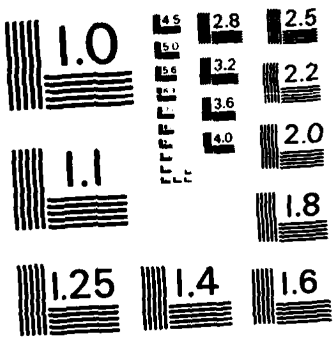
AD-A161 327 AREA-TIME LOWER-BOUND TECHNIQUES WITH APPLICATION TO SORTING(U) ILLINOIS UNIV AT URBANA APPLIED COMPUTATION THEORY GROUP G BILARDI ET AL. 25 APR 85 ACT-59 1/1

UNCLASSIFIED N00014-84-C-0149 F/G 9/5 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS-1963-A

(12)

ACT-59 APRIL 1985

AD-A161 327

AREA-TIME LOWER-BOUND TECHNIQUES WITH APPLICATION TO SORTING

G. BILARDI
F.P. PREPARATA

DTIC
SELECTE D
NOV 20 1985
E

APPROVED FOR PUBLIC RELEASE. DISTRIBUTION UNLIMITED.

DTIC FILE COPY

REPORT R-1042 UILU-ENG 85-2217

N

11 18 - 85 023

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: N/A

1b. RESTRICTIVE MARKINGS: None

2a. SECURITY CLASSIFICATION AUTHORITY: N/A

2b. DECLASSIFICATION/DOWNGRADING SCHEDULE:

3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S): ACT-59 UILU-ENG-85-2217 R-report # R-1042

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: IBM Fellowship, National Science Foundation Joint Services Electronics Program

6c. ADDRESS (City, State and ZIP Code): 1101 W. Springfield Avenue Urbana, Illinois 61801

7b. ADDRESS (City, State and ZIP Code): 800 N. Quincy St. Arlington, VA 22217

8a. NAME OF FUNDING/SPONSORING ORGANIZATION: IBM Fellowship, NSF, JSEP

8b. OFFICE SYMBOL (If applicable): N/A

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: Contract # N00014-84-C-0149

8c. ADDRESS (City, State and ZIP Code): 800 N. Quincy Street Arlington, VA 22217

10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
N/A	N/A	N/A	N/A

11. TITLE (Include Security Classification): Area-time Lower-Bound Techniques with Application to Sorting

12. PERSONAL AUTHOR(S): Bilardi, G., Preparata, F.P.

13a. TYPE OF REPORT: technical

13b. TIME COVERED: FROM ________ TO ________

14. DATE OF REPORT (Yr., Mo., Day): April 25, 1985

15. PAGE COUNT: 38

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)

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

The area-time complexity of VLSI computations is constrained by the flow and the storage of information in the two-dimensional chip. We study here the information exchanged across the boundary of the cells of a square-tessellation of the layout. When the information exchange is due to the functional dependence between variables respectively input and output on opposite sides of a cell boundary, lower bounds are obtained on the AT^2 measure (which subsume bisection bounds as a special case). When information exchange is due to the storage saturation of the tessellation cells, a new type of lower bound is obtained on the AT measure.

In the above arguments, information is essentially viewed as a fluid whose flow is uniquely constrained by the available bandwidth. However, in some computations, the flow is kept below capacity by the necessity to transform information before an output is produced. We call this mechanism computational friction and show that it implies lower bounds on the AT/log A measure.

Regimes corresponding to each of the three mechanisms described above can appear by varying the problem parameters as we shall illustrate by analyzing the problem of sorting n

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐

21. ABSTRACT SECURITY CLASSIFICATION:

22a. NAME OF RESPONSIBLE INDIVIDUAL:

22b. TELEPHONE NUMBER (Include Area Code):

22c. OFFICE SYMBOL:

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

SECURITY CLASSIFICATION OF THIS PAGE

keys each of k bits, for which AT^2, AT, and At/log A bounds are derived. Each bound is interesting, since it dominates the other two in a suitable range of key lengths and computation times.

Accession For

NTIS GRA&I [X]
DTIC TAB [ ]
Unannounced [ ]
Justification

By
Distribution/

Availability Codes

Dist	Avail and/or Special
A-1	

QUALITY INSPECTED 3

AREA-TIME LOWER-BOUND TECHNIQUES WITH APPLICATION TO SORTING

G. Bilardi+ and F. P. Preparata++

ABSTRACT

The area-time complexity of VLSI computations is constrained by the flow and the storage of information in the two-dimensional chip. We study here the information exchanged across the boundary of the cells of a square-tessellation of the layout. When the information exchange is due to the functional dependence between variables respectively input and output on opposite sides of a cell boundary, lower bounds are obtained on the AT^2 measure (which subsume bisection bounds as a special case). When information exchange is due to the storage saturation of the tessellation cells, a new type of lower bound is obtained on the AT measure.

In the above arguments, information is essentially viewed as a fluid whose flow is uniquely constrained by the available bandwidth. However, in some computations, the flow is kept below capacity by the necessity to transform information before an output is produced. We call this mechanism computational friction and show that it implies lower bounds on the AT/log A measure.

Regimes corresponding to each of the three mechanisms described above can appear by varying the problem parameters, as we shall illustrate by analyzing the problem of sorting n keys each of k bits, for which AT^2, AT, and AT/log A bounds are derived. Each bound is interesting, since it dominates the other two in a suitable range of key lengths and computations times.

+Department of Computer Science, Cornell University, Ithaca, NY 14853

++Coordinated Science Laboratory, Departments of Electrical Engineering and Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

This work was supported in part by the National Science Foundation Grant ECS-84-10902, by an IBM predoctoral fellowship, and by the Joint Services Electronics Program under Contract N00014-84-C-0149.

1. Introduction

Considerable attention has been devoted in recent years to the establishment of lower bounds to the principal measures of performance of VLSI circuits, that is, chip area A and computation time T. Typically, these lower bounds are in the form of area-time tradeoffs and are based on minimum requirements on the amount of information that must cross suitably chosen sections of the circuit chip.

Lower bounds to performance measures are valid within a well-defined computation model. In keeping with common practice, in this paper we adopt the so-called synchronous VLSI model.

A computational problem Π is a boolean mapping from a set $\mathcal{J}$ to a set $\mathcal{O}$ of input and output variables, respectively. The mapping embodied by Π is realized by a boolean machine described as a computation graph, $G = (V,E)$, whose nodes V are information processing devices or input/output ports and whose arcs E are wires.

A VLSI chip is a two-dimensional embedding of this computation graph, according to the prescriptions of the model. The model is characterized by a collection of rules concerning layout, timing, and input/output (I/O) protocol; in addition, the model restricts the class of computation graphs to those having bounded fan-in and fan-out (without loss of generality, this bound is assumed to be two).

The layout rules are:

1. Wires (arcs) have minimum width λ and at most ν wires ($\nu \geq 2$) can overlap at any point;
2. Nodes have minimum area $c\lambda^2$, for some $c \geq 1$.

No loss of generality is incurred if the layout is restricted to be an embedding of the computation graph in a uniform grid, typically the square grid: the latter is the plane grid the vertices of which have integer coordinates (layout grid). The layout rules may contain the additional specification that all I/O ports be placed on the boundary of the layout. Chips obeying this constraint are referred to as boundary chips; unless otherwise noted, we shall consider unrestricted chips, where I/O ports can occur anywhere in the layouts.

The timing rules specify that a bit requires a fixed time τ_0 (hereafter, assumed equal to 1) to propagate along a wire, irrespective of its length (synchronous system). Finally, the I/O protocol is semellective (each input is received exactly once), unilocal (each input is received at exactly one input port), time- and place-determinate (each I/O variable is available at prespecified port and time, for all instances of the problem).

For a given problem Π, a tradeoff between the chip area A and the computation time T is conveniently expressed by the family of functions

$$A = \alpha_n(T), \quad T \in [T_{min}(n), T_{max}(n)]$$

where n is the input size, $\alpha_n(T)$ is the area of the smallest design that solves Π in time T, T_{min} is the minimum time required to solve Π (regardless of the area), and T_{max} is a time such that, for $T > T_{max}, \alpha_n(T)$ is constant with respect to T. Equivalently, a tradeoff is expressed as a relation $g(A,T) = \Omega(f(n))$, for a suitable function g.

To date, almost all known area-time lower bounds belong to one of the three following classes:

(1) Input-output flow bounds. They are of the form $AT = \Omega(|\mathcal{I}| + |\mathcal{O}|)$,

and follow directly from the fact that the maximum number of bits that can be exchanged with the exterior in a time unit is proportional to the number of I/O ports, which in turn is at most proportional to the chip area.

(2) Internal-flow bounds. They are typically of the form $AT^2 = \Omega(I^2)$ where I is the bisection-information of Π, i.e. the minimum amount of information that must be exchanged across any section that equipartitions the set V [1]. When I is known, the bound follows from the fact that the area is at least proportional to the square of the minimum bisection width b, and that the number of bits that can be exchanged across this bisection in a time unit is at most proportional to b.

(3) Node-degree bounds. To our knowledge, this type of bound has been developed only in connection with integer addition [2,3] and can be stated in the form $AT/\log A = \Omega(|\mathcal{J}|)$. Such bound hinges on the fact that, since the computation time of output functions depends on the number of their arguments, information must reside within the chip for a certain duration.

In this paper, with the intent to take a further step toward the development of a coherent theory of VLSI complexity, we develop a finer analysis of internal flow by considering subdivisions of the chip, which are more demanding on the information flow than balanced bisection.

In Section 2, we introduce a novel technique, called "square tessellation", which is based on a partition of the chip into square cells of equal size, and on the information exchanged across the boundary of these cells.

When the mechanism forcing the information exchange is the functional dependence between variables that are input inside and output outside a tessellation cell (or vice versa), the tessellation technique leads to bounds on the AT^2 measure (of which bisection bounds are a special case).

We also identify a new mechanism of exchange, due to storage saturation of tessellation cells, which leads to bounds on the AT measure. Indeed, storage limitations may affect the information exchange in several ways. For example, during the computation, a cell may fill its storage (a situation referred to as "saturation") and hence be forced to send some information outside, just for temporary storage, and to receive it back at a later time. In other situations, some information input outside of a cell is needed by the cell at several different times. If the amount of this information exceeds the storage capacity of the cell, the information must be transmitted through the boundary each time.

