Fault-simulation programs for integrated-circuit yield estimations

by C. H. Stapper

Three programs are described here which have been used for integrated-circuit yield modeling at the IBM facility in Essex Junction, Vermont. The first program generates negative binomial distributions which are used to represent the frequency distribution of the number of faults per chip. Calculations with the generalized combination function *A ! B* in *APL* are limited to simulations of up to 99 999 faults, and can take too much computer time to run. These limitations are eliminated when the calculations make use of the scan function. The second program simulates clustered fault locations on a map. The clusters are initially generated using a radial Gaussian probability distribution. Each fault location is stored as a complex number, which facilitates the use of cluster-shaping programs that are also described. In a third program, another simulator of fault maps, faults are added as a function of time. This program also results in fault distributions that are clustered. In addition, it produces frequency distributions that very closely approximate negative binomial distributions.

Introduction

Since 1973 computer programs written in *APL* have been used successfully for yield planning and yield analysis of a variety of memory chips, logic chips, and microprocessor chips manufactured at the IBM facility in Essex Junction, Vermont, including the IBM 64K, 256K, 288K, and 1-Mb dynamic random-access memory (DRAM) chips [1, 2]. All of the DRAM chips contained redundant bits to replace defective ones, while partially good chips were also used. This had to be taken into account in developing yield estimations for these chips. The models used for doing this have been published previously [3–7]. Described here are *APL* programs that have been used to evaluate new yield models needed to estimate yields and factory requirements for 4-Mb DRAM chips and memory chips with higher bit densities [8].

Negative binomial distribution

The frequency distributions of the number of fabrication faults occurring on integrated-circuit chips can often be modeled with negative binomial distributions. These distributions have been used for the following purposes: estimation of the yield of fault-tolerant VLSI multiprocessors by Koren et al. [9–11]; evaluation of the yields of memory chips containing redundant circuits by Stewart [12]; simulation of integrated circuit yield by Walker [13, 14]; simulation of wafer scale integration yield by Harden and Strader [15]; and optimization of the productivity of programmed logic arrays by Wey [16]. Negative binomial distributions have also led to formulations of yield variations by Foard Flack [17], and

to interval estimates of yield by Winter and Cook [18]. Such distributions can be expressed mathematically as

$$P(X = k) = \frac{\Gamma(\alpha + k)}{k!\Gamma(\alpha)} \frac{(\lambda/\alpha)^k}{(1 + \lambda/\alpha)^{\alpha+k}}, \quad (1)$$

where Γ represents the gamma function, λ the average number of faults per chip, and α a cluster parameter. In this expression the quantity

$$\frac{\Gamma(\alpha + k)}{k!\Gamma(\alpha)} \quad (2)$$

can be calculated with an *APL* program that uses the primitive dyadic generalized combination function " ! " in the form `K ! ALPHA+K-1`. However, with `K ≥ 100 000` and typical values of `ALPHA < 1` this function produces *DOMAIN ERROR* messages. The probabilities can therefore only be evaluated for up to 99 999 faults per chip.

The *APL* function originally used for calculating the binomial distribution has the form

```
     ∇
[0]    X←N NBINOM LA;ALPHA;LOA;K
[1]    ⍝THIS FUNCTION CALCULATES THE
       NEGATIVE BINOMIAL
[2]    ⍝OR POLYA-EGGENBERGER
       DISTRIBUTION WITH
[3]    ⍝PARAMETERS LAMBDA=LA[1],
       ALPHA=LA[2]
[4]    X←(K ! ALPHA+K-1)
       ×((LOA÷1+LOA)*K←0,⍳N)
       ÷(1+LOA←+/LA)*ALPHA←LA[2]
     ∇  1987-03-12 17.44.33 (GMT-4).
```

However, this function is unacceptably slow for values of $N > 5000$, taking more than a minute of computer running time.

An average of 5000 faults per chip may seem high, and such numbers have not been needed previously for yield calculations. However, the presence of more than 100 000 faults per chip may occur in future chips. Consider for example a 1% fault level for a 16-Mb DRAM chip. This would correspond to 160 000 faults on such a chip. Fault-tolerance schemes that can deal with more than 100 000 faults on a chip are now being evaluated. The *APL* programs used to estimate the yield of such chips must therefore be capable of processing numbers of this magnitude.

