CMU-CS-84-104

A Robust Sorting Network

Larry Rudoiph
Department of Computer Science
Carnegie-Mellon University
August 1983

Abstract

Beginning with the recently introduced 'balanced sorting network' that sorts an input vector of size $N=2^n$ and consists of n *identical* blocks where each block is composed of n phases of $N/2$ comparators per phase, we propose a shuffle-exchange type layout consisting of a single block with the output recirculated back as input until sorting is achieved. The main advantage of the proposed design is that no comparator in the network is *critical* in the sense that any faulty comparator can be bypassed without disturbing the functionality of the network (just its speed). The novelty of the design is that the robustness is derived from the underlying algorithm. The network will sort in the presence of many faulty comparators. Moreover, of the $N \log N/2$ comparators, only N pairs of comparators are critical. That is, the network fails only when both comparators in any of these pairs fail. These results enable one to build large sorting networks on a single wafer so that a high percentage of the fabricated wafers can be used; some of the wafers will sort very quickly (the ones with no faulty components), most will sort at somewhat slower than optimal speeds, but only a few will fail to be useful as sorting networks (due to too many, badly placed faults).

The advent of VLSI technology is impacting almost all aspects of society. Unfortunately, one of the major problems facing VLSI technology increasing manufacturing costs is the low yield in mass production of chips and wafers. That is, although it is cheap to produce large quantities of a single chip or wafer, only a small fraction of them function correctly. The rest are rendered useless due to random flaws introduced during the fabrication process. The flaws tend to have the most adverse effect on the active elements (e.g., gates, transistors) and less effect on the wires.

One way of increasing the yield is by employing designs that function despite fabrication flaws. We present a layout for a sorting network that can withstand many faulty components. Although a small network can usually fit on a single chip, a much larger network could be made to fit on a single wafer. Previously known sorting networks shared the property that almost all the components (comparators) are crucial, and so large networks produced on a single wafer would be expensive due to the resulting low yield. This is not the case with our layout; most faulty comparators can be bypassed and still allow the network to sort, albeit somewhat slower.

Related work can be classified into two different areas, sorting networks and fault-tolerant systems. There has been much research in sorting networks and the related area of parallel sorting algorithms. Batcher [Batcher 68] introduced the Bitonic network as well as the Odd-Even network (see also [Knuth 68]) both requiring $O([\log N]^2)$ steps to sort input vectors of size N (see also, Hong and Sedgewick [Hong and Sedgewick 82] and Perl [Perl 83] for additional insights into such networks). There has also been much research in sorting algorithms for parallel processors, some parts of which are relevant to sorting networks, for example, Valiant [Valiant 75], Borodin and Hopcroft [Borodin and Hopcroft 82], and Kruskal [Kruskal 83]. Rief and Valiant [Reif and Valiant 83] give an algorithm that requires $O(\log N)$ expected time to sort on a particular type of network, whereas, Ajtai et. al [Ajtai el al 83] recently showed that there are $O(N \log N)$ sized networks that can sort in $O(\log N)$ steps, although the large constants make their network impractical. Winslow and Chow [Winslow and Chow 83] review and compare various sorting machines that make different assumptions about how the input and the output are connected to the machine.

The structure of the paper is as follows: We first describe the 'balanced sorting network', which has recently been introduced [Dowd et al 83a, Dowd et al 83b]. The 'crucial comparators' are then identified and their effect analyzed. The third section first reviews the layout proposed in the introductory paper and then modifies the layout in two ways: first to reduce the number of critical comparators and then to eliminate all of them. An analysis of the increased yield is also presented.

1. The Balanced Sorting Network

In this section we review the design and layout of the "balanced sorting network" introduced by Dowd et. al [Dowd et al 83a]. The network requires $[\log N]^2$ stages of $N/2$ comparators to sort N items and consists of a sequence of $\log N$ identical merging blocks, where each block possesses a highly regular, recursive design (see Figure 1-1 for a merging network of size 16). A novel aspect of the network is that the blocks are *identical* -- not smaller recursive versions as in those of Batcher [Batcher 68].

Specifically, a basic unit of the network is a two input, two output comparator transforming the arbitrary order of the two input elements into nondecreasing order. Each *phase* of the balanced merging network is composed of $N/2$ of these comparators with the first phase comparing elements $x(0)$ with $x(N-1)$, $x(1)$ with $x(N-2), \cdots, x(N/2-1)$ with $x(N/2)$, where x is the input vector. Taking the approach of an 'oblivious' algorithm in that even though the first phase does not guarantee a

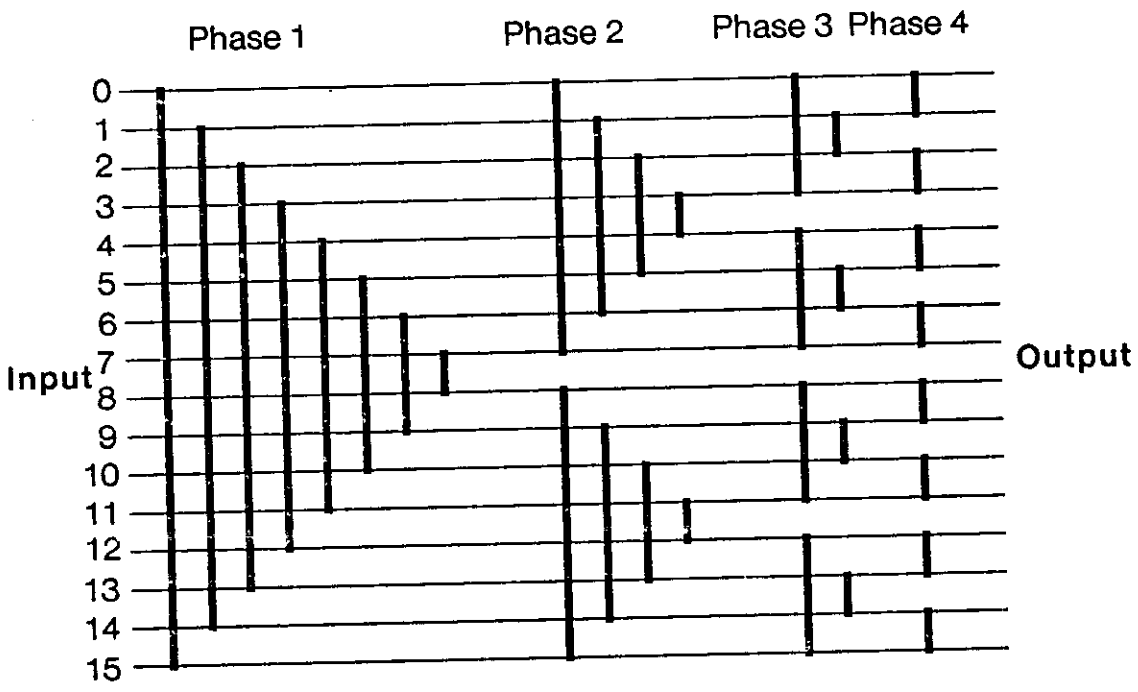


Figure 1-1: A Balanced Merging Block of Size 16

partition of the input into two halves, we pretend that it does and so continue to apply the same procedure to both halves of the output of the first phase recursively. Thus, $\log N$ phases comprise a merging network. We number the phases from 1 to $\log N$.

Figure 1-1 depicts a balanced merging network for $N = 16$ elements using Knuth's [Knuth 68] comparator-network representation where horizontal lines represent the input lines $x(i), 0 \le i < N$, and vertical lines represent comparisons between the elements on the corresponding input lines. Since the output of a merging network (from now on we call such a merging network a block) may not be sorted, we continue to apply these blocks until sorting is obtained. (Figure 1-2 shows the full balanced sorting network for size 8.)

Each block, as its name implies, is a merging network, however, it is not easily observed exactly what is being merged. The first phase of the merging network applied to a *recursively balanced* vector partitions the elements so that the $N/2$ smallest elements are in the first half of the vector and the $N/2$ largest elements are in the second half of the vector. Moreover, each half is recursively balanced so that each subsequent phase acts recursively to sort the input.

The balanced sorting network is very similar to the bitonic and the odd-even sorting networks introduced by Batcher [Batcher 68]. These networks also consist of $[\log N]^2$ stages with a stage composed of $N/2$ comparators. Moreover, they are both build upon merging networks, however, despite their similarity, there is no permutation between the input lines of either of Batcher's two networks and the balanced sorting network. The differences between the balanced network and those of Batcher are evident from the following lemma which is satisfied by neither the odd-even nor bitonic merge networks.

