Cellular Automata Modeling in MEMS Design *

Ted J. Hubbard
Assistant Professor of Mechanical Engineering
Technical University of Nova Scotia

Erik K. Antonsson
Associate Professor of Mechanical Engineering
California Institute of Technology

Corresponding address:
Technical University of Nova Scotia
PO Box 1000, Halifax, Nova Scotia, Canada, B3J 2X4
FAX: 902/423-6711, email: hubbardt@tuns.ca

February 5, 1999

Abstract

This paper presents a robust cellular automata model which simulates anisotropic etching and predicts the three-dimensional etched shape as a function of time for arbitrary etchants and arbitrary initial mask shapes. This method applies the basic approach of all cellular automata to etching: to divide the spatial domain into small cells, provide each cell with simple and primitive (but appropriate) behavior, and the aggregate behavior of many cells will mirror the complex behavior of physical systems. The predictions of the model are compared with experimental results and shown to be in agreement.

KEYWORDS:
MEMS, micromachines, CAD, etching simulation, cellular automata

* Manuscript prepared for submission to *Sensors and Materials*

1 Introduction

Many useful micro-electro-mechanical systems (MEMS) are now being built using silicon etching technologies. Proposals for MEMS computer-aided design (CAD) systems have been made in recent years and considerable work has been done to establish the best architecture for such a system [1, 2]. While much work has been done in other parts of CAD systems, there remains a need for an improved etch simulator. The fundamental problem is how to model the complex transformation from the initial two-dimensional input mask to the final three-dimensional output shape, particularly when highly anisotropic etchants are used.

An etch simulation method, based on cellular automata, is presented in this paper. The basic approach is to divide a wafer of silicon into small cubic cells and give each cell a few primitive rules governing its rate of removal when exposed to an etchant. If these few simple rules are properly written, then the aggregate behavior of all the cells will accurately represent the complex geometry of a silicon wafer being etched.

When silicon is etched with anisotropic etchants, the etched shape changes as a function of time. A number of different approaches exist to accurately predict the etched shape given an initial mask [1, 3, 4, 5]. The ASEP program by Buser [6] uses traveling planes and the intersections between them to define the shape as a function of time. The Slowness method of Sequin [7] examines the corners of a shape and predicts the trajectory of those corners. This is done using a vector expression involving the inverse of the rate (the slowness) of the planes which make up the corner. The Eshape method of Hubbard and Antonsson [8] precalculates the traveling planes and then extracts the etched shape section by section. These models deal well with most basic shapes. However, complex shapes are more difficult for these methods to simulate. For example, when two distinct shapes merge to form a new shape (termed through-cut), substantially more computation must be done to detect and locate the new intersections and corners.

The robust cellular automata model presented here predicts the three-dimensional etched shape as a function of time for arbitrary etchants and arbitrary initial mask shapes. Previous work has been done on atomistic approaches [7, 9], as well as on cellular automata, in particular, Than et al. [10]. This paper will focus specifically on cellular automata and address, at greater length, issues such as generality, efficiency, and speed.

2 Materials and Methods

The ASEP, Slowness, and Eshape methods (mentioned above) are high level; the fundamental modeling unit or primitive is defined in terms of endpoints with an associated interior. The intersection points between two or more units are calculated and new endpoints delimiting new interiors are determined. In two dimensions the units are line segments, while in three dimensions they are planar polygons. For example, a sharp, two-dimensional corner is defined by the moving intersection point of the two segments that make up that corner.

The advantage of this approach is that each unit is defined at a relatively high level of abstraction. Thus, relatively few units are needed to define shapes. The disadvantage of this approach is the difficulty of calculating interactions between etching primitives. In

general, as the etched shapes are modeled by fewer, more complex primitives the computational intensity required to calculate the change in primitives increases. When only local calculations are performed (i.e., between connected or adjacent primitives), the number of computations is low, since only only a few neighbors may interact. Unfortunately, many interesting phenomena require global calculations (i.e., between unconnected or initially distant primitives); all primitives must be checked against each other for interactions.

The cellular automata model in contrast, uses a low-level representation, where each shape is represented by a sequence of cells. Each cell interacts *only* with its neighbors, there are no global interactions. Moreover, all interactions are computationally efficient. Thus, cellular automata always perform many calculations, but the individual calculations are small and the net cost (number of calculations times the cost of each calculation) can be less than other methods for typical etched shapes and resolutions.

