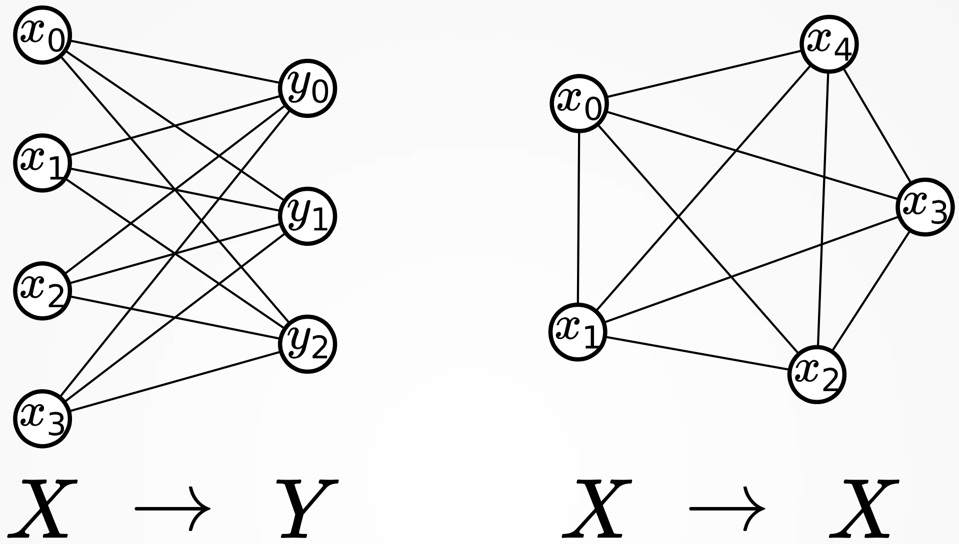
A Locality-Preserving One-Sided Binary Tree – Crossbar Switch Wiring Design Algorithm

Devrim Şahin
Bilkent University

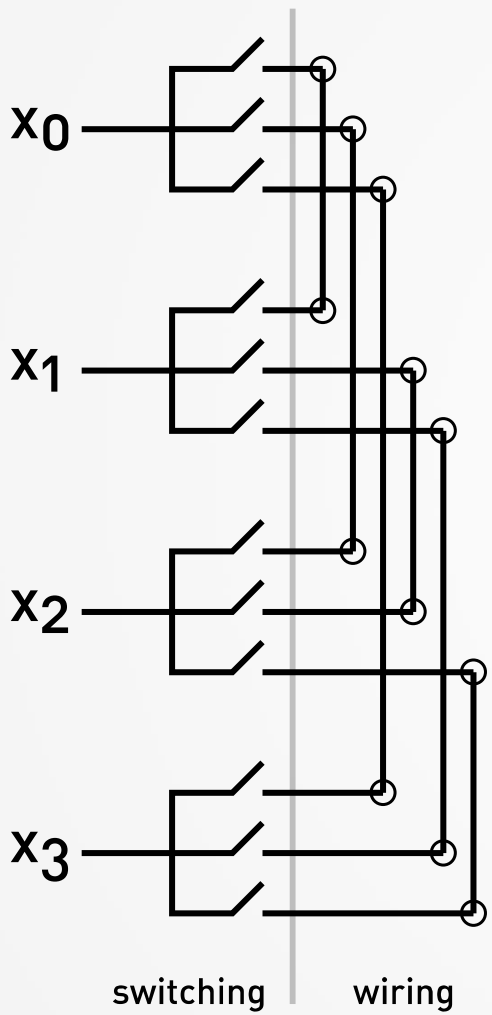
INTRODUCTION



One-sided switches make connections from a set *X* onto itself, instead of another set *Y*