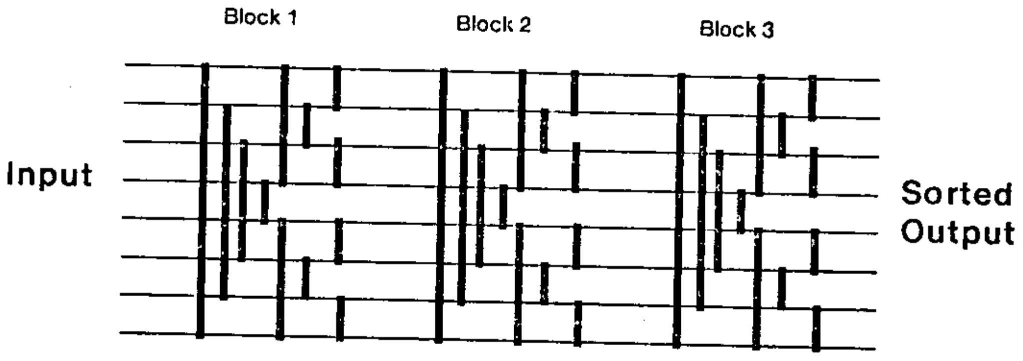


Figure 1-2: A Balanced Sort Network of Size 8

Lemma 1:

i) If no exchange occurs during a block, then the input of the block is sorted.

ii) A network which sorts any input can be constructed by serially composing a finite number of blocks.

Proof:

i) A single block performs all comparisons $x(i-1)$ with $x(i)$ for $1 \leq i < n$ (among others) and thus if no exchange occurs the input must be sorted.

ii) Each exchange decreases the number of inversions (that is, pair of elements which are out of order). Since a permutation has at most $\binom{n}{2}$ inversions, part (i) implies that $\binom{n}{2}$ blocks suffice to sort. □

In addition to differentiating between the sorting networks, this lemma is important in two respects. It demonstrates that only some of the comparisons are needed to sort. It also suggest a procedure for deciding when the output is sorted. The following implementation strategy, which is assumed throughout the rest of the paper, arises from these observations.

Since a succession of identical blocks are required, only one block is actually needed. The output of the block is recirculated back as input (see Figure 1-3). Moreover, by the first part of the lemma, the decision to recirculate can be based on whether any exchanges occurred within a block. Not only does this allow a faster completion time for certain input vectors and the elimination of a $\log N$ counter, but it also enables a more fault tolerant network, as will be shown later.

2. Critical Comparators

In this section we identify the 'critical comparators' of the recirculating network (see Figure 1-3). Since many fabricated chips, or more significantly, wafers, are likely to contain faulty comparators, it is desirable that the fabricated product still sort once the faulty comparators are bypassed. Recall that we are considering a recirculating network consisting of one block of comparators with the output recirculated back whenever there is at least one exchange occurring in the block. We compare a complete balanced sorting network with one missing some of its comparators. The term

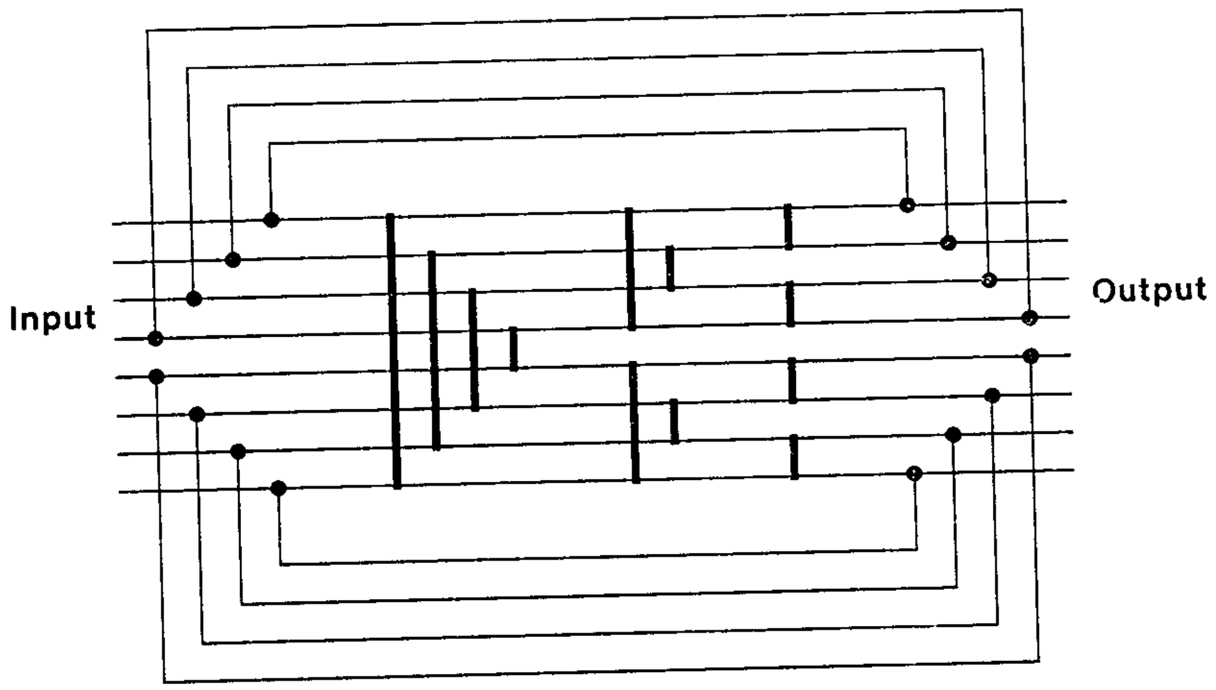


Figure 1-3: A Single Block Balanced Sort Network of Size 8

iteration is used to indicate the movement of data through one block of the complete network and the term *pass* refers to the movement of data through one block of the incomplete network. When some comparators are missing, there will usually be more passes than iterations to produce the sorted output for a given input.

Consider a size 8 network with an input vector x consisting of all zeros except for a one in the fourth position, i.e. $x(3) = 1$ or $x = [0,0,0,1,0,0,0,0]$ (the result after the sort should be $[0,0,0,0,0,0,0,1]$). It is easy to see that the $x(3){:}x(4)$ comparison (a first phase comparison) is crucial. Without it, the 1 will never change its position. On the other hand, the $x[4]{:}x[7]$ comparator (a second phase comparator) is not crucial. In the first block, the first phase exchanges $x[3]$ with $x[4]$, the next phase does nothing, the third phase exchanges $x[4]$ with $x[5]$. A second pass through the block will produce the sorted output using the $x[5]{:}x[6]$ comparator in the second phase and the $x[6]{:}x[7]$ comparator in the third phase.

Before presenting the main theorem, we introduce some notation and quote a few lemma's proved in [Dowd et al 83a].

- Greek letters represent a string of bits with a superscript to indicate the number of bits. For example $\alpha^{k-j}\beta^{j-1}1$ indicates a string of k bits, the first (high order) $k-j$ bits of which are denoted by α^{k-j}, and the last bit is 1. We omit superscripts of 1.

- Comparators are specified by the indices of the lines they compare; the two indices are separated by a colon (:). For example, the first phase of a size 16 network contains the comparator $0 : 15$. The indices will often be written in a binary templet form in which some of the bits are left unspecified in order to represent a set of comparators.

We first identify the critical comparators with the following definition and later show that these are indeed the only comparators that are required for sorting.

Definition 2: In a balanced sorting network of size $N=2^n$, the *critical comparators* of the $j+1^{st}$ phase are of the form

$$\alpha^j 0 1^{n-(j-1)} : \alpha^j 1 0^{n-(j-1)}.$$

The other $(N \log N/2)-N$ comparators are referred to as *noncritical.*

Proposition 3: In a size $N=2^n$ balanced sorting network, the j^{th} phase compares all pairs of entries whose indices have identical high-order (leftmost) $j-1$ bits and complementary low-order (rightmost) $n-(j-1)$ bits. In the above notation, comparisons in the j^{th} phase are between elements whose indices are of the form

$$\alpha^{j-1}\beta^{n-(j-1)} : \alpha^{j-1}\bar{\beta}^{n-(j-1)}$$

Definition 4: The level *i chains* are all those with the same rightmost i bits. The level i *cochains* are all those with the same or complementary rightmost $i+1$ bits.

Lemma 5: Applying the i^{th} phase $(j<i\leq n)$ to an input whose level $n-(j-1)$ cochains are sorted, preserves this property.

Lemma 6: Applying the i^{th} phase $(j+1<i\leq n)$ to an input whose level $n-j$ chains are sorted preserves this property.

The next theorem will make use of the following lemma. The proof of this lemma explicitly identifies a set of three comparators for each noncritical comparator whose combined effect is the same as that of the noncritical comparator.

