ACT-46 FEBRUARY 1984

CSL COORDINATED SCIENCE LABORATORY
APPLIED COMPUTATION THEORY GROUP

THE VLSI OPTIMALITY OF THE AKS SORTING NETWORK

BILARDI, GIANFRANCO
PREPARATA, FRANCO P.

REPORT R-1008 UILU-ENG 84-2202

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release, distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) R-report # 1008; UILU-ENG 84-2202; ACT-46	5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A
6a. NAME OF PERFORMING ORGANIZATION Coordinated Science Laboratory, Univ. of Illinois 6b. OFFICE SYMBOL (If applicable) N/A	7a. NAME OF MONITORING ORGANIZATION Joint Services Electronics Program
6c. ADDRESS (City, State and ZIP Code) 1101 W. Springfield Avenue Urbana, IL 61801	7b. ADDRESS (City, State and ZIP Code) 800 N. Quincy Street Arlington, VA
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Joint Services Electronics Program 8b. OFFICE SYMBOL (If applicable) N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER Contract N00014-79-C-0424
8c. ADDRESS (City, State and ZIP Code) 800 N. Quincy St. Arlington, VA	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: N/A PROJECT NO.: N/A TASK NO.: N/A WORK UNIT NO.: N/A
11. TITLE (Include Security Classification) The VLSI Optimality of the AKS Sorting Network	
12. PERSONAL AUTHOR(S) Bilardi, Gianfranco and Preparata, Franco P.	
13a. TYPE OF REPORT 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day) February 1984 15. PAGE COUNT 11
16. SUPPLEMENTARY NOTATION N/A	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Sorting networks, VLSI complexity, optimal VLSI networks

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A VLSI implementation is given for the sorting network proposed by Ajtai, Komlos, and Szemeredi, which can be laid out in $O(n^2)$ area and works in $O(\log n)$ time. This performance is optimal under the (synchronous) VLSI model of computation.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code) 22c. OFFICE SYMBOL NONE

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

THE VLSI OPTIMALITY OF THE AKS SORTING NETWORK

G. Bilardi and F. P. Preparata
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign

Introduction

Ajtai, Komlos, and Szemeredi [1] recently proposed a sorting network (referred to hereafter as the AKS network), of O(nlogn) comparators and O(logn) depth. Their construction is of great theoretical interest, for it shows that O(nlogn) comparisons suffice to sort n elements, even under the constraint that comparisons be nonadaptively executed in O(logn) parallel stages. At present, the AKS network appears not suitable for practical implementations, due to the large value of the constants; however, improvements are conceivable that could make the network more attractive for real-world applications.

It is therefore natural to ask what is the performance of the AKS network in the synchronous VLSI model of computation which has been proposed [2] to capture the essential features of planar very large scale integration as a computing environment.

In this model it is known that any chip capable of sorting n words of length $q = (1+\alpha)\log n$, with $\alpha > 0$, must satisfy the relationship $AT^2 = \Omega(n^2 \log^2 n)$, where A is the chip area, and T is the computation time. This lower bound has been originally obtained by Thompson [2] under the word local restriction (all the bits of the same word enter the circuit at the same input port). Recently Leighton [3] has shown that the lower bound holds valid even for non-word-local designs.

This work has been supported in part by the Joint Services Electronics Program under Contract N00014-79-C-0424 and by the IBM Predoctoral Fellowship Program.

Many designs of VLSI sorters have already been proposed (see Thompson [4] for a survey). We mention here the ones that achieve minimum area $A = \theta(n^2 \log^2 n/T^2)$ at their computation time T:

- the mesh-connected [1,5,6] bitonic sorter [7], for $T = O(\sqrt{n})$.
- the pleated-cube-connected-cycles (PCCC) [8] also implementing bitonic sorting for T in the range $[\Omega(\log^3 n), O(\sqrt{n \log n})]$.
- a hybrid architecture based on the cube-connected-cycles and the orthogonal trees interconnections [9], which implements the enumeration sorting schemes of [10], and works in minimum computation time $T = O(\log n)$.
- a hybrid architecture consisting of orthogonal trees and permuter networks [3], which implements a generalization of the even-odd sort [7], and also works in time $T = O(\log n)$.

