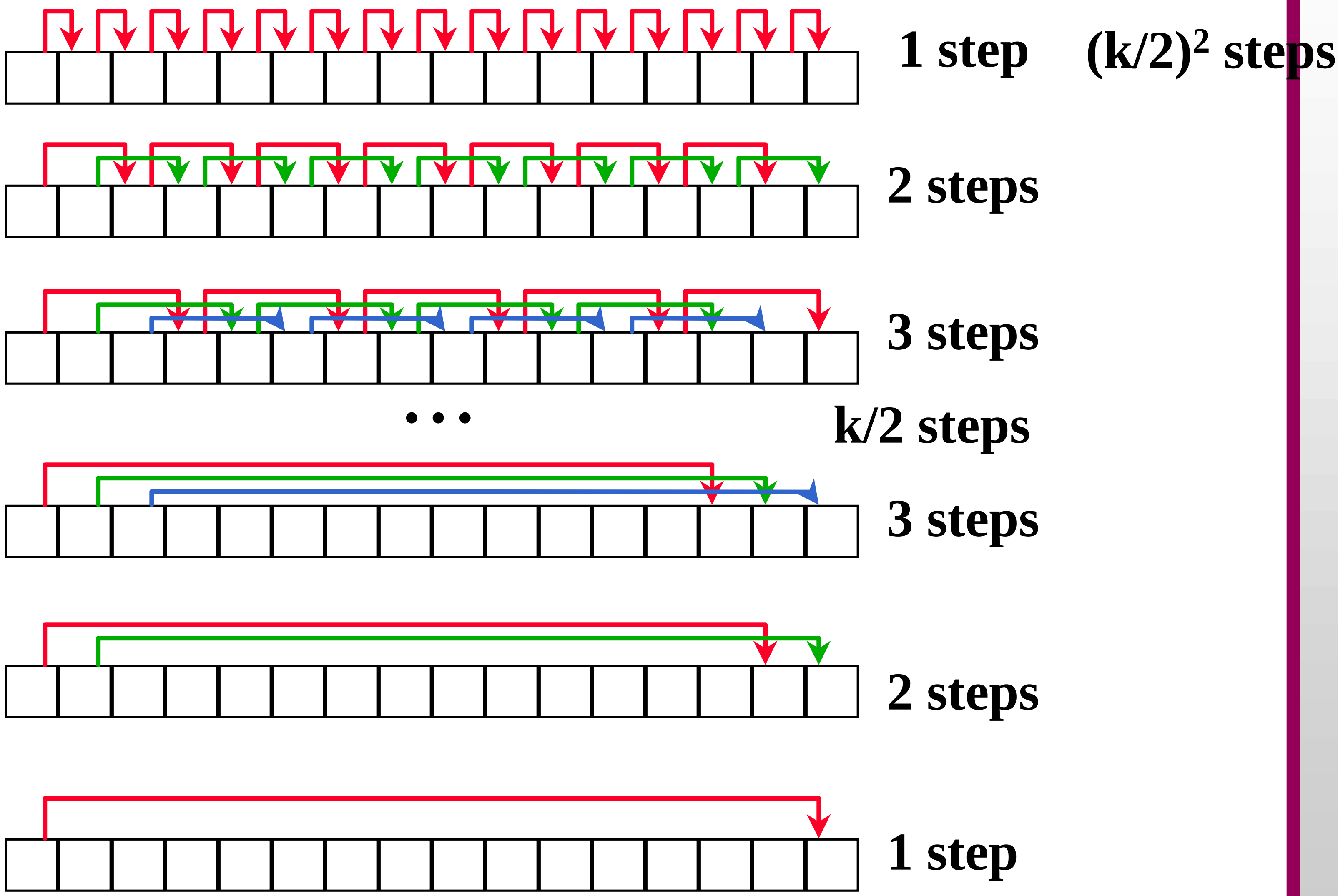
vertical all-to-all



horizontal all-to-all

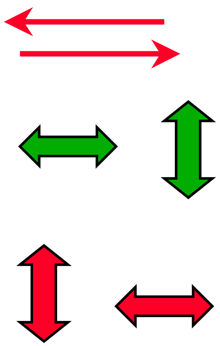


Use of bus



sorting in optimal time
Kunde / Schröder

$(k/2)^2$ steps
$k=n^{1/3}$
each step takes $n^{1/3}$ time
--> $T= n/4$



x 2

x 2

/2

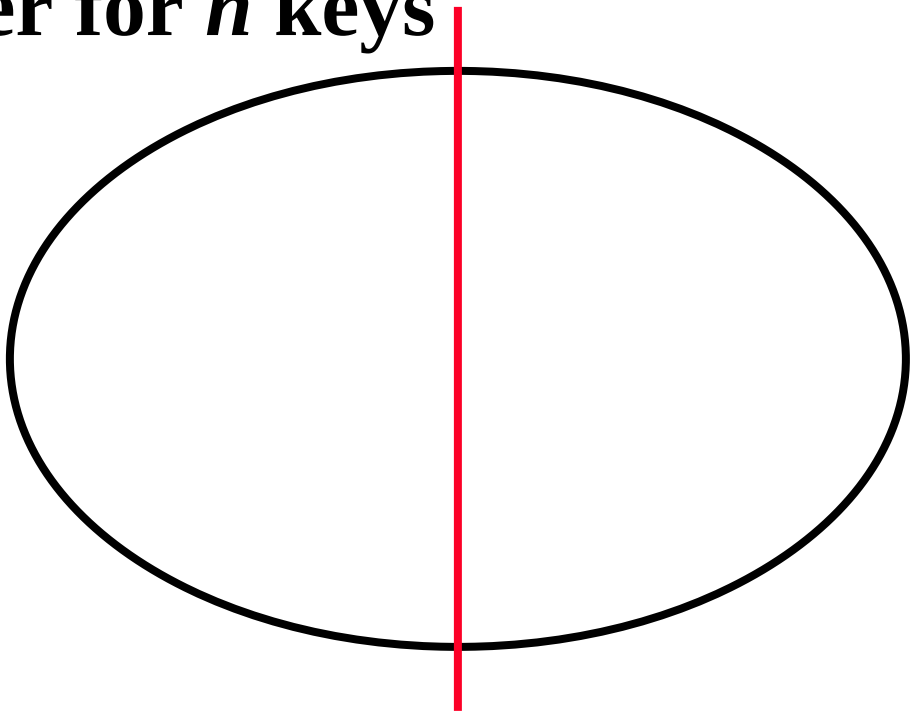
$$T_{\text{all-to-all}} = n/2$$

Sorting:
sort blocks $(O(n^{2/3}))$
all-to-all $(n/2)$
sort blocks $(O(n^{2/3}))$
all-to-all $(n/2)$
sort blocks $(O(n^{2/3}))$

time: $n + o(n)$

Why optimal?

Sorter for *n* keys



Bisection of data with *k* wires

→ **Sorting time > n/k**

PARC

Use of theorem

1.) n **keys on a** ***kxk*** **RM:**

Time >= ***n/k***

Proof:

Wherever the data is stored there is always a bisection of length ***k***

-- this can be demonstrated sweeping left right through the array.

