Breaking the $\Theta(n \log^2 n)$ Barrier for Sorting with Faults

Tom Leighton[1] *Yuan Ma*[1] *C. Greg Plaxton*[2]

Abstract

In this paper, we study the problem of constructing a sorting circuit, network, or PRAM algorithm that is tolerant to faults. For the most part, we focus on fault patterns that are random, i.e., where the result of each comparison is independently faulty with probability upper-bounded by some constant. All previous fault-tolerant sorting circuits, networks, and parallel algorithms require $\Omega(\log^2 n)$ depth and/or $\Omega(n \log^2 n)$ comparisons to sort n items. In this paper, we construct:

- a passive-fault-tolerant sorting circuit with $O(n \log n \log\log n)$ comparators, thereby answering a question posed by Yao and Yao in 1985,
- a reversal-fault-tolerant sorting network with $O(n \log^{\log_2 3} n)$ comparators, thereby answering a question posed by Assaf and Upfal in 1990, and
- a deterministic $O(\log n)$-step $O(n)$-processor EREW PRAM fault-tolerant sorting algorithm, thereby answering a question posed by Feige, Peleg, Raghavan, and Upfal in 1990.

The results are based on a new analysis of the AKS circuit, which uses a much weaker notion of expansion that can be preserved in the presence of faults. Previously, the AKS circuit was not believed to be fault-tolerant because the expansion properties that were believed to be crucial for the performance of the circuit are destroyed by random faults.

Extensions of our results for worst-case faults are also presented.

[1]Department of Mathematics and Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139. Supported by AFOSR Contract F49620-92-J-0125, DARPA Contract N00014-91-J-1698, and DARPA Contract N00014-92-J-1799. Email: ftl@math.mit.edu, yuan@cs.stanford.edu.

[2]Department of Computer Science, University of Texas at Austin, Austin, TX 78712. Supported by NSF Research Initiation Award CCR–9111591, and the Texas Advanced Research Program under Grant No. 003658–461. Email: plaxton@cs.utexas.edu.

1 Introduction

Sorting circuits have intrigued and challenged computer scientists for decades. They have also proved to be very useful for a variety of applications, including circuit switching and packet routing [10]. In this paper, we study the problem of constructing sorting circuits that are tolerant to a potentially large number of faults.

The study of fault-tolerant sorting circuits was initiated by Yao and Yao [20] in 1985. In particular, Yao and Yao proposed a fault model in which a **faulty comparator** simply outputs its two inputs without comparison (i.e., the items are output in the same order in which they are input); we will refer to such a fault as a **passive fault**. They defined an n-input **fault-tolerant** sorting circuit to be a circuit that remains a sorting circuit with probability at least $1 - \frac{1}{n}$ even if each comparator is independently faulty with probability upper-bounded by a constant strictly less than 1. They observed that any sorting circuit can be made into a passive-fault-tolerant sorting circuit if each of the original comparators is replicated $O(\log n)$ times. (In this paper, all logarithms are taken base 2 unless otherwise specified.) This immediately yields a passive-fault-tolerant sorting circuit with $O(\log n)$ **depth** and $O(n \log^2 n)$ **size**. (The depth of a circuit or network is defined to be the number of levels in the circuit or network where each register is associated with at most one comparator in each level. The size of a circuit or network is defined as the number of comparators in the circuit or network.) Whether or not there is an alternative approach to fault-tolerance that requires fewer comparators has remained an interesting open question for several years. In particular, Yao and Yao conjectured that $\omega(n \log n)$ comparators are needed to construct a fault-tolerant sorting or merging circuit, but no proof of this conjecture has yet been discovered.

Since Yao and Yao, many researchers have studied fault-tolerant circuits, networks, and algorithms for sorting-related problems in various models. (See [4, 6, 7, 12, 13, 18, 19, 20].) Despite all of these efforts, the $O(\log n)$-gap between the trivial upper and lower bounds has remained open for Yao and Yao's question for both sorting and merging. One approach to narrowing the $O(\log n)$-gap was investigated by Leighton, Ma, and Plaxton [13], who constructed an $O(n \log n \log \log n)$ size circuit that sorts any permutation with probability at least $1 - \frac{1}{n}$. Yet, this circuit does not yield an answer to Yao and Yao's question because sorting any input permutation with high probability is not sufficient to guarantee sorting all input permutations with high probability, and hence it is not sufficient to guarantee that a faulty version of the circuit will be a sorting circuit with high probability. In other words, for a randomly generated fault pattern, there are likely to be some input permutations for which the circuit of [13] fails to sort. (Formally, a **fault pattern** completely specifies which comparators, if any, are faulty and how they are faulty. That is, a fault pattern of a circuit or network contains all the information needed to specify the functionality of all the comparators in the circuit or network.)

Since 1985, several other fault models have also been formulated for the study of fault-tolerant sorting circuits [4, 13]. In the **reversal fault** model, a faulty comparator outputs the two inputs in reversed order regardless of their input order. In the **destructive fault** model, a faulty comparator can output the two inputs in reversed order, and it can also output one of the two inputs in both of the output registers (i.e., the result of a comparison between x and y can be (f, g) where f, g can be any of $\{x, y, \min(x, y), \max(x, y)\}$). In order to tolerate destructive and/or reversal faults, Assaf and Upfal [4] introduced a new computational model for the study of the sorting problem. In their new model, more than n registers are allowed to sort n items and **replicators** are used to copy the item stored in one register to another register. We will call this model the **sorting network model** to distinguish it from the classic **sorting circuit model** in which only n registers are used to sort n inputs and no replicators are allowed. For example, a 2-input sorting network with replicators that is tolerant to any single reversal or destructive fault is illustrated in Figure 1.

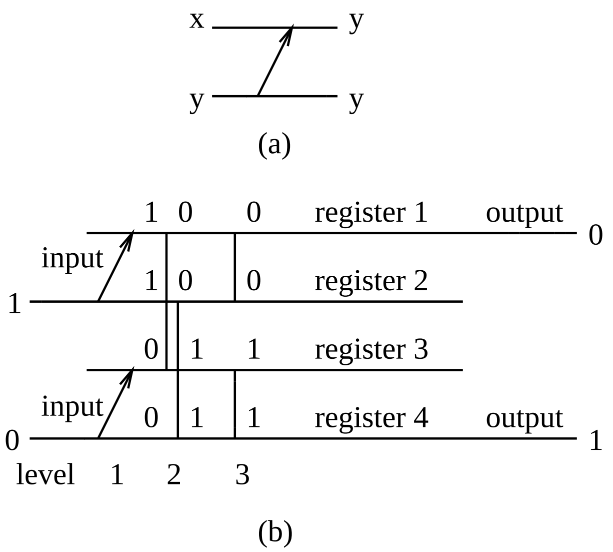

Figure 1: (a) A replicator. (b) A 2-input sorting network that tolerates any single reversal or destructive fault.

In [4], Assaf and Upfal described a general method for converting any sorting circuit into a reversal-fault-tolerant or destructive-fault-tolerant sorting network. In particular, given an n-input sorting circuit with depth d and size s, the fault-tolerant network produced by the Assaf-Upfal transformation has depth $O(d)$ and size $O(s \log n)$. (Asymptotically, it makes no difference whether or not the replicators are counted towards the size since an optimal network would make a copy of an item only if the item were to be input to a comparator.) When used in conjunction with the AKS sorting circuit [1], this provides a reversal-fault-tolerant or destructive-fault-tolerant sorting network with $O(n \log^2 n)$ size.[1]

The Assaf-Upfal method proceeds by making $\Theta(\log n)$ copies of each item and replacing each comparator with $\Theta(\log n)$ comparators, followed by a majority-enhancing device that is constructed from expanders. As a consequence, the size of the resulting network is increased by a $\Theta(\log n)$ factor. Whether or not there is an alternative approach to fault-tolerance that can avoid the $\Theta(\log n)$ factor blowup in size (even for the much simpler problem of merging)

[1] In this paper, as in [4], we assume that the replicators are fault-free. This is not a particularly unreasonable assumption since replicators can be hard-wired and they do not contain any logic elements. In fact, some of our results can be extended to handle a model in which replicators are also allowed to fail.

was an interesting open question posed in [4]. For destructive faults, this question was recently answered by Leighton and Ma [12] with an $\Omega(n \log^2 n)$ lower bound for both merging and sorting. The question concerning reversal-fault-tolerant sorting networks remained open, however.

The problem of sorting with faults has also been studied in the PRAM model of computation. An algorithm is called a **fault-tolerant sorting algorithm** if it sorts any n items with probability at least $1 - \frac{1}{n}$ even if an answer to any comparison query is incorrect with probability upper bounded by a constant strictly less than $\frac{1}{2}$. (Note that when the fault probability is equal to $\frac{1}{2}$, we cannot obtain any useful information from a comparison.) Feige, Peleg, Raghavan, and Upfal [7] designed a randomized fault-tolerant sorting algorithm that uses $O(\log n)$ expected time on an $O(n)$-processor CREW PRAM. They left open the question of whether or not there is a deterministic fault-tolerant sorting algorithm that runs in $o(\log^2 n)$ steps on $O(n)$ processors.

In this paper, we develop a fault-tolerant sorting circuit, network, and PRAM algorithm that beats the $\Theta(n \log^2 n)$ barrier in each of the preceding models, thereby partially or wholly resolving the questions posed by Yao and Yao [20], Assaf and Upfal [4], and Feige et al. [7]. In particular, we construct:

(i) a passive-fault-tolerant sorting circuit with $O(\log n \log\log n)$ depth and $O(n \log n \log\log n)$ size, which resolves the question posed by Yao and Yao [20] to within an $O(\log\log n)$ factor,

(ii) a reversal-fault-tolerant sorting network with size $O(n \log^{\log_2 3} n)$, which partially resolves the question of Assaf and Upfal [4], and

(iii) a fault-tolerant sorting algorithm that runs in $O(\log n)$ steps on an $O(n)$-processor EREW PRAM, which resolves the question of Feige, et al. [7].

All of these results are based on a surprisingly strong fault-tolerance property of the AKS circuit [1]. The results are surprising because the AKS circuit was not previously believed to be useful in the context of fault-tolerant sorting algorithms; when a constant fraction of the comparators fail to work, the expansion property, which plays a central role in the functionality of the fault-free AKS sorting circuit, is lost and it appears that the AKS circuit cannot sort at all. (This is perhaps the main reason that people were unable to make progress on Yao and Yao's question.) The novelty of our work is to show that some loose expansion properties, which can be preserved even in the presence of faults, are sufficient to "approximate-sort", and that approximate-sorting can be combined with other methods to sort.

Although we will mainly focus on the study of circuits, networks, and algorithms that are tolerant to random faults, all of our techniques apply for constructing circuits, networks, and algorithms that are tolerant to worst-case faults. Our results for worst-case faults include the first asymptotically non-trivial upper bound for the depth of worst-case passive-fault-tolerant sorting circuits, a worst-case reversal-fault-tolerant sorting network, and an optimal worst-case fault-tolerant EREW PRAM algorithm for sorting. The techniques in this paper can also be applied to other sorting-related problems. For example, we will construct a passive-fault-tolerant selection circuit with the asymptotically optimal size of $O(n \log n)$.

Throughout the paper, we use n to denote the number of input items and $\rho < 1$ to denote the upper bound on the failure probability for each comparator, unless otherwise specified. A circuit, network, or algorithm for solving a problem Q is defined to be (ρ, ϵ)**-fault-tolerant** if it satisfies the following condition: when each comparator or comparison is independently faulty with probability upper bounded by ρ, then with probability at least $1 - \epsilon$, a faulty version of the circuit, network, or algorithm remains a circuit, network, or algorithm that solves Q on all possible input instances. When we simply refer to a circuit, network, or algorithm as **fault-tolerant**, we mean that the circuit, network, or algorithm is $(\rho, \frac{1}{n})$-fault-tolerant for some constant $\rho < 1$. For any constant c, all of our constructions can be made into constructions with success probability at least $1 - \frac{1}{n^c}$ with no more than a constant factor increase in size, but we will be content to achieve a success probability of at least $1 - \frac{1}{n}$ in most parts of the paper.

As an alternative, we could have defined the notion of (ρ, ϵ)-fault-tolerance by assuming that ρ is exactly the failure probability of each comparator or comparison, as opposed to an upper bound on the failure probability. In general, these two definitions are not equivalent, and it is straightforward to show that any (ρ, ϵ)-fault-tolerant circuit, network, or algorithm defined in the preceding paragraph is also (ρ, ϵ)-fault-tolerant according to the alternative definition.[2] Hence, to get the strongest possible results, we will use the definition of the preceding paragraph for all upper-bound results, which include all but Theorems 4.1 and 4.2, and use the alternative definition for all the lower-bound results, which include Theorems 4.1 and 4.2.

Finally, we point out that all of the comparators used in the constructions of our circuits and networks move the small input to the top register and the large input to the bottom register. Following the notation of Knuth [9], this means that all of our circuits and networks are **standard**. All of our lower bounds are proved for the general case, i.e., we do not assume that the circuit is standard in the lower bound proofs. In the fault-free case, it has been proved that any non-standard sorting circuit can be converted into a standard sorting circuit with the same depth and size (see Exercise 16 on page 239 of [9]). However, we do not know if a similar result is true when the circuit is subject to passive faults.

The remainder of the paper is organized as follows. In Section 2, we prove that the AKS circuit has certain useful fault-tolerance properties. In Section 3, we use the AKS circuit to construct an $O(\log n \log \log n)$-depth passive-fault-tolerant sorting circuit. In Section 4, we describe the reversal-fault-tolerant approximate-sorting circuits and sorting networks. In Section 5, we describe the fault-tolerant PRAM sorting algorithm. Finally, in Section 6, we extend our results to worst-case faults.

We remark that this paper is combined from the results presented in [11] and [13].

2 The Analysis of the AKS Circuit

In this section, we show that the AKS circuit [1] has certain fault-tolerance properties under both the passive and reversal fault models. These fault-tolerance properties will be the cornerstone for most of the fault-tolerant sorting circuits, networks, and algorithms in the paper.

[2] A detailed discussion of this phenomena in the context of Boolean circuits with noisy gates can be found in [17].

We believe that our new analysis of the AKS circuit is of separate interest in its own right. The section is divided into three parts: Section 2.1 explains why the previously known analyses of the AKS circuit are not sufficient to establish the desired fault-tolerance properties and highlights the major difficulties in the new analysis. Section 2.2 contains a brief description of the AKS circuit and the relevant parameter choices. Section 2.3 proves the main theorem of the section and its corollary.

2.1 The Need for a New Analysis of the AKS Circuit

The key component of the AKS circuit is the ε-halver. An m-input circuit is called an ε-**halver** if, on any m distinct inputs, it outputs at most εk of the k smallest (largest) inputs into the bottom (top) $\frac{m}{2}$ registers for any $k \le \frac{m}{2}$. For any constant $\varepsilon > 0$, a bounded-depth ε-halver can be built from an expander as follows. Take an $\frac{m}{2} \times \frac{m}{2}$ d-regular $(\varepsilon, \frac{1-\varepsilon}{\varepsilon})$-bipartite expander with vertex sets A and B, where d is a constant dependent on ε. (A bipartite $\frac{m}{2} \times \frac{m}{2}$ graph with vertex sets A and B is called an (α, β)-**expander** if any $k \le \alpha m$ nodes in A (or B) are connected to at least βk nodes in B (or A). Explicit constructions of expanders can be found in [15].) Assign each vertex in A to a register in the top half of the circuit, and assign each vertex in B to a register in the bottom half of the circuit. Partition the edges of the expander into d disjoint matchings. Assign a comparator between two registers at level i in the halver if and only if the corresponding vertices are connected by an edge in the ith matching of the expander.

To see why this construction yields an ε-halver, we assume for the purposes of contradiction that the circuit is not an ε-halver. Without loss of generality, we can assume that there exist m distinct inputs and an integer $k \le \frac{m}{2}$ such that strictly more than εk of the k smallest inputs are output into R, a set of strictly more than εk registers in the bottom half of the circuit. Let R' be the set of registers in the top half of the circuit that are connected to some registers in R. It is not hard to show that all registers in R' contain outputs with rank at most k. Therefore, all of the $|R| + |R'| \ge |R| + \frac{1-\varepsilon}{\varepsilon}\varepsilon k = |R| + (1-\varepsilon)k > k$ output items contained in either R or R' have rank at most k. This is a contradiction.

It would be nice if the ε-halver could tolerate random faults automatically or if the ε-halver could be made fault-tolerant with a $o(\log n)$ factor increase in the depth. (For example, if this were possible, Yao and Yao's question would have an easy answer.) As we have seen in the previous paragraph, however, the fact that $|R'| \ge (1-\varepsilon)k$ is critical to guarantee the ε-halver property, and this fact in turn depends on the expansion property of the expander. Unfortunately, the following observation indicates that the expansion is lost in the presence of faulty comparators and that the cost of achieving fault-tolerant expansion is very high. For example, if $d = \Theta(1)$ and each comparator in the ε-halver constructed above is independently faulty with constant probability, then with high probability there exists a set of $k = \Theta(m)$ registers in the bottom half of the circuit for which all associated comparators are faulty. Hence, if the k smallest inputs are all input to these registers, then the inputs cannot be moved to the top half of the circuit. This shows that the ε-halver itself cannot withstand random faults. Moreover, even if we increase the depth of an ε-halver by a nonconstant $o(\log n)$ factor, any constant number of registers are connected to only $o(\log n)$ comparators, and, with probability $\omega(\frac{1}{n})$, these registers are not connected to any working comparators.

(Using more careful arguments, we can actually show that with probability approaching 1, there exists a set of $\omega(1)$ registers that are not connected to any working comparators.) Hence, if a constant number of the smallest inputs are input to these registers, they cannot be reliably moved to the top half of the halver.

Since expansion plays a central role for both the functionality and the correctness proof of the AKS circuit, the loss of such expansion in the presence of faulty comparators is a fatal problem for all previously known analyses of the AKS circuit. In fact, the loss of the expansion property even makes the approach of using the AKS circuit to construct fault-tolerant sorting circuits seem to be hopeless. The novelty of our work is to show that, even without guaranteeing "local" expansion at any single expander, it is possible to enforce a certain "global" expansion property that is sufficient to guarantee that the AKS circuit functions reasonably well.

2.2 The Description of a Modified AKS Circuit

In this section, we describe a modified AKS circuit that will be shown to possess certain fault-tolerance properties. In Section 2.2.1, we describe a (slight) modification of the AKS circuit described by Paterson in [16]. In particular, we modify some parameter choice, and replace the so-called *separators* of [16] by a new family of building blocks that we call *partitioners*. In Section 2.2.2, we further modify the AKS circuit into an l-AKS circuit, where l is a parameter to be specified later that corresponds to the amount of fault-tolerance attained by the circuit.

We will be content with proving that certain parameter choices guarantee the desired fault-tolerance properties. No attempt will be made to keep the involved constants small. In particular, an extremely large constant (much larger than the previously best known constant for the AKS circuit) is hidden behind the O-notation for all of our circuits, networks, and algorithms.

2.2.1 The AKS Circuit

In this section, we describe Paterson's version of the AKS circuit [16]. For simplicity, we closely follow the description in [16] whenever possible. In particular, we use the same letters to denote the same parameters as in [16] unless otherwise specified.

At a high level, the AKS circuit is structured around a complete binary tree. For ease of notation, we will refer to the tree to be defined as the **AKS tree** and refer to each of its nodes as an **AKS tree node**. The AKS circuit works in **stages** starting from stage 1. Each of the stages corresponds to a constant number of consecutive levels of the AKS circuit. From one stage to the next, each register of the AKS circuit is moved from one AKS tree node to another. Within a stage, a sorting-related device is applied to all registers located at each AKS tree node. The AKS circuit is then defined by specifying: (i) the structure of the AKS tree, (ii) how the registers are moved within the AKS tree, and (iii) what sorting-related device is applied and how it is applied to registers residing at each AKS tree node.

The remainder of Section 2.2.1 is divided into four parts. The first three parts address the three problems identified immediately above. The last part deals with a particular technical problem.

The AKS Tree

The AKS tree is a complete binary tree, with the root at the top. Each of the AKS tree nodes has a dynamically changing (as a function of the stage) **capacity** that specifies the maximum number of items that can be stored in the node.[3]

To describe the capacity of an AKS tree node precisely, we first choose parameters

$$A = 3 \text{ and } \nu = \frac{43}{48} + \varepsilon' \tag{1}$$

where ε' is a sufficiently small positive constant. Note that ε' has nothing to do with the parameter ε of [16], and it is used only to bound ν away from $\frac{43}{48}$. In [16], ν was first chosen to be exactly $\frac{43}{48}$, and later was slightly increased after the effect of "integer rounding" was considered. The particular choice of ε' is not important. For example, we may choose

$$\varepsilon' = 10^{-2}. \tag{2}$$

The capacity of an AKS tree node X, denoted by $\mathrm{cap}(X)$, is determined by

$$\mathrm{cap}(X) = \nu^{t-1} A^d \left(1 - \frac{1}{4A^2}\right) n, \tag{3}$$

where t denotes the stage and d denotes the level of X in the AKS tree (with the root considered to be at level 0). Intuitively, such a choice of capacity means that: (i) at any time, the capacities of all nodes at the same level of the tree are equal, and (ii) the capacity of a node decreases geometrically with the stage and increases geometrically with the level of the node. Note that the second property ensures that items will be squeezed downward in the AKS tree toward their final locations.

Let $|X|$ denote the number of items that are actually contained in an AKS tree node X. By the definition of capacity, $|X| \leq \mathrm{cap}(X)$ for all X. We say that X is **empty, full,** or **partially full** if $|X| = 0$, $|X| = \mathrm{cap}(X)$, or $0 < |X| < \mathrm{cap}(X)$, respectively. The AKS tree may be viewed to be infinite, with most of its nodes empty. For convenience, we call each of the nodes at the lowest nonempty level of the AKS tree a **leaf**.

Register Movements

Roughly speaking, the left subtree of the AKS tree corresponds to items being sorted that are smaller than the median, and the right subtree of the AKS tree corresponds to the items larger than the median. Recursively, both of the subtrees have a similar property. Initially, (almost) all of the items are located in the root of the AKS tree. As time progresses, (most) items move downward in the AKS tree, and the goal is to move all of the items to their correct locations within the AKS tree. It would be ideal if we could exactly partition all of the items residing at each AKS tree node in a constant number of steps, since then we could sort n

[3]More precisely, since we need to construct an oblivious circuit, as opposed to an adaptive algorithm, we can only move the registers (not the items themselves) among AKS tree nodes. For simplicity, however, we will not distinguish between a register and the item contained in the register, as long as the meaning is clear from the context. For example, an alternative (and perhaps more rigorous) definition of the capacity of a node is the maximum number of registers (instead of items) allowed in the node.

items in the desired $O(\log n)$ time. Unfortunately, it is easy to see that exactly partitioning m items requires $\Omega(\log m)$ steps in the circuit model. Hence, any sorting circuit based on *exact* partitioning has $\Omega(\log^2 n)$ depth. One novelty of the AKS circuit is to use a certain approximate-partition gadget at each AKS tree node to move *most* of the items residing at the node to the correct child of the node in a constant number of steps. A detailed and rigorous treatment of this gadget is given in the next part of Section 2.2.1.

Assuming that the approximate-partition gadget can be constructed, we now describe how the registers are moved within the AKS tree. In what follows, we will specify various sizes of sets of registers (or items) as real numbers. How to round these real numbers to some close integers is the subject of the last part of Section 2.2.1.

Initially, an arbitrary set of $\left(1 - \frac{1}{4A^2}\right)n$ registers are located at the root of the tree, (i.e., the root is filled to its capacity), and all of the levels below the root are empty. In general, at each stage, all registers residing in an AKS tree node X are first partitioned into four parts (by a partitioner to be defined in the next part of Section 2.2.1): FL (far-left), CL (center-left), CR (center-right), and FR (far-right) so that

$$|FL| = |FR| = \min\left\{\frac{\lambda}{2}\,\mathrm{cap}(x),\ \frac{|X|}{2}\right\} \tag{4}$$

and

$$|CL| = |CR| = \frac{|X|}{2} - |FL|,$$

where

$$\begin{aligned} \lambda &= \frac{\nu - \frac{1}{2A}}{2A - \frac{1}{2A}} \qquad (5) \\ &= \frac{1}{8} + \frac{6}{35} \times 10^{-2}. \qquad (6) \end{aligned}$$

(In [16], λ was first set to 1/8 and then slightly increased after the effect of "integer rounding" was considered.) Then, $FL \cup FR$ are sent up to the parent of X, and CL and CR are sent down to the left and right child of X, respectively.

The preceding description of register movement is almost correct except that the root does not have a parent yet. To get around this boundary situation, we keep a subset of registers at a special node, which we call the **cold storage**, immediately above the root. In particular, we would like to treat the cold storage as simulating half of the root's parent (in terms of capacity), one quarter of its grandparent, one-eighth of its greatgrandparent, and so on. (This will be useful in the analysis of the AKS circuit; see Case 1 of the proof of Theorem 2.1. Also, we only let the cold storage simulate every other level since, in Claim 2.1, it will be proved that the levels of the AKS tree are alternately full and empty above the leaf level.) This requires that the capacity of the cold storage be

$$\begin{aligned} \frac{\mathrm{cap(root)}}{2A} + \frac{\mathrm{cap(root)}}{(2A)^3} + \cdots &= 2A\frac{\mathrm{cap(root)}}{4A^2 - 1} \quad \text{at even stages, and} \\ \frac{\mathrm{cap(root)}}{(2A)^2} + \frac{\mathrm{cap(root)}}{(2A)^4} + \cdots &= \frac{\mathrm{cap(root)}}{4A^2 - 1} \quad \text{at odd stages.} \end{aligned}$$

No comparator or partitioner is applied to the registers in the cold storage. Registers are exchanged between the root and the cold storage as if the cold storage is the parent of the root. More precisely: (i) from an odd stage to an even stage, all of the $\lambda\,\mathrm{cap}(\mathrm{root})$ registers in FL and FR of the root are sent to the cold storage, where cap(root) denotes the capacity of the root at the odd stage, and (ii) from an even stage to an odd stage, $\frac{1-\lambda}{2A}\,\mathrm{cap}(\mathrm{root})$ registers are sent from the cold storage to the root, where cap(root) denotes the capacity of the root at the even stage. (Here, we do not have to worry about the case where the root may be partially full, since, by our choice of capacities, the items tend to move downward in the AKS tree and the root is always full at odd stages and empty at even stages.)

We have thus specified the rules for moving registers within the AKS tree. The rules would not be valid unless the capacity constraints on all AKS tree nodes can be observed simultaneously. To establish the validity of the rules (and for future applications), we prove the following stronger result.

Claim 2.1 *At odd stages: (i) all the nodes at odd levels are empty, (ii) all the nodes at even levels and strictly below the leaf level are empty, (iii) all the nodes at even levels and strictly above the leaf level are full, and (iv) either all the leaves are full or all the leaves are partially full. A symmetric series of properties holds at even stages. The cold storage is always full.*

Proof: It is convenient to consider the cold storage as simulating half of the root's parent, one quarter of its grandparent, one-eighth of its greatgrandparent, and so on. Under such an assumption, the last sentence of the claim follows from the rest of the claim since the cold storage works as if it is a chain of "regular" nodes. This will enable us to avoid the undesirable complication caused by the boundary situation at the cold storage.

By equation 5, we have

$$\nu = 2\lambda A + \frac{1-\lambda}{2A}.$$

The correctness of the claim is then easily checked by induction on stages, the preceding equation, and some algebraic manipulations. ■

In the base case of our induction proof for Claim 2.1, we need the following facts during stage 1: (i) all registers are either at the cold storage or at the root, and (ii) the capacities of the root and the cold storage sum to n. The latter fact explains why we have set $\mathrm{cap}(\mathrm{root}) = n(1 - \frac{1}{4A^2})$ for $t = 1$ in equation 3.

As time progresses, the capacity of each AKS tree node gets smaller and smaller. When the capacity of the root becomes so small (i.e., smaller than a certain constant) that we are about to split the AKS tree into two (unconnected) subtrees, special care is needed to guarantee the performance of the AKS circuit. In [16, Section 6], this was handled using Batcher's sorting circuit [5]. In our application of the AKS circuit, however, we never need to run the AKS circuit all the way to completion. Instead, we stop the AKS circuit far before the capacity of the root gets close to a constant. (See the last paragraph of Section 2.2.2 for more details.) Consequently we do not need to worry about this particular problem.