Expression (1) can be rearranged in the form

$$P(X = k) = \frac{1}{(1 + \lambda/\alpha)^{\alpha}} \prod_{j=1}^{k} \frac{(\alpha - j)\lambda/\alpha}{j(1 + \lambda/\alpha)}. \quad (3)$$

This re-arrangement can be programmed with the scan function, resulting in

```
     ∇
[0]    X←N NBINOM LA;ALPHA;LOA;Z
[1]    ⍝THIS FUNCTION CALCULATES THE
       NEGATIVE BINOMIAL
[2]    ⍝OR POLYA-EGGENBERGER
       DISTRIBUTION WITH
[3]    ⍝PARAMETERS LAMBDA=LA[1],
       ALPHA=LA[2]
[4]    X←(×\1,LOA×((ALPHA+Z-1)
       ÷(Z←⍳N)×1+LOA))÷(1+LOA←+/LA)
       *ALPHA←LA[2]
     ∇  1988-09-01 14.13.34 (GMT-4).
```

The output of this program is a vector of length `N+1` containing the probabilities associated with finding 0, 1, 2, 3, etc. faults per chip. If the probability associated with a single number is needed, the scan operator " \ " is simply replaced by the reduce operator " / ".

The program is extremely fast when compared to the method of Equation (2). For example, an average time of 422 milliseconds was needed for `N←150000` on an IBM 3090 Model 600S computer with a vector facility. It was still relatively fast on an IBM 3090 Model 400 without a vector facility, where it was found to require slightly more than twice that amount of time. The maximum number of faults that can be handled by this program depends on the amount of computer memory available in the *APL* workspace. Until now the only error messages have therefore been *WS FULL*.

Cluster-map generator

The matrix capability of *APL* combined with complex variables provides a very effective approach for simulating clusters that mimic the fault clusters frequently observed on integrated-circuit wafers. The program described here has two major steps. In the first, symmetrical clusters are generated on a predetermined grid. In the second, the clusters are shaped to conform to observed patterns.

- *Symmetrical cluster generator*

The program starts by calculating coordinates for an $N \times N$ grid. This is done with

```
Z←(1J1×¯1-÷N)+(2÷N)×(0J1×⍳N)∘.+⍳N.
```

The coordinates are stored as complex numbers in a matrix. The values of these complex numbers are constrained between ±1 in both the real and the imaginary directions. For `N←5` the matrix `Z` has the form

```
¯0.8J 0.8  ¯0.4J 0.8  0J 0.8  0.4J 0.8  0.8J 0.8
¯0.8J 0.4  ¯0.4J 0.4  0J 0.4  0.4J 0.4  0.8J 0.4
¯0.8       ¯0.4       0       0.4       0.8
¯0.8J¯0.4  ¯0.4J¯0.4  0J¯0.4  0.4J¯0.4  0.8J¯0.4
¯0.8J¯0.8  ¯0.4J¯0.8  0J 0.8  0.4J¯0.8  0.8J¯0.8.
```

An example of this grid is shown in **Figure 1**. Note that the coordinates occur in the centers of square areas. The occurrence of a single fault or the absence of a fault is generated for each area. The areas must therefore be small enough to make this a valid assumption. In the use of the program by this author, values of $N \leftarrow 256$ are typically used.

The occurrence or absence of a fault is determined by assigning a probability to each grid point. This is done with a symmetrical bivariate Gaussian distribution obtained with

```
P←C××-((10○Z)÷SIGMA×2*0.5)*2,
```

where C is a constant and $SIGMA$ is the standard deviation of the distribution. Both of these quantities are single scalars. The value of C sets the density of the simulated faults, while the value of $SIGMA$ sets the radius of the cluster. The result is an $N \times N$ array of numbers.

The matrix of probabilities is compared to an $N \times N$ array of random numbers with values between 0 and 1. A fault is simulated when the value of the random number is less than the value of the probability. The locations of the faults are then collected. This is accomplished with the statement

```
L←(L≠0)/L←,Z×((?(N,N)ρ1E16)÷1E16)≤P,
```

where L contains the fault locations.

