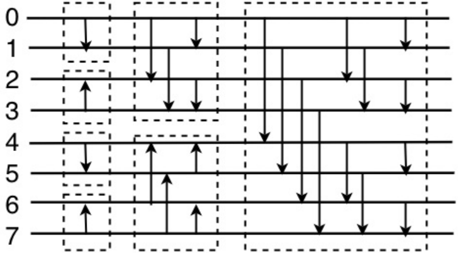
Sorting Omega Networks Simulated with P Systems: Optimal Data Layouts

Rodica Ceterchi, Mario J. Pérez Jiménez, Alexandru I. Tomescu

August 2008

- R. Ceterchi, M.J. Pérez Jiménez, A.I. Tomescu, "Simulating the Bitonic Sort Using P Systems", *G. Eleftherakis et al. (Eds.): WMC8 2007, LNCS,* vol. 4860, pp. 172-192, 2007.
- R. Ceterchi, M.J. Pérez Jiménez, A.I. Tomescu, "Sorting omega networks simulated with P systems", BWMC'08

Bitonic Sorting



- one input key on each wire
- ascending / descending comparators between wires
- sorts N keys in $O(\log^2 N)$ time

One membrane

- each wire has one associated value
- one wire can communicate with any other wire
- time complexity of the original sorting network, $O(\log^2 N)$
- we want to sort in ascending order the sequence of *distinct* integers

$$\langle x_0, x_1, \cdots x_{N-1} \rangle,$$

  codified as the multiset

$$w = {v_0}^{x_0} {v_1}^{x_1} \cdots {v_{N-1}}^{x_{N-1}}.$$

One membrane

- ascending comparator
  $C_{\oplus} = \{v_0 \to a, v_1 \to b\} \cup \{ab \to c^+ d^+, a \to d^+, b \to d^+\} \cup$
  $\{c^+ \to v_0, d^+ \to v_1\}$.
- descending comparator
  $C_{\ominus} = \{v_0 \to a, v_1 \to b\} \cup \{ab \to c^- d^-, a \to c^-, b \to c^-\} \cup$
  $\{c^- \to v_0, d^- \to v_1\}$

$$v_0{}^{x_0} v_1{}^{x_1} \to a^{x_0} b^{x_1} \to c^{+\,min(x_0,x_1)} d^{+\,max(x_0,x_1)} \to v_0{}^{min(x_0,x_1)} v_1{}^{max(x_0,x_1)}$$

$$S^+ = \{s_0{}^+, \cdots, s^+_{2^{2k}-1}\}, S^- = \dots$$

$$T^+ = \{t_0{}^+, \cdots, t^+_{2^{2k}-1}\}, T^- = \dots$$

One membrane

- Rewrite all symbols of V into start symbols for appropriate comparators, using the sets of rules

$$\{v_{2j} \rightarrow {s_{2j}}^+, v_{2j+1} \rightarrow {s_{2j+1}}^+ \mid 0 \leq j \leq 2^{2k-1} - 1 \text{ , j even}\} \cup$$
$$\cup \{v_{2j} \rightarrow {s_{2j}}^-, v_{2j+1} \rightarrow {s_{2j+1}}^- \mid 0 \leq j \leq 2^{2k-1} - 1 \text{ , j odd}\}.$$

- Apply in parallel the rewritings of symbols which correspond to the simulations of the comparators:

$$\{{s_{2j}}^+ {s_{2j+1}}^+ \rightarrow {t_{2j}}^+ {t_{2j+1}}^+, {s_{2j}}^+ \rightarrow {t_{2j+1}}^+, {s_{2j+1}}^+ \rightarrow {t_{2j+1}}^+ \mid$$
$$0 \leq j \leq 2^{2k-1} - 1 \text{ , j even}\} \bigcup$$
$$\cup \{{s_{2j}}^- {s_{2j+1}}^- \rightarrow {t_{2j}}^+ {t_{2j+1}}^-, {s_{2j}}^- \rightarrow {t_{2j}}^-, {s_{2j+1}}^- \rightarrow {t_{2j}}^- \mid$$
$$0 \leq j \leq 2^{2k-1} - 1 \text{ , j odd}\}.$$

- Rewrite back all symbols of T's into V.

$$\{v_{2j} \leftarrow {t_{2j}}^+, v_{2j+1} \leftarrow {t_{2j+1}}^+ \mid 0 \leq j \leq 2^{2k-1} - 1 \text{ , j even}\} \cup$$
$$\cup \{v_{2j} \leftarrow {t_{2j}}^-, v_{2j+1} \leftarrow {t_{2j+1}}^- \mid 0 \leq j \leq 2^{2k-1} - 1 \text{ , j odd}\}.$$