It must be pointed out that, although both the input/output flow and the internal flow due to saturation lead to AT bounds, the phenomena involved are completely different. Indeed, input-output flow occurs through a two-dimensional section and has the physical dimensions of bits/(length)^2, while internal flow occurs through a one-dimensional section and is expressed in bits/length.

When considering either I/O or internal flow, information is essentially viewed as a fluid whose flow is uniquely constrained by the available capacity (bandwidth). However, in some computations, the I/O flow is kept below capacity by the fact that information has to be transformed before being output, and that this transformation cannot be instantaneous due to the bounded fan-in and fan-out of the gates. In the context of the fluid-dynamic analogy of VLSI computation, it seems appropriate to call this mechanism "computational friction." A general framework for this phenomenon is also developed in Section 2.

In Section 3 we illustrate the techniques introduced in Section 2 by deriving new AT^2, AT and $AT/\log A$ bounds for the problem of sorting n k-bit keys. Each bound is interesting since it dominates the other two in a suitable range of key lengths and computation times. In Section 4, we briefly mention analogous results in relation to the problem of cyclic shift, merging, and record sorting.

After completion of the work reported in this paper, we have learned that some AT^2 bounds based on the notion of tessellation had been previously derived in unpublished work by Angluin [4], and that Siegel [12] reports the same AT^2 tessellation bound as our Theorem 7.

We also mention that partitions of the layout into multiple regions (different from square tessellation) have been used in [5] to study the area-time complexity of multilective protocols.

2. Square Tessellations and Lower Bounds on Performance Measures

The general background of all considerations developed in this section is a partition of the laid-out computation graph into two portions, which, when appropriate, will be identified with two processors P_1 and P_2 cooperating to solve the given problem Π.

The partition may refer either to topological properties of the graph (sets of vertices and edges), or to its computational properties (set of I/O variables). The two cases are referred to as "dichotomy" and "I/O assignment" respectively, as expressed by the following definitions:

Definition 1. If $\mathcal{V} = \mathcal{I} \cup \mathcal{O}$ is the set of I/O variables of Π, an I/O assignment for Π is a partition $(\mathcal{V}_1, \mathcal{V}_2)$ of $\mathcal{V}$.

Definition 2. Given a graph $G = (V,E)$, a dichotomy of G is a partition (V_1, V_2) of V; $\delta(V_1, V_2)$, called dichotomy width, is the number of edges connecting V_1 to V_2.

The square tessellation, to be introduced next, is a device that we use to generate partitions. Let the auxiliary grid be the translation of the layout grid by the vector $(\frac{1}{2}, \frac{1}{2})$.

Definition 3. A square tessellation with sidelength ℓ is a subgraph of the auxiliary grid formed by two sheaves $\{x = \frac{1}{2} + j\ell : j \geq 0\}$ and $\{y = \frac{1}{2} + j\ell : j \geq 0\}$ of evenly spaced rows and columns with spacing ℓ.

A square tessellation is a partition of the plane into identical $\ell \times \ell$ tiles called cells, i.e. it is a geometric partition. It can be used to produce a dichotomy (and the associated I/O assignment) by identifying an individual cell with one term of the dichotomy, and the rest of the layout with the other term. The outstanding feature of the tessellation technique is that it permits accounting for the simultaneous presence of all other cells.

We begin by obtaining a lower bound to the area A of the layout in terms of (topological) properties of the graph.

2.1. Lower bounds on the area of the chip

Given a graph $G = (V,E)$, and an integer $0 < m < |V|$, consider the set $\Gamma_m \triangleq \{(V_1,V_2): |V_1| = m\}$ and let

$$\delta(m) = \min_{(V_1,V_2)\in\Gamma_m} \delta(V_1,V_2), \tag{1}$$

i.e., $\delta(m)$ is the smallest dichotomy width over all dichotomies of V with fixed ratio $|V_1|/|V_2|$. Thus, $\delta(\lfloor|V|/2\rfloor)$ is Thompson's minimum bisection width [1]. We can now state the following theorem:

Theorem 1. For every graph $G = (V,E)$ and every $m < |V|$

$$A = \Omega(|V|\frac{\delta^2(m)}{m}) \tag{2}$$

Proof: We position the layout rectangle R with its southwest corner at $(\frac{1}{2},\frac{1}{2})$ and partition it by means of a square tessellation, with spacing $\ell = \lfloor(\delta(m) - 1)/4\rfloor$. Both sides of R have length at least $\delta(m)-1$. In fact, we can cut R by means of a vertical two-bend polygonal line on the auxiliary grid (zig-zag line), into two polygons one of which contains exactly m vertices. Thus, at least $\delta(m)$ edges cross the cut, $\delta(m)-1$ of which are horizontal, and the vertical side has length at least $\delta(m)-1$. A similar argument applies to the horizontal side. Then R contains at least 16 cells of the tessellation and the smallest tessellation rectangle R' containing R has area at most 25/16 times as large as the area of R. We now claim that each cell of R' contains fewer than m vertices of G. Indeed, if a cell contains m or more vertices we cut it, by means of a zig-zag line, into two polygons one of which contains exactly m vertices; this polygon has perimeter $p \leq 4\ell = 4\lfloor(\delta(m) - 1)/4\rfloor \leq \delta(m) - 1$, so that fewer than $\delta(m)$ edges cross the cell boundary contrary to the definition of $\delta(m)$. It follows that at least $\lceil|V|/m\rceil$ cells of R' contain some node of G, and therefore overlap with R. The area of R' is at least as large as the global area $\lceil|V|/m\rceil\ (\lfloor(\delta(m)-1)4\rfloor)^2$ of these cells, so that A is at least 16/25 of it. $\square$

The best lower bound to the area of a layout of G is obtained for the value m_0 of m that maximizes the ratio $\delta^2(m)/m$. For most of the computation graphs considered in the literature $m_0 \simeq |V|/2$, yielding $A = \Omega(\delta^2(m))$. This accounts for the success of bisection techniques. We shall see later that for the computation graph of some significant problems - such as sorting - $\delta^2(m)/m$ achieves its maximum for values of m considerably smaller than $|V|$.

Since the graphs to be considered in this paper are computation graphs, it is convenient to establish a link between the notions of dichotomy and of I/O assignment. Specifically, in a computation graph G a subset U of V is the set of I/O ports, and $v \in U$ handles $\varphi(v)$ variables during the computation; in other words, an integer-valued function φ on V adequately identifies the I/O ports, each with its multiplicity. The fact that, for some $v \in U$ we have $\varphi(v) > 1$, introduces a "coarser granularity" in the partition of the I/O variables, which must be dealt with.

To achieve a possibly useful broader generality, for G = (V,E) we consider arbitrary weighting functions $\varphi: V \to \mathbb{N}$ (the nonnegative integers), and denote by $\varphi_{max} = \max\{\varphi(v): v \in V\}$ and by $M = \sum_{v \in V} \varphi(v)$, the global weight. For an integer $m < M$, define the following class of dichotomies:

$$\mathcal{D} = \{(V_1, V_2): m - \varphi_{max} + 1 \leq \sum_{v \in V_1} \varphi(v) \leq m\} \qquad (3)$$

Intuitively, a weighting function models a general distribution of input/output variables, and the generalized definition (3) of class of dichotomies copes with the input granularity, as expressed by φ_{max}. By a straightforward modification of the proof of Theorem 1, we can then establish:

<u>Theorem 2</u>. For a graph $G = (V,E)$, let φ be a weighting function φ on V with parameters M and φ_{max}, and let Γ satisfy (3). Define

$$\delta_\Gamma = \min_{(V_1,V_2)\in\Gamma} \delta(V_1,V_2).$$

Then

$$A = \Omega\left(M\frac{\delta_\Gamma^2}{m}\right) . \qquad (4)$$

Note that when $\varphi(v) = \varphi_{max} = 1$ for every $v \in V$, $M = |V|$, Γ becomes Γ_m, and (4) subsumes (2). It must be stressed that the notion of dichotomy pertains to a <u>given</u> graph; we shall now see how relation (4) will be instrumental to obtain area-time lower bounds on <u>all</u> graphs solving a given Π.

2.2. <u>Area-time lower bounds based on information exchange (AT^2-theory)</u>

In this section we shall consider a graph $G = (V,E)$ as a computation graph solving problem Π in time T, and we shall establish a lower bound to its layout area as a function of T and of parameters of Π.

The starting point of the argument is a dichotomy (V_1,V_2) of V. This dichotomy naturally identifies two processors P_1 and P_2, where P_i is defined as the subgraph induced by vertex set V_i ($i = 1,2$). In turn, if we define $\mathcal{V}_i$ as the I/O variables of Π handled by processor P_i ($i = 1,2$), then we obtain an I/O assignment $(\mathcal{V}_1,\mathcal{V}_2)$. Such assignment is characterized by a very important parameter, called "information exchange", as expressed by the following definition (similar definitions have been considered by many authors, e.g. [1,6,7,8]):

Definition 4. The information exchange of Π under assignment $(\mathcal{V}_1, \mathcal{V}_2)$ is defined as

$I(\mathcal{V}_1, \mathcal{V}_2)$ = the minimum over all the algorithms (which solve Π under assignment $(\mathcal{V}_1, \mathcal{V}_2)$) of the maximum over all the problem instances of the number of bits exchanged between P_1 and P_2.

In other words, for any algorithm that solves Π under $(\mathcal{V}_1, \mathcal{V}_2)$ there is at least a problem instance for which P_1 and P_2 exchange $I(\mathcal{V}_1, \mathcal{V}_2)$ or more bits, and no integer larger than $I(\mathcal{V}_1, \mathcal{V}_2)$ enjoys the same property.

In the following, $(\mathcal{V}_1, \mathcal{V}_2)$ can be an arbitrary member of a class H of assignments, so that we need to lower-bound the information exchange for all members of this class. The argument is completed by considering a class of dichotomies of an arbitrary, but fixed, graph G solving Π in time T, and the corresponding class H of I/O assignments, and by bounding the dichotomy width of G in terms of the information exchange of H and the time T.

