Fábio Belotti Colombo

DESENVOLVIMENTO DE UM SOFTWARE PARA SIMULAÇÃO
ATOMÍSTICA DE PROCESSOS DE MICROFABRICAÇÃO BASEADO
EM AUTÔMATOS CELULARES

São Paulo

2011

Fábio Belotti Colombo

DESENVOLVIMENTO DE UM SOFTWARE PARA SIMULAÇÃO
ATOMÍSTICA DE PROCESSOS DE MICROFABRICAÇÃO BASEADO
EM AUTÔMATOS CELULARES

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Engenharia.

Área de Concentração:
Engenharia Elétrica

Orientador: Prof. Dr. Marcelo Nelson Paéz
Carreño

São Paulo
2011

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, de julho de 2011.

Assinatura do autor _____

Assinatura do orientador _____

FICHA CATALOGRÁFICA

Colombo, Fábio Belotti

**Desenvolvimento de um software para simulação automática de processos de microfabricação baseado em autômatos celulares / F.B. Colombo. -- ed.rev. --São Paulo, 2011.
135 p.**

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1. Softwares (Simulação) 2. Processos de microeletrônica 3. Autômatos celulares 4. Sistemas microeletromecânicos I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II. t.

DEDICATÓRIA

Aos meus pais.

AGRADECIMENTOS

Aos meus pais, pelo apoio ao longo dos meus estudos.

À Simone, pela ajuda e por ser um grande apoio.

Aos Prof. Dr. Marcelo Carreño, ao Prof. Dr. Marco Alayo e à Profa. Dra. Inés Pereyra pela ajuda e os conselhos ao longo desse trabalho.

Ao Fabio, o Hugo e o Murilo pelas conversas e pela ajuda com o software.

À FAPESP e ao CNPq, pela concessão da bolsa de mestrado para a realização desta pesquisa.

RESUMO

O presente trabalho teve como foco o desenvolvimento de um software para a simulação de processos de microfabricação em substrato e de microfabricação em superfície baseado em autômatos celulares, o simMEMS. Além disso, visando a futura incorporação de ferramentas para análise das estruturas geradas pelo programa, um módulo com funcionalidades básicas para a análise mecânica de estruturas também foi desenvolvido. No que tange à microfabricação em superfície, o software desenvolvido permite simular a corrosão anisotrópica úmida do Si em KOH e deep reactive ion etching (DRIE). O simulador de corrosão úmida utiliza um autômato celular conhecido como BCA. O simulador de DRIE usa um autômato próprio. Para a simulação dos processos de microfabricação em superfície o software fornece quatro processos: deposição de filmes, corrosão de filmes, fotolitografia e planarização. Para corrosão e deposição de filmes, diversos autômatos celulares da literatura foram analisados e os resultados dessas análises é aqui apresentado. Todos os simuladores, tanto de microfabricação em superfície como em substrato, podem ser utilizados em conjunto. Isso torna o software bastante útil e capaz de simular a fabricação de um grande número de dispositivos.

ABSTRACT

The main goal of this project is the development of a software capable of simulating both surface and bulk micromachining based on a cellular automata approach. This software has been called simMEMS. In order to enable future versions of the software to also be able to analyze the structures created by the software, a module capable of running a mechanical analysis through the finite element method is also developed. simMEMS allows the user to simulate two bulk micromachining processes: wet anisotropic KOH etching and deep reactive ion etching DRIE. The wet etching simulator uses a cellular automaton known as BCA. The DRIE simulator uses an automaton developed during this project. The surface micromachining simulator allows the user to simulate four types of processes: photolithography, film deposition, film etching and substrate planarization. Several automata for the deposition and etching of films are studied and the results of this study are presented here. All processes, be they for surface or bulk micromachining, can be used on the same substrate to simulate the entire fabrication process for a large array of devices. This makes simMEMS a very useful software.

LISTA DE FIGURAS

Figura 1 – Vizinhanças de Moore e de von Neumann. A célula central está colorida em azul e os seus vizinhos em verde.....	23
Figura 2 – Exemplo da vizinhança de Margolus. Nos passos pares (a) e (b) as células verdes são as vizinhas da célula azul e as laranja vizinhas da célula lilás. Nos passos ímpares (c) a célula azul tem como vizinhas as células amarelas e a lilás.	23
Figura 3 – Condição de contorno aplicada a uma rede bidimensional. Em (a) a vizinhança definida para o autômato celular é mostrada. Em (b) a mesma vizinhança é mostrada para uma célula na borda utilizando a condição de contorno	25
Figura 4 – A figura mostra as células da rede com borda completa e o estado em preto. As células virtuais têm bordas tracejadas e o estado em cinza. Uma vizinhança para uma célula com todos os vizinhos reais é mostrada em verde e azul na imagem (a). A mesma vizinhança é mostrada para uma célula da borda da rede em (b).	26
Figura 5 - Evolução temporal para da Regra 184 considerando a simulação de trânsito. Um carro foi colorido em roxo para auxiliar na noção de movimento.	29
Figura 6 - Evolução de uma superfície devido à deposição balística simulada pela Regra 184.....	30
Figura 7 – Uma partícula indo em direção à direita tem uma probabilidade p manter sua direção e continuar para a direita e $1 - p$ de mudar de direção e passar a se movimentar para a esquerda.....	32
Figura 8- Simulação de um processo de difusão. (a) Condição inicial. (b) Resultado após 2000 passos com $p = 0,1$. (c) Resultado após 2000 passos com $p = 0,6$	33
Figura 9 - Micro pontas de silício. Os resultados são: a) experimental; b) simulado com autômato convencional em 140 iterações; c) simulado com autômato estocástico em 125 iterações e usando as taxas padrões; d) simulado com autômato	

contínuo com a regra 3334 alterada para o valor máximo e em 280 iterações (DE OLIVEIRA JR., 2008).	34
Figura 10 – Refração de uma onda determinada pelo princípio de Huygens (NORDMANN, 2007).	36
Figura 11 - Crescimento isotrópico de uma superfície através do uso do princípio de Huygens. Na figura (a) temos a superfície original. A figura (b) mostra as ondas de matéria gerada por diversos pontos. A figura (c) mostra a demarcação do envelope para as ondas geradas por todos os pontos da superfície. A figura (d) mostra a superfície final. O detalhe mais escuro mostra quanto a superfície cresceu.	37
Figura 12 - Corrosão anisotrópica de uma superfície através do uso do princípio de Huygens. Na figura (a) temos a superfície original em cinza. A figura (b) mostra as ondas elípticas de matéria gerada por diversos pontos dessa superfície. A figura (c) mostra a demarcação do envelope, em laranja, para as ondas geradas por todos os pontos da superfície. A figura (d) mostra a superfície corroída final e a região removida em um tom mais claro.	38
Figura 13 – Evolução de uma simulação de corrosão do material "1". Na figura "0" indica uma célula vazia e "1" e "2" indicam células do substrato.	41
Figura 14 – Um elemento de pórtico com dois nós e três graus de liberdade em cada nó.	49
Figura 15 – Na direita temos uma estrutura e na esquerda a sua separação em elementos.	50
Figura 16 – Elemento de pórtico utilizado no programa. Os nós são representados pelos pontos em vermelho. Os graus de liberdade estão em laranja.	51
Figura 17 – O elemento ISOQ4 em azul com os nós em vermelho e os graus de liberdade em laranja.	54
Figura 18 – Malha para aproximar uma curva na estrutura.	54
Figura 19 – Mapeamento das coordenadas do elemento para as coordenadas Cartesianas onde a estrutura física está definida (FISH e BELYTSCHKO, 2007).	55
Figura 20 – Diagrama de classes para implementação de autômatos celulares.	59

Figura 21 – Diagrama de classe mostrando a estrutura básica da camada Core para implementação de operações.	65
Figura 22 – Diagrama de classe mostrando o Observer pattern.	68
Figura 23 – Imagem mostrando como é feita a medida experimental da taxa de deposição. A figura mostra o estado inicial para a medida da deposição em (a). A célula em vermelho é a semente para o crescimento. Em (b) tem-se o estado do autômato após n passos. A medida da distância depositada em uma direção também é mostrada em (b).	70
Figura 24 – Medida do raio para o disco ideal.	71
Figura 25 – Possíveis erros ao comparar com a forma ideal. O erro A indica uma célula que deveria ter sido depositada e não foi. O erro B indica uma célula que foi depositada mas não deveria ter sido.	72
Figura 26- Simulação de grãos de areia usando um autômato celular (a) e um autômato celular não determinístico (b).	74
Figura 27 – Comportamento dos grãos de areia utilizado para simular a Figura 26.	74
Figura 28 – Direções das partículas do FHP em (a) e a vizinhança do autômato em (b).	75
Figura 29 – Exemplo de uma configuração com soma da quantidade de movimento nula (a). A colisão dessas duas partículas pode gerar tanto o resultado em (b) como em (c).	76
Figura 30 – Deposição de um filme usando FHP para simular as partículas do filme. O número de passos simulados está ao lado das imagens. O Substrato está em verde e o filme depositado em amarelo escuro. As partículas são representadas em escala de cinza e as paredes em vermelho.	77
Figura 31 – Vários passos de uma corrosão de uma esfera de Si em solução de KOH simulado pelo BCA. A imagem mostra somente os átomos da superfície sendo corroída.	79
Figura 32 – Evolução da deposição de um filme (em oliva) sobre uma mesma superfície (em vermelho) utilizando três valores diferentes para o passo de tempo.	81

Figura 33 – Substrato inicial e a evolução do autômato celular. Cada imagem mostra a evolução após dez passos. O substrato está colorido em vermelho e o filme em oliva.....	83
Figura 34 – Evolução do autômato simples em três dimensões. Vemos claramente o achatamento da ponta, semelhante ao que ocorre em duas dimensões. As imagens de (a) a (f) mostram a evolução da deposição de dez em dez passos. Em (g) temos um corte após vinte passos.....	85
Figura 35 – Crescimento do filme, em oliva, sobre o substrato, em vermelho, usando o autômato celular para simulação da evolução de superfícies curvas.....	86
Figura 36 – Substrato inicial e a evolução do autômato celular contínuo. Cada imagem mostra a evolução após dez passos.....	88
Figura 37 - Definições das áreas A1 e A2.....	90
Figura 38 – Evolução do autômato celular para oxidação, dada de 20 em vinte passos. O substrato de Si está em cinza e o óxido em azul.	93
Figura 39- Na imagem da esquerda são mostradas as ondas de matéria partindo do centro de três células. Na direita, os envelopes para diferentes instantes formados por todas as ondas geradas são destacados. A cor indica de qual célula surge a onda que gerou cada seção do envelope.	95
Figura 40 - Passos de fabricação de um rotor.....	97
Figura 41 - Célula de diamante do Si cristalino.	99
Figura 42 - Conversão da rede cristalina do silício em uma rede cúbica. Nas figuras acima vemos um vista do topo e da lateral, respectivamente, do cristal. As figuras abaixo mostram em qual camada fica cada átomo. As camadas estão indicadas na visão lateral do cristal.	100
Figura 43 – Planos cristalográficos (110) em (a) e (111) em (b) do Si.	101
Figura 44 – Primeiros (a) e segundos (b) vizinhos usados na simulação da corrosão úmida de Si.	101
Figura 45 - Fabricação de uma microponta utilizando material de mascaramento em formato de cruz. Na esquerda são apresentados os resultados do simulador e na direita os resultados experimentais.	107

Figura 46 – Simulação de uma processo DRIE seguido de uma corrosão úmida em KOH. No detalhe, processo semelhante usando “dicing” mecânico ao invés de DRIE. A imagem em (6) é de (SHIKIDA, ODAGAKI, <i>et al.</i> , 2004).	109
Figura 47 – Fabricação de uma microponta utilizando o software. As imagens mostram os resultados da corrosão úmida em As imagens dos resultados experimentais são de (SHIKIDA, ANDO, <i>et al.</i> , 2004).....	111
Figura 48 – Detalhes da ponta. As duas imagens da esquerda mostram a ponta ampliada vista como átomos e como as células do autômato celular, respectivamente. A imagem da ponta experimental é de SHIKIDA, ANDO, <i>et al.</i> , 2004. A última imagem mostra os dois planos que formam o topo da microponta.	113
Figura 49 - Passos de fabricação de um acelerômetro utilizando microfabricação em substrato e os resultados experimentais de uma corrosão.	114
Figura 50 - Exemplo da fabricação de uma micro-lâmpada incandescente através da deposição de um filme metálico sobre um microponta de Si.	116
Figura 51 – Passos para fabricação de uma viga engastada utilizando um processo CMOS. As imagens estão na sequencia da esquerda para a direita e de cima para baixo. O si está em cinza, o óxido em laranja, as vias de tungstênio em verde e as trilhas de alumínio em azul. As máscaras fotolitográficas são representadas em vermelho transparente.	119
Figura 52 – Remoção do óxido para liberação das vigas engastadas. É possível notar nas imagens (a)-(f) como a corrosão do óxido prossegue. Na imagem (a) fica bem claro como a corrosão prossegue tanto na vertical como na horizontal para remover o óxido entre as vias. A imagem (g) mostra o resultado final do processo: as vigas engastadas de diferentes tamanhos liberadas. Resultado experimental para um processo semelhante(DAI, 2006) é mostrado em (h).....	120
Figura 53 – A estrutura analisada está em (a) com as unidades em mm. Na imagem (b) temos a saída da análise para uma força aplicada no ponto C. A imagem (c) mostra o resultado da análise com uma força aplicada no ponto B.	128
Figura 54- Um exemplo de uma estrutura sendo analisada. Em (a) vemos a malha utilizada e os apoios em vermelho. As figuras(b), (c) e (d) mostram a figura deslocada e as tensões normais na direção horizontal e vertical e a tensão de	

cisalhamento, respectivamente. A escala de cores tem vermelho como o valor mínimo e azul como o máximo.....132

Figura 55 – Resultados da análise dinâmica de uma estrutura simples. A estrutura original está em azul. As cores mostram a tensão normal na direção vertical seguindo a escala crescente de vermelho a azul.133

LISTA DE GRÁFICOS

Gráfico 1 – Medidas da taxa de deposição do autômato simples. Há um aumento considerável da taxa de deposição nas direções diagonais.....	84
Gráfico 2 – Taxas de deposição medidas para o autômato para evolução de superfícies (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2005). As taxas são mais próximas das de um crescimento isotrópico quando comparadas aos autômatos com vizinhança de Moore.....	87
Gráfico 3 – Células erradas e o erro relativo para os autômatos celulares simples com vizinhança de Moore e contínuo com a regra da equação (9).....	89
Gráfico 4 - Comparação entre os erros do autômato celular com vizinhança de Moore e o autômato contínuo usando um coeficiente $T = 0,01$	91
Gráfico 5 - Comparação entre utilização de um valor fixo para T e um valor variável para T	92

LISTA DE TABELAS

Tabela 1 - Definição da regra 184.	27
Tabela 2 - A evolução temporal de um estado inicial seguindo a Regra 184.	28
Tabela 3 - Peso, força equivalente e as componentes da força aplicada nos casos usados.	129
Tabela 4 - Comparação entre os resultados do Ftool e do módulo desenvolvido. ...	129
Tabela 5 - Comparação entre os resultados do módulo e resultados experimentais medidos por um interferômetro.	130
Tabela 6 – Medidas e valores adotados para a barra.	130
Tabela 7 - Comparação entre resultados experimentais e do módulo para análise modal.	131

LISTA DE ABREVIATURAS E SIGLAS

CA	Cellular automata
CAD	Computer aided design
CFD	Computational fluid dynamics
DRIE	Deep reactive ion etching
MEF	Método dos elementos finitos
MEMS	Microelectromechanical systems

SUMÁRIO

1	Introdução.....	18
2	Teoria.....	21
2.1	Autômatos Celulares.....	21
2.1.1	Vizinhanças.....	22
2.1.2	Condições de Contorno.....	24
2.2	Exemplo.....	27
2.3	Variações Sobre Autômatos Celulares.....	30
2.4	Modelos para a Evolução de Superfícies.....	35
2.5	Simulação de Processos de Microfabricação Usando Autômatos Celulares.....	39
2.6	Método dos Elementos Finitos.....	47
3	Metodologia.....	56
3.1	Implementação do CellularAutomata3D.....	58
3.1.1	Implementação de um Novo Autômato Celular.....	61
3.2	Implementação do simMEMS.....	62
3.2.1	A Camada Model.....	63
3.2.2	A Camada Core.....	63
3.2.3	A Camada UI.....	64
3.2.4	A Estrutura do Software.....	64
3.2.5	Implementação dos Simuladores Baseados em Autômatos Celulares.....	68
3.3	Comparação dos Autômatos.....	69
4	Resultados e Discussões.....	73
4.1	CellularAutomata3D.....	73
4.1.1	Autômatos Desenvolvidos.....	73
4.1.2	Deposição e Crescimento de Filmes.....	80

4.2	simMEMS.....	98
4.2.1	Os Autômatos Celulares.....	98
4.2.2	Visualização dos Resultados no simMEMS.....	105
4.2.3	Resultados Gerados pelo simMEMS	106
5	Conclusões.....	121
	Apêndice A - Módulo para Cálculo de Esforços pelo MEF	128

1 Introdução

Sistemas micro-eleto-mecânicos (MEMS) são sistemas de dimensões reduzidas fabricados por técnicas de microfabricação derivadas da microeletrônica, onde são integrados dispositivos e/ou circuitos microeletrônicos com elementos móveis (vigas engastadas, diafragmas, engrenagens, etc.). Apesar de serem relativamente recentes, o fato de serem fabricados por técnicas bem estabelecidas e oriundas da tecnologia dos circuitos integrados de Si possibilitou que a tecnologia dos MEMS tivesse uma evolução extraordinária, dando origem a uma indústria florescente, dinâmica e em franca expansão. Por exemplo, nas indústrias automobilística, de acessórios eletro-eletrônicos e acessórios periféricos os MEMS são uma realidade industrial que envolve um mercado bilionário para fabricação de sensores de pressão, temperatura, acelerômetros, cabeçotes para impressão por jato de tinta, microdisplays, etc.(YOLE DÉVELOPPEMENT, 2010).

Devido à complexidade de operação e fabricação desses dispositivos, que rotineiramente exigem várias etapas de microfabricação, é de grande interesse contar com software para o projeto dos MEMS. Com isso em mente, nos últimos anos o Grupo de Novos Materiais e Dispositivos, desenvolveu uma série de trabalhos visando um software para modelagem, simulação e visualização dos processos de fabricação destes sistemas. Foram desenvolvidos dois softwares, o simMEMS e o autoMEMS. O primeiro implementa recursos de CAD (Computer Aided Design) e permite a simulação de processos básicos de microeletrônica e de microfabricação em substrato e em superfície. Já o segundo implementa um simulador atomístico da corrosão anisotrópica do Si baseado em autômatos celulares.

No estágio atual, esses programas implementam diversos módulos de simulação que, integrados num ambiente gráfico de trabalho, permitem fabricar as máscaras fotolitográficas, simular diversas técnicas de microfabricação operando em conjunto, como a deposição e corrosão seletiva de filmes e substratos, e observar os resultados através de sofisticadas ferramentas de visualização.

Os módulos implementados no programa simMEMS são baseados em algoritmos geométricos. Por exemplo, o módulo de simulação da corrosão úmida do Si em KOH (microfabricação em substrato), utiliza um algoritmo de deslocamento de planos para simular a corrosão do substrato através das aberturas em um material de mascaramento. A velocidade com que avança a corrosão em cada direção é lida num arquivo contendo dados experimentais sobre a taxa de corrosão em função do ângulo. Já no programa autoMEMS, que utiliza autômatos celulares, a simulação da corrosão anisotrópica do Si é feita considerando a remoção átomo por átomo do substrato de Si. A remoção ou não de cada átomo é determinada por um conjunto de regras que leva em conta a configuração espacial da vizinhança atômica de cada átomo.

O programa autoMEMS permite trabalhar com substratos de vários formatos e com máscaras de complexidade e geometria arbitrárias, apresentando resultados extremamente promissores e em grande concordância com os resultados experimentais. O simMEMS, por sua vez, possui mais restrições sobre os formatos das aberturas no material de mascaramento bem como sobre as geometrias das superfícies geradas.

Na versão atual o autoMEMS implementa apenas a corrosão anisotrópica úmida do Si, não permitindo a inclusão de outros processos, como a deposição e corrosão de filmes sobre os substratos corroídos, e implementa recursos limitados de visualização das estruturas 3D corroídas quando comparado ao simMEMS. Por isso, era de grande interesse desenvolver novos algoritmos (baseados também em autômatos celulares) para simular a deposição e corrosão de filmes finos, sendo esse o objetivo central deste trabalho.

Os módulos atomísticos de deposição e corrosão de filmes somados à versatilidade da simulação atomística da corrosão anisotrópica de Si do programa autoMEMS e aos recursos gráficos e de CAD do programa simMEMS formam uma ferramenta poderosa para o projeto, a simulação e a visualização de MEMS. A análise dos dispositivos gerados é a única funcionalidade que restaria para permitir o projeto completo de um dispositivo utilizando apenas o simMEMS.

Com isso em mente também foram estudados métodos para análise mecânica de estruturas baseados no método dos elementos finitos. Um software para realizar análises estáticas lineares, análises dinâmicas lineares e a análise modal utilizando

o método dos elementos finitos também foi desenvolvido. O código desse software poderá servir como base para o desenvolvimento futuro de ferramentas de análise das estruturas geradas no simMEMS.

Pelo que foi apresentado acima, o foco do presente projeto foi:

1. Otimizar o módulo de simulação atomística da corrosão anisotrópica úmida do Si e desenvolver novos módulos para simular a corrosão por outros métodos (por exemplo, DRIE),
2. Desenvolver um módulo de simulação atomística (por autômato celular) do crescimento e corrosão (anisotrópica e isotrópica) de filmes,
3. Iniciar o desenvolvimento de um módulo de simulação de esforços mecânicos (por elementos finitos) que permita análises nas estruturas obtidas a partir dos outros módulos de simulação,
4. A integração de todos os módulos num único ambiente de trabalho, que integre recursos para desenho das máscaras fotolitográficas e de visualização, e onde os módulos possam ser utilizados de forma complementar.

Na seção 2 são discutidos aspectos teóricos relativos aos autômatos celulares que são utilizados nos módulos mencionados nos objetivos 1 e 2. Os princípios básicos do método dos elementos finitos, necessários para o objetivo 3, também são abordados. A metodologia aplicada para análise dos autômatos celulares estudados bem como as ferramentas utilizadas para o desenvolvimento dos softwares são discutidos no Apêndice A. Os resultados obtidos pelos softwares CellularAutomata3D e simMEMS são analisados na seção 4. Nessa seção analisamos primeiramente os resultados dos diversos autômatos celulares para simulação da deposição e corrosão de filmes estudados. Em seguida são mostrados os resultados obtidos pela versão final do simMEMS. Alguns resultados do software de análise pelo método dos elementos finitos são apresentados no final desta seção.

