Design and Analysis of Defect Tolerant Hierarchical Sorting Networks

Sy-Yen Kuo and Sheng-Chiech Liang
Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, R.O.C.

Abstract

A novel hierarchical defect-tolerant sorting network is presented in this paper. The design achieves a balance in *area-time* cost between the odd-even transposition sort and the bitonic sort. It uses less hardware than a single-level odd-even transposition sorter and reduces the wire complexity of the bitonic sorter in VLSI or WSI (wafer scale integration) implementation. The optimal number of levels in the hierarchy is evaluated and the sorting capability of each level is derived to minimize the hardware overhead. The hierarchical sorting network is very regular in structure and hence is easy to provide redundancy at every level of the hierarchy. Hierarchical reconfiguration is implemented by replacing the defective cells with spare cells at the bottom level first, and goes to the next higher level to perform reconfiguration if there is not enough redundancy at the current level. Yield analysis is performed to demonstrate the effectiveness of our approach.

I. INTRODUCTION

According to Thompson's[1] analysis, only two of the thirteen sorters discussed in his paper were designed with high degree of concurrency and thus suitable for real-time applications. One is the bitonic sort[2] and the other is the odd-even transposition sort [3]. The Batcher's bitonic sort requires $(N/2)[\log_2 N \cdot (\log_2 N + 1)]/2$ sorting elements with the concurrency factor $[\log_2 N \cdot (\log_2 N + 1)]/2$ to sort N input elements. Although this architecture has the advantage of logarithmic *processor-time* cost, it is difficult to implement in VLSI if N is very large, due to its complex communication scheme. Even with some modified sorting networks which are much more regular but require more sorting elements ($(N/2)[\log_2 N \cdot \log_2 N]$) than the bitonic sort, such as the perfect shuffle sort[4] or the balanced sort [5], they are still hard to implement in VLSI due to their complex interconnections. It is well known that the minimum area required to lay out an m-line perfect shuffle interconnection networks grows as m^2. This problem is even more significant when the sorter is implemented in WSI (Wafer Scale Integration) which can have a huge number of sorting elements fabricated in a single wafer. In addition, due to the large area and the processing technology limitation, defects are unavoidable in WSI implementation.

Therefore, the sorting networks need to have defect tolerance capabilities or to be reconfigurable such as the fault-tolerant systolic sorting array in [6]. However, this is a hardware intensive architecture since it requires $O(N^2)$ sorting elements to sort N inputs. Hence, various architectures have been proposed to obtain good *processor-time* tradeoff and ease of implementation in VLSI and WSI. The k-way bitonic sorter in [7] reduces the wire complexity of a bitonic sorter with a multi-way modularized approach. Since each module in the k-way bitonic sorter is a bitonic sorter and the interconnections between modules are shuffle-connected it is very difficult and costly to introduce redundancies into the sorter [8,9]. However, the analysis in [10] shows that it is not cost effective in replacing a defective cell with a redundant cell and its yield performance is not good enough [11]. Therefore, a hierarchical modular sorting network (*HMSN*) was presented by the authors in [10,11]. The design of *HMSN* considers the balance between the simple communication scheme of the odd-even transposition sort and the fast convergent speed of the bitonic sort.

II. HIERARCHICAL MODULAR SORTING NETWORKS

If N is very large, it is difficult to implement the N-input bitonic sorter in a single chip VLSI or WSI[10] due to the complex and long interconnections. As shown in the middle of Fig. 1, both shuffle and butterfly interconnections are used in the bitonic sorter and the longest interconnection exists between sorting elements which are $n/2$ elements away from each other if there are n elements in each

0-8186-2482-5/92 $1.00 © 1992 IEEE

stage. Although the odd-even transposition sorter is a hardware intensive architecture, it has the advantage of having simpler and shorter interconnections. As shown in the left of Fig. 1, every sorting element only communicates with its two nearest neighbors and hence the odd-even sorter is more suitable for implementation in VLSI and WSI. Therefore, in order to have fewer processing elements as well as less wire complexity and faster convergence in sorting, the sorting network can be decomposed into a two-level structure with the bitonic sorter in the bottom level and the odd-even transposition sorter in the top level.