More formally, for a class H of I/O assignments for Π, we define

$$I_H = \min_{(\mathcal{V}_1, \mathcal{V}_2) \in H} I(\mathcal{V}_1, \mathcal{V}_2) \tag{5}$$

We now relate δ_Γ of a given class Γ of dichotomies of G with the I_H of the corresponding class H of assignments:

Lemma 1. Let Γ be a class of dichotomies of a graph G, which solves Π in time T. Let $H = \{(\mathcal{V}_1, \mathcal{V}_2) : (\mathcal{V}_1, \mathcal{V}_2)$ corresponds to $(V_1, V_2) \in \Gamma\}$. Then:

$$\delta_\Gamma \geq \frac{I_H}{T} . \tag{6}$$

Proof. Let $(V_1, V_2) \in \Gamma$ be a dichotomy of V and $(\mathcal{V}_1, \mathcal{V}_2)$ the corresponding assignment. Then V_1 must be able to exchange $I(\mathcal{V}_1, \mathcal{V}_2) \geq I_H$ bits with V_2 in time T and therefore must be connected to V_2 by at least I_H/T edges. $\square$

Let $\mathcal{U} \subseteq \mathcal{V}$ be a set of I/O variables of Π and $\varphi(v)$ the number of variables of $\mathcal{U}$ handled by node v. As the class Γ we consider the one defined by relation (3), for $m \leq |\mathcal{U}|$ and $\varphi_{max} \leq T$, and obtain the following result:

Theorem 3. Let Γ be a class of dichotomies of G (which solves Π in time T) satisfying (3) for $m \leq |\mathcal{U}|$ and $\varphi_{max} \leq T$. Letting $H = \{(\mathcal{V}_1,\mathcal{V}_2):(\mathcal{V}_1,\mathcal{V}_2)$ correspond to $(V_1,V_2) \in \Gamma\}$, we have

$$AT^2 = \Omega\left(|\mathcal{U}| \frac{I_H^2}{m}\right) \quad . \tag{7}$$

Proof: Immediate, by combining Theorem 2 and Lemma 1. □

Theorem 3 is the cornerstone of the so-called AT^2-theory of VLSI computation and has far-reaching consequences. The reasons are that for most computational problems we are able:

(i) to characterize the class H corresponding to Γ that satisfies (3);

(ii) to compute or bound I_H.

To make the tradeoff more explicit, we note that, for given parameters φ_{max} and m of Γ, H is the class of I/O assignments for which

$$m - \varphi_{max} + 1 \leq |\mathcal{V}_1 \cap \mathcal{U}| \leq m \tag{8}$$

Moreover, if H_j is the subclass of H for which $|\mathcal{V}_i \cap \mathcal{U}| = j$, then $\{H_{m-\varphi_{max}+1},\ldots,H_m\}$ is a partition of H. Therefore, denoting $I(j) \triangleq I_{H_j}$, by the definition (8) of I_H, we have

$$I_H = \min\{I(m-\varphi_{max}+1),\ldots,I(m)\}.$$

Therefore we can work with I(j) to obtain a lower bound to I_H. Notice at first that $|I(j) - I(j-1)| \leq 1$, since by just sending one bit from P_1 to P_2 we can transform an assignment in H_j to one in H_{j-1}, (or vice versa). Thus, by using I(0) = 0 and the above inequality we obtain

$$I(m) \leq m \;, \tag{9}$$

$$I(m) - I_H \leq \varphi_{max} - 1 \;. \tag{10}$$

This inequality is used crucially in the following theorem.

<u>Theorem 4</u>. Let G be a computation graph solving Π in time T. Then

$$AT^2 = \Omega\left(|\mathcal{U}| \; \frac{I^2(m)}{m}\right)$$

<u>Proof</u>. Let $\varphi_{max} = \max_{v \in V} \varphi(v)$. Let Γ be the class of dichotomies of G satisfying (3) and H its corresponding class of I/O assignments. Clearly, $\varphi_{max} \leq T$, since each port reads or writes at most one bit per unit of time. Relation (10) becomes $I(m) - I_H \leq T - 1$, or equivalently $I_H \geq I(m) - T + 1$. Therefore relation (7) can be rewritten as

$$AT^2 \geq \lambda_1 \;\; |\mathcal{U}| \; \frac{(I(m)-T+1)^2}{m} \tag{11}$$

for some constant λ_1. This bound may become weak for large values of T; however, for large T, we expect AT to remain large. Indeed, we have the interplay of two contrasting phenomena, an I/O bound (prevailing for large T) and an information-exchange bound (prevailing for small T)(*). Specifically, the I/O bound $AT \geq |\mathcal{U}|$ yields

$$AT^2 \geq |\mathcal{U}| T; \tag{12}$$

Combining relations (11) and (12) we obtain

$$AT^2 \geq |\mathcal{U}| \max\{\lambda_1 \; \frac{(I(m)-T+1)^2}{m} \;, T\} \;.$$

(*) Lower-bound arguments of analogous flavor are due to Savage [9] and Brent-Kung [10].

If we choose for T the value $T_0 = I^2(m)/(2I(m)+(m/\lambda_1))$, then we have

1. $T \leq T_0$. Trivially, $I(m) - T + 1 > I(m) - T$. Since $(I(m)-T)^2$ is a decreasing function of T, it achieves its minimum at T_0, whence:
$(I(m)-T+1)^2 > (I(m)-I^2(m)/(2I(m)+(m/\lambda_1)))^2 > I^2(m)[(I+(m/\lambda_1))/(2I+2(m/\lambda_1))] = I^2(m)/$

2. $T \geq T_0$. By (9), $I(m) \leq m$, whence $T \geq T_0 \geq I^2(m)/(2m+m/\lambda_1) \geq I^2(m)/4m$ taking $\lambda_1 \leq 1/2$.

This completes the proof of the theorem. □

2.3. Saturation area-time lower bounds (AT-theory)

When we ideally isolate a region of the layout of a VLSI system, not only is the bandwidth between this region and the remaining part of the layout bounded by the perimeter of the region, but also the amount of information that can be stored within the region is bounded by its area. This fact has important consequences for the area-time performance of some computations. In this section we develop techniques to express these effects in a quantitative manner.

In the familiar framework where processors P_1 and P_2 cooperate to solve problem Π, we now assume that only a limited amount s_i of storage is available in P_i ($i = 1,2$), and refine Definition 4 as follows:

Definition 5. The information exchange of Π under assignment $(\mathcal{V}_1,\mathcal{V}_2)$ and the condition that P_i can store at most s_i bits ($i = 1,2$) is defined as:

$I(\mathcal{V}_1,\mathcal{V}_2 \mid s_1,s_2) \triangleq$ the minimum over all algorithms (which solve Π under assignment $(\mathcal{V}_1,\mathcal{V}_2)$ and storage bounds s_1 and s_2) of the maximum over all the problem instances of the number of bits exchanged between P_1 and P_2

In analogy with previous definitions, for a class H of assignments for Π we have

$$I_H(s_1,s_2) = \min_{(\mathcal{V}_1,\mathcal{V}_2)\in H} I(\mathcal{V}_1,\mathcal{V}_2 \mid s_1,s_2) \tag{13}$$

and let $I(m|s_1,s_2) = I_{H_m}(s_1,s_2)$. Note that $I((\mathcal{V}_1,\mathcal{V}_2)|s_1,s_2)$ is nonincreasing function of s_1 and s_2 (more storage never hurts), whence $I(\mathcal{V}_1,\mathcal{V}_2) = I(\mathcal{V}_1,\mathcal{V}_2 \mid \infty,\infty) \leq I(\mathcal{V}_1,\mathcal{V}_2 \mid s_1,s_2)$.

We now identify with processor P_1 a cell of a square tessellation of the layout with sidelength ℓ, and let m be the number of variables handled by P_1. Clearly the storage of P_1 is upper-bounded by ℓ^2, and we have $I(m|s_1,s_2) = I(m|\ell^2,A-\ell^2) \geq I(m|\ell^2,+\infty)$, where A is the layout area. Since the cell's perimeter is 4ℓ, we conclude

$$T \geq \frac{I(m|\ell^2,+\infty)}{4\ell} \quad . \tag{14}$$

If $|\mathcal{V}|$ variables are input/output in area A, for any $\mathcal{U}\subseteq\mathcal{V}$ there is at least one cell of the tessellation that handles at least $|\mathcal{U}|\ell^2/A^2$ variables of $\mathcal{U}$. However, due to the granularity of the input/output there may be no cell handling exactly $|\mathcal{U}|\ell^2/A^2$ variables; on the other hand, if cell C handles more than $|\mathcal{U}|\ell^2/A^2$ variables, since $\varphi_{max} \leq T$, a zig-zag cut will isolate a portion of C handling $(|\mathcal{U}|\ell^2/A^2 + h)$ variables for some h satisfying $0 \leq h < T$. Thus we can write

$$T \geq I\left(\frac{|\mathcal{U}|\ell^2}{A} + h \;\middle|\; \ell^2,\infty\right)\Big/4\ell \text{ for some } h \in [0,T-1].$$

This discussion proves the following theorem.

<u>Theorem 5</u>. Let G be a computation graph for problem Π. Let $\mathcal{U}$ be a set of I/O (binary) variables of Π. If H_m is the class of assignments such that exactly m variables of $\mathcal{U}$ are assigned to P_1, and $I(m|s,\infty)$ is the information exchange of H_m when P_1 has s bits of storage, then the area-time performance of any layout of G satisfies the bound for arbitrary ℓ:

$$T \geq \min_{0 \leq h \leq T} I\left(|\mathcal{U}|\frac{\ell^2}{A} + h \,|\, \ell^2, \infty\right) \Big/ 4\ell. \tag{15}$$

<u>Remark 1</u>. To obtain the best bound we choose the value of ℓ that maximizes the right-hand side of (15).

<u>Remark 2</u>. If $I(m|s,\infty)$ is increasing with m, then $T \geq I(|\mathcal{U}|\frac{\ell^2}{A}|\ell^2,\infty)/4\ell$.

