19TH Asilomar Conf. on Circuits, Systems & Comp. (IEEE) Nov 85

On the Area-Time Optimal Design of l-Selectors

Clark D. Thompson

Division of Computer Science
573 Evans Hall
U.C. Berkeley, CA 94720
U.S.A.

Hiroto Yasuura

Department of Information Science
Kyoto University
Kyoto, 606
Japan

ABSTRACT

We prove upper and lower bounds on the area-time complexity of the problem of selecting the l-th smallest of n k-bit integers. When chip area is restricted to $\Theta(n)$, our $O(k \log n)$-time circuit is provably area-time optimal to within a factor of $O(\log n)$. We present several circuit constructions having area other than $\Theta(n)$, but these are typically a factor of $O(l)$ from area-time optimality. The development of tight upper and lower bounds for l-selectors of nonlinear area is thus an open problem.

1. Introduction

We present upper and lower bounds on the area-time complexity of circuits that select the l-th smallest of a set of integers. Our lower bounds are trivial, but most of our upper bounds are interesting on both theoretical and practical grounds. Indeed, some of our VLSI algorithms may have application in the design of cost-effective database machines.

This work is the first to prove area-time bounds on the VLSI complexity of l-selection. Aside from some very recent work on the minimum area requirements for l-selection in VLSI,[6] almost all previous work on this problem has been done in the context of serial models of computation. On most such models, it is strictly easier to select than to sort. The l-th smallest of n integers can be found in $O(n)$ time and $O(n)$ space, whereas it takes $\Omega(n \log n)$ time and $\Omega(n)$ space to sort n integers.

In VLSI, we find it is also strictly easier to select than to sort, for almost all n, k, and l. For example, when $k \geq 2\log n$, an area-time product of $nk\sqrt{n \log n}$ is necessary and sufficient to sort n k-bit integers.[3, 5] This area-time product for sorting is strictly larger than our upper bound on l-selection (Theorem 3) of $AT = O(nk \log n)$. Our circuit is about the same size, but much faster, than Bilardi's or Cole and Siegel's area-time optimal sorting circuits. In general, we can show that selecting is strictly easier than sorting in the AT sense for all n, l, and $\epsilon > 0$, when $k > (2+\epsilon)\log\log n$.

This work was supported in part by the National Science Foundation, through its Computer Engineering program, under grant DMC 84-06408.

2. Model of VLSI Computation

We use the following standard assumptions to bound the area and time complexity of VLSI circuits. The area of a circuit is the sum of its gate, wire, and I/O pad areas. Each gate and I/O pad occupies unit area. Each wire has unit width, and there are a constant (greater than one) number of wiring layers.

We choose the simplest model of VLSI time, the unit-delay model, despite the fact that it is unrealistic in the limit of large n.[4] In the unit-delay model, each wire has unit bandwidth and unit delay, regardless of its length. Gates and I/O pads also have unit bandwidth and unit delay. No time penalty is assessed for excessive fanout or fanin. Instead we restrict all our gates, wires, and I/O pads to have $O(1)$ fanin and fanout.

Our I/O assumptions are also conventional. We assume chip I/O is semelective,[8] when-oblivious,[7] and where-oblivious.[7] By "semelective", we mean that each input bit is read by the circuit exactly once. The when- and where- oblivious assumptions mean that the timing and location of I/O events are independent of the data values. In particular, the circuit has no control over when and where it reads its inputs.

3. Problem Definition

The problem of selecting the l-th smallest of a set of n integers is of identical complexity as that of selecting the l-th largest. Hence we can restrict our attention to the case that $l \leq n/2$.

DEFINITION. A circuit is an l-selector if it is capable of outputting the l-th smallest of a (multi)set of n k-bit integers. Integers are input and output in binary format on a semelective, where- and when- oblivious I/O schedule. The parameters n, k, and l may be fixed at the time of circuit construction.

4. Related Work

There has been a lot of recent activity in a closely related topic, minimal-area sorting. Siegel proved the following tight bounds on the area of when- and where-oblivious sorting circuits that read their inputs r times:[9]

$$A = \begin{cases} \Theta(\log n + 2^k(\log\frac{n}{r} - k + 1)), & \text{if } k < \log\frac{n}{r} \\ \Theta(\log n + \frac{n}{r}(k - \log\frac{n}{r} + 1)), & \text{if } k \geq \log\frac{n}{r} \text{ and } k = O(\log n) \end{cases}$$

The semelective case ($r = 1$) of this result was proved independently.[6,3] The semelective result was proved under slightly weaker assumptions: it does not depend on Siegel's where-oblivious restriction nor on his $k = O(\log n)$ condition.