A. Hierarchical Modular Sorting Network

Although the two-level sorting network has a better *processor–time* cost measure than the odd-even transposition sorter, it is still difficult to incorporate redundancy and reconfigure for surviving from defects since the bottom-level bitonic sorter has irregular shuffle and butterfly interconnections. Therefore, it is not cost effective to use this architecture for WSI implementation where reconfiguration is necessary to tolerate defects. To minimize the cost to survive from defects, the easily reconfigurable odd-even transposition sorter can be used as the bottom-level sorter to replace a sorting element in each bitonic sorter [10].

Let $N=b\times m\times h$. Then, each bottom-level odd-even sorter can sort b inputs, each middle-level bitonic sorter can merge m sets of sorted inputs with b inputs per set, and the top-level odd-even sorter can merge h sets of inputs with $m\cdot b$ inputs in each set. In the following, a sorting element will be referred as a *cell* at the bottom-level, a *submodule* at the middle-level, and a *module* at the top-level. Each bottom-level odd-even sorter has b stages with $b/2$ *cells* in an odd stage and $b/2-1$ *cells* in an even stage if b is even [3]. If b is odd, there are $(b-1)/2$ cells in a stage. A data register "D" in the odd-even sorter is a buffer to synchronize the data movements. We refer a middle-level bitonic sorter in Fig. 1 as a *multi–way bitonic merger*. A cell (submodule) marked with a "1" ("-") means that the outputs from it are in monotonic decreasing order, otherwise, the outputs are in monotonic increasing order.

It can be shown by using the method similar to that in [3] for *merge–sort* that the multi-way bitonic merger in the middle-level with a total of $(\log_2 m+1)\cdot(\log_2 m+2)/2$ stages can completely sort $b\cdot m$ inputs if there are m (m needs to be a power of two) submodules in each stage and each module can

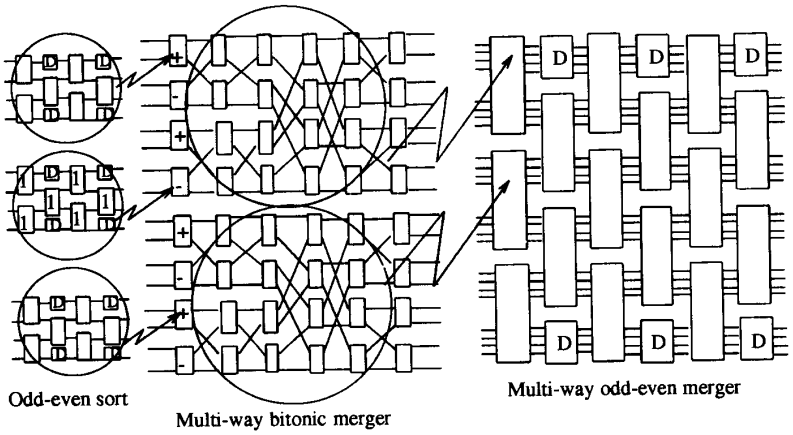

Fig. 1. Hierarchical sorter.

sort b inputs. In the top-level, the odd-even sorter is referred as a *multi-way odd-even merger* which can merge h sets of $b \cdot m$ sorted inputs into the correct order. The multi-way odd-even merger has $2 \times h - 1$ stages with h modules in each stage and can merge h sets of $b \cdot m$ sorted outputs into the correct order if each module can sort $b \cdot m$ inputs. An example three-level sorter is shown in Fig. 1 where the four-input odd-even transposition sorter is used in the bottom-level.

In addition to the bitonic sorter, a new sorting network, the balanced sorter, was proposed in [5] recently. This balanced sorter was proved essentially to be equivalent to the bitonic sorter in[12] so that it can also be a good candidate for the middle-level sorter. If we replace each bitonic sorter in Fig. 1 by a balanced sorter, a *multi-way balanced merger* is formed. Although the multi-way bitonic merger uses less sorting submodules and incurs less time latency ($\log_2 n \cdot (\log_2 n + 1)/2$ stages) to fill the pipeline than the balanced merger ($(\log_2 n)^2$ stages), the multi-way balanced merger has the advantage that it is a repetitive structure and has the same interconnection pattern between stages in a block. This repetitive architecture can simplify both the design and the operation complexity compared with the recursive architecture of the bitonic merger. In addition, unlike the multi-way bitonic merger which does not have uniform sorting submodules, the multi-way balanced merger contains only one type of submodules. The bitonic merger which has recursive structure and needs more than one type of submodules may increase the implementation complexity. Therefore, the multi-way balanced merger is more suitable for WSI implementation because of its uniform submodules and regularly repeated architecture.