2.1 Scaling

The cellular automata method is based on the principal of scaling presented in [8]. For a given mask shape, the form of the etched shape does not change if the initial mask size is scaled, although the output size and effective etch time do change. For example, if an initial mask etches to an output shape in time t, then a half scale mask etches to a half scale output, but does so in time $t/2$. Thus, the mask can be reduced to smaller and smaller sizes until the crystal nature of silicon becomes important. In effect, the cellular automata method replaces many atoms in a shape with one atom (or cell). Within the shape, groups of atoms behave as if they were only one atom obeying crystallographic laws. In this way the number of required primitives is minimized.

2.2 Empirical vs. theoretical

Our analysis of the etching of silicon has been empirical rather than theoretical. The theoretical approach involves incorporating and using the chemistry of both the silicon and the silicon etchant. However, such chemical reactions are very complex and difficult to model. Our emphasis has been on using experimental data and experimental observations to formulate empirical rules that accurately model the etching. The use of such rules allows for the efficient implementation of modelers which would otherwise be very computationally intensive. In formulating these etching rules, two guidelines must be followed: (i) the rules must produce accurate results, and (ii) the model rules must agree with the microscopic chemical behavior.

The empirical rules used in the cellular automata model reduce the complexity of the interactions between cells while still producing accurate predictions of the changing shapes as a function of time.

3 Results

3.1 Two-dimensional algorithm

The cellular automata model will first be presented in two dimensions then extended to three dimensions.

To begin developing a two-dimensional cellular automata etch simulation, consider the diamond lattice crystal structure of silicon, as shown in Fig. 1. For the purposes of simulation, the diamond structure of the lattice is converted into an array of cells. If we examine a plane of atoms parallel to the surface of a (100) wafer, one lattice unit cell deep, from above we see the pattern shown in Eq. 1 (a 1 represents an atom) and Fig. 2 (a circle represents an atom). An array of cells is constructed such that it contains a 1 when an atom is present and a 0 (or is empty) when no atom is present. Each two-dimensional unit cell resembles the number 5 on a throwing die, and has four nearest neighbors, as shown in Eq. 1.

$$cell\ array = \begin{bmatrix} 1 & & 1 & & 1 & & 1 & & \cdots \\ & 1 & & 1 & & 1 & & 0 & \\ 1 & & 1 & & 0 & & 0 & & \cdots \\ & 1 & & 0 & & 0 & & 0 & \\ \vdots & & \vdots & & \vdots & & \vdots & & \ddots \end{bmatrix} \qquad unit\ cell = \begin{bmatrix} 1 & & 1 \\ & 1 & \\ 1 & & 1 \end{bmatrix} \quad (1)$$

Once the etching begins, at each cell in sequence, the number of nearest and next-nearest neighbors is counted. From this information, the local plane is classified as either (111), (101), or (311). For example, the (111) plane has three nearest neighbors while the (101) plane has only two, as illustrated in Fig. 3.

As might be expected, the fastest planes (as measured experimentally) have fewer nearest neighbors. Having fewer next-nearest neighbors also tends to increase the etch rate, although this is a smaller effect. Table 1 shows the neighbor conditions for the different planes. These planes were chosen since they preferentially appear in most shapes etched with common bulk silicon etchants such as EDP and KOH. For different etchants, different tables would need to be formulated. The cellular automata model is not limited to any particular etchant or set of planes.

At each time step a fraction of each exposed cell is removed, based on the number of nearest and next-nearest neighbors that the cell has at this time step. The number of neighbors determines the plane on which the cell resides. The rate of etching perpendicular to these planes is known, thus a calibrated fraction of the cell is removed at each time step. Each cell contains a scalar value that represents the remaining fraction in that cell. All filled cells start with a 1 (100%) for this value. With this scheme, etchants of any rate and selectivity can be modeled, isotropic to anisotropic. When 100% of a cell is removed, it is removed from further calculations, which exposes cells behind it to etching at subsequent times steps. These steps are then repeated to find the etched shape at each time step. In this way, new planes are introduced automatically.

3.2 Three-dimensional extension

This method can easily be extended to three dimensions when a vertical stack of cell arrays is used. Once the topmost array has been calculated for a particular time, it is used to calculate the next lower array. To calculate each succeeding array, at each cell in the lower layer, the five nearby cells above it are examined. If any of these cells are present, the lower cell is protected and kept, otherwise it is removed. This lower array is then used to calculate the one beneath it. This process is repeated for each time step. Tables 2 and 3 list pseudo-code implementations of the cellular automata method.