2 Teoria

A seguir alguns dos fundamentos básicos necessários para o desenvolvimento desse trabalho serão explicados. Inicialmente os autômatos celulares, que são fundamentais para o nosso projeto, são apresentados. Em seguida apresentam-se modelos utilizados para simular a evolução de superfícies e como isso é aplicado à simulação de deposição e crescimento de filmes. Com esses dois conceitos definidos, alguns autômatos celulares descritos na literatura são estudados. Finalmente uma descrição breve do Método dos Elementos Finitos (MEF) e sua aplicação no software simMEMS é descrita.

2.1 Autômatos Celulares

Autômatos celulares são máquinas imaginárias que têm uma definição matemática exata. Esse tipo de máquina é bastante útil no estudo de problemas como a computabilidade. Uma definição mais rigorosa de um autômato celular pode ser encontrada em(CHOPARD e DROZ, 1998).

Um autômato celular consiste de uma rede regular de células. Essa rede pode cobrir parcialmente ou totalmente um espaço com um número finito de dimensões. Cada célula possui um estado pertencente ao conjunto finito de estados possíveis. A evolução temporal do autômato é dada por um conjunto fixo e finito de regras. As regras determinam o próximo estado, i.e. o estado de uma célula no instante seguinte, analisando uma vizinhança. Uma vizinhança, por sua vez, é um conjunto de células definido em função da célula sob análise. A vizinhança em geral contém a própria célula. O tempo é discreto, i.e. $T=1,2,3,\dots$, e a mudança de estados ocorre simultaneamente para todas as células da rede (CHOPARD e DROZ, 1998)(TOFFOLI e MARGOLUS, 1987).

Pelo descrito acima, para definir um autômato celular é preciso definir, pelo menos:

1. O formato da rede de células;
2. Os estados possíveis;
3. A regra de transição dos estados

Para simular o funcionamento do autômato também é necessário definir uma condição inicial, isto é, definir o estado inicial de cada célula. Nos casos em que a rede é infinita, geralmente considera-se um estado padrão para todas as células e define-se apenas as células que têm um valor diferente. Já para os casos em que o tamanho da rede é finito, é necessário definir, além do estado inicial, uma condição de contorno. Essa necessidade será ilustrada a seguir. Uma condição de contorno define os estados de células não existentes nas bordas da rede(CHOPARD e DROZ, 1998). No item seguinte algumas vizinhanças bastante comuns bem como algumas condições de contorno que também são muito utilizadas são descritas.

2.1.1 Vizinhanças

Uma vizinhança é o conjunto de células que é analisado para definir o próximo estado de uma célula da rede. As vizinhanças são definidas em função da célula cujo próximo estado deseja-se determinar. As vizinhanças, assim como as regras, não variam no tempo(CHOPARD e DROZ, 1998).

Uma notação comum para definição de vizinhanças é utilizar posições geográficas. Por exemplo, a vizinhança de von Neuman é composta pela célula sendo analisada bem como as células vizinhas norte, sul, leste e oeste. Já a vizinhança de Moore é composta pela célula sendo analisada e suas vizinhas norte, nordeste, leste, sudeste, sul, sudoeste, oeste e noroeste. A Figura 1 mostra as vizinhanças de Moore e von Neuman.

Figura 1 – Vizinhanças de Moore e de von Neumann. A célula central está colorida em azul e os seus vizinhos em verde.

As células em azul são as células centrais sendo analisadas. As células em verde são as demais células que fazem parte da vizinhança e as células em branco não fazem parte da vizinhança.

Outra vizinhança que mencionaremos aqui é a de Margolus. Nas vizinhanças de Moore e de von Neumann as vizinhanças são descritas em função de uma célula central. Esse é o padrão para um autômato celular. No caso da vizinhança de Margolus, o espaço é separado em quadrados de 2x2 células. Outro ponto importante é que quando usamos esse tipo de vizinhança, as regras são definidas para o bloco de 2x2 células (TOFFOLI e MARGOLUS, 1987). Uma descrição mais profunda desse tipo de regra em bloco pode ser encontrada em TOFFOLI e MARGOLUS, 1987. A vizinhança de uma célula é composta pelas células no bloco ao qual a célula pertence. Para transmitir informações entre os quadrados, a separação é alternada nos passos pares e ímpares. A Figura 2 mostra a vizinhança de Margolus.

Figura 2 – Exemplo da vizinhança de Margolus. Nos passos pares (a) e (b) as células verdes são as vizinhas da célula azul e as laranja vizinhas da célula lilás. Nos passos ímpares (c) a célula azul tem como vizinhas as células amarelas e a lilás.

Considerando a célula azul na figura. A vizinhança nos passos pares ($T=0, 2, 4, \dots$) da célula azul é formada pela mesma e as células verdes (Figura 2(a)). Já nos passos ímpares ($T=1, 3, 5, \dots$) a vizinhança da célula azul contém, além da célula azul, as células amarelas e a lilás (Figura 2(c)). No caso da célula lilás, a vizinhança nos passos pares inclui a célula lilás e as células laranja como vizinhas nos passos pares (Figura 2(b)) e as células amarelas e a azul nos passos ímpares (Figura 2(c)).

Essa vizinhança é interessante porque mostra uma técnica útil. Vizinhanças são definidas em função da célula cujo próximo estado queremos definir e não mudam em função do tempo ou da posição da célula. A vizinhança de Margolus, como mostrada acima, faz justamente isso. Para tanto, aumenta-se a quantidade de informação presente no estado da célula para permitir essas variações. Por exemplo, armazenando um bit para representar a paridade do estado permite alterar a vizinhança em função da paridade. De maneira semelhante é possível armazenar no estado da célula a sua posição relativa dentro do bloco. As regras do autômato são modificadas para alterar apropriadamente os valores no estado da célula. Dessa maneira podemos criar autômatos celulares que aparentam não seguir as definições da seção 2.1. Como existe a possibilidade de transformar um autômato que utiliza a vizinhança de Margolus ou efeitos semelhantes (variação das regras ou da vizinhança no tempo ou no espaço), por simplicidade autômatos com essas propriedades são também considerados como autômatos celulares.

2.1.2 Condições de Contorno

Quando o autômato celular opera sobre uma rede finita, podem existir células para as quais a vizinhança definida inclui células que não fazem parte da rede. Por exemplo, imaginando um autômato celular onde a vizinhança de uma célula contém a célula e a célula à sua esquerda. Se a rede for finita, eventualmente encontraremos uma célula, no exemplo a célula mais à esquerda, onde a vizinhança inclui uma célula que não existe. Nesses casos é preciso definir um comportamento para o autômato celular.

Uma maneira de resolver esse problema é armazenar dados a mais sobre a posição das células no estado das células e definir regras especiais para células na borda. Nesse caso existiriam bits a mais no estado para indicar que uma célula está na borda da rede. As regras do autômato seriam mais complexas para levar em conta a existência desses bits e aplicar regras que não usam o estado de células inexistentes para determinar o próximo estado. Outra opção, mais simples, é determinar um comportamento padrão para tratar essas situações. Esse comportamento é chamado de condição de contorno (CHOPARD e DROZ, 1998). Existem alguns tipos de condições de contorno. Condições de contorno que foram utilizados no projeto são descritas a seguir.

Uma condição de contorno periódica faz uma espécie de volta na rede. Em uma dimensão é o equivalente a transformar uma série de células em um círculo. Em duas dimensões um quadrado seria transformado em um toro. Assim, como mostra a Figura 3(b) a célula à esquerda de uma célula na borda esquerda da rede é a célula na mesma linha e na borda direita da rede. Uma lógica análoga é aplicada nas demais direções. Essa condição é equivalente a dizer que as células que estão sendo analisadas se repetem pelo espaço não representado.

Figura 3 – Condição de contorno aplicada a uma rede bidimensional. Em (a) a vizinhança definida para o autômato celular é mostrada. Em (b) a mesma vizinhança é mostrada para uma célula na borda utilizando a condição de contorno .

Já uma condição de contorno fixa define um mesmo valor para todos os vizinhos não definidos. Dessa maneira existem infinitas células virtuais cujo estado não precisa ser armazenado pois nunca muda. Essas células virtuais preenchem o restante do espaço da rede. Essa condição é equivalente a dizer que o conjunto de células está

imerso em um meio que não se altera. Por exemplo, ao simular uma deposição podemos considerar que todas as células virtuais estão no estado correspondente ao vácuo ao redor do substrato. A Figura 4 mostra uma pequena rede de doze células.

Figura 4 – A figura mostra as células da rede com borda completa e o estado em preto. As células virtuais têm bordas tracejadas e o estado em cinza. Uma vizinhança para uma célula com todos os vizinhos reais é mostrada em verde e azul na imagem (a). A mesma vizinhança é mostrada para uma célula da borda da rede em (b).

Na Figura 4 as células da rede têm seus estados em preto e uma borda contínua. As células virtuais têm seus estados em cinza e a borda tracejada. Na Figura 4(a) pode-se ver a vizinhança para uma célula real cuja vizinhança não inclui células virtuais. A Figura 4(b) mostra a vizinhança para uma célula na borda da rede, que requer as células virtuais. Como as células virtuais não mudam de estado, não é preciso aplicar regra alguma sobre elas.

Existem outras condições de contorno, que não foram utilizadas nos autômatos celulares descritos aqui. Em CHOPARD e DROZ, 1998, há uma descrição breve de outros tipos de condições de contorno.

2.2 Exemplo

A seguir o funcionamento de um autômato celular unidimensional simples conhecido como Regra 184 no código de Wolfram (WOLFRAM, 2002) é exemplificado. Essa regra está definida na Tabela 1. O nome da regra vem do valor, na base binária, correspondente aos próximos definidos pela regra ($10111000_2 = 184_{10}$). Para determinar os estados das células em um instante $T+\Delta t$ qualquer analisa-se, para cada célula, o seu estado e o estado das células diretamente à sua esquerda e à sua direita no instante T . Esses valores formam um conjunto de três estados que indica qual entrada na tabela de regras indica o próximo estado da célula sendo analisada.

Tabela 1 - Definição da regra 184.

	Regra 184							
Configuração	111	110	101	100	011	010	001	000
Próximo Estado	1	0	1	1	1	0	0	0

Pelo processo descrito é possível dizer que a vizinhança nesse autômato celular é composta por três células: a célula sendo analisada e as células que a precedem e a sucedem. Para simular a execução desse autômato em uma rede finita também é preciso definir uma condição de contorno. Nesse exemplo uma condição de contorno periódica é utilizada. A Tabela 2 apresenta, para os instantes T , $T+\Delta t$ e $T+2\Delta t$, o estado das células do autômato. Para facilitar a explicação as células foram numeradas de 0-9. O número de cada célula se encontra na primeira linha da tabela.

Tabela 2 - A evolução temporal de um estado inicial seguindo a Regra 184.

	0	1	2	3	4	5	6	7	8	9
T	0	0	1	0	1	1	1	0	1	0
T+Δt	0	0	0	1	1	1	0	1	0	1
T+2Δt	1	0	0	1	1	0	1	0	1	0

Para a célula 1 a evolução do instante T para o instante T+ Δt ocorre da seguinte maneira. Primeiramente, montamos a tripla dos estados da célula e suas vizinhas. O estado da célula 1 em T é "0". A célula 0 à sua esquerda está no estado "0". A célula 2 à sua direita está no estado "1". Esses valores formam então a configuração "001". Pela Tabela 1 o próximo estado para uma célula com essa configuração é "0". Pode-se verificar na Tabela 2 que o estado da célula 1 no instante T+ Δt é "0". Seguindo o mesmo raciocínio para as células 2 (configuração 010), 3 (configuração 101), 4 (configuração 011), 5 (configuração 111), 6 (configuração 110), 7 (configuração 101) e 8 (configuração 010). As células 0 e 9 são casos especiais pois é preciso aplicar a condição de contorno.

A célula 0 não possui uma célula à sua esquerda e a célula 9 não possui uma célula à sua direita. A condição de contorno periódica impõe que a célula à esquerda da célula 0 é a célula 9 e que a célula à direita da célula 9 é a célula 0. Assim é possível determinar o estado dessas células no instante T+ Δt . A célula 0 se encontra no estado "0". A célula 9 à sua esquerda se encontra no estado "0". A célula 1 à sua direita se encontra no estado "0". Forma-se assim a configuração 000. De acordo com a Tabela 1, o estado da célula 0 no instante T+ Δt será então "0". Analogamente para a célula 9, sua configuração é "100" e o próximo estado é "1".

O passo final para a simulação de autômato celular é a interpretação dos resultados. No caso da Regra 184, esse passo depende do que estamos simulando. Caso o objetivo seja apenas analisar o comportamento do autômato celular, não é preciso fazer uma interpretação dos resultados. Mas essa regra também pode ser utilizada para simular trânsito de veículos(WANG, KWONG e HUI, 1998), a deposição balística de partículas sobre uma superfície(KRUG e SPOHN, 1988) ou a

aniquilação balística de partículas(FUKS e BOCCARA, 2001), como a que ocorre quando um elétron e um pósitron colidem.

No caso de uma simulação de trânsito, consideramos que células correspondem ao espaço de uma rua. Uma célula no estado "0" representa um espaço vazio na rua e uma célula no estado "1" representa um espaço ocupado por uma carro. A Figura 5 apresenta o resultado da simulação considerando a simulação de trânsito. Note que o estado inicial é o mesmo da Tabela 2.

Figura 5 - Evolução temporal para da Regra 184 considerando a simulação de trânsito. Um carro foi colorido em roxo para auxiliar na noção de movimento.

A interpretação dos resultados quando se simula a deposição balística de partículas é bastante diferente. Nesse caso, os estados das células não representam a ocupação do espaço, mas a inclinação de uma superfície. O estado "0" agora representa uma inclinação positiva enquanto o estado "1" representa uma inclinação negativa. A deposição de partículas ocorre somente nos pontos de mínimo. Os pontos de mínimo são os pontos onde passamos de uma inclinação negativa para uma inclinação positiva. Essa regra é capaz de encontrar e atualizar corretamente esses pontos. A Figura 6 mostra a evolução da superfície seguindo esta regra para o mesmo estado inicial da Tabela 2.

Figura 6 - Evolução de uma superfície devido à deposição balística simulada pela Regra 184.

Na Figura 6 temos o estado inicial representando a inclinação da superfície. A deposição nesse caso ocorre apenas nos pontos de mínimo. As células laranja mostram as partículas que foram depositadas. Pode-se verificar que o estado no instante $T+\Delta t$ na Tabela 2 descreve a inclinação da nova superfície.

2.3 Variações Sobre Autômatos Celulares

No item 2.1.1 foi apresentada a vizinhança de Margolus. Essa vizinhança apresenta variações temporais e espaciais. Também foi apresentada uma explicação breve de porque autômatos que usam essa vizinhança também são considerados autômatos celulares. Isso apesar da definição de autômato celular fornecida explicitamente proibir tais variações. Nessa seção serão apresentadas diversas variações que existem para autômatos celulares e as suas vantagens ou desvantagens. Duas dessas mudanças geram dois novos tipos de autômatos: os autômatos celulares contínuos e os autômatos celulares probabilísticos.

No que tange à implementação de um autômato celular para usos práticos numa simulação, às vezes é preciso que cada célula tenha acesso a um mesmo dado. Por exemplo, numa simulação de corrosão de Si, talvez toda célula precise saber a temperatura ambiente. A vizinhança de Margolus requer o conhecimento da paridade do instante atual. Por simplicidade e melhor desempenho, essa informação pode ser armazenada como uma variável global. Isso, no entanto, não significa que o nosso modelo deixa de ser equivalente a um autômato celular. Esse tipo de modificação tem sido muito utilizado e por isso mencionamos aqui.

Existem modificações que efetivamente criam um novo tipo de autômato. Adicionar variáveis aleatórias às regras é uma dessas modificações. Autômatos celulares que utilizam variáveis aleatórias na suas regras são conhecidos como autômatos celulares probabilísticos. Cabe mencionar aqui que muitas vezes um mesmo tipo de autômato celular tem nomes diferentes. Utilizamos aqui o nome que é mais comum na área de simulações para microeletrônica.

Autômatos celulares probabilísticos exibem um comportamento às vezes muito mais rico que um autômato semelhante determinístico. No entanto possuem algumas desvantagens, como o fato de não serem reversíveis. Como existem vários possíveis próximos estados para um mesmo estado atual, não é possível inverter as regras. Isso impede partir de um resultado desejado e descobrir a condição inicial que gera o resultado. Outra desvantagem desse tipo de autômato é a presença de ruído. Dadas as mesmas entradas e o mesmo autômato, os resultados variam bastante. Nesse caso é interessante analisar a média dos resultados. Para obter esse resultado médio de uma simulação, que é o que em geral desejamos, é necessário rodar o autômato diversas vezes e tirar a média dos resultados. Isso é, por sua vez, requer mais tempo de processamento.

Um exemplo de autômato celular probabilístico interessante é aquele que simula a movimentação aleatória de partículas em uma dimensão. Neste autômato uma partícula pode se movimentar para esquerda ou para direita. Uma partícula indo para direita continua se movendo para direita e vice-versa até que colida com outra partícula indo na direção oposta. Todas as partículas são idênticas, então em caso de colisão não há como diferenciar as partículas, portanto não há como ver que as partículas colidiram e, de fato, as partículas se comportam como se fossem efetivamente “invisíveis” uma para a outra. Por isso mesmo, o autômato celular como descrito não é muito interessante, mas podemos adicionar uma variável aleatória p que define a probabilidade de uma partícula manter sua direção, como na Figura 7.

Figura 7 – Uma partícula indo em direção à direita tem uma probabilidade ρ manter sua direção e continuar para a direita e $1 - \rho$ de mudar de direção e passar a se movimentar para a esquerda.

Uma maneira de interpretar esse novo autômato é considerar que as células representam um substrato de silício e as partículas átomos de dopantes. A variável aleatória ρ passa então a definir a probabilidade de uma partícula desse dopante mudar de direção devido às colisões com átomos do substrato. Assim, esse autômato pode simular a difusão de dopantes em uma dimensão. O coeficiente de difusão D da Lei de Fick para difusão é dado por:

$$D = \frac{\rho}{2(1 - \rho)} \frac{\lambda^2}{\tau} \quad (1)$$

onde λ é o tamanho da célula e τ é a duração de um passo de tempo. Em (CHOPARD e DROZ, 1998) os autores demonstram como chegar a esse valor. Talvez mais interessante seja o fato de que esse valor para o coeficiente D é obtido partindo das regras do autômato celular e determinando a equação que descreve o comportamento macroscópico desse autômato celular. É possível então verificar que essa equação, que descreve o comportamento macroscópico do autômato celular probabilístico, é uma solução para a Lei de Fick. Um exemplo semelhante a esse autômato celular é também utilizado na física estatística para descrever a difusão em uma dimensão (SALINAS, 1999).

Para estudar este tipo de autômato probabilístico o processo de difusão descrito acima esse autômato celular foi implementado e o resultado de uma dessas simulações é mostrado na Figura 8.

Figura 8- Simulação de um processo de difusão. (a) Condição inicial. (b) Resultado após 2000 passos com $\rho = 0,1$. (c) Resultado após 2000 passos com $\rho = 0,6$.

Inicialmente temos uma concentração grande de partículas no centro (região branca da Figura 8a) e nenhuma partícula nas bordas. A concentração de partículas está representada em escala de cinza. A cor branca representa uma probabilidade de 1,0 de encontrar uma partícula na posição analisada e a cor preta representa uma probabilidade de 0,0. Após 2000 passos do autômato, nota-se claramente que as partículas se movimentam do centro, onde a concentração é grande, para as bordas, onde a concentração é pequena. Assim, o fluxo vai dos locais com alta concentração para os locais de baixa concentração.

Considerando os mesmos valores, que devem ser pequenos, para λ e τ , o coeficiente de difusão D é maior para o caso em que $\rho = 0,6$ do que para o caso em que $\rho = 0,1$. Consequentemente espera-se que no primeiro caso a substância branca demore mais tempo para se difundir no meio do que no segundo caso, o que efetivamente ocorre nas simulações. Na Figura 8(b) é possível notar claramente o ruído que mencionamos anteriormente; não há uma transição contínua de preto para cinza, havendo variações pequenas locais. Esse ruído é menos perceptível, mas ainda presente na Figura 8(c).

A simulação da corrosão anisotrópica do Si em soluções de KOH é outra aplicação desse tipo de autômato (DE OLIVEIRA JR., 2008). Nesse caso usamos uma variável aleatória para definir a probabilidade de um átomo ser removido. Essa probabilidade depende da vizinhança do átomo sendo analisado. Esse método é interessante porque é bastante próximo do processo real de corrosão. Ele também permite variar a taxa de corrosão em função da concentração da solução de KOH ou da temperatura em que ocorre a corrosão. No entanto, como a simulação em geral opera sobre poucos átomos, quando comparado ao número de átomos em uma lâmina de Si, o ruído também pode ser problemático (ver Figura 9(c)).

Figura 9 - Micro pontas de silício. Os resultados são: a) experimental; b) simulado com autômato convencional em 140 iterações; c) simulado com autômato estocástico em 125 iterações e usando as taxas padrões; d) simulado com autômato contínuo com a regra 3334 alterada para o valor máximo e em 280 iterações (DE OLIVEIRA JR., 2008).

A última modificação apresentada aqui é bastante utilizada tanto na simulação da corrosão úmida de Si como na simulação da deposição e corrosão de filmes. Até agora todas as células dos autômatos analisados sempre tiveram um estado discreto, até mesmo quando variáveis aleatórias são utilizadas. Ou o átomo está na rede do substrato ou não está. A partícula ou vai para esquerda, ou para a direita. Existem autômatos que não possuem estados discretos; o estado pode variar continuamente entre dois valores. Esses autômatos são conhecidos como autômatos celulares contínuos (ZHU e LIU, 2000).

Existem diversas variações de autômatos celulares contínuos que simulam o processo de corrosão úmida de Si (GOSÁLVEZ, XING, *et al.*, 2008). Nesses autômatos, as células deixam de ter estados discretos, representando a presença ou não de um átomo de Si, e passam a ter um estado contínuo que representa a “porcentagem da massa” de um sítio. Um sítio no interior do substrato está 100% presente e tem um estado de 1,0. Já um sítio na superfície pode ter apenas 35% da sua massa presente, devido à corrosão, e ter um estado de 0,35. Apesar das células ainda terem o formato da rede cristalina do Si, com cada sítio tendo a posição de um

átomo, não faz mais sentido fisicamente considerar essas células átomos. Este tipo de autômato permite melhores resultados e uma maior flexibilidade que um autômato celular comum (GOSÁLVEZ, XING, *et al.*, 2008) ou que um autômato celular probabilístico (GOSÁLVEZ, FOSTER e NIEMINEN, 2002). Autômatos contínuos também são utilizados em outros campos. Um dos métodos para Computational Fluid Dynamics (CFD) é o método Lattice-Boltzman, que é um autômato celular contínuo (CHOPARD e DROZ, 1998).

A Figura 9(d) acima também mostra resultados obtidos com um simulador atomístico contínuo, como o mencionado acima. O formato da micropona gerado por esse autômato é semelhante ao do autômato celular probabilístico. Isso é desejável nesse caso. No entanto também fica claro que o autômato contínuo não sofre com o ruído presente nos resultados do autômato celular probabilístico.