B. Easily Reconfigurable Equivalent Networks

As discussed in subsection A, the multi-way bitonic merger and the multi-way balanced merger have advantages in WSI implementations. However, it is very difficult to reconfigure them to exclude faulty elements because their shuffle type interconnections. Some modifications are necessary to make these networks easily reconfigurable. Therefore, in addition to the topological equivalence between the modified data manipulator and the Omega network we have shown that they are also functionally equivalent [10]. Hence the shuffle connected *Omega* network in the balanced merger can be replaced by the modified data manipulator without any modification. Since the modified data manipulator has simpler interconnections, the resulting network is easier to reconfigure (will be discussed in section IV) around faulty submodules.

The shuffle connections in the multi-way balanced merger can now be simplified by replacing the *Omega* network in each sorting block with the modified data manipulator. The remaining shuffle permutations (σ) in the multi-way bitonic merger and τ permutations in the multi-way balanced merger can also be replaced by the equivalent switching network which are more regular and reconfigurable since we have shown in [10] that : (1) the shuffle permutation σ is topologically equivalent to the Banyan permutation r, (2) The τ permutation is topologically equivalent to ψ permutation.

Now we can see that the sorting networks with complex connections can be replaced by equivalent networks which are inherently easier to reconfigure. Each switching element in the equivalent networks is either in the bypass state or in the exchange state and therefore, is very simple to implement such that the area and time penalty is negligible compared with the entire sorting network. The number of cells used in the modified equivalent networks is equal to that of the original networks and the time latency is not changed.

III. OPTIMAL DECOMPOSITION

In this section, we will present the analysis procedures for determining an optimal sorting capability of each level based on the wire complexity and the *processor-time* cost. We will assume that the multi-way balanced mergers are used at the intermediate levels in the analysis, however, similar analysis can be performed if the multi-way bitonic mergers are used. The analysis will begin with a three-level *HMSN* and then the *HMSN*s with more than three levels.

For a three-level sorter, let $N = b \times m \times h$ and therefore, the total number of cells is

$$H_o = b(b-1)m(\log_2 m + 1)^2 h(2h-1)/2. \quad (1)$$

The ratio R of H_o over the number of sorting cells in a single-level odd-even sorter which is $N\cdot(N-1)/2$ is then

$$R=\frac{b(b-1)\cdot m(\log_2 m+1)^2\cdot h(2\cdot h-1)}{b\cdot m\cdot h(b\cdot m\cdot h-1)}\approx\frac{(b-1)\cdot(\log_2 m+1)^2\cdot(2\cdot h-1)}{b\cdot m\cdot h-1}<\frac{2(\log_2 m+1)^2}{m}. \quad (2)$$

From equation (2), we see that the *HMSN* will have less sorting cells than the single-level odd-even sorter if $m>64$ and $R\ll 1$ if $m\gg(\log_2 m+1)^2$. Therefore, m should be as large as possible in order to reduce the number of sorting cells in a three-level *HMSN*. However, m should be under the constraints imposed by the technology and wire complexity. Thus, the optimal m is technology dependent. In the following analysis we will assume that $m=m_{max}$ is known in advance and is dependent on the technology and the wire complexity.

After the value of m has been determined in a three-level *HMSN*, we can find the values for b and h. From equation (1), since both N and m are fixed so $\log_2 m$ as well as $b\times h$ (let it be represented as p) are fixed, the minimization of H_o is then equivalent to minimizing $(b-1)(2\cdot h-1)$ = $2b\cdot h-2\cdot h-b+1$. By maximizing $2h+b$ under the constraint that $b\cdot h$ is equal to a constant p, we will have a minimum H_o. This means that b, which is an integer factor of p, should be as small as possible.