A complete one-sided switch can be represented by a K_n graph

ONE-SIDED SWITCH



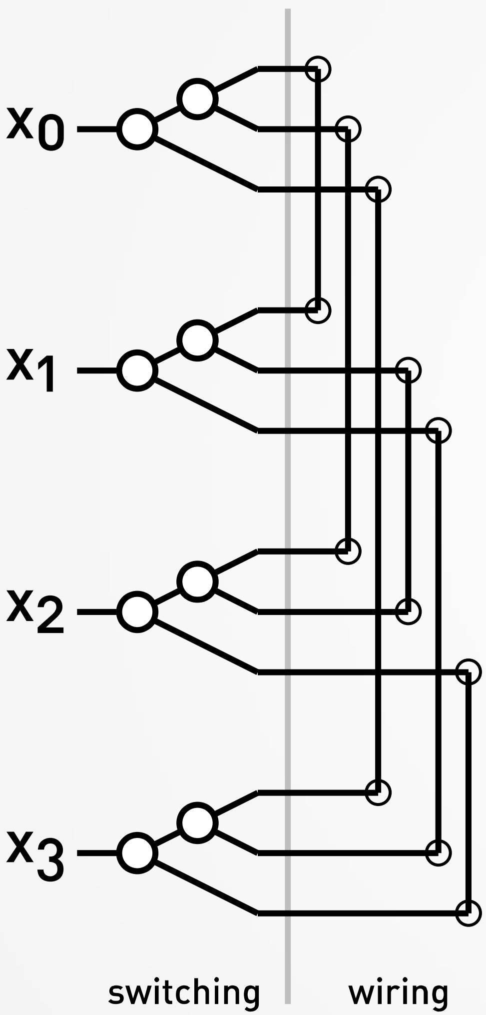
This is an implementation of a one-sided crossbar switch

Each x_i has a permanent connection to each x_j $(i \neq j)$

Each connection can be used by closing two elementary switches

Problem: Each terminal has a fanout of (n-1)

ONE-SIDED BINARY TREE – CROSSBAR SWITCH



Solution: Using binary trees

Each terminal effectively has a fanout of 2

Each binary tree will have a depth of $\lceil \log_2(n-1) \rceil$

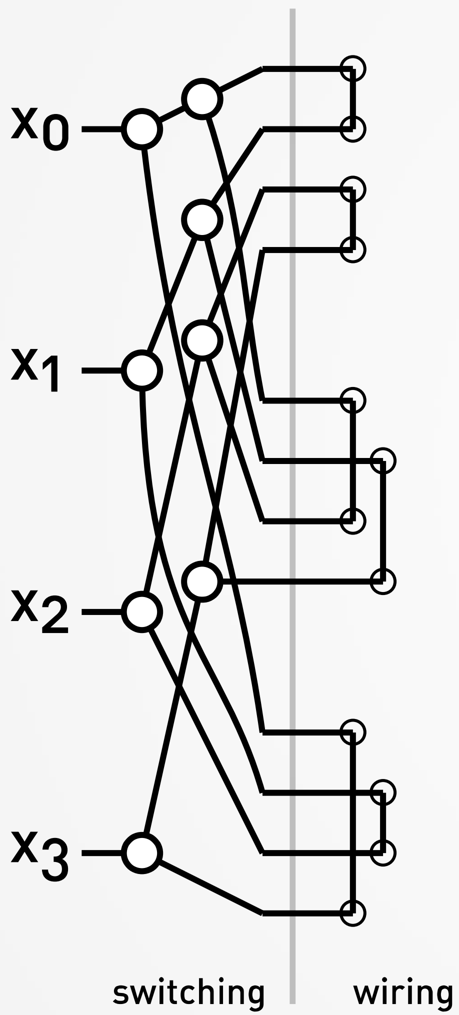
Problem: Wiring section has rather long connections

Solution: Reordering 'rows'

Rows (horizontal wires) are ordered as:

(x0,x0,x0), (x1,x1,x1), (x2,x2,x2), (x3,x3,x3)

ONE-SIDED BINARY TREE – CROSSBAR SWITCH



We can re-order rows to be:

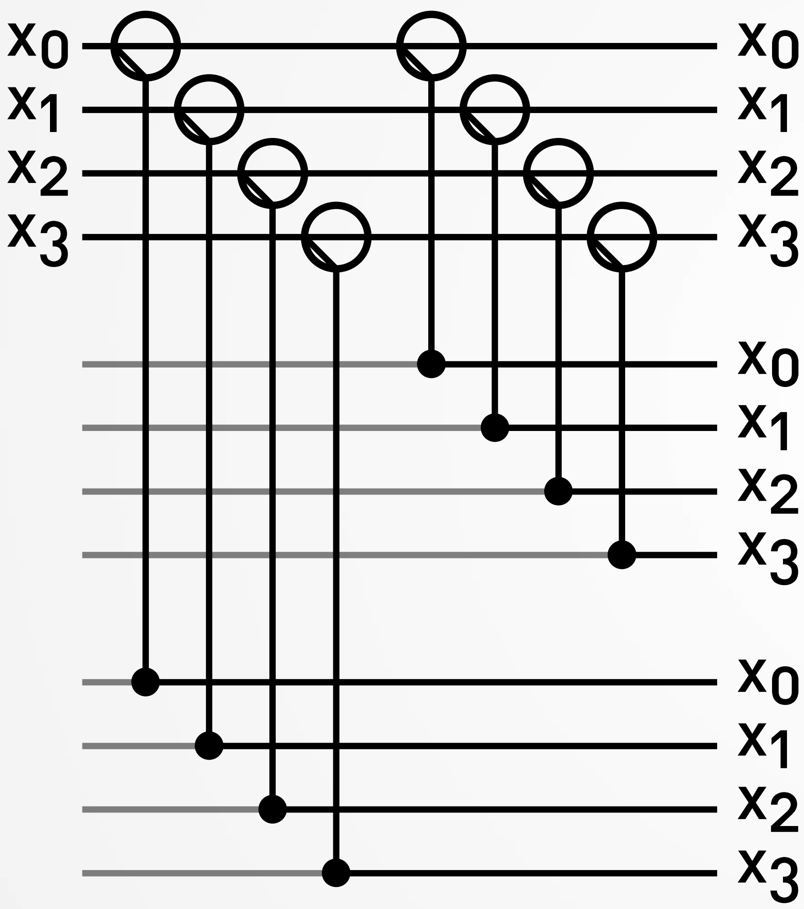
(x0,x1,x2,x3),(x0,x1,x2,x3),(x0,x1,x2,x3)

This simplifies the wiring section greatly

In fact, we know that when ordered such, there exists a wiring scheme where the total number of columns in the wiring section does not exceed $\lfloor n/2 \rfloor$ (Proof)

Problem: Switching side became complicated

IMPLEMENTING THE SWITCHING STEP ON A GRID



A simple implementation of the switching step (wiring part is hidden)

Previously, we claimed that there exists a wiring scheme with at most $\lfloor n/2 \rfloor$ columns. We propose a method to actually design that scheme.

DESIGNING A WIRING SCHEME

One method appeals to cyclic permutation groups

For our example, cyclic permutation groups are:

$$p = (0123), \quad p^2 = (02)(13), \quad p^3 = (0321)$$

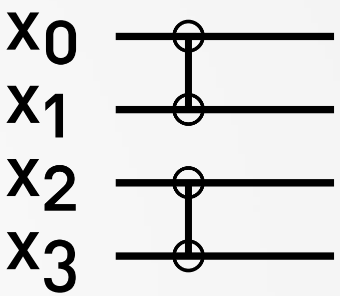
To obtain a wiring scheme, pair the numbers:

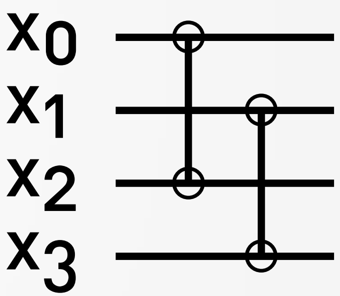
$$p = (0123), \quad p^2 = (02)(13), \quad p^3 = (0321)$$

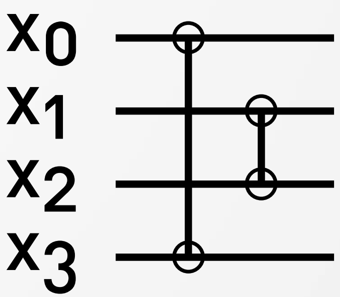
The connections to be made are:

$$(01),(23),(02),(13),(03),(21)$$

Wiring is shown on the right







DESIGNING A WIRING SCHEME

However this method might become tedious for odd n:

$$p\ \ = (01234) \rightarrow (01)(23)(4)$$

$$p^2 = (02413) \rightarrow (02)(41)(3)$$

$$p^3 = (03142) \rightarrow (03)(1)(42)$$

$$p^4 = (04321) \rightarrow (04)(3)(21)$$

Note that (43) and (13) were not generated.