2.4 Modelos para a Evolução de Superfícies

O objetivo desse software é simular a deposição e corrosão de filmes por diversos processos. Embora seja interessante simular diversas características de um filme depositado, o simMEMS permite simular apenas o formato do filme. Para tanto é preciso simular como a superfície do filme evolve ao longo da deposição ou corrosão, pois a superfície determina o formato do filme.

O problema da evolução de uma superfície pode ser descrito matematicamente da seguinte maneira: sejam $v(x, y, z)$ um campo de velocidade e ζ_1 uma superfície que separa o espaço em duas regiões, a região 1 e a região 2, no instante t_1 . O problema da evolução de uma superfície se resume em determinar no instante t_2 , dados v e ζ_1 , o formato da superfície ζ_2 que agora separa o espaço nas regiões 1 e 2. A evolução de superfícies é um problema que surge em diversos campos, como a ótica geométrica.

Uma abordagem comum consiste em separar a superfície em pedaços pequenos, até eventualmente pontos. O problema, nesse caso, se reduz a determinar a

trajetória de um ponto no espaço. Outra possibilidade é simular a evolução da superfície inteira, usando o princípio de Huygens.

Abordagens celulares utilizam o princípio de Huygens para simular a evolução de superfícies. O princípio de Huygens, no contexto da óptica geométrica, considera que cada ponto na frente de onda é na verdade a fonte de um novo trem de ondas. A nova frente de onda é dada pelo envelope desses novos trens de onda. A Figura 10 abaixo mostra como é possível analisar a refração de uma onda baseado nesse princípio.

Figura 10 – Refração de uma onda determinada pelo princípio de Huygens (NORDMANN, 2007).

Na Figura 10 os pontos amarelos são fontes de novos trens de ondas em cinza. As linhas verdes são parte dos envelopes formados pelas ondas em cinza. A onda muda de direção porque a velocidade de expansão das ondas no meio inferior é menor que no meio superior.

No contexto de processos de corrosão e de crescimento de filmes considera-se que cada ponto da superfície atual é na verdade uma fonte de matéria. A nova superfície será dada pelo envelope de todas as ondas de matéria (STRASSER e SELBERHERR, 1995), como mostrado na Figura 11.

Figura 11 - Crescimento isotrópico de uma superfície através do uso do princípio de Huygens. Na figura (a) temos a superfície original. A figura (b) mostra as ondas de matéria gerada por diversos pontos. A figura (c) mostra a demarcação do envelope para as ondas geradas por todos os pontos da superfície. A figura (d) mostra a superfície final. O detalhe mais escuro mostra quanto a superfície cresceu.

A Figura 11(a) retrata, em cinza, a superfície do substrato sobre o qual um filme será depositado. A simulação do crescimento de um filme sobre a superfície é feita considerando que todos os pontos da superfície geram uma onda de matéria. A Figura 11(b) mostra, em preto, a posição de ondas de matérias geradas por diversos pontos durante um intervalo de tempo. As ondas são circulares, representando assim um crescimento isotrópico. A Figura 11(c) mostra, em laranja, o formato do envelope de todas as ondas geradas por todos os infinitos pontos da superfície. Esse envelope determina o novo formato da superfície. A Figura 11(d) contém o formato final do filme, em cinza escuro, depositado sobre o substrato, em cinza.

Como esse método pode ser utilizado para simular a evolução de uma superfície qualquer, pode-se simular tanto a deposição quanto o crescimento de um filme. A simulação de uma corrosão é idêntica exceto pela direção do movimento. Enquanto na deposição o envelope é formado de maneira a aumentar a área do filme, na corrosão o envelope é gerado de forma a diminuir a área do filme. Em outras palavras, o envelope é gerado considerando que é o "vazio" que está crescendo, e não mais o filme.

Nas Figura 11(b) e (c) a onda de matéria é representada por um círculo. Utiliza-se um círculo quando o objetivo é simular um crescimento isotrópico. Processos CVD, por exemplo, podem ser simulados utilizando esse formato de onda. Na simulação

de um crescimento anisotrópico utiliza-se outras formas, como uma elipse. Um processo PVD, onde a deposição nas laterais de degraus é mais lenta, é um exemplo de uma situação onde uma onda em forma de elipse é mais apropriada. A Figura 12 mostra o exemplo de uma corrosão anisotrópica usando uma elipse.

Figura 12 - Corrosão anisotrópica de uma superfície através do uso do princípio de Huygens. Na figura (a) temos a superfície original em cinza. A figura (b) mostra as ondas elípticas de matéria gerada por diversos pontos dessa superfície. A figura (c) mostra a demarcação do envelope, em laranja, para as ondas geradas por todos os pontos da superfície. A figura (d) mostra a superfície corroída final e a região removida em um tom mais claro.

Diferentemente da Figura 11, agora o tom de cinza mais claro na Figura 12(d) demarca a região do substrato que foi removida. Observe, também na Figura 12(d), que a lateral vertical do degrau foi corroída mais lentamente que os degraus horizontais. Existem também casos onde o crescimento é isotrópico, mas existem regiões onde o crescimento é mais lento ou mais rápido. Isto é, dada uma pequena região qualquer do substrato o crescimento é isotrópico. No entanto analisando toda a superfície o crescimento é anisotrópico. Para simular este tipo de crescimento pode-se aplicar o mesmo conceito mas utilizar círculos de raios diferentes (STRASSER e SELBERHERR, 1995). Esse método pode ser facilmente transferido para três dimensões. Nesse caso esferas ou elipsóides são utilizados como as frentes de ondas, ao invés de círculos e elipses (STRASSER e SELBERHERR, 1995).

2.5 Simulação de Processos de Microfabricação Usando Autômatos Celulares

A corrosão anisotrópica úmida de substratos de silício cristalino é um dos processos mais importantes na fabricação de MEMS. As estruturas geradas por este tipo de corrosão podem ser de geometria bastante complexa em função da cristalografia do Si e dos padrões utilizados no início da corrosão. Podem ainda evoluir temporalmente de forma bastante difícil de prever. Isso gerou um grande interesse em simular este tipo de corrosão. Diversos métodos foram sugeridos. Podemos, de modo geral, separar estes modelos em modelos geométricos e modelos atomísticos.

A simulação da corrosão anisotrópica úmida do Si, como a deposição e corrosão de filmes, se resume ao problema de determinar a evolução de uma superfície. Novamente esse problema pode ser resolvido simplificando a superfície e transformando o problema em definir a trajetória de um ponto em um campo de velocidades. Essa abordagem gera os métodos geométricos de simulação da corrosão úmida do Si.

A abordagem atomística da simulação da corrosão anisotrópica do Si é até certo ponto semelhante à abordagem celular mencionada para a simulação da corrosão e deposição de filmes. Contudo os algoritmos atomísticos para corrosão do Si em geral são baseados nas reações envolvidas durante o processo de corrosão bem como a estrutura cristalina do silício, e não no método de Huygens.

Métodos geométricos sofrem de algumas dificuldades. No caso da corrosão do Si, é preciso ter medidas da taxa de corrosão para um grande número de planos. De modo geral quanto maior o número de planos, melhor o resultado da simulação. Essas medidas, por sua vez, variam bastante de um ambiente para outro. Embora possa ser possível, e comum, modelar a variação das taxas de corrosão com variáveis como temperatura e concentração, é muito mais difícil e incomum levar em conta variáveis como o tamanho e o formato do recipiente onde a corrosão ocorre. Outra dificuldade comum aos algoritmos geométricos, tanto para deposição e corrosão de filmes como para a corrosão úmida do Si, é a dificuldade de representar a superfície enquanto ela evolui. Por exemplo, é comum que a superfície

eventualmente cruze com ela mesma. Isso requer um tratamento especial para evitar erros. Outro exemplo diz respeito a como tratar cantos convexos que se expandem. Existem soluções para esses problemas mas os algoritmos geométricos acabam por se tornar complexos simplesmente pela dificuldade em atualizar a estrutura de dados que representa a superfície.

Modelos atomísticos consideram as reações químicas que ocorrem durante a corrosão para simular a corrosão através da remoção do material átomo por átomo. Apesar de em geral necessitar de mais recursos e tempo de processamento, esse método pode apresentar resultados muito melhores com algoritmos mais simples. No caso dos modelos baseados em autômatos celulares, os algoritmos são em geral simples e não têm limitações quanto ao formato das máscaras ou aos planos possíveis. Esses métodos têm obtido sucesso, e atualmente são utilizados em softwares de simulação comerciais (GOSÁLVEZ, XING, *et al.*, 2008)(GOSÁLVEZ. e XING, 2008). Isso se deve tanto a não apresentar problemas na representação da superfície como por diminuir o número de medidas das taxas de corrosão de diversos planos. No caso dos algoritmos atomísticos, é possível determinar a taxa de remoção de diversas configurações atomísticas a partir da medida da taxa de corrosão de uns poucos planos (GOSÁLVEZ, XING e SATO, 2008). Um autômato celular que simula a corrosão anisotrópica do Si foi implementado em conjunto com diversos algoritmos para simulação da corrosão e deposição de filmes. Uma descrição mais extensa desse autômato celular será apresentada no item 4.2.1.1.

Por razões semelhantes ao do problema simulação da corrosão anisotrópica úmida do Si, a abordagem celular também é interessante para a simulação da corrosão e da deposição de filmes. A utilização dessa abordagem celular para todos os processos também permite um casamento melhor entre a simulação de ambos os processos. A seguir são descritos de maneira rápida alguns algoritmos celulares sugeridos na literatura.

Os autores de STRASSER e SELBERHERR, 1995 sugerem aplicar o princípio de Huygens diretamente, usando vizinhanças grandes para melhorar os resultados. Nesse caso o problema é simplificado considerando que apenas o centro de cada célula é uma fonte de ondas. O algoritmo descrito no artigo não é propriamente um autômato celular mas é trivial transformar esse algoritmo em um autômato celular.

No caso de uma corrosão, o algoritmo consiste em considerar, para toda célula vazia, uma vizinhança que contém todas as células dentro de certo raio. A cada passo, todas as células que podem ser corroídas dentro desse raio são marcadas para remoção. Após todas as células serem analisadas e marcadas apropriadamente, o estado das células marcadas é atualizado para representar uma célula vazia.

O resultado dessa simulação será próximo do envelope desejado. Como os pontos de origem são discretos, o envelope pode conter falhas. No entanto se o número de células for grande o suficiente, essas falhas podem ser desprezíveis. A Figura 13 exemplifica esse método para uma corrosão do material "1".

Figura 13 – Evolução de uma simulação de corrosão do material "1". Na figura "0" indica uma célula vazia e "1" e "2" indicam células do substrato.

O número dentro da célula identifica o material que ocupa aquele espaço. Os discos em vermelho indicam o raio que consideramos para simular a corrosão. As células com fundo verde na Figura 13(a) são células que serão marcadas para remoção. Na Figura 13(b) todas as células que foram removidas na simulação do último passo de tempo têm o fundo verde claro.

No caso acima, o raio utilizado é de 2,5 vezes o comprimento do lado de uma célula. Esse raio não é fixo e é determinado pelo produto da taxa de corrosão pela duração do passo de tempo. Por exemplo, na Figura 13 pode-se considerar uma taxa de corrosão de 5 $\mu\text{m}/\text{minuto}$ e um passo de tempo de 0,5 minuto, dando o raio seria de 2,5 μm . O comprimento do lado de uma célula nesse caso é 1 μm . Isso leva ao raio visto na Figura 13(a). Pela definição do raio utilizado, fica claro que após meio minuto de corrosão, as células a até 2,5 μm da superfície atual devam ser

removidas. Assim, no próximo passo de tempo, o estado de todas as células em verde será mudado para “0”, que nesse caso representa células vazias. As células com valor “1” e “2” representam algum tipo de material, por exemplo SiO_2 e Si. No caso, as células do tipo “1” são de um material que pode ser corroído.

Esse método pode ser bastante exigente computacionalmente. Vizinhanças muito grandes são desejáveis neste caso porque diminuem erros devido ao uso de uma rede finita. No entanto quanto maior a vizinhança, maior será o custo computacional para cada passo e menor será a resolução temporal. É desejável ter um autômato celular capaz de obter resultados tão bons ou melhores utilizando uma vizinhança menor.

Vários pesquisadores têm publicado trabalhos mais recentes tentando criar um autômato celular com estas características. Um trabalho bastante interessante apresenta três autômatos para simular três processos básicos na fabricação de MEMS: corrosão de fotorresiste, oxidação térmica e deposição de filmes finos (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2002). Todos os autômatos descritos trabalham em redes bi-dimensionais. Além disso, esses autômatos usam vizinhanças pequenas, especificamente a vizinhança de Moore. Como essa característica é desejável, autômatos celulares com essas características foram estudados.

O processo de fotolitografia é muito importante tanto para a fabricação de MEMS como de circuitos integrados. Por isso foi estudado o autômato celular para simulação da corrosão de fotorresiste. Existe também uma expansão desse mesmo algoritmo para uma rede tridimensional (KARAFYLLIDIS, 1999). O autômato utiliza a vizinhança de Moore. Cada célula do fotorresiste armazena dois valores: uma taxa de corrosão e um estado. O estado indica quão corroída a célula está. Um valor de 0,0 indica que a célula está totalmente preenchida pelo fotorresiste enquanto que um valor de 1,0 indica que a célula está totalmente corroída. A taxa de corrosão indica o quão rápido a célula é corroída. A evolução do estado de uma célula é dada pela seguinte fórmula:

$$C_{i,j}^{t+dt} = C_{i,j}^t + F_1(C_{i-1,j}^t, C_{i+1,j}^t, C_{i,j-1}^t, C_{i,j+1}^t) + F_2(C_{i-1,j+1}^t, C_{i-1,j-1}^t, C_{i+1,j-1}^t, C_{i+1,j+1}^t) \quad (2)$$

onde o $C_{i,j}^t$ representa o estado da célula na posição (i,j) no instante t , $C_{i,j}^{t+dt}$ representa o estado da célula na posição (i,j) no instante $t + dt$ e dt é a duração de um passo de tempo. A função F_1 define como as células adjacentes (n , s , e , w) alteram o estado da célula central e a função F_2 define como as células diagonais (ne , nw , se , sw) alteram o estado da célula central. Essa mesma equação é utilizada para os demais autômatos celulares, apenas modificando as funções F_1 e F_2 . As equações abaixo mostram as funções utilizadas pelo autômato celular para corrosão do fotorresiste. As origens das funções reproduzidas abaixo, são descritas em mais detalhes em (KARAFYLLIDIS e THANAILAKIS, 1995).

$$F_1 = \left(\frac{R_{i,j}t}{a} - \frac{R_{i,j}(1 - C_{i-1,j}^t)}{R_{i-1,j}} \right) + \left(\frac{R_{i,j}t}{a} - \frac{R_{i,j}(1 - C_{i,j+1}^t)}{R_{i,j+1}} \right) + \left(\frac{R_{i,j}t}{a} - \frac{R_{i,j}(1 - C_{i+1,j}^t)}{R_{i+1,j}} \right) + \left(\frac{R_{i,j}t}{a} - \frac{R_{i,j}(1 - C_{i,j-1}^t)}{R_{i,j-1}} \right) - \left(\frac{R_{i,j}^2(t - a(1 - C_{i-1,j}^t)/R_{i-1,j})(t - a(1 - C_{i,j+1}^t)/R_{i,j+1})}{a^2} \right) - \left(\frac{R_{i,j}^2(t - a(1 - C_{i-1,j}^t)/R_{i-1,j})(t - a(1 - C_{i,j-1}^t)/R_{i,j-1})}{a^2} \right) - \left(\frac{R_{i,j}^2(t - a(1 - C_{i+1,j}^t)/R_{i+1,j})(t - a(1 - C_{i,j+1}^t)/R_{i,j+1})}{a^2} \right) - \left(\frac{R_{i,j}^2(t - a(1 - C_{i+1,j}^t)/R_{i+1,j})(t - a(1 - C_{i,j-1}^t)/R_{i,j-1})}{a^2} \right) \quad (3)$$

$$\begin{aligned}
F_2 = D \left[\frac{\frac{1}{2} R_{i,j}^2 (t - a(1 - C_{i-1,j-1}^t)/R_{i-1,j-1})^2}{a^2} \right. \\
+ \left. \frac{\frac{1}{2} R_{i,j}^2 (t - a(1 - C_{i-1,j+1}^t)/R_{i-1,j+1})^2}{a^2} \right] \\
+ \left[\frac{\frac{1}{2} R_{i,j}^2 (t - a(1 - C_{i+1,j-1}^t)/R_{i+1,j-1})^2}{a^2} \right. \\
+ \left. \frac{\frac{1}{2} R_{i,j}^2 (t - a(1 - C_{i+1,j+1}^t)/R_{i+1,j+1})^2}{a^2} \right] \quad (4)
\end{aligned}$$

Nas equações acima, $R_{i,j}$ é a taxa de corrosão da célula na posição (i,j) , a é o comprimento do lado da célula e D é uma constante determinada experimentalmente. O valor ótimo para D definido pelos autores é 0,18. Esse valor foi determinado variando D e comparando o resultado da simulação com uma frente circular.

A taxa de corrosão $R_{i,j}$ das células, mencionada acima, pode ser definida utilizando diversas fórmulas. Isso permite simular ou não vários efeitos. É possível, por exemplo, simular efeitos como o espalhamento de um feixe de luz UV ou efeitos de onda estacionária devido à reflexão da luz pelo substrato. Em (KARAFYLLIDIS e THANAILAKIS, 1995) os autores discutem algumas opções.

O simulador de oxidação térmica é razoavelmente simples e também emprega um autômato celular com a vizinhança de Moore. Cada célula possui um estado que representa quão oxidada uma célula está; um estado de 1,0 representa uma célula completamente oxidada e um estado de 0,0 representa uma célula completamente não-oxidada. Também existe um *flag* para representar se a célula está dentro ou fora do substrato de Si. O próximo estado de uma célula é dado por:

$$\begin{aligned}
S_{i,j}^{t+dt} = S_{i,j}^t + nS_{i-1,j}^t + sS_{i+1,j}^t + eS_{i,j+1}^t + wS_{i,j-1}^t + nwS_{i-1,j-1}^t \\
+ swS_{i+1,j-1}^t + neS_{i-1,j+1}^t + seS_{i+1,j+1}^t \quad (5)
\end{aligned}$$

Nesse caso o estado da célula (i, j) no instante t é dado por $S_{i,j}^t$. Os pesos n, s, w, e, nw, ne, sw e se são ajustados para dar o formato desejado ao resultado. Os autores não indicam os pesos a serem usados, mas informam que as seguintes relações devem valer:

$$e > ne > se \quad (6)$$

$$w > nw > sw \quad (7)$$

$$e = w, ne = nw, se = sw \quad (8)$$

Para o autômato para oxidação térmica, as regras não são baseadas em um modelo físico. O autômato celular apresentado apenas busca gerar uma figura geometricamente semelhante ao formato gerado durante uma oxidação térmica.

O terceiro simulador, de deposição de filmes, permite simular o crescimento de filmes CVD. Esse autômato celular também usa uma vizinhança de Moore. Cada célula possui um estado $C_{i,j}^t$ que representa a razão entre a área depositada e a área total da célula. Um valor de 1,0 representa uma célula totalmente preenchida. A evolução do estado de uma célula é dada por:

$$C_{i,j}^{t+dt} = C_{i,j}^t + (C_{i-1,j}^t + C_{i+1,j}^t + C_{i,j-1}^t + C_{i,j+1}^t) + 0,83(C_{i-1,j-1}^t + C_{i+1,j-1}^t + C_{i-1,j+1}^t + C_{i+1,j+1}^t) \quad (9)$$

onde $C_{i,j}^{t+dt}$ é o estado da célula na posição (i, j) no instante $t + dt$ e $C_{i,j}^t$ é o estado da célula na posição (i, j) no instante t . Os demais índices indicam as demais células da vizinhança de Moore. O valor 0,83 foi determinado experimentalmente comparando os resultados das simulações feitas usando o simulador com resultados experimentais, de maneira semelhante ao coeficiente D da equação (4) (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2001).

Existe uma variação desse autômato contínuo para deposição de filmes CVD com modificações para compensação temporal e crescimento anisotrópico (HUANG,

ZHOU, *et al.*, 2006). Uma variação da equação (9) é apresentada neste trabalho. Esta variação consiste em adicionar um valor para o passo de tempo na equação.

$$C_{i,j}^{t+dt} = C_{i,j}^t + T (C_{i-1,j}^t + C_{i+1,j}^t + C_{i,j-1}^t + C_{i,j+1}^t) + 0,83 T (C_{i-1,j-1}^t + C_{i+1,j-1}^t + C_{i-1,j+1}^t + C_{i+1,j+1}^t) \quad (10)$$

Para todos os autômatos celulares apresentados até agora, os passos de tempo são constantes. Para os autômatos onde a regra depende do valor do passo de tempo, a escolha desse valor é importante. Como é constante, é possível que o estado de uma célula ultrapasse o valor de 1,0. Esse evento significa que uma célula foi corroída (ou depositada) durante um tempo maior que o necessário para sua remoção (ou deposição). Esse efeito é indesejável e é pior para passos de tempo maiores. No entanto usar valores muito pequenos para o passo de tempo implica um custo computacional maior. Para diminuir erros devido a esse efeito sem prejudicar excessivamente o tempo de processamento, os autores sugerem, no caso da simulação de corrosão do fotorresiste, o seguinte valor para o passo de tempo:

$$t = \frac{a}{4R_{max}} \quad (11)$$

onde t é o passo de tempo, a é o tamanho do lado da célula e R_{max} é o maior valor entre as taxas de corrosão das células.

Mencionaremos aqui um último autômato celular. Esse autômato celular tem estados discretos e pode ser aplicado para descrição da evolução de superfícies (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2005). Isso o torna interessante também para simulação de processos como CVD. As regras do autômato celular não são simples e requerem a análise de uma vizinhança que pode chegar a 7x7 células. Cada célula está em um de dois estados: vazio ou depositado. Existem três regras diferentes que são alternadas no tempo, a saber:

1. Em passos múltiplos de cinco, a regra utiliza a vizinhança de Moore. Se qualquer vizinho de uma célula vazia estiver no estado depositado, a célula passa para o estado vazio.
2. Em passos múltiplos de oito, a regra utiliza a vizinhança de Von Neumann. Novamente se qualquer vizinho de uma célula vazia estiver no estado depositado a célula passa para o estado vazio.
3. Nos demais passos, uma vizinhança semelhante à de Moore de tamanho 7x7 é utilizada. As regras para esse caso são bastante complexas e não são reproduzidas aqui.

Os autores descrevem o autômato como binário apesar de, a rigor, as informações sobre qual regra usar serem armazenadas junto ao estado da célula. No entanto o autômato é interessante por ter estados discretos. Isso evita a necessidade de fazer contas em ponto flutuante, o que acelera o autômato celular e evita erros numéricos ou a necessidade de correções temporais ou passos de tempo limitados. A complexidade da regra acima, no entanto, dificulta bastante a passagem para três dimensões, necessária para o nosso software.