It is then interesting to see how the AKS algorithm, which is radically different from any other known sorting paradigm, compares with more classical sorting methods in the VLSI environment, where the heaviest demand of resources usually comes from communication, rather than from computing requirements, so that a small number of processing elements does not necessarily imply a good performance.

In this note we show that the AKS sorting network can indeed be laid out in area $A = O(n^2)$, while maintaining an $O(\log n)$ computation time, thereby establishing its optimality in the VLSI model of computation.

Layout of the AKS Network

The original description [1] of the AKS network (with n inputs) is given in terms of an n-node graph G = (V,E), whose nodes are registers, and whose edges are comparators. The set of edges E is partitioned as $E = E_1 \cup E_2 \cup \ldots \cup E_N$, where each of the E_s's is a (possibly partial) matching on V, and $N < \beta \log n$ for some (very large) constant β. Since each E_s (s = 1,...,N) is a (possibly partial) matching, all of its comparators can be simultaneously active. Thus the AKS sorting algorithm can be described as follows:

```
begin for s:= 1 to N
        for all (x,y) ∈ E_s, and x < y pardo
            (R(x),R(y)):= (min(R(x),R(y)),max(R(x),R(y)))
end
```

where R(x) is the content of the register associated with node x.

Since the embedding of a graph in a planar grid requires nodes of bounded degree, we shall modify the original description as follows. According to a scheme described by Knuth [11], we consider n lines that run parallel, say, to the horizontal axis. On line r (r = 1,2,...,n) there will be N processors P[r,1],...,P[r,N], whose capability will be specified below. For each s = 1,2,...,N, and for each $(x,y) \in E_s$, we connect processors P[x,s] and P[y,s] by a vertical line. Such vertical line supports the execution of the comparison-exchange (R(x),R(y)) : = (min(R(x),R(y)),max(R(x),R(y))), where R(x) and R(y) are respectively the operands stored in P[x,s] and P[y,s]. Once the comparison-exchanges specified by E_s have been executed, the results will be forwarded on each line (that is, from P[x,s] to P[x,s+1], x = 1,...,n).

This basic layout can be further specified by selecting the degree of parallelism of the operand transmission. Due to the amenability to pipelined operation, the q-bit operands are fed in bit-serial fashion starting with the most significant bit and each processor is equipped with a serial comparator. In each comparator, as long as the two inputs agree, they are transmitted to the next processor on the same line. As soon as a bit discrepancy is detected, a switch is set and, from then on, the remaining substrings of each of the operands will follow a fixed path independently of their value.

Thus we have ensured that the AKS network works in $T = O(\log n+q) = O(\log n)$ time, and we turn our attention to the layout area. We first observe that both the horizontal, and the vertical lines are of $O(1)$ width. It is then simple to conclude that the height of the entire layout is $O(n)$. On the other hand, any matching of n lines can be easily laid out in (at most) $n/2$ vertical tracks of constant width, by using a track for each edge of the matching. Since there are $N = O(\log n)$ matchings to be cascaded in the AKS network, it is readily proved that $O(n\log n)$ width, and therefore $O(n^2\log n)$ area, suffices for the layout. A closer analysis however, reveals that many of the matchings $E_1,\ldots,E_N$ are such that many edges can be laid out, without overlap, in the same vertical track, yielding the conclusion that the bound for the area can be lowered to $O(n^2)$.

To establish this claim we introduce the following top-down description of the layout of the AKS network. The layout could be analyzed as the assembly of suitable simpler building blocks, whose hierarchy is illustrated in Figure 1. Each of these building blocks will now be described in detail, in a top-down fashion.