One membrane for each wire

- use a P System with Dynamic Communication Graphs
- for an input of size N, take N membranes each holding two objects / values
- membranes are disposed on a 2D mesh architecture $\sqrt{N} \times \sqrt{N}$ (each membrane can communicate only with its four vertical and horizontal neighbors)
- use the *shuffled row-major* indexing on the mesh to minimize communication
- time complexity $O(\sqrt{N})$, optimal for the 2D mesh

P Systems with Dynamic Communication Graphs

A *P system with dynamic communication graphs* is a tuple

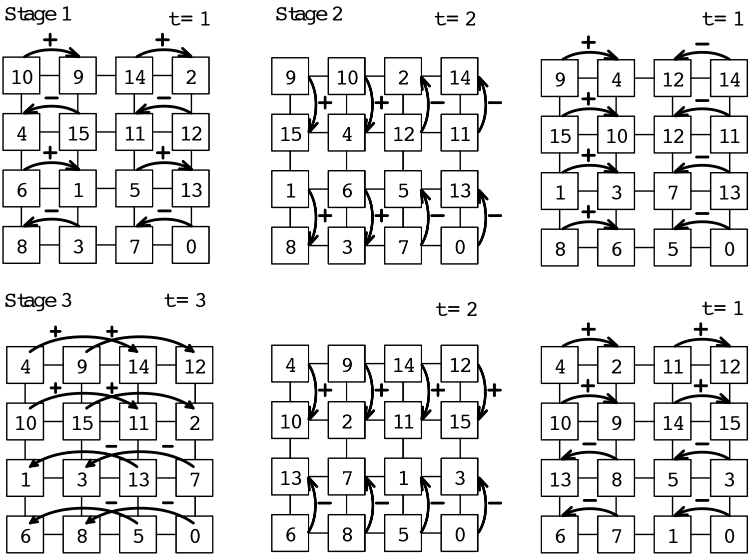
$$\Pi =< V, P_0, \cdots, P_{k-1}, R_\mu = \{R_i, G_i\}_{i\in I} >,$$

where:

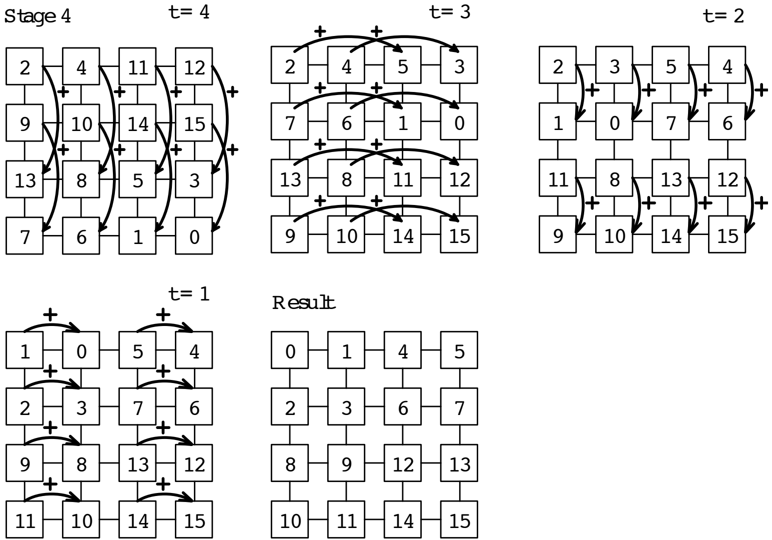
- V is an alphabet of symbols (used to codify integers contained in membranes).
- $P_0, \cdots, P_{k-1}$ are elementary membranes.
- $R_\mu = \{R_i, G_i\}_{i\in I}$ is a sequence of pairs $[rules, graph]$.
    - If $graph \subseteq G_{Id}$, its rules are *rewriting rules*.
    - If $graph \subseteq G_{total} \setminus G_{Id}$, its rules are *communication rules*.

A P system with dynamic communication graphs with *finite sequential support* if R_μ if a finite sequence.

