

Bitonic Counting Networks

Martin Rohde

VP Concurrency Programming

This seminar paper uses and adapts
Maurice Herlihy's slides for
"The Art of Multiprocessor Programming",
M.Herlihy & N.Shavit, 2008,

available at:

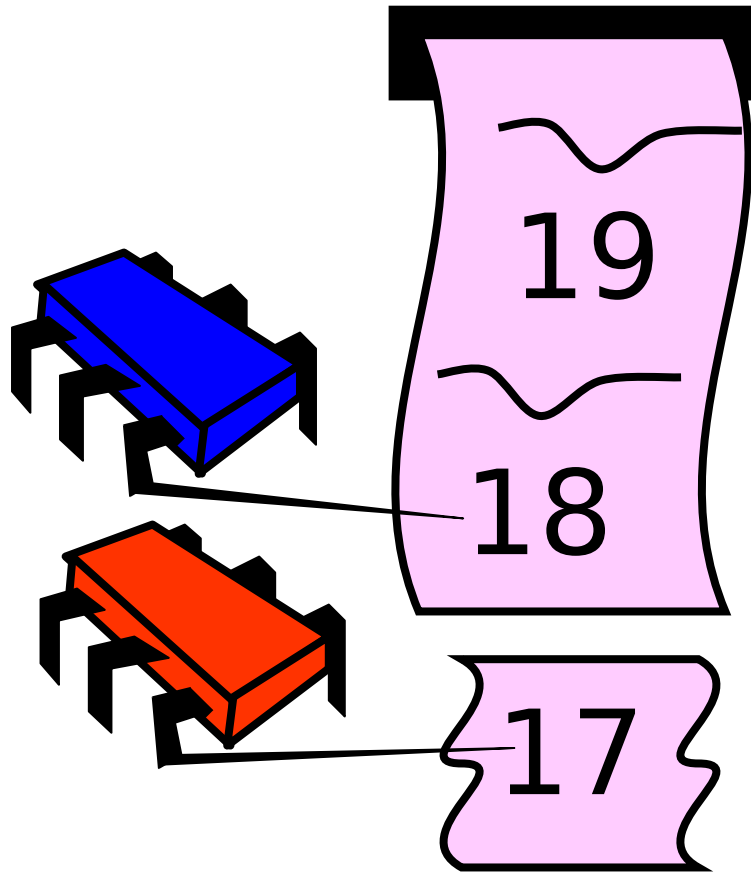
[http://www.elsevierdirect.com/companion.jsp?
ISBN=9780123705914](http://www.elsevierdirect.com/companion.jsp?ISBN=9780123705914)

licensed under:

Creative Commons Attribution-ShareAlike 2.5 License.

This license shall apply to my adaption also.