<u>Remark 3</u>. If for the value ℓ_0 of ℓ that maximizes (15) we have $I(m|\ell_0,\infty) = \beta m$ for some constant β (as is found in all applications considered so far), then we can rewrite bound (15) as

$$T \geq \beta|\mathcal{U}| \frac{\ell_0^2}{A \cdot 4\ell_0} ,$$

or equivalently

$$AT \geq \Omega(|\mathcal{U}|\ell_0). \tag{16}$$

Usually ℓ_0 is an increasing function of $|\mathcal{U}|$ so that (16) is a stronger bound than the straightforward I/O bound $AT = \Omega(|\mathcal{U}|)$.

2.4. Area-time lower bound due to computational friction (AT/logA-theory)

In all of the previous considerations we have viewed the computational process inside a VLSI chip as a fluido-dynamic phenomenon, so that the performance bounds are determined by the ability to ensure certain flows within the prescribed time. Note that this viewpoint is completely oblivious of the fact that the vertices of G have bounded fan-in and fan-out; however, computation delays are exactly due to these limitations, which we now wish to bring into the picture.

We begin by considering the notion of "functional dependence".

Definition 6. Given a function $y = f(x)$, where $\underline{x} = (x_1,\ldots,x_p)$ and $\underline{y} = (y_1,\ldots,y_q)$ are boolean vectors, we say that y_j is functionally dependent on x_i if there exist two boolean vectors $\underline{x}'$ and $\underline{x}''$ that differ only in the i-th component, such that $\underline{y}' = f(\underline{x}')$, and $\underline{y}'' = f(\underline{x}'')$ differ in the j-th component.

We now explicitly consider that all gates of our circuits have fan-in bounded by a constant f_I; if output variable y is functionally dependent on s input variables, then at least $\log_{f_I} s$ time units must elapse between the instant when the first of the input variables is read, and the instant when y is output. Hereafter we assume $f_I = 2$.

Informally, suppose that s variables $x_1,\ldots,x_s$ are input at the same time, and that there exist r output variables $y_1,\ldots,y_r$ with the following properties:

(1) each y_j is functionally dependent on all x_i's;

(2) the variables $y_1,\ldots,y_r$ carry I bits of information on $x_1,\ldots,x_s$.

Then since no y_j emerges before logs units of time, the system must be capable of storing I bits for logs units of time, or $AT \geq I \log s$. In other words, in the analogy where information is a fluid flowing from input ports to output ports, we can say that functional dependence acts as a kind of computational friction, that slows down the flow keeping it below capacity. This intuitive notion can be formalized in the following theorem:

Theorem 6. Given a computational problem Π with a set $\mathcal{I}$ of input variables and a set $\mathcal{O}$ of output variables, let $\mathcal{U}$ be a subset of $\mathcal{I}$ such that for any partition $\mathcal{U}_1,\ldots,\mathcal{U}_T$ of $\mathcal{U}$ there exists a collection $\mathcal{V}_1,\ldots,\mathcal{V}_T$ of disjoint subsets of $\mathcal{O}$ (not necessarily a partition) satisfying the following properties:

(1) Each variable in $\mathcal{V}_t$ is functionally dependent upon at least $\alpha(|\mathcal{U}_t|)$ variables of $\mathcal{U}_t$, where $\alpha(\ )$ is an increasing function of its argument.

(2) The variables in $\mathcal{I} - \mathcal{U}$ can be selected so that, for each $t = 1,2,\ldots,T$ the variables of $\mathcal{V}_t$ carry at least $\beta(|\mathcal{U}_t|)$ bits of information relative to $\mathcal{U}_t$, where $\beta(\ )$ is an increasing function of its argument.

Then, for any VLSI system that solves Π,

$$AT \geq \min_{s_1+\ldots+s_T=|\mathcal{U}|} \sum_{t=1}^{T} \beta(s_t)\log\alpha(s_t) \ . \tag{17}$$

Proof. Let us define $\mathcal{U}_t$ as the subset of variables of $\mathcal{U}$ that are input by the system (exactly) at time t. Clearly, $\mathcal{U}_1,\ldots,\mathcal{U}_T$ partition $\mathcal{U}$. Let $\mathcal{V}_1,\ldots\mathcal{V}_T$ be the corresponding collection of disjoint subsets of $\mathcal{O}$. Because of property (1), and of the bounded-fan-in assumption, no variable in $\mathcal{V}_t$ can be output before time $t + \log\alpha(|\mathcal{U}_t|)$. Because of property (2), at least $\beta(\mathcal{U}_t)$ bits of information on $\mathcal{U}_t$ have to be stored by the system in the interval $[t,t + \log\alpha(|\mathcal{U}_t|)]$. Since one bit stored per τ units of time contributes τ to the AT product, we conclude that

$$AT \geq \sum_{t=1}^{T} \beta(|\mathcal{U}_t|)\log\alpha(|\mathcal{U}_t|) \tag{18}$$

where $\sum_{t=1}^{T} |\mathcal{U}_t| = T$. The partition $\mathcal{U}_1,\ldots,\mathcal{U}_T$, and hence the right-hand side of (2) is a function of the input schedule and varies from system to system, but it is never smaller than the right-hand side of (1). □

<u>Corollary 1</u>. Under the same assumptions of Theorem 1, if $\beta(s)\log\alpha(s)$ is a downward-convex function of s, then the minimum in (1) is achieved when $s_1 = s_2 = \ldots = s_T = |\mathcal{U}|/T$, and (1) yields,

$$A \geq \beta(|\mathcal{U}|/T)\log\alpha(|\mathcal{U}|/T). \tag{19}$$

<u>Corollary 2</u>. If $\beta(s) = \beta_0 s$, and $\alpha(s) = \alpha_0 s$, then (3) can be rewritten as

$$A = \Omega(|\mathcal{U}|/T\log(|\mathcal{U}|/T)), \tag{20}$$

or

$$AT/\log A = \Omega(|\mathcal{U}|). \tag{21}$$

Theorem 6 and its Corollaries 1 and 2 apply to chips with bounded fan-in. Similar arguments yield analogous results for chips with bounded fan-out. In this case, the friction arises from the fact that, if s variables $y_1,\ldots,y_s$ are output at time t, and are all functionally dependent upon input variable x, then x must be input prior to time $t - \log s$.

3. Sorting

In this section we shall illustrate the lower-bound techniques introduced in Section 2 by applying them to the sorting problem, which we define formally as follows.

Definition 7. In the (n,k)-sorting problem.

(1) The input is a sequence of n k-bit keys, each a member of a finite set of integers.

(2) The output is a rearrangement of the input keys, so that they form a nondecreasing sequence.

Notationally, input and output are represented as $n \times k$ binary arrays $X = \{X_i^j\}$ and $Y = \{Y_i^j\}$, respectively, with $i = 0, \ldots, n-1$, and $j = k-1, \ldots, 0$; X_i is the (i+1)-st input key and X^j is the bit position of weight 2^j (or briefly, bit position j). We can view the input of (n,k)-sorting as an (n,k)-multiset, i.e. a multiset of n elements each drawn from a universe of 2^k values. As we will show, the nature of (n,k)-sorting changes considerably with the relative size of n and k, and we find it useful to classify (n,k)-multisets in multisets of short keys ($1 \le k \le \log n$), of intermediate-length keys ($\log n < k < 2\log n$), and of long keys ($2\log n \le k$). With a slight abuse of terminology, we shall use phrases like "sorting short keys" instead of "sorting a multiset of short keys."

To gain intuition on the transfer of information from input to output, we observe that

$$Y_i^j = X_{\pi(i)}^j \qquad i = 0,1,\ldots,n-1, \quad j = 0,1,\ldots,k-1,$$

where $\pi(0),\pi(1),\ldots,\pi(n-1)$ is a permutation of $0,1,\ldots,n-1$. Thus, there is an information flow from the input to the output ports of the same position, which we call primary flow. In the primary flow, each bit enters and leaves the system maintaining its identity. However, the exact destination of each bit within its own position depends on π, which, for position j, is determined by the values of the data in positions j,j+1,...,k-1. This information, flowing from most significant to least significant positions, is called secondary flow. Primary flow considerations have been used by Thompson [1] in proving the $AT^2 = \Omega(n^2\log^2 n)$ bound for word-local $(n,\log n+\theta(\log n))$-sorting. The bound has been later generalized to non-word-local protocols by Leighton [11] who succeeded in combining primary and secondary flows with the help of cyclic-shift arguments.

In Section 3.1 and 3.2, we combine a quantitative characterization of primary and secondary flows with the general techniques of Section 2, to obtain novel lower bounds for the problems of sorting short and long keys.

3.1 Short Keys

In this section we study (n,k)-sorting for $k \leq \log n$, and denote by $r \triangleq 2^k$ the size of the universe of possible keys. We shall obtain an AT^2 and an AT bound. Both bounds are based on primary flow, as expressed by the following lemma.

Lemma 2. Chosen r/2 arbitrary input bits

$$\mathcal{U}_{in} = \{X^0_{p_0}, X^0_{p_1}, \ldots, X^0_{p_{r/2-1}}\} \tag{22}$$

and r/2 arbitrary output bits

$$\mathcal{U}_{out} = \{Y^0_{q_0}, Y^0_{q_1}, \ldots, Y^0_{q_{r/2-1}}\} \tag{23}$$

with $q_i < q_{i+1}$, the remaining input bits can be set to constant values to enforce the condition

$$Y^0_{q_i} = X^0_{p_i} \qquad i = 0,1,\ldots,r/2-1 . \tag{24}$$

Proof: We set $X_{p_i} = 2i + X^0_{p_i}$, $i = 0,1,\ldots,r/2-1$ and divide the remaining $n-r/2$ input keys arbitrarily into $r/2+1$ sets such that, for $i = 0,1,\ldots,r/2-1$, the i-th set contains $(q_i-q_{i-1}-1)$ keys whose value is set to $2i(q_{-1} \triangleq 0)$, and the $(r/2)$-th set contains $(n-1-q_{r/2-1})$ keys whose value is set to $r/2-1$. The output sequence corresponding to this input is shown in Figure 1, and it satisfies Equation (24). □

We state now the AT^2 bound for short keys.

Theorem 7. Any VLSI (n,k)-sorter, with $k < \log n$ satisfies the bound

$$AT^2 = \Omega(nr), \tag{25}$$

where $r = 2^k$.