The area complexity of finding the l–th smallest of n k–bit integers in a semelective, when-oblivious fashion was recently evaluated:[6]

$$A = \begin{cases} \Theta(2^k(\log l - k + 1)), & \text{if } k < \log l \\ \Theta(\min\{n, l(k - \log l + 1)\}), & \text{if } k \geq \log l \end{cases}$$

Thus it takes at least as much area to find a median ($l = n/2$) than to sort, when $k \leq \log n$. Sorting takes strictly more area than median-finding, however, when $k \geq (1+\epsilon)\log n$.

Finally, note that the problem of finding the minimal element ($l = 1$) has area complexity $A = \Theta(\min\{n, k\})$. Below we show that area-time optimality is in fact acheivable for such minimum-area 1–selector circuits.

5. Lower Bounds

It is easy to show that the least-significant output bit of an l–selector depends on all of its nk input bits. The following theorem is then immediate from the unit bandwidth restriction on a circuit's unit-area I/O pads.

THEOREM 1. The area A and computation time T of any l–selector is bounded by $AT \geq nk$.

Since a circuit's gates have constant fan-in, it takes $T = \Omega(\log nk)$ time to compute any output depending on nk bits.

THEOREM 2. The area A and computation time T of any l–selector is bounded by $AT^\alpha = \Omega(nk(\log nk)^{\alpha-1})$, for any $\alpha \geq 1$.

6. Circuit Constructions

We divide our constructions into four categories. First come our best algorithms for general n, k, and l. Next, we treat the case of algorithms with small area. Then we describe algorithms which are good for small l. Finally, we interpret a result of Stout, obtaining the fastest known l–selector that is within n^ϵ of being area-time optimal, for any $\epsilon > 0$.

6.1. Algorithms for the General Case

Our first circuit operates on a "radix select" principle, constructing the MSB of its output from the MSBs of its inputs.[6]

THEOREM 3. The l–th smallest of a set of n k–bit integers can be found in $A = O(n)$ and $T = O(k \log n)$.

Proof sketch. From the most-significant bit of each input, the circuit is able to compute the most-significant bit of the output. The next-most significant output bit is computed next, on the basis of $n + O(\log n)$ bits of circuit state and n more bits of circuit input. Successively less significant bits are computed in an analogous fashion.

We can implement this algorithm using the H-tree layout of a parallel counter tree. The area of the circuit is thus $A = O(n)$. The parallel counter tree generates the binary representation of the number of '0's in the current set of n input bits in time $O(\log n)$. Since we need k iterations of this calculation, the algorithm takes $O(k \log n)$ time. □

Since sorting n k–bit integers takes $AT = \Omega(\min\{n\,2^{k/2}, n^{3/2} \mid 1+k-\log n \mid\})$,[5] selecting is strictly easier than sorting for $k > (2+\epsilon)\log\log n$. Of course, selection is never harder than sorting, since we can build a selector from a sorter.

A slight generalization of Theorem 3's circuit yields a speedup of $O(\log\log n)$ at an area penalty of $O(\log\log n)$. Our second circuit is thus preferable to the first whenever $\alpha > 1$ in the optimization criterion AT^α.

The idea is to reduce the number of iterations by processing $\log\log n$ bits of each input at once. We perform 1-counting computations on an H-tree, as before. Each leaf of the tree holds $\log\log n$ bits in binary format. It sends its value up the tree in unary format, in $O(\log n)$ time. An internal node of the H-tree, at level i, sends the sum of two $(i-1)$–bit integers as a i–bit result to its parent. This can be done in unit time, if a redundant carry-save notation is employed. Thus we have the following theorem.

THEOREM 4. The l–th smallest of a set of n k–bit integers can be found in $A = O(n \log\log n)$ and $T = O(k \log n / \log\log n)$.

6.2. Algorithms for small l

For $l = 1$, namely the problem of finding the minimum element, we have an l–selection algorithm that is AT–optimal for all n, k, and l. This algorithm is also AT^α–optimal for all $\alpha \geq 1$, when $k = O(\log n)$.

THEOREM 5. The smallest of a set of n k–bit integers can be found in $A = O(nk/(k+\log n))$ and $T = O(k+\log n)$.

Proof sketch. We divide the input integers into $nk/(k+\log n)$ groups. Each group thus consists of $(k+\log n)/k$ k–bit integers. We process these integers in $(k+\log n)/k$ batches. We read the first batch of integers, serially, into a binary tree of serial comparators. After $\log(nk/(k+\log n)) = O(\log n)$ time, the MSB of the smallest integer in the first batch appears at the root of the tree. We store this value in a serial shift register at the root. We pipeline this minimum-finding operation, starting the second batch of integers as soon as the LSB of the first batch has cleared the comparators at the leaves. As new values emerge from the root, they are compared in a bit-serial fashion with the value saved in the root's serial shift register. □

We can generalize the algorithm above, by using merge elements instead of bit-serial comparators at the internal nodes of the tree.