We introduce another intuitive method appealing to adjacency matrices

ADJACENCY MATRICES

We represent each connection between terminals x_i and x_j with the pair **<i,j>**

Since we assume connections are two-way (undirected), **<i,j> = <j,i>**

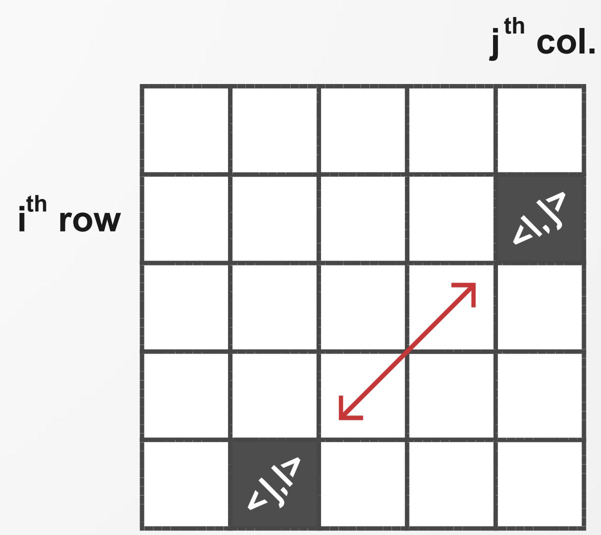
For n terminals, we can represent all the connections with an n x n adjacency matrix **A**

Here, $\mathbf{A_{i,j}}$ corresponds to the pair **<i,j>**

Then A is symmetric ($\mathbf{A_{i,j} = A_{j,i}}$)

For a complete K_n graph,

all $A_{i,j}$ must be 1 ($i \neq j$)



GROUPING PAIRS

To design a complete wiring scheme, we should cover all the upper triangle in the matrix

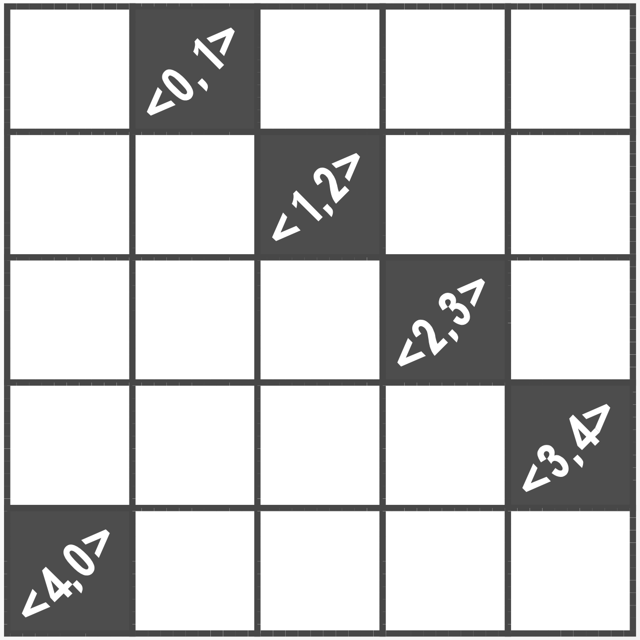
Method: Diagonally scan the matrix

Define pairs of groups as <x, (x+k) **mod** n>

Example: n=5, k=1, $0 \leq x < n$

Pairs are

<0,1>,<1,2>,<2,3>,<3,4>,<4,**0**>

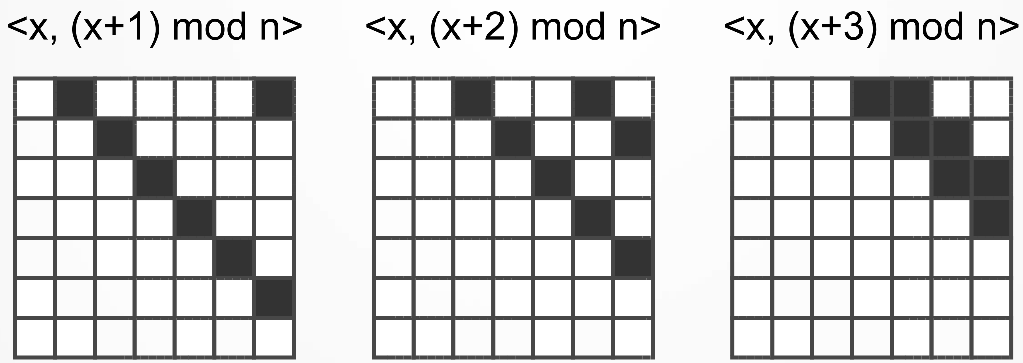


GROUPING PAIRS – ODD n CASE

We should diagonally cover the entire upper triangle by changing **k**

Example: n=7, k=1,2,3, 0 ≤ x < n

(All pairs are shown in the upper triangle)