A (λ', ϕ)-Partitioner

As we have seen in the preceding part of Section 2.2.1, a sorting-related device is needed to partition items residing at each AKS tree node into four parts: FL, CL, CR, and FR. Such a task was accomplished by using the so-called separator in [16] (and by using the so-called near-sorting circuit in [1]). Informally, a **separator** is slightly more powerful than a halver in the sense that a separator not only moves most of its inputs to the correct half but also moves most of its "extreme" inputs close to the extreme positions. Since, as discussed in Section 2.1, we cannot build ε-halvers that are both efficient and fault-tolerant, we cannot construct efficient and fault-tolerant separators either.

In [16], ε-halvers and separators are defined in terms of their functionality. The procedure given for building an ε-halver from an expander, and for building a separator from ε-halvers, represents only one of many possible constructions. In this paper, we will be interested in the specific construction given in [16], although the construction will likely fail to have the properties needed to be ε-halvers or separators once faults are introduced.

Throughout the paper, we choose ϕ as a sufficiently large constant, and we choose

$$\lambda' = \frac{1}{8}. \tag{7}$$

Note that λ' is strictly smaller than λ, but the two parameters can be arbitrarily close if ε' is arbitrarily close to 0.

We define a **ϕ-divider** with m inputs to be a circuit constructed by using an $\frac{m}{2} \times \frac{m}{2}$ d-regular bipartite $(\frac{1}{\phi+1}, \phi)$-expander to connect the top half and the bottom half of the m registers in the same fashion as we used to construct the ε-halver in Section 2.1.[4]

Comparing our ϕ-divider with an ε-halver, we see that ϕ plays a role similar to that of $\frac{1}{\varepsilon}$ in [16]. In [16], ε was set to a particular (small) value. In this paper, however, we will only show the existence of a sufficiently large constant ϕ that will ensure certain fault-tolerance properties of the AKS circuit. Such a constant ϕ appears to be much larger than the value of $\frac{1}{\varepsilon}$ that was chosen in [16].

Given the construction of a divider, an m-input **(λ', ϕ)-partitioner** at an AKS tree node X is then constructed by applying ϕ-dividers in at most $1 + \log \frac{1}{\lambda'}$ rounds: We first apply an m-input ϕ-divider to all m registers. Then, we apply an $\frac{m}{2}$-input ϕ-divider to the top $\frac{m}{2}$ registers and another $\frac{m}{2}$-input ϕ-divider to the bottom $\frac{m}{2}$ registers. Next, we apply an $\frac{m}{4}$-input ϕ-divider to the top $\frac{m}{4}$ registers and another $\frac{m}{4}$-input ϕ-divider to the bottom $\frac{m}{4}$ registers (we do not do anything to the $\frac{m}{2}$ "middle" registers). We then apply another $\frac{m}{8}$-input ϕ-divider to the top $\frac{m}{8}$ registers and another $\frac{m}{8}$-input ϕ-divider to the bottom $\frac{m}{8}$ registers. We keep doing this until we have applied a divider to a group with at most $\lambda' \operatorname{cap}(X)$ registers. Altogether, we apply the dividers for at most $1 + \log \frac{1}{\lambda'}$ rounds. For ease of notation, we will refer to the dividers applied in the ith round of a partitioner as the **ith round dividers**. For $i > 1$, there are two ith round dividers, and we refer to the divider applied to the top (resp., bottom) $\frac{m}{2^{i-1}}$ registers as the **top** (resp., **bottom**) **ith round divider**. We make the convention that the only divider in the first round of a partitioner can be treated as both a top and a bottom divider in that round.

[4] In the AKS tree, we imagine that small (resp., large) items should be sent to the left (resp., right). In a circuit (such as a divider or partitioner), we imagine that small (resp., large) items should be sent to the top (resp., bottom). Thus, top (resp., bottom) in a circuit corresponds to left (resp., right) in the AKS tree.

When X is partially full, our construction of a partitioner differs slightly from that of a separator in [16]: (i) the number of rounds of dividers used in our partitioner depends on $\mathrm{cap}(X)$, and may be less than $1+\log\frac{1}{\lambda'}$, and (ii) we do not use the so-called "virtual elements" which were used in Paterson's separator construction to fill X to its capacity so that the number of rounds of halvers in the separator is always $1+\log\frac{1}{\lambda'}$.

In place of Paterson's use of "virtual elements", we prove the following simple fact that will be useful to analyze the behavior of partially full AKS tree nodes.

Claim 2.2 *Each of the dividers in a partitioner associated with an AKS tree node X has at least* $\min\{|X|, \frac{\lambda'}{2}\mathrm{cap}(X)\}$ *inputs.*

Proof: If $|X| \le \lambda' \mathrm{cap}(X)$, then the partitioner consists of exactly one divider with $|X|$ inputs. If $|X| > \lambda' \mathrm{cap}(X)$, then the number of inputs to either of the last round dividers is at least $\frac{\lambda'}{2}\mathrm{cap}(X)$. ■

Let $\overline{FL}$ (resp., $\overline{FR}$) consist of all registers output to the top (reps., bottom) half of the top (resp, bottom) divider in the last round of the partitioner associated with X. By the partitioner construction, we immediately have $|\overline{FL}| \le \min\{\frac{\lambda'}{2}\mathrm{cap}(X), \frac{|X|}{2}\}$. Hence, by equations 4, 6, and 7,

$$|\overline{FL}| \le |FL|. \tag{8}$$

Let L (resp., R) denote the set of registers output to the top (resp., bottom) half of the partitioner associated with X. We now partition the items at X into FL, CL, CR, and FR as follows: (i) from F (resp., R), move all of the items in $\overline{FL}$ (resp., $\overline{FR}$) into FL (resp, FR); (ii) move an arbitrary subset of items remaining in L (resp., R) to FL (resp., FR) so that FL (resp., FR) satisfies equation 4; (iii) let the items remaining in L (resp., R) form CL (resp., CR).

Although our partitioner (resp., divider) construction is (almost) the same as the separator (resp., halver) construction of [16], a partitioner (resp., divider) is conceptually different from a separator (resp., halver) in that a separator (resp., halver) is defined based on its input-output behavior and a partitioner (resp., divider) is explicitly constructed from bipartite expanders. Of course, a fault-free partitioner (resp., divider) is one type of separator (resp., halver).

In the preceding description of a partitioner, we have treated all of the constituent dividers as having an even number of inputs. This may be a problem if the number of inputs to the partitioner is not divisible by 16. We will show how to get around this technical problem in the next part of Section 2.2.1.

Integer Rounding

Thus far, in the description of register movement, we have specified the "ideal" sizes of various sets of registers as real numbers. In reality, we need to choose such sizes as integers. This can be accomplished by the simple technique introduced in [16] as follows.

We first let each ideally empty AKS tree node be actually empty. Next, we consider an arbitrary *nonempty* AKS tree node X. Let T_X be the subtree rooted at X. If the ideal number of items contained in T_X is γ, then we let the actual number of items in T_X be $16\lceil\frac{\gamma}{16}\rceil$. Let γ_0 denote the ideal number of items contained in each subtree rooted at a grandchild of X, and let x and $Z(x)$ denote the ideal and actual numbers of items contained in X, respectively. Then, we have

$$Z(x) = 16\left\lceil \frac{\gamma}{16} \right\rceil - 64\left\lceil \frac{\gamma_0}{16} \right\rceil . \tag{9}$$

Given that $x = \gamma - 4\gamma_0$, we thus have

$$-64 \leq Z(x) - x \leq 16. \tag{10}$$

As we will see in Section 2.2.2, $\mathrm{cap}(X) \geq \sqrt{n}$ will hold in our applications of the AKS circuit. Thus, by Claim 2.1, equation 10 implies $Z(x) \geq 0$ if X is not a leaf. In addition, it is easy to check that $Z(X) \geq 0$ when X is a leaf. Therefore, we conclude that we can indeed maintain the relationship of equation 9 without letting X contain a negative number of registers.

We point out that, the parameters, 16 and 64, in the preceding paragraph are chosen differently from those of [16]. In particular, we have defined $Z(x)$ to be a multiple of 16. This ensures that each of the dividers in our partitioner construction has an even number of inputs, as desired. This should be compared with Paterson's use of the so-called "virtual elements" in his separator construction when the number of inputs to the separator is not divisible by 16 [16, pages 83–84].

Finally, we remark that the capacities of the AKS tree nodes are untouched in our integer rounding scheme, i.e., they remain fractional. Thus, an AKS tree node X may be slightly overloaded up to an additive constant, due to the effect of integer rounding. Similarly, the third and fourth properties of Claim 2.1 only hold up to an additive constant. Most importantly, however, inequality 8 (which will be important in our analysis of the AKS circuit) still holds after the integer rounding. This can be easily checked by using the following facts: (i) $\lambda - \lambda'$ is a positive constant, and (ii) $\mathrm{cap}(X) \geq \sqrt{n}$ for all AKS tree nodes X in our applications of the AKS circuit (see Section 2.2.2).

2.2.2 The l-AKS Circuit

If we were only interested in passive-fault-tolerant sorting circuits, the (slightly) modified AKS circuit described in Section 2.2.1 would be sufficient. However, to construct reversal-fault-tolerant circuits and networks, we need to further modify the AKS circuit into an l-AKS circuit, by using odd-even transposition circuits as described below. A $2l$-input **odd-even transposition circuit** consists of comparators between registers $2i-1$ and $2i$ ($1 \leq i \leq l$) at odd levels, and between registers $2i$ and $2i+1$ ($1 \leq i \leq l$) at even levels, where the registers in the circuit are labeled from top to bottom with $1, 2, \ldots, 2l$. It is well known that such a circuit of $2l$ levels sorts all $2l$-input permutations in the fault-free case (for example, see [9]). In this paper, we are interested in fault-tolerance properties of various circuits and will use odd-even transposition circuits with more than $2l$ levels.

For any given integer $l > 0$, we use the following general technique to modify any family of circuits $\mathcal{F}$ into another family of circuits $\mathcal{F}'$ with parameter l. In general, an m-input circuit $\mathcal{C}'$ in $\mathcal{F}'$ is constructed from an $\frac{m}{l}$-input circuit $\mathcal{C}$ in $\mathcal{F}$ as follows. For each $i \leq \frac{m}{l}$, replace the ith register in $\mathcal{C}$, r_i, by a group of l registers, $r_{i_1}, \ldots, r_{i_l}$. This group will be referred to as a **block** corresponding to register r_i. Replace each comparator in $\mathcal{C}$ that connects registers r_i and r_j by a $2l$-input and $4l$-depth odd-even transposition circuit that connects $r_{i_1}, \ldots, r_{i_l}$ and $r_{j_1}, \ldots, r_{j_l}$.

(The reason for doing this can be found in Lemma 2.1 and in the proofs of Lemma 2.2 and Theorem 2.1.) Such a circuit $\mathcal{C}'$ in $\mathcal{F}'$ will be referred to as the circuit constructed from $\mathcal{C}$ in $\mathcal{F}$ by applying $2l$-input and $4l$-depth odd-even transposition circuits. In particular, an m-input l-AKS circuit is constructed from an $\frac{m}{l}$-input modified AKS circuit described earlier by applying $2l$-input and $4l$-depth odd-even transposition circuits. For example, a 1-AKS circuit is the modified AKS circuit described earlier with each of the comparators replicated 4 times. This technique is essential to obtain reversal-fault-tolerance, and will be applied again to construct some other circuits such as the circuits in Lemma 2.2.

Assume that an l-AKS circuit $\mathcal{C}'$ is constructed from a modified AKS circuit $\mathcal{C}$ by applying $2l$-input and $4l$-depth odd-even transposition circuits. In the AKS tree for $\mathcal{C}$, each node contains a set of registers R. The **l-AKS tree** for the l-AKS circuit $\mathcal{C}'$ is constructed from the AKS tree for $\mathcal{C}$ with each register $r \in R$ replaced by the block of registers corresponding to r during the construction of $\mathcal{C}'$ from $\mathcal{C}$. The **capacity** of an l-AKS tree node X for $\mathcal{C}'$ is defined to be the maximum number of registers (not the maximum number of blocks) allowed to be contained in X, which is equal to l times the capacity of the corresponding AKS tree node for $\mathcal{C}$.

In all of our applications of the l-AKS circuit in subsequent sections, we do not run the l-AKS circuit all the way to completion. Instead, we stop an m-input AKS circuit immediately before the first stage where the capacity of the root for the l-AKS tree is strictly less than $\sqrt{m}$. This guarantees that the capacity of any node in the l-AKS tree is at least $\sqrt{m}$. For ease of reference, we call such a circuit a **partial l-AKS circuit**. Moreover, we only need partial l-AKS circuits with $l \leq m^{\frac{1}{8}}$ (see Theorem 2.1). This means that the corresponding capacity of each AKS tree node (as opposed to an l-AKS tree node) is at least $m^{\frac{3}{8}}$, and we do not have to worry about the case where the capacity of a certain node is too small.

2.3 The Main Theorem

In this section, we prove the main theorem of this paper, which implies that the l-AKS circuit has certain fault-tolerance properties. We first choose a few more parameters and introduce some notation that will be useful in the statement and proof of our main theorem.

As in [16], we choose

$$\mu = \frac{1}{36}. \tag{11}$$

Instead of using the parameter δ as in [16], we use a parameter σ such that

$$\phi = 65 \left(\frac{4\sigma A}{\nu(1 - \frac{1}{2A})} \right)^2. \tag{12}$$

In a certain sense, our parameter σ corresponds to the parameter $\frac{1}{\delta}$ in [16]. As we have said in Section 2.2.1, we will only prove that a sufficiently large ϕ is good for our purposes, and we will not specify a particular choice of ϕ. As a consequence, we will not specify the choice of σ either. For now, we merely assume that $\sigma > 1$, which can be ensured by a sufficiently large choice of the constant ϕ in equation 12.

As in [16], we assign each m-input l-AKS tree node a **natural interval** as follows: the natural interval of the root is $[1, m]$; if the natural interval of a node X is $[\alpha, \beta]$, then the natural intervals of the left and right children of X are the left and right halves of $[\alpha, \beta]$, respectively. Intuitively, when a permutation of $\{1, \ldots, m\}$ is input to the l-AKS circuit, the natural interval of a node represents the range of numbers that the registers in the node "should" contain. The following concepts of content, strangeness, and potential are all dependent on which permutation is input to the circuit and which level (time) of the circuit we are interested in, but we do not include the permutation or time as part of the notation since we will only focus on a fixed input permutation in the proof of Theorem 2.1 and since the time will be clear from the context whenever we use these concepts. We define $c(r)$, the **content** of a register r at time t, as the input contained in r at time t. We also define $s(r)$, the **strangeness** of $c(r)$ (or of r at time t), to be the number of levels that $c(r)$ needs to move upward in the l-AKS tree from $c(r)$'s current node to the first node whose natural interval contains $c(r)$. Equivalently, we say that $c(r)$ (or r at time t) is $s(r)$-strange.

Given any constant $\sigma > 1$, we define the potential of a register r as:

$$P_\sigma(r) = \begin{cases} \sigma^{s(r)-1} & \text{if } s(r) \geq 1 \\ 0 & \text{otherwise.} \end{cases}$$

We define the potential of any set of registers R as:

$$P_\sigma(R) = \sum_{r \in R} P_\sigma(r).$$

In particular, the potential of a node X in the l-AKS tree is

$$P_\sigma(X) = \sum_{r \in X} P_\sigma(r).$$

As far as we know, potential functions have not been used in any previous analysis of the AKS circuit. A similar potential function was used in [14] to prove certain fault-tolerance properties of the multibutterfly circuit for routing. Unfortunately, our use of the potential function here is much more complex than that in [14]. The next theorem provides an upper bound on the number of strange items and as such is analogous to inequality 2 in [16]. Recall that, as discussed in Section 2.2, the capacity of an l-AKS tree node is the number of registers (not the number of blocks) in the node.

Theorem 2.1 *Under both the passive and the reversal fault models, for any $l \leq m^{\frac{1}{8}}$, if ϕ is a sufficiently large constant and $\rho > 0$ is less than a sufficiently small constant, then a randomly faulty m-input partial l-AKS circuit satisfies the following inequality with probability at least $1 - \rho^{\Theta(l \log m)}$: For all input permutations and all nodes X in the l-AKS tree,*

$$P_\sigma(X) \leq \mu \operatorname{cap}(X). \tag{13}$$

In the theorem, ρ is assumed to be less than a sufficiently small constant, say, ρ_0. The constant ρ_0 and the constant behind the Θ-notation in the theorem are both dependent on ϕ and σ. Most importantly, however, ρ is not necessarily a constant even though ρ is upper-bounded by the constant ρ_0, and the constant behind the Θ-notation of the theorem is independent of ρ. It should be mentioned that ρ will be quite small in all of our applications of the theorem and its corollary (see Corollary 2.1).

To prove the theorem, we first need to prove a few lemmas. We define a circuit $\mathcal{N}$ (possibly containing faulty comparators) to be a **Δ-approximate-sorting** circuit if, on all the possible input permutations, $\mathcal{N}$ outputs every item to within Δ positions of its correct position.

Lemma 2.1 *Under both the passive and the reversal fault models, for any constant $\vartheta > 0$, when ρ is less than a sufficiently small constant (depending on ϑ), a randomly faulty $2l$-input, $4l$-depth odd-even transposition circuit is a ϑl-approximate-sorting circuit with probability at least $1 - \rho^{\Theta(l)}$.*

Proof: We will only present the proof for reversal faults. The same proof is also valid for passive faults. In fact, for passive faults, our proof technique can be used to prove that the odd-even transposition circuit is indeed a $(\rho, \rho^{\Theta(l)})$-passive-fault-tolerant sorting circuit rather than a $(\rho, \rho^{\Theta(l)})$-passive-fault-tolerant ϑl-approximate-sorting circuit. Throughout the proof, we will assume that ρ is less than a sufficiently small constant depending on ϑ. We do not know if a similar result can be proved for ρ near $\frac{1}{2}$.

Let $\mathcal{C}$ be the odd-even transposition circuit described in Lemma 2.1, and let $\mathcal{C}^*$ be a randomly generated faulty version of $\mathcal{C}$. By the 0-1 principle, we only need to show that

$$\Pr(\exists \text{ 0-1 sequence } s \text{ such that } \mathcal{C}^* \text{ does not } \vartheta l\text{-approximate-sort } s) \leq \rho^{\Theta(l)}.$$

Notice that the total number of possible 0-1 input sequences to $\mathcal{C}$ is at most 2^{2l}. Hence, when ρ is less than a sufficiently small constant, to prove the above inequality, we only need to prove that for any fixed 0-1 input sequence s,

$$\Pr(\mathcal{C}^* \text{ does not } \vartheta l\text{-approximate-sort } s) \leq \rho^{\Theta(l)}. \tag{14}$$

In what follows, we will prove inequality 14 for a fixed s. Assuming that $\mathcal{C}^*$ does not ϑl-approximate-sort s, we prove that the behavior of the comparators in $\mathcal{C}^*$ satisfies a certain condition that is only satisfied with probability upper bounded by $\rho^{\Theta(l)}$. Without loss of generality, we will assume that on input sequence s, $\mathcal{C}^*$ outputs a 0 at least ϑl away from its correct position. (This assumption only affects the probability bound in inequality 14 by at most a factor of 2.)

Let k be the number of 0s in sequence s. We label the $2l$ registers in $\mathcal{C}$ as $r_1, \ldots, r_{2l}$, from top to bottom. We focus on the positions of the 0s at each level of $\mathcal{C}^*$ as the 0s and 1s move forward through $\mathcal{C}^*$. As the 0s move forward through $\mathcal{C}^*$, they gradually move upward in the 1s. In particular, a 0 in r_i that is correctly compared to a 1 in r_{i-1} at level t will move to r_{i-1}. Intuitively, if most of the comparators involved work correctly, 0s will move upward as they move forward. The problem in analyzing the movement of the 0s, however, is that they can block each other's upward movement. In particular, if one 0 moves the wrong way, it can cause a ripple effect much like a multicar collision on a highway. In the process of randomly generating $\mathcal{C}^*$ from $\mathcal{C}$, each of the comparators in $\mathcal{C}$ can be faulty with probability up to ρ. Hence, there are likely to be many such collisions. In addition to slowing things down, such collisions also introduce dependence issues in the analysis of the probabilistic movement of the 0s.

In order to get around these difficulties, we model the moves made by the 0s with a $k \times 4l$ matrix $A = (a_{i,j})$ of random biased coins. In particular, $a_{i,j} = H$ with probability at least $1 - \rho$ and $a_{i,j} = T$ with probability at most ρ. The coin at $a_{i,j}$ will be used to determine whether or not the comparator entered by the ith 0 at level j is faulty. (We number the 0s, starting at 1, from the top to the bottom at the outset, and we never alter the relative order of the 0s in $\mathcal{C}^*$.) Note that if two 0s enter the same comparator, the two associated coin flips could conflict in determining whether or not the comparator is faulty. However, we can assume that comparisons between two 0s are resolved according to the initial ordering of the 0s; we do not need to refer to the coin flips in such a case. Note that matrix A completely determines the behavior of $\mathcal{C}^*$ on the fixed 0-1 sequence s.

If at level t the ith 0 is compared to a 1 above, then the 0 moves upward one position if and only if $a_{i,t} = H$. If at level t the ith 0 is compared to a 1 below, then the 0 moves downward if and only if $a_{i,t} = T$. If at level t the ith 0 is compared to a 0 above (i.e., if it is blocked from above by the $(i-1)$th 0), then the ith 0 stays in the same register, and we change the value of $a_{i,t}$ to Z without checking to see whether $a_{i,t} = H$. If at level t the ith 0 is compared to a 0 below (i.e., if it is blocked from below by the $(i+1)$th 0), then the ith 0 stays in the same register and we change the value of $a_{i,t}$ to Z' without checking to see whether $a_{i,t} = H$.

After these modifications, matrix A now contains some Zs and Z's. Call the new matrix $A^* = (a^*_{i,j})$. Note that A^* completely determines the functionality of $\mathcal{C}^*$ on the fixed s and vice versa. This fact makes it possible for us to prove inequality 14 by analyzing A^*.

Define t_k to be the last level where the kth 0 was blocked by the $(k-1)$th 0. In other words, t_k is the maximum integer such that $a^*_{k,t_k} = Z$. Next, define t_{k-1} to be the last level where the $(k-1)$th 0 was blocked by the $(k-2)$th 0 strictly before level t_k. In other words, t_{k-1} is the largest integer such that $t_{k-1} < t_k$ and $a^*_{k-1,t_{k-1}} = Z$. Proceeding in a similar fashion, for $j = k-2, k-3, \ldots, 2$, define t_j to be the largest integer such that $t_j < t_{j+1}$ and $a^*_{j,t_j} = Z$. (It may be that $a^*_{j,t} \neq Z$ for all $t < t_{j+1}$, in which case we set $t_j = t_{j-1} = \cdots = t_1 = 0$.) If $t_2 = 0$, let $t_1 = 0$; if $t_2 > 0$, let t_1 be the largest integer such that $t_1 < t_2$ and the first 0 is located at r_1 immediately before level t_1 (if the first 0 never reaches r_1 strictly before level t_2, then set $t_1 = 0$).

Let S denote the string of coins

$$a_{1,t_1+1}a_{1,t_1+2}\cdots a_{1,t_2-1}a_{2,t_2+1}a_{2,t_2+2}\cdots a_{2,t_3-1}\cdots a_{k,t_k+1}a_{k,t_k+2}\cdots a_{k,4l}.$$

Let n_H denote the number of heads in S and n_T denote the number of tails in S. It is easy to see that S contains $4l - k - t_1 + 1$ coins, which implies that

$$n_T + n_H = 4l - t_1 - k + 1. \tag{15}$$

Roughly speaking, the number of upward moves of the kth 0 is given by $n_T - n_H$. However, this bound is not accurate because of **boundary effects** (i.e., caused by the 0s piling up against r_1). To be more precise, we analyze the movement of the 0s by considering two cases.

Case 1: $t_j > 0$ for all j such that $1 \le j \le k$. In this case, the first 0 is received at r_1 immediately before level t_1. On the other hand, the total number of downward moves corresponding to S is at most n_T. Hence, at the end of $\mathcal{C}^*$, the kth 0 is at most n_T positions away from r_k. By our assumption that a 0 is output at least ϑl away from its correct position, the kth 0 is output at least ϑl away from its correct position r_k. Hence,

$$n_T \ge \vartheta l.$$

Case 2: $t_j = 0$ for some j such that $1 \le j \le k$. By definition,

$$t_1 = 0. \tag{16}$$

In this case, when analyzing the upward moves of 0s corresponding to S, there is no boundary effect to consider. Therefore, the number of upward moves is given by $n_H - n_T$. Since the kth 0 can initially be at most $2l - k$ positions away from r_k, and since the kth 0 is output at least ϑl away from its correct position r_k, we conclude that

$$2l - k - n_H + n_T \ge \vartheta l.$$

Adding this inequality to equation 15 and using equation 16, we find that

$$n_T \ge \frac{(2+\vartheta)}{2} l \ge \vartheta l,$$

where we have assumed that $\vartheta \le 2$ since there is nothing to prove for $\vartheta > 2$.

In both Case 1 and Case 2, we have proven that

$$n_T \ge \vartheta l. \tag{17}$$

We next show that for a random matrix A, the probability that A^* contains a sequence S such that inequality 17 holds is at most $\rho^{\Theta(l)}$.

Let us define $a_{i,j}$ to be **next to $a_{u,v}$ in A** if and only if: (i) i is equal to u or $u+1$, and (ii) j is equal to v or $v+1$. According to the construction of S, the second element of S is next to the first element of S in A, the third element of S is next to the second element of S in A, and so on. Hence, when the location of the ith element of S is given in A, there are at most 3 ways for the $(i+1)$th element in S to be located in A. In addition, the number of ways that the first element of S is located in A is upper bounded by $k \cdot 4l \le 2l \cdot 4l \le 8l^2$. On the other hand, $|S| \le 4l$. Hence, the number of ways of choosing the location of S in A is at most

$$8l^2 \cdot 3^{4l}. \tag{18}$$

By a standard Chernoff bound argument [3], when the location of S in A is given, the probability that inequality 17 holds is at most

$$\rho^{\Theta(l)} \tag{19}$$

for ρ less than a sufficiently small constant (depending on ϑ). Multiplying the bounds of inequalities 18 and 19, and setting ρ to less than a sufficiently small constant, we find that inequality 17 holds with probability at most $\rho^{\Theta(l)}$. This completes the proof of inequality 14 as well as the proof of Lemma 2.1. ■

In the next lemma, the circuit $\mathcal{N}$ is the parallel union of s disjoint circuits, $\mathcal{N}_1, \ldots, \mathcal{N}_s$. Each $\mathcal{N}_i$ is constructed from a ϕ-divider by replacing each register with a block of l registers, and each comparator with a $2l$-input $4l$-depth odd-even transposition circuit. By definition, each of the ϕ-dividers is constructed from a d-regular bipartite expander, and thus has depth d, which is a constant depending on ϕ. Hence, the depth of each $\mathcal{N}_i$ is $4dl$, as is the depth of $\mathcal{N}$. A block or a register will be called a **bottom** (**top**) block or register in $\mathcal{N}$ if it is in the bottom (top) half of some $\mathcal{N}_i$. For a set of bottom (top) registers R, we use $N(R)$ to denote the set of top (bottom) registers that are connected to at least one register in R by some odd-even transposition circuit. In the next lemma, n_i denotes the number of inputs to $\mathcal{N}_i$, and n denotes the number of inputs to $\mathcal{N}$.

Lemma 2.2 *Let R be a fixed set of bottom (resp., top) registers and b be the number of blocks that contain at least one register in R. Under both the passive and the reversal fault models, when ϕ is large enough, a randomly faulty version of $\mathcal{N}$ has the following property with probability at least $1 - \rho^{\Theta(bl+|R|)}$: On all input permutations such that each $\mathcal{N}_i$ contains at most $\frac{49}{100}n_i$ inputs with rank at most $k \leq \frac{49}{100}n$ (resp., at least $k \geq \frac{51}{100}n$), if every register in R contains an output with rank at most k (resp., at least k), then at least $\frac{\phi}{4}|R|$ registers in $N(R)$ contain outputs with rank at most k (resp., at least k).*

Note that in the $1 - \rho^{\Theta(bl+|R|)}$ lower bound for the success probability claimed in the lemma, we could omit the $|R|$ term without affecting the meaning of the lemma since $bl \geq |R|$. However, we have chosen to include the $|R|$ term for ease of future applications.