Lemma 7: Given an input vector x of size $N=2^n$ in which the level $n-(j-1)$ chains are sorted, let $k<j$, and remove the following noncritical comparator from the k^{th} phase:

$$\alpha^{k-1}0\beta^{n-k} : \alpha^{k-1}1\bar{\beta}^{n-k}.$$

After a pass through the deficient block it will be the case that the missing comparison will be compensated, i.e. its effect realized. More precisely, after a pass it is the case that:

$$x(\alpha^{k-1}0\beta^{n-k}) \leq x(\alpha^{k-1}1\bar{\beta}^{n-k}).$$

Proof: Three other comparators, one phase k and two phase r, $(k<r\leq n)$, will be shown to accomplish the effect of the missing comparator (see Figure 2-1). Since the removed comparator is noncritical, we can write β^{n-k} as $1^a 0\gamma^{n-(k+a+1)}$, for some $a\geq 0$. This is because stage j critical comparators can be characterized as having their rightmost $n-(j-1)$ bits consist of either all 0's or all 1's (Definition 2). The three other comparators are:

1. (Phase k) $\alpha^{k-1}01^a1\bar{\gamma}^{n-(k+a+1)} : \alpha^{k-1}10^a0\gamma^{n-(k+a+1)}$
2. (Phase r) $\alpha^{k-1}01^a0\gamma^{n-(k+a+1)} : \alpha^{k-1}01^a1\bar{\gamma}^{n-(k+a+1)}$
3. (Phase r) $\alpha^{k-1}10^a0\gamma^{n-(k+a+1)} : \alpha^{k-1}10^a1\bar{\gamma}^{n-(k+a+1)}$]

For shorthand we write α, μ, and ν for α^{k-1}, 01^a, and $0\gamma^{n-(k+a+1)}$, respectively. Using this notation, we replace the comparator $\alpha\mu\nu : \alpha\bar{\mu}\bar{\nu}$ with the following:

- (phase k) $\alpha\mu\bar{\nu} : \alpha\bar{\mu}\nu$
- (phase r) $\alpha\mu\nu : \alpha\mu\bar{\nu}$ and $\alpha\bar{\mu}\nu : \alpha\bar{\mu}\bar{\nu}$

In order to differentiate the value in the vector at each phase, we superscript the vector with a phase number i for the value before the i^{th} phase comparison.

As a result of the phase k comparison we have $x^{k+1}(\alpha\mu\bar{\nu}) < x^{k+1}(\alpha\bar{\mu}\nu)$. By a previous lemma, this is still true at phase r. This, along with the fact that after a comparison, the smaller value is placed into the position with the smaller index and the larger value into the position with the larger index, we have:

$$\begin{aligned} x^{r+1}(\alpha\mu\nu) &= \min\,\{x^r(\alpha\mu\nu), x^r(\alpha\mu\bar{\nu})\} \\ &\leq x^r(\alpha\mu\bar{\nu}) \\ &\leq x^r(\alpha\bar{\mu}\nu) \\ &\leq \max\,\{x^r(\alpha\bar{\mu}\nu), x^r(\alpha\bar{\mu}\bar{\nu})\} \\ &= x^{r+1}(\alpha\bar{\mu}\bar{\nu}) \end{aligned}$$

□

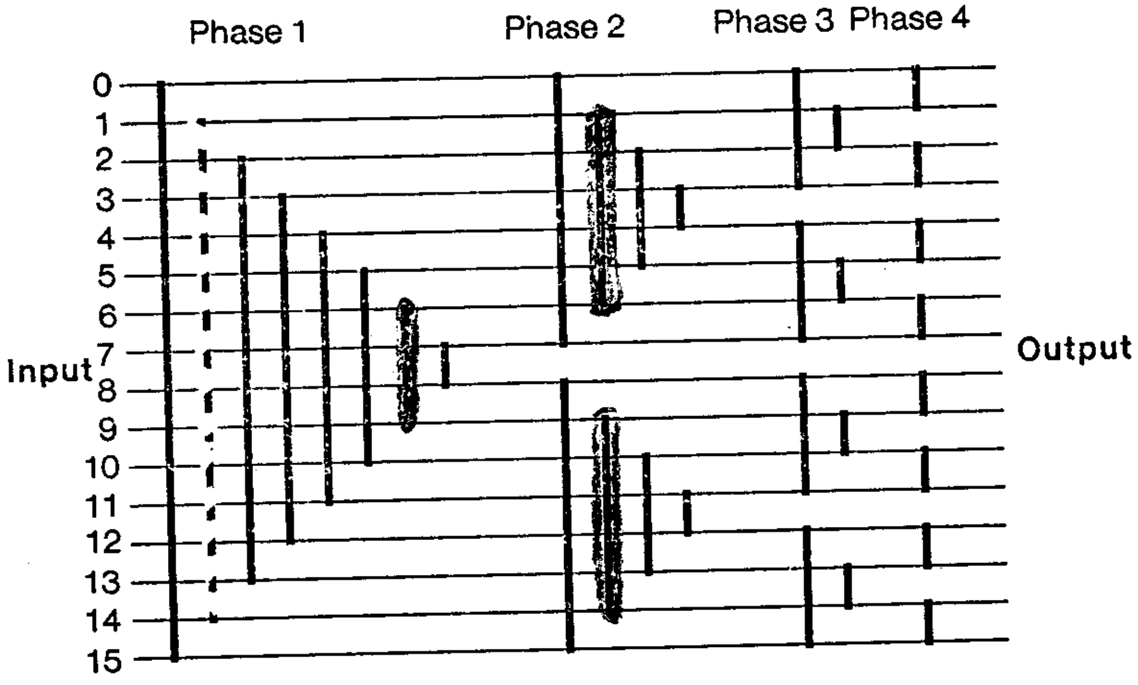


Figure 2-1: The boldface comparators compensate for the missing dotted one

The next theorem shows that the critical comparators are really the only critical ones.

Theorem 8: Given a size $N = 2^n$ balanced sorting network consisting of one block with the output of the block recirculated back as input whenever there is at least one exchange, then the critical comparators are the only ones needed to eventually produce a sorted output.

Proof: We need to show that the network sorts with noncritical comparators removed and will not sort if a critical comparator is removed. The proof consists of two parts. In the first part we first consider the removal of one noncritical comparator.

(i) Assume a noncritical comparator is removed from the k^{th} phase and consider the j^{th} iteration.

Case $k \geq j$: By lemma 6, the phases after $j+1$ have no effect.

Case $k \leq j$: Starting with level $n-(j-1)$ sorted chains, we must show that after another iteration it must be the case that the level $n-j$ chains are sorted. The first $k-1$ phases are the same in both the networks. By the previous lemma (Lemma 7) the effect of phase k is accomplished by the end of the pass. By lemma 5 it is clear that the first k phases of the subsequent pass does no harm. Therefore by the $k+1$ phase of the second pass, the items have the desired property required at the $k+1$ phase of the corresponding iteration.

At most twice the number of passes are needed to compensate for the removal of one noncritical comparator. The next question to ask is what happens if two noncritical comparators are bypassed? Let NC_1 and NC_2 be two noncritical comparators and let $COMP_1$, $COMP_2$, $COMP_3$ be the three comparators used to compensate for NC_1. If NC_2 is not one of these three then no extra passes are required. On the other hand , additional passes may be required if NC_2 is one of these three. Suppose NC_2 is $COMP_2$. Then no extra passes are required since by the end of the block the effect of $COMP_2$ will have been accomplished and thus the same for the effect of NC_1. However, if NC_2 is $COMP_1$ then the effect of $COMP_1$ may not occur until the end of the block and so an additional pass will be needed for $COMP_2$ and $COMP_3$ to have and effect. Thus in the worst case, three times as many passes will be needed if two noncritical comparators are removed.

(ii) It is clear by inspection that the critical comparators are indeed critical. Consider a phase $j+1$ critical comparator. It has the form $\alpha^j 0 1^{n-(j-1)} : \alpha^j 1 0^{n-(j-1)}$. In later phases, say phase $r \geq j$, the smaller indexed line will be compared to lines smaller than it (i.e. $\delta^{r-1} 0^{n-(r-1)} : \alpha^j 0 1^{n-(j-1)}$). Thus if the maximum key is on line $\alpha^j 0 1^{n-(j-1)}$, it will never be swapped by any other comparison.

One may wonder why Lemma 7 does not apply in this case. Upon careful examination, it is clear that for a critical comparator β^{n-k} cannot be rewritten in the required form. □

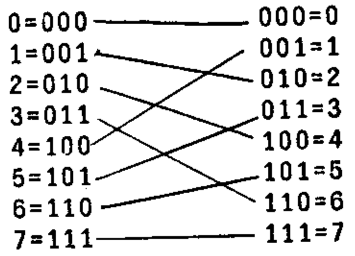
In general, when c noncritical comparators are removed, a factor of at most c increase in the number of passes will be required. Consider what happens if all the noncritical comparators from the first phase are removed, leaving only one phase 1 comparator. It is not hard to see that a factor of $\log N$ additional passes will be needed. Moreover, when all noncritical comparators are removed, the network is reduced to bubble sort [Knuth].

Corollary 9: With only critical comparators, sorting takes $N \log N$ phases.

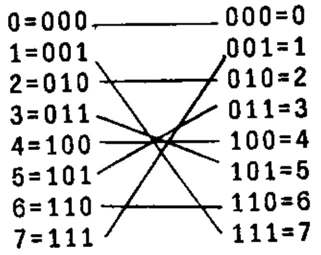
3. The Shuffle-Exchange Layout

Up to this point we described the network in terms of comparisons between the values on "horizontal lines". In this section, we first review the shuffle-exchange layout for the balanced sorting network as presented in [Dowd et al 83a], and then identify the critical comparators in this layout. A slight modification to the layout halves the number of critical comparators. Simple replication can then eliminate the rest of the critical comparators. The section concludes with an analysis of the increased robustness.

