Fault Tolerance in VLSI Circuits

Israel Koren and Adit D. Singh
University of Massachusetts at Amherst

Fault-tolerant designs of very large ICs primarily attempt to enhance yield. Such designs, first employed in memory chips, now encompass random logic VLSI and wafer-scale circuits.

The primary motive for introducing fault tolerance in VLSI circuits is *yield enhancement*, increasing the percentage of fault-free chips obtained. The active area of monolithic VLSI chips has always been limited by random fabrication defects, which appear impossible to eliminate in even the best manufacturing processes. The larger the circuit, the more likely it will contain such a defect and fail to operate correctly. Thus, the defect density (number of defects per unit of chip area) in any fabrication line limits the size of the largest defect-free chip producible with commercially viable yields. Larger circuits demand a fault-tolerance capability to overcome fabrication defects while avoiding unreasonable costs.

The first VLSI circuits, produced in the 1970s, contained a small number (by today's standards) of devices and consequently an acceptably low average number of defects per chip. Subsequent increases in circuit complexity resulted mainly from higher integration densities produced by reducing the device feature size. Reduced feature sizes tend to make circuits more sensitive to very small defects. However, this was compensated for by reduced defect densities in the more advanced fabrication lines.

As feature sizes enter the submicron level and increased circuit density from reduced feature sizes becomes more difficult to achieve, some designers have turned to larger area circuits and even full-wafer integration. They hope to obtain the cost reductions, size reductions, and performance improvements associated with higher levels of integration.

The prohibitively low defect-free yield of large circuits mandates on-chip fault tolerance for yield enhancement. The incorporation of fault tolerance is sometimes required not only to increase productivity but to ensure feasibility. For example, wafer-scale circuits would have zero yield without fault tolerance.

To see how on-chip redundancy can enhance yield, consider Figure 1, which shows the schematic of a wafer on which individual circuit modules have been fabricated. These modules can be memory arrays, microprocessors, or any other functional units. In the discussion that follows, we assume that these modules are processors. Assume a total of 240 processors fabricated on the wafer and a 0.5 probability that any individual processor is defect-free (meaning the die yield is 50 percent). We can expect an average of 120 good circuits from the wafer.

Suppose we want to implement a four-processor system on a single chip. Ignoring interconnections, each die contains four processors and is four times larger than before. We would obtain a total of 60 multiprocessor chips from the wafer. However, since all four processors must be defect-free for the die to be functional, the yield is $(0.5)^4 = 0.0625$. Thus, we can only expect, on average, $60 \cdot 0.0625$, which equals 3.75 good multiprocessor chips per wafer.

Now consider the possibility of introducing redundancy to provide fault tolerance. Suppose we put five processors on each multiprocessor chip. We now have a functional chip if no more than one processor is faulty. With this single-fault-tolerance capability, the yield of the multiprocessor chip becomes $(0.5)^5 + 5(1-0.5)(0.5)^4 = 0.1875$. Since the chips are bigger now, we will only obtain 240/5 = 48 chips from the wafer, and we can expect an average of $48 \cdot 0.1875 = 9$ good multiprocessor chips per wafer. The redundancy more than doubled the yield in our example.

So far we have assumed that introducing fault tolerance only required a spare processor and no other overhead. In practice, additional switch and interconnect structures are often required to enable the spare to take over

0018-9162/90/0700-0073$01.00 © 1990 IEEE

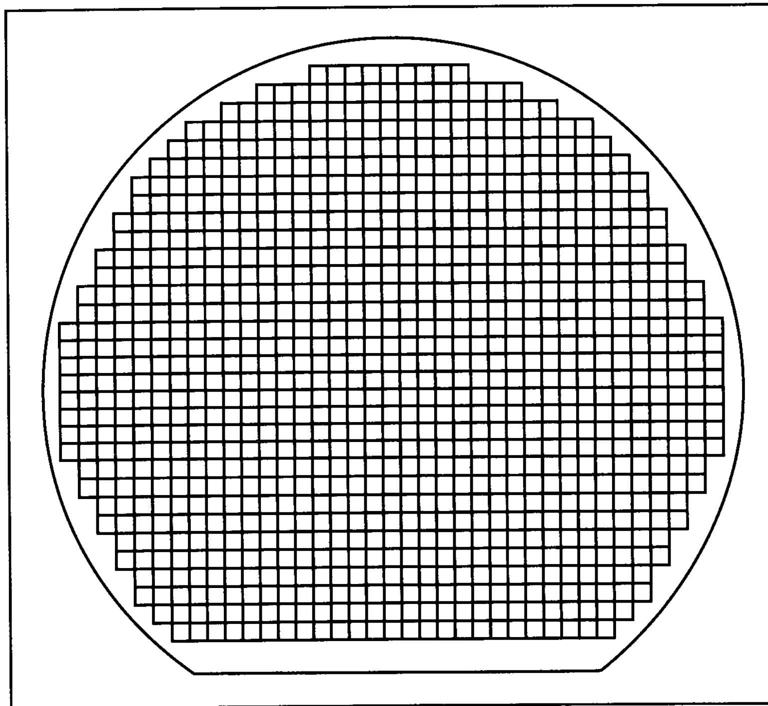

Figure 1. Schematics of a wafer with modules fabricated.

the functions of the failed module (processor). In terms of increased circuitry, this overhead can be substantial, particularly with small modules. Furthermore, often the reconfiguration hardware is not fault tolerant, so a defect in these circuits can be fatal.

To study the effect on yield of the overhead caused by the reconfiguration mechanisms, let us arbitrarily assume that the extra reconfiguration hardware for our example system has an area and yield equal to that of a single processor. The fault-tolerant chip will then have an area equal to that of six individual processors. For the chip to be functional, the reconfiguration circuitry and four processors must be defect-free. This gives a chip yield of $0.5 \cdot ((0.5)^5 + 5(0.5)^5) = 0.09375$. Since each wafer now has 240/6 = 40 chips, we can expect, on average, 3.75 good chips per wafer — no better than the nonredundant design.

A smaller reconfiguration circuit, taking half the area of a processor, does enhance yield, with 6.136 good chips expected per wafer. However, if the reconfiguration overhead exceeds the area of a processor, introducing fault tolerance into the multiprocessor chip can become counterproductive.

We therefore see that fault tolerance schemes for VLSI yield enhancement must be area efficient. Otherwise, the yield improvement due to the fault-tolerance capability might be negated by increased circuit area and the potential for additional faults in the reconfiguration circuitry.

To more accurately analyze the expected yield in the above example and in other defect-tolerant designs, we need to understand the nature of manufacturing defects and the types of restructuring techniques available in practice. In this article we describe the defects that can occur when manufacturing VLSI ICs and the potential resulting faults, some commonly used restructuring techniques for avoiding defective components, and several defect-tolerant designs of memory ICs, logic ICs, and wafer-scale circuits. We will introduce yield models for chips with redundancy to predict the yield of such chips and determine the optimal amount of redundancy.

Defects and faults

We can classify manufacturing defects as *gross area defects* (or global defects) and *spot defects*. Global defects are relatively large-scale defects, such as scratches from wafer mishandling, large-area defects from mask misalignment, over and under etching, etc. Spot defects are random local defects from materials used in the process and environmental causes, mostly unwanted chemical and airborne particles deposited during the various steps of the process.

These two classes of defects contribute to yield losses. In mature, well-controlled fabrication lines, manufacturers can minimize gross area defects. The yield loss due to random spot defects is typically much higher than the yield loss due to global defects. This proves especially true for large-area integrated circuits, since the frequency of global defects is almost independent of the die size. Consequently, spot defects concern us more.

