

ROBUST MASK-LAYOUT AND PROCESS SYNTHESIS THROUGH AN EVOLUTIONARY ALGORITHM

Lin Ma

Engineering Design Research Laboratory
California Institute of Technology
Pasadena, California 91125
ma@design.caltech.edu

Erik K. Antonsson, Ph.D., P.E.*

Engineering Design Research Laboratory
California Institute of Technology
1200 East California Blvd.
Pasadena, California 91125-4400
erik@design.caltech.edu

ABSTRACT

A method for automated mask-layout and process synthesis for MEMS is presented. The synthesis problem is approached by use of a genetic algorithm. For a given desired device shape, and several fabrication process choices, this synthesis method will produce one or more mask-layouts and associated fabrication process sequences (which when used can generate shapes close to the desired one). Given complicated device shapes and wide range of fabrication process possibilities, the designer may encounter difficulty producing the right mask-layout and fabrication procedure by experience and trial and error. An automated synthesis tool like this will be helpful to the designer by increasing the efficiency and accuracy of the design of MEMS devices.

1 INTRODUCTION

While the past decade has seen many remarkable advances in microelectromechanical systems (MEMS) design and fabrication, as the complexity of MEMS devices and the scope of MEMS applications increases, the need for structured design methods also increases (Antonsson, 1995). Many of the most interesting and useful mechanical devices intrinsically rely on 3-dimensional behavior and 3-dimensional shapes. Complex 3-dimensional shapes can be generated using fabrication procedures such as multiple process wet etching (Fruhauf and Hanemann, 1997; Kang, Haskard, and Samaan, 1997), but the mapping from the mask-layout to the final shape is usually non-intuitive and complicated. For a given 3-D device shape, it is difficult for the designer to produce a proper mask-layout as well as correct fabrication procedures by experience and trial and er-

ror. An automatic design tool which can automate the shape-to-mask-and-fabrication-process synthesis will be helpful to the development of new MEMS devices, since with its help the designer can be relieved from considering the details of fabrication and instead can focus on the functional design of the devices. Previously reported work (Li and Antonsson, 1999) utilized a genetic algorithm to synthesize mask-layouts for a single process. Here an approach to realize mask-layout and fabrication process synthesis through a genetic algorithm is introduced.

For the problem of mask and process synthesis, a genetic algorithm iteration loop is constructed to evolve the optimal mask and process sequence as shown in Figure 1. An initial population of solution candidates (mask-layouts and process parameters) are randomly generated. The fabrication of each mask-layout using the associated process parameters is then simulated by a specified fabrication simulator to produce a 3-dimensional shape. The performance of each solution candidate is measured through the shape comparison between the produced shape and the user defined final shape. During each evolutionary loop, genetic operations are applied to control the stochastic searching behavior such that the best performing solution candidates are more likely to survive and evolve even better offspring for the next generation. The iteration is stopped when one satisfying solution is found.

2 THE GENETIC ALGORITHM

Genetic algorithms (GAs) are a global stochastic optimization technique which is based on the adaptive mechanics of natural selection evolution. It was invented by John Holland (Holland, 1975) in 1975, and subsequently has been made widely popular by David Goldberg (Goldberg, 1989). GAs use two basic

*corresponding author

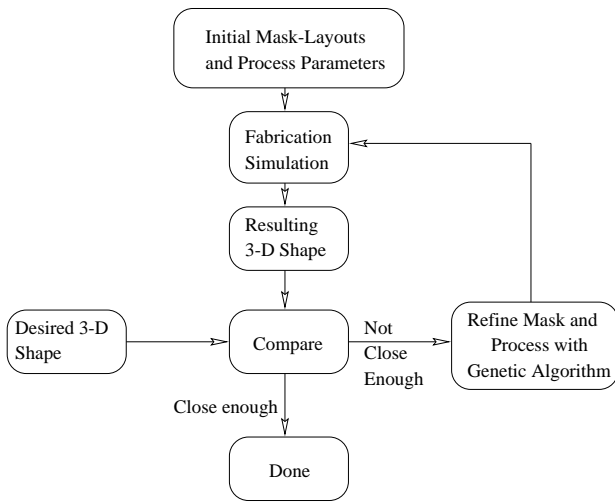


Figure 1. A schematic representation of an genetic algorithm MEMS synthesis technique.

processes from evolution: *inheritance*, or the passing of features from one generation to the next, and competition, or *survival of the fittest*, which results in weeding out individuals with bad features from the population. The algorithm maintains a population of solution candidates. Each individual in the population is encoded into a string of characters or digits, through which the original solution space is converted to an encoded space, which is easier for the algorithm to search for the global optimum. The genetic algorithm works as an iteration loop. First, an initial population is generated randomly. Then all the individuals in the initial population are evaluated by an objective function measurement program (which is problem specific), and a performance value (called a fitness value or FV) will be calculated for each individual. Then the whole population goes through genetic operations such as selection, crossover, *etc.*, to form individuals of the next generation. Because of the strong converging characteristics of GA, the new individuals will generally have better performance than the ones in last generation. Such iteration continues until an individual is found whose performance is good enough, and this individual will be taken as the solution.