THEOREM 6. The l-th smallest of a set of n k-bit integers can be found in $A = O(nkl/(k+\log l \log n))$ and $T = O(k+\log l \log n)$.

Proof sketch. Once again, we organize the computation in the form of an H-tree. This time the tree has $nk/(l(k+\log l \log n))$ leaves, and we read the integers in $(k+\log l \log n)/k$ batches. Each leaf reads l k-bit integers from each batch through l I/O ports, bit-serially, in $O(k)$ time. Each leaf is equipped with an $O(l^2)$-area bitonic sorter, so that it can sort its input with $O(\log^2 l)$ latency.[12] The internal nodes of the tree are responsible for merging the pipelined data from the leaves, discarding the largest l integers and sending the smallest l integers up the tree. Each such internal node is thus implemented as an odd-even merge network of latency $O(\log l)$ and area $O(l^2)$. At the root of the tree, we have an additional merging node and a $k \times l$-bit shift register. The root merges the l smallest integers of each batch with the l smallest integers from all previous batches. The height of the tree is $\log(nk/(l(k + \log l \log n)) = O(\log n)$, so the total latency through the tree is $O(k + \log l \log n)$. □

We note, in passing, that many variations on Theorem 6 are possible. We could have made the sorters and mergers faster, by making them word-parallel instead of bit-serial. Indeed, the whole network could be reduced to $O(\log nk)$ depth, by using an $O(\log n)$-depth sorting circuit.[1] However, such a circuit would be far from AT-optimal, since its area would be $O(n^2)$. Indeed it is easy to see that this line of attack is doomed for large l. Any l-selector that uses l-sorters on n/l sets of l k-bit integers must have an area-time product in excess of $nk\sqrt{l}$. For $l = O(n)$, this is much worse than the area-time product of the l-selectors of Theorems 3 and 4.

6.3. Algorithms with small area

For completeness, we present a few l-selectors with area sublinear in n. These circuits are not close to being AT-optimal, however, unless l and/or k are very small.

THEOREM 7.[6] The l-th smallest of a set of n k-bit integers can be found in $A = O(kl)$ and $T = O(n+k)$.

Proof sketch. We build the first l stages of a word-parallel n-comparator bubble sorter.[12] The depth of the circuit is $O(l)$. The I/O time is $O(n+k)$, due to the skewed word-parallel input format. □

Cole and Siegel's AT-optimal n-sorter for $k > 2\log n$ offers an immediate asymptotic improvement on Theorem 7, for a restricted range of k.

THEOREM 8. The l-th smallest of a set of n k-bit integers can be found in $A = O(lk)$ and $T = O(n\sqrt{\log l}/\sqrt{l})$, if $\log l < k < 2^{\sqrt{l \log l}}$.

Proof sketch. A circuit of $O(lk)$ area can sort $2l$ k-bit integers in $O(\sqrt{l \log l})$ time, if $\log l < k < 2^{\sqrt{l \log l}}$. We sequentially use this circuit n/l times, merging groups of l input integers with the l smallest integers previously encountered. The lower bound on k is due to the fact that the circuit keeps track of $\log l$-bit I/O port addresses. The upper bound on k is explained by Bilardi's $AT/\log A$ lower bound on sorting circuits for extremely long wordlengths.[3] □

When k or l is small, we can use data compression to obtain a smaller circuit:

THEOREM 9.[6] The l-th smallest of a set of n k-bit integers can be found in $A = O(\min\{2^k, l\}(|k - \log l| + 1))$ and $T = O(nk)$.

Once again, Cole and Siegel provide us with an improvement for some k.

THEOREM 10. The l-th smallest of a set of n k-bit integers can be found in $A = O(2^k(\log l - k))$ and $T = O(n\,2^{-k/2}/(\log l - k))$, if $2\log\log l < k < \log l - 2\log\log l$.

Proof sketch. We build a circuit that can sort $2l$ k-bit integers in $T = O(l/(2^{k/2}\log(2l/k)))$. It retains the smallest l integers encountered thus far in $O(2^k \log(2l/k))$ area. □

6.4. Stout's selector

None of our circuits thus far has used the "recursive filtering" concept of the $O(n)$-time serial selection algorithm. This concept, while quite efficient in a serial or parallel computer with random memory access,[2] is difficult to use efficiently in a VLSI model. For example, any implementation of a "big switch" giving n processors random access to an nk-bit shared memory with nk-bandwidth carries an AT^2 cost of $\Omega(n^2k^2)$. Quentin Stout has finessed this difficulty, developing an l-selection algorithm[10, 11] for an n-node pyramid computer that runs in time $O((\log n)^{(\log\log n)/2})$.