Some spot defects might cause missing patterns or open circuits, while others cause extra patterns or short circuits. We can further classify these defects into *intralayer defects* and *interlayer defects*. Intralayer defects occur as a result of particles deposited during the lithographic processes and are therefore also known as photolithographic defects. Examples include missing metal (or diffusion or polysilicon) and extra metal (or diffusion or poly-Si). Interlayer defects include missing vias between two metal layers or between a metal layer and poly-Si, and shorts between the substrate and metal (or diffusion or poly-Si) or between two separate metal layers. These interlayer defects occur as a result of local contamination, such as dust particles.

Not all spot defects are structural defects resulting in discrete faults such as line breaks and short circuits. A defect causes a discrete fault only if it is large enough to connect two disjoint conductors or disconnect a continuous pattern. Consider, for example, the three circular open-circuit-type defects in the layout of metal conductors depicted in Figure 2. The two left-most defects will not disconnect the corresponding conductors, but the third one will result in a discrete circuit fault.

Some random defects that do not cause structural faults can result in parametric faults, where the electrical parameters of some devices lie outside the desired operational window, affecting the performance of the circuit. For example, an open-circuit-type photolithographic defect, although too small to disconnect a transistor, might affect its performance. Parametric faults can also result from global defects that cause variations in process parameters. In principle, faults (structural or parametric) result from the interaction of defects with the layout geometry. Thus, the probability that a defect will cause a fault might depend on the exact

geometrical position of the defect and on its size, as illustrated in Figure 2.

Now we will describe the method used to determine the percentage of manufacturing defects that result in discrete faults. This type of calculation is necessary to determine the expected number of circuit faults, on the basis of which yield projection (discussed later) is carried out.

The average number of manufacturing defects of type i (for example, photolithographic defects of the open-circuit type) is usually described by d_iA, where d_i denotes the average number of defects (of type i) per unit of area and A is the chip area. To calculate the average number of circuit faults, denoted by λ_i, we define a probability θ_i that a defect of type i will result in a discrete circuit fault. Thus, $\lambda_i = \theta_i d_i A$. The product $A\theta_i$, also called the critical area for defects of type i, is denoted by $A_c^{(i)}$.

The probability that a defect will cause a circuit failure might remain constant for one type of defect or depend on the size of the defects (relative to the physical dimensions of VLSI patterns). For example, the size of an interlayer defect is in most cases relatively small, and the probability that it will cause a circuit failure equals the ratio between the area of the overlapping region and the total area. In contrast, photolithographic (intralayer) defects, like those shown in Figure 2, have a randomly distributed size comparable to that of VLSI patterns. Therefore, the probability that such a defect will cause a failure depends on the pattern shape, its dimensions relative to the size of the defect, and its exact geometrical position.

A commonly adopted assumption presumes circle-shaped defects with diameter x, as shown in Figure 2. Experimental data on defects in many wafers lead to the conclusion that the diameter x of a defect has a density function $f(x)$ that increases as x^q up to the mode x_o of the distribution (that is, the value of x for which the density function is maximal) and then decreases as $1/x^p$ up to a maximum value of x_M. The exact values of q and p are determined empirically. Typical values for these are $q \approx 1$ and $p \approx 3$, for which

$$f(x) = \begin{cases} cx/x_o^2 & \text{if } 0 \le x \le x_o \\ cx_o^2/x^3 & \text{if } x_o \le x \le x_M \\ 0 & \text{if } x > x_M \end{cases}$$

where $c = 1/(1 - 1/2(x_o/x_M)^2)$.

We define the critical area for defects of diameter x as the area in which the center of a defect (of diameter x) must fall to cause a circuit failure. We denote this critical area by $A(x)$ and compute its expected value A_c using

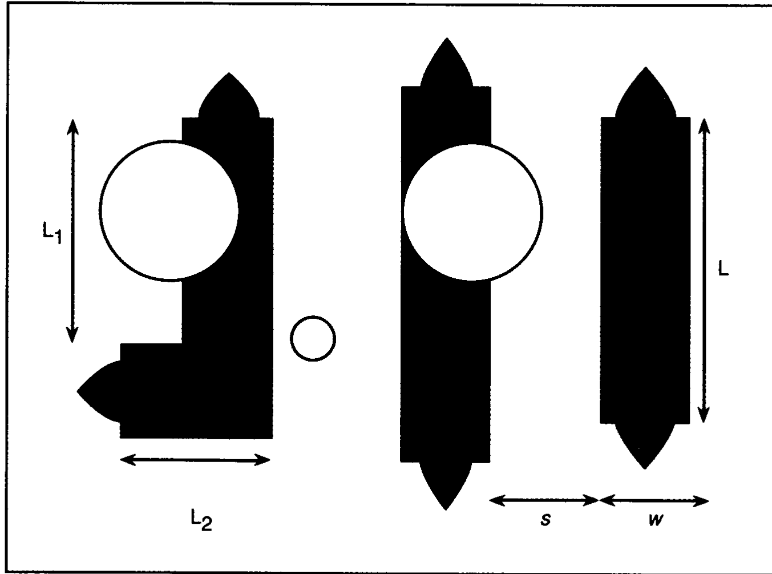

Figure 2. The critical areas in a metal layer. Three circular, open-circuit-type defects are shown.

$$A_c = \int_0^\infty A(x) f(x)\, dx$$

We omitted the subscript i from $A_c^{(i)}$ to simplify the expression. The ratio between A_c and the total area A determines the percentage of defects that cause circuit failures. Its calculation is thus necessary for yield projection. In what follows, we illustrate how to calculate critical areas through the layout in Figure 2, which shows two common geometrical patterns in VLSI layouts: vertical conductors and an L-shaped conductor.

The critical area $A(x)$ for open-circuit defects in a conductor (dark blue) of length L and width w is shown in medium blue in Figure 2. Its size is given by[1]

$$A(x) = \begin{cases} 0 & \text{if } x < w \\ (x-w)\,L + \frac{1}{2}(x-w)\sqrt{x^2-w^2} & \text{if } x \ge w \end{cases}$$

The critical area is a quadratic function of the defect size, but for $L >> w$, the quadratic term in $A(x)$ becomes negligible. Thus, for long conductors we can use the linear term only. We can obtain an analogous expression for $A(x)$ for short-circuit defects in a rectangular area of width s (between two adjacent conductors) by replacing w with s in the above equation.

The L-shaped conductor (depicted in Figure 2) is another common pattern in VLSI layouts. We can approximate its critical area for open-circuit defects with[1]

$$A(x) = \begin{cases} 0 & \text{if } x < w \\ (x-w)(L_1 + L_2) + \frac{1}{2}(x-w)\sqrt{x^2-w^2} + \frac{1}{4}\pi x^2 - (\frac{x}{2} - w)^2 & \text{if } x \ge w \end{cases}$$

The expression for the critical area in this case closely resembles the one for the vertical conductor. Again, if $(L_1 + L_2) >> w$, the linear term in $A(x)$ is the dominant one.

Common VLSI layouts consist of shapes similar to those shown in Figure 2 in different sizes and orientations. Consequently, the exact expression for the critical area of one layout will differ from that of another layout, making it difficult to calculate the critical area of all but very simple and regular layouts. Therefore, two other techniques have been proposed: Monte Carlo simulation and virtual artwork.[2] In the Monte Carlo approach, circles representing defects are placed at random locations of the layout and the critical area is estimated. In the virtual artwork approach, an artificial layout is extracted from the given layout to simplify the estimation of the critical area.[2]

Restructuring techniques

When a VLSI circuit employs fault tolerance for yield enhancement, we must test and reconfigure the components on chip to achieve fault-free operation. For most exist-

ing circuits, wafer-probe testing successfully detects and locates faults. However, several researchers have proposed the inclusion of built-in test capabilities on chip.