The three-dimensional algorithm described above is valid for etchants modeled by the (111), (100), (101), and (311) planes. This is the case for a (100) wafer etched with EDP. Different etchants have different dominant planes; with KOH etching, the (101) planes (45 degree walls) are replaced by the (010) planes (vertical walls). If a different etchant is to be modeled, a slightly different three-dimensional extension algorithm must be used.

A sample of the cellular automata simulations (using EDP as the etchant) is shown in Fig. 4. Figure 5 shows the result of the cellular automata simulation for an initial cross-shaped mask. The simulation shows good agreement with the experimental results (see Fig. 6). The inside cross corners are bounded by (111) planes, while the outside cross corners are made up of (311) and (131) planes.

3.3 Full three-dimensional algorithm

In addition to the two-dimensional model extended to three dimensions, a full three-dimensional algorithm was developed. In this case, the actual diamond lattice is modeled using a three-dimensional array having elements at each atom location. In three dimensions, each atom has four nearest neighbors. Referring to the five-atom unit cell of two dimensions (see Eq. 1), the central atom remains the same, but the upper left and lower right atoms are one atomic level lower, while the lower left and upper right atoms are one atomic level higher. Each cell also has twelve next-nearest neighbors. As with the two-dimensional case, a classification scheme is developed based on the the number of nearest and next-nearest neighbors. This algorithm has been used to predict output shapes with results similar to those for the extended two-dimensional case. However, this algorithm is slower. One reason for the slowness is that this array is not symmetric. Planes along the (upper left - lower right) line will differ in height from planes along the (lower left - upper right) line by two atom heights. Thus, the results must be averaged over two atom heights, and the number of calculations is doubled. The two-dimensional extension discussed above in effect removes this asymmetry.

4 Discussion

4.1 Two dimensions vs. three dimensions

Both the two- and three-dimensional algorithms produce similar results, and both classify planes according to the number of neighbors, with more neighbors meaning a denser plane. However, it is important to note that they do so by using two different classification

schemes. The three-dimensional algorithm is a general approach, while the extended two-dimensional algorithm is a specialized subset of the full three-dimensional case which uses certain conditions (i.e., a (100) wafer) to reduce the complexity of the rules.

The choice of classification rules involves a trade off between accuracy of modeling and speed of implementation. A very complex rule system involving many neighbors will provide a more detailed model of the etching, but at the price of increased computation. In practice, we choose the extended two-dimensional algorithm over the full three-dimensional algorithm, since it provides a sufficiently accurate result in a faster, more efficient implementation.

4.2 Generality

In this paper, the (111), (100), (110), and (311) planes were chosen since they appear often in bulk etching with EDP or KOH. The cellular automata model is not limited to such systems. Any etch process for which empirical data is available can be modeled. The user would construct a plane table as in Table 1. This new table would then be used in the cell removal loop shown in Table 2 to remove fractions of the cell based on the relative etch rates. Additional planes can be modeled by including the next-next-nearest neighbors in the classification, although the computational cost is increased.

4.3 Computational cost

Consider an area divided into N by N cells. Most higher level etch simulations scale on the order of N in two dimensions, but cellular automata fair less well. They scale as N^3 (N^2 cells are needed and the effective time scales with N). If the size of the array is doubled, then twice as many time steps are required (since the cell size is halved). In three dimensions, all models multiply an extra factor of N in the scaling.

While the actual computation time depends on the computing platform, the following is provided as a rough guide. For three-dimensional cellular automata, sample calculations for 20 time steps of a 50 by 50 cell input mask required several seconds on a work station. A 100 by 100 array for 40 time steps would require a few minutes, a 200 by 200 for 80 time steps would require on the order of half an hour. We have found that grid sides on the order of 100 to 200 cells provided acceptable resolution within reasonable times.

One proposed solution to the rapid rise in processing time is a perimeter-based cellular automata method. Such a method would examine only the boundary of shapes rather than the entire cell array, thus decreasing the number of cells which must be examined and potentially increasing the modeling speed.