A shuffle exchange layout for $N = 2^k$ 'elements' consists of a series of identical stages. Each stage consists of $N/2$ two by two comparators, numbered 0 to $N/2-1$. If we number the lines through the comparators in a stage from 0 to $N-1$ so that the lines through the i^{th} comparator are labeled 2i and 2i + 1 then output i from stage t is connected to input σ(i) in stage t + 1 where the permutation σ is the *perfect shuffle* permutation (see [Clos 53, Benes 65]): if $i_{k-1}\, i_{k-2} \cdots i_0$ is the binary expansion for i then $\sigma(i) = i_{k-2}\, i_{k-3} \cdots i_0\, i_{k-1}$ (see Figure 3-1(a)).



(a) SHUFFLE PERMUTATION σ



(b) THE PERMUTATION τ

Figure 3-1:

Each comparator comparing input lines i and j, $i<j$ can be set into four possible states:

```
1.State + :  output(i) ≤ output(j)        (larger value to the upper line)
2.State - :  output(i) ≥ output(j)        (larger value to the lower line)
3.State 0 :  output(i)  =  input(i),
             output(j)  =  input(j)       (no exchange)
4.State 1 :  output(i)  =  input(j),
             output(j)  =  input(i)       (exchange).
```

The layout realizing the balanced merging network (a single block of the balanced sorting network) consists of $\log N$ shuffle exchange stages with all comparators set to the " + " state. Each stage corresponds to a phase of the merging network. In order that the layout simulate the merging network, the inputs into the layout must be a certain permutation τ of the inputs into the network, where $\tau(2i) = 2i$ and $\tau(2i+1) = n-2i-1$, that is, τ fixes the location of the even inputs and reverses the order of the odd inputs (see Figure 3-1(b)). The balanced sorting network can be realized with $\log N$ successive shuffle-exchange blocks with the output of each block connected to the input of the next block via the τ permutation (see Figure 3-2).

Our plan is to first show the correspondence between the network and the proposed layout, which requires some additional notation, so that the critical comparators in the layout can be identified.

Definition 10: Let $\underline{\mathrm{Line}^t(i)}$ be the value on the i^{th} network line (see Figure 1-1) just before the phase t comparisons ($0 \le i \le n$, $1 \le t \le \log N$). In particular, $\mathrm{Line}^1(i)$ are the input values.

Definition 11: Let $\underline{\mathrm{In}^t(i)}$ be the value of the i^{th} input line of the t^{th} stage of the layout. This is the value for the i(mod 2)th input into the $\lfloor i/2 \rfloor$ comparator ($0 \le i \le n$, $1 \le t \le \log N$). Note that during the t^{th} stage comparator i compares $\mathrm{In}^t(2i)$ and $\mathrm{In}^t(2i+1)$.

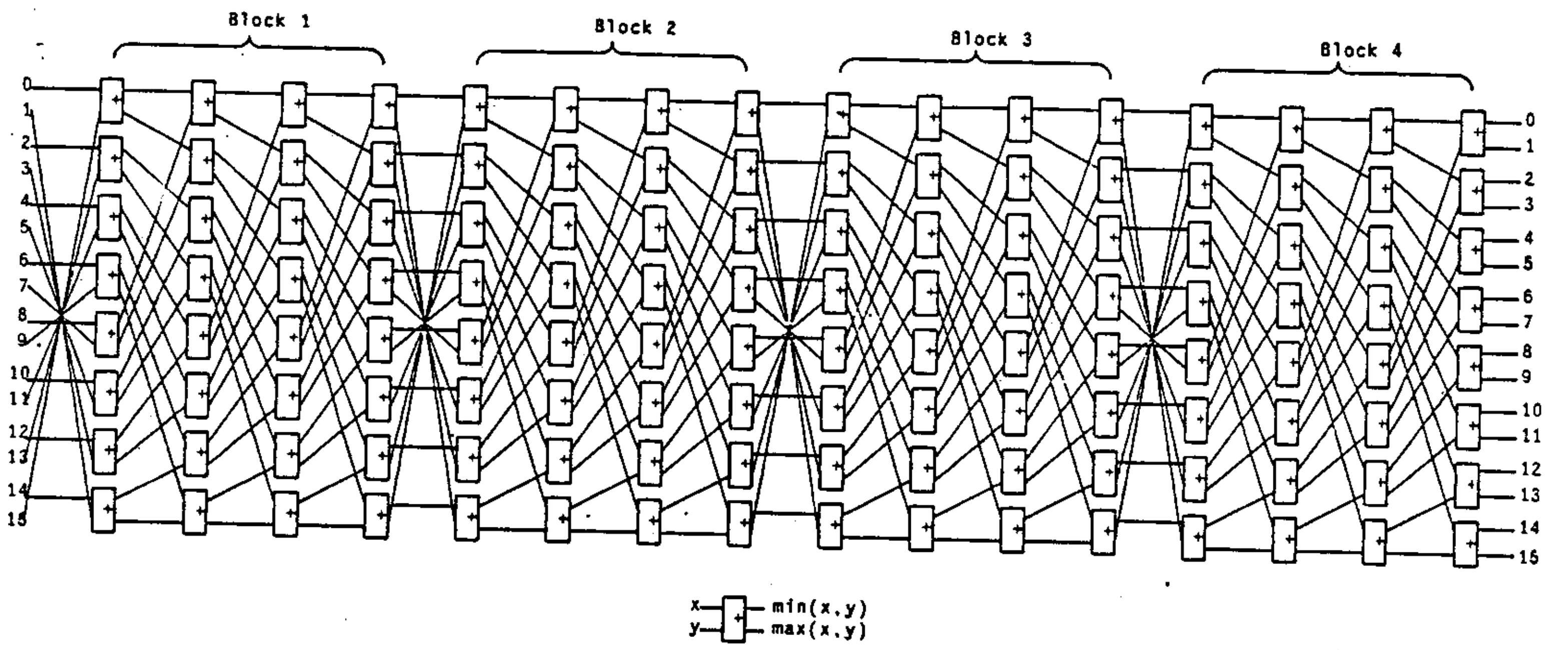


Figure 3-2: Shuffle-Exchange Layout for Sort Network of Size 16

The initial relationship between the inputs of the layout and the network is given by:

$$\text{In}^1(2i) = \text{Line}^1(2i) \quad \text{and} \quad \text{In}^1(2i+1) = \text{Line}^1(n-2i-1)$$

Thus the first stage of comparators of the layout performs the same comparisons as the first phase of the sorting network. Figure 3-3 tracks the movement through the layout over time with the numbers in the figure corresponding to the location of the corresponding input line in the balanced merging network. The motivation for the τ permutation can easily be seen from first stage/phase.

We want to show that the comparisons performed during the j^{th} phase of the network are the same as those performed during the j^{th} stage of the layout. Using the above notation, we quote the following lemma from [Dowd et al 83a].

Lemma 12: For a size $N=2^n$ balanced merging network (i.e. a block) and its corresponding shuffle layout, we have the following correspondences between the lines of the network and the inputs to the comparators in the layout:

$$\text{In}^1(\alpha^{n-1}0) = \text{Line}^1(\alpha^{n-1}0)$$
$$\text{In}^1(\alpha^{n-1}1) = \text{Line}^1(\bar{\alpha}^{n-1}1)$$

and for $j>1$,

$$\text{In}^j(\alpha^{n-j}\beta^{j-2}0\,0) = \text{Line}^j\,(\beta^{j-2}0\,\alpha^{n-j}0) \qquad \text{In}^j(\alpha^{n-j}\beta^{j-2}0\,1) = \text{Line}^j\,(\beta^{j-2}0\,\bar{\alpha}^{n-j}1)$$
$$\text{In}^j(\alpha^{n-j}\beta^{j-2}1\,0) = \text{Line}^j\,(\beta^{j-2}1\,\bar{\alpha}^{n-j}1) \qquad \text{In}^j(\alpha^{n-j}\beta^{j-2}1\,1) = \text{Line}^j\,(\beta^{j-2}1\,\alpha^{n-j}0).$$

A natural question to ask is which comparators in the layout correspond to the critical comparators of the network. It is clear that in the last stage all the comparators are critical. The following theorem supplies the general answer.

Theorem 13: In the t^{th} stage, the critical comparators are those of the form:

$(t=1)$	$10^{n-2}0 : 01^{n-2}1$
$(1<t< \log N)$	$10^{n-t-1}\beta^{t-1}0 : 10^{n-t-1}\beta^{t-1}1$
$(t= \log N)$	$\beta^{n-1}0 : \beta^{n-1}1$

Proof: We show that the 'In()' values of the above indices correspond to the 'Line()' values whose indices are those of the networks critical comparators. Lemma 12 and Definition 2 are used in the following equalities.

<u>Case</u> $(t=1)$

$$\text{In}^1(1\,0^{n-2}\,0) = \text{Line}^1(1\,0^{n-2}\,0)$$
$$\text{In}^1(1\,0^{n-2}\,1) = \text{Line}^1(0\,1^{n-2}\,1)$$

<u>Case</u> $(1<t\leq \log N)$ There are two cases arising from the low order bit of β^{t-1}.