Bitonic sorting on a 2D mesh of 4 × 4



Bitonic sorting on a 2D mesh of 4×4



compare-interchange-membr$(a, i, order)$

forall $j \leftarrow 0$ **to** $2^{i-1} - 1$ **in parallel do**

  // route right one unit in the B registers - (rAB) rule

  $E(Id_1^t) \leftarrow E(Id_1^t) \cup \{(j, j)\}$; $rules_{0,1}^t((j, j)) \leftarrow \{a \rightarrow a^*\}$

  $E(G_0^t) \leftarrow E(G_0^t) \cup \{(j, j+1)\}$; $rules_0^t((j, j+1)) \leftarrow \{(a^*, out)\}$

  $E(Id_2^t) \leftarrow E(Id_2^t) \cup \{(j+1, j+1)\}$;

  $rules_{0,2}^t((j+1, j+1)) \leftarrow \{a^* \rightarrow b\}$

for $k \leftarrow 1$ **to** $2^{i-1} - 1$ **do**

  // shift the values to the second half of the array - (rBB) rule

  forall $j \leftarrow 0$ **to** $2^{i-1} - 1$ **in parallel do**

    $E(G_k^t) \leftarrow E(G_k^t) \cup \{(j+k, j+1+k)\}$

    $rules_k^t((j+k, j+1+k)) \leftarrow \{(b, out)\}$

forall $j \leftarrow 2^{i-1}$ **to** $2^i - 1$ **in parallel do**

  // compare internally - (C) rule

  $E(Id_C^t) \leftarrow E(Id_C^t) \cup \{(j, j)\}$

  if *order is ascending* **then**

    $rules_C^t((j, j)) \leftarrow \{ab \rightarrow ab, a \rightarrow b, b \rightarrow b\}$

  else

    $rules_C^t((j, j)) \leftarrow \{ab \rightarrow ab, a \rightarrow a, b \rightarrow a\}$

for $k \leftarrow 2^{i-1} - 1$ **downto** 1 **do**

  // shift back the results - (rBB) rule

forall $j \leftarrow 0$ **to** $2^{i-1} - 1$ **in parallel do**

  // final routing back in the A registers - (rBA) rule

end

Analysis of proposed models

Advantages:

- preserve the time complexity of the original parallel architecture

Disadvantages:

- the evolution rules depend on the size of the input
- the number of membranes depends on the size of the input
- the communication graphs depend on the size of the input
- big communication overhead

Something in between

Suppose we have a system of fixed size n

- to sort N numbers use $P = N/n$ such systems and "combine" the results
- the evolution rules do not depend on the size of the input
- "combination" of results is done according to easy-to-compute communication graphs

Framework:

- sort N values using $P = N/n$ systems, according to a bitonic sorting algorithm ($N > P$)
- we call *data layout* a function $\mathcal{D} : \{0, \ldots, N-1\} \rightarrow \{0, \ldots, P-1\}$.
- $\mathcal{D}(i) = j$ iff wire i is mapped to system j.
- comparisons are allowed only between wires assigned to the same system

Goal:

- find a sequence of data layouts such that communication between systems is minimized

Sequences of data layouts at work

- at each step s of the omega network we have comparators between wires differing on bit s

Sequences of data layouts at work

- at each step s of the omega network we have comparators between wires differing on bit s
- the 1st data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{\log N}$ is mapped to system $j = a_{n+1} \dots a_{\log N}$

Sequences of data layouts at work

- at each step s of the omega network we have comparators between wires differing on bit s
- the 1st data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{\log N}$ is mapped to system $j = a_{n+1} \dots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally

Sequences of data layouts at work

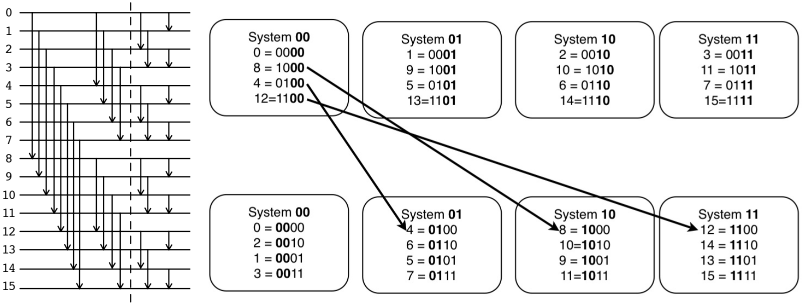
- at each step s of the omega network we have comparators between wires differing on bit s
- the 1st data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{\log N}$ is mapped to system $j = a_{n+1} \dots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally
- remap values according to the 2nd data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{2n} a_{2n+1} \dots a_{\log N}$ is mapped to system $j = a_1 \dots a_n a_{2n+1} \dots a_{\log N}$