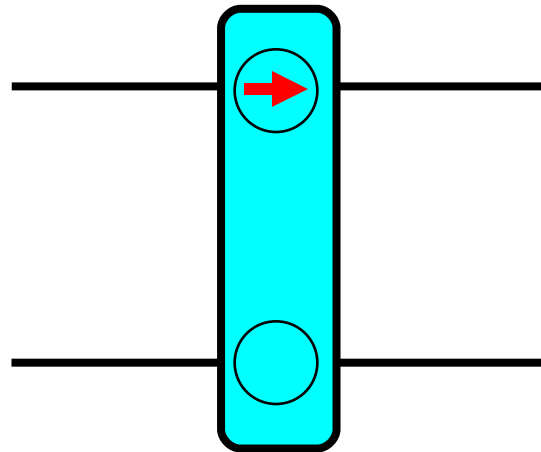
Shared Counter



each thread
takes a
number

A Balancer

Input
wires

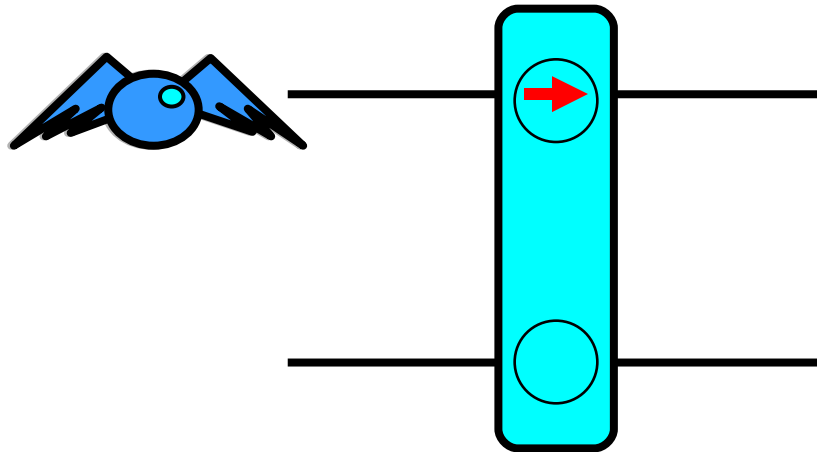


Output
wires

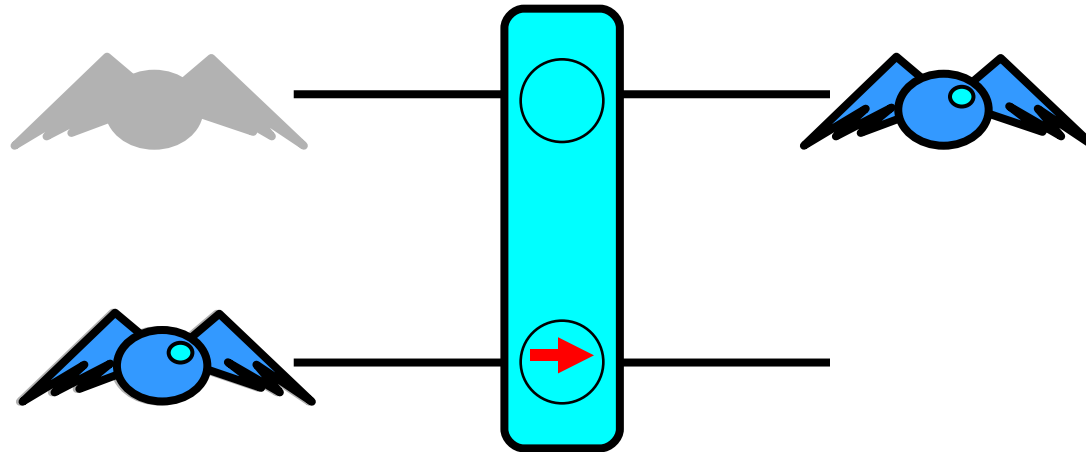
$$x_0 + x_1 = y_0 + y_1$$

x_i = Anzahl der auf i Eingehenden Tokens

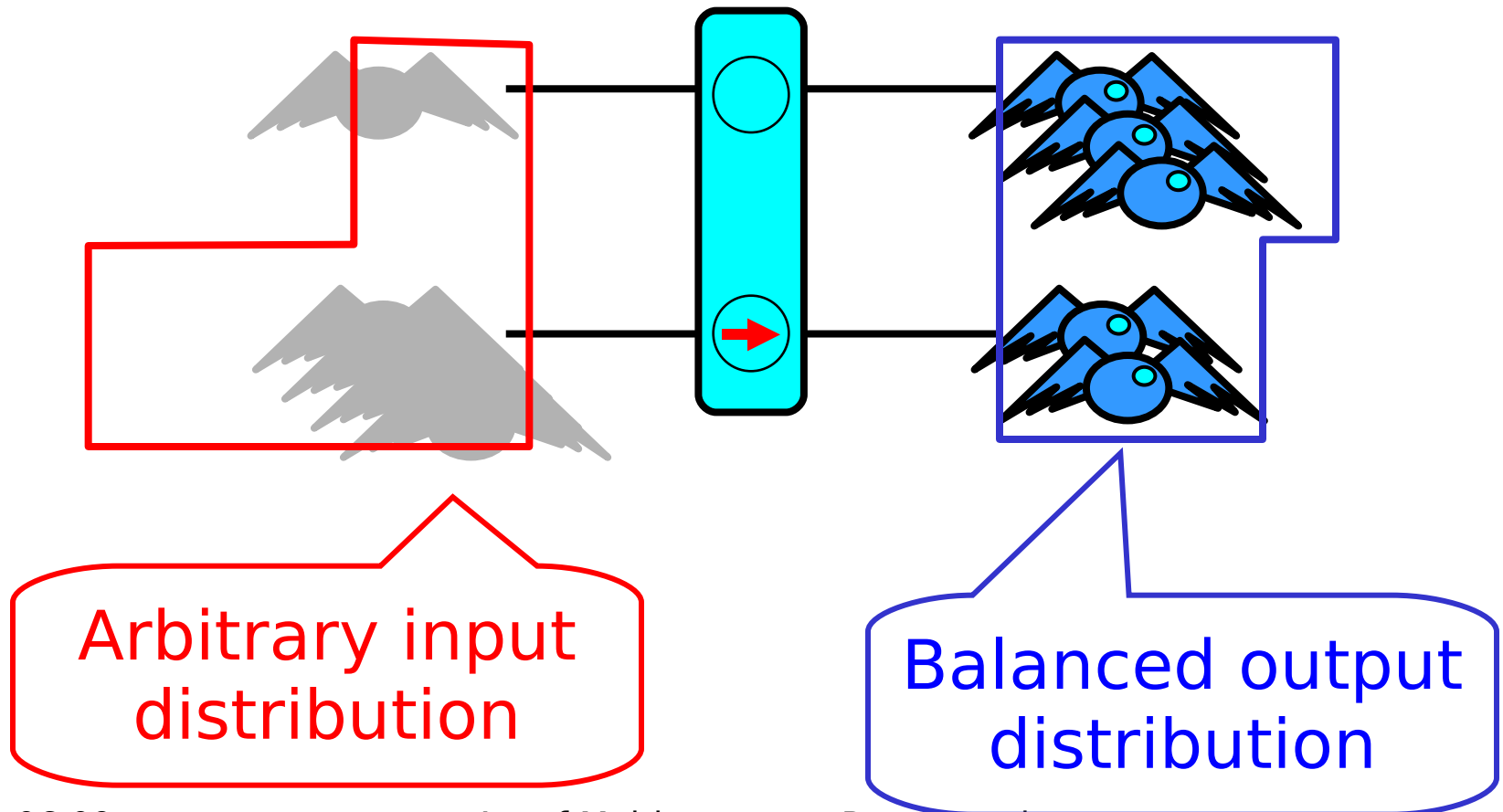
Tokens Traverse Balancers