The original coordinates in the matrix Z represent grid points in the center of square areas with sides equal to $2 \div N$. The faults in the vector L are all located at these grid points. To give them a more random character, they are therefore randomly perturbed within these areas, by means of

```
L←L+1J0×(¯1+2×(?(ρL)ρ1000000)÷1000000)÷N,
L←L+0J1×(¯1+2×(?(ρL)ρ1000000)÷1000000)÷N.
```

The real and imaginary perturbations are performed independently here because the roll function " ? " does not operate on complex numbers.

An example of a symmetrical cluster which was generated with this program is shown in **Figure 2**. The parameters used were $N \leftarrow 256$, $C \leftarrow 0.02$, and $SIGMA \leftarrow 0.2$. These programs have not been optimized for speed. Nevertheless, it was observed that they also ran more than twice as fast on the computers with vector processing capabilities.

• *Cluster-shaping programs*

The actual fault clusters observed on integrated-circuit wafers seldom have symmetrical shapes. The results from the preceding section must therefore be altered by means of shaping programs. Two different programs have been used by this author. One stretches the clusters and rotates

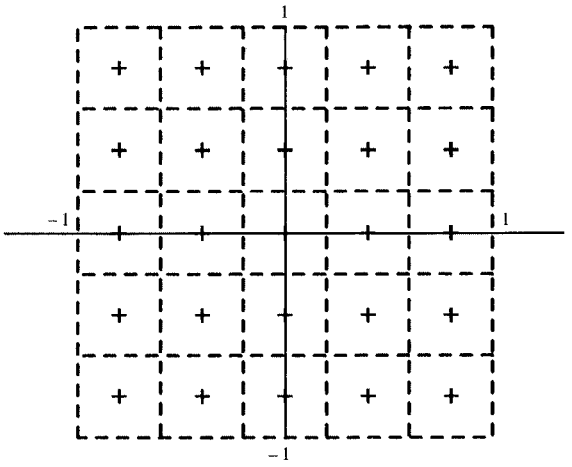

Figure 1

Coordinate grid used for cluster simulation. The coordinates are points in the complex plane between ±1 in the real and imaginary directions. The occurrence or absence of a fault is simulated for each grid point.

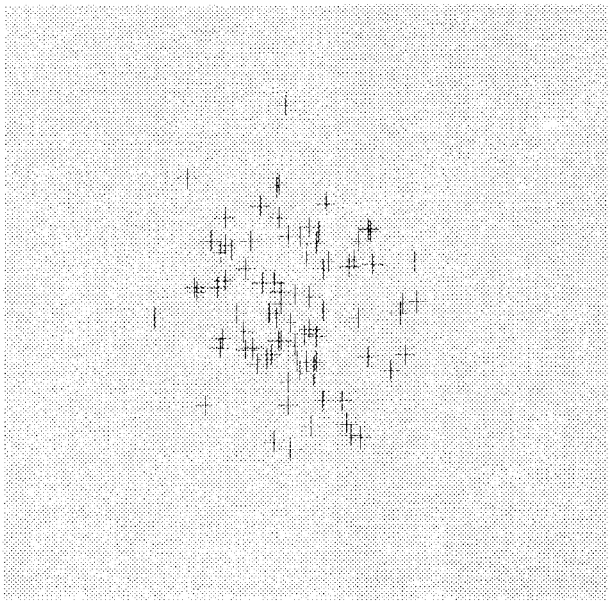

Figure 2

Example of a cluster generated with a symmetrical-cluster-generating program. The + symbols in this map represent fault locations simulated with a symmetrical-cluster generator.

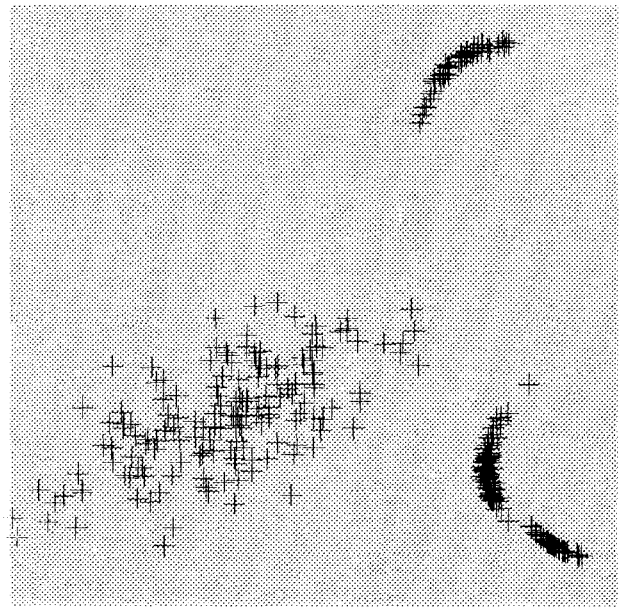

Figure 3

Examples of clusters generated by applying shaping programs to symmetrical clusters. One of the clusters is simply stretched and rotated; the others are formed into patterns that mimic those often resulting from scratches.