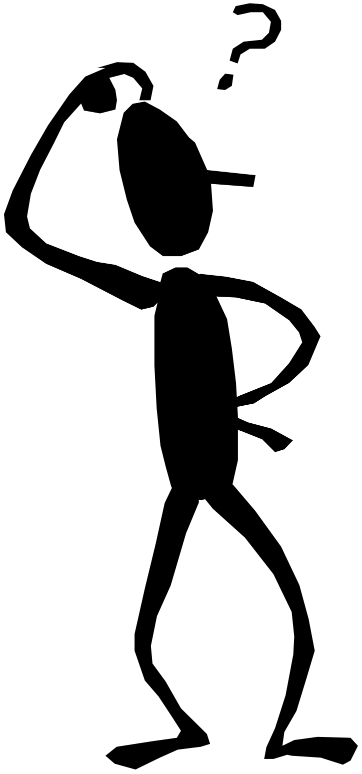
Q.e.d.

2.) nxn **keys on an** ***nxn*** **RM:**

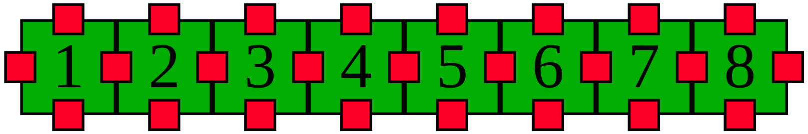
Time >= ***n.***

Proof: trivial

Optimal --- but ...



enumeration / prefix sum

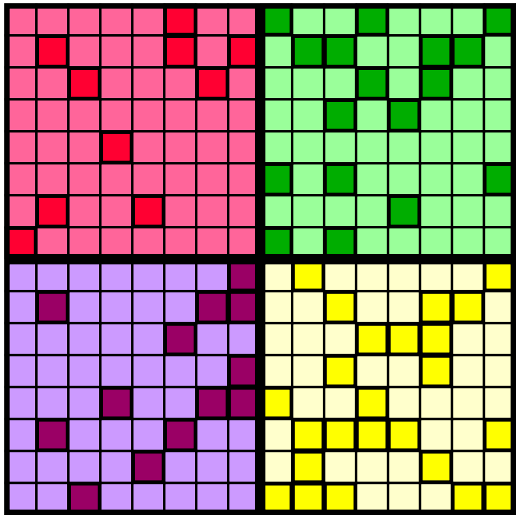


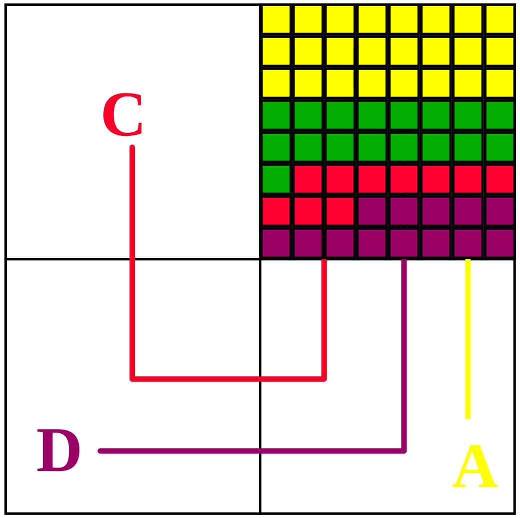
time: O(log n)