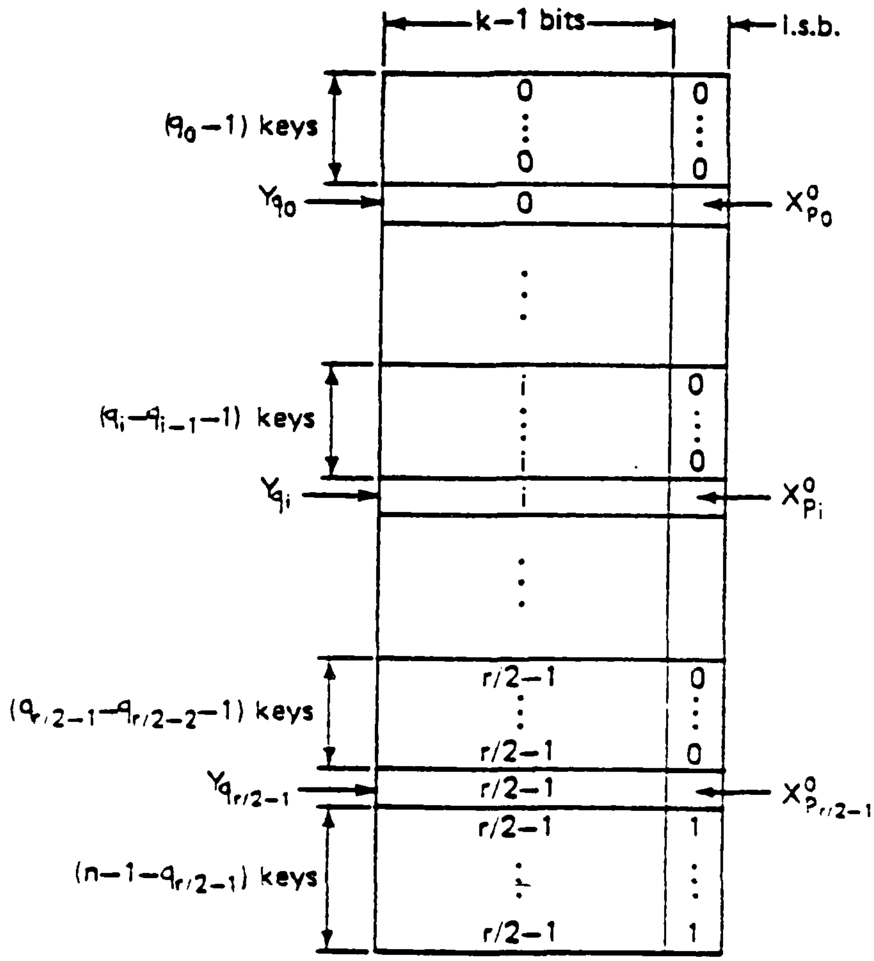


Figure 1. Sending $r/2$ arbitrary bits in X^0 to $r/2$ arbitrary positions in Y^0.

Proof. Result (25) follows from Theorem 4, with $\mathcal{U} = \{X_i^0 : i = 0,1,\ldots,n-1\}$, $m = r/2$, and from the bound $I(r/2) = \Omega(r)$, which we now prove.

Given a protocol that assigns exactly $r/2$ entries of X^0 to P_1 and $n - r/2 (\geq r/2)$ to P_2, let P_s be the processor that outputs more entries of Y^0 (break a tie arbitrarily). From Lemma 2, we can always find two sets $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ as in (22) and (23) such that $\mathcal{U}_{in}$ is input by P_{3-s} and $\mathcal{U}_{out}$ is output by P_s. Equation (24) implies that $r/2$ bits input by P_{3-s} are output by P_s, for suitable values of input variables not in $\mathcal{U}_{in}$. Hence, $I(r/2) \geq r/2 = \Omega(r)$, as desired. □

We shall now prove an AT lower bound based on information exchange under bounded storage (saturation).

Theorem 8. Any VLSI (n,k)-sorter, with $k \leq \log n$, satisfies the bound

$$AT = \Omega(n\sqrt{r}) \tag{26}$$

where $r = 2^k$.

Proof. Referring to Theorem 5, we choose $\mathcal{U} = Y^0$, and, for some real $\sigma \in [0,\frac{1}{2}]$ we consider a square tessellation with sidelength $\sqrt{\sigma r}$. Thus, there exists at least one cell C -- identified with processor P_1 -- that outputs $m \geq n\sigma r/A$ entries of Y^0. To estimate the quantity $I(m|\sigma r,\infty)$, we subdivide the interval $[0,T]$ into consecutive intervals and evaluate the information exchange in each such interval. Specifically, we form the sequence intervals $([t_i+1,t_{i+1}]: 0 \leq i < L,\ t_0 = -1,\ t_L = T)$, so that during $[t_i+1,t_{i+1}]$ C outputs m_i bits of Y^0, with $r/2 \leq m_i < r/2+\sigma r$ (this partition of $[0,T]$ is well-defined, since C outputs at most σr bits per time unit). We now consider $[t_i+1,t_{i+1}]$ individually We apply Lemma 2 by choosing the $r/2$ elements of $\mathcal{U}_{out}$ among the

m_i bits of Y^0 and $\mathcal{U}_{in} \subseteq X^0$ arbitrarily. Since X^0 must be completely input before any bit of Y^0 can be output, during $[t_i+1, t_{i+1}]$ cell C outputs r/2 bits already in the chip at t_i+1. But C stores at most σr bits, whence $(\frac{1}{2}-\sigma)r$ bits must flow across the cell boundary (of length $4\sqrt{\sigma r}$) during the interval. Summing this flow over all L intervals we have

$$I(m|\sigma r,\infty) \geq L(\tfrac{1}{2}-\sigma)r,$$

and, observing that $L \geq m/\max(m_i) \geq (n\sigma r/A)/r(\frac{1}{2}+\sigma)$ we obtain

$$I(m|\sigma r,\infty) \geq n \frac{(\frac{1}{2}-\sigma)}{\frac{1}{2}+\sigma} \frac{\sigma r}{A} \quad .$$

Since $I(m|\sigma r,\infty)$ flows across a section of length $4\sqrt{\sigma r}$ in time T we have

$$AT \geq \frac{\sqrt{\sigma}}{4} \frac{\frac{1}{2}-\sigma}{\frac{1}{2}+\sigma} n\sqrt{r} \ , \tag{27}$$

which completes our proof. (Inequality (27) yields the best bound for $\sigma \simeq 0.117$).

From Theorems 7 and 8 we know that there exist constants β_1 and β_2 such that the performance of any (n,k)-sorter, with $k \leq \log n$, satisfies the bounds $AT \geq \beta_1 n\sqrt{r}$, and $AT^2 \geq \beta_2 nr$. These bounds coincide for $T_0(r) \triangleq (\beta_2/\beta_1)\sqrt{r}$; therefore since $T \geq \log n + r$, only the AT bound is meaningful for values of k satisfying $(\beta_2/\beta_1)2^{k/2} - \log k \leq \log n$, while for values of k satisfying $(\beta_2/\beta_1)2^{k/2} - \log k > \log n$ the AT bound is stronger for $T > T_0$, and the AT^2 bound is stronger for $T < T_0$.

3.2 Long Keys

In this section, we turn our attention to (n,k)-sorting for $k \geq 2\log n$. We begin by deriving two lower bounds on the information exchange $I(\mathcal{V}_1,\mathcal{V}_2)$ for an arbitrary assignment $(\mathcal{V}_1,\mathcal{V}_2)$ of variables to processors P_1 and P_2. The two bounds will be based on primary flow and secondary flow, respectively.

Let $(\mathcal{V}_1,\mathcal{V}_2)$ be an I/O assignment for (n,k)-sorting. For some real $\gamma \in [0,\frac{1}{2}]$, we define a number of quantities, each a function of the parameter γ and of $(\mathcal{V}_1,\mathcal{V}_2)$.

$b_j \triangleq$ number of input words whose j-th bit is input by P_1,

$c_j \triangleq$ number of output words whose j-th bit is output by P_1,

$J_0 \triangleq \{j : j < k-\log n,\ \ \gamma n \leq c_j \leq (1-\gamma)n\}$

$J_1 \triangleq \{j : j < k-\log n,\ \ c_j > (1-\gamma)n\}$

$J_2 \triangleq \{j : j < k-\log n,\ \ c_j < \gamma n\}$

$$B = \sum_{j=0}^{k-\log n-1} b_j, \quad B_s = \sum_{j \in J_s} b_j, \quad s = 0,1,2.$$

Note that B is the number of bits input by P_1 in positions with index $< k-\log n$. In terms of the above quantities, we can state the following results.

<u>Lemma 3 (Primary flow)</u>. With the above definitions, for any I/O assignment $(\mathcal{V}_1,\mathcal{V}_2)$ of (n,k)-sorting, $(k \geq 2\log n)$, and for any $\gamma \in [0,\frac{1}{2}]$, the information exchange satisfies the bound

$$I(\mathcal{V}_1,\mathcal{V}_2) \geq \gamma(B-B_1) \geq \gamma(B-|J_1|n) \quad (28)$$

<u>Proof</u>. By a suitable selection of the logn most significant input positions, we can force the output sequence $Y_0,\ldots,Y_{n-1}$ to be any arbitrary cyclic shift of the input sequence $X_0,\ldots,X_{n-1}$. If we let φ_j be the information exchange pertaining to the position j over all the n different shifts, then

$$\varphi_j = b_j(n-c_j) + (n-b_j)c_j \quad .$$

In fact, each of the b_j bits input by P_1 is output by P_2 $(n-c_j)$ times, and, symmetrically, each of the $(n-b_j)$ bits input by P_2 is output by P_1 c_j times.