2.6 Método dos Elementos Finitos

Como parte do desenvolvimento do software foram implementadas funcionalidades básicas para a análise mecânicas de estruturas. O objetivo do desenvolvimento dessas funcionalidades básicas é que, no futuro, possam ser utilizadas em outros projetos. Outras ferramentas de análise baseadas no MEF poderão utilizar as mesmas ferramentas para resolver sistemas ou apresentar os resultados obtidos. O objetivo final é que o simMEMS seja capaz de simular o processo de fabricação de MEMS e permita que análises sejam feitas sobre os resultados gerados dentro do próprio simMEMS. A seguir são descritos brevemente o método dos elementos finitos e os elementos já implementados.

Muitos dos problemas que encontramos na vida real são bastante complexos. O cálculo dos esforços que agem sobre um avião durante o vôo é um problema bastante complexo. Em geral o que se faz para entender um problema complexo é dividi-lo em vários problemas mais simples, esses sim capazes de serem resolvidos diretamente.

Muitas vezes é possível fazer um modelo bom para um sistema com um número finito de elementos. Pode-se simular razoavelmente bem o funcionamento de um circuito complexo simulando o comportamento dos resistores, indutores, capacitores, amplificadores, etc. que o compõe. Mas existem sistemas onde essa subdivisão é infinita. Nesses casos utiliza-se de uma ferramenta matemática, o infinitesimal. Esses problemas são em geral descritos por equações diferenciais ou semelhantes.

O primeiro tipo de sistema é dito discreto e em geral existem métodos bem definidos para resolvê-los. A análise de malhas e a análise nodal são dois métodos bem definidos para a solução de um circuito elétrico. Existem métodos semelhantes para outros campos. Já para o segundo tipo de sistema, o sistema contínuo, não existe um método fixo para solução dos problemas. É preciso, nesses casos, encontrar uma resposta através da manipulação matemática das equações.

O MEF define uma maneira de dividir um meio contínuo em um número finito de partes para o qual o comportamento é definido por um número fixo de parâmetros. Em seguida aplica-se os métodos já conhecidos e bem definidos de análise discreta do sistema composto por esses elementos.

As partes em que separamos o sistema são os elementos. Elementos em geral têm um número fixo de nós. Cada nó tem diversos graus de liberdade, dependendo da análise que será feita. Para fazer a análise estrutural em duas dimensões precisamos pelo menos dos deslocamentos nas duas direções possíveis. Um elemento de pórtico é mostrado abaixo. Ele possui dois nós (pontos vermelhos) com três graus de liberdade cada um (representados pelas setas laranja).

Figura 14 – Um elemento de pórtico com dois nós e três graus de liberdade em cada nó.

Podemos agora escrever o sistema que relaciona as forças e deslocamentos nesse elemento.

$$\mathbf{K}_e \mathbf{U}_e = \mathbf{F}_{\text{externas}} + \mathbf{F}_{\text{internas}} \quad (12)$$

A matriz $\mathbf{K}_e$ é chamada de matriz de rigidez do elemento. O vetor $\mathbf{U}_e$ é o vetor de deslocamentos e os vetores $\mathbf{F}_{\text{externas}}$ e $\mathbf{F}_{\text{internas}}$ representam as forças atuando no sistema. O seu valor depende das chamadas funções de forma dos graus de liberdade do elemento. As funções de forma descrevem como os graus de liberdade variam entre os nós do elemento. Não será discutido aqui como encontrar essa matriz no caso geral. O vetor de deslocamentos é composto pelos graus de liberdade definidos para o elemento. O lado direito da equação contém os esforços aplicados sobre o elemento. No MEF todos os esforços devem ser transferidos para os nós.

Análise então a estrutura na Figura 15. São necessários três elementos para representar essa estrutura. Não é possível usar apenas dois elementos, pois as barras não estarão conectadas nesse caso. Note que apenas quatro nós são utilizados, e não seis nós, para os três elementos. Alguns nós são compartilhados, o que garante que os três elementos estejam conectados entre si e assim estamos efetivamente modelando a estrutura desejada.

Figura 15 – Na direita temos uma estrutura e na esquerda a sua separação em elementos.

A análise do sistema é feita primeiramente juntando as equações de todos os elementos em um único sistema. Dependendo da formulação do elemento será preciso fazer uma multiplicação por uma matriz adicional que faz a rotação do elemento para que todos os deslocamentos estejam no mesmo sistema de coordenadas. No elemento de pórtico dado como exemplo isso é necessário. Após juntarmos as equações dos N elementos do sistema teremos um novo sistema global.

$$[K] \begin{bmatrix} u_1 \\ v_1 \\ \theta_1 \\ \vdots \\ u_N \\ v_N \\ \theta_N \end{bmatrix} = \begin{bmatrix} F_{x1} \\ F_{y1} \\ M_1 \\ \vdots \\ F_{xN} \\ F_{yN} \\ M_N \end{bmatrix} \quad (13)$$

Acima a matriz K é a matriz de rigidez global. A matriz de rigidez global é uma matriz quadrada $N \times N$. Para qualquer sistema onde a relação entre força e deslocamento é linear a matriz de rigidez será simétrica e singular. Fisicamente isso ocorre porque sem nenhuma restrição ao movimento a estrutura inteira tende a se mover como um corpo rígido por menor que seja o esforço aplicado. Assim existirão infinitas soluções válidas para o sistema. Para resolver esse sistema devemos então aplicar as condições de contorno do problema. Nesse passo, informações como apoios ou deslocamentos conhecidos são adicionados ao sistema. Finalmente, resolvendo o sistema acima se encontra os deslocamentos em todos os nós do nosso sistema. Como foram utilizadas as funções de forma na elaboração da matriz de rigidez, também se calcula implicitamente o valor dos deslocamentos em qualquer ponto dentro do elemento. Basta usar as funções de forma para determinar os demais

valores. Vale mencionar que se já se conhece os deslocamentos, o mesmo sistema pode ser utilizado para determinar os esforços aplicados à estrutura.

Independente do elemento utilizado o MEF é sempre aplicado de forma semelhante. As variações são relacionadas às diferenças entre os elementos utilizados. Nem todos os tipos de elementos necessitam de uma matriz de rotação. Alguns elementos têm mais de dois graus de liberdade por nó ou têm mais nós. Explicações mais detalhadas sobre as funções de forma, matrizes e rigidez e demais aspectos do MEF podem ser encontradas em diversas fontes (FISH e BELYTSCHKO, 2007)(BATHE, 1982)(COOK, MALKUS e PLESHA, 1989)(YANG, 1986). As explicações acima se restringem à análise estática linear de sistemas mecânicos. A seguir descreveremos os elementos que foram implementados no software.

O primeiro elemento implementado no software é um elemento de pórtico. Uma melhor descrição desse elemento pode ser encontrada em YANG, 1986. É possível aplicar a teoria de Euler-Bernoulli para barras para elaborar a matriz de rigidez desse elemento. O elemento de pórtico tem uma característica interessante: se for utilizado em condições onde a teoria de Euler-Bernoulli é válida a solução dada por um elemento é igual à solução analítica. Para muitos elementos um refinamento da malha é preciso para obter resultados mais próximos do real. A figura abaixo mostra os dois nós e os seis graus de liberdade do elemento utilizado.

Figura 16 – Elemento de pórtico utilizado no programa. Os nós são representados pelos pontos em vermelho. Os graus de liberdade estão em laranja.

É bastante comum tratar a flexão e a tração/compressão separadamente. Como é possível ver na matriz de rigidez do elemento, reproduzida abaixo, a flexão e a tração/compressão são independentes. A utilização de três graus de liberdade em cada nó, tratando em conjunto esforços axiais e de flexão, permite o cálculo de pórticos. A seguir reproduzimos a matriz de rigidez para esse elemento.

$$[K] = \begin{bmatrix} \frac{AE}{l^e} & 0 & 0 & -\frac{AE}{l^e} & 0 & 0 \\ 0 & 12\frac{EI}{l^e{}^3} & 6\frac{EI}{l^e{}^2} & 0 & -12\frac{EI}{l^e{}^3} & 6\frac{EI}{l^e{}^2} \\ 0 & 6\frac{EI}{l^e{}^2} & 4\frac{EI}{l^e} & 0 & -6\frac{EI}{l^e{}^2} & 2\frac{EI}{l^e} \\ -\frac{AE}{l^e} & 0 & 0 & \frac{AE}{l^e} & 0 & 0 \\ 0 & -12\frac{EI}{l^e{}^3} & -6\frac{EI}{l^e{}^2} & 0 & 12\frac{EI}{l^e{}^3} & -6\frac{EI}{l^e{}^2} \\ 0 & 6\frac{EI}{l^e{}^2} & 2\frac{EI}{l^e} & 0 & -6\frac{EI}{l^e{}^2} & 4\frac{EI}{l^e} \end{bmatrix} \quad (14)$$

Nessa matriz l^e representa o comprimento do elemento, A é a área da seção da barra, E é o módulo de Young do material da barra e I é o segundo momento de área da barra.

Esse elemento requer que utilizemos uma matriz de rotação antes de somar a matriz de rigidez local, reproduzida acima, à matriz de rigidez global. No programa foi utilizada uma matriz de rigidez que já embute as multiplicações pela matriz de rotação para diminuir o trabalho para montar a matriz de rigidez local. Com esse tipo de elemento é preciso igualar não só os deslocamentos nos nós, mas também suas derivadas. Devido a esse fato, usamos os polinômios de Hermite como as funções de forma, pois esses polinômios permitem fazer isso.

As análises modal e dinâmica também requerem uma matriz conhecida como a matriz de massa. Para um elemento em movimento, a equação (12) não é mais utilizada. O sistema agora deve sofrer uma modificação para permitir considerar a massa do objeto e sua aceleração. A modificação é apresentada na equação (15).

$$K_e U_e + M_e \ddot{U}_e = \{f(t)\} \quad (15)$$

A equação (15) apresenta dois novos componentes. A matriz M_e é a matriz de massa do elemento. Existem diversas maneiras de calcular essa matriz (YANG, 1986). Uma matriz de massa consistente para vibrações axiais e por flexão foi

utilizada. O vetor de forças $\{f(t)\}$ pode também ser nulo. Nesse caso temos uma vibração não-forçada ou livre.

Para resolver o sistema da equação (15) existem dois métodos: a integração explícita e a integração implícita (COOK, MALKUS e PLESHA, 1989). O software desenvolvido usa o método de integração explícita. Abaixo reproduzimos a matriz de massa utilizada para o elemento de barra (YANG, 1986). A matriz é simétrica e por facilidade usamos dois valores, λ e μ , para representar, respectivamente, $\cos(\phi)$ e $\sin(\phi)$, onde ϕ é o ângulo entre o elemento e o eixo horizontal.

$$\mathbf{M} = \frac{\rho AL}{420} \begin{bmatrix} 156\mu^2 & & & & & \\ -156\lambda\mu & 156\lambda^2 & & & & \\ -22L\mu & 22L\lambda & 4L^2 & & & \\ 54\mu^2 & -54\lambda\mu & -13L\mu & 156\mu^2 & & \\ -54\lambda\mu & 54\lambda^2 & 13L\lambda & -156\lambda\mu & 156\lambda^2 & \\ 13L\mu & -13L\lambda & -3L^2 & 22L\mu & -22L\lambda & 4L^2 \end{bmatrix} \quad \text{simétrica} \quad (16)$$

Essa matriz de massa já permite a rotação do elemento, i.e. não é preciso aplicar a transformação da rotação do elemento antes de somar a matriz de massa do elemento à matriz de massa global do sistema. Um tratamento semelhante pode ser feito com a matriz de rigidez da equação (14).

O segundo elemento utilizado no programa é um elemento isoparamétrico. Um elemento isoparamétrico é um elemento onde as coordenadas físicas são mapeadas com a mesma função de forma utilizada para a aproximação. Isso significa que a mesma função que nos diz o valor dos deslocamentos entre dois nós pode ser utilizada para determinar as coordenadas (x, y) finais de um nó. O elemento que implementamos possui quatro nós e dois graus de liberdade para cada nó e é conhecido como ISOQ4. Esse elemento usa uma função de forma linear. Uma descrição mais profunda desse elemento pode ser encontrada em (FISH e BELYTSCHKO, 2007). A Figura 17 mostra o elemento ISOQ4.

Figura 17 – O elemento ISOQ4 em azul com os nós em vermelho e os graus de liberdade em laranja.

Elementos isoparamétricos são interessantes porque permitem montar malhas que aproximam melhor o formato da estrutura original, principalmente quando existem formas curvas. Como exemplo, o ISOQ4 pode ter o formato de qualquer quadrilátero. Isso não ocorre para qualquer elemento de quatro nós. A figura abaixo mostra como isso pode ser vantajoso na hora de montar a malha para análise. Um elemento isoparamétrico pode aproximar melhor o formato da curva com menos elementos do que um elemento com formato fixo. Elementos isoparamétricos com mais graus de liberdade podem inclusive ter lados curvos.

Figura 18 – Malha para aproximar uma curva na estrutura.

Essa variação na forma é possível porque o elemento ISOQ4 que desenhamos tem esse formato numa base (ξ, η) . Para todos os cálculos da matriz de rigidez é embutida uma transformação do sistema de físicas de coordenadas Cartesianas para essa base (ξ, η) em que o elemento é definido.

Figura 19 – Mapeamento das coordenadas do elemento para as coordenadas Cartesianas onde a estrutura física está definida (FISH e BELYTSCHKO, 2007).

Tanto a matriz de rigidez como a matriz de massa para o elemento isoparamétrico são obtidas por integração numérica utilizando o método de Gauss. Por isso não apresentamos aqui essas matrizes. Nas análises modais e dinâmicas foram utilizadas matrizes de massa diagonais para o elemento ISOQ4. Essas matrizes foram obtidas utilizando o método HRZ. Nesse método apenas as posições diagonais da matriz de massa são calculadas. Esses valores são posteriormente corrigidos por um fator de escala para que sua massa seja equivalente à massa total do elemento (COOK, MALKUS e PLESHA, 1989).

O elemento isoparamétrico utilizado, diferente do anterior, não fornece uma solução exata para qualquer caso. Dependendo da estrutura é preciso uma malha com mais elementos para se obter uma boa resposta. No caso do ISOQ4 a solução é melhor se o elemento não estiver muito deformado.

3 Metodologia

O software desenvolvido foi escrito na linguagem C++. O software foi escrito para ser portátil para as plataformas Windows, Linux e OS X. A interface gráfica foi desenvolvida utilizando a biblioteca Qt(NOKIA, 2010) para criar janelas, menus e barras de ferramentas, etc. A biblioteca Qt fornece não só funcionalidades para criar interfaces gráficas de maneira uniforme para diversos sistemas operacionais. Existem também outras funcionalidades voltadas a facilitar o desenvolvimento de qualquer tipo de software. Nesse sentido existem classes para, entre outros, processamento paralelo, como geração de threads e estruturas para sincronização, acesso a bancos de dados, tratamento de imagens e processamento de XML. Essas funcionalidades foram utilizadas em alguns instantes mas através de interfaces próprias para facilitar mudanças futuras.

Além das bibliotecas, existem softwares também desenvolvidas pela Nokia para auxiliar no desenvolvimento de softwares que utilizem Qt. Softwares como Qt Creator, qmake e Qt Designer foram utilizados para auxiliar no desenvolvimento do software também. O qmake em especial ajuda porque permite a criação de um único arquivo que pode ser utilizado para compilar os softwares desenvolvidos em diversas plataformas.

Para desenhar os resultados utilizamos a biblioteca gráfica OpenGL(BENJAMIN, HAEMEL e WRIGHT JR., 2007). O OpenGL é uma API que define funções básicas para desenhar objetos bidimensionais ou tridimensionais. Existem implementações para essa API nas plataformas alvo, o que levou à sua escolha. Essa biblioteca foi utilizada nas versões anteriores do simMEMS.

O módulo de análise baseados no MEF utiliza uma biblioteca externa, a ALGLIB(SERGEY BOCHKANOV, 2010), para solução dos sistemas lineares e extração dos auto-valores de algumas matrizes. Trata-se de uma biblioteca escrita para diversas linguagens de programação. A sua utilização é bastante simples pois não depende de outras bibliotecas e pode ser facilmente compilada nas três plataformas. No entanto por ser desenvolvida para várias linguagens acaba por não utilizar as facilidades fornecidas pelo C++, como classes.

Além dos softwares para o escrever e compilar os softwares, alguns outros softwares foram utilizados para auxiliar no processo de desenvolvimento. O subversion (THE APACHE SOFTWARE FOUNDATION, 2011), ou svn, foi utilizado para o armazenar o código escrito. Trata-se de um software para controle de versões. O svn que permite criar repositórios de código e acessar remotamente esse código. Modificações são enviadas ao servidor que armazena as modificações. Diversos desenvolvedores podem enviar modificações e o svn se encarrega de unir essas modificações em uma versão única com todas as modificações. Esse software é extremamente útil, principalmente quando diversas pessoas trabalham em um mesmo programa.

O Trac é um sistema web que permite gerenciar tarefas e *bugs* em programas (EDGEWALL SOFTWARE, 2011). Os usuários do sistema podem registrar tarefas ou problemas encontrados e definir que desenvolvedor deve realizar cada tarefa ou conserto. É uma ferramenta útil para gerenciar projetos pois permite visualizar as tarefas a serem realizadas e ter uma noção do desenvolvimento do projeto.

Outra ferramenta utilizada foi o Hudson, um sistema de integração contínua (ORACLE, 2011). O Hudson também possui uma interface web. Um sistema de integração contínua tem a tarefa de, a cada nova versão do software, compilar, testar e disponibilizar o software. Os três sistemas estão disponíveis na Internet, em <http://aquamarina.lme.usp.br>, e permitem o desenvolvimento e gerência remota do desenvolvimento do software.

Os sistemas Hudson e Trac foram implantados como parte de um projeto de iniciação científica realizado junto ao GNMD. O servidor svn foi implantado junto com o desenvolvimento do software simMEMS. A configuração dos projetos simMEMS e CellularAutomata3D nos sistemas foi feita como parte do presente projeto.

3.1 Implementação do CellularAutomata3D

Para avaliar os diversos autômatos celulares que poderiam ser utilizados para simulação dos processos de deposição e corrosão de filmes um software mais simples foi desenvolvido. Esse software é inteiramente voltado à execução de autômatos celulares. Os mecanismos de entrada são simplificados em relação ao simMEMS. Contudo esse software tem a vantagem de facilitar a implementação de um novo autômato.

Esse software foi chamado de CellularAutomata3D. Ele permite a execução e visualização de autômatos de uma, duas ou três dimensões. O software tem uma interface gráfica que permite visualizar passo a passo a execução dos autômatos. O usuário pode também optar por rodar um número determinado de passos de uma só vez. A interface também permite escolher vistas, importante no caso de autômatos tridimensionais, e é possível salvar os resultados como imagens.

As condições iniciais são definidas, em geral, por uma imagem. Por exemplo, o simulador de deposição de filmes usa uma imagem de entrada onde a componente vermelha da cor de cada pixel indica a altura do substrato naquele ponto. Essa imagem é como um mapa de altura. Alguns autômatos requerem mais parâmetros e existem janelas para escolha desses parâmetros. Um exemplo disso é o simulador de corrosão úmida do Si. Nesse simulador o usuário indica, através de uma caixa de diálogo, o formato do substrato de Si.

A estrutura do software está baseada na classe Automaton. Essa classe abstrata é uma interface que define as funções membro que toda implementação de um autômato celular deve implementar. A função membro *runStep* deve rodar um passo no autômato celular. A função *save* deve salvar o estado atual do autômato como uma imagem. A função *saveAll* é um auxílio para indicar se a cada passo o programa deve salvar uma imagem com o estado atual. A função *getPainter* retorna uma instância de outra interface, Painter, que é responsável por desenhar na tela o estado atual do autômato, essa classe será discutida mais adiante. As funções *getCurrentTime* e *getTimeUnit* são para informar o instante atual do autômato. As descrições acima são para o funcionamento esperado das funções. O programador

não é obrigado a implementar essas funções; as funções devem existir nas classes filhas mas não têm que fazer nada.

A maioria dos autômatos celulares implementados são bidimensionais ou tridimensionais. Para facilitar a implementação desse tipo de autômato foram criadas duas *templates*: Automaton2D e Automaton3D. Essas classes já possuem alguns atributos e implementações das funções da classe Automaton que podem ser utilizadas por várias classes. Esses *templates* agilizam a implementação evitando a necessidade de reescrever esse código para todas as implementações de autômatos. Como exemplo, ambas as *templates* já fornecem atributos para armazenar os estado atual e o próximo estado, bem como as dimensões do autômato (altura, largura e, no caso do Automaton3D, profundidade). A estrutura das classes está retratada de maneira simplificada no diagrama de classes da Figura 20.

Figura 20 – Diagrama de classes para implementação de autômatos celulares.

Três interfaces também foram definidas para auxiliar a implementação de um autômato celular que forneça todas as funcionalidades desejadas. Essas interfaces são as classes Painter, Archiver e Loader. As classes também ajudam a tornar o código mais modular. Cada implementação da classe Automaton fica somente com o código referente as regras do autômato celular. Código para desenhar ou salvar o estado atual fica separado em outra classe. Abaixo descrevemos melhor essas interfaces.

A interface Painter define algumas funções membro voltadas a desenhar na tela o estado atual do autômato. A função *paint* desenha o estado atual do autômato na tela. As funções *getWidth* e *getHeight* são utilizadas para determinar as dimensões da rede do autômato. Essas dimensões não têm que ser, obrigatoriamente, igual ao número de células na rede. Para simplificar a tarefa no caso de um autômato bidimensional com rede quadrada existe uma *template*, a Painter3D, que fornece uma implementação padrão para a classe Painter. Nesse caso a implementação da classe Automaton deve implementar uma função membro a mais, a *getColor*. Essa função retorna, para cada possível estado do autômato, um vetor do tipo *float* que define a cor correspondente a esse estado.

A interface Archiver descreve uma interface para salvar imagens. A interface ajuda a separar o código da biblioteca Qt da implementação do autômato. Usamos o código do Qt para ter uma maneira de salvar imagens em diversos formatos de maneira fácil e consistente em diversas plataformas. A interface define duas funções membro: *saveImage* e *saveNextImage*. A função *saveImage* é usada para salvar uma única imagem. A implementação padrão dessa função no software, feita pela classe AutomatonArchiver, usa a tela padrão do sistema operacional para permitir ao usuário escolher o local, nome e tipo da imagem a ser salva. Já a função *saveNextImage* salva imagens em sequência. A primeira vez que é chamada, a mesma obtém do usuário o nome base da imagem, o local onde serão salvas a série de imagens e o tipo das imagens. As chamadas subsequentes simplesmente salvam uma imagem sem pedir dados ao usuário. Cada imagem recebe um nome diferente no formato *nomeXXXX.tipo*, onde *nome* é o nome base que o usuário forneceu, *XXXX* é o número da imagem e *tipo* é o tipo da imagem (png, jpg ou bmp).