Sequences of data layouts at work

- at each step s of the omega network we have comparators between wires differing on bit s
- the 1st data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{\log N}$ is mapped to system $j = a_{n+1} \dots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally
- remap values according to the 2nd data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{2n} a_{2n+1} \dots a_{\log N}$ is mapped to system $j = a_1 \dots a_n a_{2n+1} \dots a_{\log N}$
- sort internally

Sequences of data layouts at work

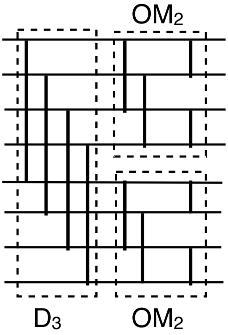
- at each step s of the omega network we have comparators between wires differing on bit s
- the 1st data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{\log N}$ is mapped to system $j = a_{n+1} \dots a_{\log N}$
- perform $\log n$ steps of the sorting algorithm internally
- remap values according to the 2nd data layout: wire $i = a_1 \dots a_n a_{n+1} \dots a_{2n} a_{2n+1} \dots a_{\log N}$ is mapped to system $j = a_1 \dots a_n a_{2n+1} \dots a_{\log N}$
- sort internally
- and so on ...



0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
System 00
0 = 0000
8 = 1000
4 = 0100
12=1100
System 01
1 = 0001
9 = 1001
5 = 0101
13=1101
System 10
2 = 0010
10 = 1010
6 = 0110
14=1110
System 11
3 = 0011
11 = 1011
7 = 0111
15=1111
System 00
0 = 0000
2 = 0010
1 = 0001
3 = 0011
System 01
4 = 0100
6 = 0110
5 = 0101
7 = 0111
System 10
8 = 1000
10=1010
9 = 1001
11=1011
System 11
12 = 1100
14 = 1110
13 = 1101
15 = 1111

Omega networks

Let D_k, $k \geq 1$ be a one-step network of $N = 2^k$ lines with a device between the pair of lines $(i, i + N/2)$, for $i = 0 \ldots N/2 - 1$. Then the omega network OM_k is recursively defined as $OM_k = D_k(OM_{k-1} \circ OM_{k-1})$.

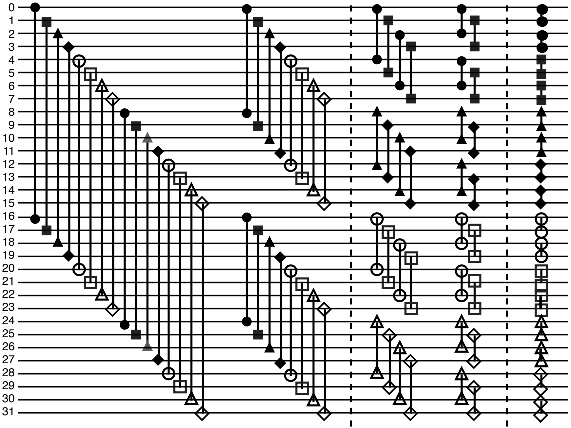


Optimal data layouts for Omega networks

- We have to sort $N = 2^k$ keys using P processors, $N > P$, each processor holding $n = N/P$ keys.
- The number of parallel steps of OM_k is k, and step t of the omega network OM_k contains devices linking lines whose bit representations differ of bit t, with $1 \leq t \leq k$.
- $bc_t : \{0, 1, \ldots, 2^k - 1\} \longrightarrow \{0, 1, \ldots, 2^k - 1\}$, the bit complement of the t-th bit,

$$bc_t(a_1 a_2 \cdots a_t \cdots a_k) = a_1 a_2 \cdots \bar{a}_t \cdots a_k$$

An omega network of size 32. Three data layouts for the omega network OM_5.



(c) An omega network of size 32. Lines marked with same shape are assigned to the same processor in one data layout.

$\boxed{0\ 0}\ 0\ 0\ 0 = 0$
$\boxed{1\ 0}\ 0\ 0\ 0 = 16$
$\boxed{0\ 1}\ 0\ 0\ 0 = 8$
$\boxed{1\ 1}\ 0\ 0\ 0 = 24$

$0\ 0\ \boxed{0\ 0}\ 0 = 0$
$0\ 0\ \boxed{1\ 0}\ 0 = 4$
$0\ 0\ \boxed{0\ 1}\ 0 = 2$
$0\ 0\ \boxed{1\ 1}\ 0 = 6$

$0\ 0\ 0\ \boxed{0\ 0} = 0$
$0\ 0\ 0\ \boxed{1\ 0} = 2$
$0\ 0\ 0\ \boxed{0\ 1} = 1$
$0\ 0\ 0\ \boxed{1\ 1} = 3$

(d) Keys mapped to processor 0

More formally...