Once we have located the faulty components, we must restructure the circuit to avoid those components. Two approaches are employed. In the first, the discretionary wiring approach, the components or cells on the chip are not interconnected to begin with. Good components are connected by laying down additional customized interconnection on the chip. In the second approach, the interconnection structure is fabricated along with the rest of the circuitry. Following testing, the signal-flow paths on the chip are reconfigured using "switches" associated with the interconnections to bypass faulty elements.

The discretionary wiring approach was first used in experimental wafer-scale circuits at Texas Instruments in the mid 1960s. Here, blocks of logic were first tested by wafer probes. A computer program then generated an interconnection pattern to connect good blocks to achieve the desired logic function. This pattern was fed to an electron-beam mask-making machine, which created appropriate interconnection masks. These were then used to lay down the metal interconnection lines and obtain the completed circuit.

These experiments successfully demonstrated the capabilities of the technology, but the cost of creating the custom masks was a major drawback. Also, the discretionary interconnect itself proved susceptible to defects, leading to significant yield losses. Consequently, this technique did not gain commercial acceptability.

Today, discretionary wiring has become an attractive approach for on-chip fault tolerance. New, direct-write, electron-beam lithography machines can quickly create the desired custom metallization patterns directly on the wafer, eliminating expensive intermediate optical masks. Such machines are now widely used to customize low-volume gate arrays. Of course, the problem of additional defects in the customized wiring remains. However, we can minimize these defects by using conservative design rules. We can also set aside uncommitted channels for the discretionary interconnects so they do not run on the uneven surface over active circuitry, where they are much more prone to defects.

Recent research efforts have also focused on other techniques for discretionary wiring. Lasers can etch patterns directly on the wafer. Deposition can then be carried out using the laser to create a chemical reaction in an appropriate gas at the wafer surface. Electron and ion beams have also served in this application. These newer discretionary wiring techniques also allow the repair of a completed VLSI circuit.

The second restructuring approach has the interconnections already fabricated when the circuit is tested. This approach involves customizing the signal paths using "switches" to bypass faulty components. This eliminates the additional processing steps after testing needed by discretionary wiring. Here, the switches that connect and disconnect signal lines can take many forms. The simplest are conventional logic gates. However, these have the problem of volatility. The chip would need reconfiguration at each power up.

Researchers at Lincoln Laboratories have developed electron-beam programmable switches similar to those used in erasable programmable read-only memories. Here, using an electron beam machine we can set a "floating" polysilicon gate isolated in oxide to a high or low logic state as desired. Because the gate is electrically isolated, it holds its charge, and the programming is nonvolatile. This approach has an important advantage: we can program the switches first to aid testing. Once we have identified all the faulty components, they can be set for field operation.

A number of other programmed switch technologies have also been developed. In defect-tolerant memory designs, the redundancy is usually programmed by blowing polysilicon fuses, either by a laser or by electrical overstressing with a high current.

Lincoln Laboratories' Restructurable VLSI technology[3] also provides an example of a switch-based restructuring approach. Here, logic modules are laid out beside uncommitted buses. After testing both the modules and the buses, we make the desired connections by linking (welding together) or cutting apart the interconnections using lasers. The wafer is restructured in steps in such a way that we obtain a sequence of increasingly large subsystems, resulting finally in the complete system. At each step, further tests check for any newly generated defects. To aid in this testing, temporary links connect the subsystem to package test points. At present, every link and cut is tested using optical probing, although the yield for these operations is so high that such testing might be unnecessary in a high-volume environment. The implemention of several different wafer-scale circuits for signal processing applications has demonstrated the viability of Restructurable VLSI technology.

Defect-tolerant designs

Memory ICs were the first integrated circuits to exploit fault-tolerant techniques. Memory chips are particularly dense and therefore extremely vulnerable to manufacturing defects. Moreover, the demand for even higher densities continues to increase. Denser memory chips allow designers to use fewer chips in digital systems, reducing system integration costs, volume, and power dissipation. In addition, the high regularity of memory arrays greatly simplifies the task of incorporating defect-tolerance into their design.

A variety of fault-tolerant techniques with a relatively small overhead have been proposed and successfully implemented in memory ICs. We will review those techniques and then present some recent proposals for defect-tolerant designs of logic ICs and their extensions to wafer-scale integration.

Most methods for incorporating fault-tolerance (that is, redundancy) into VLSI ICs have the following objectives in addition to their main goal of yield enhancement:

(1) No or very limited impact of the added redundancy on performance.
(2) Equal or higher reliability.
(3) Small additional area and power requirements.
(4) Transparency to the user (after chip reconfiguration).
(5) Fault-free ICs requiring no (or limited) additional manufacturing steps.
(6) Defective redundant elements replaceable by other redundant elements.

Increasing the yield of ICs proves especially important for new designs and manufacturing processes, which have a high density of process-induced defects and consequently a low yield. Yield improvements of early prototypes of an IC can reduce the product's introduction time and determine its commercial success. Defect tolerance has proved extremely successful in such cases, and spectacular 30-fold increases in yield have been reported.[4] Yield improvements due to defect tolerance tend to decrease as the manufacturing process matures. But even mature processes with lower defect densities have experienced 1.5-to-3-fold yield increases, proving the effectiveness of defect-tolerance techniques.

Memory ICs. Defect-tolerant designs for yield enhancement started around 1979 with 64-kilobit memories and continue today

with 4-megabit RAMs and beyond. The first approach used in memory ICs, and still the most common, adds spare rows and/or spare columns (also known as word lines and bit lines, respectively). The high regularity of memory arrays allows the use of a limited number of spare rows and columns (that is, a low redundancy overhead) for a large number of repetitive circuits. A defective row or a row containing one or more defective memory cells can be disconnected and then replaced by a spare row. Each spare row has a dedicated programmable decoder, allowing it to replace any defective row. Similarly, spare columns can replace defective ones.

The simplified schematic depicted in Figure 3 illustrates standard and spare row decoders. A fusible link connects a standard decoder to its associated row of memory cells. Similar fusible links are used at the inputs to the spare decoder. These fusible links can be blown individually using one of the techniques described earlier. A spare decoder, as shown in Figure 3, has double the number of inputs that a standard decoder has, for both true and complement inputs. By selectively blowing half of the fuses at its inputs, it can replace any standard decoder whose associated row is defective. The defective decoder will be disconnected by blowing the single fuse at its output.

Note that if a spare decoder is not required because of the small number or absence of defects, it will be deselected and not need any fuse blowing. Also, if, after programming the spare decoder, the associated row of cells is found defective, it can be disconnected by blowing the fuse at the output of the decoder. Another spare can be used.

The number of spare rows and/or columns is determined to optimize the yield, after taking into account the additional area required for the redundant circuitry and the probability of defects occurring in these circuits. We will describe the method used to determine the optimal amount of redundancy later.