4.4 Conclusion

A cellular automata etch simulation method has been developed and presented. This method exhibits several advantages over existing geometry-based etch simulators in that complex geometry may be modeled. Very little computation is performed for each cell, and the rules embodied by each cell are simple. The aggregate behavior of these simple cells reflects the geometry observed in silicon etching.

Despite the computational limitations of the method (which imposes a trade off between the spatial resolution of the cellular automata etch simulation technique and the simulation time) the approach's robustness to complex geometry provides a clear advantage for many MEMS design problems.

Acknowledgments

The authors are grateful to Professor Y.C. Tai for the use of his facilities and many helpful discussions. This material is based upon work supported, in part, by the National Science Foundation under Grant No. ECS-9023646. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsor.

References

[1] S. D. Senturia, R. M. Harris, B. P. Johnson, S. Kim, M. A. Shulman, and J. K. White. A computer-aided design system for microelectromechanical systems (MEMCAD). *Journal of Microelectomechanical Systems*, 1:3–13, March 1992.

[2] K. D. Wise. Integrated microelectromechanical systems: A perspective on MEMS in the 90s. In *MEMS '91*, pages 33–38, Institute of Electrical Engineers, 1991.

[3] G. DeLapierre. Anisotropic crystal etching: A simulation program. *Sensors and Actuators*, 31:267–274, 1992.

[4] D. W. Shaw. Morphology analysis in localized crystal growth and dissolution. *J. Cryst. Gr.*, 47:509–517, 1979.

[5] T. Thurgate. Segment based etch algorithm and modeling. *IEEE Transactions on computer-aided design*, 10(9):1101–1109, September 1991.

[6] R. A. Buser and N. F. de Rooij. ASEP: A CAD program for silicon anisotropic etching. *Sensors and Actuators*, 28:71–78, 1991.

[7] C. H. Sequin. Computer simulation of anisotropic crystal etching. *Sensors and Actuators A Physical*, 34(3):225–241, September 1992.

[8] Ted J. Hubbard and Erik K. Antonsson. Emergent Faces in Crystal Etching. *Journal of Microelectomechanical Systems*, 3(1):19–28, March 1994.

[9] H. Camon and A. Gue. Modelling of an anisotropic etching in silicon-based sensor applications. *Sensors and Actuators*, 33:103–105, 1992.

[10] O. Than and S. Buttgenbach. Simulation of anisotropic chemical etching of crytalline silicon using a cellular automata model. *Sensors and Actuators*, A45:85–89, 1994.

References

1 S. D. Senturia, R. M. Harris, B. P. Johnson, S. Kim, M. A. Shulman and J. K. White: Journal of Microelectomechanical Systems **1** (1992) 3.

2 K. D. Wise: MEMS ’91 (Institute of Electrical Engineers, city, 1991) p. 33.

3 G. DeLapierre: Sensors and Actuators **31** (1992) 267.

4 D. W. Shaw: J. Cryst. Gr. **47** (1979) 509.

5 T. Thurgate. IEEE Transactions on computer-aided design (IEEE, city, 1991) p. 1101.

6 R. A. Buser and N. F. de Rooij: Sensors and Actuators **28** (1991) 71.

7 C. H. Sequin: Transducers ’91 (publisher, San Francisco, 1991) p. 801.

8 T. J. Hubbard and E. K. Antonsson: Journal of Microelectomechanical Systems **3** (1994) 19.

9 H. Camon and A. Gue: Sensors and Actuators **33** (1992) 103.

10 O. Than and S. Buttgenbach: Sensors and Actuators **A45** (1994) 85.

Biographies

Ted J. Hubbard (ASME Student Member) received the B.Sc. degree (1987) in Physics from Dalhousie University, the B. Eng. degree (1990) in Engineering Physics from TUNS, and the Ph.D. degree (1994) in Mechanical Engineering from Caltech, Pasadena, CA. He was a Post-Doctoral Research Fellow in the Mechanical Engineering Department at Caltech in 1994. In 1995 he joined the Mechanical Engineering faculty at DalTech (TUNS), as an Assistant Professor. His research interests include engineering design, computer aided engineering design, and micro-mechanical systems..