<u>Subcase</u> $(\beta^{t-1}=\beta^{t-2}0)$

$$\text{In}^t(1\,0^{n-t-1}\,\beta^{t-2}\,0\,0) = \text{Line}^t(\beta^{t-2}\,0\,1\,0^{n-t-1}\,0)$$
$$\text{In}^t(1\,0^{n-t-1}\,\beta^{t-2}\,0\,1) = \text{Line}^t(\beta^{t-2}\,0\,0\,1^{n-t-1}\,1)$$

<u>Subcase</u> $(\beta^{t-1}=\beta^{t-2}1)$

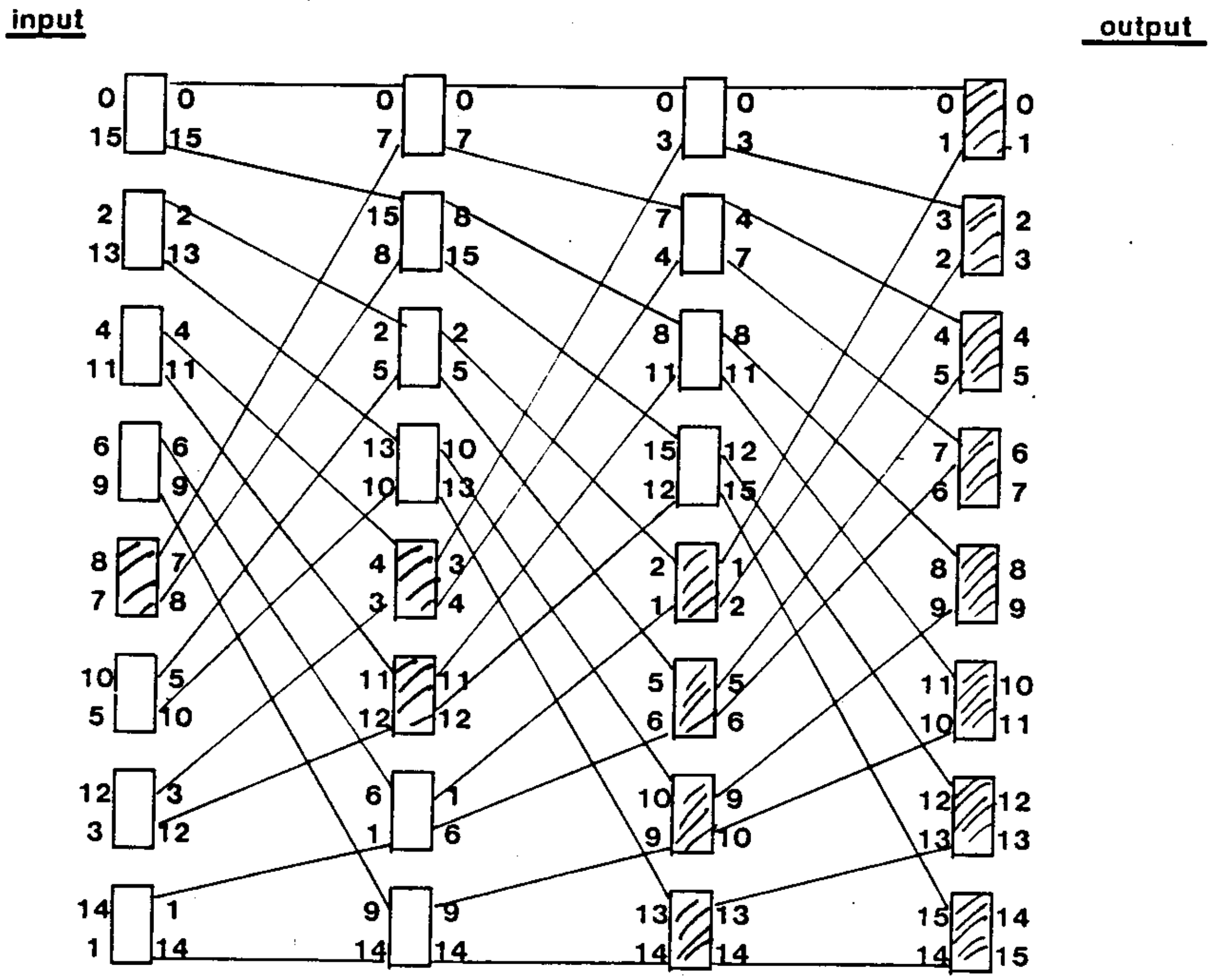


Figure 3-3: Tracks movement through the layout. Numbers correspond to horizontal lines of Figure 1-2. Critical Comparators are shaded.

$$\mathrm{In}^{l}(1\,0^{n-l-1}\,\beta^{l-2}\,1\,0) = \mathrm{Line}^{l}(\beta^{l-2}\,1\,0\,1^{n-l-1}\,1)$$
$$\mathrm{In}^{l}(1\,0^{n-l-1}\,\beta^{l-2}\,1\,1) = \mathrm{Line}^{l}(\beta^{l-2}\,1\,1\,0^{n-l-1}\,0)$$

□

In Figure 3-3, the critical comparators are shown. The input and output ports of each comparator are tagged with the corresponding Line() indices of the network. The advantage of this layout, as opposed to the shuffle-exchange layout for bitonic sort (see [Stone 71]), is that all the comparators always place the larger value on the bottom port (port 1). Note that for some of the comparators the top input port (port 0) correspond to Line values whose indices are larger than those for the bottom input port. Although this does not affect sorting, it does interfere with the application of the results from the previous section. The matter can easily be remedied by simply switching the input ports for these comparators.

4. A Single Block Layout

Having described the shuffle-exchange layout and identified the critical comparators, it is now possible to describe additional implementation details. There are a few choices for laying out the network: (i) include all[1] $(1/2)(\log N)^2$ stages with stages connected by either σ or τ, (ii) Use only one stage of $N/2$ comparators connected by σ, or (iii) a compromise scheme consisting of one block of $\log N$ stages with the output recirculated back as input. We will investigate the third option after some discussion of all three.

Since area arguments immediately eliminate option (i), we contrast the later two options. The full unfolding of a block (instead of one stage of shuffle-exchange) enables pipelining and thus very long keys can be sorted. If the high order bits match, then they can both be forwarded; as soon as a difference is detected, the appropriate sorting can be done for the rest of the bits. Thus, a key can be passing through many comparators at the same time. This is not possible in the single-staged network (option (ii)). Moreover, pipelining reduces the connection or pin requirements per comparator. The single-staged design requires the entire key to be stored in the comparator and to keep the same rate of throughput, wider channels are needed.

Perhaps even more important is the simplicity of the logic required for the single block layout. Besides all the obvious benefits of a simpler logic, it also means that each comparator requires less area (and fewer transistors). A single stage shuffle-exchange layout requires $O((N/\log N)^2)$ area (see [Kleitman et al 81]) for the connections (i.e. wires) and since there are $N/2$ comparators the total area required is:

Total Area of Single-Staged Layout is $\frac{WN^2}{(\log N)^2} + CN/2$.

Whereas, for a single block, $O(N^2)$ wire area is needed[2] (see [Wise 81, Snir 81] and Figure 4-1) and, although there are more comparators ($N \log N/2$), they are simpler and hence require less area:

---

[1] [Dowd et al 83a] shows that only the first 2 phases are needed in the first two blocks, the first 3 phases in the third block, . . . , the first k phases in the k^{th} block. Thus only $(1/2)(\log N)^2$ phases are actually needed.

[2] An appropriate placement of the comparators yields a block in which the first stage is connected to the second via an inverse shuffle permutation. There are N crossovers and thus requires $O(N)$ width. The top half of the second stage is connected by a size $N/2$ inverse shuffle to the top half of the third stage and similarly for the bottom half. There are now half as many crossovers, thereby requiring $O(N)/2$ width. Etc.

Total Area of One Block Layout is $W' N^2 + C' N \log N/2$.

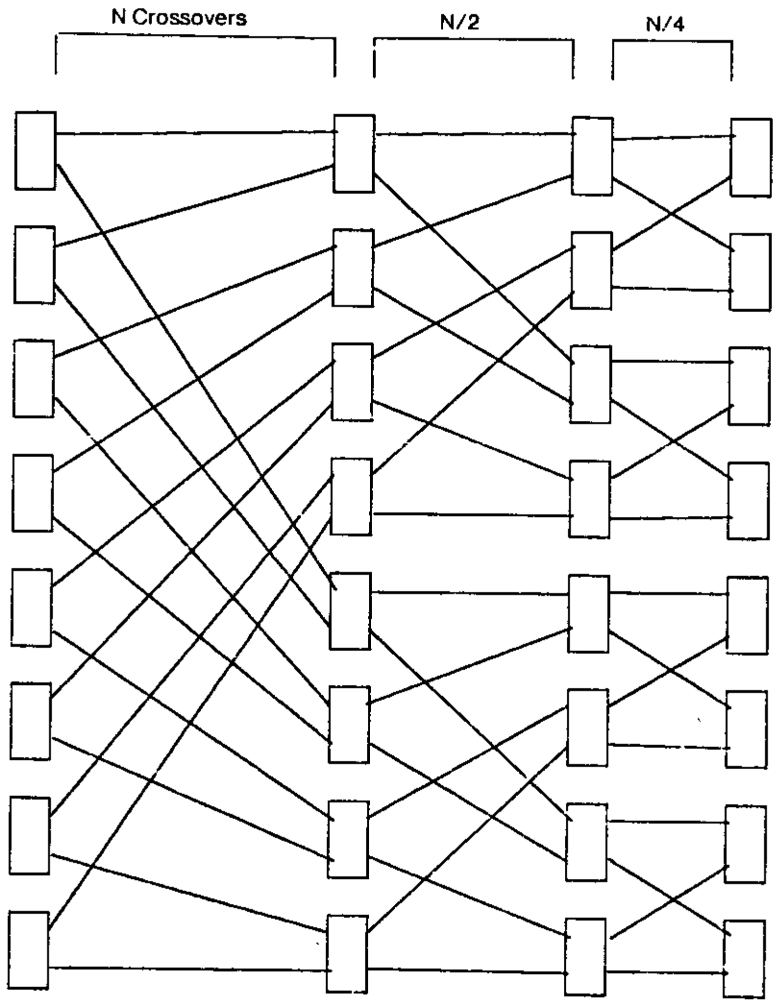


Figure 4-1: One block of layout using $O(N^2)$ wire area

Thus for specific application, the layout with minimal area depends on the relations between W, W', C, C'.