A última interface definida é a classe Loader. Essa interface define uma única função membro, chamada *getAutomaton*. A função *getAutomaton* devolve uma instância da classe Automaton. Essa interface tem o intuito de encapsular o código necessário para pegar, do usuário, os dados necessários para inicializar um autômato celular. Dados como estado inicial, tamanho, etc. devem ser obtidos antes de instanciar uma implementação da interface Automaton. Em geral cada autômato têm sua própria implementação da classe Loader.

A interface gráfica usa o mesmo software utilizada pela nova versão do software simMEMS. Essa interface gráfica também serviu de base para desenvolver o

software de simulação mecânica por elementos finitos. A interface é constituída por uma classe filha da classe QMainWindow e por uma classe filha da classe QGLWidget. A classe QMainWindow fornece a janela principal do programa com barra de menu, barra de status e diversos *widgets*. Um *widget* no contexto da biblioteca Qt pode ser qualquer coisa desde uma janela até um controle, como um botão ou caixa de seleção. A GLWidget é um *widget* que permite desenhar usando OpenGL. Isso é feito reimplementando algumas funções dessa classe.

A implementação da classe QGLWidget, chamada simplesmente GLWidget fornece algumas funcionalidades que são úteis para todos os programas. A classe é responsável por atualizar a matriz de projeção do OpenGL para que ao mudar o tamanho da janela seja possível ver mais ou menos detalhes na tela. Também permite ao usuário rodar os objetos na tela clicando e arrastando o mouse. Essa rotação é feita no estilo *trackball*, permitindo rotações ao redor dos três eixos X, Y e Z. Existe também a opção de definir uma vista digitando os ângulos de rotação ao redor dos eixos X, Y e Z. No caso especial do CellularAutomata3D, não há barra de menus. As funções são todas acessadas por uma barra de ferramentas.

Além dessas classes a interface gráfica possui algumas janelas a mais. A maioria é utilizada pelas implementações da classe Loader. As exceções são as janelas para pedir os ângulos para vista e a tela para pegar o número de passos a serem executados quando se deseja rodar vários passos em seguida. No item 3.1.1 mostra-se lógica de utilização do programa para um autômato celular no CellularAutomata3D.

3.1.1 Implementação de um Novo Autômato Celular

Nessa seção os passos para implementar um novo autômato celular seguindo a estrutura imaginada e descrita previamente são descritos. Primeiramente o programador escreve uma implementação da classe Automaton. Ele pode ou não utilizar as classes Automaton2D e Automaton3D para auxiliar nessa tarefa. Em

seguida, para visualizar os resultados, o programador deve criar uma implementação da classe Painter para desenhar o estado atual do autômato. Caso seja um autômato bidimensional ele pode aproveitar a classe Painter2D. Caso queira permitir salvar imagens, o programador também deve implementar as funções membro *save* e *saveAll* apropriadamente. A maneira mais fácil de fazer isso é usando a classe AutomatonArchiver. O programador, nesse caso, precisa apenas gerar os parâmetros de entrada das funções membro dessa classe.

Com o autômato implementado, o programador deve agora criar uma implementação da classe Loader para obter do usuário os parâmetros para inicializar o autômato celular. Esses parâmetros dependem do construtor da implementação da classe Automaton. A classe AutomataFactory deve ser modificada para instanciar o novo autômato usando o Loader criado. Finalmente o programador deve modificar a interface gráfica para adicionar o novo autômato celular.

Se o programador puder utilizar as templates e classes já desenvolvidas, essa criação pode ser bastante rápida. O CellularAutomata3D fornece uma plataforma para prototipagem rápida de novos autômatos celulares. É possível fazer isso em outros programas. No entanto ao implementar o autômato usando essa infraestrutura o programador gera um código modular para carregar, salvar, desenhar e simular o autômato celular. Isso, por sua vez, facilita a reutilização do código. Isso pode ser comprovado quando os autômatos desenvolvidos e testados no CellularAutomata3D foram utilizados também no simMEMS.

3.2 Implementação do simMEMS

O software simMEMS foi criado para simular processos de deposição e corrosão seletiva de filmes bem como a corrosão úmida do Si em soluções de KOH. Desde sua criação, o objetivo do desenvolvimento sempre foi criar um software multi-plataforma capaz de simular esses processos com entradas feitas diretamente pela

interface do programa. Isso impõe algumas restrições ao software, principalmente no que tange às linguagens e bibliotecas que podem ser utilizadas.

O software foi desenvolvido para ser expandido. Dentro do possível foi utilizada uma abordagem orientada a objetos para gerar um software modular. O simMEMS é composto por três camadas principais, chamadas de Model, Core e UI.

3.2.1 A Camada Model

A camada Model armazena o código referente às classes para armazenar os objetos vistos dentro do software. Máscaras, filmes e substratos são representados por classes dentro dessa camada. Código que altera esses objetos não está, em geral, dentro dessa camada.

Isso foge, em parte, aos princípios da orientação a objetos. Isso porque uma classe é composta por atributos (dados) e métodos que alteram esses atributos. Todavia a maior parte do sistema simMEMS é justamente o código que altera esses objetos. Para não criar classes gigantescas, decidimos colocar esses código em outras classes, na camada Core.

3.2.2 A Camada Core

A camada core armazena as operações que podem ser feitas pelo programa. Copiar, mover e girar objetos, por exemplo, são operações realizadas pela camada Core. A camada fornece uma classe base para operações que pode ser estendida para implementar uma nova operação, como uma corrosão. A camada também fornece as funcionalidades para seleção de objetos e entrada de dados, como cliques do mouse ou entrada via teclado. Além disso, a mesma fornece ainda

funcionalidades para fazer e refazer operações. Isso permite que operações sejam criadas independentemente da interface gráfica utilizada.

3.2.3 A Camada UI

A camada UI contém as classes que representam as janelas, menus e toolbars do programa. É a única camada que usa a biblioteca Qt. Apesar de ser útil, decidimos não usar todas as funcionalidades da biblioteca Qt em todas as camadas. Isso facilita a mudança para outra biblioteca, caso seja preciso. Vale notar que isso já aconteceu. Inicialmente utilizávamos a biblioteca wxWidgets, e a biblioteca Qt ainda não estava disponível sob a licença LGPL. As dificuldades encontradas ao usar a biblioteca wxWidgets nos levaram a mudar para biblioteca Qt.

3.2.4 A Estrutura do Software

A seguir, na Figura 21, mostramos um diagrama reduzido das classes do sistema. Esse diagrama não mostra todas as classes, mas é suficiente para explicar a lógica do programa e como uma nova funcionalidade pode ser adicionada.

Figura 21 – Diagrama de classe mostrando a estrutura básica da camada Core para implementação de operações.

A camada UI dispara a chamada para o `OperationManager` para criar uma nova operação e chama o método `execute`. Esse método contém o código necessário para obter do usuário todos os dados para executar as operações. Por exemplo, a operação para copiar requer que o usuário selecione os objetos que deseja copiar e a distância para colocar a cópia. A entrada de dados pode ser feita através de uma janela ou usando o mouse ou teclado. No caso do mouse ou teclado, existem classes que fornecem uma interface para seu uso.

As classes `Keyboard` e `Mouse` operam de maneira semelhante. Supondo que é preciso que o usuário clique o mouse para determinar, por exemplo, a posição de um vértice. Registra-se, nesse caso, na classe `Mouse` que a operação atual deve ser chamada quando o usuário clicar o mouse. Quando o usuário clica o mouse, a classe `Mouse` verifica se existe alguma operação registrada. Se houver, o método `execute` da operação é chamado.

As operações usam uma variável interna para determinar os dados que já foram obtidos e quais devem ainda ser pedidos. Quando todos os dados forem obtidos, o método *end* é chamado e a operação é encerrada. Isso é feito como alternativa a usar threads e mecanismos de sincronismo para que programadores mais inexperientes possam desenvolver o programa.

A operação deve, também, implementar mais algumas funções membro, dependendo do tipo de operação. Qualquer operação precisa implementar o método *cancel*. Esse método deve limpar ou desfazer qualquer mudança que foi feita durante a operação que não será mais realizada. Por exemplo, se uma operação registrou que deve ser chamada quando o usuário clicar o mouse, a operação deve limpar esse registro.

Caso a operação possa ser desfeita ou refeita, também deve implementar os métodos *undo* e *redo*. Como o nome indica, a função membro *undo* desfaz qualquer mudança feita durante a operação e a função membro *redo* refaz a operação se a mesma foi desfeita. O controle da ordem das funções que podem ser desfeitas ou refeitas é feito pela classe *UndoRedoManager*.

Vale mencionar também mais algumas classes. A classe *SelectionManager* fornece funcionalidades para seleção de objetos. Essa classe utiliza as funcionalidades do OpenGL para seleção dos objetos. Como esse código não depende da interface gráfica podemos fornecer essa funcionalidade que pode ser reutilizada independentemente da interface gráfica.

A classe *LayerManager* gerencia as camadas dos objetos. Cada objeto pertence a uma camada. A camada é responsável por desenhar os objetos na tela. A classe *LayerManager* é responsável por desenhar as camadas, i.e., chamar a função de cada camada que desenha os objetos dentro da camada.

A classe *OpenGLUpdateManager* é a classe que permite que a camada *Core* informe a camada superior que é preciso desenhar os objetos novamente. Isso pode ocorrer, por exemplo, porque algum objeto foi criado ou mudou de cor.

Como mencionamos, a classe *OperationManager* é responsável por iniciar as operações. Também é responsável por cancelar as operações. Caso a operação tenha sido finalizada (função membro *end* chamada corretamente) e possa ser desfeita, a *OperationManager* também registra a operação com o

UndoRedoManager para que a operação possa ser desfeita. Uma última funcionalidade é a de centralizadora. Para facilitar, os demais gerenciadores são acessados, pelas operações, pela OperationManager.

O software foi desenvolvido em três camadas principais. O modelo em camada busca encapsular algumas classes e funcionalidades em cada camada. Uma camada deve utilizar apenas as funcionalidades da camada abaixo dela. Assim a camada UI não deve acessar diretamente as funcionalidades da camada Model. Apenas a camada Core pode acessar as funcionalidades da camada Model. De maneira semelhante a camada Core não deve acessar a camada UI, vista que essa se encontra acima dela.

No entanto existem ocasiões onde mudanças na camada Core devem ser informadas à camada UI. Por exemplo, quando a camada Core cria um novo objeto, é preciso desenhar a tela novamente para que o usuário veja o novo objeto. A camada Core teria então que avisar dessa mudança. Existe uma solução já conhecida para esse problema. Utilizamos um *design pattern* conhecido como *Observer pattern*. Essa solução foi bastante utilizada e vale explicar melhor o seu funcionamento.

O *Observer pattern* se utiliza de uma interface, que no nosso caso chamamos de IObserver. Essa interface define uma função membro: *notify*. O sujeito cujo estado queremos monitorar permite registrar um objeto do tipo IObserver. Quando esse estado muda, a função *notify* dos objetos IObserver registrados com o sujeito é chamada. No nosso caso, diversas classes da camada Core permitem registrar observadores, como a classe OpenGLUpdateManager que permite registrar um IObserver que é notificado quando precisamos redesenhar a tela.

As implementações da interface IObserver, no entanto, ficam na camada UI. Como essas implementações ficam na camada UI, podem se utilizar das funcionalidades da camada UI. Dessa maneira é possível fazer a camada Core rodar, de certa maneira, código da camada UI sem que haja uma dependência da camada UI na camada Core. Esse mesmo mecanismo é utilizado por operações que utilizam caixas de diálogo para obter dados do usuário.

A Figura 22 mostra o diagrama de classes referente a esse *design pattern*. A classe `ConcreteOperation` representa uma classe que queremos monitorar (por exemplo, o surgimento da necessidade de redesenhar a tela).

Figura 22 – Diagrama de classe mostrando o Observer pattern.

Quando o evento que desejamos monitorar ocorre, a função membro *notifyObservers* é chamada. Essa função chama a função *notify* de todos os observadores registrados. As classes `ConcreteObserverA` e `ConcreteObserverB` são implementações da interface `IObserver` e podem ser registradas junto a classe `ConcreteOperation` para monitorá-la. A implementação da função membro *notify* dessas classes deve ser responsável pelo comportamento desejado. O `ConcreteObserverA` pode, por exemplo, redesenhar a tela enquanto o `ConcreteObserverB` conta o número de vezes que essa necessidade ocorreu para perfilar o sistema.

3.2.5 Implementação dos Simuladores Baseados em Autômatos Celulares

Os autômatos celulares para simulação utilizando a classe `TotalVolume` foram todos implementados como classes filhas da classe `GenericAutomaton`. A classe fornece uma função membro privada que deve ser implementada pelas classes filhas. A classe `GenericAutomaton` por sua vez, fornece uma função membro pública que roda um passo. Além disso a função *runStep* também é responsável por garantir que

os dados do estado atual e do próximo estado não sejam acessados durante a simulação.

A operação *RunAutomata* é quem instancia e roda os autômatos de simulação. Usamos um vetor de ponteiros para a classe *GenericAutomaton* para armazenar instâncias dos diversos tipos de autômatos celulares (Deposição de filmes, corrosão úmida, etc.). A operação possui funções para mudar qual autômato deve ser rodado bem como outras opções (máscaras, filmes a serem depositados ou corroídos, etc.). A função *execute* permite ao usuário escolher o substrato e chama a função membro *notify* do *IObserver*, caso um observador tenha sido registrado. O observador registrado cria as janelas que o usuário usa para mudar as condições da simulação. As funções *cancel* e *end* são utilizadas para informar, através de um segundo observador, que a função foi encerrada para que as janelas sejam fechadas automaticamente. As funções *undo* e *redo* não foram implementadas ainda. Os autômatos não são reversíveis, portanto não podemos reverter as mudanças feitas. Armazenar diversas cópias do volume de células também é inviável devido à memória necessária. Estamos estudando alternativas para implementar essas funcionalidades.

3.3 Comparação dos Autômatos

As medidas de taxas de corrosão e de deposição dos autômatos celulares desenvolvidos foram feitas experimentalmente com o auxílio de um software que analisa os resultados dos autômatos. Esse software roda um número definido pelo usuário de passos do autômato. Posteriormente gera como saída a posição das células na superfície resultante. O software também informa o número de passos (ou o tempo) simulado e a posição do centro da rede. O software usa, no caso da deposição, uma rede com uma única célula no centro como semente. As taxas são calculadas a partir da distância das células da superfície até essa célula central pela fórmula abaixo:

$$T_{dir} = \frac{|\overrightarrow{R_{dir}}|}{T} \quad (17)$$

onde T_{dir} é a taxa de deposição ou corrosão na direção $\overrightarrow{R_{dir}}$ na superfície e T é tempo de simulação. As distâncias são medidas entre os centros das células. A figura abaixo mostra um exemplo de medida experimental da taxa de deposição.

Figura 23 – Imagem mostrando como é feita a medida experimental da taxa de deposição. A figura mostra o estado inicial para a medida da deposição em (a). A célula em vermelho é a semente para o crescimento. Em (b) tem-se o estado do autômato após n passos. A medida da distância depositada em uma direção também é mostrada em (b).

Quanto maior a rede utilizada para simulação maior o número total de passos que podem ser rodados. Um número grande de células depositadas é desejável para a medida. Visto que dessa forma são obtidas medidas para um número maior de direções. A influência da rede também é menor quando $|\overrightarrow{R_{dir}}|$ é muito maior que o tamanho das células.

Outra avaliação realizada foi a comparação do formato do filme crescido com o formato considerado ótimo. Devido a simplicidade na avaliação, a rede com uma única célula no centro representando o substrato foi utilizada, como na Figura 23(a). Nesse caso, a forma ideal é um disco. Portanto o formato do filme crescido foi comparado ao de um disco.

Para realizar a comparação, é preciso definir o raio do disco ideal. Em todas as comparações, o raio do disco ideal foi determinado medindo a distância da semente

até a última célula depositada na mesma linha que a semente. O raio poderia ter sido medido em outras direções, desde que todos os autômatos celulares sejam comparados utilizando a mesma direção. A escolha da direção horizontal foi baseada na facilidade da medida e na sua consistência. Medir o raio na direção (1,3), por exemplo é complicado em uma rede quadrada.

Figura 24 – Medida do raio para o disco ideal.

Determinado o raio do disco ideal, a comparação é relativamente simples de se executar. À medida que o filme cresce, foram verificadas o número de células no estado vazio, ou não depositadas, que se encontravam dentro do disco ideal. Tal configuração foi considerada um erro (erro A na Figura 25). Células no estado filme, ou depositadas, que se encontravam fora do disco ideal também são contabilizadas como um erro (erro B na Figura 25). Em ambos os casos a determinação de se uma célula encontra-se dentro ou fora do disco ideal é feita comparando a distância do centro da célula ao centro da semente ao raio do disco ideal.

Figura 25 – Possíveis erros ao comparar com a forma ideal. O erro A indica uma célula que deveria ter sido depositada e não foi. O erro B indica uma célula que foi depositada mas não deveria ter sido.

A razão entre o número de erros durante a deposição e o número de células dentro do disco ideal também foi calculada. Essa razão permite verificar, com o passar do tempo, a tendência do filme crescido através do autômato celular. Se a razão tende a aumentar com o aumento do raio do disco ideal, o resultado piora ao aumentar o nível de detalhe da simulação. Inversamente, se a razão diminui com o aumento do raio, o resultado melhora com o aumento do nível de detalhes. Se a razão se mantém constante, o resultado final tende a ter um formato constante e os resultados não melhores ou piores com uma rede mais fina.

4 Resultados e Discussões

Nesta seção apresentamos os resultados obtidos. Os resultados são apresentados em função dos softwares desenvolvidos. Primeiramente os softwares desenvolvidos são apresentados. Em seguida os resultados gerados por esses softwares são apresentados e discutidos.

4.1 CellularAutomata3D

Nesta seção serão descritos alguns dos autômatos celulares desenvolvidos no CellularAutomata3D. Os autômatos desenvolvidos no CellularAutomata3D e posteriormente implementados no simMEMS são descritos no item 4.2. Também são apresentados alguns dos resultados gerados pelo software CellularAutomata3D. Finalmente uma comparação entre os resultados gerados pelos diversos autômatos celulares implementados é feita.

4.1.1 Autômatos Desenvolvidos

Alguns autômatos celulares foram implementados para familiarização com os conceitos dos autômatos celulares, as diferentes condições de contorno e vizinhanças bem como suas vantagens e desvantagens. O autômato celular probabilístico que simula difusão em uma dimensão descrito no item 2.3 é um dos autômatos celulares implementados no CellularAutomata3D. Também foram implementadas duas versões, uma determinística e outra não-determinística, de um

autômato celular para simulação do comportamento de grãos de areia que utilizam a vizinhança de Margolus. A Figura 26 mostra exemplos da execução desses autômatos celulares. A descrição de ambos pode ser encontrada em CHOPARD e DROZ, 1998.

Figura 26- Simulação de grãos de areia usando um autômato celular (a) e um autômato celular não determinístico (b).

Na Figura 26 simulamos um relógio de areia utilizando os autômatos para simulação de areia. Os grãos de areia são representados por pontos pretos. Paredes fixas são representadas por pontos em vermelho e pontos em branco são espaços vazios. Apesar das condições iniciais e o número de passos serem os mesmos, os resultados são bem diferentes. Fica claro que há uma grande diferença entre os centros da Figura 26(a) e da Figura 26(b). Isso ocorre porque analisamos sempre blocos de 2x2 células. As regras são definidas considerando como os grãos de areia se comportam nesses blocos. Se esses pequenos blocos têm grãos de areia em todas as células, nada ocorre. Portanto a coluna no centro da Figura 26(a) só cai pelas bordas. A Figura 27 mostra exemplos da evolução dos blocos de 2x2 células

Figura 27 – Comportamento dos grãos de areia utilizado para simular a Figura 26.

A modificação do autômato probabilístico, em relação ao original, é considerar um possível atrito quando um bloco possui dois grãos de areia no topo e dois espaços vazios em baixo. No autômato original ambos os grãos sempre caem. No autômato celular estocástico, há uma variável aleatória que determina se os grãos caem ou não. Isso torna a queda mais gradual, como visto na Figura 26(a), e o resultado é mais próximo ao real.

Os exemplos prévios mostram bem como a adição da incerteza pode tornar o comportamento de um autômato celular mais realístico. Outro autômato implementado que é de interesse científico é o FHP, assim batizado pelo nome dos seus criadores: Frisch, Hasslacher e Pomeau (FRISCH, HASSLACHER e POMEAU, 1986). Esse autômato celular é usado para simulação de fluídos e se baseia no comportamento microscópico das partículas de um fluído para simular o comportamento macroscópico do fluído. Ele funciona simulando as partículas do fluído e mantendo a quantidade de movimento e o número dessas partículas constantes. Células hexagonais são utilizadas. Partículas podem chegar a cada célula de seis direções diferentes, mostradas na Figura 28(a). Valores binários são utilizados para indicar se uma partícula vem da direção V_i , $i = 0,1,2,3,4,5$. O autômato usa a vizinhança mostrada na Figura 28(b), composta pela célula e suas seis vizinhas diretas. A regra é simples, o valor V_i no instante $T+\Delta t$ é definido pelo valor de V_i da vizinha na mesma direção e sentido oposto do vetor velocidade $\vec{V}_i$. Por exemplo, o valor de V_3 em $T+\Delta t$ da célula é o mesmo que o valor de V_3 da célula à esquerda (célula na mesma direção e sentido oposto ao vetor $\vec{V}_3$) em T .

Figura 28 – Direções das partículas do FHP em (a) e a vizinhança do autômato em (b).

A incerteza, nesse caso, é inserida quando se tem configurações em que soma da quantidade de movimentos de todas as partículas que chegam a uma célula é nula. Nesses casos pode-se escolher qualquer outra configuração de partículas desde que o número de partículas e a soma dos momentos continuem inalterados. Quando duas partículas se chocam, o resultado será que elas partem em sentidos opostos, mas a direção dessas partículas não importa, desde que seja a mesma para ambas. A Figura 29 mostra um exemplo disso.

Figura 29 – Exemplo de uma configuração com soma da quantidade de movimento nula (a). A colisão dessas duas partículas pode gerar tanto o resultado em (b) como em (c).

Como ocorreu na Figura 29(a), a soma da quantidade de movimento das partículas é nula. Nesse caso podemos manter a configuração ou considerar que após a colisão o caso (b) ou (c) ocorreu. Uma variável aleatória é utilizada para escolher o próximo estado nessas condições. Essa escolha é feita antes da determinação do estado em $T+\Delta t$ já descrita.

A implementação desse autômato celular foi especialmente interessante porque uma pequena modificação nos permitiu criar um simulador para deposição balística. Nesse tipo de deposição imaginamos que as partículas vêm de direções aleatórias e quando se chocam com o substrato são depositadas. Modificamos o autômato FHP original para ter mais estados, permitindo assim definir um filme sobre o qual as partículas do fluido poderiam ser depositadas. Quando uma partícula colide com o substrato há uma probabilidade p de que a partícula seja depositada, aumentando o filme. Nesse caso o espaço que antes era vazio passa agora a ser preenchido pelo filme sendo depositado. A Figura 30 mostra o resultado de uma simulação feita com esse autômato celular modificado.