Erik K. Antonsson (IEEE Affiliate '85, IEEE Member '93, ASME Member '82) received the Ph.D. degree (1982) in mechanical engineering from the Massachusetts Institute of Technology. In 1983 he joined the Mechanical Engineering faculty at the University of Utah, as an Assistant Professor. In January 1984 he became the Technical Director of the Pediatric Mobility and Gait Laboratory at the Massachusetts General Hospital. He simultaneously joined the faculty of the Harvard University Medical School as an Assistant Professor of Orthopaedics. In September 1984 he joined the faculty of the California Institute of Technology as an Assistant Professor of Mechanical Engineering, where he is now an Associate Professor of Mechanical Engineering. His research interests include computation in the preliminary phase of engineering design, imprecision in preliminary engineering design, the design of micro-electro-mechanical systems, and optical surface geometry acquisition.

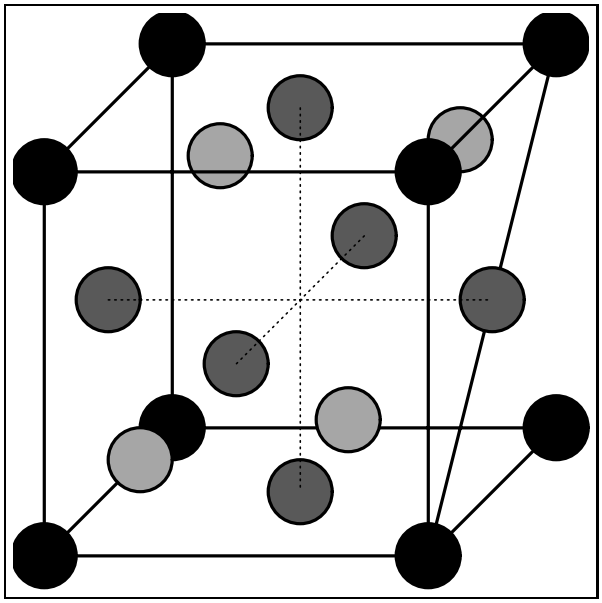

Figure 1: Silicon basis cell.

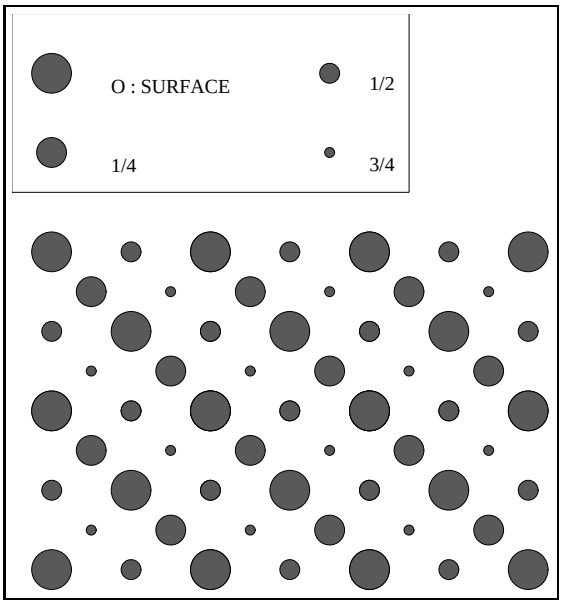

Figure 2: Silicon basis cells viewed from above, size indicates depth. This particular figure shows 3 by 2 by 1 unit cells