In the problem addressed here, the solution space comprises a mask-layout space and a fabrication process space. Mask-layouts are treated geometrically as 2-dimensional polygons. In the initial test stage, the process space is relatively simple. For example, for a multiple process wet etching application, the process space will be all the fabrication procedures that can be used, and each individual solution candidate includes parameters such as the etchant number(s) to be used, the etch time duration value for each applied etchant, *etc.* The process space is application specific, and is simpler than the mask-layout space. The mask-layout space is more complex and therefore requires more at-

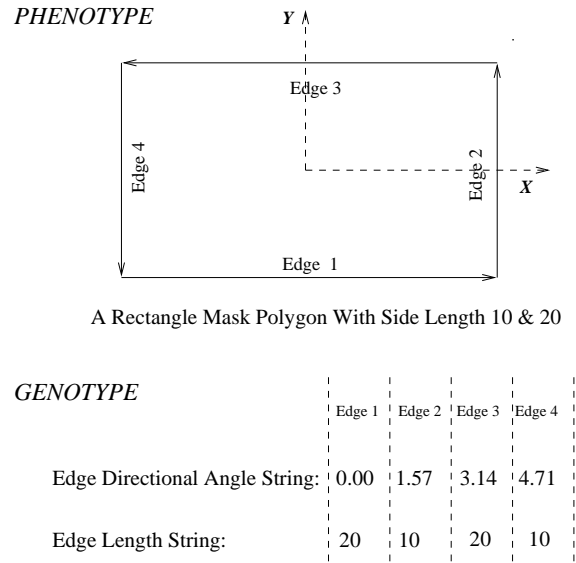


Figure 2. A schematic illustration of the GA coding scheme.

tention during encoding and manipulation, which are described below.

2.1 Coding Scheme

An appropriate coding scheme is needed to encode a 2-D polygon to a string which can be easily manipulated by genetic operations. There are many ways to describe a 2-D polygon. Here the edge length and edge directional angle (turning angle) are chosen to describe each edge of a polygon (Arkin, Chew, Huttenlocher, Kedem, and Mitchell, 1991; Li and Antonsson, 1999). A mask polygon is encoded into two real strings. One string contains edge directional angles and the other contains edge lengths. The size of each string is equal to the number of polygon sides. Two elements, one from each string with the same element position, describe an edge of the polygon. In GAs, the physical objects in the solution space which are encoded are called phenotypes, and the encoded strings are called genotypes. The searching and manipulation are conducted on genotypes. The two real strings are genotypes for the mask-layouts in the GA. A schematic illustration of the coding scheme is shown in Figure 2.

There are generally two kinds of coding schemes, binary coding, and real coding. A real coding scheme is used because it provides adequate precision with a short string length. The edge length and turning angle representation of 2-D polygon provides a convenient way to design the associated crossover and mutation schemes.

2.2 Selection Scheme

The core of a selection scheme is the method to assign a sampling rate to each individual in the generation. The sampling rates decide the possibilities for each individual to be selected for crossover, and the individual with larger sampling rate has more chance to survive and evolve. In the application addressed here, a rank-based selection algorithm is chosen to assign the sampling rates based on an individual's rank among the whole population (Baker, 1985) through the following formula:

For an individual whose rank is i ,

$$\text{Sampling rate} = 1.0 + \text{bias} - 2 * \text{bias} * i / (\text{size} - 1)$$

- i = individual's rank, from 0 to size-1
- size = population size
- bias = extra fitness awarded to the top-ranked individual compared to the average-ranked individual whose fitness is 1.0

The bias value is in general between 0.0 and 1.0, and it can be used to control the selection pressure. In this application, the bias value is gradually increased as the genetic iteration proceeds to control the rate of convergence. During the initial stage the selection pressure is light so that premature convergence can be prevented, and during the final stage the selection pressure is heavy so that the population will effectively converge to the final result.

2.3 Crossover and Mutation

Crossover is the main operator used for reproduction. It combines portions of two parents to create two new individuals, called *offspring*, which inherit a combination of the features of the parents. There are several popular crossover schemes. In *one-* and *two-point crossover*, one or two gene positions are randomly selected, and the two parents are crossed at those points. An example of one-point crossover is shown in Figure 3. The offspring is identical to the first parent up to the crossing point and identical to the second parent after the crossing point. In *uniform crossover*, each gene position is crossed with some probability, typically one-half. Crossover combines the building blocks, or schemata, from different solutions in various combinations. Smaller good building blocks are converted into progressively larger good building blocks over time until an entire good solution is found. Crossover is a random process, and the same process results in the combination of bad building blocks to result in bad offspring, but these are eliminated by the selection operator in the next generation.