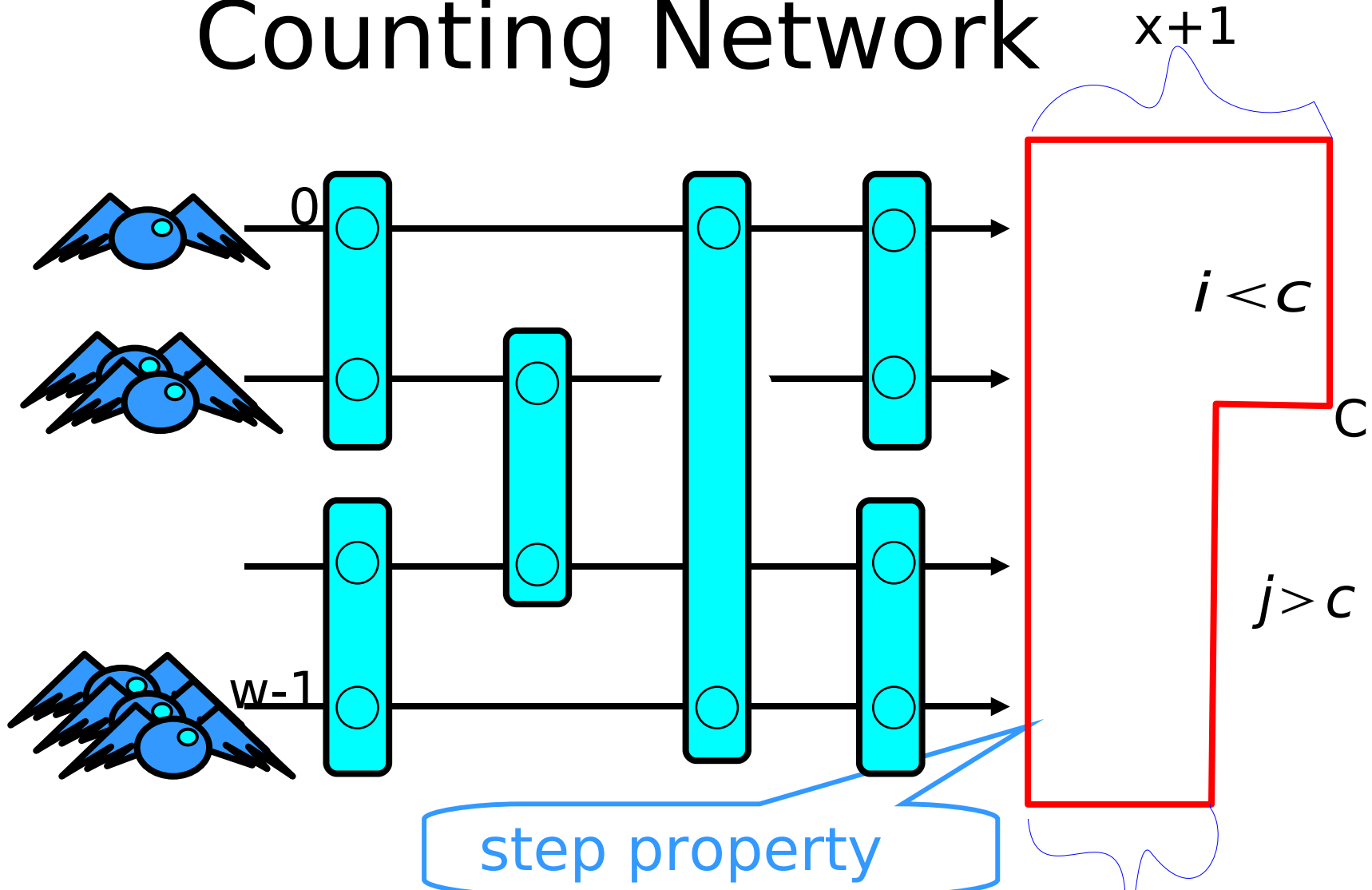
Tokens Traverse Balancers



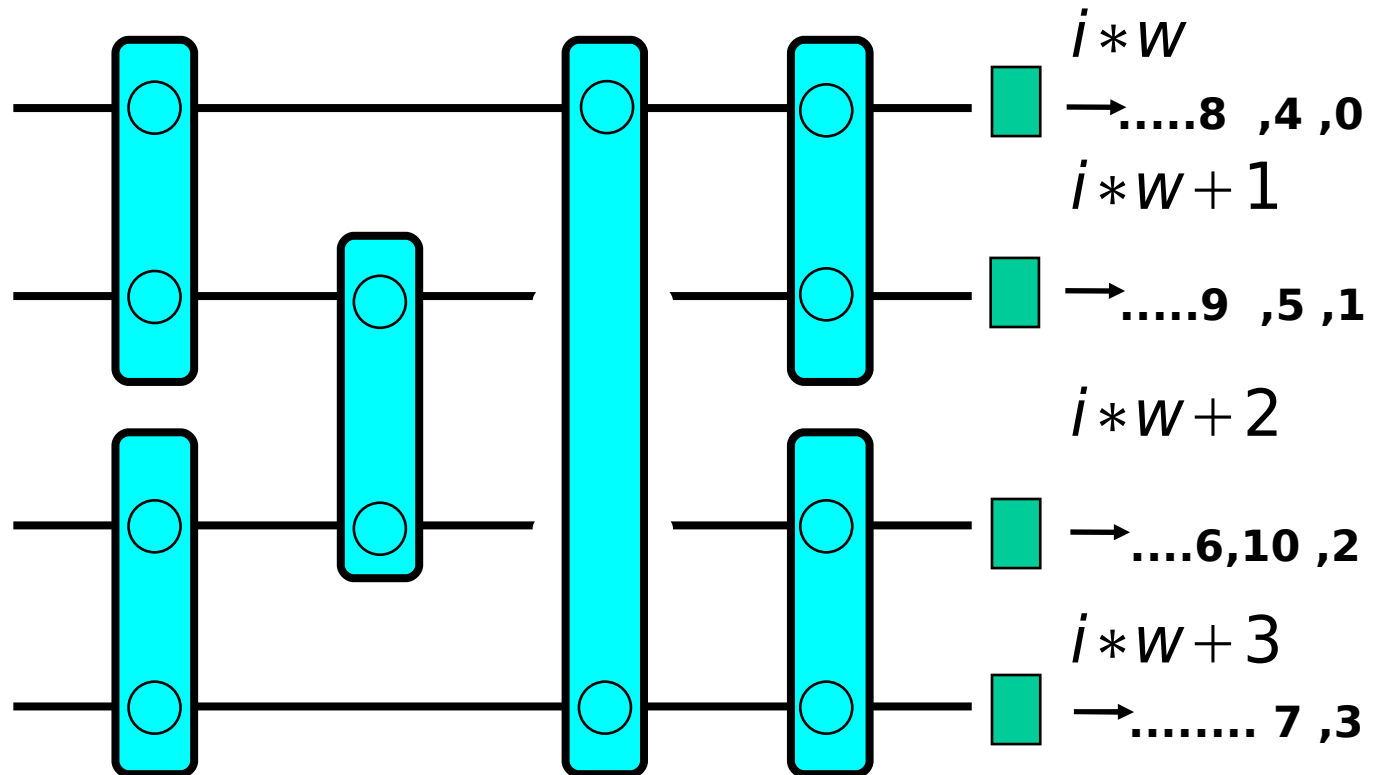
Tokens Traverse Balancers



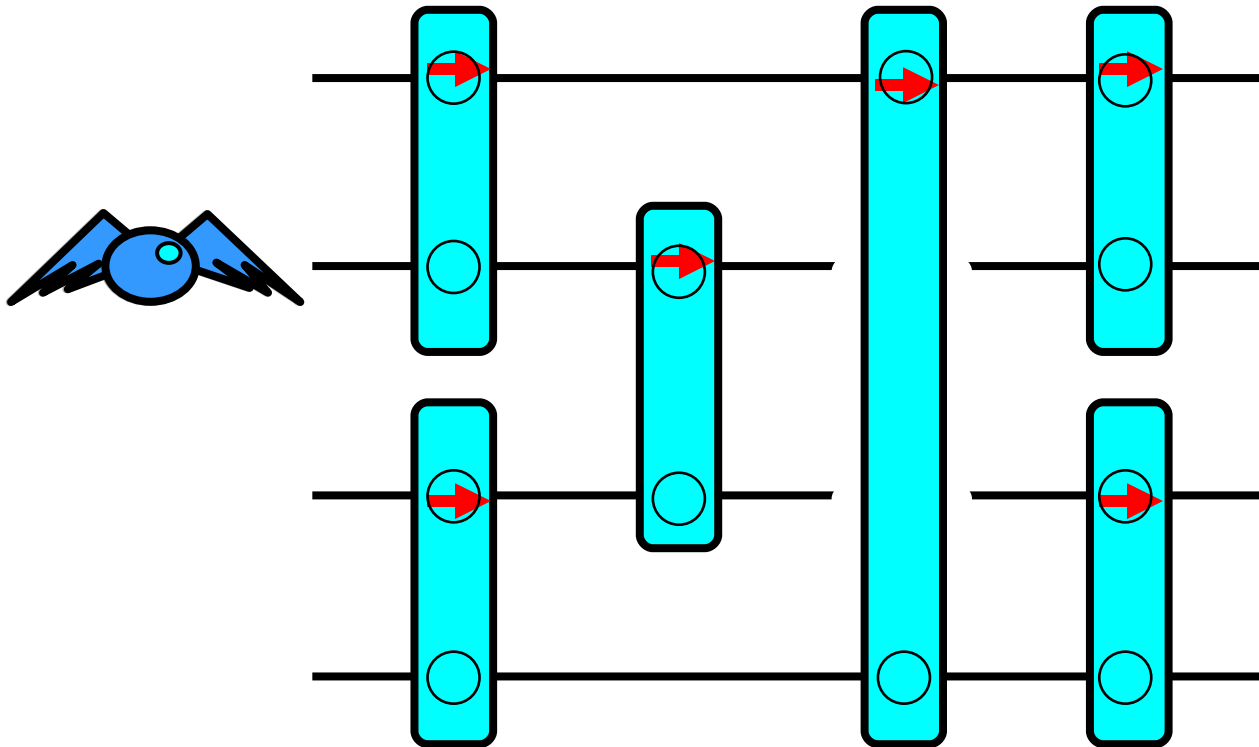
Counting Network



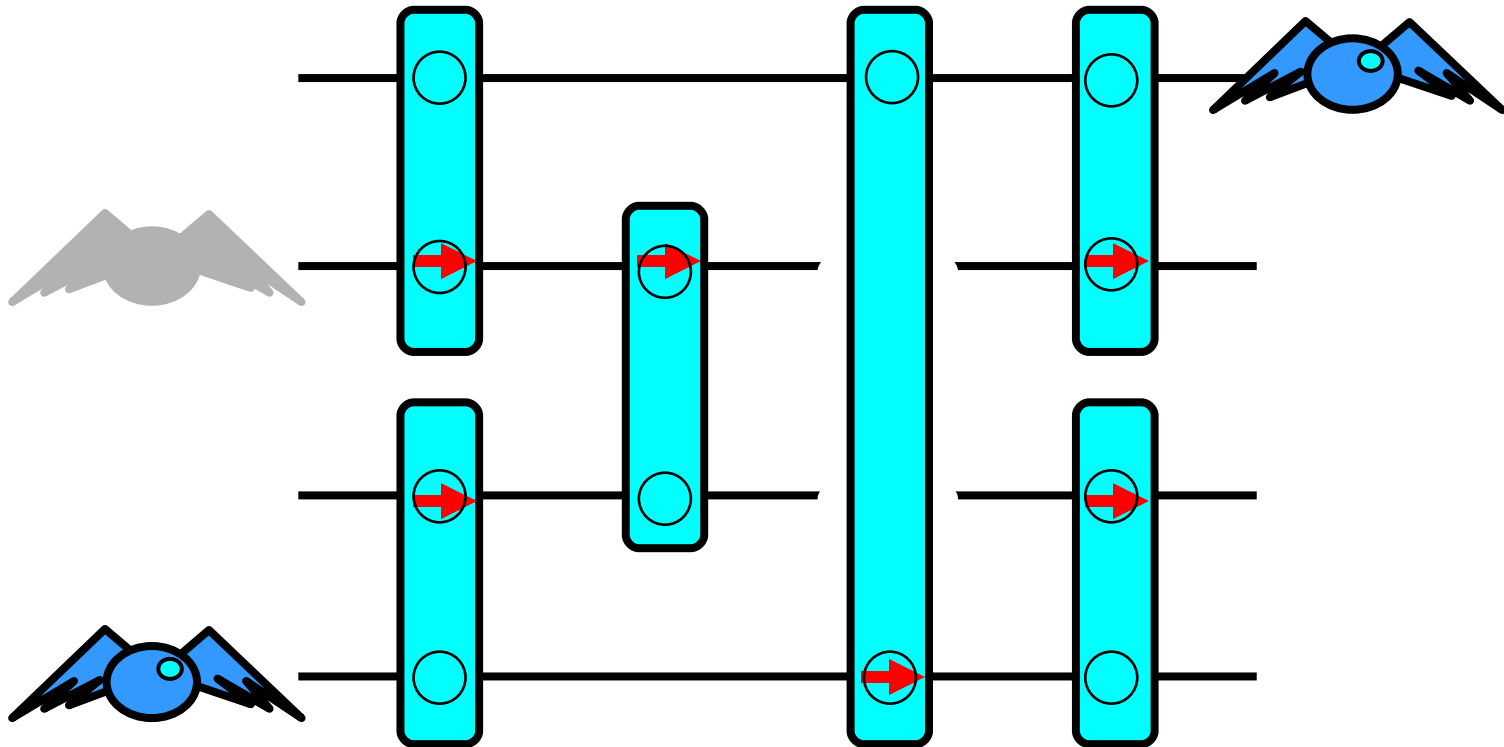
Counting Networks Count!



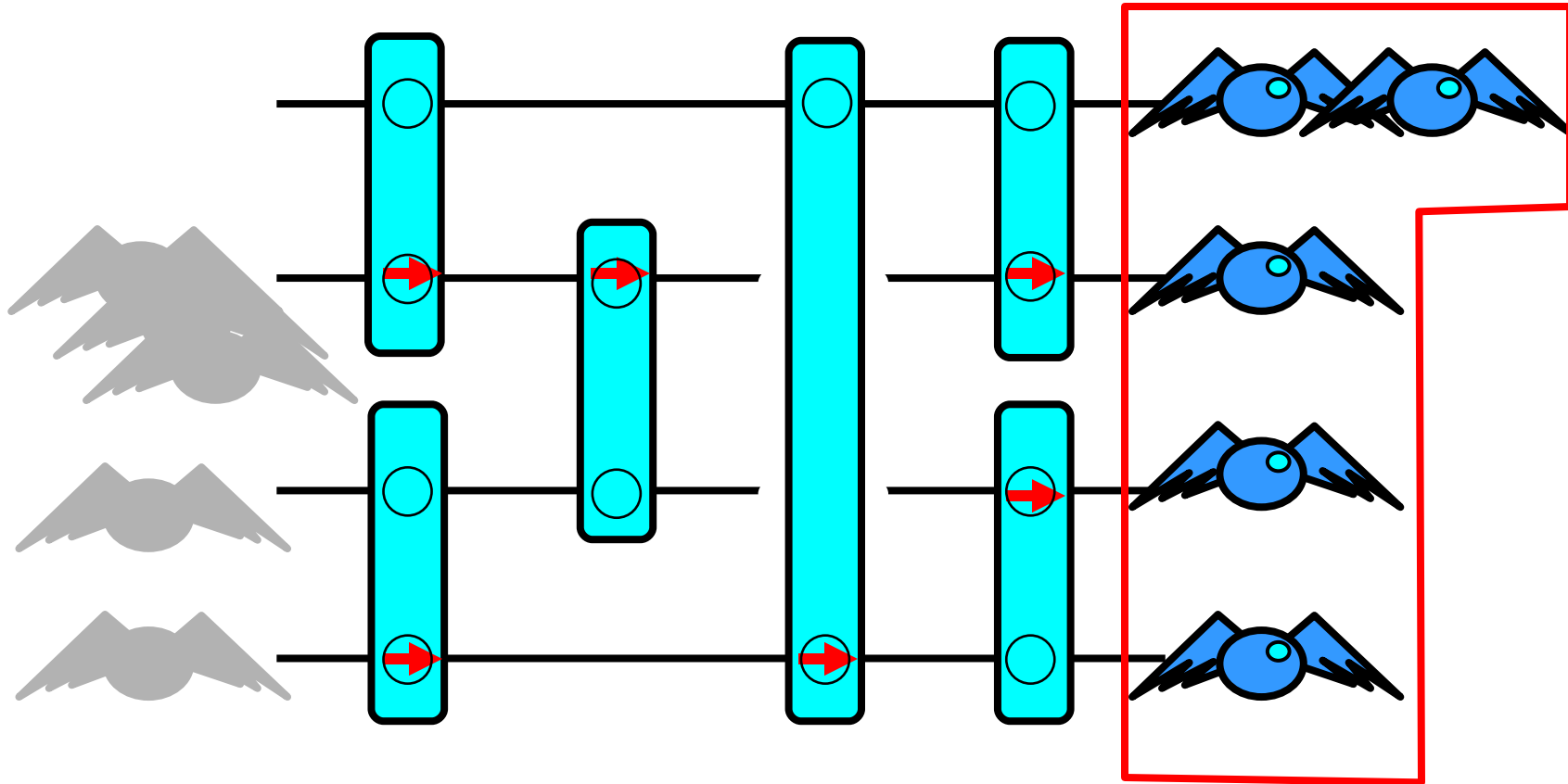
Bitonic[4]



Bitonic[4]



Bitonic[4]