Figura 30 – Deposição de um filme usando FHP para simular as partículas do filme. O número de passos simulados está ao lado das imagens. O Substrato está em verde e o filme depositado em amarelo escuro. As partículas são representadas em escala de cinza e as paredes em vermelho.

Na Figura 30 posições em branco representam espaços vazios. Um grau de cinza indica o número de partículas que chegam a uma certa posição. Esse valor varia de 0 a 6, sendo branco para 0 e preto para 6. O substrato inicial está em verde e as paredes sobre o qual o filme não cresce em vermelho. O filme depositado está em oliva. O número sobre as imagens indica o número de passos simulados.

Esse modelo para a deposição gera muitos poros no filme resultante. Como simula o comportamento do fluído, também é lento. No entanto, devido ao fato de simularmos o fluído, também temos alguns efeitos que ocorrem naturalmente. Podemos ver uma diferença entre a velocidade de crescimento nas quinas inferiores do “U” que ocorre naturalmente devido à maior dificuldade das partículas de alcançar esse ponto. Com o andar da simulação o filme já depositado oculta essas regiões, eventualmente cobrindo totalmente a região. Isso pode ser visto após 60 passos da simulação. O “miolo” vazio do “U” também para de crescer devido à exaustão das partículas do gás. Um efeito semelhante causa uma diminuição na taxa de crescimento com o passar do tempo pois com menos partículas o tempo entre colisões de partículas com o substrato aumenta. Devido ao ruído do próprio autômato também é preciso rodar várias simulações para tirar o comportamento médio. Isso torna o autômato pouco prático para simulação de filmes. Um autômato celular que gere resultados semelhantes sem a necessidade de simular o fluído seria mais interessante.

Como parte dos estudos um simulador de corrosão anisotrópica também foi desenvolvido. Isso permitiu o desenvolvimento, posteriormente, de um simulador único capaz de simular tanto processos de microfabricação em substrato como em superfície. O simulador escolhido foi o Basic Cellular Automaton(BCA). Trata-se de um simulador simples mas bastante útil. Os resultados obtidos por ele são muito próximos aos resultados experimentais em diversos casos. A título de exemplo os resultados da corrosão de uma esfera de Si são apresentados.

Os resultados dessa simulação apresentam uma boa coerência com resultados experimentais. A corrosão de esferas de silício é interessante porque uma esfera contém infinitos planos cristalográficos. Justamente por isso muitos pesquisadores usam substratos com essa forma de para medir taxas de corrosão úmida do Si. A Figura 31 mostra algumas imagens dessa corrosão geradas pelo CellularAutomata3D.

Figura 31 – Vários passos de uma corrosão de uma esfera de Si em solução de KOH simulado pelo BCA. A imagem mostra somente os átomos da superfície sendo corroída.

Nessas imagens os pontos verdes são átomos que possuem dois primeiros vizinhos. Os pontos azuis são átomos que possuem três primeiros vizinhos e mais de nove segundos vizinhos e os pontos laranjas são átomos que possuem três primeiros vizinhos mas menos de nove segundos vizinhos. Uma descrição mais profunda desse autômato se encontra no item 4.2.1.1.

O CellularAutomata3D também foi utilizado para testar vários dos autômatos celulares mencionados no item 2.5. No item seguinte são apresentados alguns resultados gerados utilizando esses autômatos. Os resultados obtidos também são discutidos.

4.1.2 Deposição e Crescimento de Filmes

No item 2.5 um autômato celular que aplica o princípio de Huygens diretamente para a simulação de processos de corrosão e deposição de filmes foi descrito. Esse autômato celular foi implementado no software e na Figura 32 estão alguns dos resultados gerados por este simulador. Os passos de tempo foram virados para evidenciar a influência da duração do passo sobre a isotropia do resultado.

As imagens mostram como essa abordagem produz ótimos resultados para crescimentos isotrópicos, especialmente quando a duração do passo de tempo é longa. As figuras para os passos mais longos claramente apresentam facetas menores em comparação com as imagens com passos maiores. Em compensação perde-se precisão no tempo de deposição ao aumentar o passo de tempo. Não é possível, por exemplo, simular uma deposição de 30 minutos com passos de 20 minutos.

Figura 32 – Evolução da deposição de um filme (em oliva) sobre uma mesma superfície (em vermelho) utilizando três valores diferentes para o passo de tempo.

Em processos CVD e PVD é comum que paredes verticais como as das imagens acima causem um efeito de sombreamento que reduz a taxa de deposição em algumas regiões. Para simular bem esse tipo de deposição, precisamos alterar o valor da taxa de deposição a cada passo. Isso leva a um dilema onde para ter resultados mais próximos ao real precisamos de um passo de tempo menor, mas uma diminuição no passo de tempo acarreta um aumento na anisotropia. Para tentar resolver esse tipo de problema, partimos para os autômatos celulares com vizinhanças menores, como os descritos na seção 2.5.

Um autômato simples próprio para a deposição de filmes foi desenvolvido. Utilizando a vizinhança de Moore, podemos criar um autômato com três estados: Substrato, Filme e Vazio. Uma célula no estado Substrato representa o formato inicial sobre o qual o filme será depositado. Uma célula no estado Filme representa um volume do filme depositado. Uma célula no estado Vazio representa um volume não preenchido que pode sofrer uma deposição.

Uma regra bastante simples é mudar o estado de uma célula no estado Vazio para o estado Filme se houver uma célula vizinha qualquer no estado Filme ou Substrato. Caso contrário as células mantêm o seu estado.

Esse autômato tem a vantagem de ser bastante simples de implementar. Os resultados não são altamente satisfatórios. No entanto, em comparação aos resultados previamente publicados para o simMEMS se mostra com algumas vantagens no caso tridimensional. A Figura 33 mostra a evolução dos filmes em duas dimensões.

O formato dos filmes resultantes é semelhante aos obtidos pelo algoritmo geométrico utilizado nas primeiras versões do simMEMS. Algumas vantagens do uso de um autômato celular são que superfícies com qualquer ângulo podem ser tratadas e que a interseção de frentes não é um problema. Vale notar que apesar de alguns formatos arredondados se manterem, a deposição não é isotrópica.

Figura 33 – Substrato inicial e a evolução do autômato celular. Cada imagem mostra a evolução após dez passos. O substrato está colorido em vermelho e o filme em oliva.

Um exemplo do crescimento anisotrópico é o achatamento da ponta central devido às diferenças nas taxas de deposição. Nesse autômato, o crescimento é mais rápido nas direções NE, NW, SE e SW do que nas direções N, S, E e W. Isso leva a ponta a crescer para os lados mais rapidamente que para cima e sua influência logo desaparece porque toda a parte inferior do "U" é coberta pelo filme.

O Gráfico 1 apresenta a taxa de deposição em função da direção medida experimentalmente seguindo a metodologia descrita na seção 3. Fica claro que o autômato é bastante anisotrópico. Como mencionamos acima, as taxas de crescimento são perceptivelmente maiores nas direções diagonais. A medida foi feita rodando 300 passos desse autômato numa rede de 601x601 células.

Gráfico 1 – Medidas da taxa de deposição do autômato simples. Há um aumento considerável da taxa de deposição nas direções diagonais.

A passagem para 3D do autômato celular simples com vizinhança de Moore descrito anteriormente é bastante simples. Modificamos a vizinhança para incluir todas as vinte e seis células que envolvem a célula sendo analisada. Se uma célula está no estado Vazio e qualquer célula vizinha estiver no estado Substrato ou Filme, o estado da célula sendo analisada é modificado para Filme. Os estados são mantidos em qualquer outra situação. As figuras abaixo mostram a evolução do autômato celular tridimensional descrito. O substrato está em vermelho e o filme depositado em oliva. Simulamos a deposição de um filme sobre uma ponta. Vale mencionar que a versão atual do software simMEMS que usa um algoritmo geométrico é incapaz de fazer esse tipo de simulação. Um algoritmo conhecido como “marching cubes” (LORENSEN e CLINE, 1987) foi utilizado para gerar as superfícies.

Figura 34 – Evolução do autômato simples em três dimensões. Vemos claramente o achatamento da ponta, semelhante ao que ocorre em duas dimensões. As imagens de (a) a (f) mostram a evolução da deposição de dez em dez passos. Em (g) temos um corte após vinte passos.

Tanto o autômato simples bidimensional como o autômato simples tridimensional se mostraram incapazes de simular um crescimento isotrópico. Mudando o número de vizinhos sendo analisados ou a regra do autômato celular são maneiras de diminuir o problema da anisotropia, mas não resolvem totalmente. O autômato celular descrito em SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2005 também foi implementado. A Figura 35 mostra o mesmo exemplo simulado com esse autômato celular.

Figura 35 – Crescimento do filme, em oliva, sobre o substrato, em vermelho, usando o autômato celular para simulação da evolução de superfícies curvas.

Como é possível ver na figura acima, o crescimento não é tão quadrado, mas também não é isotrópico. O crescimento nas quinas é bastante facetado quando comparado ao autômato de Huygens. A deposição sobre a ponta também cresce mais lentamente e a influência da ponta desaparece rapidamente.

A medida experimental das taxas de crescimento do autômato descrito, no Gráfico 2, confirma que ainda há certa anisotropia. Percebe-se que há uma melhora bastante perceptível nas taxas diagonais, mas ainda é possível visualizar diferenças entre algumas direções. A taxa de crescimento é máxima nas quatro direções cardeais (N, S, E e W).

Gráfico 2 – Taxas de deposição medidas para o autômato para evolução de superfícies (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2005). As taxas são mais próximas das de um crescimento isotrópico quando comparadas aos autômatos com vizinhança de Moore.

Como mencionado anteriormente, outra maneira de diminuir a anisotropia do crescimento é utilizar um autômato celular contínuo. Alguns autômatos celulares foram propostos na literatura para tal tarefa. Esses autômatos também permitem diferenciar as taxas de deposição ou corrosão das células, o que é bastante interessante, pois permite incorporar efeitos como o sombreamento causado pelo crescimento do filme ou para simular crescimentos não conformes. Efeitos como a exposição do fotoresiste também podem ser incorporados.

Primeiramente foi implementado o mais simples dos autômatos, utilizado para simular a deposição de filmes. Novamente tal autômato pode ser também utilizado para simulação da corrosão de filmes apenas mudando a interpretação dos resultados. A Figura 36 mostra os resultados da simulação de uma deposição. Mais

uma vez um substrato em forma de "U" com uma ponta no meio foi utilizado. Novamente o substrato está em vermelho e o filme depositado em oliva.

Figura 36 – Substrato inicial e a evolução do autômato celular contínuo. Cada imagem mostra a evolução após dez passos

Podemos ver que esse autômato simula melhor a deposição de filmes sobre a ponta. No entanto ainda ocorre o achatamento da ponta no interior do "U". Isso ocorre, novamente, devido à anisotropia no processo de deposição. Um aspecto positivo é que a influência da ponta demora bem mais para desaparecer, ainda estando presente após 50 passos. Os resultados para o exemplo acima são muito bons. Somente após 30 passos é possível notar visualmente diferenças entre os resultados do autômato contínuo e do autômato de Huygens. O achatamento do filme sobre a ponta é a maior diferença entre os dois autômatos.

O valor de 0,83 para o coeficiente de crescimento no sentido diagonal da equação (9) é bom para deposições em pequenas redes, como no exemplo utilizado. Quando aumentamos as dimensões da rede, para valores bem maiores que no exemplo

acima, esse valor não gera resultados tão bons. Como a determinação do valor pelos autores é visual, não é possível ajustar dinamicamente este valor.

Apesar dos autores afirmarem que esse algoritmo aproxima melhor um crescimento com o aumento da rede, isso não ocorre para a regra descrita no artigo (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2001). O gráfico abaixo mostra uma comparação entre o autômato celular simples com a vizinhança de Moore e o autômato celular contínuo com a regra na equação (9). Pode-se visualizar que os resultados para ambos os autômatos é próximo, principalmente quando aumenta-se a rede.

Gráfico 3 – Células erradas e o erro relativo para os autômatos celulares simples com vizinhança de Moore e contínuo com a regra da equação (9).

Os resultados foram obtidos comparando a área depositada com um disco de raio igual ao número de passos vezes o tamanho da célula lateral. Células consideradas erradas são células no estado Filme fora do disco ou células no estado Vazio dentro

do disco. O número de células erradas estaria sempre crescendo, apesar do disco cada vez mais se aproximar do formato ideal.

O Gráfico 3 também mostra a razão entre o número de células erradas e o total de células contidas dentro do disco ideal. Esse valor é mais útil porque o simples fato do número de células erradas crescer não indica um erro crescente do autômato celular. Como exemplo, imagine um autômato celular que gera um disco com raio que é uma célula unitária maior que o raio do disco ideal. O número de células erradas cresce com o aumento do raio, mas a razão entre o número de células erradas e o número de células dentro do disco tende a zero nesse caso.

O Gráfico 3 mostra que, para ambos os autômatos celulares, a razão entre as áreas se aproxima do valor 0,27. Esse valor, por sua vez, é próximo ao da razão entre a diferença da área do quadrado que circunscreve o disco e a área do disco (área A1 na Figura 37) e a área do disco (área A2 na Figura 37).

Figura 37 - Definições das áreas A1 e A2.

Abaixo calculamos o valor dessa razão.

$$\frac{A1}{A2} = \frac{(2R)^2 - \pi R^2}{\pi R^2} = \frac{(4 - \pi)R^2}{\pi R^2} = \frac{4 - \pi}{\pi} \cong 0,2732 \quad (18)$$

Uma implementação do autômato celular utilizando a modificação apresentada na equação (10) também foi feita. Uma simulação com essa variante e a análise dos resultados também foi realizada. Como é possível observar no Gráfico 4, um valor baixo para o tempo T melhora o desempenho do autômato celular contínuo. No

entanto essa modificação também aumenta o número de passos necessários para se obter um filme de mesma espessura. Outra dificuldade que surge dessa modificação é definir a taxa de deposição do filme utilizando esse método. Isso ocorre porque ao mudar o valor de T o número de passos necessários para depositar uma célula também muda.

Gráfico 4 - Comparação entre os erros do autômato celular com vizinhança de Moore e o autômato contínuo usando um coeficiente $T = 0,01$.

Nota-se o mesmo comportamento das curvas de erro com o autômato contínuo com a regra da equação (10). Após atingir um raio de 100 células, a melhora na aproximação da frente circular com o aumento da rede é desprezível. Isso se deve à anisotropia do próprio autômato celular. A diminuição do passo de tempo é incapaz de melhorar a aproximação. Isso pode ser verificado modificando a implementação e utilizando passos de tempo variáveis. Nessa variação, antes de rodar um passo, determinamos o maior valor para o passo de tempo de forma que o valor do estado de uma célula nunca ultrapasse o valor máximo de 1,0. O Gráfico 5 mostra que a

diferença entre os resultados obtidos utilizando um passo de tempo variável e um passo fixo de 0,01 são muito parecidos.

Gráfico 5 - Comparação entre utilização de um valor fixo para T e um valor variável para T .

Os valores dos erros, tanto no Gráfico 4 quanto no Gráfico 5, indicam a comparação feita quando o raio da área depositada atinge o valor marcado no eixo horizontal pela primeira vez. Assim os erros acima não apresentam o menor erro que pode ser obtido. No entanto, se o objetivo é simular um crescimento isotrópico, esse método é melhor para a comparação dos resultados visto que a distância do centro até a superfície deve ser, idealmente, igual para todas as células da superfície.

As diferenças entre os valores para os dois autômatos nas primeiras 50 medidas mostram que, com $T=0,01$ existem células cujo estado ultrapassa o valor máximo de 1,0. Contudo fica claro pelo comportamento para raios maiores que não é essa a principal fonte do erro nesse autômato celular. No Gráfico 4, fica claro que o uso de um valor pequeno, menor que 1, para o coeficiente T melhora os resultados gerados por esse autômato celular contínuo. Entretanto a quantidade de trabalho e o tempo

necessários para gerar os resultados também aumenta. Usando um valor de $T = 0,01$ são necessários 6816 passos para atingir o raio de 275 células. O autômato celular com passo variável necessita de 6920 passos para atingir o mesmo raio. Os dois autômatos mais simples necessitam de apenas 275 passos visto que nas direções vertical e horizontal depositam uma célula por passo.

O autômato celular para simulação da oxidação térmica descrito na seção 2.5 também foi implementado. A Figura 38 mostra um exemplo gerado com esse autômato.

Figura 38 – Evolução do autômato celular para oxidação, dada de 20 em vinte passos. O substrato de Si está em cinza e o óxido em azul.

A região em cinza é o substrato de Si. A região em branco é o ar e a região azul é o óxido. Uma representação do material de mascaramento utilizado para essa simulação foi adicionado posteriormente nas imagens. O material de mascaramento ocupa a área delimitada pelas linhas pontilhadas.

O autômato celular utilizado para gerar as imagens da Figura 38 não altera a taxa de crescimento com o tempo. Seguindo o modelo Deal-Grove para oxidação térmica do Si (DEAL e GROVE, 1965), o processo de oxidação depende da difusão de oxigênio do meio para o óxido e da difusão do oxigênio dentro do óxido, que deve chegar até a interface óxido/silício, onde a reação de oxidação efetivamente ocorre. Assim a taxa de crescimento varia com a espessura do óxido já crescido. O simulador não considera o material de mascaramento utilizado para a oxidação. Dado o objetivo de ter a possibilidade de simular diversos processos em um mesmo substrato, isso é um problema grave. Por outro lado, ajustando os coeficientes seguindo as restrições (6), (7), e (8), a razão entre a porção de óxido acima da superfície e abaixo da superfície é de $17/15 \cong 54/46$. Esse último valor é o esperado para uma oxidação térmica.

Esse autômato celular para oxidação, apesar de estudado, não foi implementado no simMEMS. Essa decisão foi tomada por duas razões. Primeiramente, devido às diferenças entre os resultados gerados pelo autômato e os resultados esperados da literatura. Talvez mais importante é a necessidade de modificar significativamente as regras para permitir o deslocamento de filmes depositados sobre o substrato para simular o crescimento do óxido para simulação multi-processo.

Quanto aos autômatos celulares para deposição que foram implementados, cabe mencionar, além dos resultados já apresentados, alguns pontos de interesse. Primeiramente, o primeiro trabalho mencionado (STRASSER e SELBERHERR, 1995) descreve um método de simulação que serve tanto para deposição como para corrosão. Esse comportamento é entendido como o normal porque, em ambos os casos, a simulação é de um mesmo processo: a evolução de uma superfície. A diferença entre a corrosão e a deposição é apenas o ponto de referência; na deposição considera-se que a superfície que evolui é a do filme e na corrosão a superfície que evolui é a do "vazio". Portanto a descrição de dois autômatos celulares bem diferentes para corrosão e deposição, como os descritos em (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2002) sem uma explicação teórica para a variação é estranha.

Existe outra premissa adotada nos algoritmos descritos em (SIRAKOULIS, KARAFYLLIDIS e THANAILAKIS, 2002) que é equivocada. Os autores consideram que quanto maior o número de vizinhos de uma célula, mais rápido deve ser a

corrosão ou deposição nessa célula. O exemplo da Figura 39, que considera um crescimento isotrópico seguindo o princípio de Huygens, demonstra que, apesar da célula central ter três vizinhos, sua taxa de preenchimento não deve ser, necessariamente, maior que a de uma célula com apenas um ou dois vizinho. Nas imagens da Figura 39 abaixo temos as ondas de matéria partindo do centro das células em cinza. Ao lado é possível ver o envelope formado. A cor indica a fonte da onda adotado como envelope em cada ponto. Note que nenhuma parte do envelope uma célula em branco é coberta primeiro por uma onda de matéria que não tenha origem na célula diretamente abaixo da própria célula branca.

Figura 39- Na imagem da esquerda são mostradas as ondas de matéria partindo do centro de três células. Na direita, os envelopes para diferentes instantes formados por todas as ondas geradas são destacados. A cor indica de qual célula surge a onda que gerou cada seção do envelope.

Fica claro, analisando a deposição na célula acima da célula com o centro vermelho, que a presença de vizinhos ao sudeste e sudoeste (células com centros azul e verde, respectivamente) não influencia a velocidade de crescimento do filme na célula. No entanto existem situações onde a presença de vizinhos efetivamente deve acelerar o preenchimento da célula. Um exemplo disso é quando temos vizinhos em lados opostos da célula (norte e sul, oeste e leste). Isso é, um autômato celular deve considerar a possibilidade de aumento da taxa de deposição devido à vizinhança, mas esse aumento não é ligado ao número de vizinhos que uma célula possui, mas sim à disposição desses vizinhos.

Pelas comparações feitas, vemos que para redes grandes, o autômato celular simples é equivalente ao autômato contínuo com $T = 1$. Como o autômato celular contínuo requer mais memória e processamento, optamos por não utilizá-lo. O

mesmo autômato contínuo com $T \ll 1$ gera melhores resultados. Porém esse autômato celular também não foi implementado no simMEMS por duas razões. Primeiramente pelo custo computacional adicional, que em redes tridimensionais é ainda maior. A segunda razão é a dificuldade em determinar a taxa de deposição, que dificulta uma eventual aplicação de parâmetros físicos.

O autômato celular de Huygens foi implementado no CellularAutomata3D em três dimensões. A implementação permite depositar e corroer filmes usando o autômato celular. Um processo de fotolitografia também foi desenvolvido. O autômato simples com vizinhança de Moore foi implementado em conjunto. Assim é possível realizar deposições e corrosões com ambos os autômatos no mesmo substrato. O autômato celular de Huygens, no entanto, se mostrou bastante lento em três dimensões. Foi feita uma escolha então por implementar primeiramente o autômato simples no simMEMS. A Figura 40 mostra o que pode ser feito nesse implementação conjunta dos autômatos. Os processos de deposição e corrosão seletiva de filmes foram utilizados para criar um pequeno rotor que pode girar livremente ao redor do seu eixo.

(a)

(b)

(c)

(d)

Figura 40 - Passos de fabricação de um rotor.

O processo de fabricação, visto na Figura 40, envolve diversos passos. Primeiramente uma camada de sacrifício é depositada (a) e fotogravada (b). Essa primeira camada garante que o corpo do rotor não fique preso ao substrato. Em seguida uma camada estrutural é depositada (c). Essa camada formará o rotor e é fotogravada para dar o formato ao rotor (d). Em seguida uma segunda camada de sacrifício é depositada (e) e fotogravada (f). Essa camada impede que o rotor fique

preso ao eixo. A segunda camada estrutural é depositada em seguida (g). Essa camada forma o eixo. Essa camada também é fotogravada (e) com uma máscara que permite que o topo dessa camada prenda o rotor, como pode ser visto em (l). A segunda camada de sacrifício é removida em seguida (j). Finalmente a primeira camada de sacrifício é também removida (j), liberando o rotor. O resultado final é um rotor, em oliva, que pode girar em torno do eixo, em vermelho. O rotor também está preso ao substrato pelo eixo. Assim é possível, por exemplo, girar o rotor sem que esse se desprenda do substrato.

4.2 simMEMS

Nesta seção serão descritos os autômatos celulares implementados no software simMEMS. Também serão apresentados alguns resultados gerados pelo simMEMS. Comparações entre os resultados gerados pelo software e resultados experimentais também foram realizadas.