wire efficiency ! -- (compared with tree)
1/2 number of processors

ABCD-routing

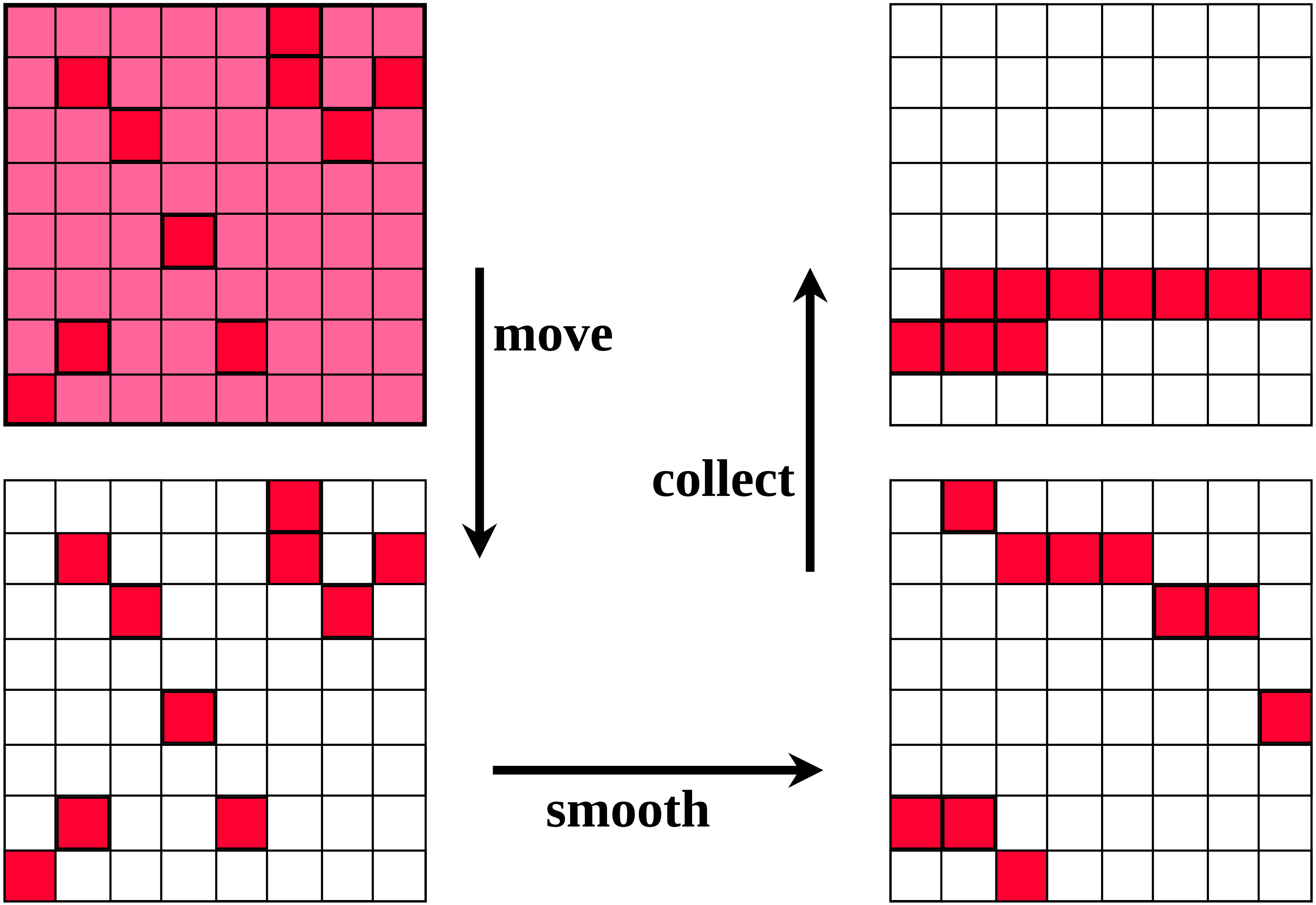
- move and smooth



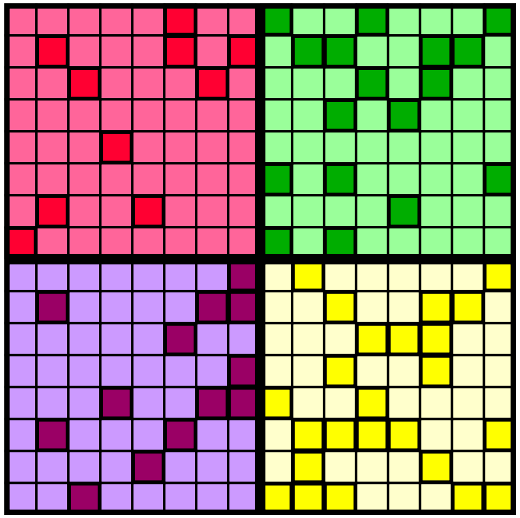


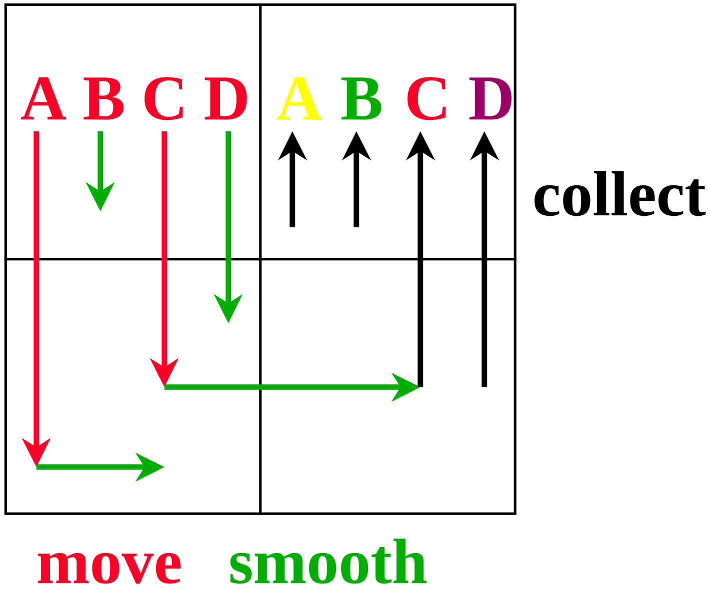
Row-major enumeration of A, B, C and D packets within each quadrant in time 4 log n. Determine destination position of each packet.

elementary steps



time analysis

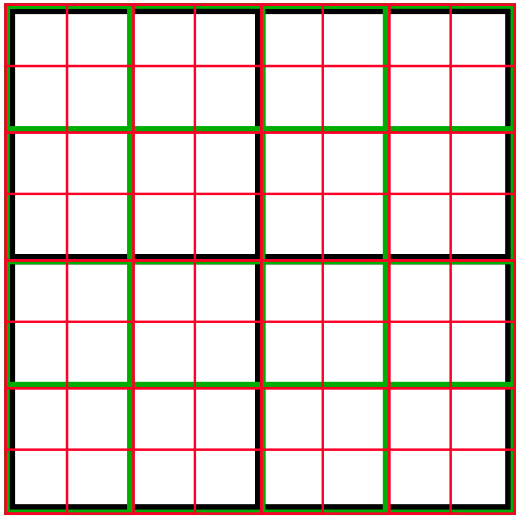




time: 3 x n/2

T=3n+o(n)

T < 2n



4 destination squares
time: 3n + 4 log n

16 destination squares
time: 2n + 16 log n

64 destination squares
time: 12/7 n + 64 log n

mesh-diameter: 2n

enough of routing/sorting

Constant factor !

Can we do better ?

What kind of problems ?

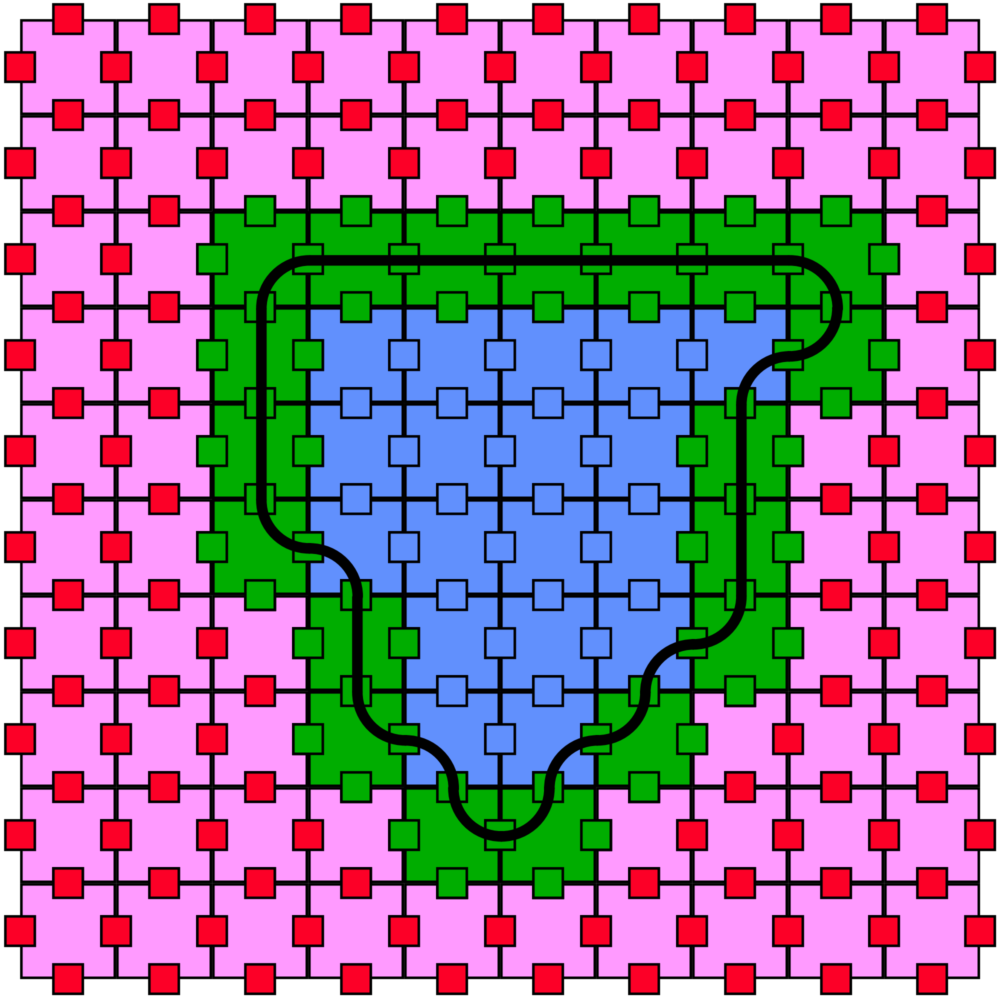
Image processing

Sparse problems !