them. The second generates clusters that mimic faults caused by scratches. During simulation runs these programs are selected pseudorandomly.

In the vector L created above, the fault locations are stored as complex numbers. This feature simplifies the stretching operation by making use of projections with randomly generated angles in the following set of steps:

```
⍝STRETCHING THE CLUSTER ALONG THE X
 DIRECTION
 L←(0J1×11○L)+(9○L)÷(?1000000)
 ÷1000000
⍝STRETCHING THE CLUSTER ALONG THE Y
 DIRECTION
 L←(1J0×9○L)+0J1×(11○L)÷(?1000000)
 ÷1000000.
```

The scratch generation is achieved as follows:

```
L←(1J0×9○L)+0J0.25×(11○L)
  ×(⌈/9○L)-⌊/9○L
L←0J¯1+(1J0×9○L)+0J1×(11○L)
  +(1-2×(9○L)*2).
```

All of the generated clusters are subsequently rotated with a random angle in the statement

```
L←L×*0J1×((?1000000)÷1000000)×○2.
```

The simplicity of this operation is the result of using complex numbers.

The clusters obtained with this program are all located in the center of a map. The next step therefore consists in randomly displacing them and making sure that the results fit within a ±1 frame. This is accomplished by means of

```
⍝COORDINATE OFFSET IN THE X DIRECTION
 L←L+¯1J0+2J0×(?1000000)÷1000000
⍝COORDINATE OFFSET IN THE Y DIRECTION
 L←L+0J¯1+0J2×(?1000000)÷1000000
⍝CROPPING THE FIELD OF VIEW
 L←(~((9○L)<¯1)∨((9○L)>1)∨((11○L)<¯1)
 ∨(11○L)>1)/L.
```

Four clusters generated with this program are shown in **Figure 3**. One of these is a stretched and rotated cluster; the other three represent scratches. Two of the scratches in the lower right corner line up with each other to appear like a single cluster. These results are intended to be only an example of the output of the simulator, because scratches are usually observed less frequently in practice. The pseudorandom selection process used by this author transforms less than 10% of the symmetrical clusters into scratches.

This cluster simulator has been used to evaluate fault-tolerance schemes for memory chips. The simulations tend to produce a relatively large number of faults per chip. The results resemble the faults often observed during the early phases of chip development and manufacture. Such faults usually contribute to the tails of the faults per chip distribution, and occasionally clusters of this type have produced second modes in these distributions. In the occurrence of such bimodal distributions, the other mode can usually be modeled with negative binomial distributions. A cluster-map generator that generates negative binomial clusters is therefore also needed. A program that can be used for this purpose is described next.

Negative binomial cluster generator

Negative binomial fault distributions can be simulated with a technique that randomly generates faults as a function of time. It does so when, during an incremental time interval Δt, the probability of obtaining a fault on a chip is linearly related to the number of faults already on that chip [5]. A Poisson distribution results when this probability is constant. A modification of this technique is used here.

In the simulation program, a random number generator is used to determine whether a fault is to be added to a chip during a time interval Δt. The probability of adding a fault to a given chip is assumed to be linearly dependent not only on the number of faults present on

that chip, but also on the number of faults on its four nearest neighbors.