Mutation is an incremental change made to each member of the population, with a very small probability. Mutation enables new features to be introduced into a population so that premature convergence can be avoided.

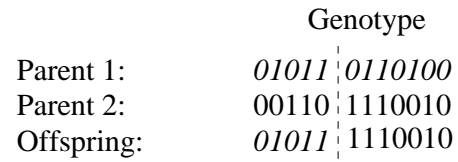


Figure 3. One-point crossover of binary strings

2.4 Crossover of Polygons

The crossover method used here has two important aspects. First, as mentioned above, real coding is used for the mask-layout and fabrication process synthesis problem. Two real strings are used to represent a mask-layout polygon. The crossover scheme used is a real crossover called BLX- α (Eshelman and Schaffer, 1991), which produces an offspring by uniformly picking a value for each gene from the range formed by the values of two corresponding parent genes with some extension on each side. If the distance between the two parent genes is I , the length of the extension on each side is αI , and α is a user specified parameter. Second, when the real strings are crossovered, the real strings are representations of 2-D simple polygons. If every pair of genes of the parents is crossed using BLX- α , the offspring are not guaranteed to represent a simple polygon because the geometric constraint may not be satisfied. In order to make sure the offspring from the crossover is simple (closed) polygon, some care must be taken to satisfy the geometric constraint.

The scheme used here is shown in Figure 4. The crossover is done for angle strings and distance strings separately. In Figure 4, two parent polygons have 6 sides, and angle strings and distance strings with gene length of 6 are paired up. For each string pair, the worst matched pair of genes is chosen first. For the angle string pair, the worst matched pair of genes is gene 6, and for the distance string pair, gene 3. Now every gene pair is crossovered except gene 6 on the angle string and gene 3 on the distance string, and a child is generated, with gene 6 on angle string, a_6 , and gene 3 on distance string, r_3 , undetermined. These two undetermined values come from the geometric constraint that the two strings should represent a closed, 2-D simple polygon. After the reconstruction, a_6 and r_3 are calculated.

3 Variable Gene Length GA

The Genetic Algorithm described above has a limitation. The crossover scheme presented here only works on two polygons with the same number of sides. Users need to guess the appropriate number of sides for the mask-layout synthesis in advance, and therefore only a subspace of the whole polygon space, polygons with this pre-specified number of sides, are explored. The optimal mask-layout may not be found just because it lays outside of this subspace. A variable gene length GA was developed so that polygons with different numbers of sides are ex-

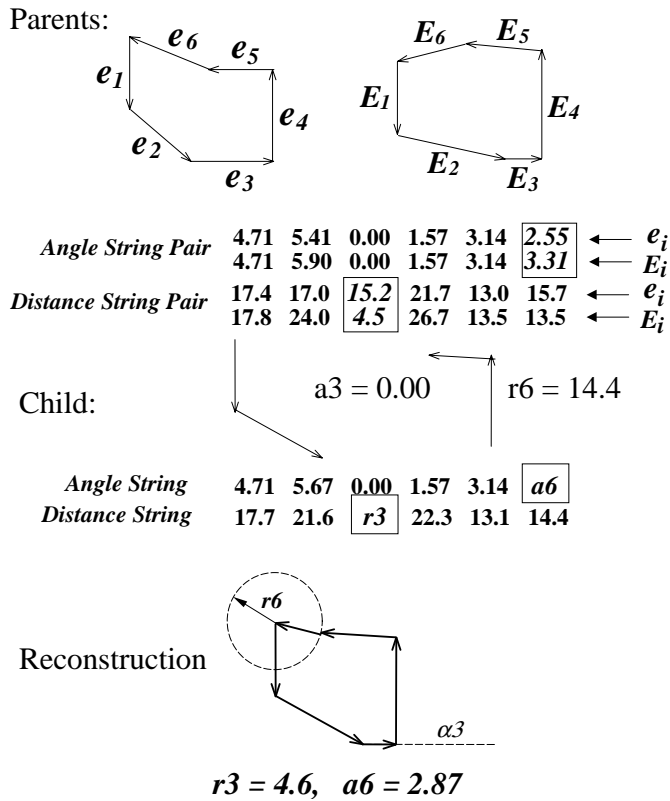


Figure 4. Crossover of Two Polygons

explored by the algorithm and the chance to find the optimum is greatly increased.