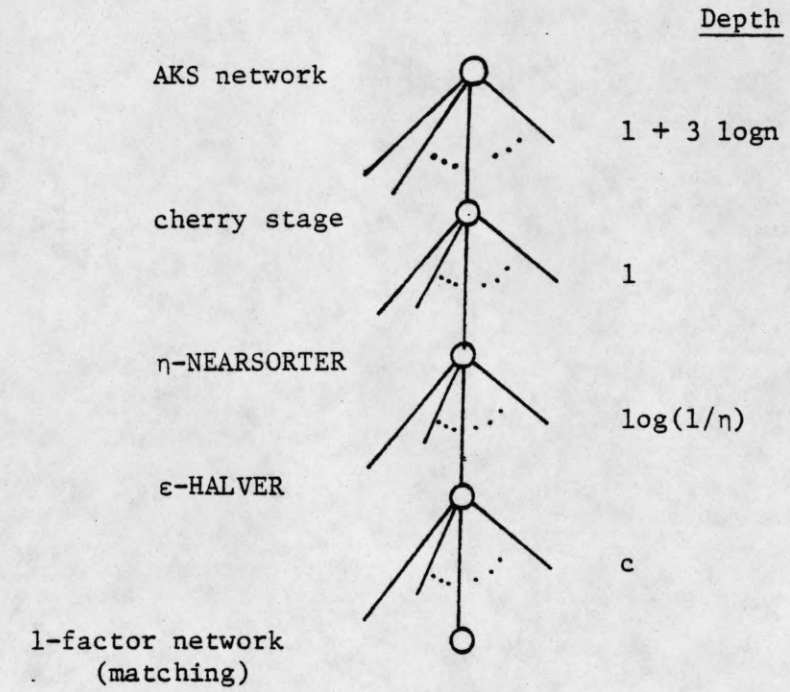

Figure 1. Hierarchy of building blocks of the AKS network. The depth is expressed as the length of the cascade of blocks of the immediately lower level.

(1) The AKS network on $n = 2^d$ inputs is the cascade of (1+3d) stages, called <u>cherry stages</u>, and denoted by $S_0, S_{11}, S_{12}, S_{13}, \ldots, S_{d1}, S_{d2}, S_{d3}$ (Figure 2).

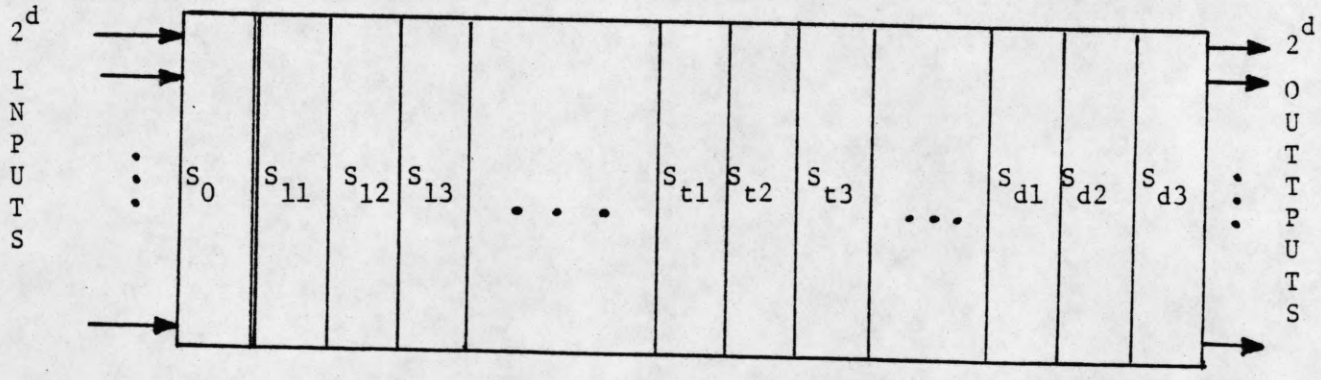

Figure 2. The AKS network on 2^d input is the cascade of (1+3d) cherry stages.