THEOREM 11. The l-th smallest of a set of n k-bit integers can be found in $A = O(nk^2/(\log k)^2)$ and $T = O(\log k (\log n)^{(\log\log n)/2})$.

Proof sketch. We build the fastest possible VLSI implementation of Stout's algorithm. Each of his n processors is realized as a fully-parallel k-bit processor of $O(k^2/\log^2 k)$ area. Elementary k-bit operations on such processors take $O(\log k)$ time. □

We think it is possible to improve on Stout's algorithm for general l, although at present we only have an improvement for the case $l = n^\beta$, $\beta < 1$.

7. Conclusions

Our unrestricted results are summarized in the following table. In addition, we have an excellent circuit for minimum-finding (Theorem 5), as well as improvements on Theorems 7 and 9 for restricted ranges of k.

	Area	Time
Thm 3.	n	$k \log n$
Thm 4.	$n \log\log n$	$k \log n / \log\log n$
Thm 6.	$nkl/(k+\log l \log n)$	$k+\log l \log n$
Thm 7.	kl	$n+k$
Thm 9.	$\min\{2^k, l\}(\lvert k-\log l\rvert + 1))$	nk
Thm 11.	$nk^2/(\log k)^2$	$\log k (\log n)^{(\log\log n)/2}$

Table 1. Area-time performance of various l-selectors.

The selectors of Theorems 3, 4, and 11 are within $O(n^{\epsilon}k)$ of area-time optimality, for all n, k, l and for any $\epsilon > 0$. Theorems 3 and 4 are our best results for the general case, coming within a factor of $\log n$ of the trivial $AT = \Omega(nk)$ lower bound.

Our nearly tight results, Theorems 3 and 4, are circuits of approximately $O(n)$ area. Our only selection circuits with significantly non-linear area (Theorems 6, 7, and 9) are area-time non-optimal by as much as a factor of l. To put this remark into perspective, note that $l = n/2$ for median-finding. In many circumstances, therefore, the best known l-selector is an l-sorter (Theorems 8 and 10) with an area-time performance that is $\min\{\sqrt{l}, \sqrt{2^k}\}$ worse than our area-time lower bound on l-selection.

Of course we would like to close the gap between our upper and lower bounds. This closure will be difficult, judging from the $AT = O(nk)$ performance that is possible for $l = 1$ (Theorem 5). Theorems 7 and 9 indicate that $AT \approx nk$ performance can be obtained for $l = O(1)$. Thus a tight area-time bound will depend critically, in some unknown fashion, on the relative sizes of n, k, and l.

References

1. M. Ajtai, J. Komlós, and E. Szemerédi, "An O(nlogn) sorting network," in *Proc. 15th Annual ACM Symp. on Theory of Computing*, pp. 1-9, April 1983.
2. Selim G. Akl, "An optimal algorithm for parallel selection," *IPL*, vol. 19, pp. 47-50, July 26, 1984.
3. Gianfranco Bilardi, "The area-time complexity of sorting," ACT-52 (Ph.D. dissertation), Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, December 1984.
4. G. Bilardi, M. Pracchi, and F. P. Preparata, "A critique of network speed in VLSI models of computation," *IEEE Journal of Solid-State Circuits*, vol. SC-17, no. 4, pp. 696-702, August 1982.
5. Richard Cole and Alan Siegel, "On information flow and sorting: new upper and lower bounds for VLSI circuits (extended abstract)," in *Proc. 26th Annual Symp. on Foundations of Computer Science*, pp. 208-221, IEEE Computer Society, October 1985.
6. Pavol Ďuriš, Ondrej Sýkora, Clark Thompson, and Imrich Vrťo, "A minimum-area circuit for l-selection," *submitted to Algorithmica*, June 1985.
7. Richard J. Lipton and Robert Sedgewick, "Lower bounds for VLSI," in *Proc. 13th Annual ACM Symp. on Theory of Computing*, pp. 300-307, May 1981.
8. J. Savage, "Planar circuit complexity and the performance of VLSI algorithms," in *VLSI Systems and Computations*, ed. H. T. Kung, Bob Sproull, Guy Steele, pp. 61-68, Computer Science Press, October 1981.
9. Alan Siegel, "Tight area bounds and provably good AT^2 bounds for sorting circuits," Report number 122, Courant Institute, NYU, 22 pp., June 1984.
10. Q. F. Stout, "Sorting, merging, selecting, and filtering on tree and pyramid machines," in *Proc. of 1983 International Conference on Parallel Processing*, pp. 214-221, 1983.
11. Q. F. Stout, "Mesh-connected computers with broadcasting," *IEEE Trans. on Computers*, vol. c-32, no. 9, pp. 826-830, September 1983.
12. C. D. Thompson, "The VLSI complexity of sorting," *IEEE Trans. Computers*, vol. C-32, no. 12, pp. 1171-1184, December 1983.