However, with redundancies included in every level, finding the minimum with respect to b is not possible with analysis. Simulation is thus necessary to select the minimum H_o and determine b and h. The reason is that the minimization procedure involves finding the minimum value of a fourth order function of b and b not only has to be an integer but also must be a factor of p. Let H_r be the number of sorting cells in a *HMSN* with redundancy and assume that there are d, l, and n redundant rows as well as f, g, q redundant columns in each sorter at the bottom-level, middle-level, and top-level, respectively. Then,

$$H_r=(\frac{b-1}{2}+d)\cdot(b+f)\cdot(m+l)\cdot[(\log_2 m+1)^2+g]\cdot(h+n)\cdot(2\cdot h-1+q). \quad (3)$$

Finding the minimum H_r with respect to various b is equivalent to finding the minimum C where

$$C=(b-1+2d)\cdot(b+f)\cdot(h+n)\cdot(2h-1+q)=[b\cdot h+(2d-1)h+nb+n(2d-1)]\cdot[2b\cdot h+2f\cdot h+(q-1)b+f(q-1)] \quad (4)$$

Since m in equation (3) and the redundancies added to each sorter are fixed, only h $(=p/b)$ is related to b in finding the minimum H_r with respect to b.

Example cases with various amounts of redundancy are used to illustrate how to find the minimum C (or H_r) with respect to a given p. It should be noted that H_r is equal to $k\times C$ where k can be viewed as a constant and is equal to $(m+l)\cdot[(\log_2 m+1)^2+g]/2$. These example cases have one or two redundant rows and columns in each level. The amount of redundancy in a level for each case is shown in Table I where nr represents n redundant row and mc represents m redundant columns. Tables II presents the results for these example cases with $p=20$. The C_{min} is the minimum C and the corresponding b is listed as b_{min}.

In Fig. 2, we show the cost versus b graphically for cases 1, 2, 3 and 4 with $p=100$ and $p=105$. The cost decreases rapidly before $b=b_{min}$ for case 2 which has two redundant columns in the bottom-level sorter and increases rapidly after $b>b_{min}$ for case 3 which has two redundant columns in the top-level sorter. Since case 4 has two redundant columns in both the bottom-level and the top-level sorters, the curve in Fig. 2 shows the cost decreases before b_{min} and then increases after $b=b_{min}$. The cost does

Table I. The amount of redundancy for each case.

	Case No.							
level	1	2	3	4	5	6	7	8
bottom	1r	1r2c	1r	1r2c	2r	2r2c	2r	2r2c
middle	1r	1r	1r	1r	2r	2r	2r	2r
top	1r	1r	1r2c	1r2c	2r	2r	2r2c	2r2c

Table II. The cost C with p=20.

	Case No.							
p_1	1	2	3	4	5	6	7	8
4	1080	1620	1320	1980	1764	2646	2156	3234
5	1050	1470	1350	1890	1680	2352	2160	3024
10	990	1188	1650	1980	1560	1872	2600	3120
C_{min}	990	1188	1320	1890	1560	1872	2156	3024
b_{min}	10	10	4	5	10	10	4	5

not change significantly with respect to b for case 1 with no redundant columns.

Comparing case 2 with case 3 in Fig. 2, we see that the minimum cost of adding two redundant columns in the bottom-level sorter is less than that of adding two redundant columns in the top-level sorter. Case 4 has the highest hardware cost among the first four cases and it costs about 20-25% more than case 1 or 2. Case 8 has the highest cost among all cases since it has the most redundancy in every level. However, the optimal amount of redundancy depends not only on the area overhead but also on the yield improvement achieved over the original structure with no redundancy. Therefore, the results on these cases will be used in section V to determine the optimal amount of redundancy in each level.

Based on our analysis in [13] , normally a three-level network is sufficient for most applications unless a huge number of inputs (more than 128×256×256 inputs) is to be sorted. Practically, it may not be possible to implement a sorting network to sort more than 128×256×256 inputs in a single wafer. Hence, in the rest of this paper we will concentrate on the three-level structure only.

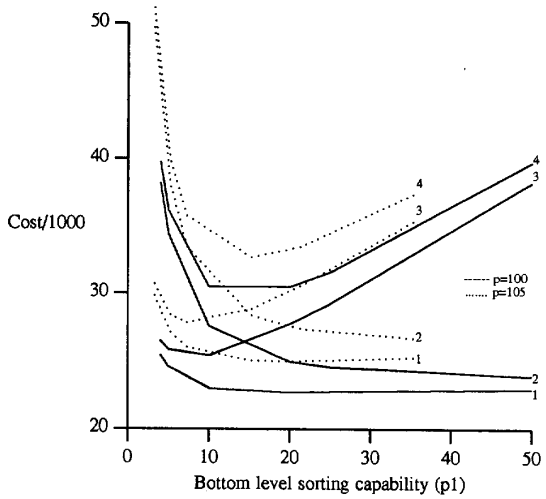

Fig. 2. Cost C vs. b for cases 1, 2, 3 and 4 with p = 100 and 105.