After manufacturing, testing the chips determines the location of defects. Then the chips are reconfigured by disabling the defective rows and/or columns and programming the decoders of spare rows and columns to replace the defective ones. Thus, the repair of defective memory ICs consists of three phases: a diagnosis phase to detect and locate all defective memory cells; a repair-analysis phase to allocate spare rows or columns to all faulty cells; and a repair phase to disconnect the defective cells and program the allocated spares. All three phases are performed in a fully automatic manner and require no manual intervention.[4]

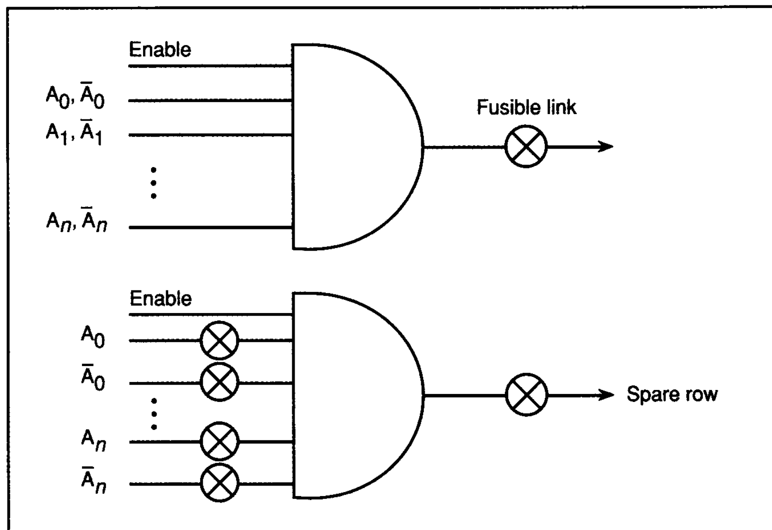

Figure 3. Simplified schematic of standard and spare row decoders.

Error-correcting codes. Manufacturers frequently use error-correcting codes (ECCs) in large memory systems to mask intermittent faults. Thus, using ECCs for yield enhancement can contribute to reliability improvement as well. However, the associated area overhead is much higher than with the simple spare row/column scheme.

For an example of a memory IC employing an ECC for yield enhancement, consider Mostek's 1-megabit ROM, in which seven parity bits added to the 64 data bits increased area more than 11 percent. The 71 memory cells selected simultaneously are positioned within the array so that any two selected cells are separated by 15 unselected cells. This allows the chip to tolerate not only single-cell failures but also clusters of multiple cell failures. A maximum of 16 kilobit failures out of the 1 megabit can be corrected. The use of ECCs might slow the memory, since the error-detection circuitry lies in the critical path. This circuit was therefore designed to minimize the increase in access time.

Recently, IBM developed an experimental 16-megabit dynamic RAM, adding nine check bits to every 128 data bits. This chip combines ECCs with more traditional bit and word-line redundancy and achieves higher yield enhancement.

Associative approach. The spare row/column approach applies only to the replacement of individual faulty rows or columns. If we need to replace larger blocks of cells, as might happen with clustered rather than uniformly distributed defects, an associative approach as developed by Haraszti at Hughes Aircraft[5] looks attractive. The address of the defective block is stored in an associative memory, and any incoming request to an address within the defective block will be redirected to a spare block. The spare block has a smaller size compared to the main memory array and, consequently, its access time is substantially smaller. Thus, even with the additional time required to access the associative memory (accessed in parallel to the main memory) before the spare block can be accessed, the overall access time increase is less than 2 percent.[5] The increase in power consumption is insignificant (less than 0.6 percent), but the area increase is substantially higher than for the spare row/column scheme, ranging from 10 percent for 64 kilobits to 27 percent for 1 megabit.

The associative approach can be extended to a hierarchical replacement scheme, where a large spare block can itself be repaired by another, smaller, spare block.

Partially good chips. A different approach to yield enhancement suggests the use of partially good chips. If sections in a 1-megabit memory are defective beyond repair, we can reconfigure the chip to a usable 0.5-megabit chip or even a 0.25-megabit chip. To do this, we must partition the circuitry of the chip in such a way that fault-free sections can function independently. Several manufacturers of memory ICs, like

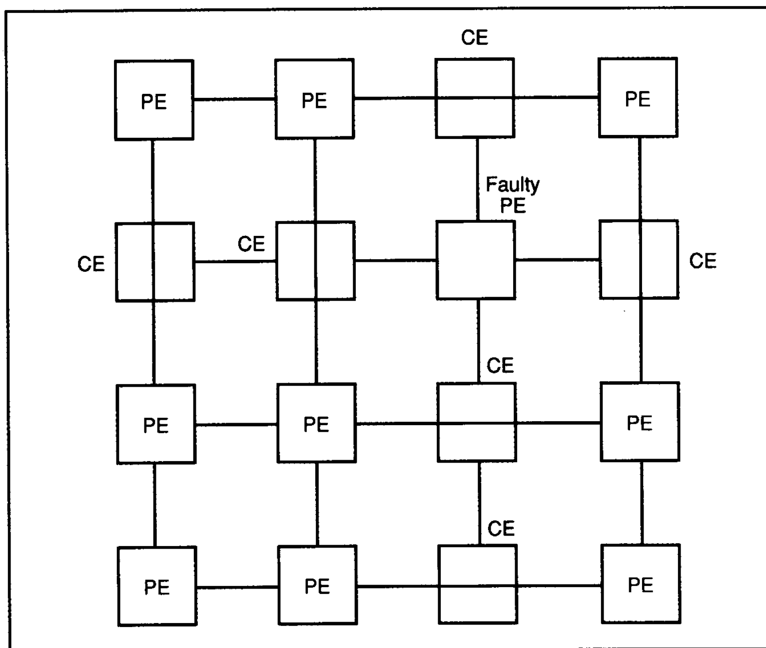

Figure 4. Row and column exclusion scheme. (PE = processing element; CE = connecting element)

Motorola, IBM, and Westinghouse, have used this technique successfully.

Note that this technique is orthogonal to other defect-tolerance schemes. For example, the individual sections within the chip might have spare rows and columns. Only when the available redundancy within a section is insufficient to overcome all the defects present in this section will the section be declared unusable.

IBM refined the idea of using partially good chips, further dividing the independent sections in the memory chip into smaller blocks. These can be used separately after proper alignment by steering the data bits to the right positions.

Logic ICs. The development of efficient defect-tolerant designs for random logic ICs like microprocessors is considerably more complex than for memory ICs. However, if some regularity exists in the structure of a given logic circuit, it might be possible to incorporate redundancy. A natural target for defect-tolerant designs — programmable logic arrays (PLAs) — have a regular structure and implement random logic circuits in VLSI chips. The control sections of many microprocessors use large PLAs. Some designs have employed PLAs with as many as 50 inputs and almost 200 product terms. Since these PLAs require large silicon areas, the incorporation of redundancy in their design can considerably improve the overall yield.

Researchers have investigated defect-tolerant designs of PLAs[6] and proposed adding spare programmable product lines, input lines, and output lines to protect against all types of possible defects. This technique resembles the redundant row/column scheme for memory ICs. However, unlike memory ICs, where all defects can be identified by applying test patterns externally, the identification of defects in a PLA requires some built-in testing aids, like adding inputs to the AND plane.

Defect-tolerant PLAs with spare programmable product lines and added inputs for defect identification have recently been implemented within a 16-bit microprocessor.[7] This microprocessor also includes a defect-tolerant data path. A microprocessor's data path includes arithmetic and logic units, registers, and buses and usually occupies a large percentage of the overall area. A bit-sliced data path, with the inclusion of one or more spare slices, can exploit the regularity in the circuit. However, not all parts of the data path are regular. For example, the logic circuits associated with the status bits are highly irregular. Since we cannot replace such parts with common spare circuits, we must exclude them from the bit-slice organization.