At the start of the program, no faults are present in a simulated array of $M \times N$ chips. It slowly builds up a pattern of faults by stepping through consecutive time increments. The process stops when the desired average number of faults is reached. The following program has been used to achieve this:

```
     ∇
[0]    X←NM NBCG L;B;C;M;N
[1]    N←NM[1]
[2]    M←NM[2]
[3]    B←0.001×⍟1+L
[4]    C←0.3×B
[5]    BI←1
[6]    X←NM⍴0
[7]   LOOP:X←X+(1E 9×C+B×X+BI×X
       [;(1+⍳M-1),1]+X[;M,⍳M-1]+
       X[(1+⍳N-1),1;]+X[N,⍳N-1;])
       ≥?NM⍴1E 9
[8]    →(((0.0625×L××/NM)-⌊.05×N)
       >+/,X)/LOOP
     ∇ 1988-06-27 14.46.12 (GMT-4).
```

The variable MN specifies the dimensions of the chip array for which the faults are to be generated, L sets the average number of faults per chip to be simulated, while B, BI, and C are constants that determine the nature of the clustering. The use of a loop in this program is unfortunate, because of the inherent inefficiency of such an approach in *APL*. The loop results from having to use the simulated numbers sequentially.

This program produces an array of numbers which is adequate for most yield estimations. It has been used extensively in this form to evaluate fault-tolerance schemes for the IBM 4-Mb memory chip [8]. However, in order to display the results, it is often useful to generate fault-location maps. This can be accomplished with the following program:

```
     ∇
[0]    XY←XYG M
[1]    I←J←K←1
[2]    XY←1 2⍴0
[3]   LOOP:→(M[I;J]=0)/INCR
[4]    XY←XY,[1]((M[I;J],2)⍴J,I)
       -(M[I;J],2)⍴.001×?(2×M[I;J])
       ⍴1000
[5]   INCR:→((⍴M[;1])≥I←I+1)/LOOP
[6]    I←1
[7]    →((⍴M[1;])≥J←J+1)/LOOP
[8]    XY←1 0↓XY
     ∇ 1988-07-29 14.29.52 (GMT-4).
```

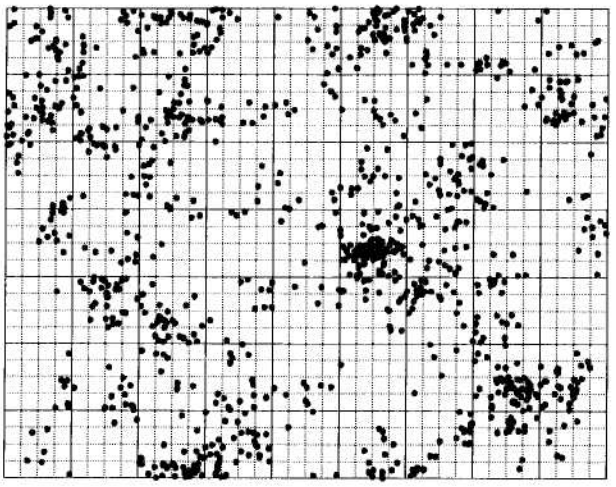

Figure 4

A map generated with a negative binomial cluster generator (from [8], reproduced with permission.

The resulting coordinates are real numbers, and do not correspond to those of the preceding section.

A typical map of simulated fault clusters generated with these programs is shown in **Figure 4**. The parameters used were L←16, BI←1, and MN←36 28. The grid of superimposed squares was used for an analysis of partially good chips with redundancy, as described in [8]. The solid grid lines demarcate squares that represent chips; sections of chips are bounded by dotted and solid lines. The distribution of the number of faults per chip and the distribution of the number of faults per section result in negative binomial distributions.

Conclusion

The programs described here, and variants of them, have been used by this author to evaluate memory-chip architectures under consideration at the IBM facility in Essex Junction, Vermont. Fault-tolerant chips that can operate properly when they contain a large number of faults are currently being manufactured (see, e.g., [8]).

Another use of the programs has been in the development of sampling techniques for in-line manufacturing inspection with defect-mapping tools. Tools such as the Insystem holographic inspection system and the KLA 2000-series wafer-inspection systems have confirmed the presence of defect clustering at various steps of the manufacturing process. This clustering seriously affects the sampling statistics used in the inspection procedure. Existing sampling procedures are often of limited use, and can lead to incorrect

conclusions. The cluster-generating programs described here are used in the study of such limitations and means to overcome them.