By the pigeon-hole principle there is a shift size with information exchange not smaller than the average, which ensures that

$$I(\mathcal{V}_1,\mathcal{V}_2) \geq \frac{1}{n} \sum_{j=0}^{k-\log n-1} \varphi_j \quad . \quad (29)$$

For $j \in J_0 \cup J_2$, we have $n-c_j \geq \gamma n$ and $\varphi_j \geq b_j(n-c_j) \geq b_j\gamma n$. Thus, (29) yields

$$I(\mathcal{V}_1,\mathcal{V}_2) \geq (\frac{1}{n}) \cdot \sum_{j \in J_0 \cup J_2} b_j \gamma n = \gamma(B_0+B_2) = \gamma(B-B_1) \ .$$

The proof is completed by observing that $B_1 = \sum_{j \in J_1} b_j \leq |J_1|n$, trivially. □

Lemma 4 (Secondary flow). For any I/O assignment $(\mathcal{V}_1,\mathcal{V}_2)$ of (n,k)-sorting, $(k \geq 2\log n)$, and for any $\gamma \in [0,\frac{1}{2}]$, the information exchange satisfies the bound

$$I(\mathcal{V}_1,\mathcal{V}_2) \geq (1-\gamma)n \min(|J_1|,|J_2|,\log n) - \tfrac{1}{2}n\log n - n \ . \tag{30}$$

Proof. Assume, without loss of generality, that P_1 reads at most half of the bits in the logn most significant positions of X. We now construct a set J of logn positions, so that the content of $\{X^i : k-\log n \leq i < k\}$ can be recovered from $\{Y^j : j \in J\}$. Specifically, if $|J_1| \geq \log n$, J is just any selection of logn positions of J_1; else, we augment J_1 with $(\log n-|J_1|)$ positions not in J_1 and of index less than k-logn.

We then consider the following class of input instances.

(1) The leading logn bits of X_i represent integer $\pi(i)$, where π is a permutation of 0,...,n-1.

(2) The logn bits of X_i which belong to positions in J represent i.

(3) All remaining bits are zero.

Then, the output array Y has the following structure.

(1) The leading logn bits of Y_i represent integer i.

(2) The logn bits of Y_i which belong to positions in J represent integer $\pi^{-1}(i)$.

(3) All remaining bits are zero.

Thus, π can be recovered from the output positions $\{Y^j : j \in J\}$. Since P_1

outputs at least $\min(|J_1|, \log n)\cdot(1-\gamma)n$ bits of these positions, and it reads at most $\frac{1}{2}n\log n$ bits among the $(n\log n - n + O(\log n))$ bits that specify π, we have that

$$I(\mathcal{V}_1,\mathcal{V}_2) \geq (1-\gamma)n\cdot\min(|J_1|,\log n) - \tfrac{1}{2}n\log n - n .$$

Considering the alternative case (i.e., P_2 reads at most half of the bits in the logn most significant positions) we obtain (30). ☐

3.2.1 <u>AT^2 bound</u>

Lemmas 3 and 4 can now be combined to prove:

<u>Theorem 9</u>. Any VLSI (n,k)-sorter, with $k \geq 2\log n$, satisfies the bound

$$AT^2 = \Omega(k\ n^2 \log n). \tag{31}$$

<u>Proof</u>. With the notation of Theorem 4, we choose $\mathcal{U} = \{X_i^j : 0 \leq i < n, 0 \leq j < k-\log n\}$ and select P_1 so that $m = n\log n$. (Note that B, the number of bits of $\mathcal{U}$ input by P_1, is equal to m since $\mathcal{U}$ is a set of input variables.) With this choice, $|\mathcal{U}| = (k-\log n)n$; assuming, for simplicity, $k \geq 3\log n$, the result follows from Theorem 4, if we can show $I(n\log n) = \Omega(n\log n)$.

Indeed, from Lemma 3 and $B = n\log n$, for any $\gamma \in [0,\frac{1}{2}]$ we have:

$$I(n\log n) \geq \gamma(n\log n - n|J_1|). \tag{32}$$

If we reverse the roles of P_1 and P_2 in Lemma 3, and note that P_2 reads $(k-2\log n)n \geq n\log n$ bits of $\mathcal{U}$, we also obtain

$$I(n\log n) \geq \gamma(n\log n - n|J_2|).$$

If $\min(|J_1|,|J_2|) \geq \log n$, then inequality (30) yields

$$I(n\log n) \geq (1-\gamma)n\log n - \tfrac{1}{2}n\log n - n = \Omega(n\log n)$$

by selecting $\gamma = 0$. Otherwise, (30) becomes

$$I(n\log n) \geq (1-\gamma)n\cdot\min(|J_1|,|J_2|) - \tfrac{1}{2}\, n\log n - n \ . \qquad (33)$$

Assuming, without loss of generality, that $|J_1| \leq |J_2|$, we multiply both sides of (32) by $(1-\gamma)$ and both sides of (33) by γ, and add corresponding sides; we obtain

$$I(n\log n) \geq \gamma(\tfrac{1}{2}-\gamma)n\log n - \gamma n = \Omega(n\log n),$$

by selecting $\gamma = \frac{1}{4}$. □

3.2.2 AT bound

We now turn our attention to saturation flow. Let integer $t(j) \in [0,T]$ be the instant at which the last bit of X^j is input; we then define the following sets:

$$J_1(t) \stackrel{\Delta}{=} \{j : j \in J_1 \text{ and } t(j) = t\}.$$

The sets $J_1(t) : t = 0,\ldots,T-1$ form a partition of J_1 (note that, for some t, $J(t)$ may be empty). Note that no bit belonging to a position in $J_1(t)$ can be output prior to $t+1$. These sets are instrumental in establishing the following result:

Lemma 5 (Saturation flow). Let s be the number of storage bits of P_1. Given an I/O protocol inducing an I/O assignment $(\mathcal{V}_1,\mathcal{V}_2)$, and an interval $[\tau_1,\tau_2] \subseteq [0,T]$, let $Z \subseteq J_1(\tau_1) \cup J_1(\tau_1+1) \cup \ldots \cup J_1(\tau_2)$ be a set of $|Z| \leq \log n$ bit positions. Then, for any $\gamma \in [0,\frac{1}{2}]$, we have

$$I(\mathcal{V}_1,\mathcal{V}_2|s,\infty) \geq \frac{(1-\gamma)}{2}\, n|Z| - s \ . \qquad (34)$$

Moreover, if $[\tau_1^i,\tau_2^i]$ are pairwise disjoint time intervals for $i = 1,2,\ldots,L$, and $Z_i \subseteq J_1(\tau_1^i) \cup \ldots \cup J_1(\tau_2^i)$, with $|Z_i| \leq \log n$, then

$$I(\mathcal{V}_1,\mathcal{V}_2|s,\infty) \geq \frac{(1-\gamma)}{2}\, n \sum_{i=1}^{L} |Z_i| - sL \ . \qquad (35)$$

Proof. Let z be the number of bits belonging to positions in Z that are output by P_1 in $[\tau_1,\tau_2]$. We first consider the primary flow, under saturated conditions. All $|Z|n$ bits belonging to positions in Z have been input by time τ_2, but none is output prior to τ_1. Since $(1-\gamma)|Z|n$ of them are output by P_1 (note that $Z \subseteq J_1$), then $(1-\gamma)|Z|n-z$ are output by P_1 after τ_2. Since P_1 stores at most s of them, $(1-\gamma)|Z|n-z-s$ are stored in P_2 at τ_2 and must be transferred from P_2 to P_1 after τ_2. It follows that

$$I(\mathcal{V}_1,\mathcal{V}_2|s,\infty) \geq (1-\gamma)|Z|n-z-s \quad (36)$$

This primary flow bound is strong for small z; for large z, we have a correspondingly large secondary flow. Indeed, since $|Z| \leq \log n$, we augment Z to Z* by adjoining $(\log n-|Z|)$ arbitrary positions of index $< k-\log n$. Arguing as in Lemma 4, we can show that $n\log n$ bits relative to $\{X^i : i = k-\log n,\ldots,k-1\}$ can be extracted from $\{Y^j : j \in Z^*\}$. But P_1 outputs z of these bits during $[\tau_1,\tau_2]$, and, since at most s are in P_1 at τ_1, z-s must be transferred from P_2 to P_1 in $[\tau_1,\tau_2]$. It follows that

$$I(\mathcal{V}_1,\mathcal{V}_2|s,\infty) \geq z-s. \quad (37)$$

Adding corresponding sides of (36) and (37) and dividing by 2 we obtain (34).

To prove the general result (35) we now claim that, if Z_i and Z_j correspond to nonoverlapping intervals, then their contributions to $I(\mathcal{V}_1,\mathcal{V}_2|s,\infty)$ are independent and can be added.

Indeed, for the primary flows generated by Z_i and Z_j, the claim follows from the fact that, although possibly overlapping in time, these two flows carry independent information on the input values $(Z_i \cap Z_j = \emptyset)$.

For the secondary flows generated by Z_i and Z_j the claim follows from the fact that they occur in different periods of time ($[\tau_1^i,\tau_2^i] \cap [\tau_1^j,\tau_2^j] = \emptyset$). Intuitively, P_1 needs information on the leading input positions $X^{k-1},\ldots,X^{k-\log n}$, at several different times and, since it cannot afford to store that information permanently, every time it must receive it from P_2. □

This result is now used to establish the following theorem:

Theorem 10. Any VLSI (n,k)-sorter, with $k \geq 2\log n$, satisfies the bound

$$AT = \Omega(kn\sqrt{n\log n}). \tag{38}$$

Proof. With the notation of theorem 5 we again choose $\mathcal{U} = \{X_i^j : 0 \leq i < n, 0 \leq j < k-\log n$ and select P_1 with area (storage) $s = \sigma n\log n$ for a suitable constant σ. With this choice $|\mathcal{U}| = (k-\log n)n$; if we can show that $I(m|\sigma n\log n,\infty) = \Omega(m)$, then bound (38) follows from (16), since $\ell_0 = 4\sqrt{s} = 4\sqrt{\sigma n\log n}$.

[In the following arguments we introduce parameters γ, σ, ε, and ξ whose values could be chosen for a fine tuning of the lower bound. A feasible choice that gives the right feeling for the range of these parameters is $\gamma = 1/12$, $\sigma = 5/24$, $\varepsilon = 1/4$, $\xi = 29/36$.]