The most difficult part developing a GA with variable gene length is crossover: how to crossover two individuals with different gene lengths. In the problem addressed here, the entities that need to be crossovered are 2-D polygons with different numbers of sides, and the only crossover scheme previously available works on polygons with the same number of sides. An add-then-remove scheme is applied to extend the fixed gene length crossover to work on variable gene length cases.

Figure 5 shows the add-then-remove scheme introduced here. Two polygons to be crossovered, are shown at the top of the figure: one with 4 sides, the other with 6 sides. The basic idea of the add-then-remove scheme is to add two vertices into the polygon with 4 sides so that this polygon can be viewed as a 6-sided polygon and then crossover on these two 6-sided polygons can be conducted. After crossover, some vertices of the child polygon will be removed (usually the most collinear ones) so that the final child polygon will have side number between the side numbers of the parent polygons.

First, a scheme to determine where to put the new vertices on the polygon with less sides is needed. The positions of new vertices should be chosen in such a way that after insertion of new

vertices and crossover, the offspring polygon combines common features of both parents. In other words, the insertion of new vertices should not break the already established common feature (edge) correspondence, and new edge correspondence should also be established by the insertion. To determine the insertion positions, two parent polygons are scaled to same perimeter length (1.0 in the example). Using a line segment from 0.0 to 1.0 to represent a polygon, each vertex of the polygon corresponds to a point on the line segment. For the square, the vertices are 0.25, 0.5, 0.75, and 1.0, and for the other polygon, the vertices are 0.2, 0.35, 0.5, 0.7, 0.85, and 1.0.

In order to determine where to put the new vertices on the polygon with fewer sides (the square), it must be established which pair of vertices on both polygons correspond to each other. For each vertex on the polygon with fewer sides, the vertex closest to it on the line segment of the other polygon (the polygon with more sides) is chosen to be the corresponding vertex. In the example shown in Figure 5, for vertex 0.25 of the square, the vertex 0.2 of the other polygon is the corresponding vertex. Vertex 0.5 corresponds to vertex 0.5; vertex 0.75 to vertex 0.7, and vertex 1.0 to vertex 1.0. The two segments of the two polygons with corresponding vertices as end points are called corresponding segments. For example, segment 0.25 to 0.5 of the first polygon and segment 0.2 to 0.5 of the second polygon are corresponding segments. Now two vertices are left uncorresponded on the polygon with more sides, vertex 0.35, and vertex 0.85. Vertex 0.35 is on segment 0.2 to 0.5 which has corresponding segment 0.25 to 0.5 on the other polygon (the square), and a new vertex is generated on this corresponding segment of the square with its position relative to this corresponding segment exactly the same as vertex 0.35 relative to its corresponding segment on the other polygon. The corresponding vertex of vertex 0.85 is generated the same way. The “×” symbols in Figure 5 show where the new vertices are inserted.

After the insertion, the two polygons (with the same number of sides) are crossovered to generate the initial child polygon, and then a number between the side numbers of the two parents is randomly generated as the side number of the final child polygon, and some vertices of the initial child polygon (usually the most collinear vertices) are then removed to produce the final polygon with the specified number of sides. In the example, the parents have side numbers of 4 and 6, the initial child polygon has 6 sides. The randomly generated final polygon side number is 5, and one vertex of the child polygon (the one in the circle) is removed to get the final polygon with 5 sides.

4 APPLICATION

A specific application of this GA method on multiple process wet etching synthesis has been conducted. The bulk wet etching mask-layout and process synthesis was chosen as an example because of its high level of geometric complexity. For bulk

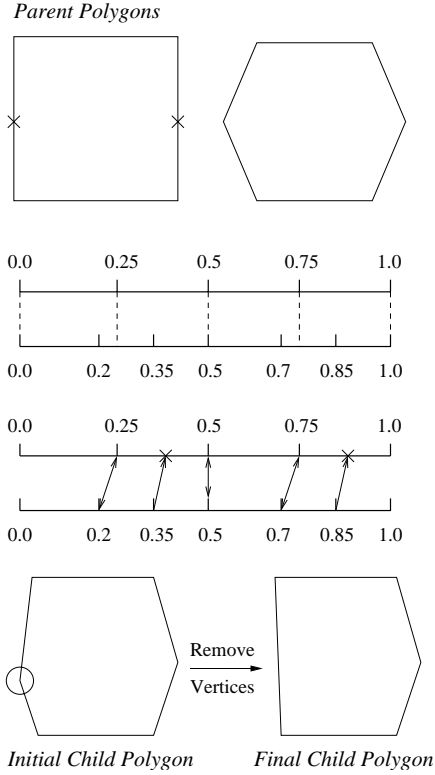


Figure 5. Add-then-remove scheme for polygons with different numbers of sides

wet etching, there is no simple geometrical correlation (in general) between the masks and process sequences and the evolved 3-D structures. A multiple process wet etching simulator called SEGS (Hubbard and Antonsson, 1996) was used as the fabrication simulation and performance evaluation program.