In logic ICs with irregular structures, duplication or even triplication of certain circuits might prove beneficial. If we use duplication, we must employ fault identification and then restructuring after manufacturing. In the case of triplication, we can avoid these additional steps by using a majority voter at the output, if only one defective circuit out of the identical ones is allowed to fail.

In its attempt to build a mainframe based on wafer-scale emitter-coupled-logic technology, Trilogy employed replication for defect tolerance in random logic. However, the extremely large overhead (2-fold and up) associated with these techniques has substantially limited their use in general.

Wafer-scale integration. As we have seen, some manufacturers have already successfully employed on-chip fault tolerance to enhance the yield of high-density semiconductor memory chips. That approach works because, given the regular structure of memory cell arrays, a small amount of redundancy only slightly increases the circuit area while significantly increasing the number of good dies obtained from a wafer.

Since regular structures seem better suited for on-chip redundancy, processor arrays look like attractive candidates for full-wafer integration. Array architectures are being widely investigated to speed up the performance of computer systems through parallelism. For well-matched problems, they promise orders of magnitude improvement in performance over traditional sequential computers. If we could sufficiently reduce the cost and size of such specialized parallel processing arrays, they would be much more widely used.

Realizing the desired cost and size reductions for parallel processing arrays depends on VLSI and wafer-scale integration (WSI) technology. Implementing a processor array in a monolithic integrated package can also facilitate significantly greater operating speeds, because interprocessor signals will not have to be driven off chip. Off-chip signal propagation in metal-oxide-semiconductor VLSI systems is considerably slower than signal propagation within the chips. The same problem exists, to a lesser extent, with the new silicon-on-ceramic and silicon-on-silicon hybrid technologies competing with WSI for high-density packaging.

As device dimensions shrink further into the submicron range, reducing channel transit times to tens of picoseconds, this problem

will become more acute. Signal propagation delays can critically limit the operating speed of VLSI systems, particularly highly pipelined array architectures employing fine-grained pipeline stages and extremely high clock rates. For example, a 330-MHz pipelined array multiplier reported by Siemens in West Germany would likely not be viable in a multichip implementation. Since many proposed array architectures require considerably more silicon than a conventionally sized die, the designs will clearly require WSI technology to achieve their full potential. Such an implementation might also improve reliability by eliminating mechanical and electrical failures often observed at the pins and interconnection of traditional board-level designs.

Several defect-tolerance approaches have been proposed for processor arrays. In one of the earliest schemes, the row and column exclusion approach,[8] redundant rows and columns of processors are implemented in the array. Each processor converts into a connecting element (CE) if required. If so, it no longer performs computational operations but merely passes signals from input to output. If a processor in the array fails, all other processors in the row and column containing the failed processor become CEs, as shown in Figure 4. A reconfigured fault-free array with one less row and column results. The failure of a horizontal (vertical) link between two adjacent processors requires turning only the processors along the corresponding row (column) into CEs.

While attractive in its simplicity and low hardware overhead, the above scheme only proves effective when we expect few failures. Since an entire row and column must usually be disabled for each fault, multiple faults quickly degrade the array. To get around this problem, several other restructuring schemes have been proposed.

Figure 5 illustrates a simple scheme that adds spare columns to the array. The processors are reindexed in their rows so as to skip over the faulty processors. Once the reindexing has completed, the appropriate vertical connection can be made. For s spare columns, this scheme can tolerate up to s faults in each row. However, it requires a complex switch and interconnection structure to support this reconfiguration. Increased interconnection complexity can reduce yield because of increased area and also because of the possibility of defects in the interconnection.

A large number of other restructuring schemes for mesh arrays have been proposed. The objective generally is to reduce the probability that an available spare cannot replace a failed processor, while minimizing the restructuring overhead. Obviously, the more elaborate schemes only benefit arrays of relatively large processors, where the total silicon area consumed by the processors is large compared to the redundant interconnection.

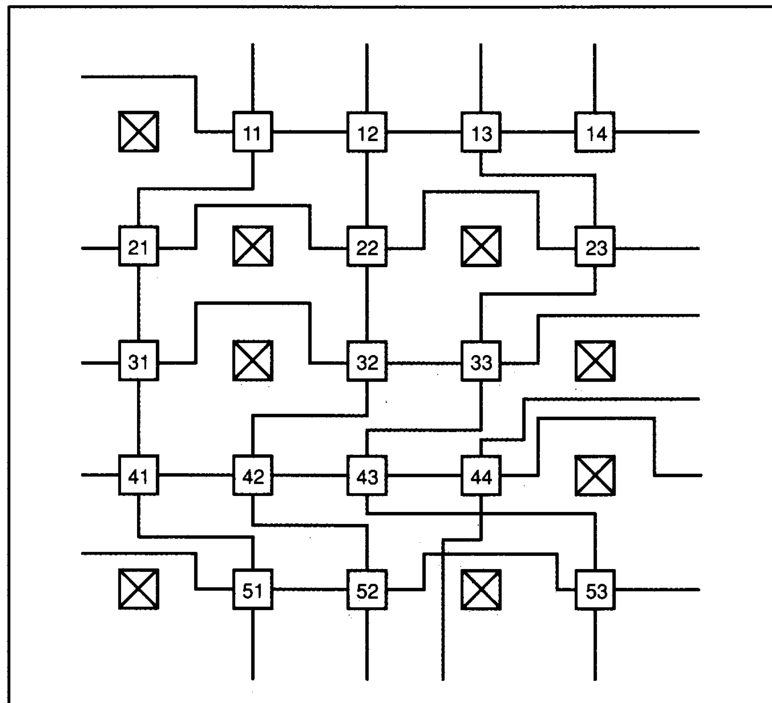

Figure 5. An improved reconfiguration strategy.

The reconfiguration approach employed in Figure 5 can lead to relatively long links. This can negate the performance benefits of WSI, particularly in synchronous designs, where we must slow the clock to accommodate the longest delay. Furthermore, since we do not know a priori which interconnections will need reconfiguration to bypass failed nodes, we must implement all interconnections with powerful driver circuits capable of driving the worst-case restructured interconnections with acceptable delay. This can impose very significant area, power, and delay penalties on the design.

An alternative allows us to ensure at design time that the restructured interconnects will be short and bounded in length. The interstitial redundancy scheme[9] illustrated in Figure 6 achieves this. This approach systematically introduces spare processors throughout the array. Each spare can replace any neighboring primary processor. Since the reconfiguration is local, restructured interconnections stay short. Optimal assignment of spares to failed primary processors employs a bipartite graph that has failed primary processors as one set of vertices and operational spares as the other set. An edge connects a failed primary to a spare if the spare can replace it. Well-known matching algorithms can quickly find an assignment that covers all the failed primary processors, if such an assignment exists.

Researchers using defect-tolerance strategies similar to those outlined above have implemented a number of experimental wafer-scale systems. Digital signal processing systems fabricated through Restructurable VLSI technology developed at Lincoln Labs have demonstrated the practicality of wafers with a few heterogeneous cell types. The ESPRIT (European Strategic Programme for Research in Information Technology) project also aggressively pursues WSI, including memory, microprocessor, and array processor designs.