Proof of Lemma 2.2: The two claims of the lemma are symmetric, and so we consider only the case where R is a set of bottom registers. We make use of the 0-1 principle. Suppose that each comparator is randomly set to be faulty (with probability ρ or less) ahead of time. We will show that with probability at least $1 - \rho^{\Theta(bl)}$, the resulting circuit has the following property: On all 0-1 input sequences with exactly k 0s such that each $\mathcal{N}_i$ contains at most $\frac{49}{100}n_i$ 0-inputs, if all of the registers in R contain a 0-output, then at least $\frac{\phi}{4}|R|$ registers in $N(R)$ contain a 0-output.

Let

$$\theta = \frac{1}{d\phi}. \tag{20}$$

We focus on the bottom blocks containing at least one register in R. We say that such a block is **dense** if it contains strictly more than $d\theta l$ registers in R; we say that such a block is **sparse** if it contains at least one but at most $d\theta l$ registers in R. Note that when $d\theta l < 1$, there is no sparse block. Let b_{i1} be the number of dense blocks in $\mathcal{N}_i$, and let b_{i2} be the number of sparse blocks in $\mathcal{N}_i$. We call a top (bottom) block B' a **neighboring block** of a bottom (top) block B if there is an odd-even transposition circuit connecting B and B' in $\mathcal{N}$. Note that each block has exactly d neighboring blocks because each of the corresponding ϕ-dividers has depth d. We call a block B **good** if all of the (no more than $d + d^2$) odd-even transposition circuits associated with B or associated with any of the d neighboring blocks of B are θl-approximate-sorting circuits. Let b'_{i1} be the number of good dense blocks in $\mathcal{N}_i$, and b'_{i2} be the number of good sparse blocks in $\mathcal{N}_i$.

When a good block B is connected to a top block B' by a θl-approximate-sorting circuit $\mathcal{M}$, the correctness of the following two simple observations is straightforward:

Observation 1. At most θl 0s can be moved from B' to B through $\mathcal{M}$.

Observation 2. If B contains at least one 0 at the end of $\mathcal{M}$, then B' contains at least $(1-\theta)l$ 0s at the end of $\mathcal{M}$.

The goal of our proof of Lemma 2.2 is to find a large number of 0s in $N(R)$ in comparison with the number of 0s in R. This goal will be achieved as follows. From observation 1 above, each good dense block contains 0s throughout the circuit. Therefore, we can hope to use the expansion property of the ϕ-divider to find many 0s in the neighboring blocks of a good dense block. From observation 2, for each good sparse block, its unique neighboring block at the end of $\mathcal{N}$ contains many 0s, compared with the number of 0s contained in the sparse block. In particular, we prove the lemma by considering the following two cases. (Recall that s denotes the total number of disjoint parallel subcircuits forming $\mathcal{N}$.)

Case 1: $\sum_{1\le i\le s} b_{i1} \ge \frac{1}{\phi}\sum_{1\le i\le s} b_{i2}$.

By Lemma 2.1, each dense block is good with probability at least $1-(d+d^2)\rho^{\Theta(l)} = 1-\rho^{\Theta(l)}$ provided that ρ is less than a sufficiently small constant (specifically, we can assume ρ to be small compared with $d+d^2$). A standard application of the Chernoff bound [3] now implies that when ρ is less than a sufficiently small constant, the following inequality holds with probability at least $1-\rho^{\Theta(l\sum_{1\le i\le s} b_{i1})}$:

$$\sum_{1\le i\le s} b'_{i1} \ge \frac{2}{3}\sum_{1\le i\le s} b_{i1}. \tag{21}$$

By the assumption of Case 1, we have $\sum_{1\le i\le s} b_{i1} \ge \frac{1}{\phi+1}\sum_{1\le i\le s}(b_{i1}+b_{i2}) = \frac{b}{\phi+1}$. Hence, inequality 21 holds with probability at least $1-\rho^{\Theta(bl)} = 1-\rho^{\Theta(bl+|R|)}$ (where the constant behind the Θ-notation is allowed to depend on ϕ, d, and θ). In what follows, we need only show that at least $\frac{\phi}{4}|R|$ registers in $N(R)$ contain a 0 whenever inequality 21 holds.

Consider any good dense block B_{i1} in $\mathcal{N}_i$. Since B_{i1} contains more 0-outputs than could possibly come from its d neighboring blocks through the d θl-approximate-sorting circuits, B_{i1} contains 0s throughout all the levels of $\mathcal{N}_i$. Hence, by observation 2, each neighboring block of B_{i1} contains at least $(1-\theta)l$ 0s just after being compared with B_{i1}. Moreover, by observation 1, each of these neighboring blocks of B_{i1} contains at least $(1-\theta d)l$ 0s at the end of $\mathcal{N}$ since such a block may lose at most $(d-1)\theta l$ 0s through the later $d-1$ or fewer θl-approximate-sorting circuits.

Now, assume for the purposes of contradiction that $b'_{i1} \ge \frac{n_i}{2l(\phi+1)}$ for some i. Then, we can choose $\frac{n_i}{2l(\phi+1)}$ good dense blocks in $\mathcal{N}_i$. By the expansion property of the ϕ-divider, these blocks have at least $\frac{\phi n_i}{2l(\phi+1)}$ neighboring blocks in the top half of $\mathcal{N}_i$. By the discussion of the preceding paragraph, each of these neighboring blocks has at least $(1-\theta d)l$ 0-outputs at the end of $\mathcal{N}_i$. Thus, the number of 0-outputs of $\mathcal{N}_i$ is at least

$$\frac{\phi n_i}{2l(\phi+1)}(1-\theta d)l = \frac{\phi-1}{2(\phi+1)}n_i, \tag{22}$$

where we have used equality 20. When ϕ is large enough, the quantity in equation 22 is strictly larger than $\frac{49}{100}n_i$, which is larger than the number of 0-inputs to $\mathcal{N}_i$. This is a contradiction.

Hence, we conclude that for all i

$$b'_{i1} < \frac{n_i}{2l(\phi+1)}. \tag{23}$$

By inequality 23 and the fact that each $\mathcal{N}_i$ is constructed from a $(\frac{1}{\phi+1}, \phi)$-expander, all of the $\sum_{1\le i\le s} b'_{i1}$ good dense blocks have at least $\phi\sum_{1\le i\le s} b'_{i1}$ neighboring blocks of top registers in $N(R)$. By the argument used for deriving inequality 22, we know that the number of 0s contained in these $\phi\sum_{1\le i\le s} b'_{i1}$ top blocks of registers in $N(R)$ is at least

$$\begin{array}{lll}
 & \phi\sum_{1\le i\le s} b'_{i1}(1-\theta d)l & \\
= & (\phi-1)\sum_{1\le i\le s} b'_{i1}l & \text{(by equality 20)} \\
\ge & \frac{2}{3}(\phi-1)\sum_{1\le i\le s} b_{i1}l & \text{(by inequality 21)} \\
\ge & \frac{1}{3}(\phi-1)\sum_{1\le i\le s}(b_{i1}l + \frac{b_{i2}l}{\phi}) & \text{(by the assumption of Case 1)} \\
= & \frac{1}{3}(\phi-1)\sum_{1\le i\le s}(b_{i1}l + b_{i2}d\theta l) & \text{(by equality 20)} \\
\ge & \frac{\phi-1}{3}|R| & \text{(by the definitions of } b_{i1} \text{ and } b_{i2}\text{)} \\
\ge & \frac{\phi}{4}|R| & \text{(for } \phi \text{ sufficiently large).}
\end{array}$$

Case 2: $\sum_{1\le i\le s} b_{i1} < \frac{1}{\phi}\sum_{1\le i\le s} b_{i2}$.

By Lemma 2.1 and a standard Chernoff bound argument [3], we know that the following inequality holds with probability at least $1-\rho^{\Theta(\sum_{1\le i\le s} b_{i2}l)} = 1-\rho^{\Theta(bl+|R|)}$ provided that ρ is sufficiently small:

$$\sum_{1\le i\le s} b'_{i2} \ge \frac{2}{3}\sum_{1\le i\le s} b_{i2}. \tag{24}$$

Next, we show that at least $\frac{\phi}{4}|R|$ registers in $N(R)$ contain a 0 provided that inequality 24 holds.

By observation 2, for each good sparse block, its unique neighboring block at the end of $\mathcal{N}$ contains at least $(1-\theta)l$ 0s. Moreover, since the dividers are constructed from d-regular bipartite expanders, different blocks have different neighboring blocks at the end of $\mathcal{N}$. Hence, the number of 0-outputs contained in $N(R)$ is at least

$$\begin{array}{lll}
 & \sum_{1\le i\le s} b'_{i2}(1-\theta)l & \\
\ge & \frac{2}{3}\sum_{1\le i\le s} b_{i2}(1-\theta)l & \text{(by inequality 24)} \\
\ge & \frac{1-\theta}{3}\sum_{1\le i\le s}(b_{i2}l + b_{i1}\phi l) & \text{(by the assumption of Case 2)} \\
= & \frac{(1-\theta)\phi}{3}\sum_{1\le i\le s}(b_{i1}l + b_{i2}d\theta l) & \text{(by equality 20)} \\
\ge & \frac{(1-\theta)\phi}{3}|R| & \text{(by the definitions of } b_{i1} \text{ and } b_{i2}\text{)} \\
\ge & \frac{\phi}{4}|R| & \text{(for } \phi \text{ sufficiently large).}
\end{array}$$

■

Lemma 2.3 *Let $b_{i(1)} \leq \cdots \leq b_{i(\frac{p}{4})}$ be a subsequence of a positive non-decreasing sequence $b_1 \leq \cdots \leq b_p$. Then, there exists an integer $s \geq \frac{p}{8}$ such that*

$$\sum_{1 \leq j \leq s} b_{i(j)} \geq \frac{1}{8} \sum_{1 \leq j \leq i(s)} b_j.$$

Proof: Take the minimum s such that

$$i(s) \leq 8\,(s - \frac{p}{8}). \tag{25}$$

(Such an s exists because $s = \frac{p}{4}$ satisfies inequality 25.) By the minimality of s,

$$i(t) > 8\, \left(t - \frac{p}{8}\right) \tag{26}$$

for all t such that $\frac{p}{8} + 1 \leq t \leq s - 1$. By the monotonicity of the sequence $b_1 \leq \cdots \leq b_p$ and inequality 26,

$$b_{i(t)} \geq \frac{1}{8} \sum_{8(t - \frac{p}{8} - 1) < j \leq 8(t - \frac{p}{8})} b_j \tag{27}$$

for all t such that $\frac{p}{8} + 1 \leq t \leq s - 1$. By inequality 25, we have $i(s) - 8\,(s - \frac{p}{8} - 1) \leq 8$. Hence,

$$b_{i(s)} \geq \frac{1}{8} \sum_{8(s - \frac{p}{8} - 1) < j \leq i(s)} b_j. \tag{28}$$

Adding inequality 28 with inequalities 27 (i.e., for all t), we obtain

$$\sum_{\frac{p}{8} < j \leq s} b_{i(j)} \geq \frac{1}{8} \sum_{1 \leq j \leq i(s)} b_j,$$

which is actually stronger than the claimed inequality. ∎

Proof of Theorem 2.1: We focus on a particular faulty partial l-AKS circuit that violates inequality 13 on a particular input permutation Π, and we prove that the faulty circuit has certain properties that can be satisfied by a randomly generated faulty partial l-AKS circuit with probability at most $\rho^{\Theta(l \log m)}$.

We choose the first stage t during which inequality 13 is not satisfied at a certain node X. By the minimality of t, we have

$$P_\sigma(Y) \leq \mu\, \mathrm{cap}(Y) \tag{29}$$

for any l-AKS tree node Y before stage t. Let Y_i denote the number of i-strange items in node Y. Then, the potential function at node Y can be written as

$$P_\sigma(Y) = \sum_{k \geq 1} \sum_{r \in Y,\, s(r) = k} \sigma^{k-1} = \sum_{k \geq 1} Y_k \sigma^{k-1}. \tag{30}$$

Therefore, inequality 29 can be rewritten as

$$\sum_{i \geq 1} Y_i \sigma^{i-1} \leq \mu \operatorname{cap}(Y).$$

Thus,

$$\sum_{i \geq j} Y_i \sigma^{i-1} \leq \mu \operatorname{cap}(Y)$$

for all $j \geq 1$. Since $\sigma > 1$, the previous inequality implies that

$$\frac{\sum_{i \geq j} Y_i}{\operatorname{cap}(Y)} \leq \left(\frac{1}{\sigma}\right)^{j-1} \mu \tag{31}$$

for all $j \geq 1$. Inequality 31 gives an upper bound on the ratio of the number of items at node Y with strangeness j or more to the capacity of Y, and it will be useful when we upper bound the number of strange items inductively. (It is analogous to inequality 2 in [16].)

On the other hand, by the assumption that $P_\sigma(X) > \mu \operatorname{cap}(X)$ and equation 30,

$$\sum_{k \geq 1} X_k \sigma^{k-1} > \mu \operatorname{cap}(X),$$

where X_k denotes the number of k-strange items in X. Therefore, there exists an integer $k \geq 1$ such that

$$X_k > \left(\frac{1}{2}\right)^k \left(\frac{1}{\sigma}\right)^{k-1} \mu \operatorname{cap}(X). \tag{32}$$

We choose the minimum k that satisfies inequality 32 and analyze how these X_k k-strange items are misplaced into X. By doing so, we derive some necessary properties of the faulty l-AKS circuit. Then, we prove that these properties can be satisfied with probability at most $\rho^{\Theta(l \log m)}$. We will consider two cases: $k = 1$ and $k > 1$. The case $k = 1$ is the hard case in [16] and proceeds without much additional work once we have Lemma 2.2. Unfortunately, the case $k > 1$ requires much more work than its fault-free counterpart in [16].

Case 1: $k = 1$.

At the beginning of the first stage, all items are either at the root or the cold storage, and nothing is strange. Hence, our choice of t guarantees $t \geq 2$. We now trace back how the X_1 1-strange items at node X arrived there from P, C, and C', the parent and two children of X, respectively. It is easy to see that

$$\begin{aligned} X_1 &= |\{i : i \text{ is 1-strange in } X, \text{ and } i \text{ came from } C \text{ or } C'\}| \\ &\quad + |\{i : i \text{ is 1-strange in } X, \text{ and } i \text{ came from } P\}|. \end{aligned} \tag{33}$$

Since a 1-strange item in X is 2-strange in either C or C', we can upper bound the first term of equation 33 by $C_2 + C'_2$, where C_2 and C'_2 denote the number of 2-strange items in C and C' respectively. By inequality 31, we have

$$C_2 \leq \frac{1}{\sigma} \mu \operatorname{cap}(C)$$

and

$$C_2' \leq \frac{1}{\sigma}\mu \operatorname{cap}(C').$$

Hence, the first term of equation 33 is at most

$$\frac{1}{\sigma}\mu \operatorname{cap}(C) + \frac{1}{\sigma}\mu \operatorname{cap}(C') = \frac{2}{\sigma}\mu \operatorname{cap}(X)\frac{A}{\nu}. \tag{34}$$

In what follows, we will use Paterson's argument aided by Lemma 2.2 to upper bound the second term of equation 33. This is fairly complicated because we have to deal with items that become strange for the first time (in other words, some items may not be strange in P but may be strange in X). An item can be misplaced into X due to one of the following two reasons: (i) the first round divider at P may not divide all the items into the correct halves, and (ii) P may contain too many items that should go to X's sibling, X'.

We assume that the number of items that are misplaced into X due to reason (i) above is equal to

$$\zeta|P| \leq \zeta\frac{1}{2A\nu}\operatorname{cap}(X). \tag{35}$$

Here, we use $|P|$, the number of registers in P, instead of $\operatorname{cap}(P)$ since P may not be full. We will apply Lemma 2.2 where $|P|$, instead of $\operatorname{cap}(P)$, will be a parameter.

We next upper bound the number of items that are misplaced into X due to reason (ii) above. Let V be the set of all items of strangeness 0 with respect to X' (some of these items may not be located in node X'). Following the terminology of Paterson, the "natural" positions for V correspond to the subtree rooted at X' plus one half of P, one-eighth of P's grandparent, and so on. (We can assume an infinite chain of ancestors for this argument due to our use of the cold storage. Note that at stage $t-1$ (1) X' is empty, and (2) by Claim 2.1, the levels of the l-AKS tree are alternatively empty and full above X'.) Ideally, if all items in V are in V's "natural" positions, then P cannot contain too many items that should go to X'. In reality, some of the items in V may not be in V's "natural" positions. In such a case, some of V's "natural" positions are occupied by items not in V. In particular, it is not hard to see that the number of items that belong to V but will be forced into X due to the capacity constraint at X' is equal to the number of items that are not in V but are occupying V's "natural" positions not in P. We next upper bound the former quantity by giving an upper bound for the latter quantity.

Clearly, an item not in V is 2 or more strange in a child of X', 4 or more strange in a greatgrandchild of X', and so on. Since inequality 31 holds before we find the first violation of inequality 13 at some stage t, a child of X' can contain at most

$$\mu\left(\frac{1}{\sigma}\right)\frac{A}{\nu}\operatorname{cap}(X)$$

items not in V, a greatgrandchild of X' can contain at most

$$\mu\left(\frac{1}{\sigma}\right)^3\frac{A^3}{\nu}\operatorname{cap}(X)$$

items not in V, and so on. Hence, the total number of items not in V but occupying V's "natural" positions (strictly) below X' is thus upper bounded by

$$\begin{aligned} & 2\mu\left(\frac{1}{\sigma}\right)\frac{A}{\nu}\operatorname{cap}(X) + 8\mu\left(\frac{1}{\sigma}\right)^3\frac{A^3}{\nu}\operatorname{cap}(X) + 32\mu\left(\frac{1}{\sigma}\right)^5\frac{A^5}{\nu}\operatorname{cap}(X) + \cdots \\ \leq\ & \frac{2\mu A\operatorname{cap}(X)}{\sigma\nu(1-4(\frac{A}{\sigma})^2)}. \end{aligned} \tag{36}$$

On the other hand, V's "natural" positions strictly above P may be fully occupied by items not in V, but the number of such positions is at most

$$\frac{\operatorname{cap}(X)}{2^3\nu A^3} + \frac{\operatorname{cap}(X)}{2^5\nu A^5} + \frac{\operatorname{cap}(X)}{2^7\nu A^7} + \cdots = \frac{\operatorname{cap}(X)}{\nu(8A^3-2A)}. \tag{37}$$

By the argument of the preceding paragraph, the number of items in P that should go to X' but will be forced into X due to capacity constraint is upper bounded by the the sum of the quantities in equations 36 and 37:

$$\frac{2\mu A\operatorname{cap}(X)}{\nu\sigma(1-4\frac{A^2}{\sigma^2})} + \frac{\operatorname{cap}(X)}{\nu(8A^3-2A)}. \tag{38}$$

Now, adding the quantities in equations 34, 35, and 38, we obtain an upper bound for X_1 in equation 33:

$$X_1 \leq \frac{2\mu A\operatorname{cap}(X)}{\sigma\nu} + \frac{2\mu A\operatorname{cap}(X)}{\nu\sigma(1-4\frac{A^2}{\sigma^2})} + \frac{\operatorname{cap}(X)}{\nu(8A^3-2A)} + \frac{\zeta\operatorname{cap}(X)}{2A\nu}.$$

(We remark that a corresponding formula in [16] contains a term of $\frac{\mu\operatorname{cap}(X)}{\nu A}$, which does not appear in our formula. Such a difference occurs since we count the number of items with strangeness exactly 1, whereas Paterson counted the number of items with strangeness at least 1.) By inequality 32,

$$X_1 > \frac{\mu\operatorname{cap}(X)}{2}.$$

Combining the last two inequalities, we obtain

$$\frac{2\mu A}{\sigma\nu} + \frac{2\mu A}{\nu\sigma(1-4\frac{A^2}{\sigma^2})} + \frac{1}{\nu(8A^3-2A)} + \frac{\zeta}{2\nu A} > \frac{\mu}{2}. \tag{39}$$

Hence,

$$\zeta > \nu\mu A - \frac{4\mu A^2}{\sigma}\left(1 + \frac{1}{1-\frac{4A^2}{\sigma^2}}\right) - \frac{1}{4A^2-1}. \tag{40}$$

By choosing ϕ sufficiently large, we can ensure that

$$\frac{4\mu A^2}{\sigma}\left(1 + \frac{1}{1-\frac{4A^2}{\sigma^2}}\right) < \frac{1}{576}.$$

By equations 1 and 11 and inequality 40,

$$\zeta > \frac{43}{576} - \frac{1}{576} - \frac{1}{35} > \frac{4}{100}.$$

By the choice of ζ in inequality 35, the preceding inequality implies that the partitioner at P outputs at least $\frac{4}{100}|P|$ of the items into the wrong half. Hence, at the end of the first round divider at P, at least $\frac{4}{100}|P|$ items are output into the wrong half. Without loss of generality, we assume that at the end of the first round divider, at least $\frac{4}{100}|P|$ items that belong to the top half are output to the bottom half. Among the $\frac{4}{100}|P|$ or more items that belong to the top half but are output to the bottom half, at most $\frac{1}{100}|P|$ have ranks greater than $\frac{49}{100}|P|$, and all of the other at least $\frac{4}{100}|P| - \frac{1}{100}|P| \geq \frac{|P|}{100}$ items have ranks $\frac{49}{100}|P|$ or less. Let us define a pair (P, R) to be a **bad pair** if R is a set of bottom registers at the end of the first round divider at P such that $|R| = \frac{|P|}{100}$ and R contains output items with ranks $\frac{49}{100}|P|$ or less at the end of the first round divider at P. The above arguments show that the faulty partial l-AKS circuit violating inequality 13 on input permutation Π has a bad pair (P, R). So to complete our analysis of Case 1, we need only show that the probability there exists a bad pair is at most $\rho^{\Theta(l \log m)}$.

For any fixed pair (P, R) such that $|R| = \frac{|P|}{100}$, by Lemma 2.2 with $s = 1$, $k = \frac{49}{100}|P|$, and ϕ sufficiently large so that $|R| + \frac{\phi}{4}|R| > k$,

$$\Pr((P, R) \text{ is a bad pair}) \leq \rho^{\Theta(|R|)} = \rho^{\Theta(|P|)}. \tag{41}$$

Hence,

$$\begin{aligned}
& \Pr(\exists \text{ a bad pair } (P, R)) \\
\leq\ & \textstyle\sum_P \sum_{\substack{R \subset P \\ |R| = \frac{|P|}{100}}} \Pr((P, R) \text{ is a bad pair}) \\
\leq\ & \textstyle\sum_P \binom{|P|}{\frac{|P|}{100}} \rho^{\Theta(|P|)} && \text{(by inequality 41)} \\
\leq\ & \textstyle\sum_P 2^{|P|} \rho^{\Theta(|P|)} && \text{(since } \tbinom{x}{y} \leq 2^x\text{)} \\
\leq\ & \textstyle\sum_P \rho^{\Theta(|P|)} && \text{(for } \rho \text{ sufficiently small)} \\
\leq\ & O(m \log m) \rho^{\Theta(|P|)}, && (42)
\end{aligned}$$

where the last inequality holds because there are at most $O(m \log m)$ l-AKS tree nodes. On the other hand, since priority is given to the upward movement of registers in any l-AKS tree node and since X is not empty at stage t, P contains at least $\lambda \operatorname{cap}(P)$ registers at stage $t-1$. Therefore, $|P| \geq \lambda \operatorname{cap}(P) \geq \lambda\sqrt{m}$ (since any node in the l-AKS tree for the partial l-AKS circuit has capacity at least $\sqrt{m}$). Hence,

$$O(m \log m)\ \rho^{\Theta(|P|)} \leq O(m \log m) \rho^{\Theta(\lambda\sqrt{m})} \leq \rho^{\Theta(l \log m)}, \tag{43}$$

where the last inequality holds due to the assumption that $l \leq m^{\frac{1}{8}}$ and the fact that λ is a constant. Combining inequalities 42 and 43 completes the proof for Case 1.

Case 2: $k > 1$.

This case is much more complicated than the preceding one. The source of the difficulty is that all previously known analyses of the AKS circuit rely on the expansion property for arbitrarily small sets of registers and, as discussed in Section 2.1, such an expansion property cannot be preserved with high probability in the presence of faults. To get around this problem, we will trace back $k-1$ stages to see how the X_k k-strange items in X are eventually moved into X. Strange items can come either from below or above in the tree. In the former case, the items would be more strange one stage earlier, and we can apply inequality 31. In the latter case, if a good number of the comparators associated with the items work correctly, we will get a certain expansion property. Our hope is to show that even under the loss of the local expansion property for possibly many small sets of registers, with high probability the circuit maintains a certain global expansion property that ensures its functionality as a sorting circuit.

Since we have k-strange items in the l-AKS tree at the beginning of stage t, we have $t \geq k+1$ (otherwise all the nodes with depth more than k in the l-AKS tree would be empty at stage t and no item could be k-strange). In what follows, we will trace backward $k-1$ stages and see how these X_k k-strange items are misplaced into X. We will inductively define a sequence of sets $R_t, \ldots, R_{t-k+1}$ such that each R_{t-i} is a set of registers at the beginning of stage $t-i$ with strangeness at least $k-i$. For ease of notation, let $c(R)$ be the set of all the items contained in R for any set of registers R, and let

$$r_{t-i} = |R_{t-i}|. \tag{44}$$

Base step: Take R_t as the set of all X_k registers at the beginning of stage t that are k-strange in X. By definition, all registers in R_t have strangeness at least k.

Inductive step: Assuming that R_{t-i} has been defined for some $0 \leq i < k-1$, we now define R_{t-i-1}. For each register r in an l-AKS tree node X, $c(r)$ may come from either the parent of X or a child of X. In the former case, we say that $c(r)$ **comes from above**; in the latter case, we say that $c(r)$ **comes from below**. Let

$$\alpha_{t-i} = \frac{|\{\text{items in } c(R_{t-i}) \text{ that come from above}\}|}{r_{t-i}},$$

where r_{t-i} is defined in equation 44. Given R_{t-i}, a set of registers with strangeness at least $k-i$ at the beginning of stage $t-i$, we construct a set of registers R_{t-i-1} at the beginning of stage $t-i-1$ as follows.

Case (a): $\alpha_{t-i} \leq \frac{1}{2}$ (at most half of the items in R_{t-i} come from above). We simply choose R_{t-i-1} as the set of all r_{t-i} registers at the beginning of stage $t-i-1$ that contain an item in $c(R_{t-i})$.

Case (b): $\alpha_{t-i} > \frac{1}{2}$ (more than half of the items in R_{t-i} come from above). We focus on the $\alpha_{t-i}r_{t-i}$ items in $c(R_{t-i})$ that come from above. By the induction hypothesis, each of these $\alpha_{t-i}r_{t-i}$ items is at least $(k-i)$-strange in a register of R_{t-i}. Hence, each is either too small or too large for the node in which it currently resides. Without loss of generality, we assume that more than half of the items are too small for the nodes in which they currently reside. Let $rank_i$ be the maximum of these small items, and let $c(r)$ be any one of these small items. Let W be the AKS-tree node that contains $c(r)$ at stage $t-i-1$. By the choice of $c(r)$, $c(r)$

is moved from W to a child of W from stage $t-i-1$ to stage $t-i$. Hence, we immediately have

$$|W| \geq \lambda\,\mathrm{cap}(W). \tag{45}$$

By inequality 8, at the end of stage $t-i-1$, $c(r)$ cannot possibly be located in the regions $\overline{FL}$ or $\overline{FR}$ of W. In other words, at the end of stage $t-i-1$, $c(r)$ is located in the bottom half of the top ith round divider at W for some $i \leq 1 + \log \frac{1}{\lambda'} = 4$.