(2) To each cherry stage $S_{t,h}$ ($t = 1,\ldots,d$; $h = 1,2,3$) there corresponds a partition $P_{t,h}$ of the integers (lines) $1,2,\ldots,n$. Although the assignment of the integers to the partition blocks is too complicated to be repeated here (the reader is referred to [1]), what is important now are the properties of $P_{t,h}$ that are relevant to the layout. Specifically, $P_{t,h}$ consists of the following (disjoint) blocks:

$$P_{t1} = P_{t3} = \{T_t(2i,j): i = 0,1,\ldots,\lfloor (t-1)/2 \rfloor;\ j = 1,2,\ldots,2^{2i}\}$$

$$P_{t2} = \{T_t(2i-1),j): i = 1,2,\ldots,\lfloor t/2 \rfloor;\ j = 1,2,\ldots,2^{2i-1}\} \cup \{T_t(-1,0)\}.$$

To stage S_0 there corresponds the trivial partition P_0 consisting of one block only.

If we now define as span(T) the smallest interval of $\{1,\ldots,n\}$ containing $T \subseteq \{1,\ldots,n\}$, we have the following properties:

(1) For given t and i, and $j' \neq j$, $\text{span}(T_t(i,j)) \cap \text{span}(T_t(i,j')) = \phi$.

(2) $|\text{span}(T_t(i,j))| \leq n/2^i$ for every t and j.

(3) $|T_t(i,j)| \leq \gamma\, n/2^i\, A^{i-t}$ for every j, where γ and $A = 2^a > 1$ are constants.

The lines numbered by the integers in a block $T_t(i,j)$ are involved in a network of comparators called an η-nearsorter (see Figure 3). Properties (1) and (2) show that for any fixed t and i, all η-nearsorters corresponding to $\{T_t(i,j) : j = 1,2,\ldots,2^i\}$ can be laid out in the same vertical strip as shown in Figure 3. Moreover, all nearsorters in the same cherry stage can operate in parallel (indeed, no two share a line).