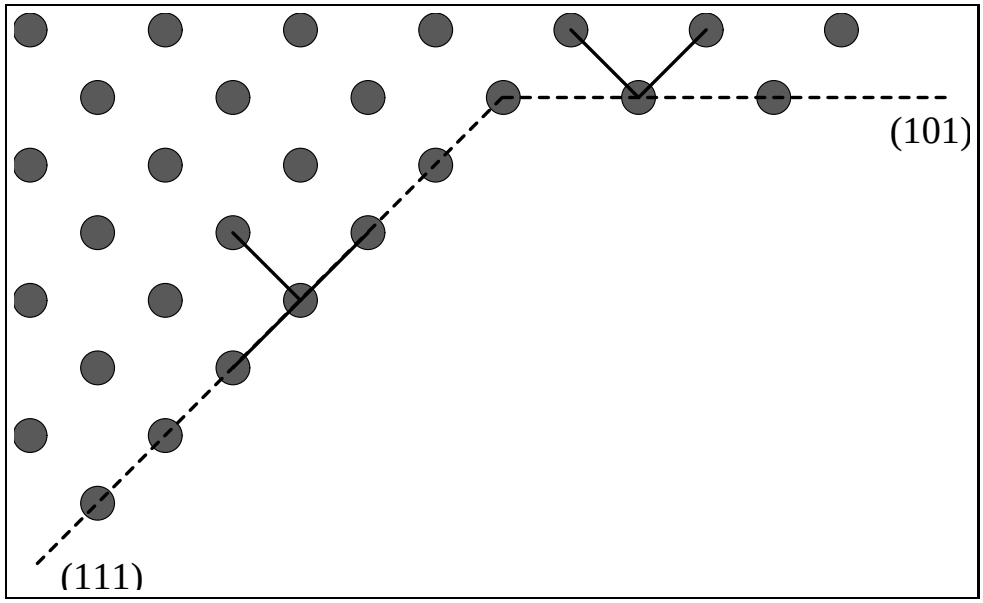

Figure 3: Classification of planes based on neighbors.

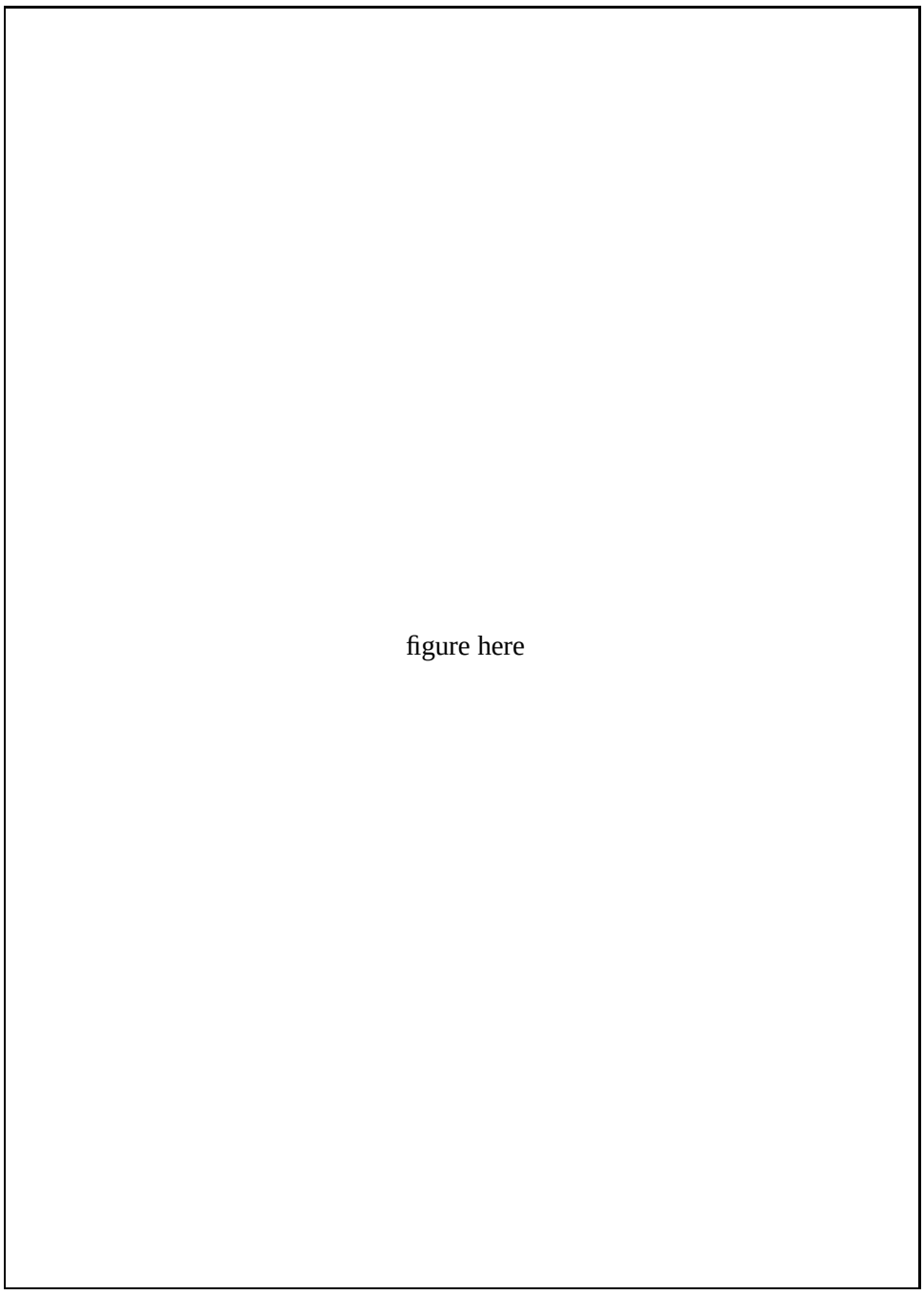

Figure 4: Sample of cellular automata algorithm output.