We choose the smallest $h \leq 4$ such that at least $\frac{1}{8}\alpha_{t-i}r_{t-i}$ of such $c(r)$'s are located in the bottom halves of the top hth round dividers at W's, each of which corresponds to a $c(r)$. (Note that different $c(r)$'s correspond to different W's and that there are at least $\frac{1}{2}\alpha_{t-i}r_{t-i}$ such $c(r)$'s.) We trace back where these $\frac{1}{8}\alpha_{t-i}r_{t-i}$ or more items are located at the end of the corresponding top hth round dividers. Let U_{t-i} be the set of registers containing these items at the end of the corresponding top hth round dividers. Let u'_{t-i} be the number of registers in $N(U_{t-i})$ (within the top hth round dividers at the corresponding W's) that contain an item less than or equal to $rank_i$ (the meaning of $N(R)$, where R denotes a set of registers in the bottom halves of a collection of dividers, can be found immediately before Lemma 2.2). Note that such registers are at least $(k-i-1)$-strange. If $u'_{t-i} \geq r_{t-i}$, let R_{t-i-1} be any set of registers at the beginning of stage $t-i-1$ that contain $\min\{\phi, \lfloor \frac{u'_{t-i}}{r_{t-i}} \rfloor\} r_{t-i}$ of the u'_{t-i} items. If $u'_{t-i} < r_{t-i}$, then we abandon everything that has been established in Case (b) and simply use the method of Case (a) to choose R_{t-i-1}.

Claim 2.3 *If $\mathcal{D}$ is the hth round divider at an AKS tree node W and $\mathcal{D}$ contains at least one register in U_{t-i}, then $\mathcal{D}$ has at least $\frac{\lambda'}{2}\mathrm{cap}(\mathrm{W})$ registers.*

Proof: Straightforward from inequality 45 and Claim 2.2. ■

By the induction hypothesis, the items in $c(R_{t-i})$ are at least $(k-i)$-strange in the registers in R_{t-i}. Since each register in R_{t-i-1} is at most one level higher in the l-AKS tree than its corresponding register in R_{t-i}, all of the registers in R_{t-i-1} are at least $(k-i-1)$-strange. This finishes our inductive construction of R_{t-i-1} from R_{t-i}.

For $0 \leq i \leq k-2$, let

$$\phi_{t-i} = \frac{r_{t-i-1}}{r_{t-i}}. \tag{46}$$

Claim 2.4 *For $0 \leq i \leq k-2$, ϕ_{t-i} is an integer in $[1, \phi]$.*

Proof: Straightforward from equation 46 and the construction of R_{t-i-1} from R_{t-i}. □

For $0 \leq i \leq k-2$, the strangeness of any item in R_{t-i-1} is at least its strangeness in R_{t-i} minus 1. So by counting all of the items from above or below, we have

$$P_\sigma(R_{t-i-1}) \geq \phi_{t-i}\sigma^{-1}P_\sigma(R_{t-i})$$

for all i, $0 \leq i \leq k-2$. If $\alpha_{t-i} \leq \frac{1}{2}$, then, by the fact that the items from below should be one more strange in R_{t-i-1} than in R_{t-i},

$$P_\sigma(R_{t-i-1}) \geq \frac{\sigma}{2}P_\sigma(R_{t-i})$$

for all i such that $0 \le i \le k-2$ and $\alpha_{t-i} \le \frac{1}{2}$. From the preceding pair of inequalities, we have

$$\begin{aligned} P_\sigma(R_{t-k+1}) &\ge P_\sigma(R_t) \prod_{0\le i\le k-2,\ \alpha_{t-i}\le\frac{1}{2}} \left(\frac{\sigma}{2}\right) \prod_{0\le i\le k-2,\ \alpha_{t-i}>\frac{1}{2}} \left(\frac{\phi_{t-i}}{\sigma}\right) \\ &\ge \mu \operatorname{cap}(X) \prod_{0\le i\le k-2,\ \alpha_{t-i}\le\frac{1}{2}} \left(\frac{\sigma}{2}\right) \prod_{0\le i\le k-2,\ \alpha_{t-i}>\frac{1}{2}} \left(\frac{\phi_{t-i}}{\sigma}\right). \end{aligned} \tag{47}$$

Since we start from a set of registers in node X and we move at most one level upward and one level downward in the l-AKS tree when constructing R_{t-i-1} from R_{t-i}, R_{t-k+1} is located within $k-1$ levels of X. Therefore, by equality 3, the total capacity of all nodes that can possibly contain registers in R_{t-k+1} is upper bounded by

$$\begin{aligned} &\operatorname{cap}(X)\ \nu^{-(k-1)} \left[(2A)^{k-1} + (2A)^{k-2} + \ldots + A + 1 + A^{-1} \ldots + A^{-(k-2)} + A^{-(k-1)}\right] \\ < \ &\operatorname{cap}(X)\ \left(\frac{2A}{\nu}\right)^{k-1} \frac{1}{1-\frac{1}{2A}}. \end{aligned} \tag{48}$$

By inequality 48 and the fact that inequality 13 is not yet violated at stage $t-k+1$,

$$P_\sigma(R_{t-k+1}) \le \mu\ \operatorname{cap}(X) \left(\frac{2A}{\nu}\right)^{k-1} \frac{1}{1-\frac{1}{2A}}. \tag{49}$$

Combining inequalities 47 and 49, we get

$$\prod_{0\le i\le k-2,\, \alpha_{t-i}\le\frac{1}{2}} \left(\frac{\sigma}{2}\right) \prod_{0\le i\le k-2,\, \alpha_{t-i}>\frac{1}{2}} \left(\frac{\phi_{t-i}}{\sigma}\right) \le \left(\frac{2A}{\nu}\right)^{k-1} \frac{1}{1-\frac{1}{2A}}.$$

Hence,

$$\prod_{0\le i\le k-2,\, \alpha_{t-i}>\frac{1}{2}} \phi_{t-i} \le \left(\frac{4A}{\nu}\right)^{k-1} \frac{1}{1-\frac{1}{2A}}\, \sigma^{x-y}, \tag{50}$$

where $x = |\{i : 0 \le i \le k-2,\ \alpha_{t-i} > \frac{1}{2}\}|$ and $y = |\{i : 0 \le i \le k-2,\ \alpha_{t-i} \le \frac{1}{2}\}|$. From equality 12, the fact $\phi_{t-i} \ge 1$ (see Claim 2.4), and inequality 50, we obtain

$$\left(\frac{4\sigma A}{\nu(1-\frac{1}{2A})}\right)^{2z} = \left(\frac{\phi}{65}\right)^{z} \le \prod_{0\le i\le k-2,\, \alpha_{t-i}>\frac{1}{2}} \phi_{t-i} \le \left(\frac{4A}{\nu}\right)^{k-1} \frac{1}{1-\frac{1}{2A}}\, \sigma^{x-y}, \tag{51}$$

where $z = |\{i : 0 \le i \le k-2,\ \alpha_{t-i} > \frac{1}{2},\ \phi_{t-i} \ge \frac{\phi}{65}\}|$. Let β be a constant such that

$$\frac{4A}{\nu(1-\frac{1}{2A})} = \sigma^{\beta}. \tag{52}$$

From inequality 51 and the fact that $k \ge 2$, we have

$$\sigma^{(\beta+1)2z} \le \sigma^{(x-y)+\beta(k-1)}.$$

Therefore,

$$z \le f(\beta) = \frac{\beta(k-1) + (x-y)}{2(1+\beta)}. \tag{53}$$

By simple calculus, we have

$$f'(\beta) = \frac{(k-1) - (x-y)}{2(1+\beta)^2} \ge 0.$$

Thus, $f(\beta)$ is monotonically increasing. According to equation 52, we can enforce $\beta \le \frac{1}{2}$ by letting σ be large. Therefore, by inequality 53,

$$z \le f\left(\frac{1}{2}\right) = \frac{\frac{1}{2}(k-1) + x - y}{3} \le \frac{x-y}{2} + \frac{k-1}{4},$$

where the last inequality holds since, for σ sufficiently large, inequality 50 (together with Claim 2.4) implies $x - y \ge -\frac{k-1}{2}$. Thus,

$$|\{i : 0 \le i \le k-2,\ \alpha_{t-i} > \frac{1}{2},\ \phi_{t-i} < \frac{\phi}{65}\}| = x - z \ge x - \left(\frac{x-y}{2} + \frac{k-1}{4}\right) = \frac{k-1}{4}, \tag{54}$$

where the last equality holds since $x + y = k - 1$ by definition. Recall that for any i such that $0 \le i \le k-2$ and $\alpha_{t-i} > \frac{1}{2}$, we have used U_{t-i} as an intermediate group of registers in constructing R_{t-i-1} from R_{t-i} (see page 27). We define U_{t-i} to be **bad** if $\alpha_{t-i} > \frac{1}{2}$ and $\phi_{t-i} < \frac{\phi}{65}$. Inequality 54 implies that the number of bad U_{t-i} is at least $\frac{k-1}{4}$.

Let $i(1) < \cdots < i(q)$ be the increasing sequence of all integers i such that $0 \le i \le k-2$ and U_{t-i} is bad. We have $q \ge \frac{1}{4}(k-1)$. Applying Lemma 2.3 to the sequences

$$r_t \le r_{t-1} \le \cdots \le r_{t-(k-2)} \quad \text{and} \quad r_{t-i(1)} \le r_{t-i(2)} \le \cdots \le r_{t-i(q)},$$

we know that there exists an integer $s \ge \frac{k-1}{8}$ such that

$$\sum_{1 \le j \le s} r_{t-i(j)} \ge \frac{1}{8} \sum_{0 \le j \le i(s)} r_{t-j}. \tag{55}$$

We will finish our analysis of Case 2 by proving that the probability there exists such a sequence of bad sets (which will be referred to as a **bad sequence** hereafter) $U_{t-i(1)}, \ldots, U_{t-i(s)}$ is small. For $1 \le j \le s$, let B_j be the set of blocks that contain at least one register in $U_{t-i(j)}$, $u_j = |U_{t-i(j)}|$, and $b_j = |B_j|$, i.e., the number of blocks that contain at least one register in $U_{t-i(j)}$.

Claim 2.5 *For a given sequence $U_{t-i(1)}, U_{t-i(2)}, \ldots, U_{t-i(s)}$,*

$$\Pr\Big(U_{t-i(1)}, U_{t-i(2)}, \ldots, U_{t-i(s)} \text{ is a bad sequence}\Big) \le \rho^{\Theta(\sum_{1 \le j \le s} l b_j)}.$$

Proof: If $U_{t-i(j)}$ is bad, then $\phi_{t-i(j)} < \frac{\phi}{65}$ by definition. Hence

$$\left\lfloor \frac{u'_{t-i(j)}}{r_{t-i(j)}} \right\rfloor < \frac{\phi}{65}, \tag{56}$$

where $u'_{t-i(j)}$ is defined in the construction of $R_{t-i(j)-1}$ from $R_{t-i(j)}$. For a sufficiently large choice of the constant ϕ (so that $\frac{\phi}{65}+1<\frac{\phi}{64}$), inequality 56 implies

$$\frac{u'_{t-i(j)}}{r_{t-i(j)}} < \frac{\phi}{64}. \tag{57}$$

According to our inductive construction of $R_{t-i(j)-1}$ from $R_{t-i(j)}$ and the fact $\alpha_{t-i(j)} > \frac{1}{2}$ (since $U_{t-i(j)}$ is assumed to be bad), we have $|U_{t-i(j)}| \geq \frac{1}{8}\alpha_{t-i(j)} r_{t-i(j)} \geq \frac{r_{t-i(j)}}{16}$. Therefore, inequality 57 implies

$$\frac{u'_{t-i(j)}}{|U_{t-i(j)}|} < \frac{\phi}{4}. \tag{58}$$

Note that inequality 31 holds at stage $t-i(j)$ and that all of the items in $U_{t-i(j)}$ are at least 1-strange. Hence, by Claim 2.3 and the fact $\frac{\mu}{\lambda'/2} < 0.48$, $U_{t-i(j)}$ satisfies the condition on R in Lemma 2.2. Note that we have chosen to compare $\frac{\mu}{\lambda'/2}$ with 0.48 (as opposed to 0.49) so that Lemma 2.2 can be applied even if the effect of integer rounding is taken into account (recall that all of the capacities under consideration are at least $\sqrt{m}$). Thus, by applying Lemma 2.2 to all of the dividers associated with $U_{t-i(j)}$, the probability that inequality 58 holds is at most $\rho^{\Theta(l\, b_j)}$. Hence, we have

$$\Pr(U_{t-i(j)} \text{ is bad }) \leq \rho^{\Theta(l\, b_j)}.$$

The claim now follows from the independence of the $U_{t-i(j)}$'s, $1 \leq j \leq s$. □

In the next few claims, we upper bound the number of possible ways of choosing the sequence $U_{t-i(1)}, \ldots, U_{t-i(s)}$ and show that even after this number is taken into account, the probability that a faulty l-AKS circuit contains a bad sequence $U_{t-i(1)}, \ldots, U_{t-i(s)}$ is very small.

Claim 2.6 *For fixed k, the number of ways of choosing the sequence $r_t, r_{t-1}, \ldots, r_{t-k+1}$ is at most $m\phi^{k-1}$.*

Proof: The number of ways of choosing r_t is at most m. By Claim 2.4, when r_{t-i} is given, the number of ways for choosing r_{t-i-1} is at most ϕ. Overall, the number of ways of choosing $r_t, r_{t-1}, \ldots, r_{t-k+1}$ is at most $m\phi^{k-1}$. □

Claim 2.7 *When $\operatorname{cap}(X)$ and r_t are both given, the number of ways for choosing R_t is at most*

$$O(\sqrt{m}\log m)\binom{\operatorname{cap}(X)}{r_t}.$$

Proof: Since the partial l-AKS circuit is run for $O(\log m)$ stages and the l-AKS tree has at most $(m/\operatorname{cap}(\text{root})) \leq \sqrt{m}$ nonempty nodes at each stage, the total number of ways of choosing node X is upper bounded by

$$O(\sqrt{m}\log m).$$

When X is given, the number of ways of choosing R_t, a set of registers contained in X, is at most

$$\binom{\operatorname{cap}(X)}{r_t}.$$

Multiplying the quantities in the last two formulae, we obtain the desired upper bound. □

Claim 2.8 *When the sequence $r_{t-i(1)}, \ldots, r_{t-i(s)}$ is given, the number of ways of choosing the sequence $b_1, \ldots, b_s$ is at most*

$$O\left(\prod_{1\le j\le s} r_{t-i(j)}\right).$$

Proof: This is because $b_j \le r_{t-i(j)}$ for each $1 \le j \le s$. □

Claim 2.9 *If R_t and the sequences $r_t, r_{t-1}, \ldots, r_{t-i(s)}$, $i(1), i(2), \ldots, i(s)$, and $b_1, \ldots, b_s$ are given, then the number of ways of choosing the sequence $B_1, \ldots, B_s$ is at most*

$$2^{O(\sum_{1\le j\le s} l\,b_j)}.$$

Proof: Let $\tilde{B}_{t-i}$ be the set of blocks that contain at least one register in R_{t-i} for $i = 0, 1, \ldots, i(s)+1$. We first upper bound the number of possible sequences

$$\tilde{B}_t, \tilde{B}_{t-1}, \ldots, \tilde{B}_{t-i(s)-1}.$$

(Note that by the choice of s stated before inequality 55, $i(s) \le k-2$ and hence $\tilde{B}_{t-i(s)-1}$ is well-defined.) Clearly, $\tilde{B}_t$ is fixed when R_t is given. We next count the number of ways for choosing $\tilde{B}_{t-i-1}$ when $\tilde{B}_{t-i}$ is given. Each block in $\tilde{B}_{t-i-1}$ is connected by a $(t-i-1)$th stage partitioner to some block in $\tilde{B}_{t-i}$. Since each block is connected to at most d blocks by a divider, it is connected to at most d^4 blocks by a partitioner. On the other hand, there are at most r_{t-i} blocks in $\tilde{B}_{t-i}$. Therefore, when $\tilde{B}_{t-i}$ is given, the number of ways of choosing $\tilde{B}_{t-i-1}$ is at most

$$\binom{d^4\, r_{t-i}}{r_{t-i-1}} = \binom{d^4\, r_{t-i}}{\phi_{t-i} r_{t-i}} \le \left(\frac{e\ d^4}{\phi_{t-i}}\right)^{\phi_{t-i} r_{t-i}} \le 2^{O(r_{t-i})}, \qquad \text{for } i = 0, \ldots, i(s) \tag{59}$$

where the constant behind the O-notation is dependent on ϕ and d but not dependent on ρ.

On the other hand, B_j corresponds to $U_{t-i(j)}$, and $\tilde{B}_{t-i(j)-1}$ corresponds to $R_{t-i(j)-1}$. Moreover, by the description of U_{t-i} on page 27, $R_{t-i(j)-1}$ and $U_{t-i(j)}$ correspond to the same stage of the AKS tree. Hence, B_j and $\tilde{B}_{t-i(j)-1}$ correspond to registers within the same stage. Therefore, by an argument similar to the preceding paragraph, we know that when $\tilde{B}_{t-i(j)-1}$ is given, the number of ways of choosing B_j is at most

$$2^{O(r_{t-i(j)-1})} = 2^{O(r_{t-i(j)})}, \qquad \text{for } j = 1, \ldots, s, \tag{60}$$

where we have used Claim 2.4. Multiplying all the quantities (i.e., for $i = 1, \ldots, i(s)$) in inequality 59 with all the quantities (i.e., for $j = 1, \ldots, s$) in equation 60, we obtain an upper bound on the number of ways to choose the sequence $B_1, \ldots, B_s$ under the assumption of the claim:

$$\begin{aligned} & 2^{O(\sum_{0\le j\le i(s)} r_{t-j})} \cdot 2^{O(\sum_{1\le j\le s} r_{t-i(j)})} \\ = \;& 2^{O(\sum_{1\le j\le s} r_{t-i(j)})} && \text{(by inequality 55)} \\ = \;& 2^{O(\sum_{1\le j\le s} l\,b_j)} && \text{(since } r_{t-i(j)} \le l\,b_j\text{).} \end{aligned}$$

□

Claim 2.10 *If the sequence $B_1, \ldots, B_s$ is given, then the probability there exists a bad sequence $U_{t-i(1)}, \ldots, U_{t-i(s)}$ is at most*

$$\rho^{\Theta(\sum_{1 \le j \le s} l\, b_j)}. \tag{61}$$

Proof: For each j such that $1 \le j \le s$, there are b_j blocks of size l in B_j. Hence, when B_j is given, the number of ways to select the items in $U_{t-i(j)}$ is at most $2^{l\, b_j}$. Thus, the number of ways to select the sequence $U_{t-i(1)}, \ldots, U_{t-i(s)}$ is upper bounded by

$$2^{l \sum_{1 \le j \le s} b_j}.$$

Therefore,

$$\begin{aligned}
& \Pr\left(\exists \text{ a bad sequence } U_{t-i(1)}, \ldots, U_{t-i(s)} \text{ given } B_1, \ldots, B_s\right) \\
\le\ & 2^{\sum_{1 \le j \le s} l\, b_j} \cdot \Pr\left(\text{a fixed sequence } U_{t-i(1)}, \ldots, U_{t-i(s)} \text{ is bad}\right) \\
\le\ & 2^{\sum_{1 \le j \le s} l\, b_j} \rho^{\Theta(\sum_{1 \le j \le s} l\, b_j)} \\
\le\ & \rho^{\Theta(\sum_{1 \le j \le s} l\, b_j)},
\end{aligned}$$

where the second inequality follows by Claim 2.5, and the last inequality holds for ρ sufficiently small. □

Claim 2.11 *Let* $\operatorname{cap}(X)$*, k, and sequences $r_t, \ldots, r_{t-k+1}$ and $i(1), \ldots, i(s)$ be given. Then*

$$\Pr(\exists \textit{ a bad sequence } U_{t-i(1)}, \ldots, U_{t-i(s)}) \le \rho^{\Theta(l \log m)}.$$

Proof: Claims 2.7 to 2.10 imply that for given $\operatorname{cap}(X)$, k, and sequences $r_t, \ldots, r_{t-k+1}$ and $i(1), \ldots, i(s)$,

$$\begin{aligned}
& \Pr(\exists \text{ a bad sequence } U_{t-i(1)}, \ldots, U_{t-i(s)}) \\
\le\ & O(\sqrt{m}\, \log m) \binom{\operatorname{cap}(X)}{r_t} O\left(\prod_{1 \le j \le s} r_{t-i(j)}\right) 2^{O(\sum_{1 \le j \le s} l\, b_j)} \rho^{\Theta(\sum_{1 \le j \le s} l\, b_j)} \\
\le\ & O(\sqrt{m}\, \log m) \binom{\operatorname{cap}(X)}{r_t} \rho^{\Theta(\sum_{1 \le j \le s} l\, b_j)},
\end{aligned} \tag{62}$$

where the last inequality follows since $l\, b_j \ge r_{t-i(j)}$ and ρ can be chosen sufficiently small.

To prove the claim, we need to show that the quantity in equation 62 is at most $\rho^{\Theta(l \log m)}$. Since $s \ge \frac{k-1}{8}$ and $l\, b_j \ge r_{t-i(j)} \ge r_t$ for each j such that $1 \le j \le s$, it suffices to show that, for ρ sufficiently small,

$$O(\sqrt{m}\ \log m) \left(\frac{\operatorname{cap}(X)\, e}{r_t}\right)^{r_t} \rho^{\Theta((r_t + l)(k-1))} \le \rho^{\Theta(l \log m)}.$$

We do this by proving

$$\left(\frac{\operatorname{cap}(X)\, e}{r_t} \rho^{k-1}\right)^{r_t + l} \le \rho^{\Theta(l \log m)}. \tag{63}$$

Let C_0 be a constant such that

$$\left(\frac{1}{2}\right)^{C_0 \log m+1} \left(\frac{1}{\sigma}\right)^{C_0 \log m} \mu \sqrt{m} = m^{\frac{1}{4}}.$$

We establish inequality 63 by considering the following two cases.

Case (a): $k-1 < C_0 \log m$.

By inequality 32 and the fact that $\mathrm{cap}(X) \geq \sqrt{m}$, we have

$$r_t = X_k > \left(\frac{1}{2}\right)^{k} \left(\frac{1}{\sigma}\right)^{k-1} \mu\, \mathrm{cap}(X) \geq \left(\frac{1}{2}\right)^{C_0 \log m+1} \left(\frac{1}{\sigma}\right)^{C_0 \log m} \mu \sqrt{m} = m^{\frac{1}{4}}.$$

Thus, $r_t + l \geq m^{\frac{1}{4}} \geq l \log m$ since $l \leq m^{\frac{1}{8}}$. Therefore, in order to show inequality 63, we need only show that for ρ sufficiently small,

$$\frac{\mathrm{cap}(X)\, e}{r_t} \rho^{k-1} < \rho^{\Theta(1)}.$$

For sufficiently small ρ, the last inequality follows since $k \geq 2$ and, by inequality 32, $\frac{\mathrm{cap}(X)}{r_t} = \frac{\mathrm{cap}(X)}{X_t} \leq \frac{2}{\mu}(2\sigma)^{k-1}$.

Case (b): $k-1 \geq C_0 \log m$.

By Claim 2.1, it is easy to see that each nonempty node has capacity at most $O(m)$. (This is true even after the integer rounding.) Hence, by the fact that node X is not empty at stage t, we have $\mathrm{cap}(X) = O(m)$. Therefore, for ρ sufficiently small,

$$\frac{\mathrm{cap}(X)\ e}{r_t} \rho^{k-1} \leq O(m)\, e\, \rho^{C_0 \log m} \leq \rho^{\Theta(\log m)},$$

establishing inequality 63. $\Box$

Continuing with the proof of Theorem 2.1, the number of choices for k is at most

$$O(\log m),$$

and (as argued in Case (b) of Claim 2.11) the number of ways of choosing $\mathrm{cap}(X)$ is at most

$$O(m).$$

By Claim 2.6, when k is given, the number of choices for the sequence $r_t, \ldots, r_{t-k+1}$ is at most

$$m\phi^{k-1} \leq O(m) 2^{O(\log m)} = 2^{O(\log m)}.$$

Furthermore, when k is given, the number of choices for the sequence $i(1), \ldots, i(s)$ is at most

$$\sum_{\frac{k-1}{8} \leq s \leq k-1} \binom{k-1}{s} \leq 2^{k-1} \leq 2^{O(\log m)},$$

since $\frac{k-1}{8} \leq s \leq k-1$ and $i(1) < i(2) < \cdots < i(s)$. Therefore, the number of choices for k, $\mathrm{cap}(X)$, and the sequences $r_t, \ldots, r_{t-k+1}$ and $i(1), \ldots, i(s)$ is at most

$$O(\log m) \cdot O(m) \cdot 2^{O(\log m)} \cdot 2^{O(\log m)} = 2^{O(\log m)}.$$

Multiplying the above number with the quantity in Claim 2.11, we obtain the desired upper bound on the probability that a bad sequence $U_{t-i(1)}, \ldots, U_{t-i(s)}$ exists. This completes our consideration of Case 2, as well as the inductive step for proving the theorem. ■

Corollary 2.1 *Let $\beta = 1 - \frac{1}{8\log 6}$ and let X be the set of all output registers of an m-input partial l-AKS circuit (where $l \leq m^{\frac{1}{8}}$), with ϕ and σ being sufficiently large constants. Under both the passive and the reversal fault models, there exists a fixed partition of X into disjoint sets $\{S, X_1, \ldots, X_{\overline{m}}\}$ where $\overline{m} = \Theta(m^{1-\beta})$, $|S| = O(m^{\frac{3}{4}})$, and $|X_1| = \ldots = |X_{\overline{m}}| = \Theta(m^{\beta})$ such that when ρ is less than a sufficiently small constant, a randomly faulty l-AKS circuit has the following property with probability at least $1 - \rho^{\Theta(l \log m)}$: On all input permutations, the items in X_i are smaller than the items in X_j for $1 \leq i < j \leq \overline{m}$.*

Proof: We prove the corollary by exhibiting a fixed partition of X with the necessary properties. Assume that at the last stage of the partial l-AKS circuit, the l-AKS tree has depth d (i.e., nodes at level d are nonempty and nodes strictly below d are all empty). If we were to run the l-AKS circuit for another stage, then the root of the l-AKS tree would have capacity less than $\sqrt{m}$. Hence, the capacity of the root at the last stage of the l-AKS tree is at most $\frac{\sqrt{m}}{\nu}$. Therefore, at the last stage of the partial l-AKS circuit, the total capacity of all nonempty nodes and the cold storage in the l-AKS tree is at most

$$\begin{aligned} & \frac{\sqrt{m}}{\nu} + \frac{\sqrt{m}}{\nu} \cdot 2A + \ldots + \frac{\sqrt{m}}{\nu} \cdot (2A)^{d-1} + \frac{\sqrt{m}}{\nu 2A} + \frac{\sqrt{m}}{\nu (2A)^2} + \frac{\sqrt{m}}{\nu (2A)^3} + \ldots \\ = & \frac{(2A)^d}{2A-1} \cdot \frac{\sqrt{m}}{\nu} \\ < & 6^d \sqrt{m}, \end{aligned} \tag{64}$$

where the last inequality follows from equation 1. By Claim 2.1 and the fact $\mathrm{cap}(\mathrm{root}) \geq \sqrt{m}$, we also know that (even after integer rounding) the number of registers in the l-AKS tree is at least

$$\begin{aligned} & \frac{1}{2A+1}\left(\sqrt{m} + \sqrt{m} \cdot 2A + \ldots + \sqrt{m} \cdot (2A)^{d-2}\right) \\ = & \frac{(2A)^{d-1} - 1}{(2A+1)(2A-1)} \sqrt{m} \\ > & 6^{d-3} \sqrt{m}. \end{aligned} \tag{65}$$

Combining inequalities 64 and 65 with the fact that there are actually m registers in the l-AKS tree (including the cold storage), we find that

$$\sqrt{m}\, 6^{d-3} < m < 6^d \sqrt{m},$$

and hence

$$\log_6 \sqrt{m} < d < \log_6 \sqrt{m} + 3. \tag{66}$$

Let S be the set of output registers in the top $\lceil \frac{d}{2} \rceil$ levels of the l-AKS tree. Consider the $\lfloor \frac{d}{4} \rfloor$th level of the l-AKS tree (the root is assumed to be at level 0). Label all of the $2^{\lfloor \frac{d}{4} \rfloor}$ nodes at this level from left to right with $1, \ldots, \overline{m}$, where

$$\overline{m} = 2^{\lfloor \frac{d}{4} \rfloor} = \Theta(m^{1-\beta}). \tag{67}$$

(Here, we have used inequality 66.) For $i \leq \overline{m}$, let T_i be the set of registers contained in the tree rooted at the node labeled i. Let

$$X_i = T_i - S.$$

The sets S and X_i, $1 \leq i \leq \overline{m}$, completely determine the partition $X = S \cup X_1 \cup \ldots \cup X_{\overline{m}}$, and we only need to show that the partition has the claimed property.