Lemma

At each step $1 \leq t \leq k$ of OM_k, and for any $0 \leq i < 2^k$, line i is linked by a device only with line $bc_t(i)$.

Lemma

In OM_k, for any $0 \leq i < 2^{k-m}$, $1 \leq m \leq k$ and $0 \leq t \leq k - m$, in steps $t + 1, \ldots, t + m$ there is no device linking lines in the set $P_i^{t,m} = \{a_1 a_2 \cdots a_k \mid a_1 \cdots a_t a_{t+m+1} \cdots a_k = i$, where $a_1 \cdots a_k$ is a bit representation$\}$ with lines from $\{0, \ldots, 2^k - 1\} \setminus P_i^{t,m}$.

More formally...

Definition

Given $N = 2^k$ keys and $P = 2^{k-m}$ processors, which can store $n = 2^m$ values, $m \geq 1$, the sequence of optimal data layouts consists of $\lceil \log N / \log n \rceil = \lceil k/m \rceil$ data layouts. In each data layout $\mathcal{D}_s$, $0 \leq s \leq \lceil k/m \rceil - 1$, values in the set $P_i^{sm,m}$ are mapped to processor P_i, for all $0 \leq i \leq 2^{k-m}$. More formally, for any $0 \leq u < 2^k$ such that $u \in P_i^{sm,m}$, we have $\mathcal{D}_s(u) = i$.

Lemma

The maximum number of successive steps of the omega network that can be executed locally, under any data layout is $\log n$*, where* $n = N/P$*.*

Inside one processor/membrane

Inside one processor, several comparisons are performed, in parallel, between the n pieces of data

for $t \leftarrow 1$ **to** m **do**
  forall $i < bc_t(i)$ **in parallel do**
    compare$(a_i, a_{bc_t(i)})$;

Algorithm 1: A parallel algorithm for the bitonic merger

The parallel comparisons at each step t

forall $i < bc_t(i)$ **in parallel do**
    compare$(a_i, a_{bc_t(i)})$;

will be simulated in a membrane P by the rules

$$\{a_i \to s_i \mid i = 0, 1, \cdots, n-1\} \cup$$
$$\cup\ \{s_i s_j \to t_i t_j, s_i \to t_j, s_j \to t_j \mid i = 0, 1, \cdots, n{-}1, i < j = bc_t(i)\} \cup$$
$$\cup\ \{t_i \to a_i \mid i = 0, 1, \cdots, n-1\}.$$

A P System with dynamic communication graphs which simulates the Omega Network

$$\Pi =< V = \{a_0, \ldots, a_{n-1}\} \cup \mathcal{A},$$

$$\langle [a_0^{x_0^0}, a_1^{x_1^0}, \ldots, a_{n-1}^{x_{n-1}^0}]_0, \ldots, [a_0^{x_0^{P-1}}, a_1^{x_1^{P-1}}, \ldots, a_{n-1}^{x_{n-1}^{P-1}}]_{P-1}\rangle, R_\mu >$$

R_μ sequence of pairs $[graph, rules]$. R_μ is generated algorithmically.

Lemma

Given $N = 2^k$ *keys and* $P = 2^{k-m}$ *membranes, which can store* $n = 2^m$ *values,* $m \geq 1$*, after the computation for the data layout* $\mathcal{D}_s$ *is finished, symbol* a_i *of membrane* j *codifies the value corresponding to wire* $u \in \{0, \ldots, N-1\}$*, where the bit representation of* u *is* $u = j_1 \ldots j_{sm} i_1 \ldots i_m j_{sm+1} \ldots j_{k-m}$*. By* $j_1 \ldots j_{k-m}$ *and by* $i_1 \ldots i_m$ *we denoted the bit representations of* j*, and* i*, respectively.*