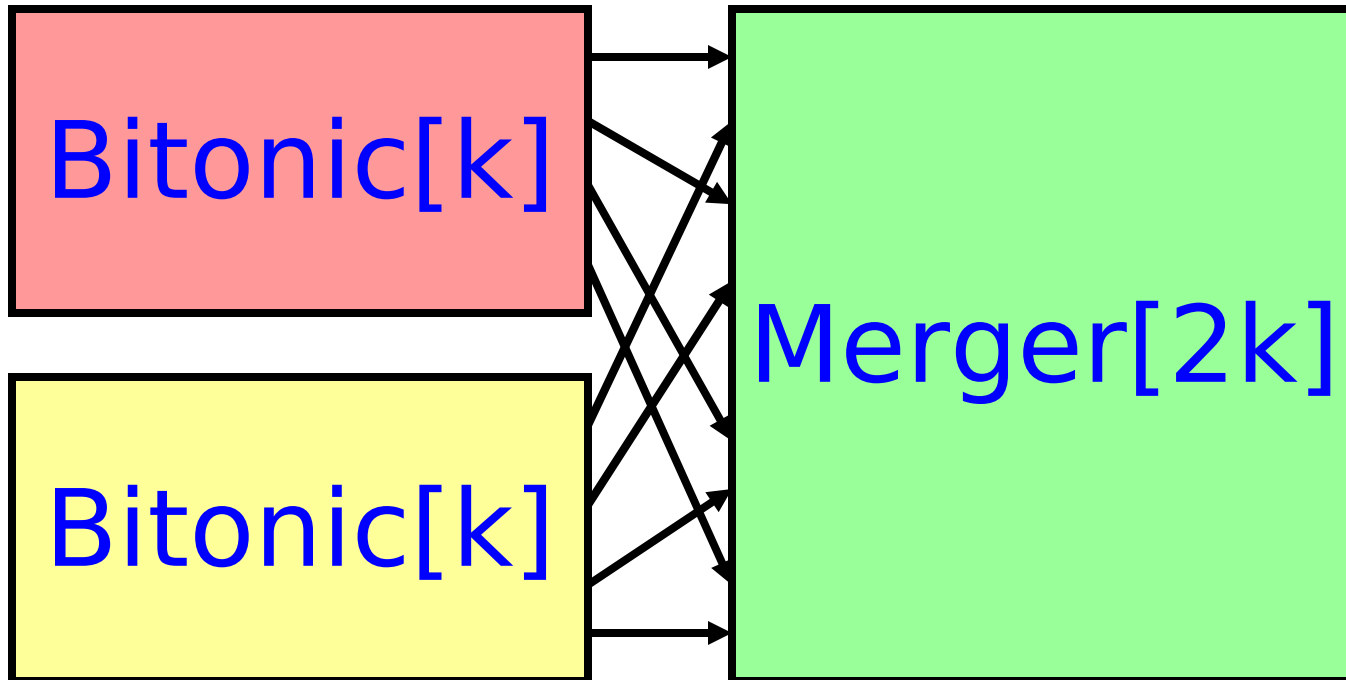


$$n = \sum x_i = \sum y_i = m$$

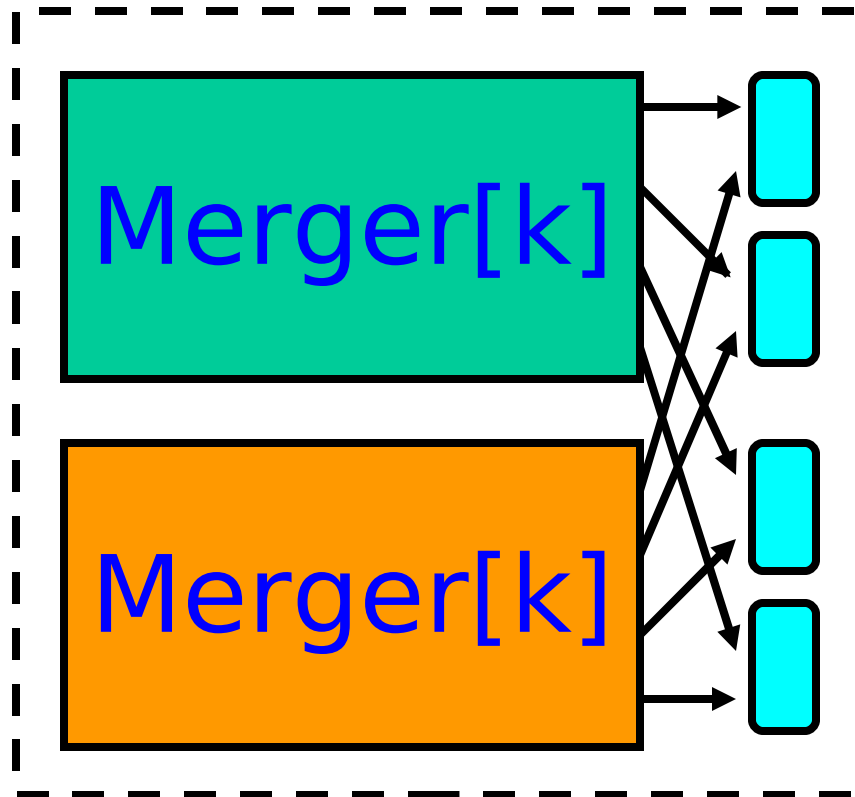
Shared Memory Implementation

```
class Balancer {  
    boolean toggle;  
    ...  
    public synchronized int traverse() {  
        try{  
            if(toggle)  
                return 0;  
            else  
                return 1;  
        }finally{ toggle =! toggle}  
    }  
}
```

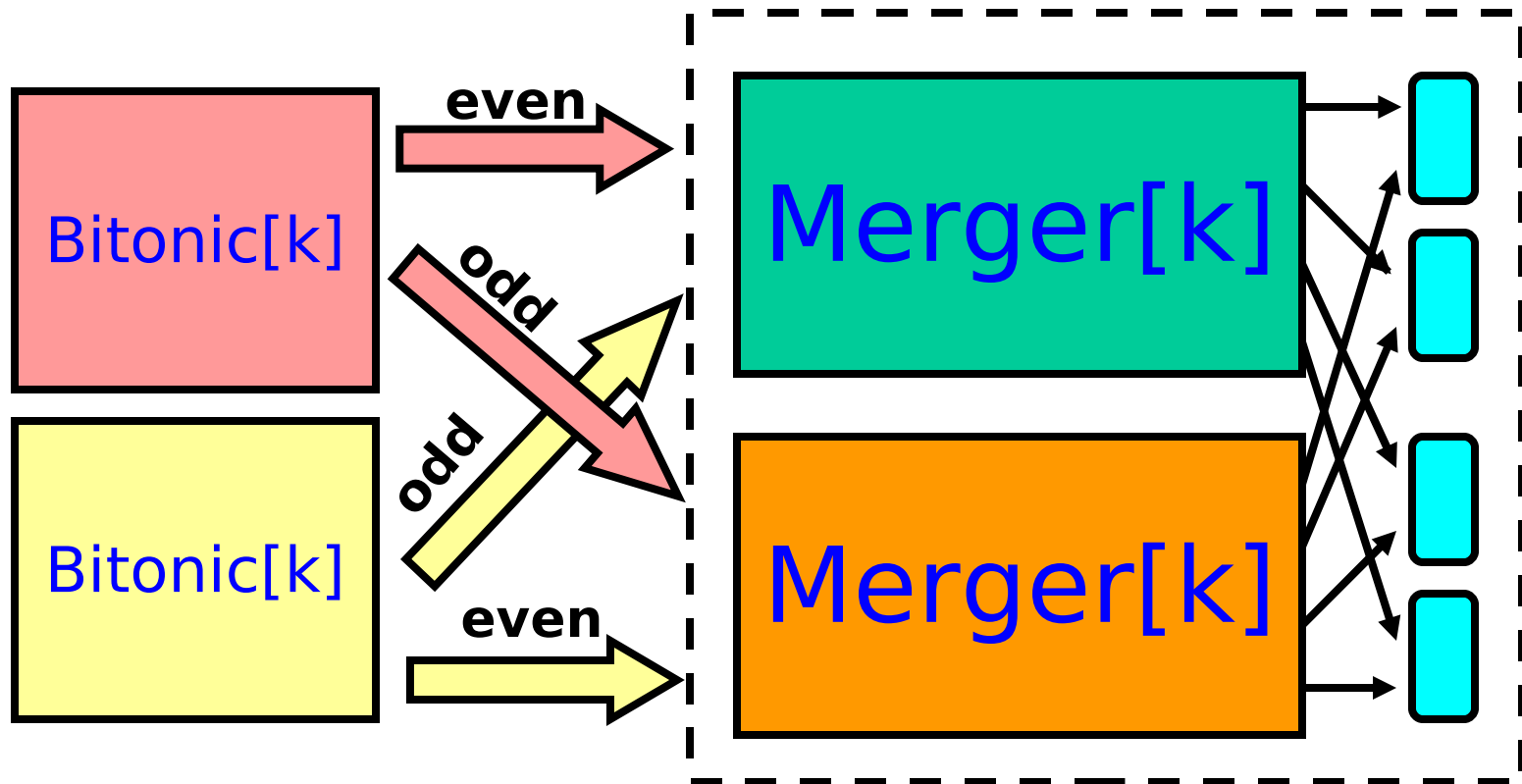
Bitonic[2k] Schematic



Merger[2k] Schematic



Merger[2k] Schematic



Shared Memory Implementation

```
class Merger {
    Merger[] half;
    Balancer[] layer;
    ...
    public Merger(int size) {\\merger of size n
        layer = new Balancer[size / 2];\\ n/2 Balancers
        for (int i = 0; i < size / 2; i++)
            layer[i] = new Balancer();
        if (size > 2) \\ recursive definition of merger
        half=new Merger[]{new Merger(size/2),newMerger(size/2)};
    }
}
```

Shared Memory Implementation

```
class Merger {
    Merger[] half;
    Balancer[] layer;
    ...
    public int traverse(int input) {
        int output = 0;
        if (size > 2) \\if size > 2 visit sub-mergers
            output = half[input % 2].traverse(input / 2);
        \\traverse throu the final layer of the merger
        return output + layer[output].traverse();
    }
}
```