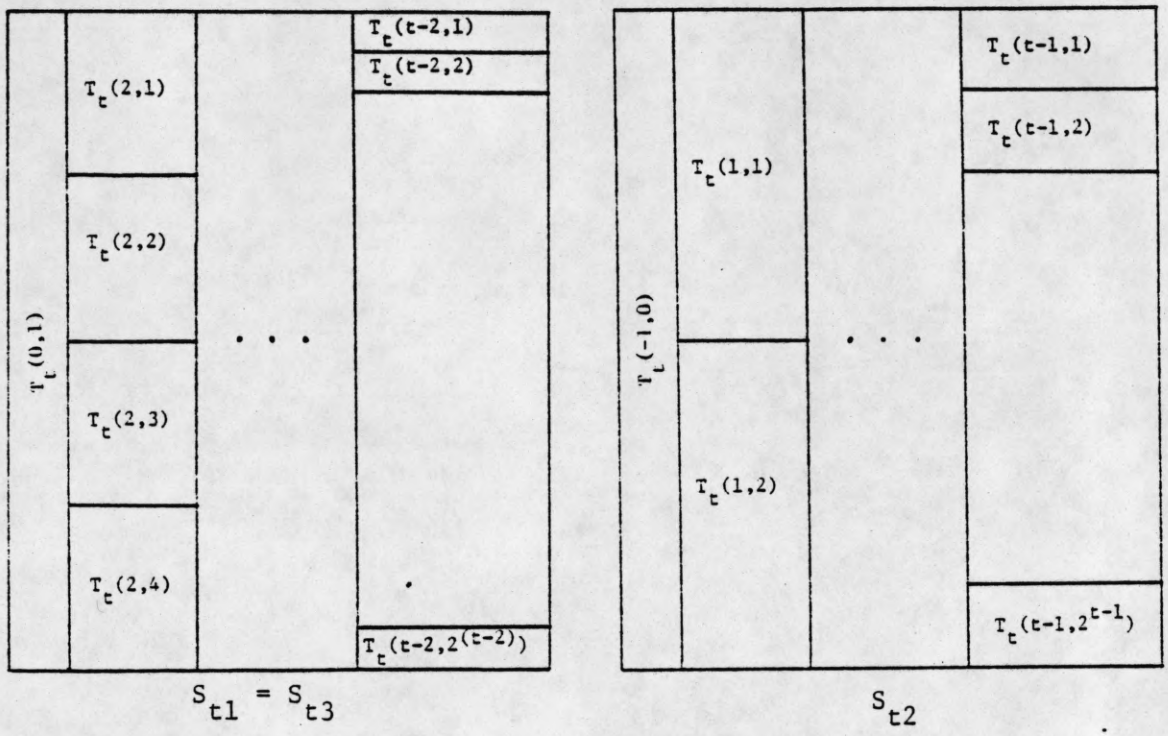

Figure 3. Typical <u>cherry stages</u> S_{t1} and S_{t2} (t is even in the figure). The region labelled $T_t(i,j)$ correspond to the layout of an η-nearsorter.

(3) An η-NEARSORTER, corresponding to block $T_t(i,j)$, has the structure of a full binary tree of depth $\log_2 \frac{1}{\eta}$. Each node of this tree is a network of comparators, called an ε-HALVER (see (4)), encompassing an interval of lines (Figure 4). If $m = |T_t(i,j)|$, then the root encompasses m lines; if a node v of the tree encompasses s lines, then its two offsprings encompass each (approximately) s/2 lines.