Algorithmic generation of the communication graph

$E(C_s^j) \leftarrow \emptyset$;
for $j \leftarrow 0$ **to** $P-1$ **do**
  for $i \leftarrow 0$ **to** $n-1$ **do**
    let j have bit representation $j_1 \cdots j_{sm} j_{sm+1} \cdots j_{k-m}$;
    let i have bit representation $i_1 \cdots i_m$;
    `// the destination membrane of value encoded by`
    `a_i in membrane j`
    $z \leftarrow j_1 \cdots j_{sm} i_1 \cdots i_m j_{(s+1)m+1} \cdots j_{k-m}$;
    `// the destination symbol of value encoded by`
    `a_i in membrane j`
    $t \leftarrow j_{sm+1} \cdots j_{sm+m}$;
    $E(C_s^j) := E(C_s^j) \cup \{(j,z)\}$;
    $\text{rules}_{C_s^j}((j,z)) := a_i \rightarrow a'_t$;

Algorithm 2: Generation of the sequence of P communication graphs when passing from data layout $\mathcal{D}_{s-1}$ to $\mathcal{D}_s$, with $0 < s \leq$

Algorithmic generation of internal computation rules

$SimOM \leftarrow \lambda$;
for $t \leftarrow 1$ **to** $m = \log n$ **do**
  forall $p \leftarrow 0$ **to** $P-1$ **in parallel do**
    $\text{rules}_{t,1}((p,p)) \leftarrow \{a_i \rightarrow s_i \mid i = 0, 1, \ldots n-1\}$;
    $\text{rules}_{t,2}((p,p)) \leftarrow \{s_i s_j \rightarrow t_i t_j, s_i \rightarrow t_j, s_j \rightarrow t_j \mid i = 0, 1, \ldots, n-1, i < j = bc_t(i)\}$;
    $\text{rules}_{t,3}((p,p)) \leftarrow \{t_i \rightarrow a_i \mid i = 0, 1, \cdots n-1\}$;
  $SimOM \leftarrow SimOM \cdot [Id, \text{rules}_{t,1}] \cdot [Id, \text{rules}_{t,2}] \cdot [Id, \text{rules}_{t,3}]$;

Algorithm 3: Generation of the sequence *SimOM* which simulates the omega network of size n.

Algorithmic generation of R_μ

$R_\mu \leftarrow \lambda$;
for $s \leftarrow 1$ **to** $\lceil k/m \rceil - 1$ **do**
    $R_\mu \leftarrow R_\mu \cdot SimOM$;
    for $j \leftarrow 0$ **to** $P - 1$ **do**
        $R_\mu \leftarrow R_\mu \cdot [C_s^j, \text{rules}_{C_s^j}]$;
    $R_\mu \leftarrow R_\mu \cdot [Id, \text{rules-endcomm}]$;
$R_\mu \leftarrow R_\mu \cdot SimOM$;

Algorithm 4: Generation of the sequence R_μ which guides the computation.

Computation Complexity

- we have $\frac{\log N}{\log n}$ data layouts;
- in each data layout we have
  - $3 \log n$ steps are needed for *SimOM*;
  - $P + 1 = \frac{N}{n} + 1$ steps are needed for communication;
- hence the length of R_μ is $3 \log N + \frac{N \log N}{n \log n}$;

- a sorting network can be obtained by a serial connection of $\log N$ omega networks, giving a time complexity of

$$O(\log^2 N + \frac{N \log^2 N}{n \log n})$$

- for $n = N$ the complexity is $O(\log^2 N)$;
- for $n = 2$ the complexity increases to $O(N \log^2 N)$.

Bibliography

- R. Ceterchi, M.J. Pérez Jiménez, A.I. Tomescu, "Simulating the Bitonic Sort Using P Systems", *G. Eleftherakis et al. (Eds.): WMC8 2007, LNCS,* vol. 4860, pp. 172-192, 2007.
- R. Ceterchi, M.J. Pérez Jiménez, A.I. Tomescu, "Sorting omega networks simulated with P systems", BWMC'08
- M.F. Ionescu, "Optimizing Parallel Bitonic Sort", *Tech. Report TRCS96-14,* Dept. of Comp. Sci., Univ. of California, Santa Barbara, July 1996.
- M.F. Ionescu, K.E. Schauser, "Optimizing parallel bitonic sort", *Proc. 11th Int'l Parallel Processing Symp.,* pp. 303-309, 1997.
- D.E. Knuth, *The art of computer programming*, volume 3: sorting and searching, second ed. Redwood City, CA: Addison Wesley Longman, 1998.
- K.E. Batcher, "Sorting networks and their applications", *Proc. AFIPS Spring Joint Comput. Conf.,* vol. 32, pp. 307-314, Apr. 1968