For given real $\varepsilon \in [0,1]$, we partition the sequence of times $0,1,\ldots,T-1,T$ into two subsequences $t_1' < t_2' < \ldots < t_u'$ if $|J_1(t_h')| \geq \varepsilon\log n$ and $t_1'' < t_2'' < \ldots < t_v''$ if $|J_1(t_h'')| < \varepsilon\log n$. Obviously

$$\sum_{h=1}^{u} |J_1(t_h')| + \sum_{h=1}^{v} |J_1(t_h'')| = |J_1| . \tag{39}$$

We now consider separately the contributions to the information exchange of positions belonging to $J_1(t_h')$ (t'-sequence) and $J_1(t_h'')$ (t''-sequence).

<u>t'-Sequence</u>. At time t'_h, all the $|J_1(t'_h)|n$ bits pertaining to positions in $J_1(t'_h)$ are in the chip, and - by definition of J_1 - at least $(1-\gamma)|J_1(t'_h)|n$ of them have to be output by P_1. Due to the bound on the storage of P_1, at least $(1-\gamma)|J_1(t'_h)|n-\sigma n\log n$ of these bits are in P_2 at time t'_h and will be eventually transferred to P_1. Thus, the overall contribution I' to the information exchange associated with the t'-sequence is bounded below as

$$I' \geq (1-\gamma)n \sum_{h=1}^{u} |J_1(t'_h)| - u\sigma n\log n. \tag{40}$$

Since $|J_1(t'_h)| \geq \varepsilon\log n$, $u\varepsilon\log n \leq \sum_{h=1}^{u} |J_1(t'_h)|$ and (40) can be rewritten as

$$I' \geq (1-\gamma-\sigma/\varepsilon)n \sum_{h=1}^{u} |J_1(t'_h)|. \tag{41}$$

<u>t"-Sequence</u>. We decompose $[0,T]$ into the sequence of intervals $([t''_{h_i}+1, t''_{h_i+1}]$: $i = 0,\ldots,L$, $t''_{h_0} = -1$ and $t''_{h_L+1} = T)$ as follows: For some real ξ $(\varepsilon < \xi \leq 1)$ and for $i = 0,\ldots,L-1$

$$(\xi-\varepsilon)\log n < \sum_{h=h_i+1}^{h_{i+1}} |J_1(t''_h)| \leq \xi\log n.$$

Such a decomposition always exists, since $|J_1(t''_h)| < \varepsilon\log n$. Moreover,

$$L \leq \sum_{h=1}^{v} |J_1(t''_h)|/((\xi-\varepsilon)\log n), \tag{42}$$

since at least $(\xi-\varepsilon)\log n$ positions complete their input in any given interval. We can now apply Lemma 5 with $[\tau_1^i,\tau_2^i] = [t''_{h_i}+1, t''_{h_{i+1}}]$, $Z_i = J_1(t''_{h_i+1}) \cup \ldots \cup J_1(t''_{h_{i+1}})$, and $s = \sigma n\log n$ to conclude that the overall contribution I" to the information exchange associated with the t"-sequence is bounded below as

$$I'' \geq \frac{(1-\gamma)}{2} \quad n \sum_{h=1}^{v} |J_1(t_h'')| - L\sigma\log n$$

which, by using (42) can be restated as

$$I'' \geq \left(\frac{1-\gamma}{2} - \frac{\sigma}{\xi-\varepsilon} \right) n \sum_{h=1}^{v} |J_1(t_h'')| \ . \tag{43}$$

By choosing $\xi = \varepsilon + (1/\varepsilon - (1-\gamma)/\ 2\sigma\ )^{-1}$ the coefficients of the two bounds (41) and (43) become identical and, by (39), we obtain

$$I(m|\sigma n\log n,\infty) \geq I' + I'' \geq (1-\gamma-\sigma/\varepsilon)n|J_1| \ . \tag{44}$$

We now observe that Lemma 3 (with B = m) can be invoked, yielding:

$$I(m|\sigma\log n,\infty) \geq I(m) \geq \gamma(m-|J_1|n). \tag{45}$$

If we choose $\sigma/\varepsilon = 1-2\gamma$, then (44) and (45) yield

$$I(m|\sigma n\log n,\infty) \geq \gamma/2\ m$$

and the theorem is proved. □

3.2.3. AT/logA bound

We shall now consider the constraints posed by computational friction on sorting long keys and obtain a new lower bound which turns out to be stronger than both the AT^2 and the AT bounds for k large enough and for a suitable range of computation times.

Theorem 11. Any VLSI (n,k)-sorter, with $k \geq 2 \log n$, satisfies the bound

$$AT \geq \beta_0 nk \log(nk/T) \tag{46}$$

for some constant $\beta_0 > 0$. Equivalently,

$$AT/\log A = \ (nk)\ . \tag{47}$$

Proof. We want to apply Theorem 6 to the set of input variables $\mathcal{U} \triangleq \{X_i^j : 0 \leq i < n,\ 0 \leq j < k-\log n\}$. To the set $\mathcal{U}_t$ of elements of $\mathcal{U}$ input exactly at time t, we associate the set $\mathcal{Y}_t$ of the $\lfloor|\mathcal{U}_t|/2\rfloor$ least significant variables in $\{Y_i^j : X_i^j \in \mathcal{U}_t\}$. Ties (variables pertaining to the same bit position) are broken arbitrarily. All the variables in $\mathcal{Y}_t$ are functionally dependent upon the $\lceil|\mathcal{U}_t|/2\rceil$ most significant variables of $\mathcal{U}_t$ so that - in the terminology of Theorem 6 - $\alpha(s) = \lceil s/2\rceil$. If we set the bits $X_i^{k-1} \ldots X_i^{k-\log n}$ to be the binary representation of i, $Y_i^j = X_i^j$ for all i's and j's, and $\mathcal{Y}_t$ trivially carries $|\mathcal{U}_t|/2$ bits of information about $\mathcal{U}_t$ so that $\beta(s) = \lfloor s/2\rfloor$. Thus, all the assumptions of Theorem 6 and Corollary 2 are satisfied, and bounds (46) and (47) follow from (20) and (21) with $|\mathcal{U}| = \theta(nk)$. ☐

3.2.4 Remarks

From Theorem 9 and Theorem 10 we know that there exist constants β_1 and β_2 such that the performance of any (n,k)-sorter, with $k \geq 2\cdot\log n$ satisfies the bounds $AT \geq \beta_1 kn\sqrt{n\log n}$, and $AT^2 \geq \beta_2 kn(n\log n)$. These bounds coincide at time $T_0 = (\beta_2/\beta_1)\sqrt{n\log n}$. The AT bound is stronger for $T > T_0$, and the AT^2 bound is stronger for $T < T_0$.

If we compare the friction bound (46) with the saturation bound (38), written as $AT > \beta_1 kn\sqrt{n\log n}$, we see that the former is stronger when $k/T > g(n)/n$, with $g(n) \triangleq 2^{(\beta_1/\beta_0)\sqrt{n\log n}}$. If we now consider the crude fan-in argument that prescribes $T \geq \tau\log(kn)$ (for a suitable constant τ), we see that $k/T > g(n)/n$ is satisfied by feasible computation times only if $k/T > k_0/T_0$ where $k_0 = \theta(g(n)\sqrt{\log n/n})$ is the solution of $k/(\tau\log(kn)) = g(n)/n$, and $T_0 = \theta(\sqrt{n\log n})$ (see Figure 2).

3.5 Summary

We conclude this section with a summary of known lower bounds on sorting given in Table I. Bounds on T follow from trivial fan-in arguments. Bounds on A are due to Leighton [11] for long keys, and to Siegel [12] for short and intermediate-length keys. Bisection bounds on AT^2 for long keys are due to Thompson [1] (for word-level protocols) and Leighton [11] (for arbitrary protocols). For short and medium-length keys bisection bounds are due to Siegel [12,13]. An alternative proof of $AT^2 = \Omega(n^2h^2)$, based on primary and secondary flow, is given in [14]. The remaining bounds are those given in this paper. The $AT^2 = \Omega(nr)$ result has been independently obtained in [12].

The above bounds show that the area-time complexity of sorting is determined by different computational mechanisms, each of which dominates for a particular range of keylengths and computation times. An effective overview of the different bounds is provided by Figure 2.

All the lower bounds of Table I are known to be tight or nearly tight. Several optimal circuits for $(n, \log n + \theta(\log n))$-sorting, (the first case of sorting analyzed in the VLSI model, which partially overlaps with sorting of intermediate-length and long keys), have been proposed by the authors [15,16,17] and by Leighton [11]. Optimal circuits for keys of any intermediate length are giv in [14]. Constructions that attain the AT^2, AT and AT/logA bounds for long keys, as well as constructions that are near optimal for short keys are described in [18]. Further optimal designs for several key lengths are reported by Cole and Siegel [19] (personal communication). Several designs of VLSI sorters, potentially of practical interest even when asymptotically suboptimal, are surveyed by Thompson [20]. A systematic discussion of VLSI sorting can be found in [14].