Lemma 1 is used as a basis for deciding whether or not the output is sorted. The lemma refers to the comparisons in the sorting networks: if there are no swaps during a block, then the output is sorted. Due to the mapping, this condition is slightly different in the layout -- the output is sorted if some of the comparators swap and some of them do not. It is easy to slightly modify the layout to eliminate this unpleasantness (see Figure 4-2).

At first blush, an AND gate for $N \log N/2$ boolean values appears to be required, however, the decision to recirculate can be generated dynamically with a much smaller gate. A bit is associated with each key indicating whether or not, based on the history of the key, a recirculation is needed. As two keys pass through a comparator, this bit is updated appropriately. After the last stage of the

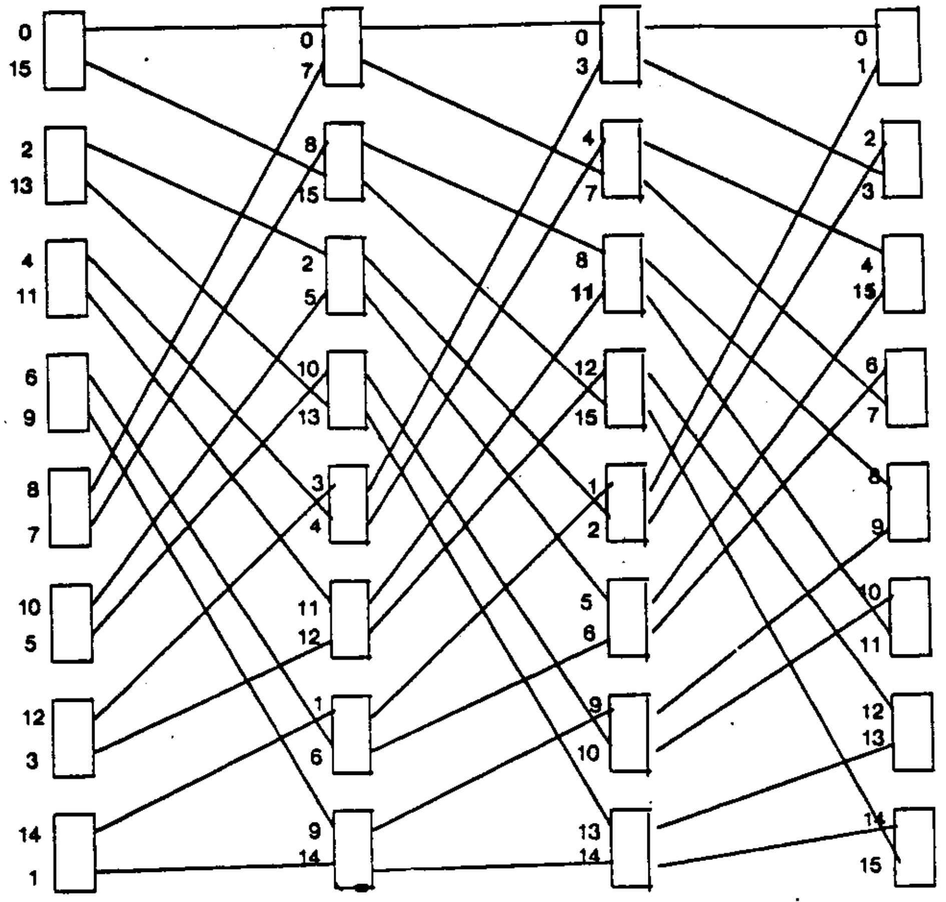


Figure 4-2: Layout slightly modified so that output is sorted iff no comparator swaps

block, the AND of these $N/2$ bits solves the recirculation decision.

4.1. Modifications

The layout just described can be modified to achieve greater robustness in the following way. The comparators are grouped into pairs in such a way that during alternate passes the comparators in each pair alternate roles. As a result, the pair becomes critical, requiring both comparators to fail in order to declare the pair as faulty; the failure of any single comparator does not interfere with the functionality.

Our modification affects all but the last stage which must be dealt with in a separate fashion. Figure 3-3 shows the critical comparators in one block of the shuffle layout. Note that it is our choice of τ that causes the critical comparators to be in the bottom half of the layout. Indeed, choosing a slightly different τ places all the critical comparators on the top half of the layout. Let τ' be such a permutation defined as:

$$\tau'(2i) = N-2i-1 \text{ and } \tau'(2i+1) = 2i.$$

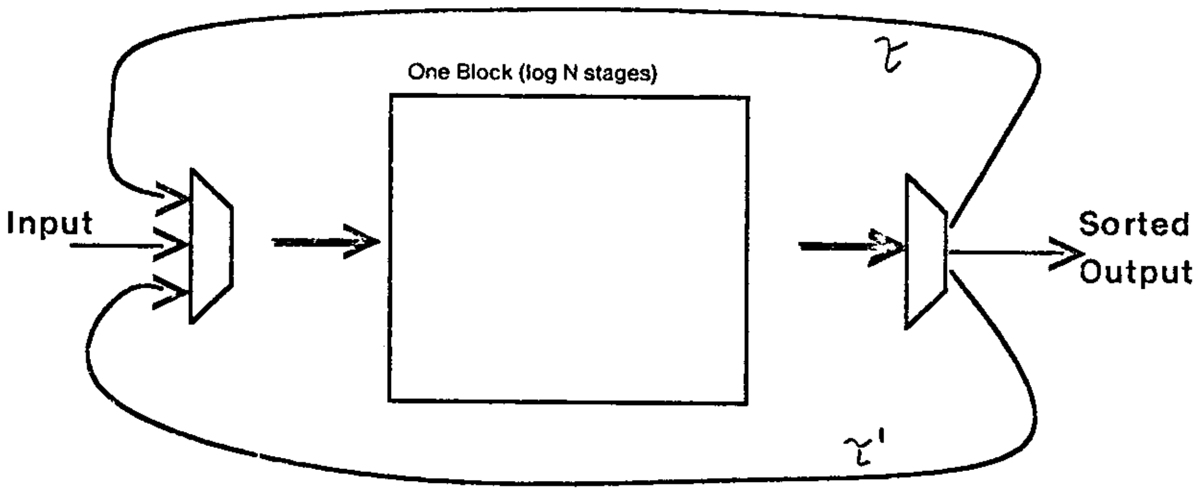


Figure 4-3: A Single block with the output circulated back by τ or τ'. If there are no exchanges within the block, the output is sorted.

Consider a one-block recirculating layout in which the recirculation alternates between τ and τ' (see Figure 4-3). During even passes the critical comparators will be in the lower half and during odd passes they will be in the upper half. Thus, any particular critical comparator in any but the last stage is no longer critical; any single comparison in the network will be performed by two different comparators in the layout, thereby allowing any single comparator to be bypassed.

In order to extend this property to the entire layout, we propose that the last stage be replicated thereby making all comparators noncritical. Since there are $N \log N/2$ comparators in a block, the addition of $N/2$ will not be too costly. With this addition, we can now state:

Proposition 14: At the very worst, the loss of a single critical comparator will double the sorting time, although generally one would expect a much smaller performance loss. Moreover, of the $(N(\log N-1))/2$ comparators, there are just N critical pairs. The layout will fail to sort only when both comparators in a pair fail.

4.2. Analysis

A layout containing fewer critical comparators is more robust than one with more critical comparators. We show the probability that an entire network functions correctly assuming a uniform distribution of faulty comparators. A layout is said to function correctly if its faulty comparators can be bypassed in such a way that it will eventually sort any given input vector.

We compare four layouts, all of which consist of a single block of $\log N$ stages, with each stage containing N/2 comparators:

1. All the comparators are crucial. This is the case of a single block layout of either of the two Batcher networks.

2. N crucial comparators. The original proposed single block layout of the balanced sort network.

3. $N/2$ crucial comparators and $N/4$ crucial pairs of comparators. This is the situation once τ and τ' are incorporated.

4. No crucial comparators and N crucial pairs of comparators. This can be easily accomplished by replicating each of the $N/2$ comparators of the last stage.