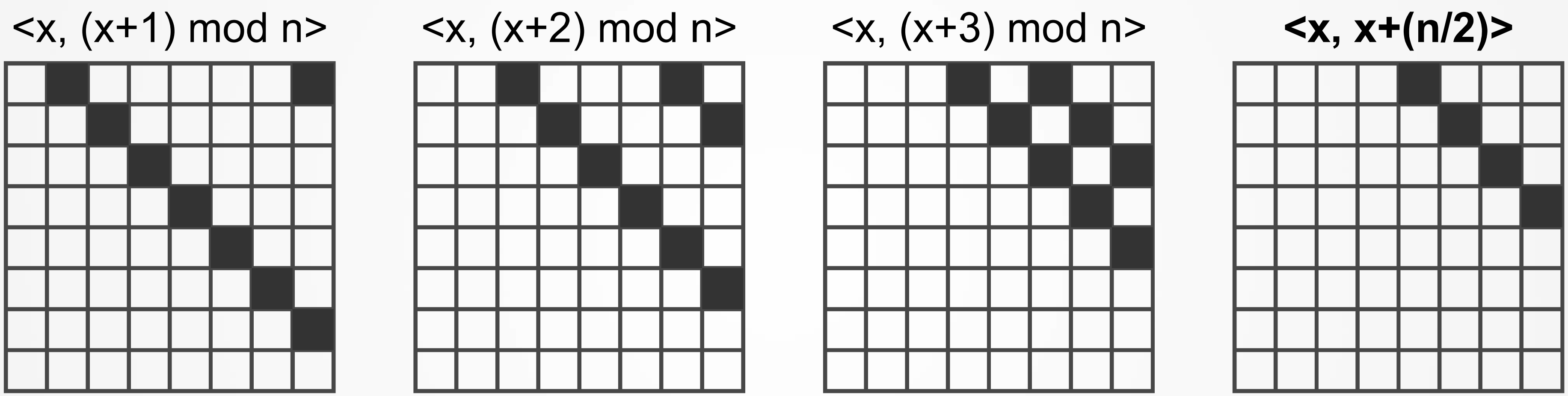
We repeat for all k in the range **[1, floor(n/2)]**

This works well **for odd n**

GROUPING PAIRS – EVEN n CASE

For even n, should consider the case k=n/2 separately

Example: n=8, k=1,2,3,**4**, $0 \le x < n$



If we used *<i, (i+k)* ***mod*** *n>* for k=n/2; we would cover one diagonal twice

We can solve this special case by changing the range of x for k=n/2:

<x,x+n/2>   0 ≤ x **< n/2** (instead of n)

GROUPING PAIRS – FORMAL DEFINITION

Below are the formal definitions of these two cases:

For odd n:

$\langle x, (x+k) \bmod n \rangle, \quad 0 \leq x < n, \quad 1 \leq k < n/2$ **(Eqn.1)**

For even n:

$\langle x, (x+k) \bmod n \rangle, \quad 0 \leq x < n, \quad 1 \leq k < n/2$ **(Eqn.1)**

and

$\langle x, x+n/2 \rangle, \quad 0 \leq x < n/2$ **(Eqn.2)**

GROUPING PAIRS – WHY?

What is the advantage of grouping pairs in this fashion?

For each k in Eqn. 1, we have n pairs

Each terminal x_i appears twice in these pairs *(Proof in the paper)*: <i, •> and <•, i>

To implement all pairs for a given k, we need exactly two copies of every x_i

There are no interconnections between different k

From the switching section, we can assign two blocks $(x_0,x_1,\ldots,x_{n-1})$ to each k and wire them independently