SUMMARY OF LOWER BOUNDS FOR (n,k)-SORTING

Lower Bound Techniques / Length of the Keys	$1 \leq k \leq \log n$ ($r = 2^k$)	$\log n < k < 2\log n$ ($h = k - \log n$)	$2\log n \leq k$
Bipartition + Information Exchange	$AT^2 = \Omega(r^2\log^2(1+n/r))$	$AT^2 = \Omega(n^2h^2)$	$AT^2 = \Omega(n^2\log^2 n)$
Square Tesselation + Information Exchange	$AT^2 = \Omega(nr)$	—	$AT^2 = \Omega(nk(n\log n))$
Square Tesselation + Saturated Information Exchange	$AT = \Omega(n\sqrt{r})$	—	$AT = \Omega(nk\sqrt{n\log n})$
Friction			$\frac{AT}{\log A} = \Omega(nk)$
Storage	$A = \Omega(r\log(1+n/r))$	$A = \Omega(nh)$	$A = \Omega(n\log n)$
Bounded Fan-in	$T = \Omega(\log n)$	$T = \Omega(\log n)$	$T = \Omega(\log n + \log k)$

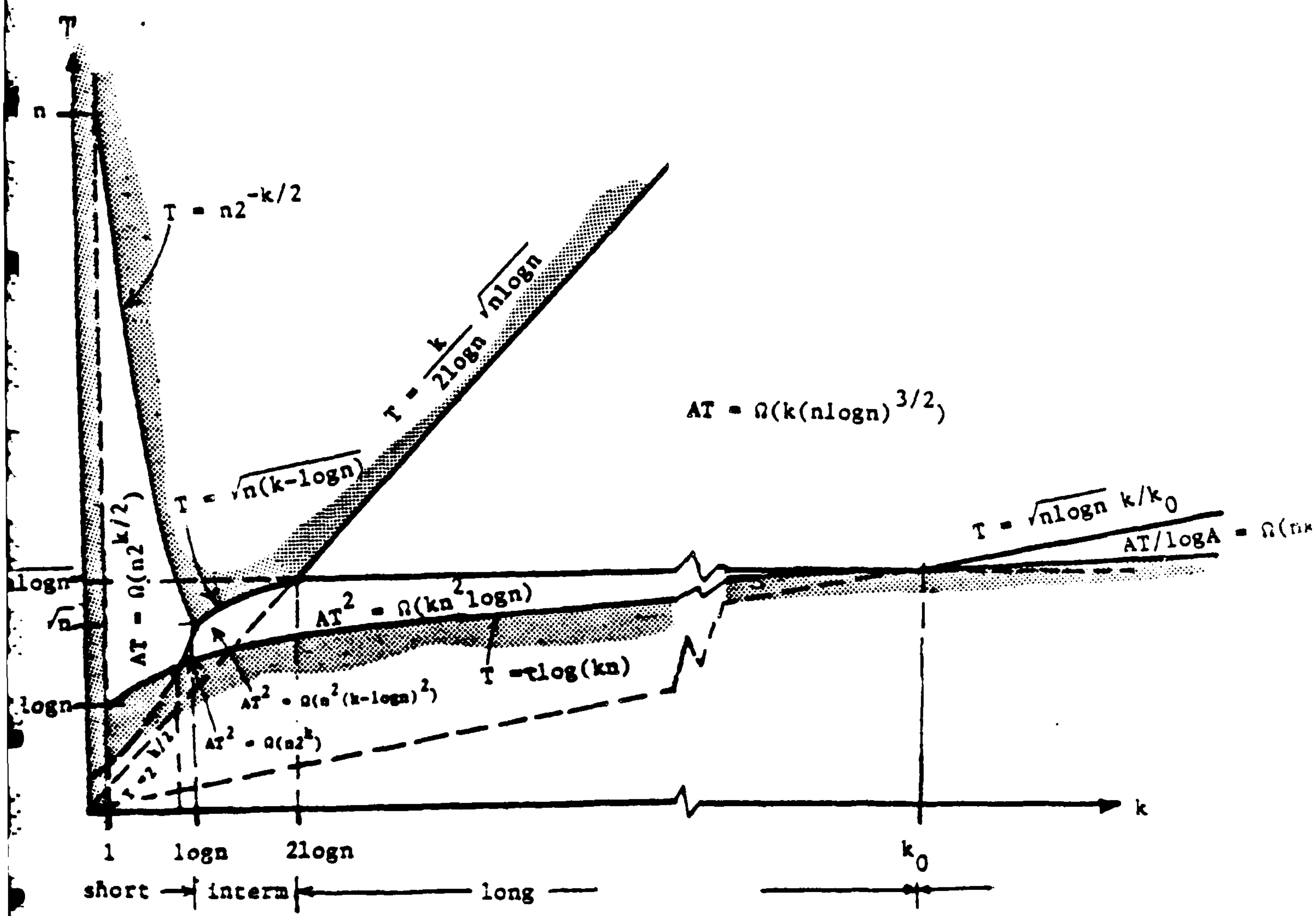


Figure 2. Regions of the (k,T) plane and their dominant regimes.

4. Other Applications

The lower-bound techniques of Section 2 afford the analysis of several other problems beside those discussed in Section 3. A few results are stated and commented on below. The proofs are similar in flavor to those given in Section 3, and are omitted here for the sake of brevity.

CYCLIC SHIFT. The input of the (n,k)-cyclic-shift problem is a pair (X,s), where X is an $n \times k$ binary array, $s \in \{0,1,\ldots,n-1\}$, and the output is a binary array Y such that $Y_j = X_{(j-s) \bmod n}$.

<u>Theorem 12</u>. Any VLSI (n,k)-cyclic-shifter satisfies the bounds

$$AT^2 = \Omega(kn^2) \ , \tag{48a}$$

$$AT = \Omega(kn^{3/2}) \ . \tag{48b}$$

<u>Comments</u>. A detailed proof of Theorem 15 is given in [14]. Result (48a) has also been reported in [4].

MERGING. The (n,k)-merging problem is the specialization of (n,k)-sorting when the input subsequences $(X_0,\ldots,X_{n/2-1})$ and $(X_{n/2},\ldots,X_{n-1})$ are sorted.

<u>Theorem 13</u>. Any VLSI (n,k)-merger satisfies the following bounds. For $k \leq \log n$, and $r \triangleq 2^k$:

$$AT^2 = \Omega(nr) \ , \tag{49a}$$

$$AT = \Omega(nr^{1/2}) \ . \tag{49b}$$

For $k > \log n$:

$$AT^2 = \Omega((k-\log n)n^2), \tag{49c}$$

$$AT = \Omega((k-\log n)n^{3/2}), \tag{49d}$$

$$AT/\log A = \Omega((k-\log n)n) \ . \tag{49e}$$

Comments. Bounds (49a), (49b) and (49e) are identical in the order to those obtained for (n,k)-sorting. Bounds (49c) and (49d) are a factor of logn smaller. The reason is that while primary flow is of the same order in both problems, the secondary flow is a factor of logn smaller for merging. Indeed, $\theta(n)$ bits are necessary and sufficient to specify a merging permutation.

RECORD SORTING. A formulation of sorting, more general than the one considered in Section 2, assumes that the n items to be sorted are records of two fields, the key (of k bits) and the information (of p bits). The output is the multiset of input records rearranged in order of nondecreasing keys. Sorting is called stable when records with the same key preserve in the output sequence the same relative order they had in the input sequence.

Theorem 14. A VLSI stable sorter of records, with $k \leq p$ and $k \leq \log n$, satisfies the bounds

$$AT^2 = \Omega(pn^2k) \ , \tag{50a}$$

$$AT = \Omega(pn(nk)^{1/2}) \ . \tag{50b}$$

Comments. Proofs are similar to those of Theorems 9 and 10. However, observe that there is no analogous to Theorem 14, since the bit positions of the information field do not interact with each other.

The lower bound techniques of this paper are certainly applicable to other problems, and we hope they will contribute to a coherent formulation of VLSI computation theory.

REFERENCES

1. C. D. Thompson, A Complexity Theory for VLSI, Ph.D. Thesis, Dept. of Comp. Science, Carnegie-Mellon University; August 1980.

2. R. B. Johnson, "The complexity of a VLSI adder," Information Processing Letters, vol. 11, n. 2, pp. 92-93; October 1980.

3. G. M. Baudet, "On the area required by VLSI circuits," in H. T. Kung, R. Sproull, and G. Steele (eds.), VLSI Systems and Computations, pp. 100-107, Computer Science Press, Rockville, MD; 1981.

4. D. Angluin, "VLSI: On the merits of batching," manuscript; April 1982.

5. Z. Kedem, "Optimal allocation of computational resources in VLSI," Proc. 23rd Annual Symposium on the Foundations of Computer Science, Chicago, IL, pp. 379-386; November 1982.

6. A. C. C. Yao, "Some complexity questions related to distributive computing," Proc. 11th Annual ACM Symposium on Theory of Computing, Atlanta, GA, pp. 209-213; April 1979.

7. A. C. C. Yao, "The entropic limitations on VLSI computations," Proc. 13th Annual ACM Symposium on Theory of Computing, Milwaukee, WI, pp. 308-311; April 1981.

8. J. Ja'Ja' and V. K. P. Kumar, "Information transfer in distributed computing with applications to VLSI, Journal of the ACM, vol. 31, n. 1, pp. 150-162; January 1984.

9. J. E. Savage, "Area-time tradeoffs for matrix multiplication and related problems in VLSI models," Journal of Computer System Science, vol. 22, n. 2, pp. 230-242; April 1981.

10. R. P. Brent and H. T. Kung, "The chip complexity of binary arithmetic," Journal of the ACM, vol. 28, n. 3, pp. 521-534; July 1981.

11. F. T. Leighton, "Tight bounds on the complexity of parallel sorting," Proc. 16th Annual ACM Symposium on Theory of Computing, Washington, D. C., pp. 71-80; April 1984. (Also IEEE Trans. on Comput.; April 1985.)

12. A. Siegel, "Minimal storage sorting circuits," IEEE Trans. on Comput., vol. C-34, n. 4; April 1985.

13. A. Siegel, "Tight area bounds and provably good AT^2 bounds for sorting circuits," Tech. Report #122 Courant Institute, New York University; June 1984.

14. G. Bilardi, The Area-Time Complexity of Sorting, Ph.D. Thesis, Univ. of Illinois; December 1984.

15. G. Bilardi and F. P. Preparata, "An architecture for bitonic sorting with optimal VLSI performance," IEEE Trans. Comp., vol. C-33, n. 7, pp. 640-651; July 1984.

16. G. Bilardi and F. P. Preparata, "A minimum area VLSI network for O(logN) time sorting," Proc. 16th Annual ACM Symposium on Theory of Computing, Washington, D. D., pp. 64-70; April 1984. (Also IEEE Trans. on Comput., April 1985.)

17. G. Bilardi and F. P. Preparata, "The VLSI optimality of the AKS sorting network," Information Processing Letters, vol. 20, n. 2, pp. 55-59, Feb. 1985.

18. G. Bilardi and F. P. Preparata, "The influence of key length on the area-time complexity of sorting," I. C. A. L. P., Nauplion, Greece, July 1985 (to appear).

19. R. Cole and A. Siegel, "Optimal VLSI network that sort n numbers in an arbitrary, fixed range," manuscript.

20. C. D. Thompson, "The VLSI complexity of sorting," IEEE Trans. Comp., vol. C-32, n. 12, pp. 1171-1184; December 1983.

END

FILMED

1-86

DTIC