The synthesis problem is stated as follows. A desired device shape is given, and three different etchants can be chosen. The fabrication procedures allowed here are two wet etching steps, which means two etchants can be applied sequentially, each for a predetermined time duration, to generate the final shape. The goal is to find a mask-layout, and the order and time duration of the two etchants to use to produce a desired 3-D shape. A multi-process wet etching procedure can generate highly geometrically complicated shapes and will require automated design tools. The two-process example shown here illustrates the feasibility of this approach.

4.1 Shape Comparison

A way to calculate the closeness between the evolved 3-D shape and the specified target shape is needed. The shapes are represented by a stack of polygonal layers. The etching simulation program ensures that the polygon layers of both shapes ver-

tically match. The mismatch between two 3-D shapes (contours of the target shape and a candidate evolved shape at the same vertical position) is decomposed into pure shape mismatch and size mismatch. For these two polygons, the size mismatch is calculated as the length ratio of the two polygons. The pure shape mismatch between two polygons is obtained using the method introduced by Arkin *et al.* (Arkin *et al.*, 1991). The pure shape mismatch of the two shapes is calculated as the *weighted* sum of the mismatch between the two polygon layers in each vertical level. The weights need to be carefully chosen such that the important shape information describing the overall vertical characteristics of the shape which can not be found from the shape mismatch of two polygon layers is conserved. The following formulas are used to calculate the pure shape mismatch and size mismatch between the two 3-D shapes:

$$\text{SizeMismatch} = \sum_{i=1}^n \frac{\text{SiM}(i)}{n}$$

$$\text{PureShapeMismatch} = \sum_{i=1}^n \frac{\left(\text{ShM}(i) * \frac{\text{SiM}(i)}{\text{ASiM}} \right)}{n}$$

where n is the total number of layers, and ShM and SiM are pure shape mismatch value and size mismatch value of two polygon layers in the same level, and ASiM is average size mismatch value of all layers.

Using shape mismatch and size mismatch to represent the closeness between two 3-D shapes has a disadvantage: the difference between the side wall slopes of the two shapes is not taken into consideration. For some devices with small z -direction dimension compared with xy -plane dimensions, two shapes with similar xy -plane cross section, but different side wall slopes, may be considered very close using the shape and size mismatches criteria described above, but the difference in the side wall angles of these two shapes may make one device an unacceptable replacement for the other.

Due to the preceding consideration, a scheme to represent side wall slope using contours of a shape is constructed. For two contours with a vertical distance of h ,

$$\text{SideWallSlope} = \text{ArgTan} \left(\frac{h * 2\pi}{L_{\text{upper}} - L_{\text{lower}}} \right)$$

where L_{upper} and L_{lower} are perimeter lengths of the upper layer and lower layer contours. For two shapes, the slope mismatch at a particular depth is the difference between the side wall slopes at this depth calculated as shown above. Then the average of the slope mismatches at all depths is considered as the overall slope mismatch between these two shapes.

4.2 Fitness Evaluation

The fitness value of each individual is evaluated in the following steps:

1. The 3-D etching simulation (SEGS (Hubbard and Antons-son, 1996)) is run on each candidate solution. Each candidate solution consists of a mask polygon and process parameters including two etchant numbers and two etch time duration values. The simulation is run using the mask, two etchants for those two etch time durations, and as a result an evolved 3-D shape represented by a stack of polygonal layers is produced.
2. The pure shape mismatch value, size mismatch value, and slope mismatch value between each of the evolved 3-D shapes and the specified target shape are calculated. These values are stored as the objective function values of each individual in the generation.
3. After obtaining the objective function values of all the individuals in the generation, the fitness values are calculated from the objective function values. This is a typical multiple objective optimization problem, in which an overall performance value is constructed from three different objectives. Several different ways to construct the fitness value were tried, and the non-compensating maximum-of-all scheme seems to work best:

$$\text{Fitness}(i) = \text{Max} \left(\frac{\text{AShM}}{\text{ShM}(i)}, \frac{\text{ASiM}}{\text{SiM}(i)}, \frac{\text{ASIM}}{\text{SIM}(i)} \right)$$

where ShM, SiM and SIM are shape, size, and slope mismatch values of individual i , and AShM, ASiM, and ASIM are the respective average values of the generation.

4.3 Result

The synthesis task was conducted on a Sun Ultra10 workstation, and took 38 minutes. Some parameters used in the iterative synthesis are listed below:

- population size: 60;
- number of edges in the mask-layouts: 16;
- maximum number of iterations: 60.