WIRING

Eqn.2 is simple to implement

<x,x+n/2>, $0 \le x < n/2$

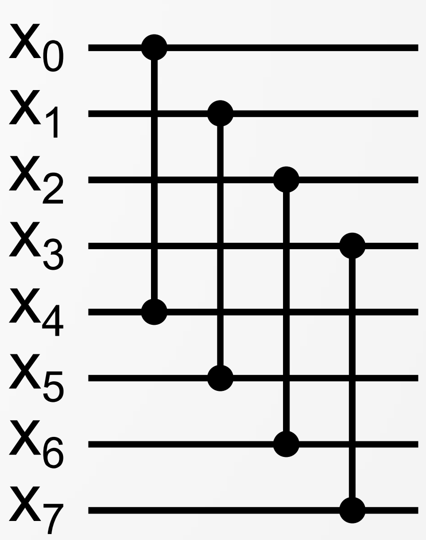
Each x_i appears **once**, therefore one block is sufficient

Example: n=8, k=4

Pairs: <0,4>,<1,5>,<2,6>,<3,7>

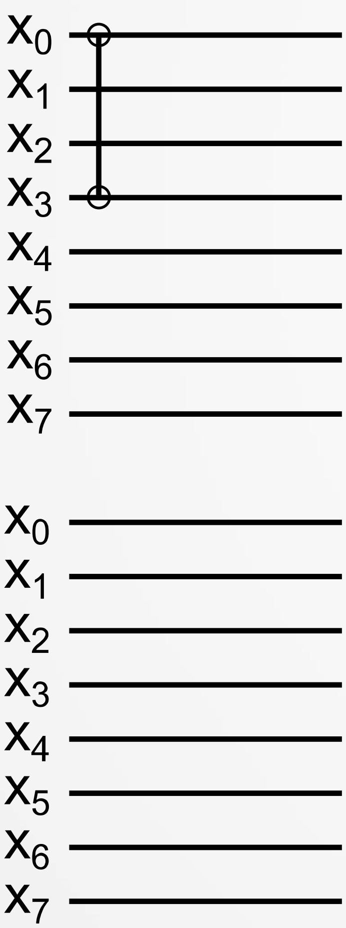
Connections are trivial

Number of columns: **n/2** ✓



WIRING

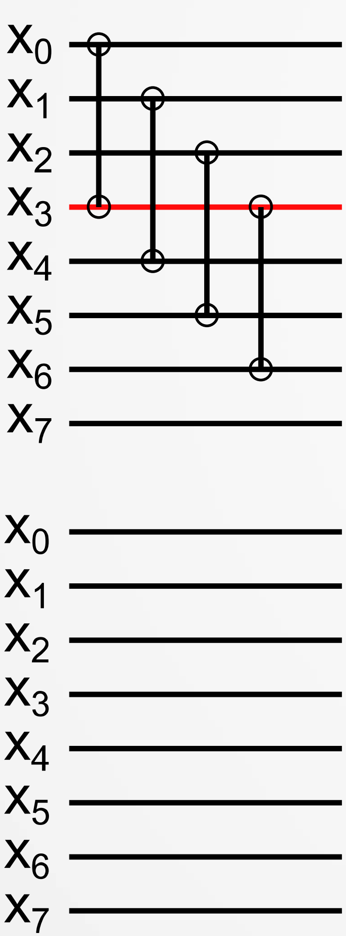
Implementation of Eqn.1 for a given k over an example: **n=8, k=3**