IV. DEFECT TOLERANCE

A. Bottom-Level and Top-Level Reconfigurable Structures

Since the odd-even sorter is used in both the bottom-level and the top-level of the sorting network, the reconfigurable structures in these two levels are the same and similar to that in [11]. It has been shown shown in [6] that by adding two extra stages in an odd-even sorter, any single fault which makes a sorting cell perform an incorrect swap (we call this a functional fault) can be recovered automatically. In addition to the fault masking property for the single functional fault, each sorting cell includes bypass registers such that the faulty cells generating nonfunctional errors can be bypassed without affecting system synchronization. Therefore, the odd-even transposition sorter can tolerate up to two faulty stages by simply bypassing the faulty cells without the need to restructuring the entire sorter.

For the top-level sorter, since the bottom-level sorter in each submodule can be bypassed, each module can also be bypassed by setting all sorters in the submodules of this module bypassed. Therefore, the same reconfigurable structure with redundant rows and columns of modules or cells can be applied to both levels.

B. Middle-Level Reconfigurable Structure

However, with clustered defects, it is possible that there are more than two faulty stages in a bottom-level sorter. If the number of faults in a bottom-level sorter is larger than the amount of redundancy it has or if any physical defect causes a faulty cell unable to be bypassed, this sorter will be declared as unrepairable and switched out by the reconfiguration scheme described in the following and replaced by a redundant submodule.

Input lines and output lines of a submodule in this level are connected to three submodules in the preceding stage and succeeding stage. Each submodule has two switches, one in the input port to select two out of three inputs and the other in the output port to direct data to two of the three output lines. It should be noted that after the transformation to the equivalent network, the shuffle interconnections in the bitonic merger are replaced by the Banyan permutation and therefore, the bitonic merger is now like a modified data manipulator. The *Omega* network in a sorting block of the balanced merger and the τ permutation between sorting blocks are replaced by the modified data manipulator. Therefore, the balanced merger is now connected by a series of modified data manipulators.

The major drawback of the defect tolerant multi-way merger is that the reconfigurable butterfly interconnections between two stages have wrap-around connections. An example butterfly interconnection between two stages is shown in Fig. 3(a), where each stage has k=8 sorting elements and one redundant element (R=1). If each element has n=3 outputs with one bit line per output, then there will be 8 ($=(k+r-1)n/3$) warp-around interconnections. Let a represent the wire width of an interconnection and z the space between two interconnections as shown in Fig. 3. Then the distance between two stages in a butterfly interconnections is $k/(2\sqrt{3})\times(h+z)$ (d1 = d2 in Fig. 3). The length on the longest interconnection due to wrap-around will be about $(k/2+k+1)\times(h+z)$ which is $3\sqrt{3}$ times longer than the shortest interconnection.

This drawback can be avoided by replacing the wrap-around interconnections with interconnections directly from the source to the destination. Let the submodules in a stage be numbered from 0 to k. As shown in Fig. 3(b), the three wires of a sorting element s ($0\le s\le k$) connect respectively to the sorting elements s, $k/2$, and $k/2+1$ in the next stage, if $s\le k/2$, otherwise, they connect to the sorting elements s, $s-k/2$ and $s-k/2-1$ in the next stage. The configuration of Fig. 3(b) is then equivalent to Fig. 3(a). The length of the longest interconnection in Fig. 3(b) becomes $2(k/2+1)(h+z)/\sqrt{3}$ which is approximately $2/(3\sqrt{3})$ that of the longest interconnection in the wrap-around structure.