By calculations similar to those used to derive equation 64, and by inequality 66, we have

$$|S| \leq 6^{\lceil \frac{d}{2} \rceil} \sqrt{m} = O(m^{\frac{3}{4}}). \tag{68}$$

For $i \leq \overline{m}$,

$$\begin{array}{rcll} |X_i| & \geq & |T_i| - |S| & \\ & \geq & \frac{m-|S|}{\overline{m}} - |S| & \\ & \geq & \Omega(m^\beta) & \text{(by equation 67 and inequality 68).} \end{array}$$

On the other hand, by equation 67,

$$|X_i| \leq \frac{m}{\overline{m}} = O(m^\beta).$$

Thus,

$$|X_i| = \Theta(m^\beta),$$

as claimed.

It remains to prove that with probability at least $1 - \rho^{O(l \log m)}$, on all input permutations,

$$x_i < x_j \text{ for any pair of items } x_i \in X_i \text{ and } x_j \in X_j \text{ where } i < j.$$

By Theorem 2.1, we need only to show that the above relationship holds provided inequality 13 is true. Assume for the purposes of contradiction that for a given permutation Π and some $i < j$, there exist items x_i in X_i and x_j in X_j such that $x_i > x_j$. Then, either x_i is not in the natural interval of the root of T_i, or x_j is not in the natural interval of the root of T_j. Without loss of generality, assume that item x_i is not in the natural interval of the root of T_i. Then x_i is at least $\frac{d}{4}$-strange since x_i is at least $\lceil \frac{d}{2} \rceil - \lfloor \frac{d}{4} \rfloor \geq \frac{d}{4}$ levels away from the root of T_i. Let r_i be the register that contains x_i. By the definition of the potential function together with inequality 66,

$$P_\sigma(r_i) \geq \sigma^{\frac{d}{4}-1} \geq \sigma^{\frac{\log_6 \sqrt{m}}{4}-1}. \tag{69}$$

On the other hand, by inequality 13,

$$P_\sigma(r_i) \leq P_\sigma(X_i) \leq \mu \operatorname{cap}(X_i) \leq \mu m. \tag{70}$$

Inequalities 69 and 70 now yield a contradiction for a sufficiently large choice of the constant σ. ■

3 Passive-Fault-Tolerant Circuits

In this section, we use the fault-tolerance of the l-AKS circuit to construct circuits that are tolerant to passive faults. The most important result of this section is the construction of a passive-fault-tolerant sorting circuit with $O(\log n \log\log n)$ depth. Since a circuit contains at most $\frac{n}{2}$ comparators at each level, our circuit has $O(n \log n \log\log n)$ size. This provides the first nontrivial upper bound for sorting circuits that tolerate random passive faults, and answers the open question posed by Yao and Yao [20] up to an $O(\log\log n)$ factor.

In [20], Yao and Yao conjectured that any passive-fault-tolerant sorting or merging circuit has $\omega(n \log n)$ size. As an interesting application of our technique for sorting, we prove a tight bound of $\Theta(n \log n)$ on the size of passive-fault-tolerant **selection** circuits. (A selection circuit is defined as a circuit that outputs the median to a specified output register.) This result would imply a separation of the complexities for merging and selection if Yao and Yao's conjecture is correct. To the best of our knowledge, no such separation result is currently known for merging and selection.

Theorem 3.1 *There exists an explicit construction of a passive-fault-tolerant sorting circuit with $O(\log n \log\log n)$ depth.*

Roughly speaking, we construct the circuit in the following fashion. We first use a partial 1-AKS circuit to move all but $O(n^{\frac{3}{4}})$ special items to within $O(n^{\beta})$ of the correct positions, where $\beta < 1$ is the constant in Corollary 2.1. (In fact, we will only use 1-AKS circuits throughout this section.) Then, we use an approximate-insertion circuit (to be described below) to move the special items close to their correct positions. After these two steps, all items are within $O(n^{\alpha})$ of the correct position, for some constant $\alpha < 1$, and we can complete the sort recursively. Given the fault-tolerance of the partial l-AKS circuit, our construction is slightly simpler than that of [13]: we do not need a subcircuit to isolate the extreme items. More importantly, by using the fault-tolerance properties of the partial l-AKS circuit, we can prove that our circuit works on all input permutations, whereas the circuit in [13] only works for any fixed input permutation (or most input permutations). (For a detailed discussion of this aspect, see [13].) The basic approach in this section will be used again later in the paper to construct other fault-tolerant circuits, networks, and algorithms for sorting-related problems.

We first prove two lemmas before proving Theorem 3.1. An $(m+1)$-input circuit $\mathcal{C}$ is defined to be a **Δ-approximate-insertion** circuit if $\mathcal{C}$ outputs every item to within Δ of the correct position provided that the input sequence consists of a sorted list of m items plus another "unknown" item to be input to a given register.

Lemma 3.1 *For any ρ less than a sufficiently small constant, there exists an explicit construction of an $(m+1)$-input $(\rho, \rho^{\Theta(\log m)})$-passive-fault-tolerant $m^{\frac{6}{7}}$-approximate-insertion circuit with $O(\log m)$ depth.*

Proof: As mentioned in the introduction, all the circuits constructed in this paper are standard. So we will prove the lemma by explicitly constructing a standard circuit with the claimed properties. We will first construct a circuit that receives the unknown item at the top register. A circuit that receives the unknown item at the bottom register can be constructed in an entirely symmetric fashion. We then use these two circuits to construct a circuit that solves the general approximate-insertion problem where the unknown item is input to an arbitrary register in a general position.

In what follows, we describe an $m^{\frac{6}{7}}$-approximate-insertion circuit $\mathcal{C}$ that receives the unknown item at the top register. By the definition of approximate-insertion, the input list consists of a sorted list, L, of length m, and an unknown item, x, which will be input to the top register of $\mathcal{C}$ according to our assumption. Let $t = \lceil \log m \rceil$. Rather than solving the given approximate-insertion problem "directly", we first partition the m items of the sorted list L into $r = \lfloor \frac{m}{s} \rfloor$ contiguous group of size $s = 2^{t - \lfloor t/6 \rfloor}$, except for the last group, which has $m - (r-1)s$ items. Clearly, the last group contains at least s and strictly less than $2s$ items. To construct circuit $\mathcal{C}$, it is useful to think of all of the items in any one group as being indistinguishable. Conceptually, we now solve an insertion problem with only $r+1$ inputs (one for each group, plus the unknown item x), and we only need to move x into the group to which it belongs. This is because each group contains at most $2s$ items and the maximum distance between any pair of items in any single group is at most $2s \leq 2 \cdot 2^{\lceil \frac{5}{6} t \rceil} < 4 \cdot 2^{\frac{5}{6} \lceil \log m \rceil} < m^{\frac{6}{7}}$ for m sufficiently large.

We call a register **clean at level** i if it is not the top register and if it has not been compared with any other register strictly before level i; we call a register **unclean at level** i otherwise. Intuitively, a register that is clean at level i has no chance of containing x at level i; and a register that is unclean at level i has a non-zero chance of containing x. We also assign an index to each of the m registers as follows: (i) assign index 0 to the top register; (ii) for each j such that $1 \leq j \leq r-1$, assign index j to all registers whose inputs are taken from the jth group, which contains items with ranks between $(j-1)s+1$ and js in list L; (iii) assign index r to registers whose inputs are taken from the last group, which contains items with ranks between $(r-1)s+1$ and m in list L. Circuit $\mathcal{C}$ will have $5 \log r$ depth and the following properties for each i such that $1 \leq i \leq 5 \log r$: (i) the MIN (resp., MAX) input of every comparator at level i is unclean (resp., clean), and (ii) each register that is unclean at level i is compared with a register with a larger index that is clean at level i. Since $\mathcal{C}$ has $5 \log r$ depth, the total number of clean registers required in the whole process is

$$\sum_{1 \leq i \leq 5 \log r} 2^{i-1} = 2^{5 \log r} - 1 = r^5 - 1 < \left(\frac{m}{s}\right)^5 \leq s,$$

which is less than or equal to the number of items contained in any single group. Hence, we cannot possibly run out of clean registers in any group.

Given these restrictions, we can complete the description of circuit $\mathcal{C}$ by specifying the unique comparator at the first level, and by specifying for each register y that is output from level $i-1$, the index of the register against which y will be compared at level i. In particular, $\mathcal{C}$ is inductively constructed as follows. The unique comparator at the first level connects input x and a register with index 1, which is certainly clean at level 1. The second level of $\mathcal{C}$ will have two comparators. The MIN output of the comparator at the first level is fed into the MIN input of a comparator whose MAX input takes a register with index 1 that is clean at level 2; The MAX output of the comparator at the first level is fed into the MIN input of a comparator whose MAX input takes a register with index $1+2=3$ that is clean at level 2. In general, for each comparator at level $i-1$ that connects two inputs with indices j and $j+h$, the MAX output (which certainly has index $j+h$) is fed into the MIN input of a comparator at level i whose MAX input has index $\min\{j+h+2h, r\} = \min\{j+3h, r\}$; the MIN output (which certainly has index j) is fed into a comparator at level i whose MAX input has index $j + \max\{\lfloor \frac{h}{2} \rfloor, 1\}$. This completes the description of circuit $\mathcal{C}$.

We next show that circuit $\mathcal{C}$ has the claimed property. By the 0-1 principle, a circuit is a Δ-approximate-insertion circuit if it Δ-approximate-sorts every input sequence of the form 0^x1^y or 10^x1^y. Clearly, a standard circuit always sorts input sequences of the form 0^x1^y (which are already sorted). Hence, to show that $\mathcal{C}$ has the claimed property, we only need to prove that with probability at least $1-\rho^{\Theta(\log m)}$, a randomly generated faulty version of $\mathcal{C}$ outputs an $m^{\frac{6}{7}}$-approximate-sorted list on all input sequences of the form 10^x1^y. Moreover, since there are only $m-1$ sequences of the form 10^x1^y, for ρ sufficiently small, it suffices to show that on any fixed input sequence s of the form 10^x1^y, with probability at least $1-\rho^{\Theta(\log m)}$, a randomly generated faulty version of $\mathcal{C}$ produces an $m^{\frac{6}{7}}$-approximate-sorted list.

In what follows, we will focus on a particular input sequence s of the form 10^x1^y and a particular random faulty version, $\mathcal{C}^*$, of $\mathcal{C}$. We need to show that with probability at least $1-\rho^{\Theta(\log m)}$, $\mathcal{C}^*$ outputs an $m^{\frac{6}{7}}$-approximate-sorted list on input sequence s. We make a special mark on the 1 input to the top register, and we assume without loss of generality that the marked 1 will always be output to the MIN register when being compared with another 1. In addition, at each level, we mark the unique comparator that receives the marked 1 as one of its two input items. Clearly, the movement of the marked 1 is completely determined by the functionality of the marked comparators in $\mathcal{C}^*$ (it has nothing to do with the functionality of other comparators). Let w be the index of the group to which x belongs. To prove that $\mathcal{C}^*$ $m^{\frac{6}{7}}$-approximate-sorts s with high probability, it suffices to show that the marked 1 will be successfully moved to a register with index w with high probability.

When ρ is less than a sufficiently small constant, by a standard Chernoff bound argument [3], with probability at least $1-\rho^{\Theta(\log r)} = 1-\rho^{\Theta(\log m)}$, at most $\frac{\log r}{3}$ of the $5\log r$ marked comparators are faulty. In what follows, we will show that if at most $\frac{\log r}{3}$ marked comparators are faulty, then the marked 1 will be successfully moved to a register with index w. Given s and $\mathcal{C}^*$, we observe how the marked 1 moves within $\mathcal{C}^*$. For each $i = 1, 2, \ldots, 5\log r$, let d_i be the nonnegative integer such that the marked 1 is contained in a register with index $w - d_i$ immediately before level i; let h_i be the integer such that the other input register to the unique marked comparator at level i has index $w - d_i + h_i$. By our construction of $\mathcal{C}$, $h_i \geq 1$, for $i = 1, 2, \ldots, 5\log r$. We next prove that the marked 1 will be moved to the correct group of registers at the end of $\mathcal{C}^*$ by showing that $d_{5\log r} = 0$. For any nonnegative integer k, we define $b(k)$ to be the number of bits in the binary representation of k (assume $b(0) = 0$). Define a potential function F as follows:

$$F(i) = 3b(d_i) + |b(d_i) - b(h_i)|.$$

Claim 3.1 *The following inequalities hold when $d_i \geq 1$ and the marked comparator at level i is correct: (i) $F(i+1) \leq F(i) + 1$, and (ii) $F(i+1) \leq F(i) - 1$.*

By the inequalities $|x| - |y| \leq |x - y|$ and $|x + y| \leq |x| + |y|$,

$$F(i+1) - F(i) \leq 3\,(b(d_{i+1}) - b(d_i)) + |b(d_{i+1}) - b(d_i)| + |b(h_{i+1}) - b(h_i)|. \tag{71}$$

Hence, the first inequality in the claim follows from the fact that

$$b(d_{i+1}) \leq b(d_i) \tag{72}$$

and

$$b(h_{i+1}) \leq b(h_i) + 1. \tag{73}$$

Assuming that the marked comparator at level i is correct and that $d_i \geq 1$, we next prove

$$F(i+1) \leq F(i) - 1. \tag{74}$$

When $b(d_{i+1}) < b(d_i)$, inequalities 71 and 73 immediately imply inequality 74. Hence, by inequality 72, we only need to check inequality 74 under the assumption that

$$b(d_{i+1}) = b(d_i). \tag{75}$$

Given equality 75 and the assumption that the marked comparator at level i is correct, we conclude that $b(h_i) \neq b(d_i)$. There are two cases to consider.

Case 1: $b(h_i) > b(d_i)$. In this case, the marked 1 will be output to the MIN output of the marked comparator at level i. Hence, $h_{i+1} = \max\{\lfloor \frac{h_i}{2} \rfloor, 1\} = \lfloor \frac{h_i}{2} \rfloor$ since $b(h_i) > b(d_i) \geq 1$ implies $h_i \geq 2$. Hence,

$$b(h_{i+1}) = b(h_i) - 1.$$

This, together with the assumption of Case 1 and equality 75, implies that

$$F(i+1) - F(i) = (b(h_{i+1}) - b(d_{i+1})) - (b(h_i) - b(d_i)) = b(h_{i+1}) - b(h_i) = -1.$$

Case 2: $b(h_i) < b(d_i)$. In this case, the marked 1 will be output to the MAX output of the marked comparator at level i. Hence, either $h_{i+1} = 2h_i$ or the marked 1 will be compared against a register with the largest index, r, at level $i+1$. In the former case, we have

$$b(h_{i+1}) = b(h_i) + 1. \tag{76}$$

In the latter case, $h_{i+1} \geq d_{i+1}$, which implies $b(h_{i+1}) \geq b(d_{i+1}) = b(d_i) > b(h_i)$ (where we have used equality 75 and the assumption of Case 2). Again, equality 76 holds (note that $b(h_{i+1}) \leq b(h_i) + 1$ holds trivially). Using equalities 75 and 76 and the assumption of Case 2, we have

$$F(i+1) - F(i) = (b(d_i) - b(h_i) - 1) - (b(d_i) - b(h_i)) = -1.$$

This proves inequality 74 and concludes the proof of Claim 3.1.

By Claim 3.1, we know that before d_i becomes 0, each correct marked comparator decreases the potential function by at least 1, and each faulty marked comparator increases the potential function by at most 1. Since we have assumed that at most $\frac{\log r}{3}$ of the $5 \log r$ marked comparators are faulty, the potential decreases by at least $\frac{14}{3} \log r - \frac{1}{3} \log r = \frac{13}{3} \log r$ unless

$d_i = 0$ for some $i \leq 5\log r$. Since the initial potential is $F(0) \leq 3\lfloor \log r \rfloor + |\lfloor \log r \rfloor - 1| \leq 4\log r$, this means that $F(5\log r)$ is negative unless $d_i = 0$ for some $i \leq 5\log r$. Therefore, $d_i = 0$ for some $i \leq 5\log r$, which means that at a certain level, the marked 1 is moved into a register with index w. Since we only have passive faults, once the marked 1 is moved into the register with index w, it will stay there until the end of $\mathcal{C}^*$. This proves that $\mathcal{C}$ has the claimed property.

By using an entirely symmetric construction, we can construct an $m^{\frac{6}{7}}$-approximate-insertion circuit that receives the unknown item at the bottom register. The same argument can be used to prove that the circuit has the desired property.

To construct an approximate-insertion circuit that receives the unknown item at an arbitrary position, we use the following simple technique to increase the success probability of each comparator, as in [20]. If we apply l consecutive comparators, each of which fails to work with probability upper bounded by ρ, to two registers, then the probability that the items contained in the two registers are unsorted after these l consecutive comparators is at most ρ^l. The l consecutive comparators can be viewed as a comparator whose failure probability is at most ρ^l. This technique will be used in many other passive-fault-tolerant circuit constructions, and will be referred to as the **replication** technique. For example, by using the replication technique, the approximate-insertion circuit that receives the unknown item at the top or bottom register can be made into an $O(l\log m)$-depth $(\rho, \rho^{\Theta(l\log m)})$-passive-fault-tolerant $m^{\frac{6}{7}}$-approximate-insertion circuit that receives the unknown item at the top or bottom register.

In general, when the unknown item has to be input to a given register r_x in a general position, we can construct the desired circuit by applying a circuit to all of the registers above and including r_x followed by another circuit to all of the registers below and including r_x. In one of the two circuits, we may have to use the replication technique to achieve the desired success probability if the number of inputs to that circuit is not large enough. ■

Lemma 3.2 *For any ρ less than a sufficiently small constant, there exists an explicit construction of an m-input $(\rho, \rho^{\Theta(\log m)})$-passive-fault-tolerant m^α-approximate-sorting circuit with $O(\log m)$ depth, where $\alpha < 1$ is a fixed constant.*

Proof: We construct the claimed circuit as follows. First, we apply a partial 1-AKS circuit to all m inputs. Let $X = S \cup X_1 \cup \ldots \cup X_{\overline{m}}$ be the partition of the output registers of the partial 1-AKS circuit as specified in Corollary 2.1, where $\overline{m} = \Theta(m^{1-\beta})$, $|S| = O(m^{\frac{3}{4}})$, and $|X_1| = \ldots = |X_{\overline{m}}| = \Theta(m^\beta)$.

For $i \leq \overline{m}$ and $j \leq |X_1|$, let x_{ij} be the jth register in X_i, and let $Y_j = \{x_{ij} \mid 1 \leq i \leq \overline{m}\}$. By Corollary 2.1, with probability at least $1 - \rho^{\Theta(\log m)}$, the item contained in x_{kj} is smaller than that contained in x_{lj} for any $k < l$, and Y_j thus contains a sorted list for $j \leq |X_1|$. For $j \leq |S|$, let s_j be the jth register in S. In parallel, we apply the $(\overline{m}+1)$-input $\overline{m}^{\frac{6}{7}}$-approximate-insertion circuit of Lemma 3.1 to $\{s_j\} \cup Y_j$ for each $1 \leq j \leq |S|$. By Lemma 3.1, with probability at least $1 - |S|\,\rho^{\Theta(\overline{m})} = 1 - \rho^{\Theta(\log m)}$, all of the items in $\{s_j\} \cup Y_j$ are at most $\overline{m}^{\frac{6}{7}}$ away from their correct positions within $\{s_j\} \cup Y_j$ for all $j \leq |S|$. Since distance one within Y_j corresponds to distance at most $|X_1|$ within $\cup_{1 \leq j \leq |X_1|} Y_j = \cup_{1 \leq i \leq \overline{m}} X_i$, distance one within $\{s_j\} \cup Y_j$ corresponds to distance at most $|X_1| + |S| = \Theta(m^\beta)$ in the whole circuit. Thus, distance $\overline{m}^{\frac{6}{7}}$ within $\{s_j\} \cup Y_j$ corresponds to distance of at most $\overline{m}^{\frac{6}{7}}\Theta(m^\beta) = \Theta(m^{(1-\beta)\frac{6}{7}+\beta}) \leq m^\alpha$ (where α can be any constant strictly less than 1 and strictly greater than $(1-\beta)\frac{6}{7}+\beta$) in the whole circuit. ■

Proof of Theorem 3.1: To construct our passive-fault-tolerant sorting circuit with $O(\log n \log\log n)$ depth, we will repeatedly apply the approximate-sorting circuit of Lemma 3.2. In order to apply Lemma 3.2, we need to ensure that the failure probability for each comparator is upper bounded by a sufficiently small constant. This can be achieved by using the replication technique described on page 40. In particular, for any ρ and ϵ, we can construct a circuit that simulates a comparator with failure probability at most ϵ by simply replicating $\lceil \log_\rho \epsilon \rceil$ times a comparator with failure probability at most ρ. The circuit thus constructed will be referred to as a (ρ, ϵ)-**enforced** comparator hereafter. In order to apply Lemma 3.2, we need only set ϵ to a sufficiently small constant, which in turn leads to only constant factor replication. Then, we can build the whole circuit from these (ρ, ϵ)-enforced comparators. By applying this replication technique, the size and the depth of the whole circuit are both increased by only a constant factor. Ignoring this constant factor, we will assume that all of the comparators in our construction have a sufficiently small ρ and that Lemma 3.2 can be applied.

Let α be the constant in Lemma 3.2. Our circuit consists of $O(\log\log n)$ rounds. The ith round circuit has $O(\log n)$ depth, and outputs a Δ_i-approximate-sorted list with probability at least $1-\frac{1}{n^2}$ provided that the list produced by the $(i-1)$th round circuit is Δ_{i-1}-approximate-sorted, where Δ_i is a parameter determined by the recurrence:

$$\Delta_i = (4\Delta_{i-1})^\alpha, \qquad \text{for } i \geq 2, \tag{77}$$

with the boundary condition

$$\Delta_1 = n^\alpha. \tag{78}$$

The reason that we need to upper bound the failure probability of each round by $\frac{1}{n^2}$ instead of $\frac{1}{n}$ is that we will have $O(\log\log n)$ rounds and we will upper bound the overall failure probability by $O(\frac{1}{n^2}\log\log n) \leq \frac{1}{n}$ for n sufficiently large.

Recurrence 77 can be rewritten as

$$a\Delta_i = (a\Delta_{i-1})^\alpha,$$

where $a = \left(\frac{1}{4}\right)^{\frac{\alpha}{1-\alpha}}$. Solving this recurrence, we find that

$$\Delta_i = \Theta(n^{\alpha^i}). \tag{79}$$

In the first round of our circuit, we apply the n^α-approximate-sorting circuit of Lemma 3.2 to all n inputs. In particular, we use the replication technique to make the failure probability of each comparator sufficiently small so that the term $\rho^{\Theta(\log n)}$ in Lemma 3.2 is smaller than $\frac{1}{n^2}$. Lemma 3.2 implies that, with probability at least $1-\frac{1}{n^2}$, the outputs of the first round form a Δ_1-approximate-sorted list. The depth for the first round is $O(\log n)$.

For $i \geq 2$, we construct the ith round of the circuit so that with probability at least $1-\frac{1}{n^2}$, the outputs of the ith round form a Δ_i-approximate-sorted list provided that the outputs from the $(i-1)$th round form a Δ_{i-1}-approximate-sorted list. At the beginning of the ith round, we group all the output registers from the $(i-1)$th round as follows. For $1 \le k \le \frac{n}{2\Delta_{i-1}}$, let X_k be the set of registers in positions $(k-1)2\Delta_{i-1}+1$ to $k2\Delta_{i-1}$. In parallel, we apply an approximate-sorting circuit of Lemma 3.2 to $X_1 \cup X_2$, another approximate-sorting circuit of Lemma 3.2 to $X_3 \cup X_4$, and so on. Then, in parallel, we apply an approximate-sorting circuit of Lemma 3.2 to $X_2 \cup X_3$, another approximate-sorting circuit of Lemma 3.2 to $X_4 \cup X_5$, and so on. We need the second set of approximate-sorting circuits to bring items across the boundaries of the first set of approximate-sorting circuits.

To analyze the behavior of the ith round circuit, we will make use the 0-1 principle. For any 0-1 sequence $s = (s_1, s_2, \ldots, s_m)$, we define the **dirty window** of s to be a subsequence of s of the form $(s_i, s_{i+1}, \ldots, s_j)$ such that: (i) $s_i = 1$, (ii) $s_j = 0$, (iii) $s_k = 0$ for all $k < i$, and (iv) $s_k = 1$ for all $k > j$. In other words, the dirty window of s is the shortest subsequence of s such that every item strictly before the subsequence is 0 and every item strictly after the subsequence is 1. Intuitively, the dirty window of a 0-1 sequence s is the part of s that we need to work on in order to sort s.

We next use the 0-1 principle to prove that if all of the constituent approximate-sorting circuits in the ith round work correctly as $(4\Delta_{i-1})^{\alpha}$-approximate-sorting circuits, and if the list produced by the $(i-1)$th round is Δ_{i-1}-approximate-sorted, then the ith round circuit produces a Δ_i-approximate-sorted list. Let us restrict our attention to 0-1 inputs and assume that the list produced by the $(i-1)$th round is Δ_{i-1}-approximate-sorted. Then, the list input to the ith round contains a dirty window of size $2\Delta_{i-1}$ or less. This dirty window is fully contained in one of the approximate-sorting circuits involved in the ith round. (Recall that we have two sets of $(4\Delta_{i-1})$-input approximate-sorting circuits in the ith round, where the second set is offset from the first by $2\Delta_{i-1}$. Also, note that passive faults cannot increase the size of the dirty window.) Hence, after the two sets of approximate-sorting circuits in the ith round, the output list is $(4\Delta_{i-1})^{\alpha}$-approximate-sorted, i.e., it is Δ_i-approximate-sorted.

In order to make sure that with sufficiently high probability, all of the constituent approximate-sorting circuits in the ith round work correctly, we need to do some careful calculations. In the above construction, if we simply apply the approximate-sorting circuits of Lemma 3.2 as in the first round, then the failure probability for each of the approximate-sorting circuits is $\frac{1}{(4\Delta_{i-1})^2} = \frac{1}{O(n^{2\alpha^{i-1}})}$, which is too large in comparison with the goal of $\frac{1}{n^2}$. To overcome this difficulty, we need to make use of the replication technique again. In particular, we construct the approximate-sorting circuits from (ρ, ϵ)-enforced comparators where ϵ satisfies

$$\epsilon^{\Theta(\log(4\Delta_{i-1}))} \le \frac{1}{n^3}, \tag{80}$$

and where the constant behind the Θ-notation is the same as that in Lemma 3.2. By doing so, each of the approximate-sorting circuits fails with probability at most $\frac{1}{n^3}$. Hence, the probability that all of the constituent approximate-sorting circuits in the ith round are good is at least $1 - \frac{1}{n^2}$. To construct the (ρ, ϵ)-enforced comparators, we replicate each of the original comparators

$$\log_\rho \epsilon = O\left(\frac{3\log_{\frac{1}{\rho}} n}{\log(4\Delta_{i-1})}\right) = O\left(\frac{\log_{\frac{1}{\rho}} n}{\log n^{\alpha^{i-1}}}\right)$$

times (where the second equality follows from equation 79). This replication results in an

$$O\left(\frac{\log_{\frac{1}{\rho}} n}{\log n^{\alpha^{i-1}}}\right)$$

blowup in the original $O(\log(4\Delta_{i-1}))$-depth construction. Hence, the total depth of the ith round is

$$O\left(\log(4\Delta_{i-1})\right) O\left(\frac{\log_{\frac{1}{\rho}} n}{\log n^{\alpha^{i-1}}}\right) = O\left(\log n^{\alpha^{i-1}} \frac{\log_{\frac{1}{\rho}} n}{\log n^{\alpha^{i-1}}}\right) = O(\log_{\frac{1}{\rho}} n).$$

We repeatedly apply the above construction until every item is moved to within a constant number of positions of the correct position, i.e., we use i rounds such that $\Delta_i = O(1)$, which is equivalent to $n^{\alpha^i} = O(1)$ or $i = O(\log\log n)$ by equation 79. Note that we should not apply the above construction all the way to the end to sort every item precisely to the correct position, since we have only established the fault-tolerance property of the l-AKS circuit for large m. When every item is within a constant number of positions of the correct position, we can combine the replication technique with any sorting circuit with a constant number of inputs (hence also constant depth and size) to achieve exact sorting. This costs $O(\log n)$ additional depth. Overall, we have $O(\log\log n)$ rounds each of which has $O(\log n)$ depth and works with probability at least $1 - \frac{1}{n^2}$. Therefore, the total depth is $O(\log n \log\log n)$ and the failure probability is $O(\log\log n)\frac{1}{n^2} \leq \frac{1}{n}$ for n sufficiently large. ■

As we have pointed out at the beginning of the paper, the construction can be made $(\rho, \frac{1}{\mathrm{poly}(n)})$-passive-fault-tolerant with only a constant factor increase in depth and size. In particular, the proof of Theorem 3.1 can easily be extended to establish the following corollary.