The results of synthesis using the approach described here are shown in Figure 6 and Table 1. In Figure 6, the lower-right frame shows the target shape, and the other frames show the best candidate mask-layouts (the black polygons) at five different iterations during the synthesis loop, and the simulated 3-D shapes. The convergence of the evolved shapes to the target shape can be easily observed. Detailed information about these five iterations is shown in Table 1. The shape, size and slope mismatch values between the simulated shape and the target shape for each iteration are getting smaller as the synthesis proceeds, and the

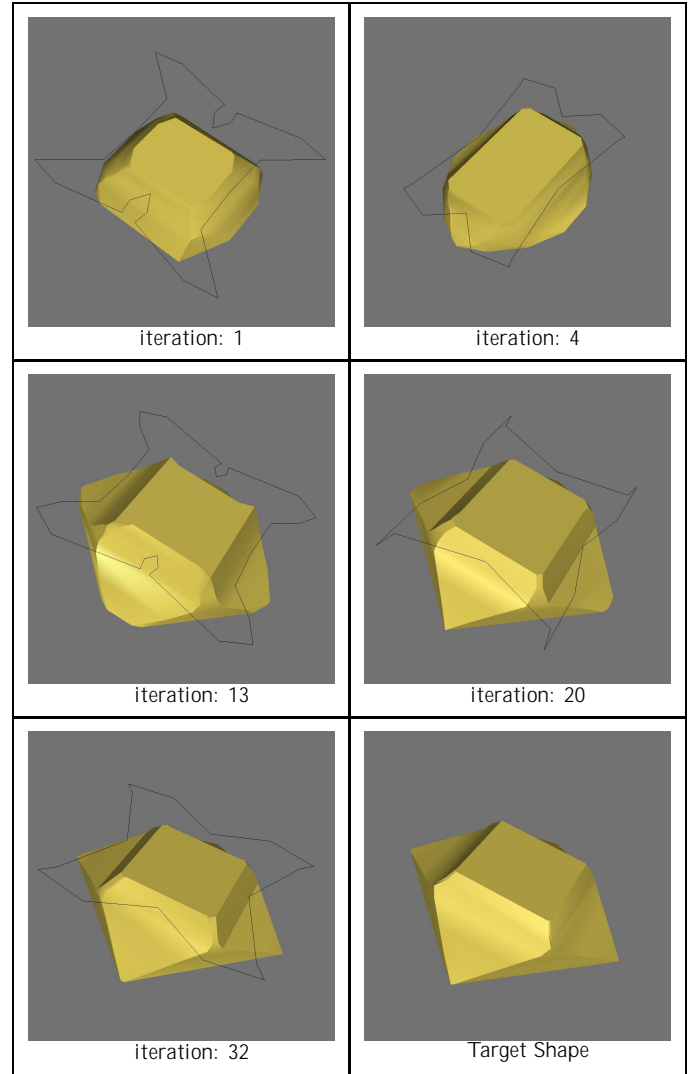


Figure 6. Mask-layouts and evolved shapes

etchant numbers and etch times are converging to 2, 1 and 3, 3 respectively. By iteration 32, though the mask shape is non-intuitive (actually the mask shape shows corner compensation), the resulting shape closely approaches the target shape. During the synthesis, mask-layout polygons with different numbers of sides are explored and the final mask-layout has 16 sides. Figure 7 shows the convergence curves of the shape, size, and slope mismatch values. Figure 8 shows the convergence of the side numbers of the mask-layout polygons to 16.

5 Robust Design

A large fraction of studies on GAs emphasize finding a globally optimal solution. But if a global optimal solution is sensitive

Table 1. Synthesis data showing convergence

Iteration	Etchant No.s	Etch Times	No. of Sides	Shape Match	Size Match	Slope Match
1	2, 2	4, 2	24	0.261	0.109	0.211
4	2, 1	5, 1	16	0.220	0.238	0.245
13	2, 1	4, 2	28	0.107	0.046	0.123
20	2, 1	3, 3	16	0.090	0.099	0.046
32	2, 1	3, 3	16	0.021	0.018	0.014