V. YIELD ANALYSIS

The yield of a WSI array processor is defined as the probability that during the manufacturing process, defects are distributed into cells, switches, and wires of the array in such a way that all defective elements can be tolerated [14]. In order to evaluate the improvements on yield after

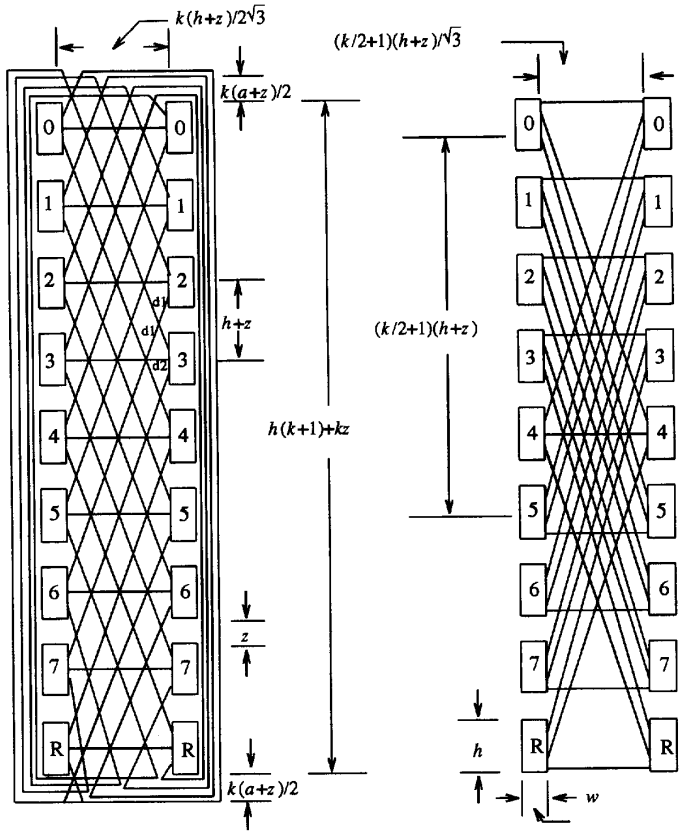

Fig. 3. Butterfly interconnections with and without wrap-around wires.

redundancies are introduced in each level, yield modeling and analysis are developed in this section. Our approach is to start the analysis at a stage of the bottom-level odd-even sorter.

In the real manufacturing environment, the defects have a tendency towards clustering. Therefore, the yield Y follows the more accurate negative binomial distribution, *i.e.*, $Y = (1 + DA/\alpha)^{-\alpha}$ [15]. α is a parameter representing the level of clustering, which usually takes a value around 1 or 2. The probability of having k defects in a stage is then[15]

$$P_w(k) = \frac{\Gamma(k+\alpha)(\frac{DA}{\alpha})^k}{k!\Gamma(\alpha)(1+\frac{DA}{\alpha})^{\alpha+k}}.$$

The yield Y_{bs} of a stage at the bottom-level with $n-r$ normal cells and r redundant cells can then be

derived as

$$Y_{bs} = \sum_{k=0}^{r} \frac{\Gamma(k+\alpha)(\frac{DA}{\alpha})^k}{k!\Gamma(\alpha)(1+\frac{DA}{\alpha})^{\alpha+k}} \times P_{knr}, \quad (8)$$

where P_{knr} is the reconfiguration coverage which represents the probability of successfully reconfiguring a stage of n cells with r redundant cells and k defects. For our reconfiguration scheme, $P_{knr}=1$ if $k \leq r$, and in order to simplify the analysis we will assume that $P_{knr}=0$ for $k>r$ to obtain the lower bound of Y_{bs}. Actually, our scheme can tolerate more than k defects if some of the defects fall on the same cell.

Defective wires or switches between two stages can also be tolerated by using alternative paths and bypassing the corresponding cell which can be viewed as a faulty cell and replaced by a redundant cell. Therefore, the effect of defects in wires and switching elements on the yield can be included into the model Y_{bs} by adding the area of switches and interconnections to the total area A in equation (8).

Therefore, the yield of a bottom-level sorter, Y_b, is equal to $Y_{bs}{}^m$, if there are m stages and no redundant stage in the sorter. If two redundant stages are included, the bottom-level sorter can tolerate up to two consecutive faulty stages, and the yield is

$$Y_b = Y_{bs}{}^{m+2}+(m+2)Y_{bs}{}^{m+1}(1-Y_{bs})+(m+1)Y_{bs}{}^m(1-Y_{bs})^2.$$

Since a submodule is an odd-even sorter, the yield of a stage in the middle-level sorter is

$$Y_{is} = Y_b{}^{i+2}+(i+2)Y_b{}^{i+1}(1-Y_b)+(i+2)(i+1)Y_b{}^i(1-Y_b)^2/2$$

assuming there are i submodules plus two redundant submodules in each stage.