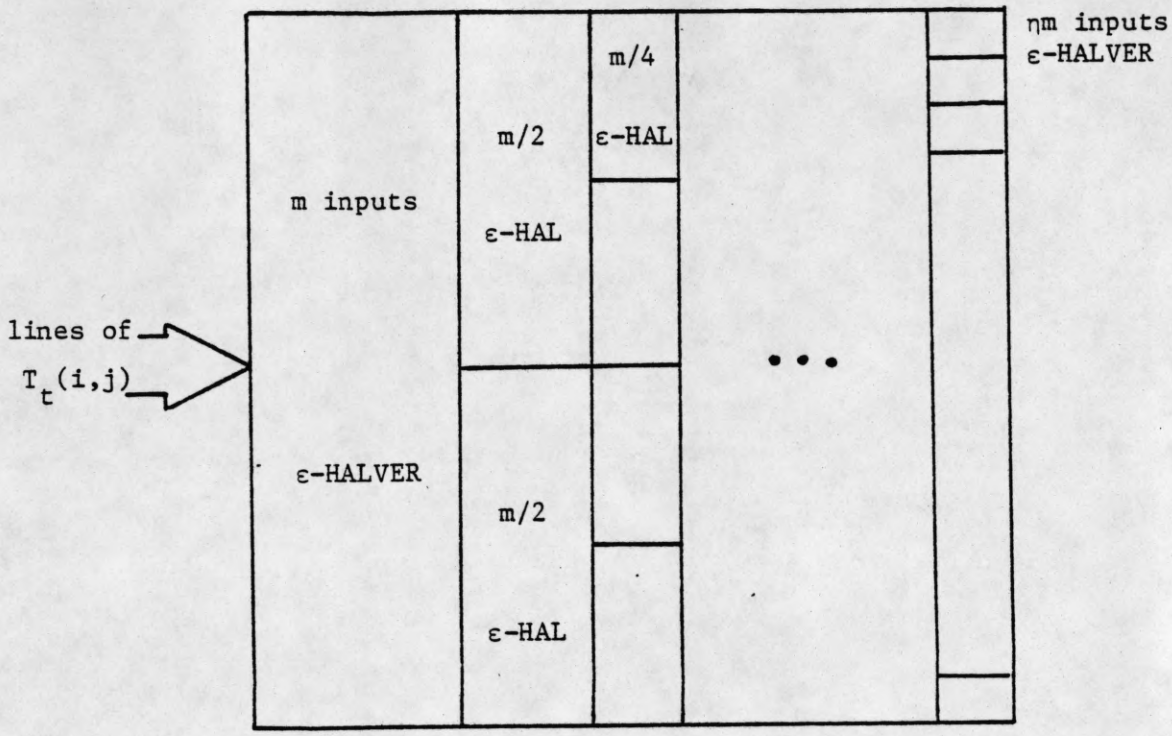

Figure 4. An η-NEARSORTER is a full binary tree of ε-HALVERS.

(4) An ε-HALVER stage on m lines (with $\varepsilon < \eta/(\log 1/\eta)$) consists of the cascade of c (where c is a function of ε, but is independent of m) <u>one-factor stages</u> (matching stages). (When the network is viewed as a graph G = (V,E), i.e. when each line is shrunk to a single node, the ε-HALVER becomes an expander graph on the set of nodes on which its edges are incident.) (See Figure 5.)

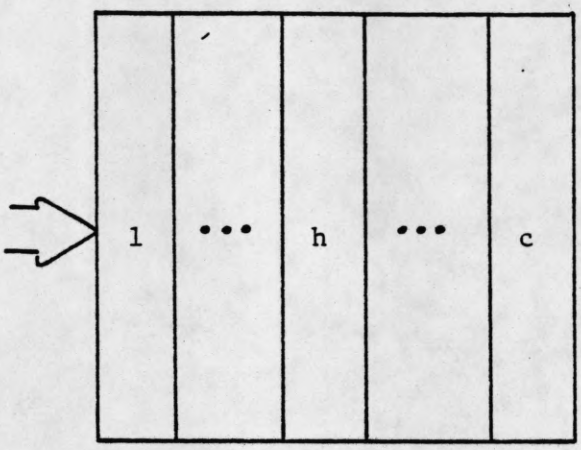

Figure 5. An ε-HALVER is a cascade of a constant number of one-factors.

(5) Finally a one-factor stage on m lines is a matching between the lower and the upper half of these lines, and it is a subset of exactly one of the sets $\{E_s : s = 1,\ldots,N\}$ introduced earlier. (See Figure 6.)

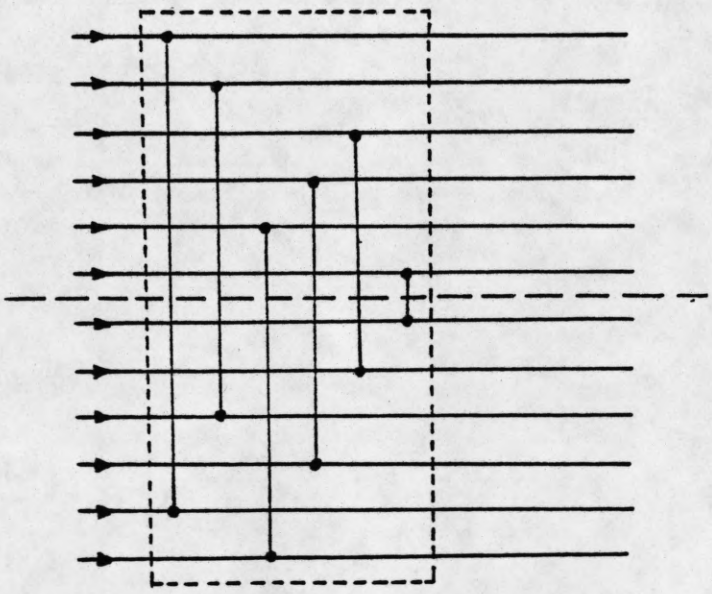

Figure 6. A one-factor is a matching between the top and the bottom half of lines.