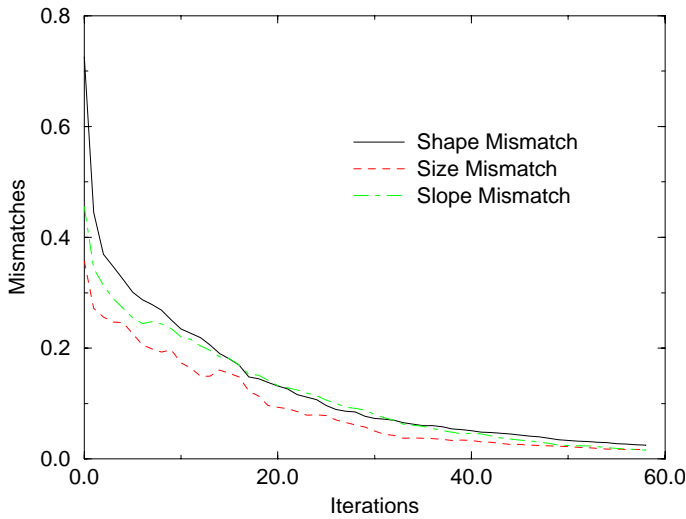


Figure 7. Convergence curves of shape, size, and slope mismatches

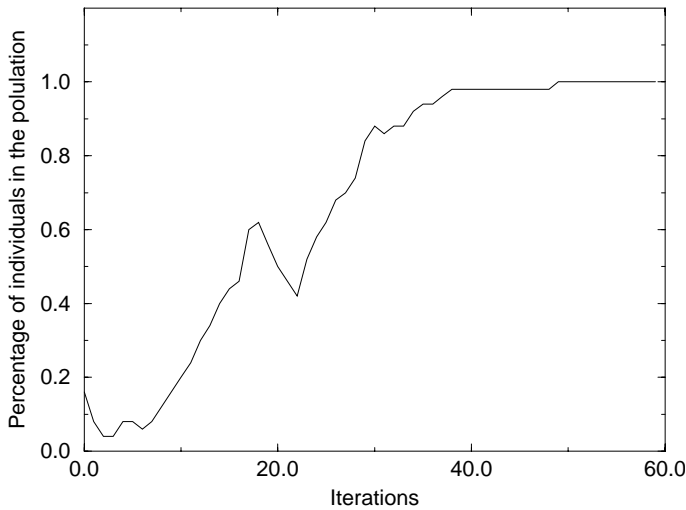


Figure 8. The percentage of individuals having 16 as the side number of mask-layout polygon converges to 100%

to noise or perturbations in the environment then there may be cases where it is not good to use this solution. The goal of robust design is to develop stable products and processes that exhibit minimum sensitivity to uncontrolled operational and manufacturing variations. To extend the applications of GAs to domains that require robust solutions, schemes combining GAs and robust solution searching techniques have been introduced (Fitzpatrick and Grefenstette, 1988; Miller and Goldberg, 1996; Tsutsui and Ghosh, 1997). For the mask-layout and process synthesis problem a robust design scheme way applied, similar to the one introduced by Tsutsui *et al.* (Tsutsui and Ghosh, 1997), which is outlined below.

In a GA, if $X = (x_1, x_2, \dots, x_m)$ is a phenotypic parameter vector, and $f(X)$ is the evaluation function, instead of calculating the fitness value of the individual as $f(X)$, an evaluation function of the form $f(X + \Delta)$ is used, where $\Delta = (\delta_1, \delta_2, \dots, \delta_m)$ is a random noise vector. The solutions thus determined are expected to be more robust against perturbations or noise having the appropriate tested distribution. It should be noted that adding noise in the form $f(X + \Delta)$ may appear like a mutation operation on a real-valued coding, but actually it is operationally different from mutation, since it does not have any direct effect on individual genotypes. The perturbations are used only for judging the quality of a solution and for selection.

For commercial silicon wafers, the alignment accuracy (between the flat and the crystal orientation) is usually around $\pm 1^\circ$, additionally, the mask may not be perfectly aligned to the wafer flat. This misalignment can affect the precision of shapes fabricated. For example, the size of diaphragms formed after etching through a $500\mu\text{m}$ thick silicon safer can vary by $50\mu\text{m}$ if the accuracy of the alignment is of the order of 1° . For this application, the optimum mask-layout and process used to fabricate the target shape is searched assuming perfect alignment, and in actual fabrication, the misalignment of the wafer flat, which implies the misalignment of mask-layout, is the noise factor which affects the quality of the fabricated device, and this variation of mask-layout misalignment can be taken as the uncontrolled variation when the robust design is conducted.

A robust synthesis of mask-layout and process is conducted, in which the mask-layout misalignment is considered as the uncontrolled noise with Gaussian distribution. For every individual in a generation, a randomly generated misalignment is assigned to the mask-layout before the fitness value is calculated. The final mask-layout and process found will have high robustness relative to mask misalignment. In the approach presented here, each mask-layout is evaluated for Gaussian distributed misalignment noise, and this greatly increases the computational cost since every mask-layout needs to be re-evaluated using a different misalignment even for the mask-layout conserved from last generation. To save computational cost, another approach to evaluate mask-layout with respect to misalignment is taken: for each newly generated mask-layout (from random initializa-