Let A_{iw} represent the area of the interconnections and the switching elements between two neighbor stages of a middle-level sorter. The yield Y_{iw} of the interconnection area can be derived by using equation (8). The yield of a stage including the interconnection area is then $Y_{is}'=Y_{is}\times Y_{iw}$. Therefore, the yield of a middle-level sorter, Y_i, will be $(Y_{is}')^q$ if there are q stages.

Similarly, the yield, Y_{ts}, of a stage in the top-level sorter is

$$Y_{ts} = Y_i{}^{j+2}+(j+2)Y_i{}^{j+1}(1-Y_i)+(j+2)(j+1)Y_i{}^j(1-Y_i)^2/2$$

if there are j rows plus two redundant rows in each stage and $Y_t=Y_{ts}\times Y_{tw}$ where Y_{tw} is the yield of the interconnection area between two top-level stages. Then, Y, the yield of the top-level sorter (*i.e.*, the yield of the *HMSN*) is

$$Y = Y_t{}^{l+2}+(l+2)Y_t{}^{l+1}(1-Y_t)+(l+1)Y_t{}^l(1-Y_t)^2, \quad (9)$$

if there are l stages plus two redundant stages in the top-level odd-even sorter.

An example sorting cell in[6] is used in the following simulation to evaluate how much yield improvement can be achieved by various amount of redundancies in each level. The height of a cell is assumed to be between 5 μm (micrometers) and 50 μm. From equation (7), the area A_{iw} of the butterfly interconnections in the middle-level is proportional to $(kh)^2$ where $k=m_{max}$ is the sorting capability of the middle-level sorter. Therefore, the larger the A_{iw} is the smaller the Y_{iw} will be. Since $Y_{is}'=Y_{is}\times Y_{iw}$, any small decrease of the value of Y_{iw} will reduce the value of Y_{is}' and thus, drop the yield of the middle-level sorter significantly. This is due to the fact that $Y_i=(Y_{is}')^q$, where q is the number of stages in a sorter at this level and is no less than 64 (*i.e.*, $m_{max}=128$).

Yield with respect to p for the example cases in Table I are shown in Fig. 4(a) and (b). The defect density D in the cell area is assumed to be two defects per cm^2 and α is 2. However, since the wires and switches are much simpler and more regular than the cells, they are less vulnerable to defects and hence we assume that defect density in the interconnection area is one tenth of D. If no redundancy is included in an *HMSN*, the yield is zero for all cases in Table II. From Fig. 4(a), we see that the *HMSN* performs well in every case with $p=20$ and $m_{max}=256$ (the solid lines) for cell heights less than 25 μm. The dotted line in Fig. 4(a) shows the yield with $m_{max}=128$. We only show cases 1 and 2 which have about 92% yield even if the cell height = 50 μm. For cases 3 to 8, the yields are close to

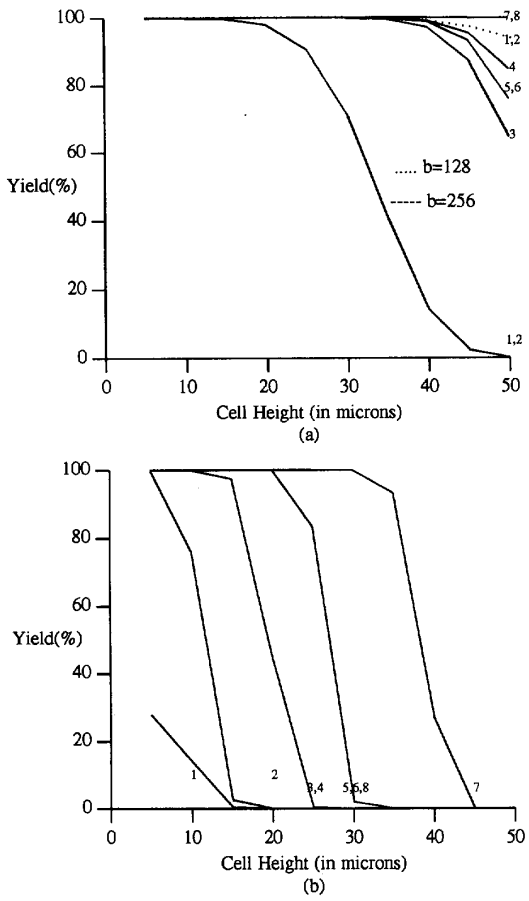

Fig. 4. Yield analysis.