Shared Memory Implementation

```
public class Bitonic{
    Bitonic[]    half;
    Merger layer;

    public Bitonic(int size) {
        layer = new Merger(size); \\the last merger look 28
        if (size > 2) \\ if size >2 two sub_mergers
            half = new Bitonic[]{new Bitonic(size/2), new
                Bitonic(size/2)};
        }
    }
}
```

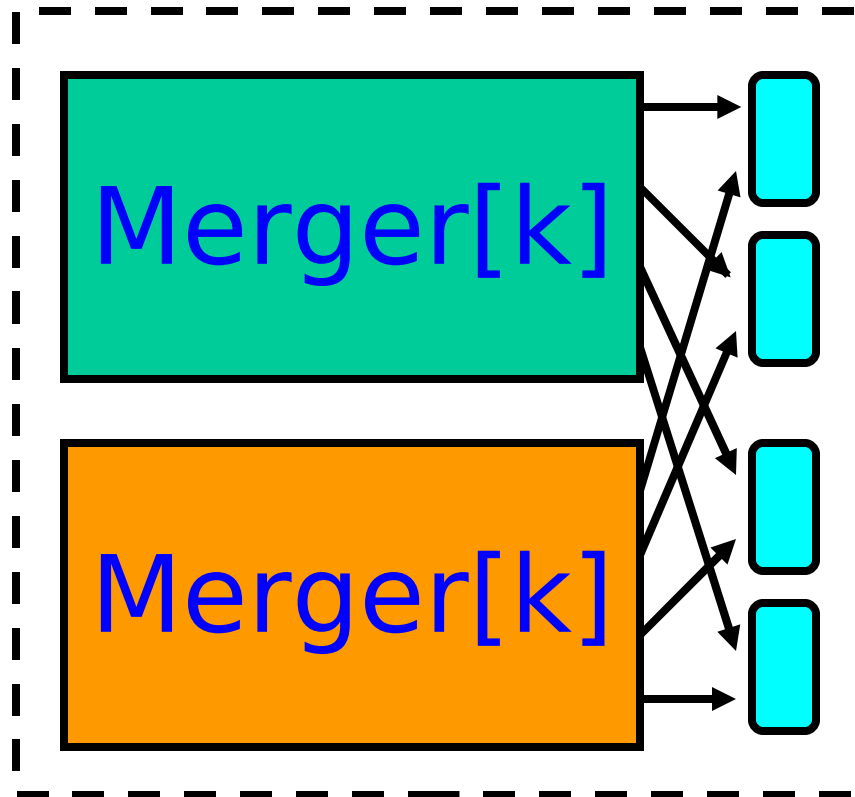
Shared Memory Implementation

```
public class Bitonic{
    Bitonic[] half;
    Merger layer;
    ...
    public int traverse(int input) {
        int output = 0;
        if (size > 2) \\the sub-merger
            output = half[input % 2].traverse(input / 2);
        \\ the last layer
        return output + layer.traverse(output);
    }
}
```

Bitonic[k] Depth

- Width $n = 2^k$
- Depth is $(\log_2 n)(\log_2 n + 1)/2$

Merger[2k] Schematic



Bitonic[k] Depth

Proof:

Depth of the Mergers:

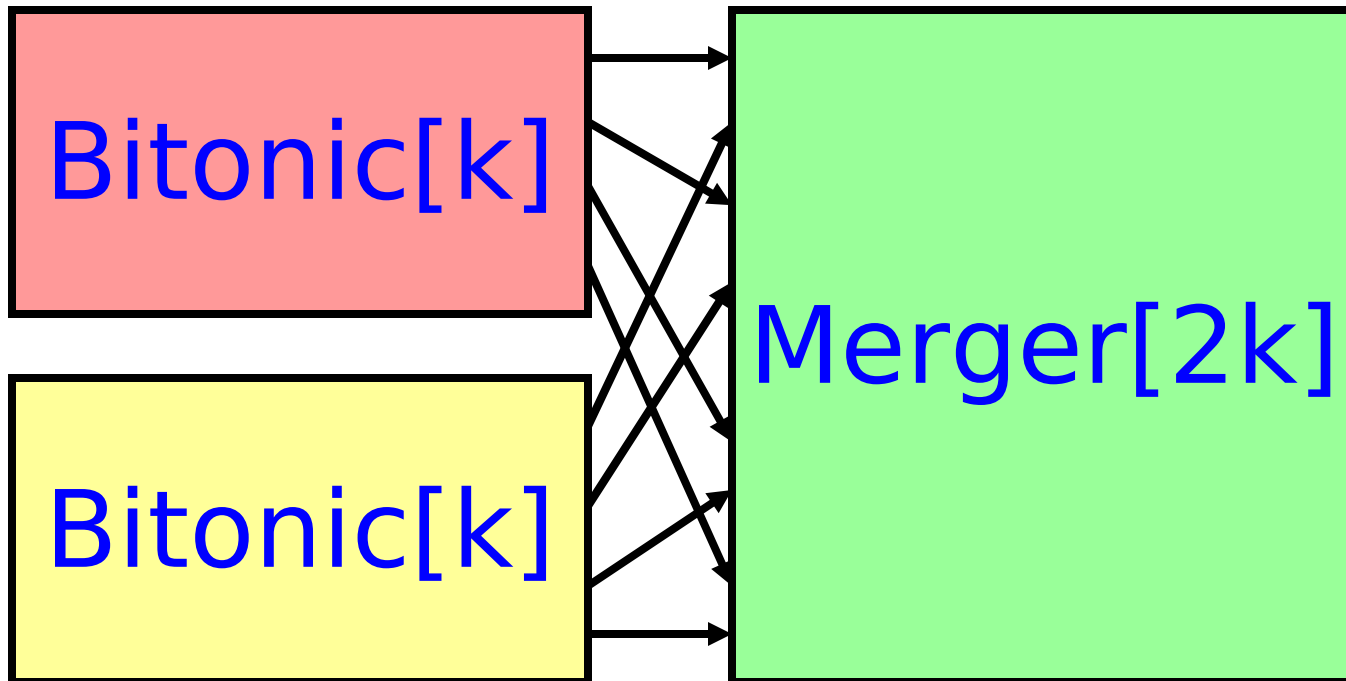
$$T(n) = T(n/2) + 1$$

$$T(2) = 1$$

$$\rightarrow O(\log(n))$$

Proof: complete induction

Bitonic[2k] Schematic



Bitonic[k] Depth

- Depth of the Bitonic Network

$$T(n) = T(n/2) + \log(n) \rightarrow O(\log^2(n))$$

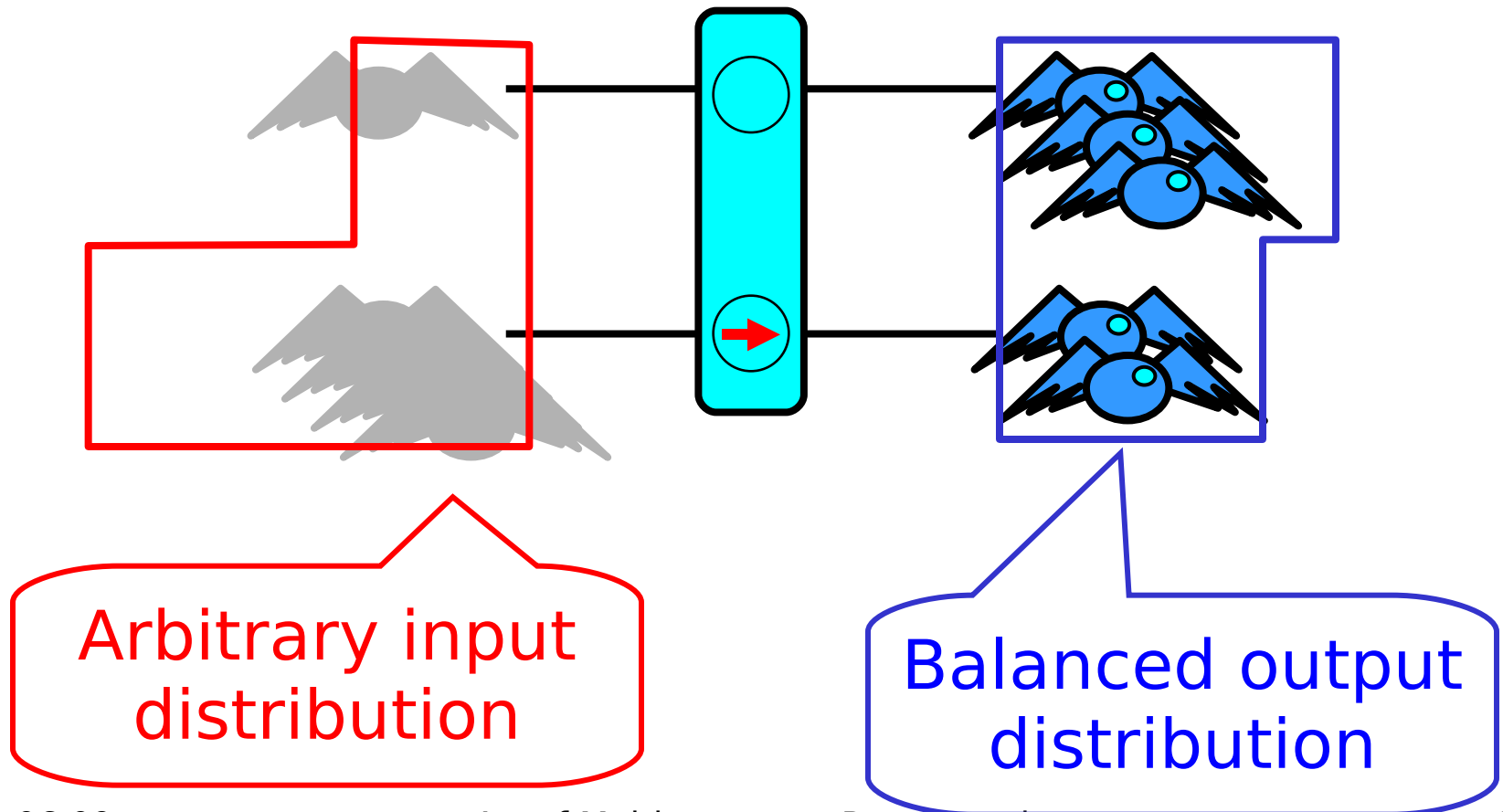
Correctness of the Method

Proof: complete induction:

I.H.: if two sequences of size $n/2$ have the step property, the final sequence (after merging) of length n will have the step property

I.B.: $k=1 \implies n=2^1$

Tokens Traverse Balancers

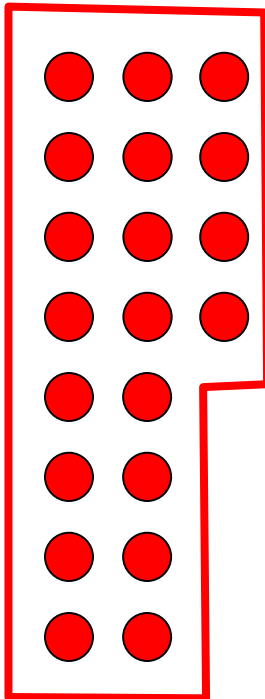


Correctness of the Method

Proof: complete induction:

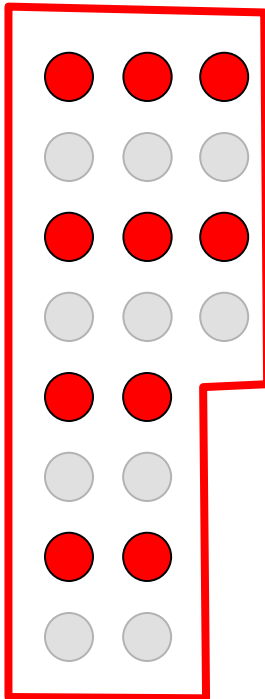
$$\text{I.B.: } k \rightarrow k+1 \iff 2^k \rightarrow 2^{k+1}$$

Lemma 1



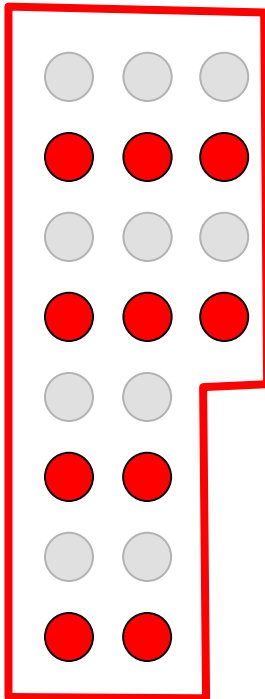
If a sequence has the
step property ...

Lemma 1



So does its even
subsequence

Lemma 1



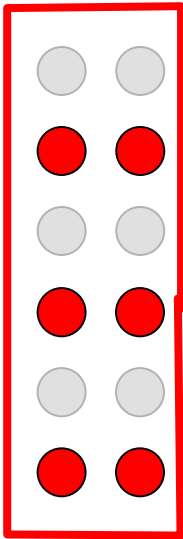
And its odd
subsequence

Lemma 2

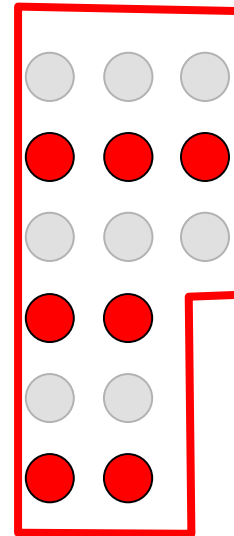
if a sequence has a step property, then its even and odd sequences satisfy:

$$\left(\sum_{i=0}^{k/2-1} x_{2i}\right) = \left\lfloor \sum_{i=0}^{k-1} x_i / 2 \right\rfloor \wedge \left(\sum_{i=0}^{k/2-1} x_{2i+1}\right) = \left\lceil \sum_{i=0}^{k-1} x_i / 2 \right\rceil$$

Lemma 2



case a



case b

$$x_{2*i} = 3, i = 1$$

$$x_{2*i+1} = 2, i = 1$$

Lemma 2

Proof:

$$\forall i, j: 0 \leq i < j \leq k-1: 0 \leq x_i - x_j \leq 1 \text{ (step property)}$$

$$\rightarrow \dots \rightarrow \forall i: 0 \leq i < k/2: x_{2i} = x_{2i+1} \vee \exists! j: x_{2j} = x_{2j+1} + 1$$

$$\text{the first case: } \rightarrow \sum_{i=0}^{k/2-1} x_{2i} = \sum_{i=0}^{k/2-1} x_{2i+1} = \sum_{i=0}^{k-1} x_i / 2$$

$$\text{the second case: } \rightarrow \sum_{i=0}^{k/2-1} x_{2i} = \left\lfloor \sum_{i=0}^{k-1} x_i / 2 \right\rfloor \wedge \sum_{i=0}^{k/2-1} x_{2i+1} = \left\lceil \sum_{i=0}^{k-1} x_i / 2 \right\rceil$$

Lemma 2

sei $n = \sum_{i=0}^{k/2-1} x_{2i+1} \rightarrow \sum_{i=0}^{k-1} x_i = 2n + 1$

dann $(2n+1)/2 = n, \dots$

$$\rightarrow \lfloor n, \dots \rfloor = n = \sum_{i=0}^{k/2-1} x_{2i+1}$$

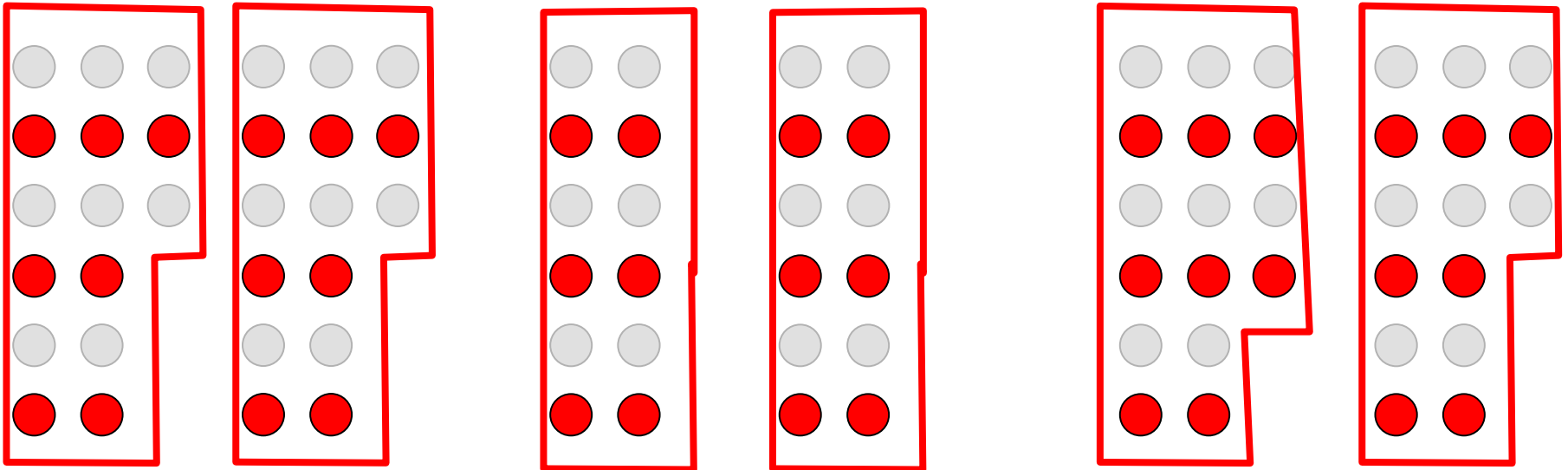
$$\rightarrow \lfloor n, \dots \rfloor = n + 1 = \sum_{i=0}^{k/2-1} x_{2i}$$

Lemma 3

let x and y be arbitrary two sequences having the step property:

If $\sum x_i = \sum y_i$ then $x_i = y_i \forall 0 \leq i < k$

Lemma 3



Lemma 3

Proof:

$$m = \sum_{i=0}^{k-1} x_i \rightarrow x_i = \lfloor (m-i)/k \rfloor \text{ (step property)}$$

$$m = \sum x_i = \sum y_i \text{ According to the step property:}$$

$$x_i = y_i = \lfloor (m-i)/k \rfloor$$

Lemma 4

let x and y be **arbitrary** two sequences having the step property:

$$\text{If } \sum x_i = \sum y_i + 1 \text{ then } \exists! j: 0 \leq j < k: x_j = y_j + 1 \\ \forall i \neq j, 0 \leq i < k: x_i = y_i$$

Lemma 4

Proof:

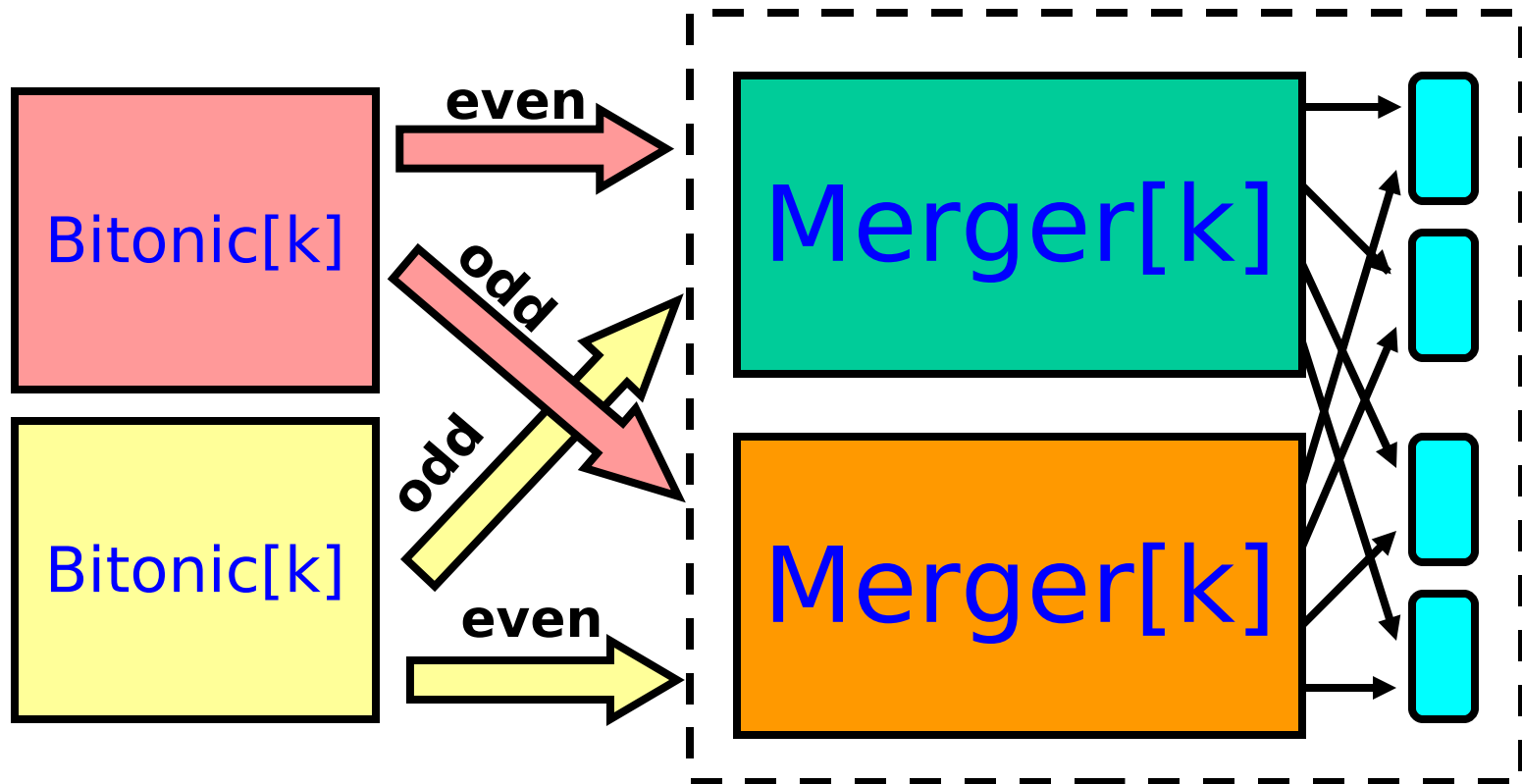
$$\text{Let } m = \sum x_i = \sum y_i + 1$$

according to the step property

$$x_i = \lfloor (m-i)/k \rfloor \wedge y_i = \lfloor (m-1-i)/k \rfloor$$

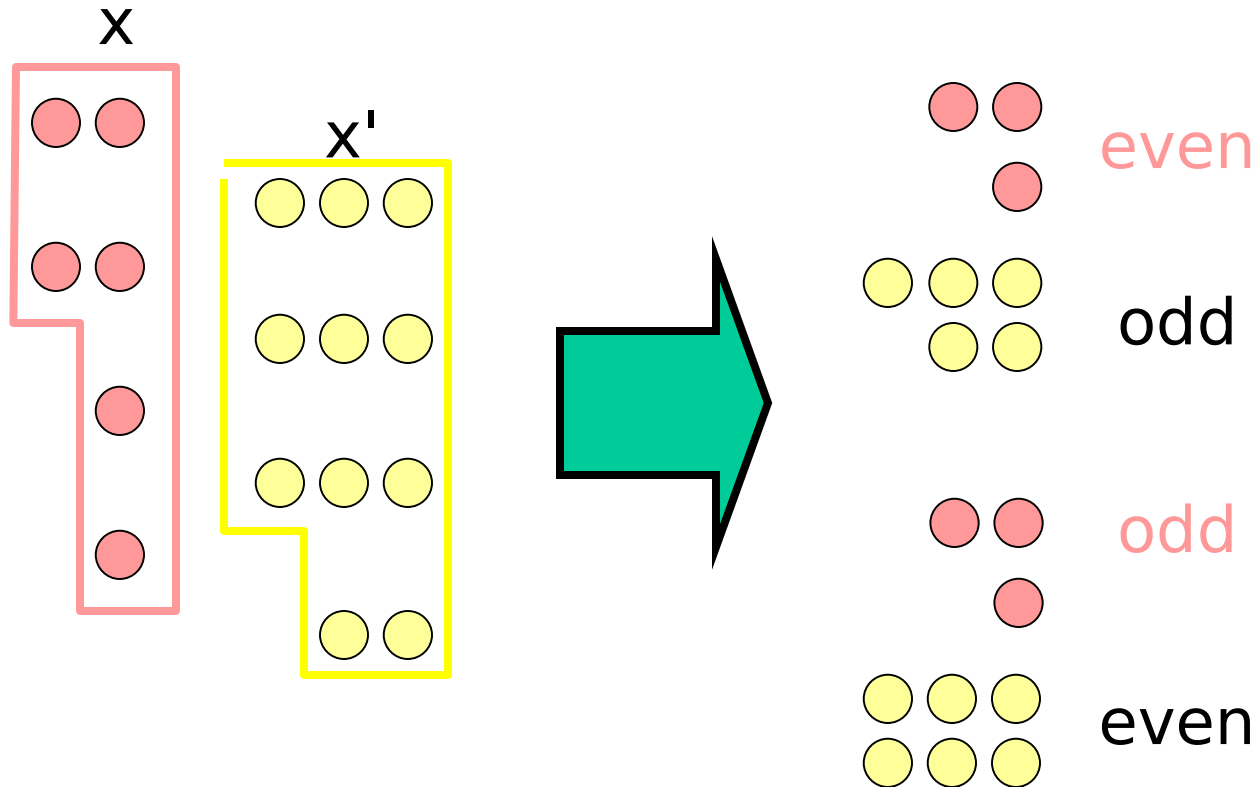
$$\rightarrow j = i = m-1 \pmod k$$

Merger[2k] Schematic



Proof Outline

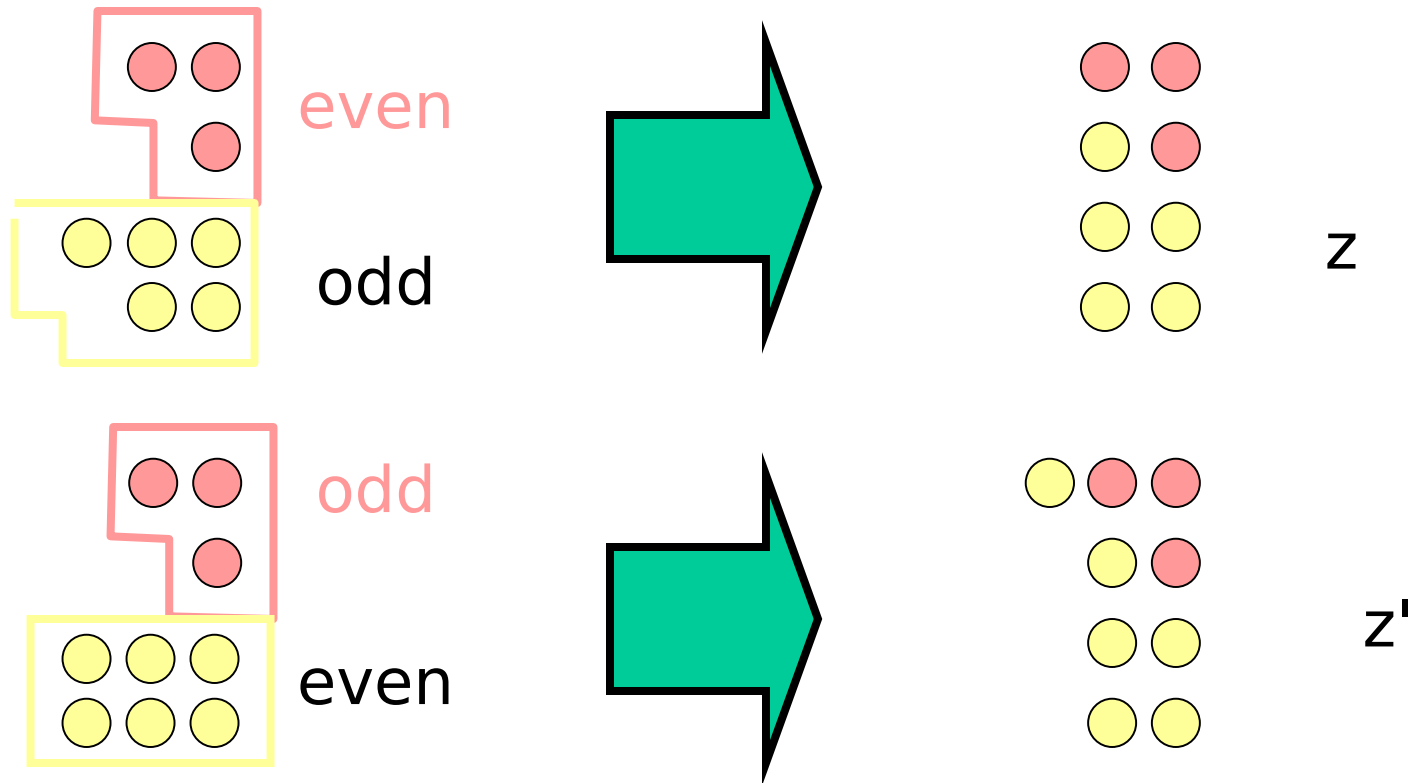
(according to Lemma 1)