The ELSA (European Large SIMD Array) two-dimensional array processor[10] employs a two-level hierarchical defect-tolerance approach. Since the processors operate on a single bit at a time, they are quite small, and several (12×7) can fit on a conventionally sized chip measuring 6×6 square

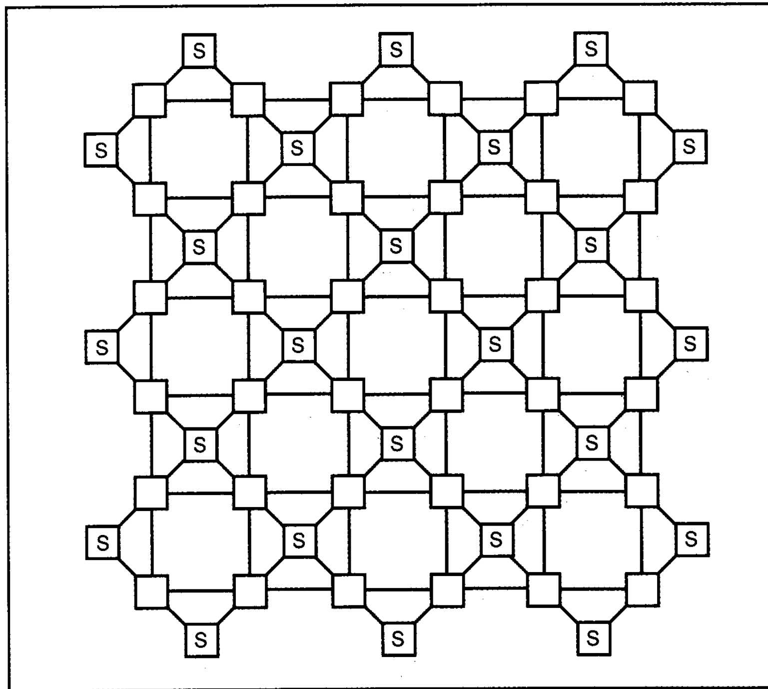

Figure 6. The interstitial redundancy scheme.

millimeters. One column within this chip is redundant, so it can tolerate a single fault and still form a 12×6 array. At the wafer level, the chips are connected by a two-track interconnection network which can bypass faulty chips that contain more than one faulty processor or fail for other reasons.

The 3D Computer being developed by Hughes Research Laboratory from wafer-scale circuits employs the interstitial redundancy scheme to ensure short restructured interconnections. The physical location of a cell is close enough to its logical location in the array to permit making the interwafer connections between the cells. A 32×32 processor prototype rated at 600 million operations per second has already been demonstrated, and a 128×128 design is under development.

Yield estimations

Designers considering a defect-tolerance technique for a VLSI circuit must estimate the projected yield. This allows them to determine the optimal amount of redundancy and the suitability of a proposed reconfiguration scheme.

The difficulty in modeling the yield of fault-tolerant IC chips arises mainly from the clustering of manufacturing defects during chip fabrication. Yield modeling proves relatively simple when we use Poisson statistics to describe the distribution of faults per chip. According to this distribution, the probability of having exactly x faults in a chip is given by

$$Prob\ \{X = x\} = \frac{e^{-\lambda}\lambda^x}{x!} \quad (1)$$

where X is a random variable denoting the number of faults and λ denotes the average number of faults expected per chip. For chips with no redundancy the yield is

$$Y = Prob\ \{X = 0\} = e^{-\lambda} \quad (2)$$

As discussed earlier, the average number of faults per chip is given by

$$\lambda = \sum_i d_i A_c^{(i)} \quad (3)$$

where d_i represents the density of type i defects and $A_c^{(i)}$ is the critical chip area for type i defects.

Practitioners have known since the early days of IC manufacturing that the above yield formula is too pessimistic and leads to predicted chip yields lower than actual yields. It later became clear that the low predicted yield resulted from ignoring the clustering of faults, a phenomenon observed in practice.

Proposed modifications to the above yield formula attempt to account for fault clustering. The most commonly used modification assumes the number of faults to be Poisson distributed, but considers the parameter λ to be a random variable rather than constant. Making λ a random variable results in clustering of faults, no matter what the type of distribution assumed for λ.

We obtain the modified yield formula by averaging yield formula (2) with respect to a probability density function of λ, denoted by $f(\lambda)$:

$$Y = \int_0^\infty e^{-\lambda} f(\lambda)\, d\lambda \quad (4)$$

The function $f(\lambda)$ is known as a compounding function. Several compounding functions proposed in the past lead to different yield formulas. A common one, the Gamma distribution,[11] results in the well-known yield formula

$$Y = (1 + \bar{\lambda}/\alpha)^{-\alpha} \quad (5)$$

where α is called the clustering parameter and $\bar{\lambda}$ is the average number of faults per chip. We can show that $\bar{\lambda}$ is, in effect, the expected value of l. When the clustering parameter α is large, that is, when $\alpha \to \infty$, the yield in expression (5) becomes equal to yield formula (2). This represents the case of random faults and no clustering. Smaller values of a indicate increased clustering. Experimentally derived values for α typically range between 0.3 and 5.

Applying the same compounding procedure to the Poisson probability function for the number of faults in expression (1) results in the negative binomial distribution:

$$Prob\ \{X = x\} = \frac{\Gamma(\alpha + x)}{x!\,\Gamma(\alpha)} \frac{(\bar{\lambda}/\alpha)^x}{(1 + \bar{\lambda}/\alpha)^{\alpha + x}} \quad (6)$$

Yield formula (5) accounts only for faults resulting from spot defects. To account for gross area defects affecting large wafer areas, we must include a gross yield factor Y_0 in the yield model:

$$Y = Y_0(1 + \bar{\lambda}/\alpha)^{-\alpha} \quad (7)$$

Yield models for chips with redundancy. IC chips frequently include several replicated circuit modules. We can often use chips containing a number of identical modules (of one type or more) even if some of the mod-

ules do not function correctly, obtaining in this way partially good chips. Alternatively, we can add a few redundant modules to the design and accept only those chips with the necessary number of fault-free modules.

Consider chips with a single type of identical modules. Let N denote the number of these modules. Define the following probability:

$$a_{M,N} = Prob\ \{\text{Exactly } M \text{ out of the } N \text{ modules are fault-free}\} \quad (8)$$

We can use this probability to calculate the yield of chips with redundancy and the yield of partially good chips. For example, if R out of N modules are spares, meaning that a chip with at least $(N-R)$ fault-free modules is acceptable, then the yield of the chip is given by

$$Y = Y_0 \sum_{M=N-R}^{N} a_{M,N} \quad (9)$$

To derive an expression for $a_{M,N}$, we need to know how to calculate the yield of a subset of M modules. To this end we have to make some assumptions regarding the change in the parameters $\bar{\lambda}$ and α of the yield formula when considering partial areas. The average number of faults depends linearly on the number of modules, M. However, the dependence of the clustering parameter, α, on M is less straightforward.

Most papers on IC yield that took fault clustering into account assumed the parameter α is the same when considering the whole chip or only part of the chip. This assumption was based on "large-area clustering," meaning the clusters of defects exceed the chip size. This assumption often proved reasonable, since most clustering is caused by wafer-to-wafer variations of fault densities, especially for small-area chips.