Image processing

- Border following
- Edge detection
- Component labeling
- Skeletons
- Transforms

Component labelling



Object

Define border (candidates)

Set bus

While own label is not received:

1.) Candidates brake bus and send their label

a) clockwise

b) anti-clockwise

2.) Candidates switch off and restore bus if they see smaller label

Time: O(1) -- O(log n)

Transforms

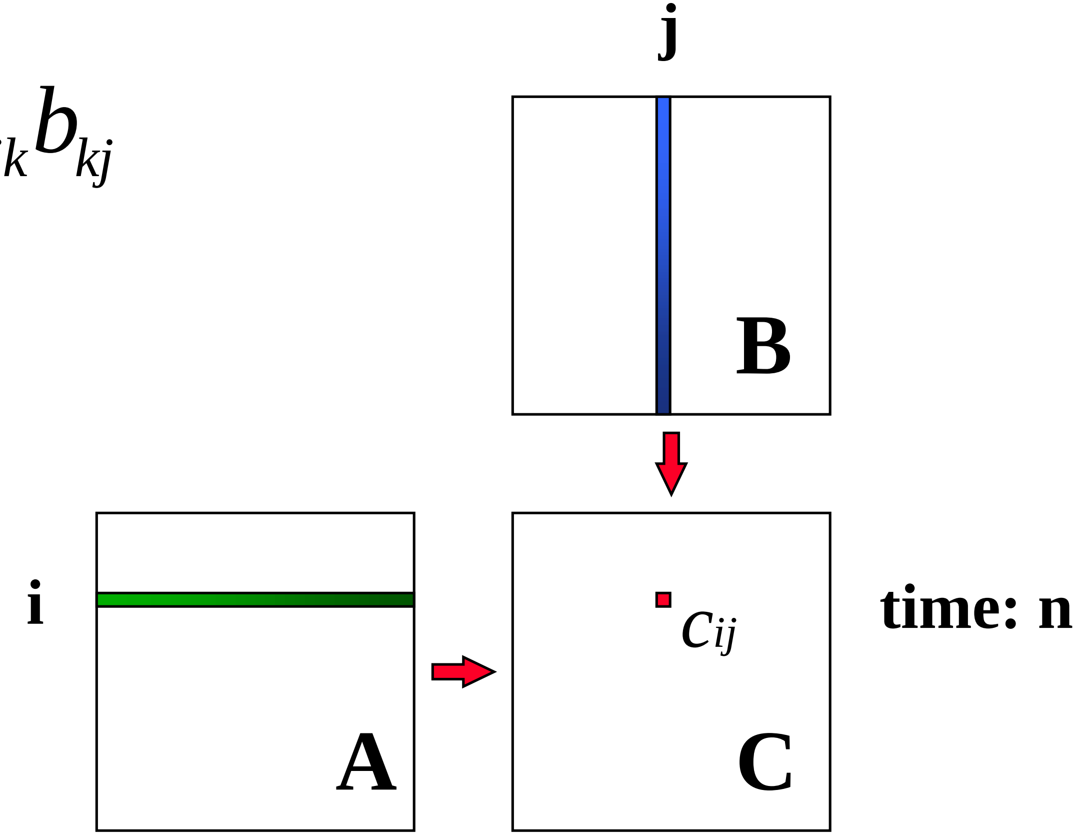
- **Wavelet transform: Time *log n* on RM**

  -- time *n* on mesh
- **FFT: Time *n* on RM and mesh**
- **Hough transform: Time *m x log n* on RM**

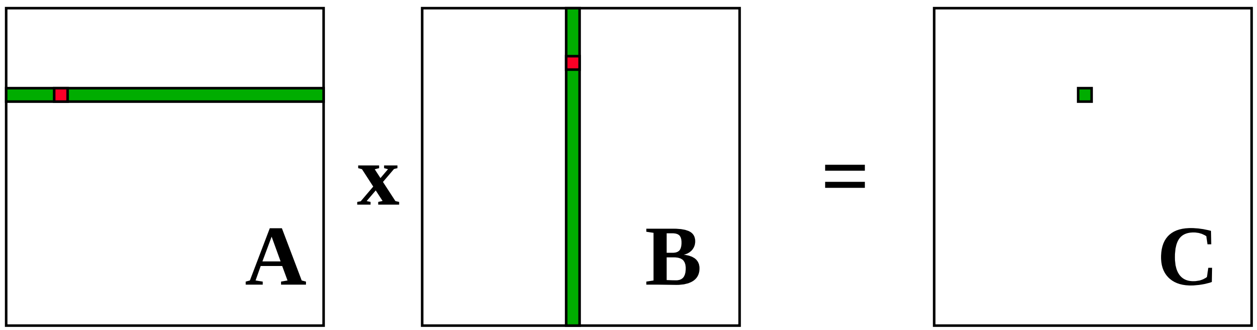
  -- time *m x n* on mesh

systolic matrix multiplication

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$



sparse matrix multiplication



Time: n (nxn mesh)

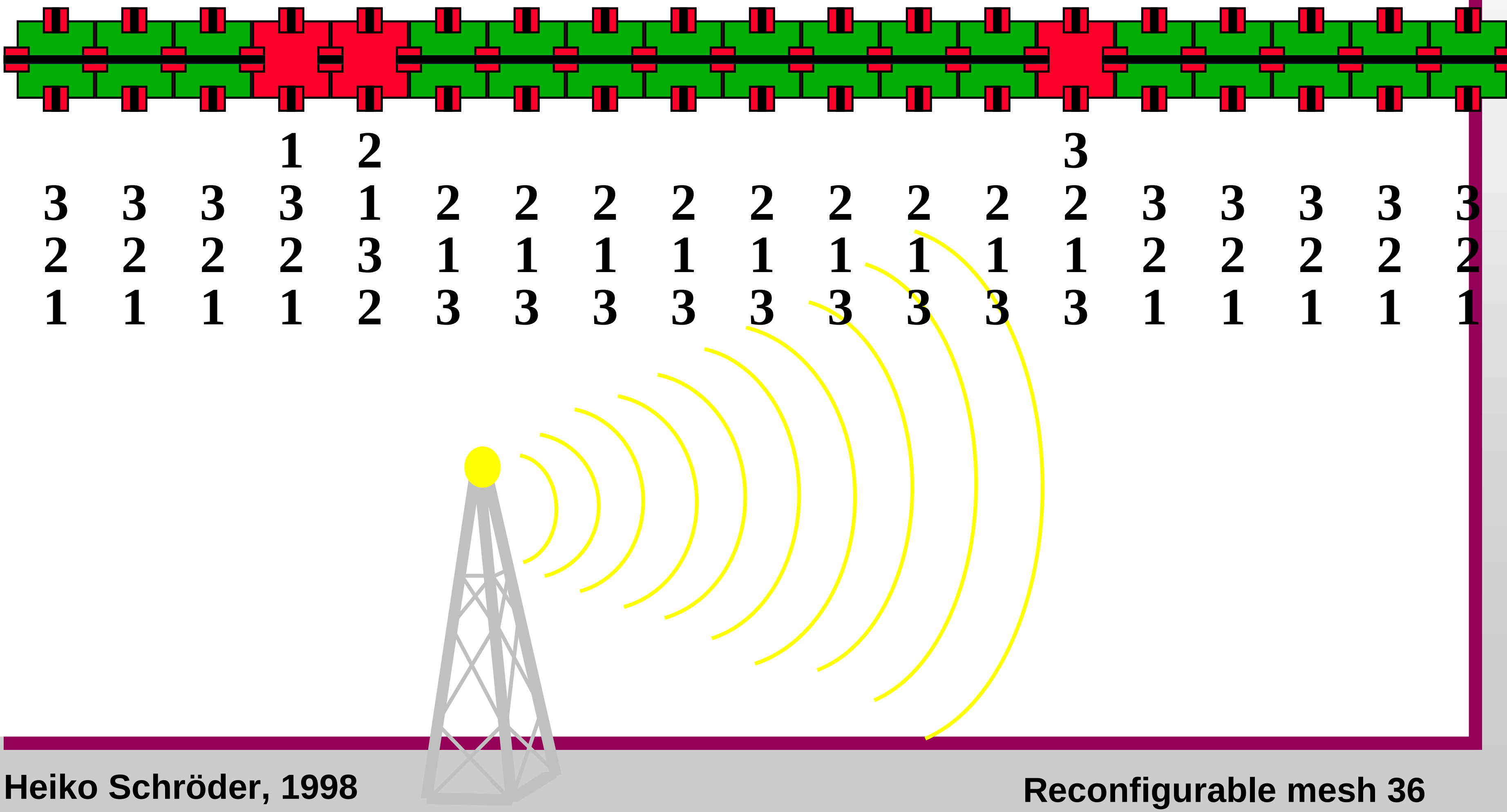
$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