Corollary 3.1 *For any constants $\rho < 1$ and c, there exists an explicit construction of a $(\rho, \frac{1}{n^c})$-passive-fault-tolerant sorting circuit with $O(\log n \log\log n)$ depth.*

Theorem 3.2 *There exists an explicit construction of a passive-fault-tolerant selection circuit with asymptotically optimal size of $\Theta(n \log n)$.*

Proof: An $\Omega(n \log n)$ lower bound on the size of selection circuits was proved by Alekseyev [2] even in the fault-free case (see also Theorem A on pages 234-235 of [9]). In what follows, we give a passive-fault-tolerant construction with $O(n \log n)$ size.

Let α be the constant of Lemma 3.2, and let $\rho < 1$ be a constant upper bound on the failure probability of each comparator. Take a $(\rho, \frac{1}{n^2})$-passive-fault-tolerant n^α-approximate-sorting circuit $\mathcal{C}_1$ as in Lemma 3.2. Take another $(2n^\alpha + 2)$-input $(\rho, \frac{1}{n^2})$-passive-fault-tolerant-sorting circuit $\mathcal{C}_2$ as in Corollary 3.1, e.g., we can choose $c = \frac{3}{\alpha}$ in Corollary 3.1. Our passive-fault-tolerant selection circuit $\mathcal{C}$ consists of $\mathcal{C}_1$ followed by $\mathcal{C}_2$ with the middle $2n^\alpha + 2$ outputs of $\mathcal{C}_1$ being the inputs of $\mathcal{C}_2$. Clearly, the size of $\mathcal{C}$ is $O(n \log n) + O(n^\alpha \log n \log\log n) = O(n \log n)$.

We next show that the circuit constructed is a $(\rho, \frac{1}{n^2})$-passive-fault-tolerant circuit that outputs the median to the middle output register of $\mathcal{C}_2$. By the choice of $\mathcal{C}_1$ and $\mathcal{C}_2$, the probability that $\mathcal{C}_1$ is an n^α-approximate-sorting circuit and that $\mathcal{C}_2$ is a sorting circuit is at least $1 - \frac{2}{n^2} \geq 1 - \frac{1}{n}$. Hence, to show that $\mathcal{C}$ is a $(\rho, \frac{1}{n})$-passive-fault-tolerant selection circuit, we need only prove that when $\mathcal{C}_1$ is an n^α-approximate-sorting circuit and $\mathcal{C}_2$ is a sorting circuit, $\mathcal{C}$ always outputs the median input to its middle output register. When $\mathcal{C}_1$ is an n^α-approximate-sorting circuit, by the definition of approximate-sorting, the top $\frac{n}{2} - n^\alpha - 1$ outputs of $\mathcal{C}_1$ all contain items smaller than the median input to $\mathcal{C}$, and the bottom $\frac{n}{2} - n^\alpha - 1$ outputs of $\mathcal{C}_1$ all contain items larger than the median input to $\mathcal{C}$. Hence, the median input to $\mathcal{C}$ will be the median among all of the inputs to $\mathcal{C}_2$. When $\mathcal{C}_2$ is indeed a sorting circuit, the median input to $\mathcal{C}_2$, which is identical to the median input to $\mathcal{C}$, will be output to the middle output register of $\mathcal{C}_2$, which is the same as the middle output register of $\mathcal{C}$. ■

4 Reversal-Fault-Tolerant Circuits and Networks

In this section, we present our results for reversal faults. We consider both the circuit model and the network model. In Section 4.1, we use the fault-tolerance of the l-AKS circuit to construct a reversal-fault-tolerant $O(\log n)$-approximate-sorting circuit (for ρ less than a sufficiently small constant) with $O(n \log n (\log\log n)^2)$ size and $O(\log^2 n)$ depth. In addition, we present some general lower bounds for reversal-fault-tolerant approximate-sorting circuits. In Section 4.2, we use the approximate-sorting circuit of Section 4.1 to construct a reversal-fault-tolerant sorting network (for ρ less than a sufficiently small constant) with $O(n \log^{\log_2 3} n)$ size. This provides the first $o(n \log^2 n)$-size reversal-fault-tolerant sorting network, thereby answering the open question posed by Assaf and Upfal [4]. In light of a lower bound established in [12, Theorem 2], this result separates the size complexities of sorting networks with reversal and destructive faults. In all of the upper bound results of this section (except Lemma 4.4), we will need the assumption that ρ is less than a sufficiently small constant. Whether or not such an assumption is necessary is an open question. Note that no assumption will be made on ρ in the lower bound results.

4.1 Approximate-Sorting Circuits

In this section, we study approximate-sorting circuits that are tolerant to reversal faults. As defined in Section 2, a Δ-approximate-sorting circuit is a circuit that outputs every item to within Δ of its correct position on all input permutations.

In Theorem 4.1 below, we show that any reversal-fault-tolerant Δ-approximate-sorting circuit has $\Delta = \Omega(\log n)$. This lower bound leads us to focus our attention on the study of reversal-fault-tolerant $O(\log n)$-approximate-sorting circuits in the remainder of Section 4.1. We continue in Theorem 4.2 with an $\Omega(\log^2 n)$ lower bound on the depth of any reversal-fault-tolerant $O(\log n)$-approximate-sorting circuit. The main result of Section 4.1 is Theorem 4.3, where we construct a reversal-fault-tolerant $O(\log n)$-approximate-sorting circuit (for ρ less than a sufficiently small constant) with $O(n \log n (\log\log n)^2)$ size and asymptotically optimal depth of $O(\log^2 n)$. The size of this circuit is of particular importance when we further modify the circuit into a network with $o(n \log^2 n)$ size in Section 4.2.

Theorem 4.1 *For any positive constant $\gamma < 1$, there exists a positive constant c depending on γ such that for any n-input circuit, where n is sufficiently large (depending on ρ and γ),* *with probability at least* $1 - \exp(-\frac{n^\gamma}{\log \frac{1}{\rho}})$ *there is an input permutation for which the circuit outputs some item at least* $c \log_{\frac{1}{\rho}} n$ *away from the correct position.*

Proof: See Appendix A. ■

When ρ is a constant, the preceding theorem has the following immediate corollary, where the constant c depends on ρ.

Corollary 4.1 *There is some constant* c *for which no reversal-fault-tolerant* $(c \log n)$*-approximate-sorting circuit exists.*

Theorem 4.2 *For any positive constants* c*,* ρ*, and* $\gamma < 1$*, any* (ρ, n^{-c})*-reversal-fault-tolerant* n^γ*-approximate-sorting circuit has* $\Omega(\log^2 n)$ *depth.*

Proof: See Appendix B. ■

We remark that, as indicated in the proof, the $\Omega(\log^2 n)$ depth lower bound of Theorem 4.2 actually holds for the much simpler problem of n^γ-approximate-insertion.

Theorem 4.3 *For any* ρ *less than a sufficiently small constant, there exists an explicit construction of a reversal-fault-tolerant* $O(\log n)$*-approximate-sorting circuit with* $\Theta(\log^2 n)$ *depth and* $O(n \log n (\log \log n)^2)$ *size.*

The remainder of Section 4.1 is devoted to the proof of Theorem 4.3. The next few lemmas will be very useful.

Lemma 4.1 *For any* ρ *less than a sufficiently small constant and for* $l \geq \log \frac{m}{l}$*, there exists an explicit construction of an* m*-input* $(\rho, \rho^{\Theta(l)})$*-reversal-fault-tolerant* l*-approximate-insertion circuit with* $O(l \log \frac{m}{l})$ *depth and* $O(ml)$ *size.*

Proof: Without loss of generality, we assume that both m and l are integral powers of 2. We label the registers $r_1, \ldots, r_m$ from the top to the bottom, and we assume that the unique unknown item is input to a particular register r_x. For $1 \leq i \leq \log \frac{m}{l}$, let $R_i = \{r_j \mid j \equiv x \bmod \frac{m}{2^i l}\}$. Our circuit consists of $\log \frac{m}{l}$ rounds. In the ith round, we group all the registers in R_i from the top to the bottom into groups $X_{i,1}, X_{i,2}, \ldots, X_{i,2^i}$ so that each of the groups contains l registers. First, in parallel, we apply a set of $2l$-input and $4l$-depth odd-even transposition circuits to $X_{i,1} \cup X_{i,2}$, $X_{i,3} \cup X_{i,4}$, and so on. Second, in parallel, we apply another set of $2l$-input and $4l$-depth odd-even transposition circuits to $X_{i,2} \cup X_{i,3}$, $X_{i,4} \cup X_{i,5}$, and so on.

Since each R_i contains 2^i groups of the form $X_{i,j}$, the total number of odd-even transposition circuits in the ith round is at most $2 \cdot 2^{i-1} = 2^i$. Hence, the total number of odd-even transposition circuits used in the entire circuit is at most

$$\sum_{1 \leq i \leq \log \frac{m}{l}} 2^i \leq 2^{1+\log \frac{m}{l}} = \frac{2m}{l}. \tag{81}$$

Since each of the odd-even transposition circuits has size $O(l^2)$, the size of the entire circuit is $O(\frac{m}{l}l^2) = O(ml)$. Also, since we have $\log \frac{m}{l}$ rounds, each of which consists of two sets of $O(l)$-depth odd-even transposition circuits, the depth of the entire circuit is $O(l \log \frac{m}{l})$.

By Lemma 2.1 with $\vartheta = \frac{1}{32}$, each of the odd-even transposition circuits used in the entire circuit is an $\frac{l}{32}$-approximate-sorting circuit with probability at least $1 - \rho^{\Theta(l)} = 1 - \rho^{\Theta(l)}$. Hence, by inequality 81, the probability that all of the constituent odd-even transposition circuits are $\frac{l}{32}$-approximate-sorting circuits is at least

$$1 - O\left(\frac{m}{l}\right)\rho^{\Theta(l)} = 1 - \rho^{\Theta(l)},$$

where we have used the fact that ρ is sufficiently small and $l \geq \log \frac{m}{l}$.

Thus, to prove the lemma, it remains to show that the circuit thus constructed is an l-approximate-insertion circuit when all of the constituent odd-even transposition circuits are $\frac{l}{32}$-approximate-sorting circuits. Assuming that all of the constituent odd-even transposition circuits are $\frac{l}{32}$-approximate-sorting circuits, we prove by induction on i that the items of R_i form an $\frac{l}{8}$-approximate-sorted list after the ith round. (This suffices to prove that our circuit is an l-approximate-insertion circuit since $R_{\log \frac{m}{l}}$ contains all m registers.)

The base case $i = 1$ is trivial, since the first round actually consists of a single $\frac{l}{32}$-approximate-sorting circuit. Assuming that the items from the $(i-1)$th round form an $\frac{l}{8}$-approximate-sorted list, we now show that the items from the ith round form an $\frac{l}{8}$-approximate-sorted list. By the 0-1 principle, we need only consider the case where the inputs consist of 0s and 1s. Recall that, as defined in the proof of Theorem 3.1, the dirty window of a 0-1 sequence s is the shortest subsequence of s such that every item strictly before the subsequence is 0 and every item strictly after the subsequence is 1. Since the sequence from the $(i-1)$th round is assumed to be $\frac{l}{8}$-approximate-sorted, the input sequence to the ith round is an $\frac{l}{4}$-approximate-sorted list and has a dirty window of size at most $\frac{l}{2}$. There are two cases to consider.

Case 1: The dirty window of the sequence input to the ith round is fully contained in a circuit that belongs to the first set of $\frac{l}{32}$-approximate-sorting circuits in the ith round. In this case, the size of the dirty window will be decreased to $\frac{2l}{32}$ or less by the first set of $\frac{l}{32}$-approximate-sorting circuits in the ith round. Then, the second set of $\frac{l}{32}$-approximate-sorting circuits may increase the size of the dirty window by at most an additive term of $\frac{2l}{32}$. (Note that this is different from the case of passive faults, which cannot increase the size of a dirty window.) Hence, the final output sequence from the ith round has a dirty window of size at most $\frac{4}{32}l = \frac{l}{8}$, and is $\frac{l}{8}$-approximate-sorted.

Case 2: The dirty window of the sequence input to the ith round is *not* fully contained in any of the circuits in the first set of $\frac{l}{32}$-approximate-sorting circuits in the ith round. In this case, the first set of $\frac{l}{32}$-approximate-sorting circuits in the ith round may increase the size of the dirty window by an additive term of $\frac{2l}{32}$. Hence, the sequence input to the second set of $\frac{l}{32}$-approximate-sorting circuits has a dirty window of size at most $\frac{l}{2} + \frac{2l}{32} = \frac{9l}{16}$. By the assumption of Case 2 and the fact that the boundaries of the first set of $\frac{l}{32}$-approximate-sorting circuits are the centers of the second set of $2l$-input $\frac{l}{32}$-approximate-sorting circuits, this dirty window of size at most $\frac{9l}{16}$ is fully contained in a circuit that belongs to the second set of $\frac{l}{32}$-approximate-sorting circuits. Hence, the final output sequence produced by the second set of $\frac{l}{32}$-approximate-sorting circuits in the ith round has a dirty window of size at most $\frac{2l}{32}$, and is thus $\frac{l}{8}$-approximate-sorted. ■

Lemma 4.2 *For $l \leq m^{\frac{1-\beta}{2}}$ (where β is the constant specified in Corollary 2.1) and for any ρ less than a sufficiently small constant, there is a constant $\alpha < 1$ such that there exists an explicit construction of an m-input $(\rho, \rho^{\Theta(l \log m)})$-reversal-fault-tolerant m^α-approximate-sorting circuit with $O(l \log^2 m)$ depth and $O(ml \log m)$ size.*

Proof: We use a similar technique as in the proof of Lemma 3.2. First, we apply the partial l-AKS circuit of Theorem 2.1. Then, we apply a set of $(\rho, \rho^{\Theta(l \log m)})$-reversal-fault-tolerant $(l \log m)$-approximate-insertion circuits of Lemma 4.1 in a fashion similar to that in the proof of Lemma 3.2. The assumption $l \leq m^{\frac{1-\beta}{2}}$ implies $l \leq m^{\frac{1}{8}}$, which makes it possible to apply Corollary 2.1 in the first step. In the second step, we can argue that every item is output to within $O(l \log m)\Theta(m^\beta)$ of the correct position. The assumption $l \leq m^{\frac{1-\beta}{2}}$ guarantees that every output item is within $O(l \log m)\Theta(m^\beta) < m^\alpha$ of the correct position for any $\alpha > \frac{1+\beta}{2}$. In particular, the constant α in this lemma is different from that of Lemma 3.2.

By Theorem 2.1 and Lemma 4.1, the depth of the entire circuit is

$$O(l \log m) + O(l \log m \log \frac{m}{l \log m}) = O(l \log^2 m),$$

and the size of the entire circuit is

$$O(ml \log m) + O(ml \log m) = O(ml \log m).$$

■

Lemma 4.3 *For any ρ less than a sufficiently small constant and $l \geq c \log m$, where c is a sufficiently large constant, there exists an explicit construction of an m-input $(\rho, \rho^{\Theta(l)})$-reversal-fault-tolerant l-approximate-sorting circuit with $O(l \log^2 m)$ depth and $O(ml \log^2 m)$ size.*

Proof: We first construct a $(\rho, \rho^{\Theta(l)})$-reversal-fault-tolerant circuit that l-approximate-merges two l-approximate-sorted lists. Given two t-item lists L_1 and L_2, define $L_{1\alpha}$ and $L_{2\alpha}$ to consist of the even-index items of L_1 and L_2 (resp.), and define $L_{1\beta}$ and $L_{2\beta}$ to consist of the odd-index items of L_1 and L_2 (resp.). If L_1 and L_2 are l-approximate-sorted, then so are $L_{1\alpha}$, $L_{2\alpha}$, $L_{1\beta}$, and $L_{2\beta}$. Next recursively l-approximate-merge $L_{1\alpha}$ with $L_{2\beta}$ to form L_α, and $L_{1\beta}$ with $L_{2\alpha}$ to form L_β. By definition, L_α and L_β are l-approximate-sorted. Next, shuffle L_α with L_β to form a list L' that is $2l$-approximate-sorted. Finally, partition L' into contiguous blocks consisting of $16l$ items each, and apply a $16l$-input and $32l$-depth odd-even transposition circuit as described in Lemma 2.1 (with $\vartheta = \frac{1}{32}$) to each block. Finish up by repartitioning the resulting list into contiguous blocks of size $16l$ so that the boundaries of the earlier blocks fall in the centers of the current blocks, and then applying a $16l$-input and $32l$-depth odd-even transposition circuit to each of the blocks.

By an argument similar to that used in the proof of Lemma 4.1, we can show that if all of the odd-even transposition circuits in the entire approximate-merging circuit work correctly as $\frac{l}{4}$-approximate-sorting circuits, then the entire circuit works correctly as an l-approximate-merging circuit. By Lemma 2.1 with $\vartheta = \frac{1}{32}$, the probability that a particular odd-even transposition circuit is not an $\frac{l}{4}$-approximate-sorting circuit is at most $\rho^{\Theta(l)}$. On the other hand, the total number of odd-even transposition circuits in the entire circuit is upper bounded by a polynomial in m. Hence, the probability that all of the odd-even transposition circuits function correctly as $\frac{l}{4}$-approximate-sorting circuits is at least $1 - \text{poly}(m) \cdot \rho^{\Theta(l)} = 1 - \rho^{\Theta(l)}$ when ρ is sufficiently small and $l \geq c \log m$ for a sufficiently large constant c.

The depth of the l-approximate-merging circuit thus constructed is determined by the recurrence

$$M(t) = M\left(\frac{t}{2}\right) + O(l)$$

with the boundary condition $M(t) = \Theta(1)$ for $1 \leq t \leq l$, where $M(t)$ is defined as the depth of a circuit for l-approximate-merging two t-item lists. Solving the recurrence, we find that our circuit has depth $M(t) = O(l \log \frac{t}{l})$.

Given the approximate-merging circuit thus constructed, we can construct the claimed l-approximate-sorting circuit by partitioning the m items to be sorted into two subset of $\frac{m}{2}$ items each, l-approximate-sorting the subsets recursively, and then l-approximate-merging them. This technique leads to the following recurrence for the depth of the l-approximate-sorting circuit

$$D(m) = D\left(\frac{m}{2}\right) + M\left(\frac{m}{2}\right) = D\left(\frac{m}{2}\right) + O\left(l \log \frac{m}{l}\right),$$

with the boundary condition $D(m) = \Theta(1)$ for $1 \leq m \leq l$. Solving the recurrence, we have $D(m) = O(l \log^2 \frac{m}{l})$. Since each level of the circuit contains at most $\frac{m}{2}$ comparators, the size of the circuit is $O(ml \log^2 \frac{m}{l})$. ■

Proof of Theorem 4.3: We construct the circuit in a fashion similar to that of Theorem 3.1. In $O(\log \log n)$ rounds, we repeatedly apply the approximate-sorting circuit of Lemma 4.2 with some appropriately chosen value of l. The ith round consists of two sets of approximate-sorting circuits where the boundaries of the second set of circuits fall in the centers of the first set of circuits. Our hope is that with probability at least $1 - \frac{1}{n^2}$, the ith round outputs every item to within $O(n^{\alpha^i})$ of the correct position (where α is the constant specified in Lemma 4.2) provided that each of the outputs from the $(i-1)$th round is $O(n^{\alpha^{i-1}})$ within the correct position. The detailed parameter choices governing the application of the approximate-sorting circuits in each round are similar to those in the proofs of Theorem 3.1 and Lemma 4.1, and the detailed argument of Lemma 4.1 can be applied to show that our circuit has the desired property provided that all of the constituent approximate-sorting circuits function correctly.

In the proof of Theorem 3.1, we have used the replication technique to guarantee a success probability of at least $1 - \frac{1}{n^2}$ in each round. For reversal faults, however, if we replicate a comparator many times, then the outcome is completely determined by the behavior of the last comparator. Thus, we need a new method to replace the replication technique. Lemma 2.1 will essentially play a role similar to that of the replication technique. Note that we have assumed ρ to be sufficiently small in both Lemma 2.1 and Theorem 2.1. Hence, we can prove the current theorem only under the assumption that ρ is less than a sufficiently small constant.

Returning to the construction of the reversal-fault-tolerant $O(\log n)$-approximate-sorting circuit, let l_i denote our choice of the parameter l in Lemma 4.2 for the ith round, in which we need circuits with $\Theta(n^{\alpha^{i-1}})$ inputs each. We will choose $l_i = \Theta(\frac{1}{\alpha^{i-1}\log\frac{1}{\rho}})$ so that

$$\Theta(l_i \log n^{\alpha^{i-1}}) = \log_{\frac{1}{\rho}} n^3, \tag{82}$$

where the constant behind the Θ-notation is the same as that of Lemma 4.2. This guarantees that the failure probability for the ith round is at most $n\rho^{\Theta(l_i \log n^{\alpha^{i-1}})} \leq \frac{1}{n^2}$. By Lemma 4.2 and equation 82, the depth and the size for the ith round are

$$O\left(l_i \log^2 n^{\alpha^{i-1}}\right) = O\left(\log n \log n^{\alpha^{i-1}}\right) \tag{83}$$

and

$$O\left(nl_i \log n^{\alpha^{i-1}}\right) = O\left(n \log n\right), \tag{84}$$

respectively. In order to apply Lemma 4.2, we stop this procedure immediately before step i for some i such that

$$l_i = \Theta\left(\left(n^{\alpha^{i-1}}\right)^{\frac{1-\beta}{2}}\right), \tag{85}$$

where β is the constant specified in Corollary 2.1. Let i_0 be the least i satisfying equation 85. Simple calculations show that $l_{i_0} = \Theta(\log n)$, $n^{\alpha^{i_0-1}} = \Theta\left(\log^{\frac{2}{1-\beta}} n\right)$, and $i_0 = O(\log\log n)$. Hence every item is within $O(n^{\alpha^{i_0-1}}) = O(\log^{\frac{2}{1-\beta}} n)$ of the correct position at the end of the $(i_0 - 1)$th round.

Given this $O(\log^{\frac{2}{1-\beta}} n)$-approximate-sorted list, we apply the $O(\log n)$-approximate-sorting circuits of Lemma 4.3 with $l = O(\log n)$ to contiguous blocks of $O(\log^{\frac{2}{1-\beta}} n)$ items twice where the boundaries of the second set of circuits fall in the centers of the first set of circuits. (Detailed parameter choices are similar to those obtained in the proof of Lemma 4.1.) The depth and size of this final round are $O(\log n(\log\log n)^2)$ and $O(n\log n(\log\log n)^2)$, respectively.

Now, by expressions 83 and 84, the depth the entire circuit is

$$O\left(\log n(\log\log n)^2\right) + \sum_{1\leq i<i_0} O\left(\log n \log n^{\alpha^{i-1}}\right) = O\left(\log^2 n\right),$$

and the size of the entire circuit is

$$O\left(n\log n(\log\log n)^2\right) + \sum_{1\leq i<i_0} O\left(n\log n\right) = O\left(n\log n(\log\log n)^2\right).$$

■

If we are only interested in a circuit with $O(\log^2 n)$ depth and do not particularly care about its size, there is a much simpler construction that does not depend on Theorem 2.1. In fact, the proof technique of Lemma 2.2 enables us to construct an m-input $(\rho, \frac{1}{n^2})$-reversal-fault-tolerant ϵ-halver with $O(\log n)$ depth for $m = \Omega(\log n)$ and any positive constant ϵ. Replacing each of the ϵ-halvers in the AKS circuit by a reversal-fault-tolerant ϵ-halver constructed in this manner, we can get a reversal-fault-tolerant $O(\log n)$-approximate-sorting circuit with $O(\log^2 n)$ depth relatively easily. Since the fault-tolerance of the l-AKS circuit is very difficult to prove, the existence of the simpler construction with the asymptotically optimal depth is worth mentioning. Nevertheless, we need the fault-tolerance of the l-AKS circuit to achieve the better size bound, which is of crucial importance in Section 4.2.

4.2 Sorting Networks

In this section, we use the reversal-fault-tolerant $O(\log n)$-approximate-sorting circuit designed in Section 4.1 to construct a reversal-fault-tolerant sorting network of $O(n \log^{\log_2 3} n)$ size (for ρ less than a sufficiently small constant). This is the first reversal-fault-tolerant sorting network of $o(n \log^2 n)$ size, and it answers the open question posed by Assaf and Upfal [4]. As pointed out in the introduction, we will assume (as in [4]) that the replicators are fault-free. This is not a particularly unreasonable assumption since replicators can be hard-wired and they do not contain any logic elements.

Theorem 4.4 *For any ρ less than a sufficiently small constant, there exists an explicit construction of a reversal-fault-tolerant sorting network with $O(n \log^{\log_2 3} n)$ size.*

The next lemma addresses how to use a network to compute a certain majority-like function. (As in the case of sorting, the network consists of comparators, which are subject to reversal faults, and replicators; but unlike a sorting network, the result of the computation is output to only one of the many registers in the network.) In particular, we are interested in a network that computes the majority function correctly provided that a large fraction (much larger than $\frac{1}{2}$) of the inputs are either all 0s or all 1s. Formally, for any constant $r \in (\frac{1}{2}, 1)$, we define an r-**MAJORITY function** with n inputs to be a Boolean function that outputs 1 if more than rn of the inputs are 1s and that outputs 0 if more than rn of the inputs are 0s. (We do not care how the function behaves if neither the number of 0s nor the number of 1s exceeds rn.)

Lemma 4.4 ([8]) *For some constant $r \in (\frac{1}{2}, 1)$ and for any constant $\rho < \frac{1}{2}$, there exists an explicit construction of a (ρ, ϵ)-reversal-fault-tolerant network with $O\left((\log_\rho \epsilon)^{\log_2 3}\right)$ size that computes the r-MAJORITY function of $O(\log_\rho \epsilon)$ inputs.*

Proof: See [8, Lemma 3.2]. ■

Proof of Theorem 4.4: Our network consists of two parts. In the first part, we apply the reversal-fault-tolerant $O(n \log n (\log n \log n)^2)$-size $O(\log n)$-approximate-sorting circuit of Theorem 4.3. (We actually need the circuit to be $(\rho, \frac{1}{2n})$-fault-tolerant instead of $(\rho, \frac{1}{n})$-fault-tolerant. This requirement can be satisfied since the circuit of Theorem 4.3 is actually $(\rho, O(\frac{\log^2 n}{n^2}))$-fault-tolerant.) At the end of this part, with probability at least $1 - \frac{1}{2n}$, every item is within $O(\log n)$ of the correct position. Observe that this part of the network does not use any replicators, and does not use more than n registers.

The second part of our network moves all of the items to the correct positions with probability at least $1-\frac{1}{2n}$, provided that each of the inputs to this part is at most $O(\log n)$ away from the correct position. To construct the second part, it suffices to construct an $O(\log n)$-input $(\rho, \frac{1}{n^2})$-reversal-fault-tolerant sorting network with $O(\log n \log^{\log_2 3} n)$ size, since we can finish the second part by applying such networks to contiguous blocks of $O(\log n)$ items twice, where the boundaries of the second set of networks fall in the centers of the first set of networks.