4.2.1 Os Autômatos Celulares

Dos diversos autômatos celulares testados com o auxílio do software CellularAutomata3D, apenas alguns foram implementados no simMEMS. Utilizamos o simulador de corrosão úmida bem como simulador de DRIE. Para a simulação de deposição e corrosão de filmes, utilizamos apenas o autômato simples com a vizinhança de Moore. Não implementamos o autômato celular de Huygens descrito porque, principalmente em três dimensões, ele é bastante lento. Nenhum dos autômatos contínuos foram implementados. Os simuladores se mostraram mais complexos de implementar, e mais lentos, mas não geram resultados que justifiquem

o trabalho adicional. A seguir descreveremos de maneira mais extensa a maneira como os autômatos escolhidos foram implementados.

4.2.1.1 Simulador da Corrosão Úmida do Si

O simulador de corrosão úmida de silício é baseado no autômato celular conhecido como BCA(GOSÁLVEZ, XING, *et al.*, 2008). Esse autômato roda sobre a estrutura cristalina do silício. O silício cristalino possui uma estrutura cúbica de diamante, como mostrado na Figura 41. É possível executar o autômato celular diretamente sobre essa estrutura. Nesse caso cada átomo seria uma célula do autômato e as ligações indicam as vizinhanças.

Figura 41 - Célula de diamante do Si cristalino.

No entanto essa estrutura é inconveniente para outros autômatos celulares. Filmes depositados sobre esse substrato teriam a mesma estrutura cristalina que o silício, o que não faria sentido em todos os casos. Seria mais útil utilizar uma estrutura cúbica.

Uma maneira simples de utilizar uma estrutura cúbica é mapear a estrutura cristalina do Si em uma matriz tridimensional de células. Esse raciocínio pode ser acompanhado na figura abaixo. O cristal de Si nesse caso é girado 45° em relação ao cristal da Figura 41 para se alinhar com a orientação padrão de uma lâmina (100) de Si.

Figura 42 - Conversão da rede cristalina do silício em uma rede cúbica. Nas figuras acima vemos um vista do topo e da lateral, respectivamente, do cristal. As figuras abaixo mostram em qual camada fica cada átomo. As camadas estão indicadas na visão lateral do cristal.

Pela Figura 42 podemos facilmente perceber que o cristal pode ser gerado repetindo-se o bloco de 2x2x4 células mostrado em cinza. Essa representação também apresenta algumas dificuldades para deposição de filmes. Seria necessário modificar as regras dos autômatos para impedir a deposição de filmes nos espaços vazios entre os átomos nessa estrutura. Outro problema é relativo ao consumo de memória para a simulação. Essa estrutura precisa de muitas células vazias para representar uma pequena parte do cristal. Das 16 células apenas 4, 25% do total, representam átomos.

Outra representação que é mais compacta armazena apenas os átomos do cristal. Utilizamos atualmente esta representação para simulação. A representação compacta evita o problema da deposição de filmes além de remover as células vazias desnecessárias. No entanto é importante notar que ao remover as células vazias estamos deslocando os átomos. Assim não podemos interpretar os resultados da simulação observando diretamente a rede cúbica. Podemos, contudo, reproduzir a estrutura cristalina do Si a partir dessa matriz e depois fazer a análise dos resultados. A estrutura compacta é feita agrupando as quatro células (marcadas em cinza na Figura 42) de cada camada em uma única célula.

A corrosão é simulada analisando a vizinhança dos átomos de Si. Primeiramente analisamos os primeiros vizinhos. Esses são os átomos da rede que estão ligados

ao átomo sendo analisado (Figura 44(a)). Para simular corretamente a corrosão é preciso analisar não apenas os primeiros vizinhos, mas também os segundos vizinhos dos átomos (THAN e BÜTTGENBACH, 1994). Essa necessidade pode ser visualizada de maneira fácil considerando os planos cristalográficos (110) e (111) na Figura 43.

Figura 43 – Planos cristalográficos (110) em (a) e (111) em (b) do Si.

Em ambos os casos átomos na superfície desses planos possuem três primeiros vizinhos. No entanto as taxas de corrosão desses planos são bastante diferentes (SEIDEL, CSEPREGI, *et al.*, 1990).

Os segundos vizinhos são os átomos que estão ligados aos primeiros vizinhos do átomo sendo analisado (Figura 44(b)). Se levarmos em conta essa vizinhança estendida, é possível diferenciar a família de planos {110} da família de planos {111}. O plano (111) possui três vizinhos diretos. No entanto todos esses três vizinhos estão no substrato, fora da superfície. Isso pode ser visto na Figura 43. Átomos nesse plano, portanto, possuem nove segundos vizinhos. Já os átomos no plano (110), apesar de possuírem três vizinhos diretos, possuem apenas sete segundos vizinhos. Isso ocorre porque dois desses vizinhos encontram-se também na superfície do plano (110).

Figura 44 – Primeiros (a) e segundos (b) vizinhos usados na simulação da corrosão úmida de Si.

Portanto, analisando os segundos vizinhos é possível determinar se os primeiros vizinhos de um átomo na superfície se encontram ou não também na superfície do cristal. As regras do BCA são listadas abaixo. Ao invés de definir as regras em função do número de primeiros vizinhos na superfície, como no artigo original (THAN e BÜTTGENBACH, 1994), as regras foram definidas em função do número de primeiros e segundos vizinhos. Isso foi feito porque esses valores são os que devem efetivamente ser verificados. Abaixo estão as regras do autômato:

1. Células no estado **átomo** com nenhum, um ou dois primeiros vizinhos mudam para o estado **vazio**.
2. Células no estado **átomo** com três primeiros vizinhos e até oito segundos vizinhos mudam para o estado **vazio**.
3. Células em estados diferente de **átomo** ou em outras configurações não mudam de estado.

4.2.1.2 Simulador de DRIE

O autômato celular para simular processos de DRIE também é simples. No entanto é capaz de simular o comportamento ideal desse tipo de corrosão. O autômato usa uma vizinhança reduzida composta pela célula acima da célula sendo analisada. O usuário não define uma máscara como no simulador do software CellularAutomata3D. O mascaramento nesse autômato é feito pelos filmes depositados. As regras do autômato celular também são simples. A regra de evolução é:

1. Se a célula se encontra no estado Si e a célula acima se encontra no estado vazio, a célula é removida.
2. Células em outros estados ou em outras configurações não são removidas.

O estado Si corresponde a uma célula preenchida por silício. A corrosão nesse caso é anisotrópica, mas ocorre só na direção vertical.

4.2.1.3 Simulador de Deposição e Corrosão de Filmes

A simulação de deposição de filmes foi feita utilizando o mesmo autômato celular tridimensional com a vizinhança de Moore expandida para três dimensões descrito no item 4.1.2. O autômato deve rodar na mesma rede que o autômato de corrosão úmida. Logo a rede não é cúbica. Isso modifica um pouco a anisotropia do autômato em relação ao gráfico mostrado previamente. O software atualmente permite a deposição de 12 tipos de filme; o usuário seleciona em uma janela qual tipo de filme deseja depositar. Cada filme tem uma cor diferente para ajudar a identificação do filme. As cores podem ser editadas pelo usuário. As regras do autômato são mostradas abaixo:

1. Se a célula se encontra no estado Vazio e todas as células na vizinhança estão no estado Vazio, a célula continua no estado Vazio.
2. Se a célula se encontra no estado Vazio e alguma célula na vizinhança não está no estado Vazio, a célula muda para o estado Filme.
3. Células em qualquer outro estado não mudam de estado.

O estado Filme acima indica o estado correspondente ao filme selecionado para deposição pelo usuário. O autômato celular utilizado para corrosão de filmes é semelhante ao autômato de deposição de filmes. Utiliza a mesma vizinhança. As regras, apresentadas abaixo, são semelhantes, apenas mudamos o ponto de referência. O usuário também seleciona qual filme deve ser removido. Nesse caso o estado Filme se refere agora ao filme selecionado para corrosão pelo usuário.

1. Se a célula se encontra no estado Filme e todas as células na vizinhança não estão no estado Vazio, a célula continua no estado Filme.
2. Se a célula se encontra no estado Filme e alguma célula na vizinhança está no estado Vazio, a célula muda para o estado Vazio.
3. Células em qualquer outro estado não mudam de estado.

4.2.1.4 Simulação de Fotolitografia

A simulação de processos de fotogração pode ser feita utilizando um outro autômato celular que corrói filmes. Esse autômato utiliza uma vizinhança mais restrita, considerando apenas as células acima e abaixo da célula sendo analisada. A regra é semelhante à corrosão de filmes embora utiliza uma outra variável, que representa a máscara fotolitográfica. O software permite ao usuário selecionar duas máscaras, uma para a parte inferior e uma para a parte superior do substrato. A máscara pode ser definida de duas maneiras. O usuário pode definir máscaras selecionando uma imagem ou desenhando a máscara no simMEMS. A seguir descrevemos as regras do autômato celular.

1. Se a célula se encontra no estado Filme, a célula acima dela se encontra no estado Vazio e a máscara superior não protege a célula, o estado da célula muda para Vazio.
2. Se a célula se encontra no estado Filme, a célula abaixo dela se encontra no estado Vazio e a máscara inferior não protege a célula, o estado da célula muda para Vazio.
3. Células em qualquer outro estado não mudam de estado ou em qualquer outra situação não são removidas.

O autômato celular acima é simples, porém representa a maneira ideal para um processo de fotolitografia. Novamente o estado Filme acima se refere ao estado correspondente ao filme selecionado para fotogração pelo usuário.

4.2.1.5 Processo de Planarização

O simMEMS também permite realizar uma planarização. Esse processo é utilizado na fabricação de alguns circuitos quando se utiliza um processo CMOS. No caso do simMEMS, o usuário pode definir uma altura mínima e uma altura máxima para a planarização. Todo o material, seja ele um filme ou o próprio Si, é removido abaixo da altura mínima ou acima da altura máxima. Isso é equivalente a realizar uma planarização no topo e na base do substrato.

4.2.2 Visualização dos Resultados no simMEMS

Os mecanismos de visualização do volume de células no simMEMS são diferentes dos métodos utilizados no CellularAutomata3D. Não utilizamos o algoritmo *marching cubes* para gerar a superfície. A superfície agora é gerada analisando as vizinhanças das células para determinar quais faces da célula precisam ser desenhadas. De maneira simples, a face de uma célula só precisa ser desenhada se a célula vizinha que compartilha essa face for preenchida por um material diferente. A geração dessa superfície é mais rápida que a utilização do *marching cubes*. Para evitar a geração dessa superfície cada vez que a tela é desenhada, utilizamos *display lists* (BENJAMIN, HAEMEL e WRIGHT JR., 2007) para armazenar as superfícies geradas.

O software permite desenhar os filmes como objetos sólidos ou objetos transparentes. Também é possível visualizar as arestas das células da superfície. O substrato de Si também pode ser visualizado como átomos. O usuário pode escolher entre visualizações que mostram somente os átomos da superfície, os átomos da superfície e os átomos no limite do substrato, ou todos os átomos do substrato de Si.

4.2.3 Resultados Gerados pelo simMEMS

Como descrevemos acima, o software é capaz de diversos processos de microfabricação. A seguir mostramos alguns dispositivos cujo processo de fabricação pode ser simulado pelo software. O primeiro exemplo é uma microponeta fabricada através da corrosão anisotrópica do Si utilizando um material de mascaramento em formato de cruz. Na Figura 45 apresentamos tanto resultados do simulador como experimentais. Nos resultados gerados pelo simMEMS novamente utilizamos a visualização atomística dos resultados para facilitar a identificação dos planos cristalográficos. Nas imagens os átomos que possuem um primeiro vizinho estão em amarelo, os átomos que possuem dois primeiros vizinhos estão em verde e os átomos que possuem três primeiros vizinhos estão em azul.

Esse exemplo é interessante porque podemos comparar bem os resultados obtidos experimentalmente com os resultados obtidos pelo BCA. Para tanto, é preciso determinar dois fatores: o tamanho das células e a duração de um passo do autômato celular. Na Figura 45 o braço da cruz possui uma largura de $50\mu\text{m}$. Para simular essa corrosão no BCA, deve-se criar um braço que também tenha $50\mu\text{m}$. No exemplo o braço tem uma espessura de 100 células. Portanto para terem o mesmo tamanho, a célula deve medir $0,5\mu\text{m}$. A taxa de corrosão dos planos envolvidos não estava disponível, no entanto sabemos que a corrosão do Si sob os braços levou aproximadamente 20 minutos. No simulador, esse mesmo processo levou 53 passos. Esses dados permitem obter a duração de um passo de tempo: 22,6 segundos.

Figura 45 - Fabricação de uma micropona utilizando material de mascaramento em formato de cruz. Na esquerda são apresentados os resultados do simulador e na direita os resultados experimentais.

Com essas medidas podemos obter as demais medidas de tempo e tamanho necessários para comparar os resultados experimental e simulado. Analisando a estrutura simulada após 94 passos (cerca de 35 minutos de corrosão), pode-se ver claramente que restou um pequeno triângulo na base da microponta. Esse triângulo aparece também no resultado experimental. Nesse instante, as bases do triângulo experimental e simulado têm o mesmo tamanho, cerca de $10\mu\text{m}$. É possível ver também que o topo da microponta está facetado de maneira similar em ambos os casos. Contudo a microponta simulada apresenta um plano único nas laterais, enquanto o resultado experimental apresenta dois planos.

As duas últimas imagens da Figura 45 apresentam resultados que são bastante interessantes. Primeiramente, em ambos os casos as micropontas já se descolaram do material de mascaramento. Isso leva 130 passos no simulador, o equivalente a 49 minutos de corrosão. Isso é próximo do valor real necessário para a ponta vista no resultado experimental, que leva pouco mais de 50 minutos.

A altura das duas micropontas é também bastante próxima. A microponta simulada possui $64\mu\text{m}$ enquanto a microponta real possui $52\mu\text{m}$. A altura final depende fortemente do tamanho dos braços. Esse valor próximo significa que o simulador pode ser utilizado, por exemplo, para estudar o tamanho da ponta para uma dada largura de braço. Isso permitiria gerar uma microponta com a altura desejada com um número menor de experimentos.

Inicialmente é feito um processo DRIE para a coluna vista no passo 2 da Figura 46. Isso é semelhante ao processo de *dicing* mecânico realizado em (SHIKIDA, ODAGAKI, *et al.*, 2004). Em seguida um processo de corrosão em KOH para gerar as micropontas é realizado. A evolução dessa corrosão (passos 3 a 5) é bastante semelhante aos resultados experimentais (imagens (c) a (f)). As imagens retratam os átomos na superfície do substrato. Os átomos são coloridos em função do número de vizinhos. Átomos amarelos possuem um primeiro vizinho, átomos verdes possuem dois primeiros vizinhos e átomos azuis possuem três primeiros vizinhos. O material de mascaramento é definido pela face vermelha semitransparente. O substrato se encontra abaixo do plano verde da imagem (1). Apenas os átomos da superfície exposta foram desenhados.

Figura 46 – Simulação de uma processo DRIE seguido de uma corrosão úmida em KOH. No detalhe, processo semelhante usando “dicing” mecânico ao invés de DRIE. A imagem em (6) é de (SHIKIDA, ODAGAKI, *et al.*, 2004).

Na Figura 47 é apresentado outro processo que gera uma microponeta com um formato de caneta. O método seguido é similar ao descrito em (SHIKIDA, ANDO, *et al.*, 2004). Esse tipo de microponeta tem aplicações médicas e são descartáveis. Para serem comercialmente viáveis, seu processo de fabricação deve ser barato.

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

(i)

(j)

(k)

(l)

Figura 47 – Fabricação de uma micropona utilizando o software. As imagens mostram os resultados da corrosão úmida em As imagens dos resultados experimentais são de (SHIKIDA, ANDO, *et al.*, 2004).

Os autores sugerem um método de fabricação que permite alterar o formato das pontas e não requer processos caros como fotolitografia ou corrosões por plasma. Em vez disso um processo mecânico para fabricar colunas quadradas (*dicing*) é utilizado para formar as colunas. Em seguida pequenos degraus são feitos, também mecanicamente, em dois lados das colunas. O tamanho desses degraus é essencial para definir o formato final da microponeta. Em seguida todo o substrato é corroído em uma solução de KOH. Esse método de fabricação pode ser inteiramente simulado no simMEMS substituindo os processos mecânicos por DRIE.

Na Figura 47 os passos de fabricação de uma microponeta no simMEMS são reproduzidos. O *dicing* mecânico é substituído por um processo DRIE. Primeiro um filme é depositado e fotogravado para dar o formato da coluna. Em seguida um processo DRIE é utilizado para criar a coluna (Figura 47(a)). Ao final desse processo o material de mascaramento é corroído novamente para mudar o seu formato. Um processo DRIE é utilizado novamente para fazer os degraus (Figura 47(b)). Após esse passo o restante do filme utilizado como material de mascaramento é corroído (Figura 47(c)). Depois de removido o material de mascaramento, a corrosão úmida do substrato é simulada para gerar as micropontas (Figura 47(e), (g), (i) e (k)).

Comparando os resultados gerados pelo simMEMS com os resultados experimentais é possível verificar, novamente, que os formatos dos mesmos são bastante próximos. A base da ponta apresenta oito lados em ambas as imagens. Também é possível notar que a coluna central tem uma orientação semelhante nas duas imagens. O topo das pontas também é bastante parecido. É difícil visualizar nos resultados experimentais, mas o topo da caneta é formado pelo encontro de dois planos da família $\{311\}$. Isso pode ser visto mais claramente no resultado do simulador, onde as duas faces que formam o topo da caneta são efetivamente pertencentes à família de planos $\{311\}$. A Figura 48 mostra detalhes da ponta. Os átomos amarelos possuem um primeiro vizinho, os verdes dois primeiros vizinhos e os laranjas três primeiros vizinhos e menos de nove segundos vizinhos. Os átomos azul turquesa são átomos que não estão na superfície.

Figura 48 – Detalhes da ponta. As duas imagens da esquerda mostram a ponta ampliada vista como átomos e como as células do autômato celular, respectivamente. A imagem da ponta experimental é de SHIKIDA, ANDO, *et al.*, 2004. A última imagem mostra os dois planos que formam o topo da microponta.

Quanto às medidas, o autômato foi ajustado para a taxa de corrosão do plano (110), que no experimento era próxima de $1,375 \mu\text{m}/\text{minuto}$. Isso leva, obrigatoriamente, à uma medida correta da largura final das agulhas, de $10 \mu\text{m}$. Os formatos para corrosão após 10, 20, 30 e 40 minutos são parecidos com os experimentais. A altura da microponta, no entanto, é diferente. Enquanto a microponta real media cerca de $230 \mu\text{m}$, a microponta simulada possui $291 \mu\text{m}$. Isso pode ser devido à diferença entre as taxas de corrosão dos planos {311} no simulador e no experimento. Como apenas a taxa de corrosão do plano (110) pode ser ajustada, os valores das taxas de corrosão para os planos {100} e {311} não necessariamente estarão corretos.

O simMEMS também permite realizar a corrosão do substrato por ambos os lados. Os passos da fabricação de uma estrutura que pode ser utilizada como um acelerômetro são reproduzidos na Figura 49. Um filme é depositado sobre ambos os lados do substrato, o filme é fotogravado para criar as aberturas para liberar a viga engastada. Em seguida um segundo filme é depositado e fotogravado. Esse filme formará a trilha e os contatos que permitirão medir uma variação de resistência devido à movimentação da viga engastada. Finalmente a corrosão úmida é utilizada para liberar a estrutura.

Figura 49 - Passos de fabricação de um acelerômetro utilizando microfabricação em substrato e os resultados experimentais de uma corrosão.

Outro processo que o simMEMS simula é a deposição de filmes sobre substratos já corroídos. Um exemplo complexo da deposição de filmes sobre um substrato corroído se encontra na Figura 50.

(i)

(j)

(k)

(l)

(m)

(n)

(o)

(p)

(q)

(r)

Figura 50 - Exemplo da fabricação de uma micro-lâmpada incandescente através da deposição de um filme metálico sobre um microponta de Si.

O processo começa com o substrato sobre o qual um material de mascaramento é depositado (Figura 50(a) e(b)). Em seguida o material de mascaramento é fotogravado (Figura 50(c), (d) e (e)). Uma micropona é fabricada através da corrosão úmida do Si utilizando esse material de mascaramento (Figura 50(f), (g), (h), (i) e (j)). Esse material de mascaramento, após criada a micropona, é removido. Sobre a micropona gerada um filme de óxido é crescido (Figura 50(k)). Um filme metálico é depositado (Figura 50(l)) e fotogravado (Figura 50(m), (n), (o) e (p)) de maneira a deixar apenas uma trilha que passa pelo topo da micropona (Figura 50(q)). Note que para essa fotografação um processo mais parecido ao real foi utilizado. Um fotorresiste é depositado, revelado e usado como material de mascaramento para o metal. Em seguida uma nova camada de óxido é crescida para proteger a trilha da oxidação. Na Figura 50(r) o óxido foi deixado semitransparente para permitir visualizar o metal entre as camadas de óxido.

Devido à maior dificuldade em depositar o filme sobre a ponta em relação ao resto do substrato, o filme depositado sobre a ponta é mais fino. Ao passar uma corrente pela trilha, essa região mais fina apresenta maior resistência elétrica e aquece, eventualmente emitindo luz. Para que o condutor aqueça e para evitar a oxidação do metal, uma camada de óxido é depositada antes da deposição do metal e depois da sua fotografação. A camada inferior evita transferência de calor para o Si e a camada de cima evita a oxidação. A forma da micropona ajuda a espalhar a luz gerada em diversas direções. Esse espalhamento permite um melhor acoplamento da luz gerada com dispositivos óticos presentes na lâmina de Si, como um guia de onda. O simulador permite simular, sem problemas, a deposição e a corrosão do filme sobre a superfície altamente irregular da ponta. Devido ao autômato simples atualmente em uso, no entanto, não é possível visualizar o afinamento do metal no ápice da micropona.

O simMEMS pode ser utilizado também para simular processos de microfabricação em superfície. Nesses processos não é preciso utilizar o BCA ou o simulador de DRIE. Como exemplo, mostramos a fabricação de uma viga engastada utilizando processos CMOS. Esse tipo de dispositivo é bastante interessante porque, por usar principalmente processos CMOS, os dispositivos podem ser fabricados em *foundries*. Essa possibilidade garante alta reprodutibilidade no processo de fabricação além de permitir a integração da eletrônica de controle no dispositivo.

No caso da viga engastada, o processo de fabricação envolve depositar camadas sobrepostas de trilhas de alumínio. Essas trilhas são conectadas entre si por vias de tungstênio. Depois de terminado o processo CMOS, as vigas estarão presas pelo óxido crescido entre as trilhas de alumínio. A Figura 51 mostra o processo CMOS utilizado e os passos para fabricação. O processo consiste em uma oxidação, fotogravação do óxido, deposição do metal da via, fotogravação da via, deposição do metal da via, fotogravação das trilhas seguida de uma nova oxidação. O processo se repete e após a oxidação realizamos uma planarização. As duas imagens na última linha mostram apenas o resultado de todos os passos para adição de uma nova camada às vigas. A última imagem mostra a mesma estrutura que a penúltima imagem com o óxido transparente para facilitar a visualização das vias.