100%.

In Fig. 4(b), m_{max}=256 and p=100, the yield drops very fast when only one redundant row is incorporated in every level for case 1. Also, we can see that case 7 performs better than case 8 even case 8 has two more redundant columns in the bottom-level sorter. The reason is that when two redundant rows are included in the bottom level, the yield of a bottom level sorter will be almost 1, and two more redundant columns cannot generate any further significant improvement on yield. Therefore, the difference on yield between case 7 and case 8 depends on the value b. Since A_{iw} grows with $(b \times cell\text{-}height)^2$, a difference in b can generate a large difference in Y_{iw}. From Table II, since b=10 for case 7 and b=20 for cases 6 and 8, case 7 has a higher yield than cases 6 and 8. The yield of case 5 with b=25 is less than that of case 6, but it is not significant. The same reason can be applied for the

differences among cases 2, 3, and 4.

VII. SUMMARY

A novel approach to designing a defect-tolerant hierarchical modular sorting network is presented in this paper. The defect-tolerant *HMSN* uses less hardware and converges faster than a single-level odd-even transposition sorter and the wire complexity problem of the bitonic sorter in VLSI or WSI is alleviated. A cost function is derived to determine the optimal sorting capability at each level and minimize the hardware complexity when redundancy is provided at every level of the hierarchy. Hierarchical reconfiguration strategy is used to tolerate the defective elements in an efficient manner. Detailed yield analysis is performed on the hierarchical sorting networks. Yield improvements for cases with various number of spares are evaluated. The simulation results show that the defect tolerant *HMSN* achieves a significant yield increase over a nonredundant sorting network.

REFERENCES

1. C. D. Thompson, "The VLSI complexity of sorting," *IEEE Trans. Comput.* **c-32** pp. 1171-1184 (Dec. 1983).
2. K. E. Batcher, "Sorting networks and their applications," *Proc. AFIPS Conf.* **32** pp. 307-314 (1968).
3. D. E. Knuth, *The art of computer programming - searching and sorting*, Addison-Wesley, Reading, MA (1973).
4. H. S. Stone, "Parallel processing with the perfect shuffle," *IEEE Trans. Comput.* **c-20** pp. 153-161 (Feb. 1971).
5. M. Dowd, Y. Perl, and M. Saks, "The balanced sorting network," *Proc. ACM Princ. Distrib. Comput.*, pp. 161-172 (1983).
6. S.-C. Liang and S.-Y. Kuo, "Concurrent error detection and correction in real-time systolic sorting arrays," *Proc. 20th Annu. Symp. on Fault-Tolerant Computing*, (1990).
7. T. Nakatani, S. -T. Huang, B. W. Arden, and S. K. Tripathi, "K-way bitonic sort," *IEEE Trans. Comput.* **c-38** pp. 283-288 (Feb. 1989).
8. S. Horiguchi, "Fault tolerance performance of WSI systolic sorter," *Int. Con. on Wafer Scale Integration*, pp. 196-202 (1990).
9. S. Horiguchi, I. Numata, and M. Kimura, "Self-reconfigurable algorithm of WSI sorting network," *Int. Con. on Wafer Scale Integration*, pp. 249-255 (1991).
10. S.-C. Liang and S.-Y. Kuo, "Defect tolerant sorting networks for WSI implementation," *Int. Con. on Wafer Scale Integration*, pp. 131-137 (1990).
11. S.-Y. Kuo and S.-C. Liang, "Design and analysis of defect tolerant hierarchical sorting networks," *submitted to IEEE Trans. on Computers*, (1991).
12. G. Bilardi, "Merging and sorting networks with the topology of the Omega network," *IEEE Trans. Comput.* **c-38** pp. 1396-1403 (Oct. 1989).
13. S.-Y. Kuo and S.-C. Liang, "Defect tolerant hierarchical sorting networks for wafer scale integration," *IEEE Journal of Solid-State Circuits*, (Sept. 1991).
14. M. Wang, M. Cutler, and S. Y. H. Su, "Reconfiguration of VLSI/WSI mesh array processors with two-level redundancy," *IEEE Trans. Comput.* **c-38** pp. 547-554 (April 1989).
15. C.H. Stapper, F.M. Armstrong, and K. Saji, "Integrated circuit yield statistics," *Proceedings of the IEEE* **71** pp. 453-470 (April 1983).