A and B column sparse (k^2)
A and B row sparse (k^2)
A row sparse, B column sparse (k^2)
A column sparse, B row sparse $\left(k\sqrt{n}\right)$

PÅRC

unlimited bus length

- ring broadcast



A row-sparse   B column-sparse

Repeat k times

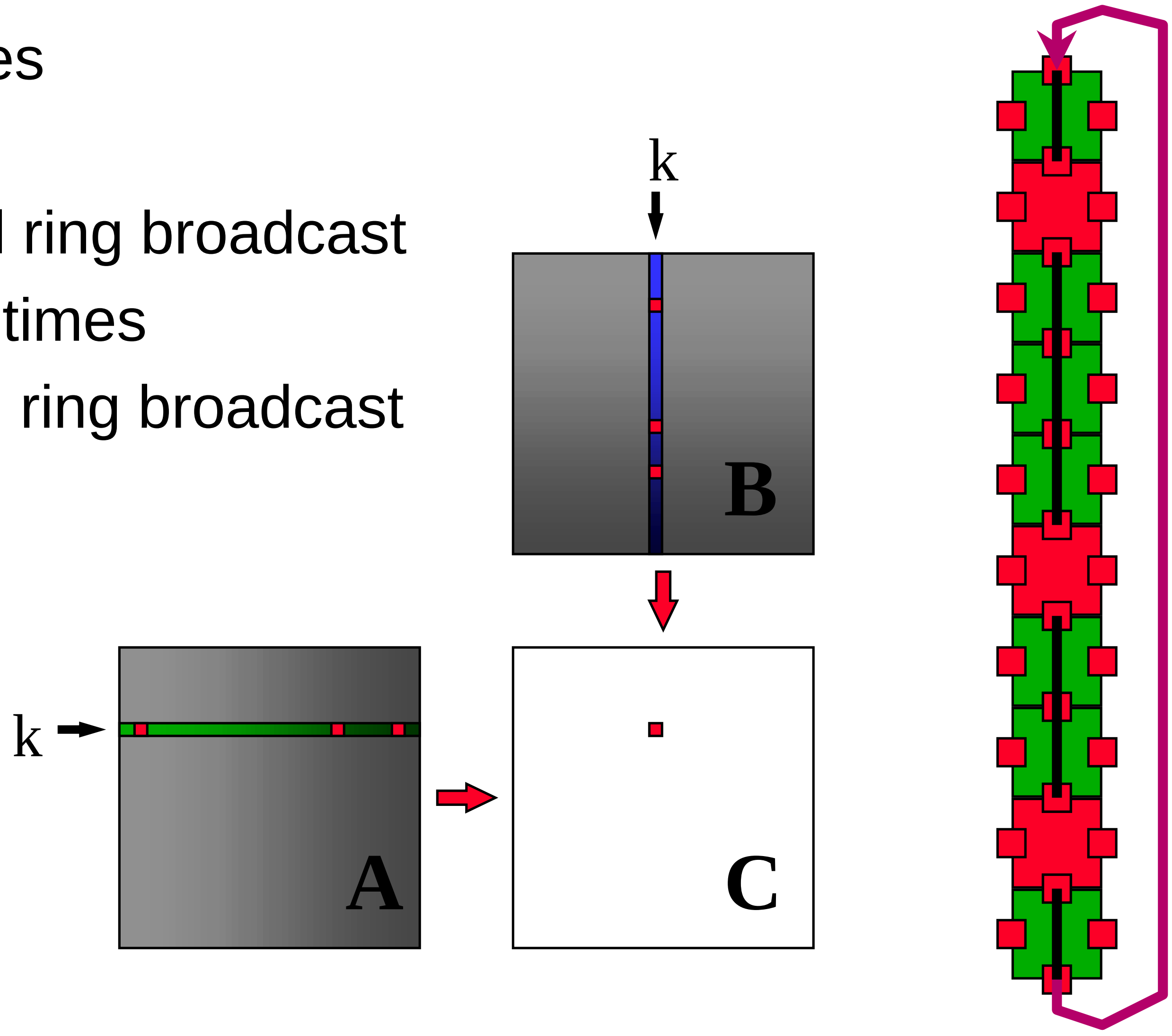
Begin

horizontal ring broadcast

Repeat k times

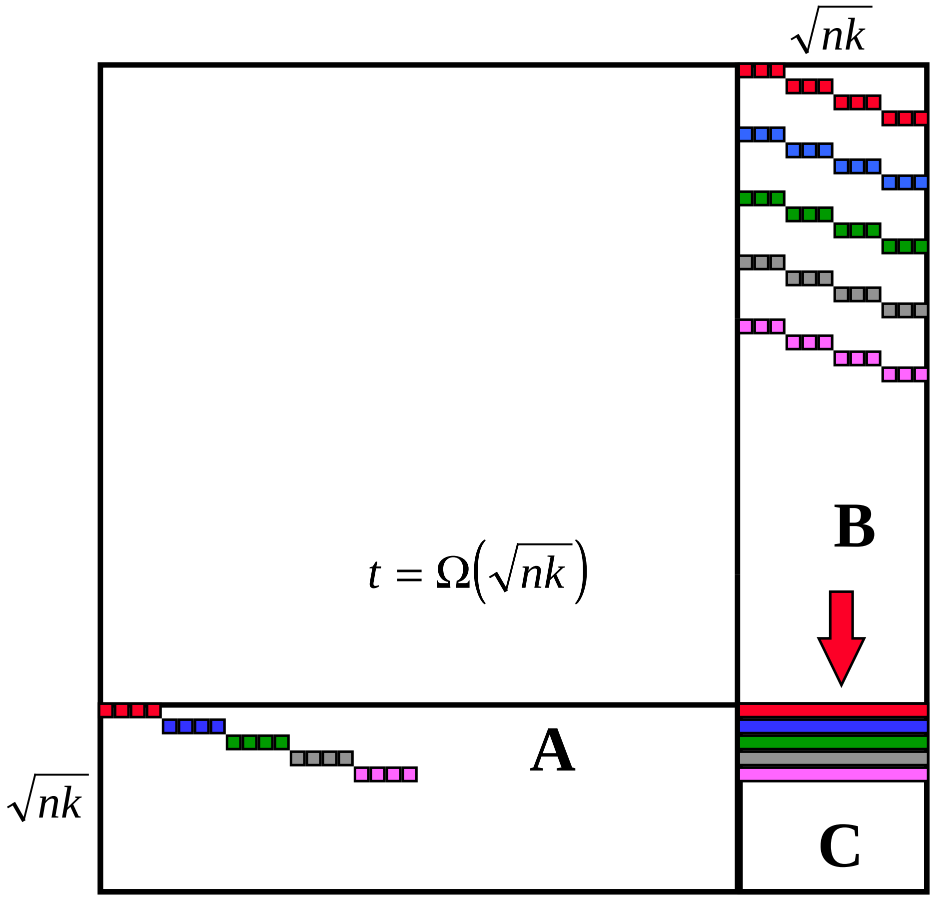
vertical ring broadcast

End.