Such an $O(\log n)$-input $(\rho, \frac{1}{n^2})$-reversal-fault-tolerant sorting network can be constructed by using the Assaf-Upfal method [4] with some modifications. Let $\mathcal{C}$ be an $O(\log n)$-input sorting circuit with $O(\log\log n)$ depth, e.g., let $\mathcal{C}$ be an AKS circuit with $O(\log n)$ inputs. Replace each original register r_i by a block of registers $R_i = \{r_{i1}, \ldots, r_{is}\}$ where $s = O(\log n)$. Also, replace each comparator between r_i and r_j by s comparators that connect r_{ik} and r_{jk} for each $k \leq s$. After each set of comparators that correspond to a single level in $\mathcal{C}$, apply the expander-like construction of the so-called **majority preserver** designed in [4] to each of the blocks. For any fixed constant $r < 1$, by carefully choosing the parameters involved in the majority preservers, we can use the argument of [4] to show that with probability at least $1 - \frac{1}{2n^2}$, for all i, more than rs of the items contained in R_i are the same as that contained in r_i in the fault-free case. The details of the construction and its proof of correctness can be found in [4]. We complete the $O(\log n)$-input $(\rho, \frac{1}{n^2})$-reversal-fault-tolerant sorting network by applying (in parallel) the s-input $(\rho, \frac{1}{2n^2})$-reversal-fault-tolerant r-MAJORITY network of Lemma 4.4 to each of the $O(\log n)$-size blocks of the form R_i. ■

5 An Optimal EREW Fault-Tolerant Sorting Algorithm

In this section, we design a fault-tolerant sorting algorithm for the EREW PRAM model. In the PRAM model, as pointed out in the introduction, we assume that faults only occur as incorrect answers to comparison queries and that no item is lost in any comparison. The main result in this section is a fault-tolerant EREW PRAM sorting algorithm that runs in optimal $\Theta(\log n)$ time on n processors. This answers an open question posed by Feige, Peleg, Raghavan, and Upfal [7]. The only previously known $o(\log^2 n)$ time fault-tolerant PRAM sorting algorithm on n processors is a randomized algorithm [7].

Theorem 5.1 *There exists an explicit deterministic fault-tolerant EREW PRAM sorting algorithm with $\Theta(\log n)$ running time on n processors.*

The following lemma is due to Feige, Peleg, Raghavan, and Upfal [7].

Lemma 5.1 ([7]) *There exists an explicit deterministic $(\rho, \rho^{\Theta(\log m)})$-fault-tolerant EREW PRAM algorithm that selects the maximum of m items in $\Theta(\log m)$ time on m processors.*

Proof: See [7, Theorem 20]. ■

Proof of Theorem 5.1: We use the approach of Theorem 3.1, with modifications to achieve the claimed $\Theta(\log n)$ running time (recall that the depth bound in Theorem 3.1 is $O(\log n \log\log n)$).

For any constant ϵ, by simple majority vote, we can easily obtain a comparison scheme with $O(1)$ running time that yields the correct answer with probability at least $1-\epsilon$. In our algorithm, we first use the PRAM to simulate the partial 1-AKS circuit (i.e, the partial l-AKS circuit with $l=1$). In order to apply Corollary 2.1, we need to make sure that the fault probability of each comparison is upper bounded by a sufficiently small constant. This can be achieved by the majority-vote scheme, which only causes a constant slowdown. By Corollary 2.1, all of the outputs from the PRAM simulation of the partial 1-AKS circuit can be partitioned as $X = S \cup X_1 \cup \ldots \cup X_{\overline{n}}$ such that all the items in X_i are smaller than all the items in X_j for $i<j$, where $\overline{n} = \Theta(n^{1-\beta})$.

In the proof of Theorem 3.1, we went on to use the approximate-insertion circuit of Lemma 3.1. Here, in order to achieve the claimed running time, we need a better insertion scheme so that we will not have to deal with the boundary problem that occurred in the proof of Theorem 3.1. In particular, we use the following non-oblivious strategy. In parallel, we apply the selection algorithm of Lemma 5.1 to X_i for each $i \leq \overline{n}$. Let M_i be the maximum item in X_i as reported by the selection algorithm. We use all n processors to sort the set

$$P = S \ \cup \ \{M_1, \ldots, M_{\overline{n}}\}.$$

Since we have n processors and P contains at most $\overline{n} + |S| = O(n^{\frac{3}{4}})$ items, we can sort P with probability at least $1 - \frac{1}{n^2}$ by simulating the m-input, $O(\log m)$-time, and $O(m \log m)$-register $(\rho, \rho^{\Theta(\log m)})$-destructive-fault-tolerant sorting network designed in [4] with $m = \overline{n} + |S|$.

Based on the sorted order of P, we can derive the approximately correct position for each item in S. In particular, we can partition X as

$$X = \bigcup_{1 \leq i \leq \overline{n}} Y_i$$

where $Y_i = X_i \cup \ \{s \in S : M_{i-1} < s \leq M_i\}$ for $i < \overline{n}$, and $Y_{\overline{n}} = X_{\overline{n}} \cup \ \{s \in S : s > M_{\overline{n}-1}\}$ (M_0 is assumed to be $-\infty$). It is easy to see that

$$\Theta(n^{\beta}) = |X_1| \leq |Y_i| \leq |S| + |X_1| \leq O(n^{\frac{3}{4}}) + \Theta(n^{\beta}) = \Theta(n^{\beta}), \tag{86}$$

where β is the constant specified in Corollary 2.1. For $i<j$, all items in Y_i are smaller than all items in Y_j provided that the simulation of the partial 1-AKS circuit and the sorting of P are both done correctly. Note that this output order is "cleaner" than that obtained after the first round of the circuit of Theorem 3.1: in the present case, we have no boundary problems to cope with.

To sort the items within Y_i recursively, we have used in Theorem 3.1 a more and more intensive application of the replication technique for smaller and smaller set of items in order to guarantee that each of the $O(\log \log n)$ rounds works with probability at least $1-\frac{1}{n^2}$, instead of $1 - \frac{1}{O(n^{\alpha^i})}$. To achieve the $O(\log n)$ running time, here we use an adaptive approach to avoid the $O(\log \log n)$ running time blowup caused by this replication technique.

In parallel, we apply a $(\rho, \frac{1}{|Y_i|})$-fault-tolerant EREW sorting algorithm to Y_i for each $i \leq \overline{n}$. A standard Chernoff bound argument [3] shows that the number of unsorted groups of the form Y_i is at most $\overline{n}\frac{1}{|Y_i|} + \overline{n}^{\frac{3}{4}}$ with probability at least $1 - e^{-\frac{\overline{n}^{\frac{2}{4}}}{2}}$. By inequality 86, this means that with probability at most $1-\frac{1}{n^2}$, the number of unsorted groups of the form Y_i is at most $O(\overline{n}^{\frac{3}{4}})$.

We now detect which groups of the form Y_i remain unsorted. For each i in parallel, we first assume that the order for Y_i reported by the recursive sorting algorithm is correct. Then, we check the correctness of the order by repeatedly comparing adjacent items $O(\log n)$ times. With probability at least $1-\frac{1}{n^2}$, we will detect all of the $O(\overline{n}^{\frac{3}{4}})$ unsorted groups. The total number of items contained in the unsorted groups is at most $O(\overline{n}^{\frac{3}{4}}n^{\beta}) = O(n^{\frac{3}{4}(1-\beta)+\beta}) = O(n^{\frac{3+\beta}{4}})$. Since we have n processors to sort the $O(n^{\frac{3+\beta}{4}})$ unsorted numbers, we can proceed by simulating the m-input, $O(\log m)$-time, and $O(m\log m)$-register $(\rho, \rho^{\Theta(\log m)})$-destructive-fault-tolerant sorting network designed in [4] with $m = O(n^{\frac{3+\beta}{4}})$, which succeeds with probability at least $1-\frac{1}{n^2}$,

It is easy to see that the failure probability of the whole algorithm is $O(\frac{1}{n^2}) \le \frac{1}{n}$. Furthermore, this construction leads to the recurrence

$$T(n) = O(\log n) + T\left(O(n^{\beta})\right),$$

where $T(m)$ denotes the time of the optimal $(\rho, \frac{1}{m})$-fault-tolerant EREW sorting algorithm using m processors. Solving this recurrence, we find that $T(n) = O(\log n)$. ■

6 Worst-Case Fault-Tolerance

In this section, we extend our results for random faults to construct worst-case fault-tolerant sorting circuits, networks, and algorithms. All previous work on worst-case faults seems to have focused on passive or destructive faults [12, 18, 19, 20]. We are not aware of any previous work on sorting networks that are tolerant to worst-case reversal faults, and no results were known for PRAM sorting algorithms that are tolerant to worst-case faults. We define a circuit, network, or algorithm to be **k-fault-tolerant** if the circuit, network, or algorithm remains a sorting circuit, network, or algorithm even if any k or fewer comparators (or comparisons in the case of an algorithm) are faulty.

Throughout this section, we will use the following simple scheme to construct k-fault-tolerant circuits, networks, and algorithms: Take a (ρ, ϵ)-fault-tolerant circuit, network, or algorithm where $\epsilon = \rho^{k+1}$. Despite different technical details for different fault and computational models, our hope is that such a circuit, network, or algorithm will be able to tolerate up to k worst-case faults. This is formally proved in the next lemma, where Q should be interpreted as a sorting related problem.

Lemma 6.1 *Let $\mathcal{A}$ be a circuit, network, or algorithm for solving a certain problem Q. If $\mathcal{A}$ is (ρ, ϵ)-fault-tolerant, then $\mathcal{A}$ is k-fault-tolerant for $k = \log_{\rho}\epsilon - 1$.*

Proof: Assume for the purposes of contradiction that $\mathcal{A}$ is not k-fault-tolerant. Then, there exists a set S of k or fewer comparators (or comparisons if $\mathcal{A}$ is an algorithm) such that if all the comparators (or comparisons) in S are faulty and all of the other comparators (or comparisons) not in S are correct, then the resulting faulty version of $\mathcal{A}$ fails to solve the problem Q. On the other hand, if we make each comparator (or comparison) in S to faulty with probability ρ and each comparator (or comparison) not in S faulty with probability 0, then $\mathcal{A}$ fails to solve Q with probability $\rho^{|S|} = \rho^k = \rho^{\log_\rho \epsilon - 1} > \epsilon$. This contradicts the assumption that $\mathcal{A}$ is (ρ, ϵ)-fault-tolerant. ∎

The remainder of Section 6 is organized into four parts. Sections 6.1 to 6.3 contain results for passive faults, reversal faults, and EREW PRAM algorithms, respectively. We conclude in by pointing out in Section 6.4 that all of the results in Section 6 can be proved independently of the fault-tolerance of the l-AKS circuit established in Theorem 2.1.

6.1 Results for Passive Faults

In this section, we construct a k-passive-fault-tolerant sorting circuit with nontrivial depth. Although a tight bound on the size of worst-case passive-fault-tolerant sorting circuits was derived by Yao and Yao [20] in 1985, our result provides the first asymptotically nontrivial upper bound on the depth of such circuits, and is itself asymptotically optimal over a large range of k.

Theorem 6.1 *There exists an explicit construction of a k-passive-fault-tolerant sorting circuit with $O(\log n + k \log \frac{\log n}{\log k})$ depth.*

There is a trivial lower bound of $\Omega(\log n + k)$ on the depth of k-passive-fault-tolerant sorting circuits. The $\Omega(\log n)$ lower bound follows from the trivial $\Omega(\log n)$ lower bound for the fault-free case. In order for a circuit to tolerate k passive-faults, each register r in the circuit has to be connected to at least $k+1$ comparators. (Otherwise, all of the comparators associated with r could be faulty, and the circuit would fail if the item input to r should not be output to r.) This implies a depth lower bound of $\Omega(k)$. Combining the $\Omega(k)$ and $\Omega(\log n)$ lower bounds, we have a depth lower bound of $\Omega(\log n + k)$. Therefore, the upper bound of Theorem 6.1 is actually tight when $k = O(\frac{\log n}{\log \log n})$ or $k = \Omega(n^\alpha)$ for any constant $\alpha > 0$.

In fact, when $k = O(\log n)$, by taking the circuit of Theorem 3.1 and applying Lemma 6.1 with ρ set to a constant, we immediately get a k-passive-fault-tolerant sorting circuit with $O((\log n + k) \log \log n)$ depth. For larger k, we can construct a k-passive-fault-tolerant sorting circuit with $O((\log n + k) \log \log n)$ depth by replicating each comparator of the $O(\log n)$-passive-fault-tolerant sorting circuit $\Omega\left(\frac{k}{\log n}\right)$ times. However, to achieve the better depth bound claimed in Theorem 6.1, we will use a slightly different approach.

Lemma 6.2 *For some constant $\alpha < 1$, there exists an explicit construction of a k-passive-fault-tolerant m-input m^α-approximate-sorting circuit with depth $O(\log m + k)$.*

Proof: An m-input $(\log m)$-passive-fault-tolerant m^α-approximate-sorting circuit can be easily constructed by applying Lemmas 6.1 and 3.2. This proves the theorem for the case $k \leq \log m$. When $k > \log m$, the claimed circuit can be constructed by replicating each comparator of the $(\log m)$-passive-fault-tolerant circuit $\left\lceil \frac{k}{\log m} \right\rceil$ times. ∎

Lemma 6.3 *There exists an explicit construction of an m-input $O(m)$-passive-fault-tolerant sorting circuit with $O(m)$ depth.*

Proof: This follows from Lemmas 2.1 and 6.1. (Recall that as pointed in the proof of Lemma 2.1, for passive faults, the circuit of Lemma 2.1 is actually a $(\rho, \rho^{\Theta(\log m)})$-fault-tolerant circuit for sorting, as opposed to approximate-sorting.) ■

Proof of Theorem 6.1: We first apply the k-passive-fault-tolerant n^α-approximate-sorting circuit of Lemma 6.2 to all n items. The outputs of this circuit form an $O(n^\alpha)$-approximate-sorted list. Then, we apply two sets of $O(n^\alpha)$-input $O(n^{\alpha^2})$-approximate-sorting circuits of Lemma 6.2 so that the boundaries of the second set of circuits fall in the centers of the first set of circuits. We can repeat this construction within smaller and smaller groups until the group size is no more than k, at which point we can finish up by using the circuit of Lemma 6.3. (Detailed parameter choices are similar to those given in the proof of Theorem 3.1.) This construction leads to the following formula on the depth $D(n)$:

$$D(n) = O(k + \log n) + O(k + \log n^\alpha) + \cdots + O(k + \log n^{\alpha^i}) + O(k)$$

where i is the smallest integer such that $O(n^{\alpha^i}) \leq k$. This yields $D(n) = O(\log n + k \log \frac{\log n}{\log k})$. ■

6.2 Results for Reversal Faults

We begin this section by proving that $\Delta \geq k$ for any k-reversal-fault-tolerant Δ-approximate-sorting circuit. Then, we prove a lower bound on the depth of any k-reversal-fault-tolerant k-approximate-sorting circuit. The most important result in Section 6.2 is the construction of a k-reversal-fault-tolerant k-approximate-sorting network with nontrivial size and depth. In light of a lower bound established in [12, Theorem 1], this result separates the size complexities of sorting networks with worst-case reversal and destructive faults.

Theorem 6.2 *For any $k < n$, there is no k-reversal-fault-tolerant $(k-1)$-approximate-sorting circuit.*

Proof: See Appendix C. ■

Theorem 6.3 *Any k-reversal-fault-tolerant k-approximate-sorting circuit has $\Omega(k \log \frac{n}{k})$ depth.*

Proof: See Appendix D. ■

We remark that the above proof can be extended to obtain the same lower bound for the much simpler problem of k-approximate-insertion. This requires slightly more effort, and the proof is omitted.

Theorem 6.4 *There exists an explicit construction of a k-reversal-fault-tolerant k-approximate-sorting circuit with $O(k \log n + k \log^2 k)$ depth and $O(n(\log n + k \log \frac{\log n}{\log k} + k \log^2 k))$ size.*

Lemma 6.4 *There exists an explicit construction of an m-input k-reversal-fault-tolerant k-approximate-insertion circuit with $O(k \log m)$ depth and $O(mk)$ size.*

Proof: By using the proof technique of Lemma 2.1, it is straightforward to prove that for any constant $c > 1$, there exists a ck-input odd-even transposition circuit of $O(k)$ depth that is a k-reversal-fault-tolerant k-approximate-sorting circuit. Using this fact and the argument for Lemma 4.1, it is easy to show that the circuit of Lemma 4.1 with $l = O(k)$ has the desired property. Note that the assumption $l \geq \log \frac{m}{l}$ in Lemma 4.1 was used only in the probabilistic analysis. Hence, this assumption is unnecessary when considering worst-case faults. In fact, for worst-case faults, all of the odd-even transposition circuits involved in the analysis are k-approximate-sorting circuits. ■

Lemma 6.5 *For $k \leq m^{\frac{1-\beta}{2}}$ (where β is the constant specified in Corollary 2.1), there exists an explicit construction of an m-input k-reversal-fault-tolerant m^{α}-approximate-sorting circuit with $O(k \log m)$ depth and $O(m(k + \log m))$ size, where α is some constant strictly less than 1.*

Proof: By Lemma 6.1 and Corollary 2.1, for some $l = O(\max\{\frac{k}{\log m}, 1\})$, an m-input partial l-AKS circuit moves every item, except those in a set S with $O(m^{\frac{3}{4}})$ items, to within $O(m^{\beta})$ of the correct position, even in the presence of up to k worst-case reversal faults. This initial part of our circuit has $O(\log m + k)$ depth and $O(m(\log m + k))$ size. We complete the claimed circuit by applying a set of k-approximate-insertion circuits of Lemma 6.4 in a fashion similar to that used in the proof of Lemma 3.2. This latter part of our circuit has $O(k \log m)$ depth and $O(mk)$ size. ■

Lemma 6.6 *There exists an explicit construction of an m-input k-reversal-fault-tolerant k-approximate-sorting circuit with $O(k \log^2 m)$ depth and $O(mk \log^2 m)$ size.*

Proof: By using the argument of Lemma 4.3, it is not hard to show that the circuit of Lemma 4.3 with $l = k$ is a k-reversal-fault-tolerant $O(k)$-approximate-sorting circuit with $O(k \log^2 m)$ depth and $O(mk \log^2 m)$ size. (The assumption $l \geq c \log m$ was used in the probabilistic argument for Lemma 4.3; such an assumption is not needed for worst-case faults, since we can actually argue that all of the constituent odd-even transposition circuits are k-reversal-fault-tolerant.) Given the $O(k)$-approximate-sorted list thus produced, we can finish up by applying an odd-even transposition circuit of $O(k)$ depth to the entire list. By the argument of Lemma 2.1, we can show that after $O(k)$ steps of the odd-even transposition circuit, every item is within k of the correct position. ■

Proof of Theorem 6.4: Given the circuit of Lemma 6.5, we can construct the claimed circuit in a manner similar to that used in the proof of Theorem 4.3. In particular, we repeatedly apply the circuit of Lemma 6.5 until $m = k^{\frac{2}{1-\beta}}$, at which point we can directly apply the circuit of Lemma 6.6. The depth of the entire circuit is at most

$$\begin{aligned} & O(k \log n) + O(k \log n^{\alpha}) + \cdots + O(k \log n^{\alpha^i}) + \cdots + O(k \log k^{\frac{2}{1-\beta}}) + O(k \log^2 k) \\ = \; & O(k \log n + k \log^2 k). \end{aligned}$$

The size of the entire circuit is at most

$$\begin{aligned} & O(n(k+\log n)) + O(n(k+\log n^{\alpha})) + \cdots + O(n(k+\log n^{\alpha^i})) + \cdots + \\ & + O(n(k+\log k^{\frac{2}{1-\beta}})) + O(nk\log^2 k) \\ = \;& O(n(\log n + k\log\frac{\log n}{\log k} + k\log^2 k)). \end{aligned}$$

■

Theorem 6.5 *There exists an explicit construction of a k-reversal-fault-tolerant sorting network with $O(n(\log n + k\log\frac{\log n}{\log k} + k^{\log_2 3}))$ size.*

The definition of the r-MAJORITY function in the next lemma can be found on page 50.

Lemma 6.7 ([8]) *There exists an explicit construction of a k-reversal-fault-tolerant network with $O(k^{\log_2 3})$ size that computes the r-MAJORITY function with $O(k)$ inputs for some constant $r \in (\frac{1}{2}, 1)$.*

Proof: See [8, Lemma 3.2]. ■

Proof of Theorem 6.5: We first apply the k-approximate-sorting circuit in Theorem 6.4. Then, we use the Assaf-Upfal [4] technique followed by n r-MAJORITY networks with $O(k)$ inputs each, as described in Lemma 6.7, for some constant $r \in (\frac{1}{2}, 1)$. The details are similar to those in the proof of Theorem 4.4. ■

6.3 An Optimal EREW k-Fault-Tolerant Sorting Algorithm

This section contains an optimal k-fault-tolerant EREW PRAM sorting algorithm.

Theorem 6.6 *There exists an explicit EREW PRAM k-fault-tolerant sorting algorithm that runs in asymptotically optimal $\Theta(\log n + k)$ time on n processors.*

Proof: We first prove that any k-fault-tolerant PRAM sorting algorithm on n processors runs in $\Omega(\log n + k)$ time. Since any sorting algorithm uses $\Omega(n\log n)$ comparisons, it is sufficient to prove a lower bound of $\Omega(k)$. In the PRAM model, each processor can only make c comparisons at a single step, where c is a constant. Assume for the purposes of contradiction that a k-fault-tolerant sorting algorithm runs in at most $\frac{k}{c}$ steps. Then there exists an output item x that is involved in k or fewer comparisons. If these k or fewer comparisons for x are all faulty, then we actually know nothing about the rank of x. This is a contradiction.

According to Lemma 6.1, a $(\log n)$-fault-tolerant EREW sorting algorithm $\mathcal{A}$ with $O(\log n)$ running time on n processors can be easily constructed from the algorithm of Theorem 5.1. For $k < \log n$, algorithm $\mathcal{A}$ satisfies all the claimed properties. For $k > \log n$, we construct another algorithm $\mathcal{A}'$ with the claimed property by simulating $\mathcal{A}$, as follows. In $\mathcal{A}'$, we replace each comparison query of $\mathcal{A}$ by $\frac{2k+1}{\log n}$ comparisons, and we use the majority of the answers as the final answer to that query. In order to make an answer to a comparison query in $\mathcal{A}$ be incorrect, an adversary would need to spend more than $\frac{k}{\log n}$ faults for $\mathcal{A}'$. With up to k faults, the adversary can force at most $\frac{k}{\frac{k}{\log n}} = \log n$ incorrect comparison answers in $\mathcal{A}$. Since $\mathcal{A}$ can tolerate up to $\log n$ faults, $\mathcal{A}'$ works correctly with up to k faults. Since we simulate each comparison of $\mathcal{A}$ by $\frac{2k+1}{\log n}$ comparisons in $\mathcal{A}'$, the running time of $\mathcal{A}'$ is $\Theta(\frac{k}{\log n})$ times the running time of $\mathcal{A}$. Hence, algorithm $\mathcal{A}'$ runs in $\Theta(k)$ time. ■

6.4 Remarks

Thus far, all of our worst-case fault-tolerant constructions of circuits, networks, and algorithms have been based on the corresponding constructions for random faults. Since our results for random faults are dependent on the fault-tolerance properties of the l-AKS circuit proved in Theorem 2.1, our results for worst-case faults are also dependent on Theorem 2.1.

Given the difficulty of proving Theorem 2.1, it is worth pointing out that all of our results for worst-case faults can be proved independently of Theorem 2.1. For this purpose, the substitute for Theorem 2.1 is a lemma that is essentially due to Yao and Yao [20], who proved the lemma for passive faults. In what follows, we define the Hamming distance $D(\mathbf{x}, \mathbf{y})$ of any two 0-1 sequences $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$ as the number of positions where $\mathbf{x}$ and $\mathbf{y}$ differ. For any (possibly faulty) circuit $\mathcal{C}$, we use the notation $\mathcal{C}(\mathbf{x})$ to denote the 0-1 sequence that is output by $\mathcal{C}$ on input $\mathbf{x}$. Yao and Yao's lemma states that for any circuits $\mathcal{C}$ and $\mathcal{C}^*$, where $\mathcal{C}^*$ is a faulty version of $\mathcal{C}$ with at most k passive or reversal faults, $D(\mathcal{C}(\mathbf{x}), \mathcal{C}^*(\mathbf{x})) \leq 2k$ where $\mathbf{x}$ is any 0-1 sequence in $\{0,1\}^n$. Even though Yao and Yao considered passive faults only, their proof can actually be extended to deal with reversal faults. Given Yao and Yao's lemma, we can construct circuits, networks, and algorithms by an approach used in [13]. In particular, we will need circuits that isolate extreme items into a small group of registers. Such circuits can be built by adapting the fault-tolerant minimum-finding algorithm of [7] (see Lemma 5.1), or by adapting the passive-fault-tolerant minimum-finding circuit of [6]. The details are not simple and are omitted from this paper.

Acknowledgements

The authors would like to thank Yiqun Yin for suggesting the use of the probabilistic method for a particular problem that arose in the early stages of this research. Her suggestion eventually motivated the use of the potential function that plays a very important role in the analysis of the AKS circuit.

References

[1] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log n$ parallel steps. *Combinatorica*, 3:1–19, 1983. See also the conference version, which appears in *Proceedings of the 15th Annual ACM Symposium on the Theory of Computing*, pages 1–9, May 1983.

[2] V. E. Alekseyev. Sorting algorithms with minimum memory. *Kibernetika*, 5:99–103, 1969.

[3] N. Alon and J. H. Spencer. *The Probabilistic Method.* Wiley-Interscience, New York, NY, 1991.

[4] S. Assaf and E. Upfal. Fault-tolerant sorting network. In *Proceedings of the 31st Annual IEEE Symposium on Foundations of Computer Science*, pages 275–284, October 1990.

[5] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference*, volume 32, pages 307–314, 1968.

[6] P. Donejko, K. Diks, A. Pelc, and M. Piotrów. Reliable minimum finding comparator networks. In I. Prívara, B. Rovan, and P. Ružička, editors, *Proceedings of the 19th Symposium on Mathematical Foundations of Computer Science (Košice, Slovakia, August 1995)*, LNCS 841, pages 306–315, New York, 1994. Springer-Verlag.

[7] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Computing with unreliable information. In *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 128–137, May 1990.

[8] D. Kleitman, T. Leighton, and Y. Ma. On the design of Boolean circuits that contain partially unreliable gates. In *Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 332–346, November 1994.

[9] D. E. Knuth. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison-Wesley, Reading, MA, 1973.

[10] T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes.* Morgan-Kaufmann, San Mateo, CA, 1992.

[11] T. Leighton and Y. Ma. Breaking the $\Theta(n \log^2 n)$ barrier for sorting with faults. In *Proceedings of the 34th Annual IEEE Symposium on Foundations of Computer Science*, pages 734–743, November 1993.

[12] T. Leighton and Y. Ma. Tight bounds on the size of fault-tolerant merging and sorting networks with destructive faults. In *Proceedings of the 5th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 30–41, July 1993.

[13] T. Leighton, Y. Ma, and C. G. Plaxton. Highly fault-tolerant sorting circuits. In *Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science*, pages 458–469, October 1991.

[14] T. Leighton and B. Maggs. Expanders might be practical: Fast algorithms for routing around faults in multibutterflies. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*, pages 458–469, October 1990.

[15] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.

[16] M. S. Paterson. Improved sorting networks with $O(\log N)$ depth. *Algorithmica*, 5:75–92, 1990.

[17] N. Pippenger. On networks of noisy gates. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*, pages 30–36, October 1985.

[18] L. Rudolph. A robust sorting network. *IEEE Transactions on Computers*, 34:344–354, 1985.

[19] M. Schimmler and C. Starke. A correction network for N-sorter. *SIAM J. Comput.*, 18:1179–1187, 1989.

[20] A. C. Yao and F. F. Yao. On fault-tolerant networks for sorting. *SIAM J. Comput.*, 14:120–128, 1985.

A Proof of Theorem 4.1

As pointed out in the introduction, to get the strongest possible result, ρ should be interpreted as the failure probability of each comparator in the circuit, as opposed to an upper bound on the failure probability. Throughout the proof, we further assume that $\rho \leq \frac{1}{2}$. This assumption does not affect the generality of the proof, since a comparator failure probability of ρ can be viewed as failure probability of $1-\rho$ if the MIN/MAX output assignment is reversed. We first prove the following useful lemma.

Lemma A.1 *For any m-input circuit $\mathcal{C}'$, where m is sufficiently large (depending on ρ), the probability that a randomly faulty version of $\mathcal{C}'$ is a Δ-approximate-sorting circuit is at most $1-\rho^{2\Delta+1}$ for $\Delta < \frac{m-1}{2}$.*

The lemma states that in the reversal fault model, there is an inherent limitation on the success probability that can be achieved by any approximate-sorting circuit. The lemma also expresses a tradeoff between the accuracy of approximation and the degree of reliability that can be achieved by any circuit with reversal faults. Such a tradeoff will be used to prove Theorem 4.1.