Figura 51 – Passos para fabricação de uma viga engastada utilizando um processo CMOS. As imagens estão na sequência da esquerda para a direita e de cima para baixo. O si está em cinza, o óxido em laranja, as vias de tungstênio em verde e as trilhas de alumínio em azul. As máscaras fotolitográficas são representadas em vermelho transparente.

O simMEMS ainda tem limitações sobre o tamanho total dos substratos. Isso impede o uso de um grande número de células para melhor representar o formato redondo das vias. Porém mesmo com tamanhos pequenos, é possível simular um processo bastante complexo, que inclui diversos passos. Além disso, é também possível simular a corrosão do óxido para liberação da estrutura. A abaixo mostra de maneira simplificada essa remoção. Nas imagens as trilhas foram deixadas transparentes para facilitar a visualização do óxido. É possível notar como o óxido entre as vias e sob as trilhas demora mais tempo para ser removido.

Figura 52 – Remoção do óxido para liberação das vigas engastadas. É possível notar nas imagens (a)-(f) como a corrosão do óxido prossegue. Na imagem (a) fica bem claro como a corrosão prossegue tanto na vertical como na horizontal para remover o óxido entre as vias. A imagem (g) mostra o resultado final do processo: as vigas engastadas de diferentes tamanhos liberadas. Resultado experimental para um processo semelhante(DAI, 2006) é mostrado em (h).

A mudança do uso de *marching cubes* para simplesmente desenhar os lados das células, apesar de acelerar o desenho da tela, também prejudica a noção de uma estrutura com lados redondos, como as vias. No entanto o software reproduz bem os resultados. Além disso, fica claro que o simMEMS permite a visualização prévia de um processo bastante complicado, que inclui diversos passos de fabricação e o uso de várias máscaras fotolitográficas para geração de três cantilevers com comprimentos e alturas diferentes.

5 Conclusões

Foram estudados diversos autômatos celulares para simulação da deposição e corrosão de filmes. Um software, o CellularAutomata3D, foi desenvolvido para implementar e testar esses autômatos celulares. O CellularAutomata3D se mostrou uma boa ferramenta para esse desenvolvimento. A estrutura do software permite o desenvolvimento rápido de implementações de autômatos celulares. Ao mesmo tempo, sua relativa simplicidade permite também a fácil modificação desses autômatos. Isso, por sua vez, permitiu gerar as comparações e gráficos do item 4.1.2.

Comparando os autômatos celulares, descobrimos que o desempenho dos autômatos celulares contínuos estudados estava aquém do esperado. A escolha da utilização apenas dos autômatos celulares discretos foi então tomada. Novamente o CellularAutomata3D foi de grande valia pois nele foi desenvolvido a estrutura básica dos autômatos utilizada posteriormente no simMEMS. Isso é bom porque permite a prototipagem de autômatos celulares para o simMEMS no CellularAutomata3D.

O simMEMS, por sua vez, sofreu grandes mudanças. A interface gráfica foi refeita utilizando a biblioteca Qt. A biblioteca antiga, a wxWidgets, apresentava diversos problemas. Esses iam desde inconsistências no comportamento do software entre plataformas à falhas na documentação. A biblioteca também não possuía boas ferramentas de desenvolvimento de janelas. O Qt não apresentou esses problemas durante o desenvolvimento do CellularAutomata3D, que serviu em parte como teste para a biblioteca Qt.

A nova interface facilitou o uso de janelas. No futuro as funções do simMEMS devem depender cada vez mais de entradas via janelas, diminuindo a necessidade do console. A mudança impediu, pelo tempo necessário, a implementação de todas as funcionalidades existentes antigamente no simMEMS. Apenas as funções de desenho de estruturas básicas e operações booleanas foram transferidas. No entanto também permitiu que diversas melhorias quanto ao gerenciamento de memória fossem feitas. O resultado é um software que até agora tem se mostrado mais robusto que as versões anteriores.

Ainda existem problemas com o simMEMS atual. Além da diminuição das funcionalidades, o desempenho do novo módulo de simulação de microfabricação é baixo. Para volumes de células de interesse prático, o software ainda é lento. O usuário ainda precisa de uma máquina poderosa para uma boa usabilidade com volumes grandes, com mais de 10^6 células. Existem diversas soluções para esses problemas mas não houve tempo hábil para amenizar esses problemas.

Quanto ao módulo de simulação desenvolvido, os resultados são bastante interessantes. O simMEMS possui agora um módulo de simulação capaz de simular tanto processos de microfabricação em substrato como em superfície. Apesar do autômato celular para deposição e corrosão de filmes ser simples, ele ainda é capaz de simular processos que a versão anterior do simMEMS era incapaz de simular. Vale mencionar que é trivial adicionar outros algoritmos para simulação desses processos. Como mencionado, o autômato de Huygens (STRASSER e SELBERHERR, 1995) só não foi implementado porque a nossa implementação ficou muito lenta.

O simMEMS, como todo software, ainda pode ser melhorado. A questão de desempenho é a principal barreira. Outro aspecto que precisa de trabalho é a inserção dos parâmetros físicos na simulação. Atualmente os processos são simulados como autômatos. O usuário seleciona quantos passos de qual autômato quer rodar. As comparações feitas no item 4.2.3, onde valores para o tamanho das células e a duração de um passo de tempo são definidos, são o que tornam a simulação interessante do ponto de vista técnico. Trabalhos nessa área podem levar o simMEMS de uma ferramenta útil para visualização de estruturas para um software de simulação de processos.

O módulo de análise desenvolvido também é interessante nesse sentido. Apesar da estrutura básica para análise estar desenvolvida, ainda é preciso estudar elementos tridimensionais que possam ser utilizados para análise das estruturas geradas pelo simulador de microfabricação em superfície e em substrato. Por essa razão os resultados apresentados por esse módulo foram descritos apenas de forma breve no item 0.

O projeto se mostrou ambicioso, e devido às restrições de tempo nem todos os módulos foram estudados e desenvolvidos ao máximo. Como já foi dito, há muitas otimizações que podem ser feitas. Contudo o simMEMS conta agora com a

possibilidade de simular diversos processos novos. Vemos isso como um ponto bastante positivo. A infra-estrutura para o desenvolvimento do software também foi melhorada com a adição de um sistema de integração contínua. Esse sistema, disponível em <http://aquamarina.lme.usp.br> permite baixar as últimas versões dos software CellularAutomata3D e simMEMS para Windows e OS X.

REFERÊNCIAS¹

BATHE, K.-J. **Finite Element Procedures**. Upper Saddle River, New Jersey: Prentice Hall, 1982.

BENJAMIN, L.; HAEMEL, N.; WRIGHT JR., R. S. **OpenGL SuperBible**. 4. ed. [S.l.]: Addison-Wesley Professional, 2007.

CHOPARD, B.; DROZ, M. **Cellular Automata Modeling of Physical Systems**. Cambridge: Cambridge University Press, 1998.

COOK, R. D.; MALKUS, D. S.; PLESHA, M. E. **Concepts and Applications of Finite Element Analysis**. New York: John Wiley & Sons, 1989.

DAI, C.-L. A maskless wet etching silicon dioxide post-CMOS process and its applications. **Microelectronic Engineering**, v. 83, p. 2543-2550, Julho 2006.

DE OLIVEIRA JR., J. P. **Desenvolvimento de um software para simulação atomística de corrosão anisotrópica do silício por autômato celular**. 1. ed. São Paulo: [s.n.], 2008.

DE OLIVEIRA JR., J. P.; CARREÑO, M. N. P. **Simulation of Anisotropic Etching of Silicon Using a Cellular Automata Model**. Microelectronics Technology and Devices - SBMicro 2008. Gramado, RS: ECS Transactions. 2008. p. 37-46.

DEAL, B. E.; GROVE, A. S. General Relationships for the Thermal Oxidation of Silicon. **Journal of Applied Physics**, v. 36, p. 3770-3779, 1965.

EDGEWALL SOFTWARE. The Trac Project. **The Trac Project**, 2011. Disponível em: <<http://trac.edgewall.org/>>. Acesso em: 14 abr. 2011.

FISH, J.; BELYTSCHKO, T. **A First-Course in Finite Elements**. Chichester: John Wiley & Sons, 2007.

FRISCH, U.; HASSLACHER, B.; POMEAU, Y. Lattice-gas automata for the Navier-Stokes equation. **Physical Review Letters**, v. 56, p. 1505, 1986.

FUKS, H.; BOCCARA, N. Convergence to equilibrium in a class of interacting particle systems evolving in discrete time. **Physical Review E**, v. 64, p. 016117, Junho 2001.

¹ De acordo com a Associação Brasileira de Normas Técnicas. NBR 6023.

GOSÁLVEZ, M. A. et al. Atomistic methods for the simulation of evolving surfaces. **JOURNAL OF MICROMECHANICS AND MICROENGINEERING**, 2008. 055029.

GOSÁLVEZ, M. A.; FOSTER, A. S.; NIEMINEN, R. M. Atomistic simulations of surface coverage effects in anisotropic. **Applied Surface Science**, 2002. 160-182.

GOSÁLVEZ, M. A.; XING, Y.; SATO, K. Analytical Solution of the Continuous Cellular Automaton for Anisotropic Etching. **Journal of Microelectromechanical Systems**, v. 17, p. 410-431, Abril 2008.

GOSALVEZ., M. A.; XING, Y. Home. **VisualTAPAS Home**, 2008. Disponível em: <<http://tfy.tkk.fi/~mag/VisualTAPAS/Home.html>>. Acesso em: 12 abr. 2011.

HUANG, Q.-A. et al. A modified cellular automata algorithm for the simulation of boundary advancement in deposition topography simulation. **JOURNAL OF MICROMECHANICS AND MICROENGINEERING**, v. 16, p. 1-8, 2006.

KARAFYLLIDIS, I. A three-dimensional photoresist etching simulator for TCAD. **Modelling and Simulation in Materials Science and Engineering**, p. 157–168, 1999.

KARAFYLLIDIS, I.; THANAILAKIS, A. Simulation of two-dimensional photoresist etching process in integrated circuit fabrication using cellular automata. **Modelling and Simulation in Materials Science and Engineering**, p. 629-642, 1995.

KRUG, J.; SPOHN, H. Universality classes for deterministic surface growth. **Physical Review A**, 38, 15 Outubro 1988. 4271-4283.

LORENSEN, W. E.; CLINE, H. E. **Marching cubes**: A high resolution 3D surface construction algorithm. Proceedings of the 14th annual conference on Computer graphics and interactive techniques. [S.l.]: ACM. 1987. p. 163-169.

MARTHA, L. F. Ftool - Two-dimensional Frame Analysis Tool. **Ftool Site**, 2010. Disponível em: <<https://web.tecgraf.puc-rio.br/ftool/ftooleng.html>>. Acesso em: 18 abr. 2011.

NOKIA. Qt - A cross-platform application and UI framework, 2010. Disponível em: <<http://qt.nokia.com/>>. Acesso em: 18 ago. 2010.

NORDMANN, A. File:Huygens-Fresnel principle.svg. **Wikipedia**, Abril 2007. Disponível em: <http://en.wikipedia.org/wiki/File:Refraction_-_Huygens-Fresnel_principle.svg>. Acesso em: 8 Agosto 2010.

ORACLE. Hudson Continuous Integration. **Hudson Extensible continuous integration server**, 2011. Disponível em: <<http://hudson-ci.org/>>. Acesso em: 14 abr. 2011.

SALINAS, S. R. A. **Introdução à Física Estatística**. São Paulo: Edusp, 1999.

SEIDEL, H. et al. Anisotropic etching of crystalline silicon in alkaline solutions, I. Orientation dependence and behavior of passivation layers. **Journal of the Electrochemical Society**, v. 137, p. 3612-3626, Novembro 1990.

SERGEY BOCHKANOV, V. B. ALGLIB. **ALGLIB**, 2010. Disponível em: <<http://www.alglib.net>>. Acesso em: 08 18 2010.

SERWAY, R. A.; JEWETT JR., J. W. **Princípios de Física**. 3. ed. São Paulo: Pioneira Thomson Learning Ltda., v. 4, 2002.

SHIKIDA, M. et al. Non-photolithographic pattern transfer for fabricating arrayed three-dimensional microstructures by chemical anisotropic etching. **Sensors and Actuators A**, 2004. 264–271.

SHIKIDA, M. et al. Non-photolithographic pattern transfer for fabricating pen-shaped microneedle structures. **Journal of Micromechanics and Microengineering**, 14, Novembro 2004. 1462-1467.

SIRAKOULIS, G. C.; KARAFYLLIDIS, I.; THANAILAKIS, A. A TCAD tool for the simulation of the CVD process based on cellular automata. **JOURNAL DE PHYSIQUE IV**, p. Pr3-205-Pr3-212, 2001.

SIRAKOULIS, G. C.; KARAFYLLIDIS, I.; THANAILAKIS, A. A cellular automaton methodology for the simulation of integrated circuit fabrication process. **Future Generation Computer Systems**, p. 639–657, 2002.

SIRAKOULIS, G. C.; KARAFYLLIDIS, I.; THANAILAKIS, A. A cellular automaton for the propagation of circular fronts and its applications. **Engineering Applications of Artificial Intelligence**, p. 731-744, 2005.

STRASSER, E.; SELBERHERR, S. Algorithms and Models for Cellular Based Topography Simulation. **IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS**, p. 1104-1114, 1995.

THAN, O.; BÜTTGENBACH, S. Simulation of anisotropic chemical etching of crystalline silicon using a cellular automata model. **Sensors and Actuators A**, v. 45, p. 85-89, Outubro 1994.

THAN, O.; BÜTTGENBACH, S. Simulation of anisotropic chemical etching of crystalline silicon using a cellular automata model. **Sensors and Actuators A**, v. 45, p. 85-89, 1994.

THE APACHE SOFTWARE FOUNDATION. Apache Subversion. **Apache Subversion**, 2011. Disponível em: <<http://subversion.apache.org/>>. Acesso em: 14 abr. 2011.

THE NEXUS ASSOCIATION. Network Microsystems European, 2006. Disponível em: <<http://www.nexus-mems.com/taskforce.asp>>. Acesso em: 07 Abril 2011.

TOFFOLI, T.; MARGOLUS, N. **Cellular Automata Machines**. Cambridge, Massachusetts: The MIT Press, 1987.

WANG, B.-H.; KWONG, Y.-R.; HUI, P.-M. Statistical mechanical approach to Fukui-Ishibashi traffic flow models. **Physical Review E**, Março 1998. 2568-2573.

WOLFRAM, S. **A New Kind of Science**. [S.l.]: Wolfram Media, 2002. ISBN ISBN 1-57955-008-8.

YANG, T. Y. **Finite Element Structural Analysis**. Englewood Cliffs, N.J.: Prentice-Hall, 1986.

YOLE DÉVELOPPEMENT. I-Micronews - View Report Status of the MEMS Industry, 2010. Disponível em: <<http://www.i-micronews.com/reports/Status-MEMS-Industry/159/>>. Acesso em: 07 Abril 2011.

ZHU, Z.; LIU, C. Micromachining Process Simulation Using a Continuous Cellular Automata Method. **JOURNAL OF MICROELECTROMECHANICAL SYSTEMS**, Junho 2000. 252-261.

Apêndice A - Módulo para Cálculo de Esforços pelo MEF

Como mencionamos previamente, um software para análise estática, modal e dinâmica de estruturas bidimensionais foi desenvolvido durante o projeto. Tratou-se de um desenvolvimento experimental para avaliar a viabilidade da incorporação de ferramentas de análise pelo MEF ao simMEMS. Neste apêndice alguns dos resultados obtidos com esse software serão apresentados. Na Figura 53 são exibidos alguns exemplos dos resultados gráficos gerados pelo software.

Figura 53 – A estrutura analisada está em (a) com as unidades em mm. Na imagem (b) temos a saída da análise para uma força aplicada no ponto C. A imagem (c) mostra o resultado da análise com uma força aplicada no ponto B.

A Figura 53 mostra o resultado da análise estática de uma estrutura utilizando elementos de pórtico. As deformações abaixo estão aumentadas para permitir uma melhor visualização. Uma força foi aplicada no ponto C da estrutura em (b) ou no ponto B em (c). Essa força é representada pela seta verde grande. A estrutura deslocada está em verde e a original em azul. Os apoios são representados pelos triângulos azuis. A estrutura é formada por barras homogêneas de alumínio com 1,5

mm de altura e 20 mm de largura. O valor do módulo de Young $E = 68 \text{ Gpa}$ foi adotado para o alumínio.

Três casos foram analisados e são apresentados na Tabela 3. Nos casos 1 e 2 pesos foram presos à estrutura no ponto C através de uma corda formando um ângulo de 45° com a barra no topo. No caso 3 um peso foi preso diretamente ao ponto B da estrutura. Foi adotado um valor de $9,8 \text{ m/s}^2$ para a aceleração da gravidade.

Tabela 3 - Peso, força equivalente e as componentes da força aplicada nos casos usados.

Caso	Carga (kg)	Carga (N)	Componente X (N)	Componente Y (N)
1	0,3162	3,0988	2,1912	-2,1912
2	1,131	11,0838	7,8374	-7,8374
3	1,164	11,4072	0,0	-11,4072

Os resultados obtidos pelo software desenvolvido foram comparados com os resultados de um software chamado Ftool (MARTHA, 2010). O Ftool é um programa desenvolvido utilizado para o ensino do estudo de estruturas desenvolvido junto à PUC-Rio. A Tabela 4 mostra a comparação dos resultados gerados pelo nosso módulo e pelo Ftool para as mesmas estruturas e as mesmas forças. Na tabela são comparados os deslocamentos do ponto A da Figura 53(a) devido às cargas.

Tabela 4 - Comparação entre os resultados do Ftool e do módulo desenvolvido.

Caso	Ftool (mm)		Programa Próprio (mm)	
	X	Y	X	Y
1	1,883	1,565e-4	1,883	1,565e-4
2	6,736	5,599e-4	6,737	5,599e-4
3	-3,729e-1	-4,771e-4	-3,729e-1	-4,771e-4

Uma estrutura em alumínio também foi construída e o deslocamento horizontal no mesmo ponto A foi medido. A

Tabela 5 mostra a comparação dos resultados gerados com os resultados experimentais.

Tabela 5 - Comparação entre os resultados do módulo e resultados experimentais medidos por um interferômetro.

Caso	Experimental (mm)	Calculado (mm)	Variação (%)
1	1,600	1,88346	-17,72
2	5,632	6,73668	-19,77
3	-0,448	-0,372887	16,77

Há uma variação maior entre o resultado experimental e os resultados do software. Isso pode ser devido às medidas das dimensões das barras. A espessura da barra em especial é importante nesse caso, devido ao segundo momento de área. Uma variação menor que a precisão da medida já modifica bastante os resultados do software.

Esse tipo de elemento pode ser utilizado também para calcular as frequências de oscilação de estruturas como pontes auto-sustentadas ou cantilevers. Os resultados do programa foram comparados com resultados experimentais obtidos em laboratório e mostram uma boa coerência também para a análise modal. O experimento consistiu em medir visualmente as frequências para os quatro primeiros modos de vibração de uma barra com as medidas na Tabela 6.

Tabela 6 – Medidas e valores adotados para a barra.

Dado	Valor
Comprimento	61 cm
Espessura	1,00 mm
Largura	30,84 mm
Área Seção	30,84 mm ²
Momento de Área	2,57 mm ⁴
Módulo de Young	205 GPa
Densidade	7860 kg/m ³

Os resultados da comparação foram razoavelmente bons, como pode ser visto na Tabela 7. Novamente há uma divergência entre os resultados teóricos e os experimentais. Isso pode ser explicado em parte pela medida das frequências. Como

estas foram levantadas apenas observando a medida da frequência do oscilador da barra e as medidas foram feitas apenas observando o formato da barra ao vibrar, pode haver um desvio, principalmente para os modos mais altos.

Tabela 7 - Comparação entre resultados experimentais e do módulo para análise modal.

Modo	Experimental (Hz)	Calculado (Hz)	Erro (%)
1	2,36	2,21708	6,06
2	14,618	13,8944	4,95
3	43,706	38,9046	10,99
4	85,401	76,2375	10,73

Como vemos os resultados não são ótimos, especialmente para as frequências mais altas. Como uma matriz de massa consistente foi usada, o resultado não vai se aproximar mais do experimental com mais elementos. Isso ocorre porque a frequência é aproximada por cima nesse tipo de análise e os valores calculados já são menores que os medidos (YANG, 1986).

O elemento ISOQ4 pode ser utilizado para analisar elementos planos. A análise dos resultados de uma simulação 2D podem ser simulados nesse módulo facilmente, visto que as células já podem ser usados como a malha para análise. O programa permite visualizar a estrutura deslocada devido aos esforços aplicados bem como tensões normais nas direções X e Y e a tensão de cisalhamento. A figura abaixo mostra o resultado de uma análise. Neste caso usamos cinco apoios que restringem o movimento na direção horizontal. O apoio superior também restringe o movimento na direção vertical. Não aplicamos esforços nesse caso. Ao invés disso foi definido um deslocamento vertical para o nó superior direito equivalente a metade da altura da barra. A análise determina os deslocamentos dos demais nós, bem como o deslocamento na direção horizontal do nó superior direito. Após essa análise também é possível saber a força que deve ser aplicada nesse ponto para obter esse deslocamento.

Figura 54- Um exemplo de uma estrutura sendo analisada. Em (a) vemos a malha utilizada e os apoios em vermelho. As figuras(b), (c) e (d) mostram a figura deslocada e as tensões normais na direção horizontal e vertical e a tensão de cisalhamento, respectivamente. A escala de cores tem vermelho como o valor mínimo e azul como o máximo.

Os resultados desse programa também foram comparados com valores experimentais. O software apresentou bons resultados para a análise estática, com variações desprezíveis em relação ao elemento de pórtico para malhas finas o suficiente. Os resultados para as análises modais e dinâmicas não foram os esperados. Vale mencionar que o elemento de pórtico é mais recomendado para o tipo de análise feita. No caso a análise era de uma viga engastada vibrando. Enquanto um único elemento de barra pode fazer a análise estática para pequenos deslocamentos nessa estrutura, é preciso ter centenas de elementos do tipo ISOQ4 para obter o mesmo resultado. Na Figura 55 são apresentados alguns *screenshots* do resultado para uma análise dinâmica feita como o programa.

Figura 55 – Resultados da análise dinâmica de uma estrutura simples. A estrutura original está em azul. As cores mostram a tensão normal na direção vertical seguindo a escala crescente de vermelho a azul.

Novamente usamos uma condição inicial para deslocar a estrutura. Em seguida a estrutura é liberada e podemos ver como a estrutura oscila. Os apoios restringem o movimento nas direções horizontal e vertical. Nesse exemplo foram utilizados 56 elementos para a estrutura toda. A tensão normal da direção vertical foi utilizada para colorir as imagens em todos os casos. A escala é a mesma da Figura 54.