Start adding connections <x,x+k> to the first block

WIRING

Implementation of Eqn.1 for a given k over an example: **n=8, k=3**

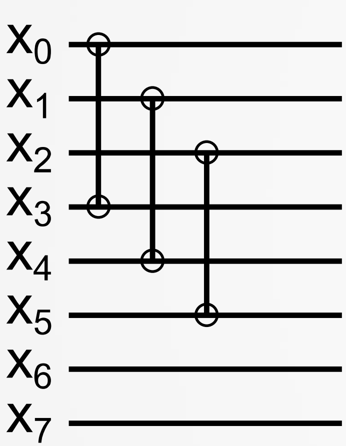


Start adding connections <x,x+k> to the first block

After adding k connections, <k,2k> will collide with <0,k>

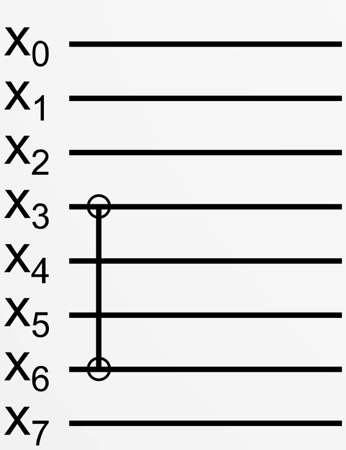
WIRING

Implementation of Eqn.1 for a given k over an example: **n=8, k=3**



Start adding connections <x,x+k> to the first block

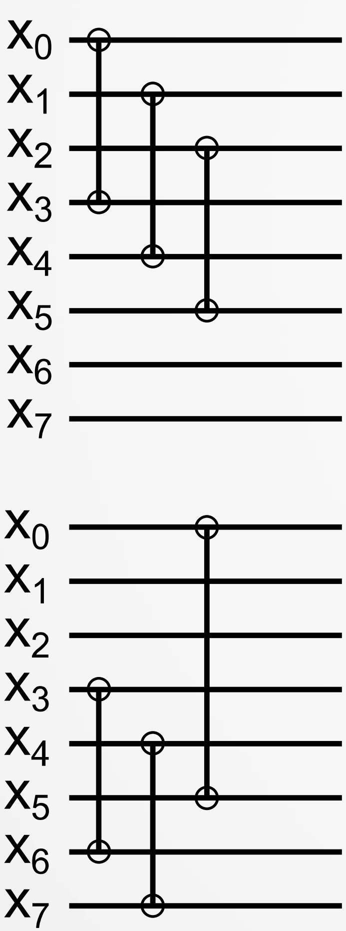
After adding k connections, <k,2k> will collide with <0,k>



Alternate to the second block, repeat this jump between blocks at every k wires

WIRING

Implementation of Eqn.1 for a given k over an example: **n=8, k=3**

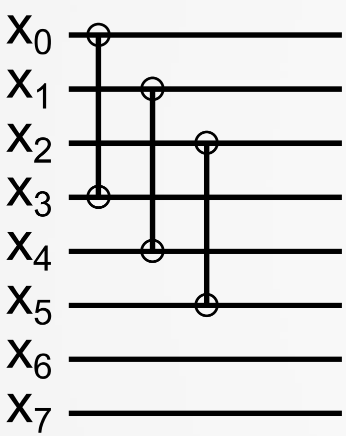


(cont'd)

Pair <5,0> is a 'reverse pair', that is, 5>0 unlike other pairs due to the modulo operator. For a reverse pair <i,j>, always pick j from the second block

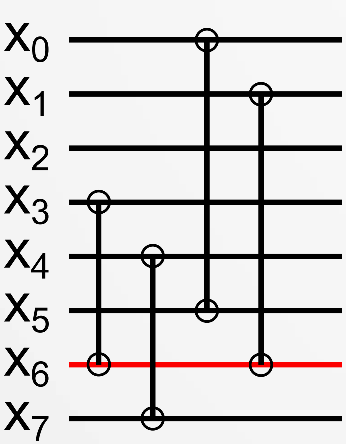
WIRING

Implementation of Eqn.1 for a given k over an example: **n=8, k=3**



(cont'd)

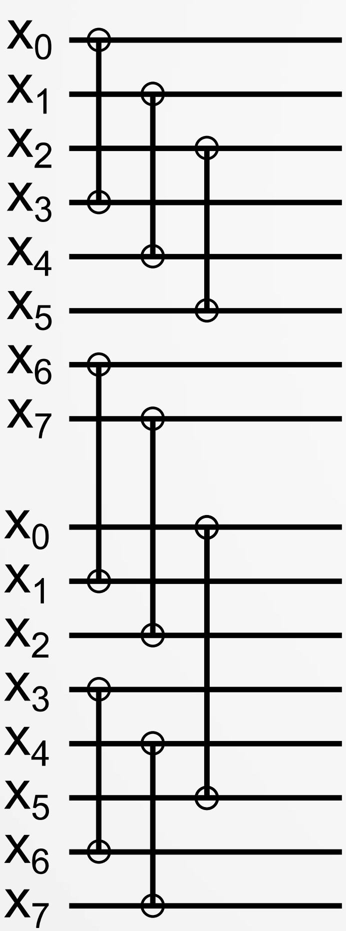
Pair <5,0> is a 'reverse pair', that is, 5>0 unlike other pairs due to the modulo operator. For a reverse pair <i,j>, always pick j from the second block



Again, we have added k connections. Collision happens, alternate to block 1

WIRING

Implementation of Eqn.1 for a given k over an example: **n=8, k=3**



(cont'd)

Pair <5,0> is a 'reverse pair', that is, 5>0 unlike other pairs due to the modulo operator. For a reverse pair <i,j>, always pick j from the second block