Let p be the probability that a comparator is faulty. We assume that the faults are uniformly distributed and independent. The following are the expected probabilities:

- Probability that each comparator works is $(1-p)$

- Probability that a pair works is $(1-p^2)$

- Probability that the last phase works is $(1-p)^{N/2}$

Layout (i)	Layout (ii)	Layout(iii)	Layout (iv)
$(1-p)^{N \log N/2}$	$(1-p)^N$	$(1-p^2)^{N/4}+(1-p)^{N/2}$	$(1-p^2)^N$

Table 4-1: Probability whole network works

The curves in Figure 4-4 show the probabilities of the network of size N functioning for various values of p.

5. Testing for Faulty Comparators

An additional nice feature about the balanced sorting network is that it is very easy to test for faulty comparators within a single block. It is easy to devise an input vector that will test any single comparator by direct examination of the network (i.e. Figure 1-1).

A single input vector checks for faulty critical comparators. Such a vector could simply be a list of sorted numbers. With another input one can test that all comparators perform a swap. Let $i = i_{n-1}, i_{n-2}, \cdots, i_0$. Then the input vector that causes all comparators to swap during the first pass is

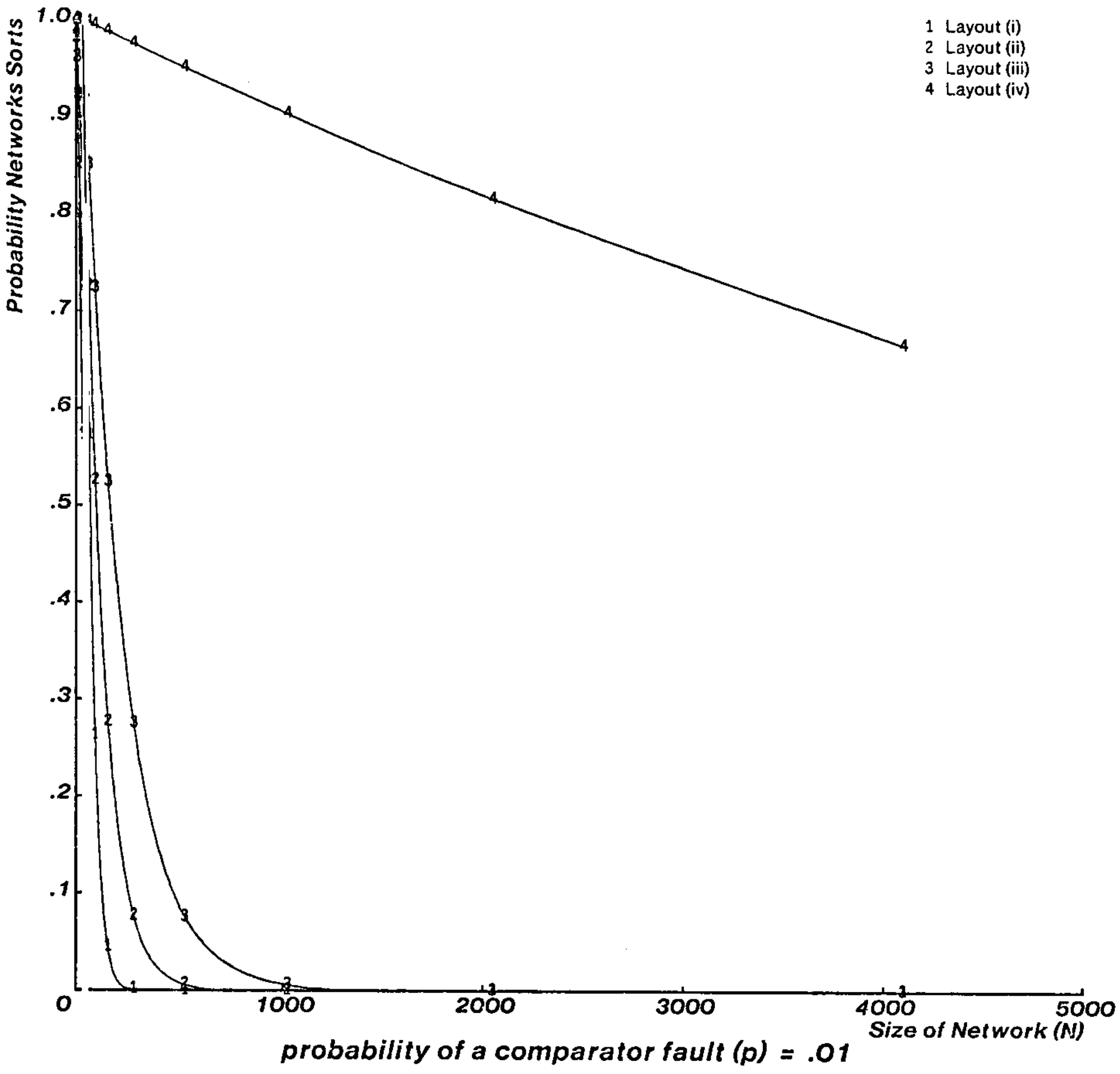


Figure 4-4: Probability of Network Working

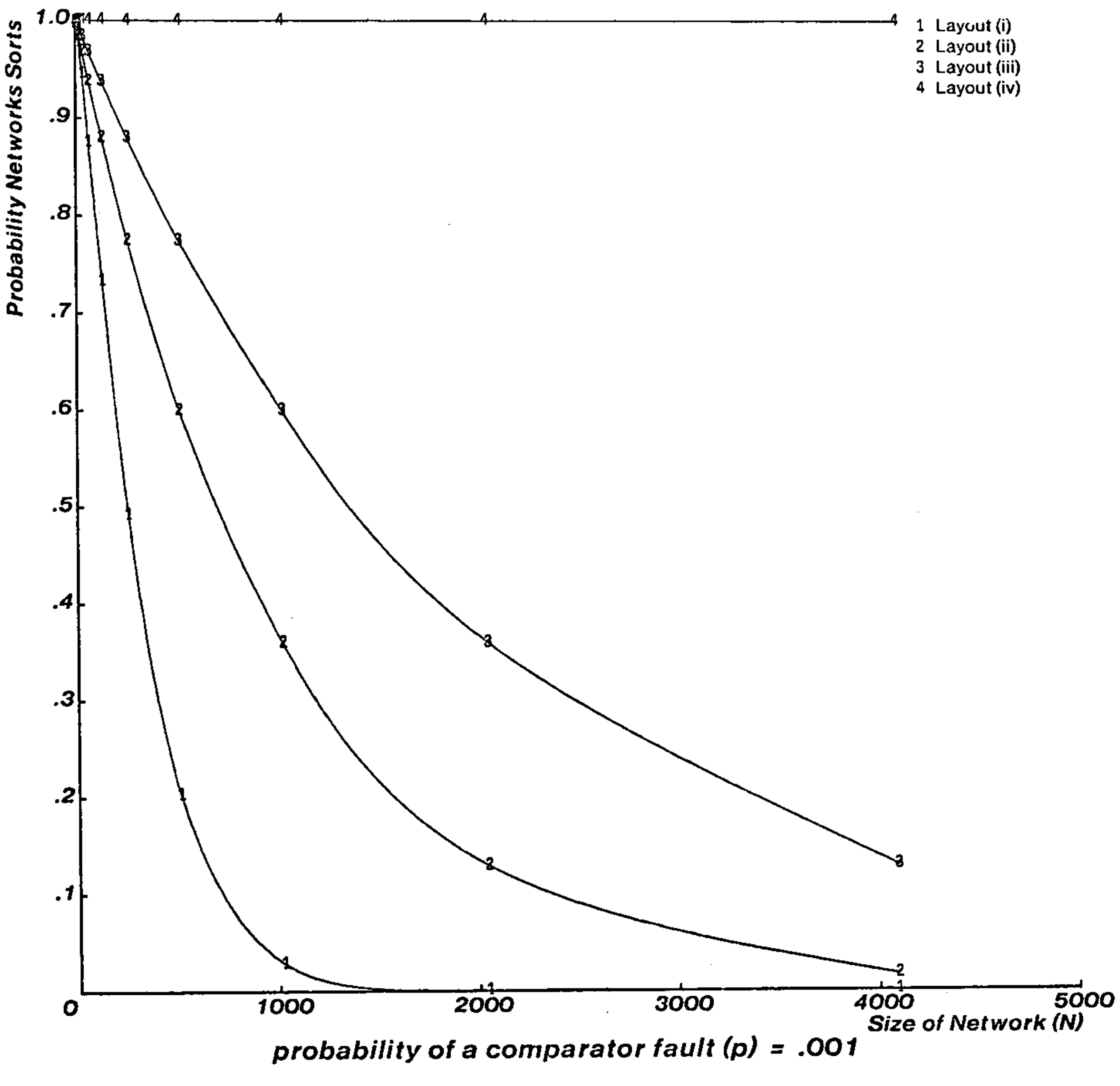


Figure 4-5: Probability of Network Working

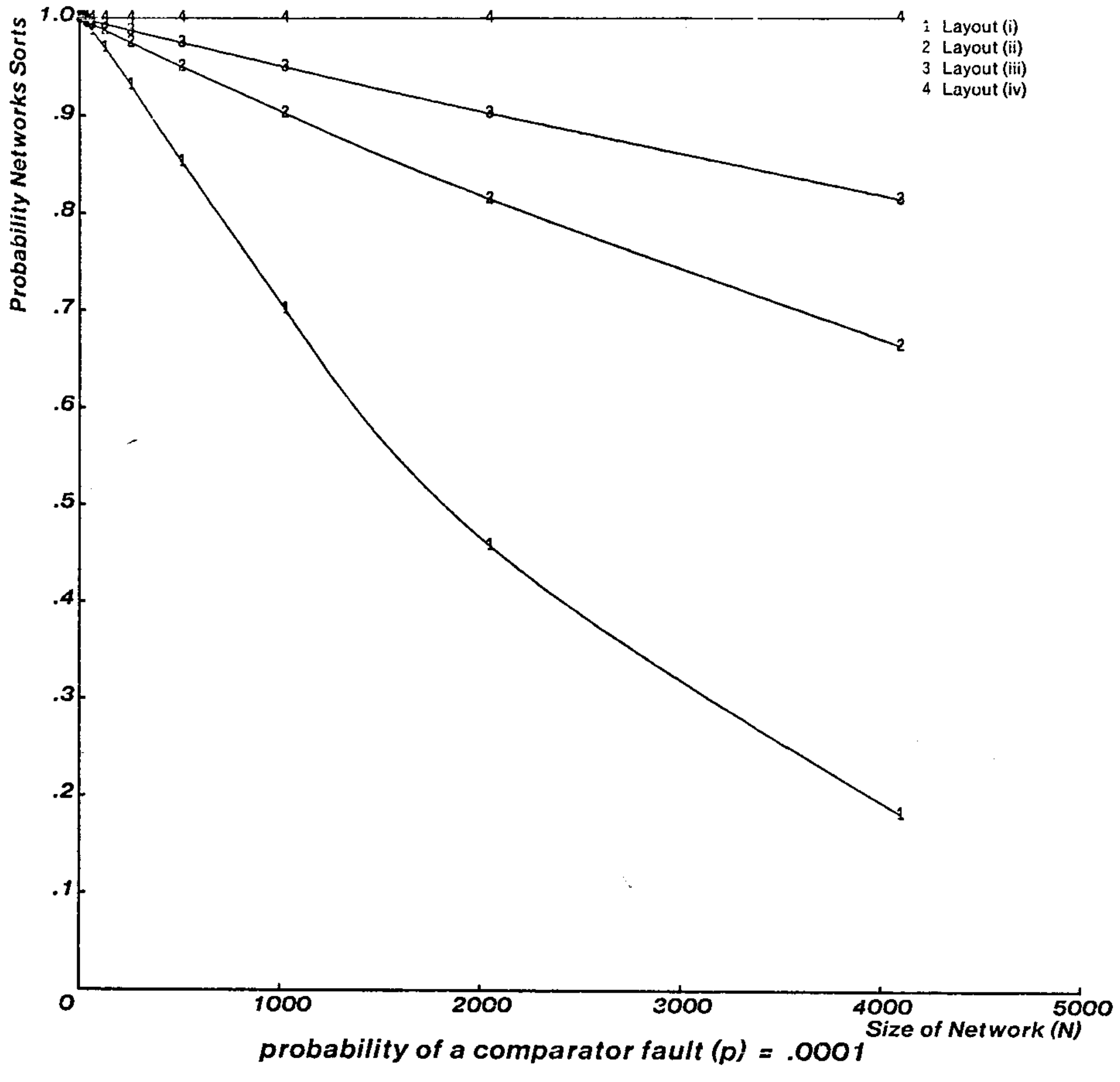


Figure 4-6: Probability of Network Working

defined as follows:

$$x(\mathrm{i}) = \bar{\mathrm{i}}_{n-1}, \mathrm{i}_{n-2}, \bar{\mathrm{i}}_{n-3}, \mathrm{i}_{n-4}, \cdots$$

In other words, every other bit is complemented. Let y be the output vector after one pass though the block. If $y(\mathrm{i}) = \mathrm{i}_{n-1}, \cdots, \mathrm{i}_{k+1}, \bar{\mathrm{i}}_k, \cdots$, then it is easy to identify the phase k comparator that failed to swap.

When actually constructing a single block recirculating sorting network as outlined above, one need not initially permute the input by τ or τ' since the initial input is unordered[3]. If this is the case then the test input vector just presented must first be permuted by τ.

Given an input that forces every comparator to execute a swap, it is then possible to design a self modifying circuit. A single input bit indicates that the circuit is in debug mode. While in this mode each comparator tests to see that it executes a swap. If a comparator does not swap then the comparator can automatically disable itself (i.e. put itself into bypass mode).

6. Conclusion

Although the balanced sorting network has the same time and space requirements as that of the bitonic sorting network, we have shown that its advantages are more than cosmetic. The network can be realized as a highly robust shuffle-exchange layout. It is thus possible to produce a fairly large sorting network on a single wafer so that most of the wafers fabricated can be used.

The major 'trick' used in our design was the alternation of roles played by each comparator. It would be interesting to find other algorithms that can exploit this trick. As the requirements for large-scale parallel processing grows, so does the need to develop 'robust' algorithms that can function in the presence of many failures. Another requirement appears to be a simply computed function that indicates termination as well as the requirement for progress to occur at each iteration.