If we assume large-area clustering, we can calculate $a_{M,N}$ by first computing the probability that a given number of faults occurs in the complete chip, then distributing these faults uniformly among the N modules. Thus, the probability that exactly $(N-M)$ modules will contain faults is

$$a_{M,N} = \sum_{x=N-M}^{\infty} Q^{(N)}_{x,(N-M)} \cdot Prob\ \{X_N = x\} \quad (10)$$

where $Prob\{X_N = x\}$ is the probability that the chip has x faults and $Q_{x,j}^{(N)}$ is the probability that, given x faults, the faults are distributed among exactly j out of N modules. Assuming that faults are distinguishable, the latter equals

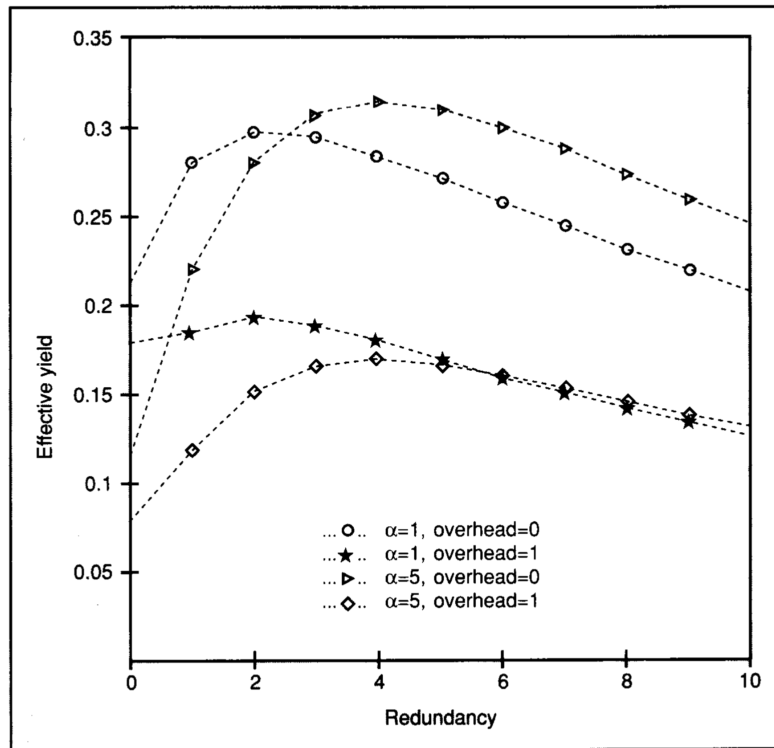

Figure 7. The effective yield versus amount of redundancy (Y_0 = 0.9).

$$Q^{(N)}_{x,j} = \sum_{k=0}^{j} (-1)^k \binom{N}{k} \binom{N-k}{N-j} \left[\frac{j-k}{N}\right]^x$$
$$\text{for } x \geq j \text{ and } 0 < j \leq N \quad (11)$$

If we assume the negative binomial distribution from expression (6), then the above equation yields

$$a_{M,N} = \sum_{k=0}^{N-M} (-1)^k \binom{N-M}{k} \binom{N}{M} \left[1 + \frac{(M+k)\bar{\lambda}}{\alpha}\right]^{-\alpha} \quad (12)$$

We obtain the negative binomial distribution from the Poisson distribution by averaging over all values of λ, using the Gamma distribution function. This compounding procedure can be applied to any statistical measure. We can derive an expression for the desired measure by assuming the convenient Poisson distribution (whose most useful property is the statistical independence between faults in different modules). We can then apply the compounding procedure to obtain the required expression for the negative binomial model. This powerful compounding procedure has been employed to derive yield expressions for interconnection buses in VLSI chips[12] and for partially good memory chips.[11]

The simple architecture analyzed in the preceding section is an idealization; actual chips rarely consist entirely of identical circuit modules. All chips include support circuits (like power supply lines, clocks, input and output buffers, etc.) shared by the replicated modules. The chips become unusable when support circuits are damaged. Since the clustering of the support circuit faults is not independent of the clustering of the module faults, we need to include in expression (12) the average number of faults that cause defects in these support circuits. This results in

$$a_{M,N} = \sum_{k=0}^{N-M} (-1)^k \binom{N-M}{k} \binom{N}{M} \left[1 + \frac{\bar{\lambda}_{CK} + (M+k)\bar{\lambda}}{\alpha}\right]^{-\alpha} \quad (13)$$

where $\bar{\lambda}_{CK}$ is the average number of fatal faults — or *chip-kill* faults — in the support circuits.

We restricted the discussion above to the case where redundancy is provided to toler-

ate faults in a single type of circuit modules. However, the results have been extended to fault-tolerant chips with multiple types of modules.[13]

The yield expression in (9) can help us find the optimal amount of redundancy for a given fault-tolerant scheme. The optimal redundancy maximizes the number of acceptable chips per wafer. When the redundancy increases, the yield of the individual chip tends to increase, but the number of chips per wafer tends to decrease. We therefore need to maximize the *effective yield*, which is the chip yield multiplied by the ratio between the number of chips with and without redundancy on the same size wafer.

Figure 7 depicts the effective yield of the four-processor chip described at the beginning, with the yield of a single processor being 0.5. Unlike the simplified analysis given earlier, here we take into account the clustering phenomenon and the gross yield factor Y_0, for which we assume the value 0.9. Also, assume defects in the reconfiguration overhead area (for switches and interconnections) are chip-kill faults.

The four curves in Figure 7 show that the effective yield increases when we incorporate redundancy into the design until we reach an optimal value of redundancy. Beyond this value, the effective yield declines; additional redundancy contributes only to the area, not to the number of acceptable chips per wafer. As shown in Figure 7, the optimal redundancy for α=5 (only limited clustering) is R=4. This is independent of the reconfiguration overhead, which changes from 0 to 1 times the area of one processor. The resulting number of acceptable chips from the wafer differs, though. We can obtain this number by multiplying the effective yield by the number of chips when no redundancy is introduced (that is, 60). Thus, an average of 10.11 and 18.88 acceptable chips per wafer are predicted for reconfiguration overheads of 1 and 0, respectively.

Clustering affects the projected yield and the resulting optimal amount of redundancy. For example, for α=1 the optimal redundancy is 2, with a projected number of acceptable chips of 11.64 and 17.84 for overheads of 1 and 0, respectively. Ignoring the clustering phenomenon for the sake of simplifying the task of yield projection might lead to incorrect decisions regarding the amount of redundancy to incorporate.

Defect-tolerant techniques for VLSI circuits have made remarkable progress in recent years. As we begin the 1990s, the theoretical approaches for introducing and optimizing redundancy, as well as restructuring technologies, appear to be in place for more widespread use of redundancy for yield enhancement. Meanwhile, as VLSI feature sizes approach physical limits, it is likely the need for larger area chips to meet the growing demand for more complex monolithic systems will become much more pressing. We can expect the use of defect-tolerance techniques to provide viable yields for these large chips to become routine in such an environment.

The largest circuits achievable through defect-tolerance techniques are full-wafer designs. After the disappointments of the early 1980s, progress here appears on track. Tadashi Sasaki, in his invited talk at the 1989 International Conference on Wafer-Scale Integration, pointed out that earlier trends (including those underpinning Japanese long-range planning) had predicted a WSI technology in place by the year 2000. However, progress has outstripped this schedule, and we can expect to see WSI systems in the 1990s.

Just as the most widespread use of defect-tolerant design today occurs in memory circuits, the first large-volume wafer-scale circuits will also likely be memory systems. One such circuit, the 40-megabyte wafer-stack module recently developed by Anamartic, consists of two 6-inch-diameter wafers. This module relies on the already well-established theory of fault tolerance in VLSI circuits and the well-developed technology of restructuring. Future circuits will as well. ■

Acknowledgments

We wish to thank the reviewers for their helpful comments. We also thank Yaron Koren for his help in preparing the figures.

Israel Koren's work was supported in part by the National Science Foundation under contract MIP-8805586.

Adit Singh's work was supported in part by the NSF under contract MIP-8808325.