For ease of notation, we next introduce the notion of a register segment as follows. As described in the introduction, the comparators in a circuit are partitioned into disjoint levels. We assume that the comparators nearest to the inputs are at level 1. Also, we assume that the inputs to a circuit are provided at level 0 and the outputs are produced at level $d+1$, where d denotes the depth of the circuit. A **register segment** is defined to be the part of a register between two consecutive levels. Note that each level is used to partition all of the registers into corresponding register segments, even in those cases where the number of comparators in the level is less than half the number of registers. Hence, an m-input d-depth circuit contains $(d+1)m$ register segments, regardless of its size.

We next prove Lemma A.1. Since the lemma is concerned with the functionality of $\mathcal{C}'$, and not the depth of $\mathcal{C}'$, we may assume without loss of generality that $\mathcal{C}'$ contains only one comparator at each level. We label the registers in $\mathcal{C}'$ as $r_1, \ldots, r_m$, from top to bottom. Let $r_{i,j}$ be the register segment of r_i between levels j and $j+1$ for $j = 0, \ldots, d$, where d denotes the depth of $\mathcal{C}'$. For $i = 1, \ldots, m$ and $j = 0, \ldots, d$, define $P_{i,j}$ to be the probability that 1 is contained in $r_{i,j}$ when a uniformly chosen permutation of $\{1, \ldots, m\}$ is fed into a randomly faulty version of $\mathcal{C}'$.

For any index set $I \subset \{1, \ldots, m\}$ such that $|I| < m$, consider the inequality

$$\sum_{i \in I} P_{i,j} \leq 1 - \rho^{|I|}. \tag{87}$$

It is easy to see that Lemma A.1 follows from inequality 87 with $j = d$ and $|I| = 2\Delta + 1$. We now prove inequality 87 by induction on j.

Base case: $j = 0$. On a uniformly chosen input permutation of $\{1, 2, \ldots, m\}$, each input register segment contains 1 with probability $\frac{1}{m}$. Hence, we need to verify that $\frac{|I|}{m} \le 1 - \rho^{|I|}$ for any I such that $|I| < m$. It suffices to show that

$$f(x) = 1 - \rho^x - \frac{x}{m} \ge 0 \tag{88}$$

for $0 \le x \le m - 1$. Clearly, $f'(x) = -\rho^x \log_e \rho - \frac{1}{m}$ and $f'(x) = 0$ has a unique root in $[0, m-1]$ when m is large. Moreover, for m sufficiently large (depending on ρ), $f'(0) > 0$ and $f'(m-1) < 0$. Hence, inequality 88 follows from: (i) $f(0) = 0$, and (ii) $f(m-1) = 1 - \rho^{m-1} - \frac{m-1}{m} = \frac{1}{m} - \rho^{m-1} > 0$ for m sufficiently large (depending on ρ).

Inductive step. Assuming that inequality 87 holds up to $j - 1$, we next prove that inequality 87 holds for j. By assumption, $\mathcal{C}'$, there is only one comparator at level j in $\mathcal{C}'$. Assume that this comparator connects two registers r_{i_1} and r_{i_2}. Clearly, if I contains both i_1 and i_2, or if I contains neither i_1 nor i_2, then $\sum_{i \in I} P_{i,j} = \sum_{i \in I} P_{i,j-1}$ and the correctness of inequality 87 for j follows from the induction hypothesis. Hence, we can assume without loss of generality that $i_1 \in I$ and $i_2 \notin I$. Let $I' = I - \{i_1\}$. By the definition of reversal faults and the fact that $\rho \le \frac{1}{2}$,

$$\begin{aligned}
\textstyle\sum_{i \in I} P_{i,j} &\le \textstyle\sum_{i \in I'} P_{i,j-1} + (1 - \rho)\left(P_{i_1,j-1} + P_{i_2,j-1}\right) \\
&\le \textstyle\sum_{i \in I'} P_{i,j-1} + (1 - \rho)\left(1 - \sum_{i \in I'} P_{i,j-1}\right) \\
&= \textstyle\rho \sum_{i \in I'} P_{i,j-1} + 1 - \rho \\
&\le \rho\left(1 - \rho^{|I'|}\right) + 1 - \rho \qquad \text{(by the induction hypothesis)} \\
&= 1 - \rho^{|I|}.
\end{aligned}$$

This concludes the inductive proof of inequality 87, as well as the proof of Lemma A.1.

In what follows, let

$$m = \frac{1-\gamma}{3} \log_{\frac{1}{\rho}} n + 1, \tag{89}$$

and

$$\Delta = \left\lfloor \frac{m-2}{2} \right\rfloor. \tag{90}$$

We next consider an arbitrary n-input circuit $\mathcal{C}$, and prove the theorem by showing that the probability a randomly faulty version of $\mathcal{C}$ is a Δ-approximate-sorting circuit is at most $e^{-\frac{n^\gamma}{\log \frac{1}{\rho}}}$.

Roughly speaking, we will use Lemma A.1 to prove Theorem 4.1, as follows. Consider an input sequence consisting of $\frac{n}{m}$ groups of size m such that all items in the ith group are smaller than all items in the jth group for $i < j$. If there is no comparator between items in different groups (intuitively, such comparisons provide no additional information), then each of the $\frac{n}{m}$ groups will be sorted by $\frac{n}{m}$ independent smaller circuits. By Lemma A.1, the probability that all of these groups will be Δ-approximate-sorted is upper bounded by

$$\left(1 - \rho^{2\Delta+1}\right)^{\frac{n}{m}} \le \left(1 - \rho^{m-1}\right)^{\frac{n}{m}} = \left(1 - n^{-\frac{1-\gamma}{3}}\right)^{\frac{n}{m}} \le \left(e^{-n^{-\frac{1-\gamma}{3}}}\right)^{\frac{n}{m}} \le e^{-\frac{n^\gamma}{\log \frac{1}{\rho}}}$$

for n sufficiently large depending on ρ and γ. To deal with the dependence problem caused by comparisons between items in different groups, it will be convenient to use some terminology introduced by Leighton and Ma in [12].

We define a **fault pattern** to be all of the information needed to specify the functionality of all comparators in $\mathcal{C}$, i.e., a fault pattern completely describes which comparators in $\mathcal{C}$ are faulty and which are not faulty. Given a fault pattern F, we use $\mathcal{C}(F)$ to denote the faulty version of $\mathcal{C}$ in which the (correct or faulty) behavior of each comparator is determined according to F. Let

$$\pi = \left(\underbrace{1, \ldots, 1}_{m}, \underbrace{2, \ldots, 2}_{m}, \ldots, \underbrace{\left\lceil \frac{n}{m} \right\rceil - 1, \ldots, \left\lceil \frac{n}{m} \right\rceil - 1}_{m}, \underbrace{\left\lceil \frac{n}{m} \right\rceil, \ldots, \left\lceil \frac{n}{m} \right\rceil}_{n-m\left(\lceil \frac{n}{m} \rceil - 1\right)} \right).$$

We define a comparator of $\mathcal{C}(F)$ to be a **crossing comparator** if it compares two distinct items when $\mathcal{C}(F)$ is executed on input sequence π. We use $\mathit{Cross}(F)$ to denote the set of all crossing comparators of $\mathcal{C}(F)$. We label the n registers as $r_1, \ldots, r_n$, from top to bottom. We next describe a procedure to decompose $\mathcal{C}(F)$ into $\left\lceil \frac{n}{m} \right\rceil$ smaller circuits, $\mathcal{C}(F)_1, \ldots, \mathcal{C}(F)_{\lceil \frac{n}{m} \rceil}$. For $i = 1, \ldots, \lceil \frac{n}{m} \rceil$, we construct $\mathcal{C}(F)_i$ by applying the following 4-step procedure: (i) input π to $\mathcal{C}(F)$ and observe how the input items move within $\mathcal{C}(F)$; (ii) include all register segments (at all levels) of $\mathcal{C}$ that receive the item i; (iii) include all comparators for which both inputs receive item i; (iv) for any crossing comparator that causes item i to move from $r_{u,j}$ to $r_{v,j+1}$ for some j and $u \neq v$, directly connect $r_{u,j}$ and $r_{v,j+1}$ in $\mathcal{C}(F)_i$. Due to the direct connections introduced in step (iv), $\mathcal{C}(F)_i$ may contain some "twisted" registers, and look abnormal. However, these abnormalities will not prevent us from applying Lemma A.1 to $\mathcal{C}(F)_i$. (As a matter of fact, we can construct a circuit that looks "normal" and that is equivalent to $\mathcal{C}(F)_i$, but we do not need this fact to apply Lemma A.1.)

We next use an argument based on conditional probabilities to show that for a random fault pattern F, with the probability claimed in Theorem 4.1, $\mathcal{C}(F)$ is not a Δ-approximate-sorting circuit. In particular, we will organize all of the fault patterns into groups and prove that for a randomly generated fault pattern F within each group, with the claimed probability, $\mathcal{C}(F)$ is not a Δ-approximate-sorting circuit.

We organize all of the fault patterns into groups according to $\mathit{Cross}(F)$, as follows. For any two fault patterns F and F', we put F and F' in the same group if and only if $\mathit{Cross}(F) = \mathit{Cross}(F')$. Within each group thus constructed, we choose an arbitrary fault pattern as a representative of that group. Then, we list all of the representatives as $F_1, \ldots, F_t$ for some t. To prove the theorem, we only need to show that for each $s \leq t$, for a randomly generated F such that $\mathit{Cross}(F) = \mathit{Cross}(F_s)$, the probability that $\mathcal{C}(F)$ is a Δ-approximate-sorting circuit is at most $e^{\frac{-n^\gamma}{\log \frac{1}{\rho}}}$. The reason such a conditional probability is easier to analyze is as follows: By definitions, the information of F on $\mathit{Cross}(F)$ completely determines the decomposition of $\mathcal{C}(F)$ into $\mathcal{C}(F)_1, \ldots, \mathcal{C}(F)_{\lceil \frac{n}{m} \rceil}$. Hence, full information of F on $\mathit{Cross}(F)$ completely determines the constructions of $\mathcal{C}(F)_1, \ldots, \mathcal{C}(F)_{\lceil \frac{n}{m} \rceil}$ (although not the functionality of these circuits), and will allow us to apply Lemma A.1 on the $\left\lfloor \frac{m}{n} \right\rfloor$ independent smaller circuits $\mathcal{C}(F)_1, \ldots, \mathcal{C}(F)_{\lfloor \frac{n}{m} \rfloor}$. Now, for each $s \le t$, we have

$$\begin{aligned}
& \Pr\left(\mathcal{C}(F) \text{ is a } \Delta\text{-approximate-sorting circuit given } \mathit{Cross}(F) = \mathit{Cross}(F_s)\right) \\
\le\ & \Pr\left(\mathcal{C}(F)_i \text{ is a } \Delta\text{-approximate-sorting circuit for } i \le \left\lfloor \tfrac{n}{m} \right\rfloor \text{ given } \mathit{Cross}(F) = \mathit{Cross}(F_s)\right) \\
=\ & \textstyle\prod_{1 \le i \le \lfloor \frac{n}{m} \rfloor} \Pr\left(\mathcal{C}(F)_i \text{ is a } \Delta\text{-approximate-sorting circuit}\right) \\
& \qquad\qquad\qquad \text{(since the behaviors of the } \mathcal{C}(F)_i\text{'s are mutually independent)} \\
\le\ & (1 - \rho^{m-1})^{\lfloor \frac{n}{m} \rfloor} \qquad\qquad \text{(by Lemma A.1 and equation 90)} \\
\le\ & \left(1 - n^{-\frac{1-\gamma}{3}}\right)^{\lfloor \frac{n}{m} \rfloor} \qquad\qquad \text{(by equation 89)} \\
\le\ & \left(e^{-n^{-\frac{1-\gamma}{3}}}\right)^{\lfloor \frac{n}{m} \rfloor} \qquad\qquad \text{(by the inequality } 1 - x \le e^{-x}\text{)} \\
\le\ & e^{-\frac{n^{\gamma}}{\log \frac{1}{\rho}}} \qquad\qquad \text{(for } n \text{ sufficiently large, depending on } \rho \text{ and } \gamma\text{)}.
\end{aligned}$$

This completes the proof of Theorem 4.1.

B Proof of Theorem 4.2

As pointed out in the introduction, to get the strongest possible result, ρ should be interpreted as the failure probability of each comparator in the circuit, as opposed to an upper bound on the failure probability. In fact, if ρ were interpreted as such an upper bound, the correctness of Theorem 4.2 would follow from Theorem 6.3 with $k = O(\log n)$.

In what follows, we assume that

$$\epsilon < \frac{1}{2}. \tag{91}$$

Let

$$\Lambda = \left\lceil \log_4 \left(\frac{n-1}{2\Delta}\right) \right\rceil - 1, \tag{92}$$

and

$$\mu = \left\lfloor \log_{\rho} 2\epsilon \right\rfloor. \tag{93}$$

The correctness of Theorem 4.1 follows from the next lemma with $\epsilon = n^{-c}$ and $\Delta = n^{\gamma}$.

Lemma B.1 *Any (ρ, ϵ)-reversal-fault-tolerant Δ-approximate-insertion circuit has depth strictly greater than $\Lambda\mu$.*

We now prove Lemma B.1. Consider an arbitrary circuit $\mathcal{C}$ that is a (ρ, ϵ)-reversal–fault-tolerant Δ-approximate-insertion circuit, and apply one of the two input permutations $(1, 2, \ldots, n)$ and $(n, 1, 2, \ldots, n-1)$, each with probability $\frac{1}{2}$, to a randomly faulty version of $\mathcal{C}$. Note that the item contained in each register segment is a random variable depending on the random input permutation and the random faults assigned to $\mathcal{C}$. (Recall that, as defined in Appendix A, a register segment is the part of a register between two consecutive levels in the circuit.)

For each register segment x, let X denote the random variable corresponding to the item received by that register segment, and let $\mathit{label}(x)$ in $[1, n]$ be chosen to satisfy

$$\Pr(X \ge \mathit{label}(x)) \ge \frac{1}{2}$$

and

$$\Pr(X \leq \mathit{label}(x)) \geq \frac{1}{2}.$$

Note that *label*(x) always exists but may not be unique.

Define a register segment y to be a **descendant** of a register segment x if the item received by x could subsequently enter y in some faulty or nonfaulty version of $\mathcal{C}$. Also, define the **descendant set** $D(x, i)$ of x immediately after level i to be the set of all descendants of x between levels i and $i + 1$.

For any i such that $0 \leq i \leq \Lambda$, and for any pair of numbers, r_1 and r_2, such that

$$r_2 - r_1 \geq \frac{n-1}{4^i} > 2\Delta,$$

we make the following pair of definitions. First, a register segment x between levels $i\mu$ and $i\mu + 1$ is defined to be (r_1, r_2)-**bad** if and only if x receives an item less than or equal to r_1 with probability at least ϵ, and receives an item greater than or equal to r_2 with probability at least ϵ. Second, a descendant set $D(x, i\mu)$ is defined to be (r_1, r_2)-**bad** if and only if one of the following two conditions holds:

(i) x receives an item less than or equal to r_1 with probability at least ϵ, and the label of every register segment in $D(x, i\mu)$ is greater than or equal to r_2, or

(ii) x receives an item greater than or equal to r_2 with probability at least ϵ, and the label of every register segment in $D(x, i\mu)$ is less than or equal to r_1.

Claim B.1 *No output register segment of $\mathcal{C}$ can be (r_1, r_2)-bad. Furthermore, if $\mathcal{C}$ has depth $\Lambda\mu$, then $\mathcal{C}$ cannot contain an (r_1, r_2)-bad descendant set $D(x, \Lambda\mu)$ for any register segment x.*

The first part of the claim is straightforward since $\mathcal{C}$ is a (ρ, ϵ)-reversal-fault-tolerant Δ-approximate-insertion circuit. Now consider the second part of the claim. Assume for the purposes of contradiction that $\mathcal{C}$ has depth of $\Lambda\mu$ and contains an (r_1, r_2)-bad descendant set $D(x, \Lambda\mu)$ for some x. Without loss of generality, we can assume that the first condition in the definition of a descendant set holds (the other case can be handled by an entirely symmetric argument). Under this assumption, with probability at least ϵ, x receives an item less than or equal to r_1. This item will eventually enter some register segment in $D(x, \Lambda\mu)$. On the other hand, every register segment in $D(x, \Lambda\mu)$ receives an item greater than or equal to r_2 with probability at least $\frac{1}{2} > \epsilon$. Hence, as required by the insertion problem, every register segment y in $D(x, \Lambda\mu)$ is assigned an output rank greater than or equal to $r_2 - \Delta$. But if any register segment in $D(x, \Lambda\mu)$ receives an item less than or equal to r_1, then $\mathcal{C}$ will fail since $(r_2 - \Delta) - r_1 > \Delta$. This proves the claim.

Claim B.2 *For any i such that $0 \leq i \leq \Lambda$, the circuit $\mathcal{C}$ contains either an (r_1, r_2)-bad register segment between levels $i\mu$ and $i\mu + 1$, or an (r_1, r_2)-bad descendant set $D(x, i\mu)$ for some register segment x.*

We prove the claim by induction on i. For the base case, $i = 0$, note that the top input register segment x receives 1 with probability $\frac{1}{2} > \epsilon$, and n with probability $\frac{1}{2} > \epsilon$. Thus, x is $(1, n)$-bad.

Now assume that the lemma holds for $i = j$, $0 \le j < \Lambda$, and consider the induction step, $i = j + 1$. Let $s = \frac{n-1}{4^j}$.

We first argue that if there is a register segment x between levels $j\mu$ and $j\mu + 1$, and a register segment y belonging to $D(x, (j+1)\mu)$, such that $|label(y) - label(x)| \ge \frac{s}{4}$, then y is either $(label(x), label(y))$-bad or $(label(y), label(x))$-bad. To see this, assume without loss of generality that $label(y) - label(x) \ge \frac{s}{4}$ (the case where $label(x) - label(y) \ge \frac{s}{4}$ can be handled by an entirely symmetric argument). Note that y receives an item less than or equal to $label(x)$ with probability at least $\frac{\rho^\mu}{2}$, which is at least ϵ by equation 93. Furthermore, y receives an item greater than or equal to $label(y)$ with probability at least $\frac{1}{2} > \epsilon$. Hence, y is $(label(x), label(y))$-bad, since

$$label(y) - label(x) \ge \frac{s}{4} = \frac{n-1}{4^{j+1}} \ge \frac{n-1}{4^\Lambda} > 2\Delta \tag{94}$$

by equation 92.

By the argument of the preceding paragraph, we now have the additional assumption that for every register segment x between levels $j\mu$ and $j\mu + 1$, and every register segment y belonging to $D(x, (j+1)\mu)$, $|label(y) - label(x)| < \frac{s}{4}$. Proceeding with the induction step, there are two cases to consider.

Case 1: There is an (r_1, r_2)-bad register segment x at level $j\mu$. Let S denote the set of labels associated with the register segments in $D(x, (j+1)\mu)$. By our additional assumption, every pair of labels in S differ by strictly less than $\frac{s}{2}$. On the other hand, $r_2 - r_1 \ge \frac{n-1}{4^j} = s$. Hence, either every label in S exceeds r_1 by at least $\frac{s}{4}$, or r_2 exceeds every label in S by at least $\frac{s}{4}$. Therefore, $D(x, (j+1)\mu)$ is either $(r_1, r_1 + \frac{s}{4})$-bad or $(r_2 - \frac{s}{4}, r_2)$-bad since $\frac{s}{4} > 2\Delta$ by equation 94.

Case 2: There is an (r_1, r_2)-bad descendant set $D(x, j\mu)$. We will assume that the first condition in the definition of an (r_1, r_2)-bad descendant set is satisfied; the other case can be handled by an entirely symmetric argument. Each register segment z in $D(x, (j+1)\mu)$ is a descendant of some register segment y in $D(x, j\mu)$. By our additional assumption, $label(y) - label(z) < \frac{s}{4}$. Furthermore, by the first condition in the definition of an (r_1, r_2)-bad descendant set, $label(y) \ge r_2$. Hence, $label(z) \ge r_2 - \frac{s}{4}$, and $D(x, (j+1)\mu)$ is $(r_1, r_2 - \frac{s}{4})$-bad. This concludes the induction step for the proof of Claim B.2.

Finally, it is immediate that any circuit of depth d (not necessarily a multiple of μ) can be viewed as a circuit of depth $\left\lceil \frac{d}{\mu} \right\rceil \mu$ with the same behavior. Hence, the correctness of Lemma B.1 follows from Claims B.1 and B.2. This completes the proof of Theorem 4.2.

C Proof of Theorem 6.2

We focus on an arbitrary n-input circuit $\mathcal{C}$ and show that some faulty version of $\mathcal{C}$ with at most k faults cannot $(k-1)$-approximate-sort. Since the theorem is concerned with the functionality of $\mathcal{C}$, and not the depth of $\mathcal{C}$, we can assume without loss of generality that each level of $\mathcal{C}$ contains only one comparator. We use the notions of fault patterns and register segments as defined on pages 1 and 60, respectively. Also, we use $\mathcal{C}(F)$ to denote the faulty version of $\mathcal{C}$ specified by fault pattern F. We label the n registers of $\mathcal{C}$ as $r_1, \ldots, r_n$, from top to bottom. Let $r_{i,j}$ denote the register segment of r_i between levels j and $j+1$ for $j = 0, \ldots, d$, where d is the depth of $\mathcal{C}$.

In what follows, we assume that x is the item that is supposed to be output to register r_1 when a permutation of $\{1, \ldots, n\}$ is input to $\mathcal{C}$. Let $I_k(j)$ denote the set of all indices i such that there exists a fault pattern F with k or fewer faults, and a permutation π of $\{1, \ldots, n\}$, such that $r_{i,j}$ contains x when π is input to $\mathcal{C}(F)$. We next prove that

$$|I_k(j)| \geq k+1 \qquad \text{for } k < n. \tag{95}$$

The fact that some faulty version of $\mathcal{C}$ does not $(k-1)$-approximate-sort follows from the preceding inequality with $j = d$, since x is supposed to be output to r_1 and there are at most k registers within $k-1$ of r_1.

We prove inequality 95 by induction on j. The base case $j = 0$ is trivial, because each of the n input register segments can contain item x.

Assuming that inequality 95 holds up to $j-1$, we prove that inequality 95 holds for j. Assume

$$I_k(j-1) = \{i_1, \ldots, i_s\}.$$

By the induction hypothesis, $s \geq k+1$. Assume that the unique comparator at level j connects registers r_u and r_v. Since the comparator between r_u and r_v is the only comparator at level j, we have

$$I_k(j) \supseteq I_k(j-1) - \{u\} \quad \text{or} \quad I_k(j) \supseteq I_k(j-1) - \{v\}. \tag{96}$$

There are three cases to consider.

Case 1: $u \notin I_k(j-1)$ and $v \notin I_k(j-1)$. In this case, $I_k(j) \supseteq I_k(j-1)$, and inequality 95 follows from the induction hypothesis.

Case 2: Exactly one of u and v is in $I_k(j-1)$. Without loss of generality, we assume that $u \in I_k(j-1)$ and $v \notin I_k(j-1)$. Hence, on some input permutation, we can force $r_{u,j-1}$ to contain x by using up to k faults before and including level $j-1$. This will cause either $r_{u,j}$ or $r_{v,j}$ to contain x. Hence, by relation 96, either $I_k(j) \supseteq I_k(j-1)$ or $I_k(j) \supseteq I_k(j-1) \cup \{v\} - \{u\}$. In either case, inequality 95 holds for j.

Case 3: $u \in I_k(j-1)$ and $v \in I_k(j-1)$. In this case, $I_k(j-1) \supseteq I_{k-1}(j-1) \cup \{u, v\}$. Hence, if neither u nor v is in $I_{k-1}(j-1)$, then

$$|I_k(j-1)| \geq |I_{k-1}(j-1)| + 2 \geq k+2,$$

where the second inequality follows from the induction hypothesis. This, together with relation 96, implies that inequality 95 holds for j. Therefore, we only need to check the case where either u or v is in $I_{k-1}(j-1)$. Without loss of generality, we assume that $u \in I_{k-1}(j-1)$. Under this assumption, on some input permutation, we can force $r_{u,j-1}$ to contain x by spending up to $k-1$ faults before and including level $j-1$. Then, by setting the unique comparator at level j to be either faulty or correct, we can force x to enter either $r_{u,j}$ or $r_{v,j}$, whichever we desire. This means $u \in I_k(j)$ and $v \in I_k(j)$. Hence, by relation 96, $I_k(j) \supseteq I_k(j-1)$, which together with the induction hypothesis implies the correctness of inequality 95 for j. This completes the inductive proof of inequality 95 as well as the proof of Theorem 6.2.

D Proof of Theorem 6.3

Let $\mathcal{C}$ be a k-reversal-fault-tolerant k-approximate-sorting circuit with depth d. We will use the notion of register segments as defined in Appendix A. In particular, let $r(i,l)$ be the register segment of $\mathcal{C}$ that contains i between levels l and $l+1$ when the identity permutation $(1,\ldots,n)$ is input to the non-faulty version of $\mathcal{C}$. For example, $r(i,0)$ is the ith input register segment. For ease of notation, we assume that $r(i,l) = r(i,0)$ for $l < 0$.

Claim D.1 *If circuit $\mathcal{C}$ contains at most k faults, then for any integers $l \geq 0$ and $i \in [1,n]$, $r(i, d-lk)$ can only contain an item in $[i-2^l k, i+2^l k]$.*

We will prove the claim by induction on l. For the base case, $l = 0$, the claim is true since (the nonfaulty) $\mathcal{C}$ should k-approximate-sort all permutations, including the identity permutation. Assuming that the claim holds for $l-1$, we now prove the claim for l.

Recall that as defined in Appendix B, a register segment y is a descendant of a register segment x if and only if the item contained in x could subsequently enter y in some faulty or non-faulty version of $\mathcal{C}$. Also, $D(x,j)$ denotes the set of descendants of x between levels j and $j+1$. We next prove that

$$D\left(r(i,d-lk), d-(l-1)k\right) \subseteq \{r(j, d-(l-1)k) : j \in [i-2^{l-1}k, i+2^{l-1}k]\}. \qquad (97)$$

Assume for the purposes of contradiction that the above inequality does not hold. Then, there exists

$$j \notin [i-2^{l-1}k, i+2^{l-1}k] \qquad (98)$$

such that $r(j, d-(l-1)k)$ is a descendant of $r(i, d-lk)$. Hence, there is a path from $r(i,d-lk)$ to $r(j, d-(l-1)k)$ that spans k levels in $\mathcal{C}$. On the other hand, $r(i, d-lk)$ receives i when the identity permutation is input to any version of $\mathcal{C}$ that is fault-free in the first $d-lk$ levels. By forcing each of the k or fewer comparators along the path from $r(i,d-lk)$ to $r(j, d-(l-1)k)$ to be faulty or correct, as appropriate, we can force $r(j, d-(l-1)k)$ to receive i. However, by the induction hypothesis, $r(j, d-(l-1)k)$ can only receive an item in $[j-2^{l-1}k, j+2^{l-1}k]$ when $\mathcal{C}$ contains at most k faults. Hence, we have $i \in [j-2^{l-1}k, j+2^{l-1}k]$, which contradicts relation 98. This proves relation 97.

By relation 97, if on any input permutation $r(i,d-lk)$ receives an item j' when $\mathcal{C}$ contains k or fewer faults, then j' is eventually moved from $r(i,d-lk)$ to a register segment $r(j, d-(l-1)k)$ for some $j \in [i-2^{l-1}k, i+2^{l-1}k]$. On the other hand, by the induction hypothesis, $r(j, d-(l-1)k)$ can only receive items in $[j-2^{l-1}k, j+2^{l-1}k]$ when $\mathcal{C}$ contains at most k faults. Thus, $j' \in [j-2^{l-1}k, j+2^{l-1}k] \subseteq [i-2^l k, i+2^l k]$. This completes the induction step for proving Claim D.1.

By Claim D.1 with $l = \left\lceil \frac{d}{k} \right\rceil$, we find that the input register segment $r(1,0)$ can only contain numbers in $[1-2^l k, 1+2^l k]$. On the other hand, the item input to $r(1,0)$ can be any number in $[1,n]$. Hence, we have $1+2^l k \geq n$, which yields $l = \left\lceil \frac{d}{k} \right\rceil \geq \log \frac{n-1}{k}$. This completes the proof of Theorem 6.3.