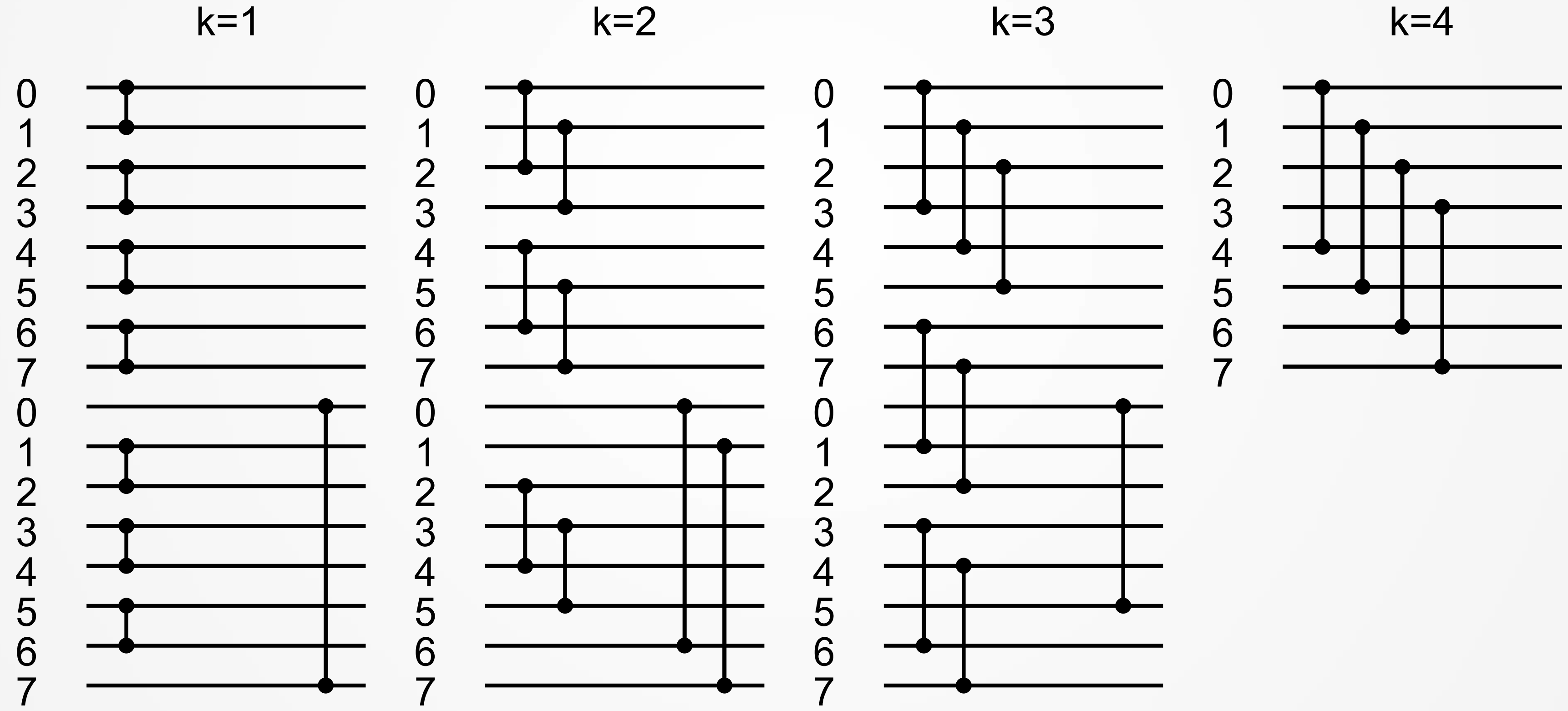
Again, we have added k connections. Collision happens, alternate to block 1

Add last wires and finalize.

WIRING

To obtain complete wiring, repeat for every k:

(Reverse pairs are shown as aligned right)



Number of columns is bounded by floor(n/2)

CONCLUSION

We have introduced a method that constructs a one-sided binary tree – crossbar switch that:

- has at most floor(n/2) columns,

- has no connections between pairs of blocks, thus preserving locality,

- is one-pass in nature,

- algorithmically simple.

An implementation can be found at

https://github.com/kubuzetto/crossbarWiring/

THANK YOU FOR YOUR PATIENCE!

Questions?