References

1. I. Koren, "The Effect of Scaling on the Yield of VLSI Circuits," *Yield Modeling and Defect Tolerance in VLSI*, W.R. Moore, W. Maly, and A. Strojwas, eds., Adam Hillger Ltd., Bristol, UK, 1988, pp. 91-99.
2. W. Maly, W.R. Moore, and A. Strojwas, "Yield Loss Mechanisms and Defect Tolerance," *Yield Modeling and Defect Tolerance in VLSI*, W.R. Moore, W. Maly, and A. Strojwas, eds., Adam Hillger Ltd., Bristol, UK, 1988, pp. 3-30.
3. P.W. Wyatt and J.I. Raffel, "Restructurable VLSI — A Demonstrated Wafer Scale Technology," *Proc. 1989 Int'l Conf. Wafer-Scale Integration*, Jan. 1989, IEEE Computer Society Press, Los Alamitos, Calif., Order No. 1901, pp. 13-20.
4. R.T. Smith et al., "Laser Programmable Redundancy and Yield Improvement in a 64K DRAM," *IEEE J. Solid-State Circuits*, Oct. 1981, pp. 506-513.
5. T.P. Haraszti, "A Novel Associative Approach for Fault-Tolerant MOS RAM," *IEEE J. Solid-State Circuits*, June 1982, pp. 539-546.
6. C.L. Wey, " On Yield Considerations for the Design of Redundant Programmable Logic Arrays," *IEEE Trans. Computer-Aided Design*, Vol. CAD-7, Apr. 1988, pp. 528-535.
7. R. Leveugle, M. Soueidan, and N. Wehn, "Defect Tolerance in a 16-Bit Microprocessor," *Defect and Fault Tolerance in VLSI Systems*, Vol. 1, I. Koren, ed., Plenum, New York, 1989, pp. 179-190.
8. I. Koren, "A Reconfigurable and Fault-Tolerant VLSI Multiprocessor Array," *Proc. 8th Ann. Symp. Computer Architecture*, May 1981, pp. 425-441.
9. A.D. Singh, "Interstitial Redundancy: An Area-Efficient Fault-Tolerance Scheme for Larger Area VLSI Processor Arrays," *IEEE Trans. Computers*, Vol. 37, No. 11, Nov. 1988, pp. 1,398-1,410.
10. J.-L. Patry and G. Saucier, "Practical Experiences on the Design of a WSI 2D Array," Preprints, *1989 IEEE Int'l Workshop Defect and Fault Tolerance in VLSI Systems*, Oct. 1989, pp. 51-63. Also to appear in *Defect and Fault Tolerance in VLSI Systems*, Vol. 2, C.H. Stapper, V.K. Jain, and G. Saucier, eds., Plenum, New York, 1990.
11. C.H. Stapper, A.N. McLaren, and M. Dreckmann, "Yield Model for Productivity Optimization of VLSI Memory Chips with Redundancy and Partially Good Product," *IBM J. Research and Development*, Vol. 20, 1980, pp. 398-409.
12. I. Koren, Z. Koren, and D.K. Pradhan, "Designing Interconnection Buses in VLSI and WSI for Maximum Yield and Minimum Delay," *IEEE J. Solid-State Circuits*, June 1988, pp. 859-866.
13. I. Koren and C.H. Stapper, "Yield Models for Defect-Tolerant VLSI Circuits: A Review," *Defect and Fault Tolerance in VLSI Systems*, Vol. 1, I. Koren, ed., Plenum, New York, 1989, pp. 1-21.

Further reading

Chean, M., and J.A.B. Fortes, "A Taxonomy of Reconfiguration Techniques for Fault-Tolerant Processor Arrays," *Computer*, Vol. 23, Jan. 1990, pp. 55-69.

Fuchs, W.K., and M.F. Chang, "Diagnosis and Repair of Large Memories: A Critical Review and Recent Results," *Defect and Fault Tolerance in VLSI Systems*, Vol. 1, I. Koren, ed., Plenum, New York, 1989, pp. 213-225.

McDonald, J.F., et al., "The Trials of Wafer-Scale Integration," *IEEE Spectrum*, Vol. 21, No. 10, Oct. 1984, pp. 32-39.

Moore, W.R., "A Review of Fault-Tolerant Techniques for the Enhancement of Integrated Circuit Yield," *Proc. IEEE*, Vol. 74, May 1986, pp. 684-698.

Stapper, C.H., F.M. Armstrong, and K. Saji, "Integrated Circuit Yield Statistics," *Proc. IEEE*, Vol. 71, Apr. 1983, pp. 453-470.

Swartzlander, E.E., ed., *Wafer-Scale Integration*, Kluwer Academic Publishers, Boston, 1989.

Tewksbury, S.K., *Wafer-Level Integrated Systems: Implementation Issues*, Kluwer Academic Publishers, Boston, 1989.

Walker, D.M.H., *Yield Simulation for Integrated Circuits*, Kluwer Academic Publishers, Boston, 1987.

Israel Koren is currently a professor of electrical and computer engineering at the University of Massachusetts, Amherst. His research interests are fault-tolerant VLSI and WSI architectures, models for yield and performance, floor-planning of VLSI chips, and computer arithmetic.

Koren received the BS, MS, and PhD degrees from the Technion, Israel Institute of Technology, in Haifa in 1967, 1970, 1975, respectively, all in electrical engineering. He edited and coauthored the book *Defect and Fault Tolerance in VLSI Systems*, Vol. 1, Plenum, 1989. Koren is a senior member of the IEEE.

Adit D. Singh is a guest editor of this issue of *Computer*. His photo and biography follow the "Guest Editors' Introduction."

Readers may contact the authors at the Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003

SUPERCOMPUTING MANAGEMENT IN THE '90s

At Sterling ZeroOne, Inc., our alliance with NASA/Ames Research Center keeps us on the leading edge in supercomputing. We provide systems programming, user services, graphics and operations support for multiple supercomputers, including a CRAY Y-MP/832 running a UNIX*-based operating system. We are continuously exploring new ideas for enhancing productivity of research scientists working in Computational Fluid Dynamics, Computational Chemistry, Space Sciences and other disciplines.

If you are interested in making a key contribution in a state-of-the-art environment, we invite you to consider the following high-visibility role:

Manager of User Services

You will manage a group of highly-skilled user consultants responsible for supporting a large community of NASA/Ames research scientists. A primary goal of this group is to enhance user productivity in the areas of distributed computing, visualization, algorithms, optimization, and performance of codes. You will establish projects and objectives, and manage resources for their completion. One of your most important responsibilities will be to establish and maintain a proactive technical liaison with the user community.

Your PhD/MS, preferably in a physical science or mathematics, must be augmented with 8 years' related experience, including large-scale scientific computing and a strong technical management background. UNIX and CRAY experience is highly desirable.

Located in San Francisco's scenic Bay Area, you'll enjoy easy access to some of the area's best attractions. Sterling ZeroOne, Inc. is a wholly-owned subsidiary of Sterling Software, Inc. We offer competitive salaries and a generous benefits package that includes a 401(k) plan with company contribution. For immediate consideration, please contact us at: Sterling ZeroOne, Inc., NASA/Ames, M/S 233-3, Dept. USM/IEEE, Moffett Field, CA 94035-1000, or FAX your resume to (415) 964-1760. We are an equal opportunity employer M/F/H/V. PRINCIPALS ONLY, PLEASE. U.S. CITIZENSHIP IS REQUIRED.

*UNIX is a registered trademark of AT&T.