Outputs from Bitonic[k]

Inputs to Merger[k]

Proof Outline



Inputs to Merger[k]

Outputs of Merger[k]

Proof

according to lemma1 odd and even subsequences of x, x' have the step property. Therefore z and z' have the step property(induction hypotheses)

Proof

according to lemma 2

$$(\sum z_i) = \left\lceil \sum_{i=0}^{k-1} x_i/2 \right\rceil + \left\lceil \sum_{i=0}^{k-1} x'_i/2 \right\rceil$$

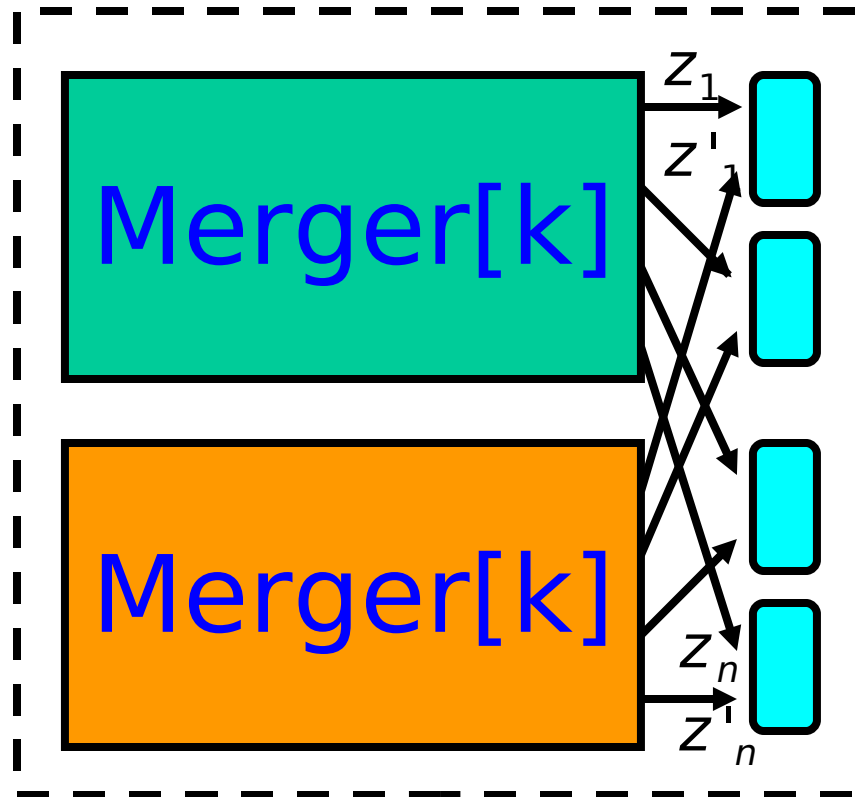
$$(\sum z'_i) = \left\lceil \sum_{i=0}^{k-1} x'_i/2 \right\rceil + \left\lceil \sum_{i=0}^{k-1} x_i/2 \right\rceil$$

therefore z and z' can differ by at most 1.

case a): $z = z'$: (Lemma 3) $z_i = z'_i, 0 \leq i < k/2$

after the final layer of balancers: $y_i = y_j \forall 0 \leq i < j \leq k-1$

Merger[2k] Schematic



Proof

case b: z and z' differs by one.

Lemma 4 implies that

$$(\exists l: 0 \leq l < k/2 : z_l = z'_l + 1 \vee z'_l = z_l + 1) \wedge \forall i: 0 \leq i < k/2, i \neq l, z_i = z'_i$$

Let $\max(z'_j, z_j) = x + 1, \min(z'_j, z_j) = x, x \in \mathbb{N}$

From the step property of z, z' :

$$\text{for all } i < l : z_i = z'_i = x + 1 \wedge \text{for all } i > l : z_i = z'_i = x$$

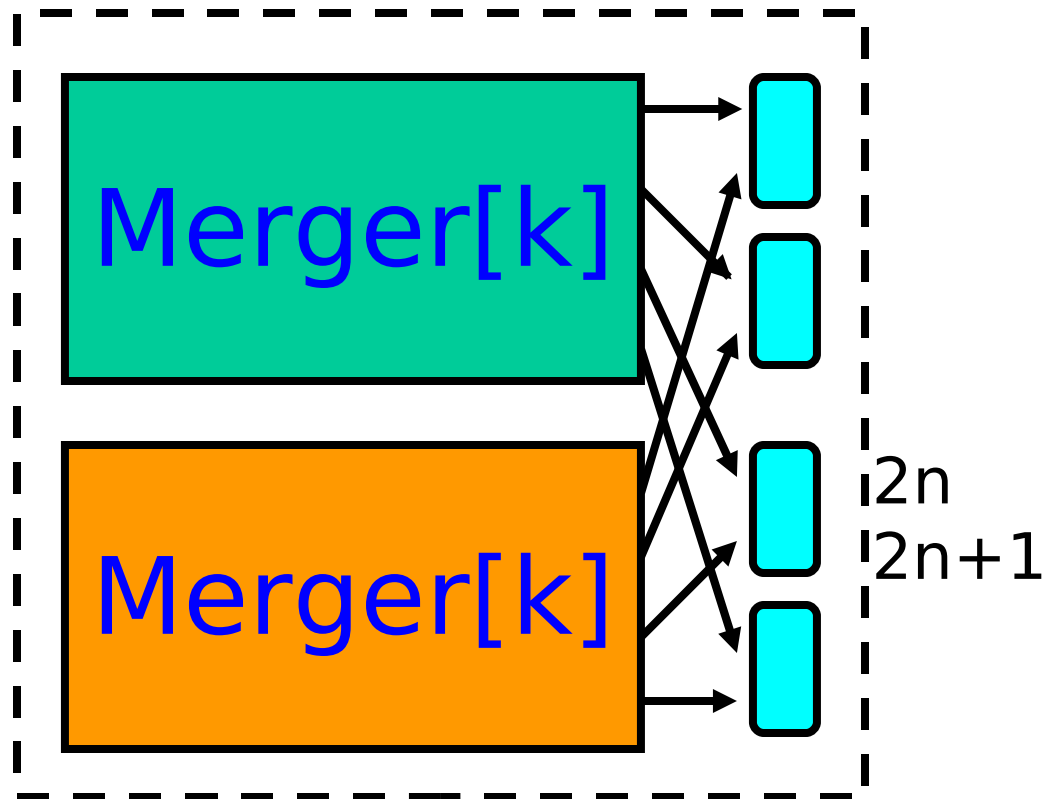
Proof

Because z_l and z'_l are connected by a merger with outputs y_{2l} and y_{2l+1} : $y_{2l} = x + 1, y_{2l+1} = x$

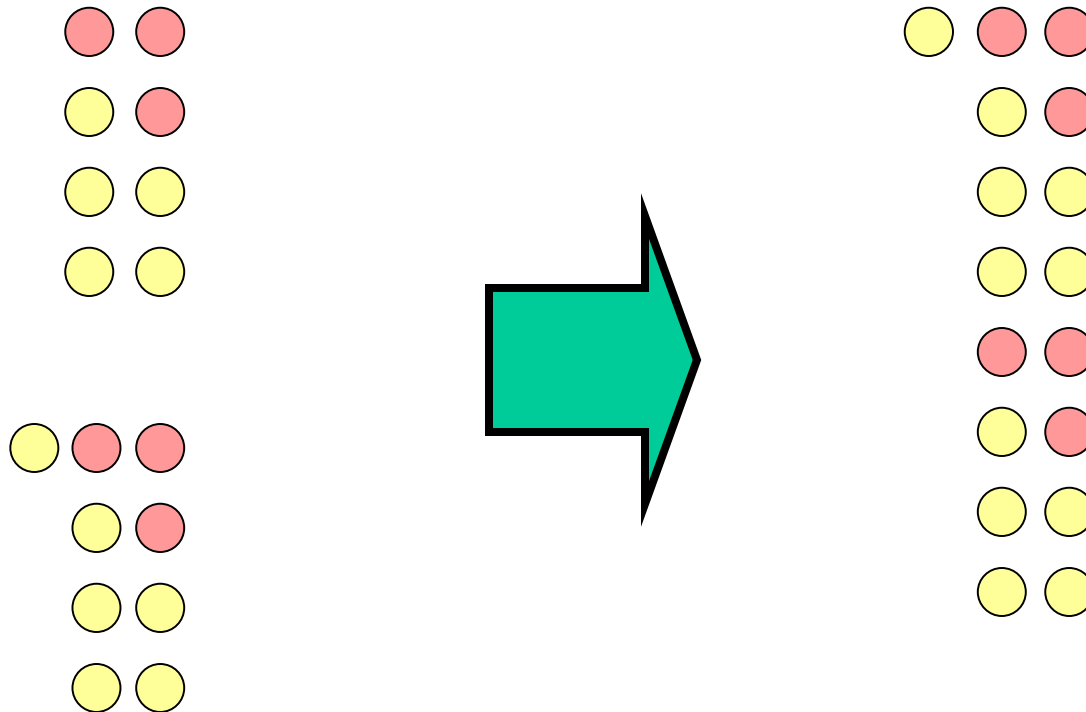
Similar : *for all $i < l : z_i = z'_i = x + 1 \wedge$ for all $i > l : z_i = z'_i = x$*

set $c = 2l + 1$. --> step property

Merger[2k] Schematic



Proof Outline



Outputs of Merger[k]

Outputs of last layer