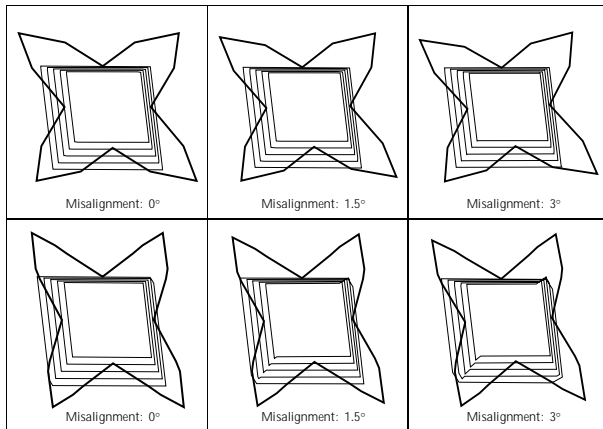


Figure 9. Comparison of GA1 and GA2

tion or crossover), it's evaluated with different levels of misalignment, and the worst fitness value will be taken as the final fitness value for this mask-layout. For example, if the maximum misalignment is 2° , then each mask-layout will be evaluated (by the simulation program) with misalignment of 0° , 1° , and 2° , and the worst fitness value among these three evaluations is taken as the final fitness value for this mask. Using this approach to calculate fitness value, the total computational cost is reduced since re-evaluation of individuals preserved from the previous generation is not needed, and since the misalignment noise factor is still included in the GA iterative procedure. The synthesis result will show greater robustness than the result from synthesis without considering mask misalignment.

Figure 9 shows the comparison of synthesis results from the robust design considering mask-layout misalignment (GA1) and the original GA without robustness consideration (GA2). The first row shows the mask-layout from GA1 and the fabricated shapes when the mask misalignments are 0° , 1.5° , and 3° . The second row is for GA2. Both mask-layouts generate shapes close to the target shape (mesa) when perfectly aligned (0° misalignment), but when the misalignment increases, the superiority of the mask-layout from GA1 can be easily observed.

6 CONCLUSIONS

A mask-layout and fabrication process synthesis technique combining a genetic algorithm and forward fabrication simulation is introduced here, and its feasibility is shown by an application of multiple process wet etching synthesis. Promising results have been obtained for initial tests, and laboratory verification is underway. As an extension of these methods, expected fabrication variation such as mask misalignment and process variation (e.g., etch rate variation) can be introduced into the fabrication simulation, and by combining these variations into

the synthesis iteration loop, the stochastic optimization will produce mask-layouts and process sequences that are least sensitive to these variations. Initial test on robust design on mask-layout misalignment has been conducted, and further development is underway. Such robust designs will greatly benefit real device development.

REFERENCES

- Antonsson, E. K., 1995, "Structured Design Methods for MEMS Final Report," *NSF MEMS Workshop* Nov.12-15,1995.
- Arkin, E. M., Chew, L. P., Huttenlocher, D. P., Kedem, K., and Mitchell, J. S. B., 1991, "An Efficiently Computable Metric for Comparing Polygonal Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 13, pp. 209–215.
- Baker, J. E., 1985, "Adaptive Selection Methods for Genetic Algorithm," In Grefenstette, J. ed., *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pp. 101–111.
- Eshelman, L. J., and Schaffer, J., 1991, "Real-Coded Genetic Algorithms and Interval-Schemata," *Foundation of Genetic Algorithms*, pp. 187–202.
- Fitzpatrick, J. M., and Grefenstette, J. J., 1988, "Genetic Algorithms in noisy environments," *Machine Learning*, Vol. 3, pp. 101–120.
- Fruhauf, J., and Hannemann, B., 1997, "Anisotropic Multi-step Etch Processes of Silicon," *J. of Micromechanics and Microengineering*, Vol. 7, pp. 137–140.
- Goldberg, D. E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning* Addison-Wesley.
- Holland, J. H., 1975, *Adaptation in Natural and Artificial Systems* The University of Michigan Press.
- Hubbard, T. J., and Antonsson, E. K., 1996, "Design of MEMS via Efficient Simulation of Fabrication," In *Design for Manufacturing Conference*. ASME.
- Kang, I., Haskard, M. R., and Samaan, N. D., 1997, "A Study of Two-step Silicon Anisotropic Etching for a Polygon-shaped Microstructure Using KOH Solution," *Sensors and Actuators A-Physical*, Vol. 62, pp. 646–651.
- Li, H., and Antonsson, E. K., 1999, "Mask-Layout Synthesis Through an Evolutionary Algorithm," In *MSM'99, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators* San Juan, Puerto Rico. IEEE.
- Miller, B. L., and Goldberg, D. E., 1996, "Genetic Algorithms, Selection Scheme, and the Varying Effect of Noise," *Evolutionary Computation*, Vol. 4, pp. 113–131.
- Tsutsui, S., and Ghosh, A., 1997, "Genetic Algorithms with a Robust Solution Searching Scheme," *IEEE Transactions on Evolutionary Computation*, Vol. 1, pp. 201–208.