References

1. C. H. Stapper, "Yield Model for 256K RAMs and Beyond," *1982 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, February 1982, pp. 12–13.
2. C. H. Stapper, "Fault Simulation for Fault-Tolerant Multi-Mbit RAMs," *Proceedings of the IEEE 1989 International Conference on Microelectronic Test Structures* **2,** 93–96 (March 1989).
3. 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. Res. Develop.* **24,** 398–409 (May 1980).
4. C. H. Stapper, "Modeling Redundancy in 64K to 16Mbit DRAMs," *1983 IEEE International Solid-State Circuits Conference, Digest of Technical Papers*, February 1983, pp. 86–87.
5. C. H. Stapper, F. M. Armstrong, and K. Saji, "Integrated Circuit Yield Statistics," *Proc. IEEE* **71,** 453–470 (April 1983).
6. C. H. Stapper, "Large-Area Fault Clusters and Fault Tolerance in VLSI Circuits: A Review," *IBM J. Res. Develop.* **33,** 162–173 (March 1989).
7. I. Koren and C. H. Stapper, "Defect and Fault Tolerance in VLSI Circuits: A Review," *Defect and Fault Tolerance in VLSI Systems*, I. Koren, Ed., Plenum Publishing Co., New York, 1989, pp. 1–21.
8. C. H. Stapper, "Block Alignment: A Method for Increasing the Yield of Memory Chips That Are Partially Good," *Defect and Fault Tolerance in VLSI Systems*, I. Koren, Ed., Plenum Publishing Co., New York, 1989, pp. 243–255.
9. I. Koren and M. A. Breuer, "On Area and Yield Considerations for Fault-Tolerant VLSI Processor Arrays," *IEEE Trans. Computers* **C-33,** 21–27 (January 1986).
10. I. Koren and D. J. Pradhan, "Introducing Redundancy into VLSI Designs for Yield and Performance Enhancement," *Proceedings of the Fifteenth Annual International Symposium on Fault-Tolerant Computing, FTCS-15*, 1985, pp. 330–334.
11. I. Koren and D. J. Pradhan, "Yield and Performance Enhancement Through Redundancy in VLSI and WSI Multiprocessor Systems," *Proc. IEEE* **74,** 699–711 (May 1986).
12. D. M. Stewart, "Laser Fix Dynamic RAMs," *Electron. Week* **58,** 45–49 (February 4, 1985).
13. D. M. H. Walker, "Yield Simulation for Integrated Circuits," Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA, July 1986, Ch. 4, pp. 44–47, Ch. 8, pp. 138–140.
14. D. M. H. Walker, *Yield Simulation for Integrated Circuits*, Kluwer Academic Publishers, Boston, 1987, Ch. 4, pp. 45–49, Ch. 8, pp. 158–160.
15. J. C. Harden and N. R. Strader II, "Architectural Yield Optimization for WSI," *IEEE Trans. Computers* **37,** 88–110 (1988).
16. C. L. Wey, "On Yield Consideration for the Design of Redundant Programmable Logic Arrays," *IEEE Trans. Computer-Aided Design* **7,** 528–535 (April 1988).
17. V. Foard Flack, "Estimating Variations in IC Yield Estimates," *IEEE J. Solid-State Circuits* **SC-21,** 362–365 (April 1986).
18. C. L. Winter and W. L. Cook, "Interval Estimates for Yield Models," *IEEE J. Solid-State Circuits* **SC-21,** 590–591 (August 1986).

Received March 15, 1989; accepted for publication July 18, 1989

Charles H. Stapper *IBM General Technology Division, Burlington facility, Essex Junction, Vermont 05452.* Dr. Stapper received his B.S. and M.S. in electrical engineering from the Massachusetts Institute of Technology in 1959 and 1960. He subsequently joined IBM at the Poughkeepsie development laboratory, where he worked on magnetic recording and the application of tunnel diodes, magnetic thin films, electron beams, and lasers to digital memories. From 1965 to 1967, he studied at the University of Minnesota on an IBM fellowship. Upon receiving his Ph.D. in 1967, he joined the IBM development laboratory in Essex Junction. His initial work there was in the areas of magnetic thin-film array development, testing and theory of magnetic bubble devices, and bipolar and field-effect transistor theory. During the early 1970s he developed a yield model for the analysis of defect monitor data. This model has been used since for line control and yield management. It has also been used extensively for productivity optimization of SRAMs and DRAMs with redundancy, as well as for planning the production of gate arrays, logic chips, and microprocessor chips. Dr. Stapper is a member of the Institute of Electrical and Electronics Engineers and Sigma Xi.