lower bound (c,r)

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$



k=3

n=48

splitting the problem

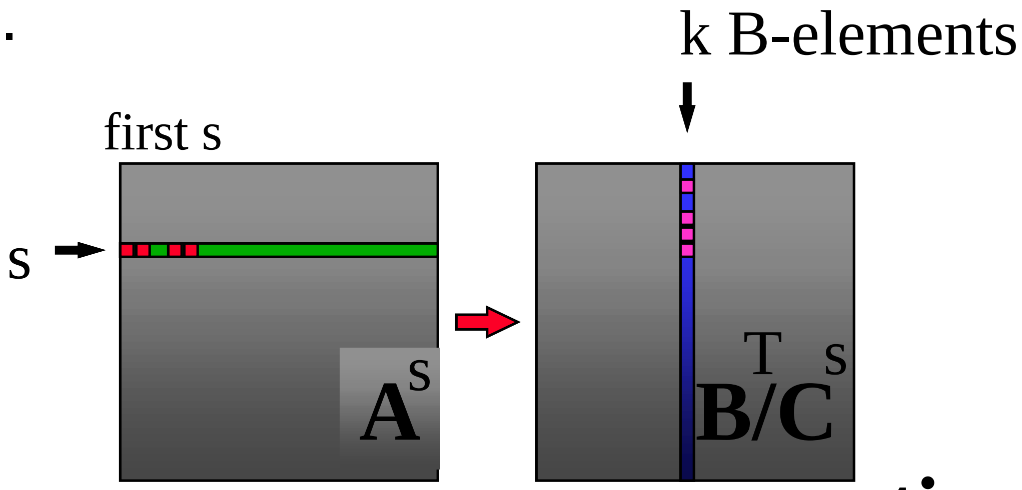
Repeat k times

- Begin
  - vertical ring broadcast
  - Repeat s times
    - horizontal ring broadcast
- End.

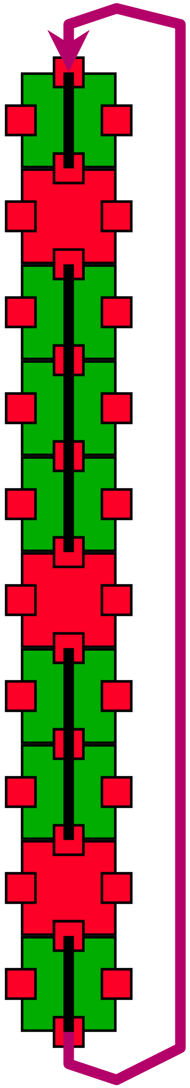
$A=A^s+A^r$

$C=A^sB+A^rB$

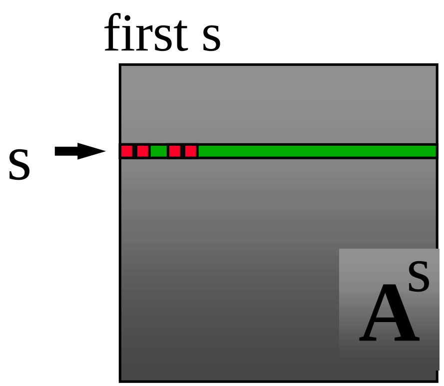
$\mathbf{C=C^s+C^r}$



time: ks



CR

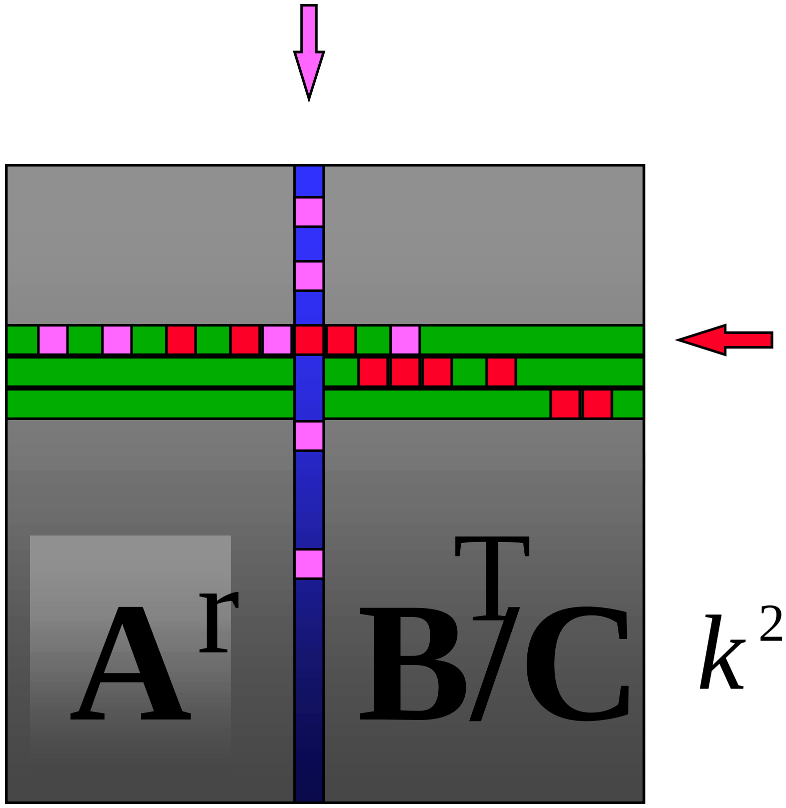


$A = A^s + A^r$

A has nk non-zero elements →
A^r has at most nk/s non-zero rows →
for s= √n A^r has at most k √n non-zero rows.

A^s B is a CC- problem → it takes time k √n .