Now we proceed, bottom-up, to analyze the area of the network.

(i) A one-factor stage on m lines can be laid out in O(m) length, by allocating a vertical track for each of the m/2 edges. The height of the layout will be proportional to the distance between the topmost and the bottommost of the input lines.

(ii) An ε-HALVER has a length of O(cm); c is the valence of the ε-HALVER.

(iii) An η-NEARSORTER has a length also of O(cm), since the length of the ε-HALVERS decreases geometrically with the level.

(iv) We now subdivide the layout into vertical slabs, with slab(t,i) containing the nearsorter on sets $T_t(i,j)$ for all suitable values of j. (There are in fact two identical copies of $T_t(i,j)$ when i is even, but this will only affect constant factors.) From point (iii) and property (3) it immediately follows that

$$\ell(t,i) \triangleq \text{length of slab}(t,i) \leq \gamma\, 2^{-i} A^{i-t}$$

Then, the total length ℓ can be obtained by summing $\ell(t,i)$ over all the vertical slabs:

$$\ell = \sum_{t=0}^{d} \sum_{i=0}^{t} \ell(t,i) = \sum_{i=0}^{d} \sum_{t=i}^{d} \ell(t,i)$$

$$\leq \gamma n \sum_{i=0}^{d} 2^{-i} \sum_{t=i}^{d} (1/A)^{t-i} \leq \frac{2\gamma}{1-(1/A)}\, n.$$

In conclusion A = height × length = O(n) × O(n) = $O(n^2)$ as claimed.

References

1. M. Aitai, J. Komlos, E. Szemeredi, "An O(NlogN) Sorting Network," Proc. 15th SIGACT, Boston, MA, April 1983, pp. 1-9.

2. C. D. Thompson, A Complexity Theory for VLSI, Ph. D. Thesis, Computer Science Department, Carnegie-Mellon Univ., Aug. 1980.

3. F. T. Leighton, "Tight Bounds on the Complexity of Parallel Sorting," Proc. 16th SIGACT, Washington, D.C., April 1984.

4. C. D. Thompson, "The VLSI Complexity of Sorting", IEEE Trans. Comp., vol. C-32, no. 12, Dec. 1983.

5. C. D. Thompson and H. T. Kung, "Sorting on a Mesh Connected Computer," Comm. of ACM, vol. 20, no. 4, pp. 263-271, April 1977.

6. D. Nassimi and S. Sahni, "Bitonic Sort on a Mesh-Connected Parallel Computer," IEEE Trans. on Computers, vol. C-28, no. 1, pp. 2-7, Jan. 1979.

7. K. E. Batcher, "Sorting Networks and Their Applications," Proc. AFIPS Spring Joint Computer Conference, vol. 32, pp. 307-314, April 1968.

8. G. Bilardi, F. P. Preparata, "A VLSI Optimal Architecture for Bitonic Sorting," Proc. 7th Conf. on Information Sciences and Systems, The Johns Hopkins University, Baltimore, MD, (March 1983); pp. 1-5.

9. G. Bilardi, F. P. Preparata, "A Minimum Area VLSI Architecture for O(logn) Time Sorting," Proc. 16th SIGACT, Washington, D. C., April 1984.

10. F. P. Preparata, "New Parallel Sorting Schemes," IEEE Trans. Comput., vol. C-27, no. 7, pp. 669-673, July 1978.

11. D. E. Knuth, The Art of Computer Programming: Sorting and Searching, Vol. 3, Reading, MA: Addison-Wesley 1973.

Keywords: VLSI complexity, area-time trade-off, sorting networks, optimal algorithms, parallel computation.

Keywords: VLSI complexity, area-time trade-off, sorting networks, optimal algorithms, parallel computation.