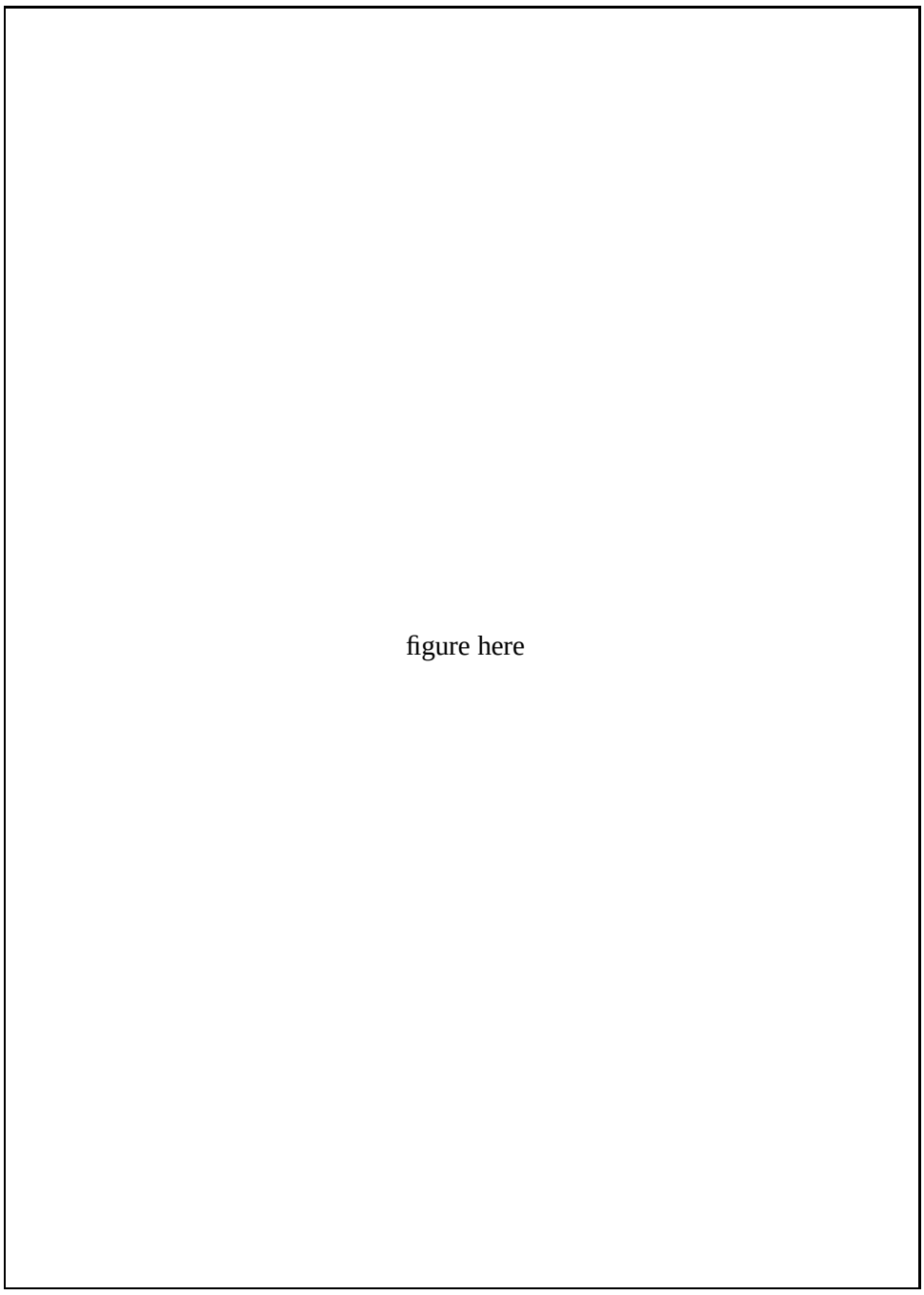

Figure 5: Cellular automata algorithm output for initially cross-shaped pegs.