k B-elements

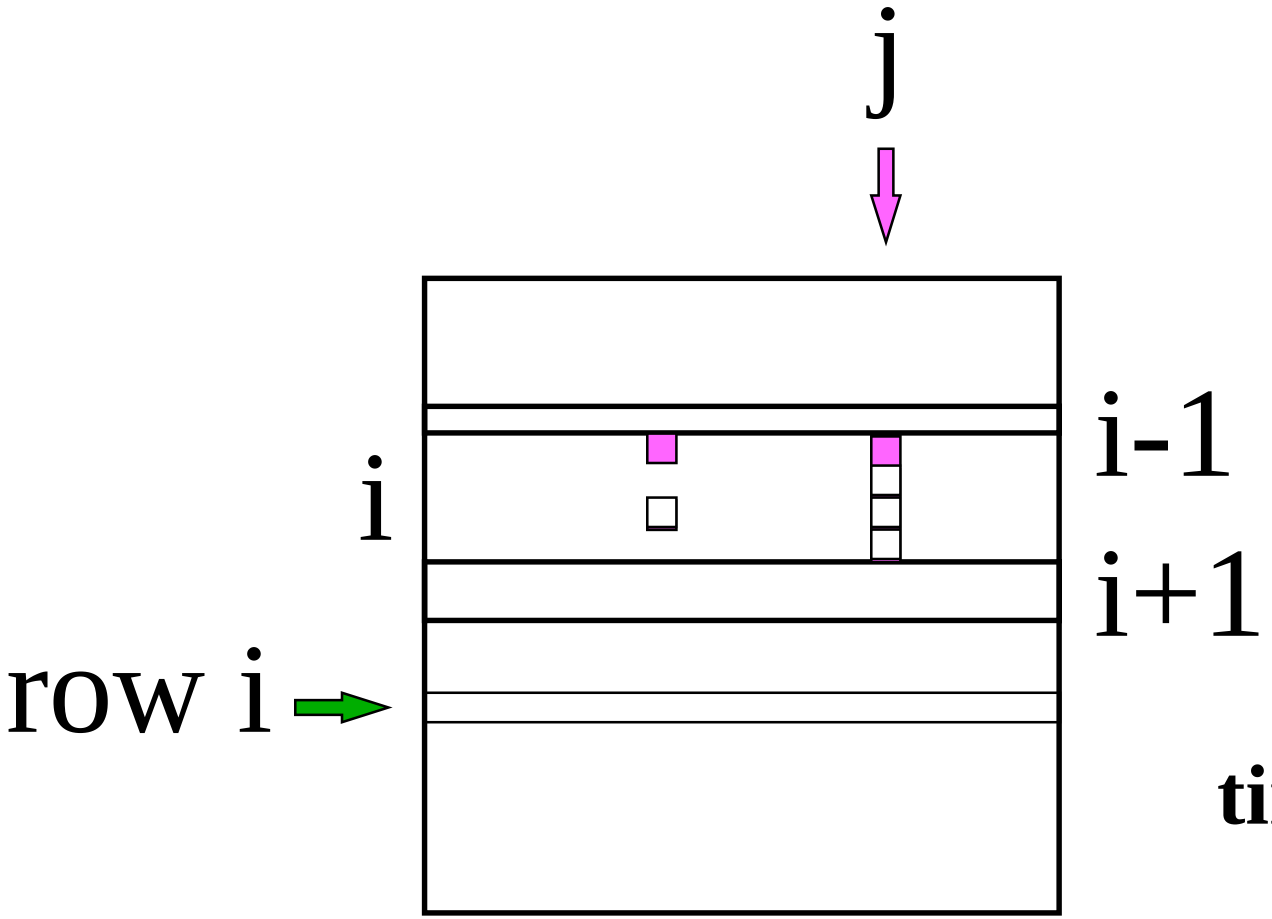


k A-elements

time: k^2

$k^2 n$ **elements**

column sum

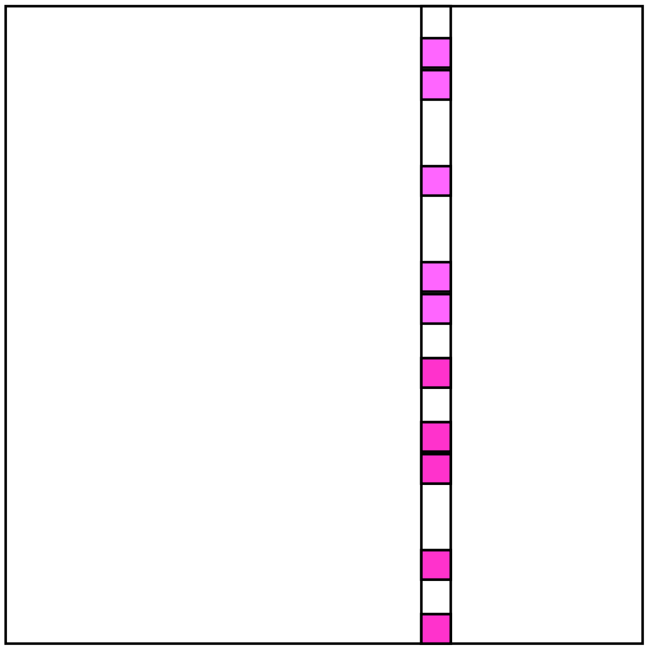


time: log n

only $k\sqrt{n}$ elements per column

rout **time:** $k\sqrt{n}$

routing within columns



rout **time:** $k\sqrt{n}$

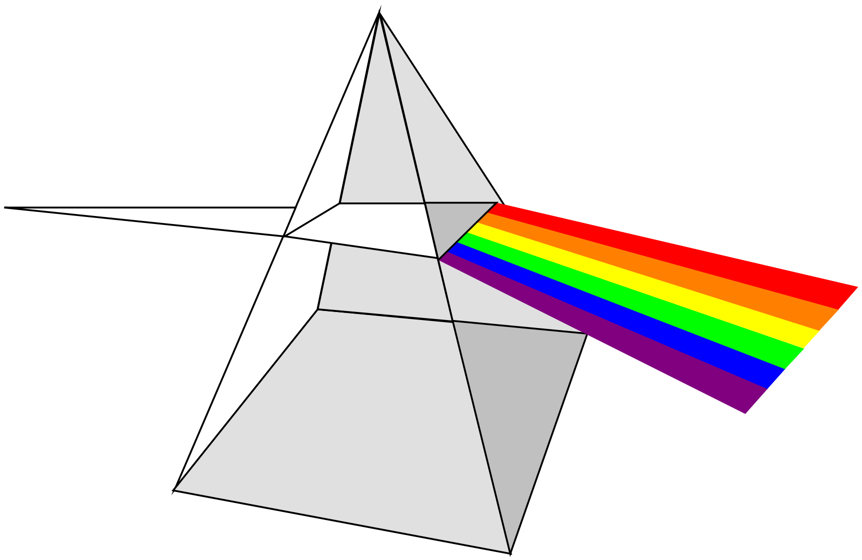
Reconfigurable mesh ?

constant diameter !

Physical laws!