An immediate application of the results of this paper may be routing networks. Our sorting network can be considered to be a routing network with each comparator sending an input value to the output port corresponding to a certain bit in the destination address (instead of as the result of a comparison). Although a single shuffle-exchange block can route an single input to any single output, some permutations of N inputs require more than one pass. Perhaps our method can be used to build a robust routing network?

[3]This may not be the case if the input is assumed to be 'almost' sorted.

References

[Ajtai el al 83] Ajtai, M., J. Komlos, and E. Szemeredi.
An C(n log n) soritng Network.
In *15th Annual ACM Symposium on Theory of Computing*, pages 1-9. 1983.

[Batcher 68] Batcher, K. E.
Sorting networks and their applications.
AFIPS Spring Joint Computer Conference 32:307-314, 1968.

[Benes 65] Benes, V. E.
Mathematical theory of connecting networks and telephone traffic.
Academic Press, 1965.

[Borodin and Hopcroft 82]
Borodin, A., and J. E. Hopcroft.
Routing, Merging and Sorting on Parallel Models of Computation.
In *14th Annual ACM Symposium on Theory of Computing*, pages 338-344. 1982.

[Clos 53] Clos, C.
A Study of Nonblocking Switching Networks.
Bell System Technical Journal 32:406-424, 1953.

[Dowd et al 83a] Dowd, M., Y. Perl, L. Rudolph, and M. Saks.
The Balanced Sort Network.
In *Proceedings of Principles of Distributed Computing*, pages 161-172. ACM, August, 1983.

[Dowd et al 83b] Dowd, M., Y. Perl, L. Rudolph, and M. Saks.
The Balanced Sort Network.
Technical Report DCS-TR-127, Rutgers University, New Brunswick, NJ, June, 1983.

[Hong and Sedgewick 82]
Hong, Z., R. Sedgewick.
Notes on Merging Networks.
In *14th Annual ACM Symposium on Theory of Computing*, pages 296-302. 1982.

[Kleitman et al 81] Kleitman, D., T. Leighton, M. Lepley and G. Miller.
A survey of new layouts for the shuffle-exchange graph.
In *13th Annual ACM Symposium on Theory of Computing*. 1981.

[Knuth 68] Knuth, D. E.
The Art of Computer Programming. Volume 3: *Searching and Sorting.*
Addison-Wesley, 1968.

[Kruskal 83] Kruskal, C. P.
Searching, Merging, and Sorting in Parallel Computation.
IEEE Transactions on Computers C-32(10), 1983.

[Perl 83] Perl, Y.
Bitonic and Odd-Even Networks are more than merging.
Technical Report, Rutgers University, New Brunswick, NJ, 1983.

[Reif and Valiant 83]
Reif, J. H. and L. G. Valiant.
A logarithmic time sort for linear size networks.
In *15th Annual ACM Symposium on Theory of Computing*, pages 10-16. 1983.

[Snir 81] Snir, M.
Lower Bounds on VLSI implementations of Communication Networks.
Technical Report 32, Courant Institute, NYU, 251 Mercer st, NY, New York, May, 1981.

[Stone 71] Stone, H. S.
Paralel Processing with the Perfect Shuffle.
IEEE Transactions on Computers C-20:153-161, 1971.

[Valiant 75] Valiant, L. G.
Parallelism in comparison problems.
SIAM Journal of Computing 4(3):348-355, 1975.

[Winslow and Chow 83]
Winslow, L. E., and Y.C. Chow.
The analysis and design of some new sorting machines.
IEEE Transactions on Computers C-32(7):677-683, 1983.

[Wise 81] Wise, D. S.
Compact layout of Banyan/FFT networks.
In Kung, Sproll, and Steele (editor), *Conference on VLSI Systems and Computations*, pages 186-195. Computer Science Press, 1981.