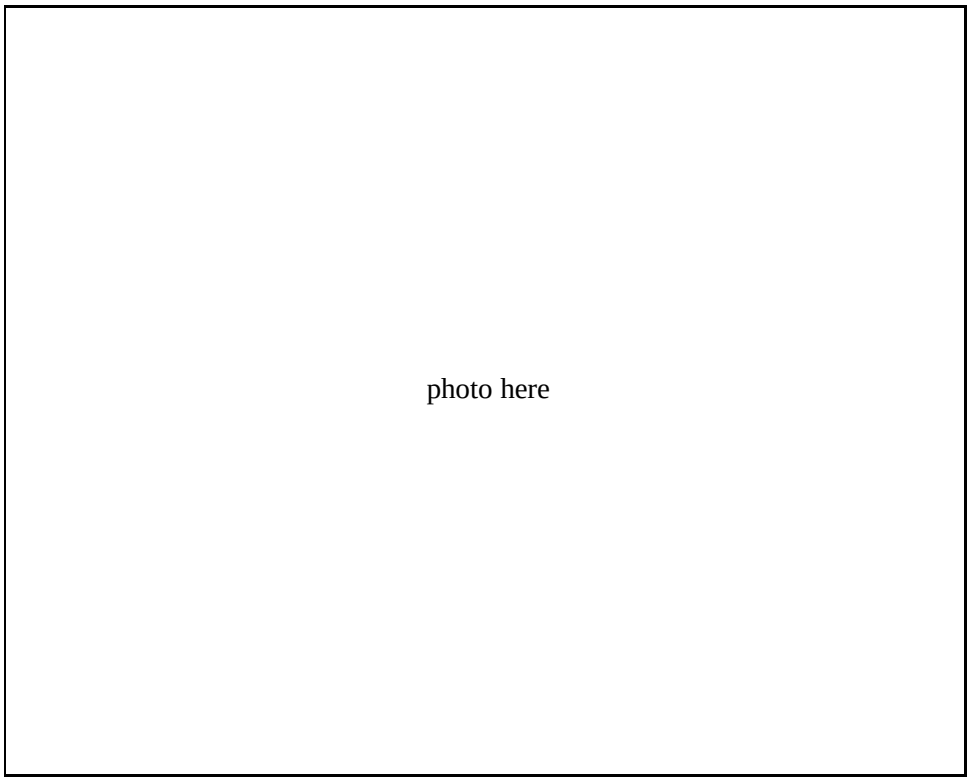

Figure 6: Experimental output shapes for initially cross-shaped pegs. Compare the largest peg with time step 6 in Fig. 5.

plane	*nearest neighbors*	*next neighbors*	*isotropic*	*anisotropic*
(111)	3	2	*same*	*slowest*
(100)	2	3	*same*	*intermediate*
(110)	2	3	*same*	*intermediate*
(311)	2	2	*same*	*fastest*

Table 1: Number of neighbors and relative etch rates for common planes.

```
Input percentages to be removed for each plane (etch data)
Create (N x N) main array
Create (N x N) temporary array
Write mask data to main array
For time step
{
    For each (i,j) element in main array
    {   count occupied nearest cells: (i+1,j-1), (i+1,j+1), (i-1,j-1), (i-1,j+1)
        count occupied next-nearest cells: (i,j+2), (i,j-2), (i+2,j), (i-2,j)
        use Table 1 to classify plane
        remove proper percentage of cell (i,j) in temporary array
    }
    Switch main array and temporary array
}
```

Table 2: Pseudo-code implementation of the two-dimensional cellular automata method.

```
Input percentages to be removed for each plane (etch data)
Create (N x N x depth) main 3d array
Create (N x N) temporary 2d array
Write mask data to main 3d array, depth level d=0
For each time step
{
    Repeat two dimensional (i,j) element loop from Table 2 on depth level d=0
    For each depth level d
    {
        For each (i,j) element in depth level d+1:
        {   count occupied nearest cells in level d: (i+1,j-1), (i+1,j+1), (i-1,j-1), (i-1,j+1)
            if count = 0 then remove cell (i,j) in level d+1 of temporary array
        }
        Switch main 3d array depth level d+1 and temporary array
        Increment depth level d
    }
}
```

Table 3: Pseudo-code implementation of the three-dimensional cellular automata method.