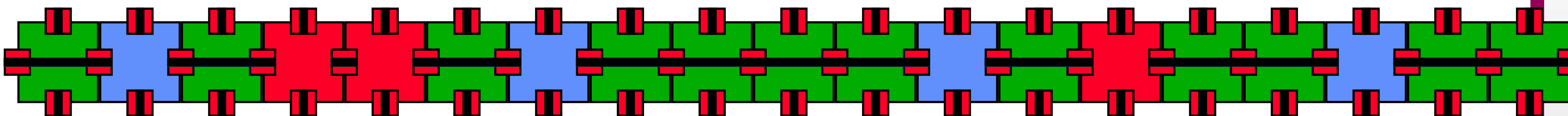
Physical limits

c=300 000 km/sec



- 30cm/ns
- on chip: 1cm/ns
- **--> bounded bus length**

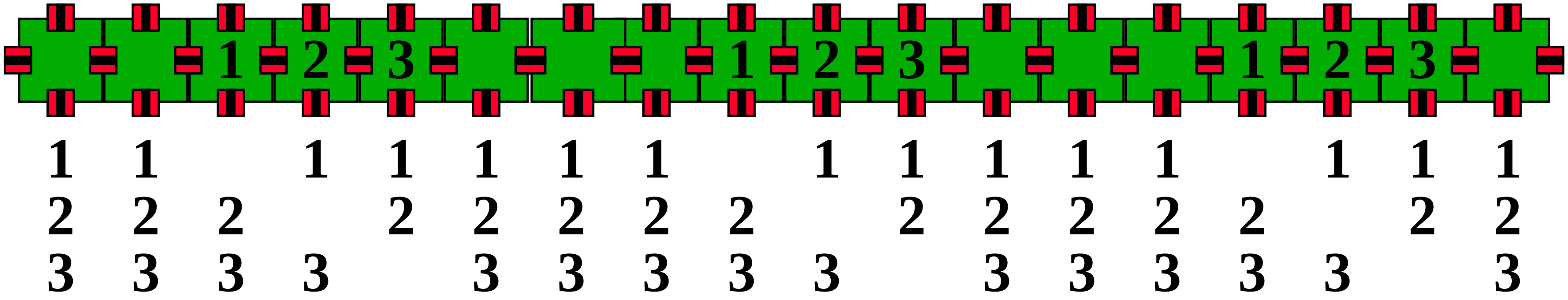
bounded broadcast



			1	2									3					
				1	2	2								3	3	3		
3	3				1	1	2	2	2	2	2						3	3
		3	3				1	1	1	1	1	2	2					
				3								1	1	2	2	2		
2	2				3	3								1	1	1	2	2
1	1	2	2				3	3	3	3	3						1	1
		1	1	2								3	3					

time: k + n/l

creating main stations



time: k

A row-sparse   B column-sparse

Create main stations 1,…,k for A and B (time: n/l+k)

For i=1,…,k do

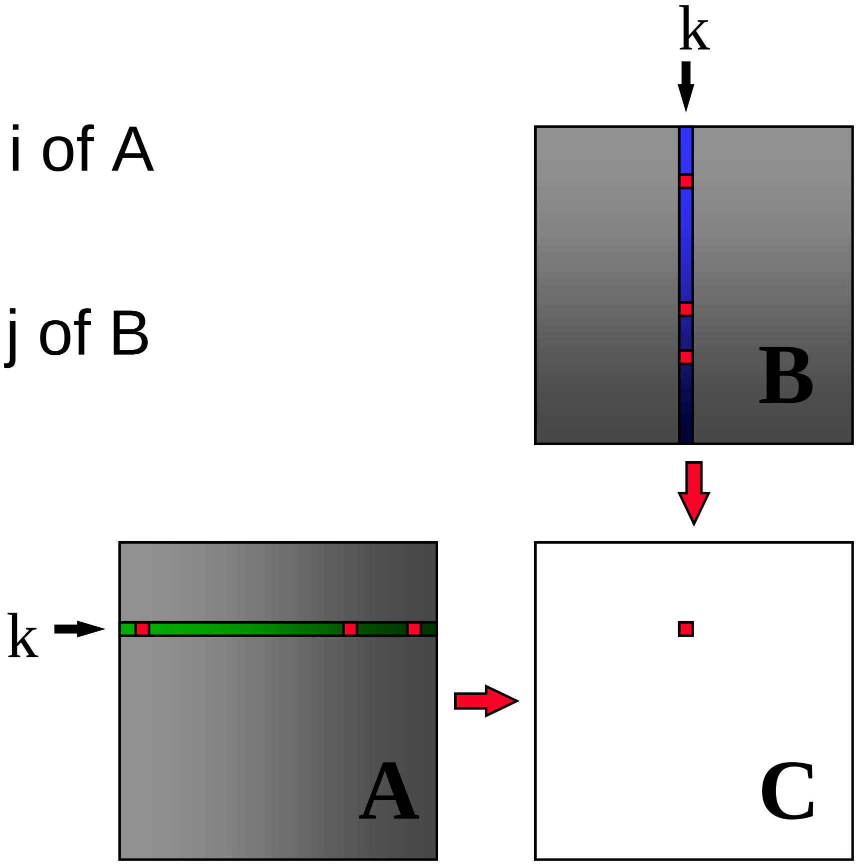
  Begin

    horizontal ring broadcast i of A

    For j=1,…,k do

      vertical ring broadcast j of B

  End.



A and B column-sparse

Create main stations 1, … , k for A (time: n/l+k)

For i=1,…,k do

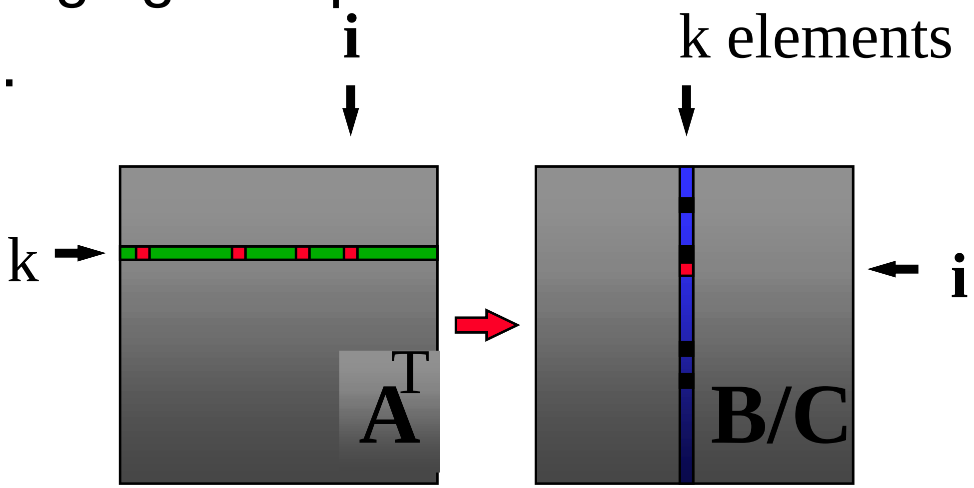
Begin

horizontal ring broadcast i

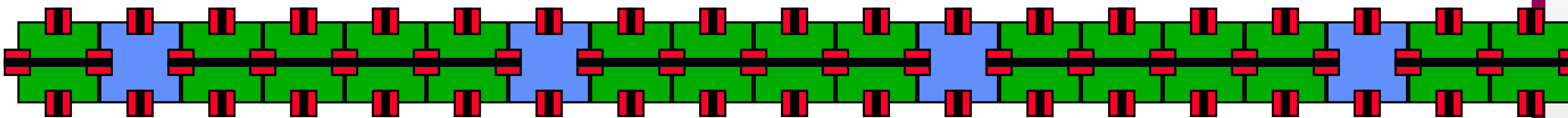
k bounded vertical broadcasts of products

merging new products

End.



remove minor stations



			1	2									3					
				1	2	2								3	3	3		
3	3				1	1	2	2	2	2	2						3	3
		3	3	3	3	3	1	1	1	1	1	2	2	2	2	2		
2	2						3	3	3	3	3	1	1	1	1	1	2	2
1	1	2	2	2	2	2						3	3	3	3	3	1	1

Time: n (nxn mesh)

A and B column sparse (k^2) (k^2+2n/l)

A and B row sparse (k^2) (k^2+2n/l)

A row sparse, B column sparse (k^2) (k^2+n/l)

A column sparse, B row sparse $\left(3k\sqrt{n}\right)$ ($+11n/l$)

(Kunde, Middendorf, Schmeck, Schröder, Turner)

- **image processing**
  - **sorting**
  - **routing**
- **load balancing**

better than the mesh !

(Kapoor, Kunde, Kaufmann, Schroeder, Sibeyn)

- The RM is in some cases "better" than PRAM
- The RM is always at least as "good" as mesh
- The RM is often "better" than the mesh

The End