Alberto de Moraes Barbosa

Modelo SPICE Compacto para Dispositivos e Sensores a Onda Acústica de Superfície

Recife

2002

Universidade Federal de Pernambuco

Programa de Pós-graduação em Engenharia Elétrica

Modelo SPICE Compacto para Dispositivos e Sensores a Onda Acústica de Superfície

Dissertação

submetida à Universidade Federal de Pernambuco

como parte dos requisitos para obtenção do grau de

Mestre em Engenharia Elétrica

Alberto de Moraes Barbosa

Recife, Maio de 2002.

Modelo SPICE Compacto para Dispositivos e Sensores a Onda Acústica de Superfície

Alberto de Moraes Barbosa

'Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Eletrônica, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco'.

.....
Edval J. P. Santos, Ph.D.
Orientador

.....
Eduardo Fontana, Ph.D.
Coordenador do Programa de
Pós-graduação em Engenharia Elétrica

Banca Examinadora:

.....
Edval J. P. Santos, Ph.D.
Presidente

.....
Antônio Sergio Sombra, D.Sc.

.....
Eurico Bezerra de Souza Filho, D.Sc.

Aos meus pais e minha esposa.

Agradecimentos

Agradeço a Deus pela inspiração e coragem, à minha esposa pela compreensão e apoio em todas as horas, aos meus pais pelo exemplo de amor, honestidade e de vida, ao meu orientador, Prof. Edval Santos, pela confiança depositada e dedicação em transmitir os seus conhecimentos, à CAPES pelo suporte financeiro, ao Projeto FINEP/RECOPE - sensores e atuadores, aos colegas de mestrado Renato Cintra, Rodrigo Ramos, Jener Toscano, Victor Miranda e Isnaldo Coêlho, que sempre se fizeram presentes com conselhos e dicas. Agradeço também a Roberto Barros, Roberto Luiz Xavier e a todos que contribuiram direta ou indiretamente para a concretização desta dissertação.

Alberto de Moraes Barbosa

Universidade Federal de Pernambuco

31 de Maio de 2002

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia Elétrica.

Modelo SPICE Compacto para Dispositivos e Sensores a Onda Acústica de Superfície

Alberto de Moraes Barbosa

Maio/2002

Orientador: Edval J. P. Santos, Ph.D.
Área de Concentração: Eletrônica
Palavras-chaves: SPICE, SAW, Sensores Inteligentes
Número de páginas: xv+155

O desenvolvimento atual em sensores tem como objetivo uma maior integração entre o elemento de transdução e o circuito eletrônico para detecção, processamento e comunicação, de maneira que todos os componentes sejam fabricados no mesmo *chip*. Esse tipo de sensor é denominado genericamente de sensor inteligente integrado (*integrated smart sensor*). A sua simulação requer modelos que possam ser utilizados em simuladores de circuito. SPICE é um simulador de propósito geral e tem sua estrutura presente em vários outros simuladores comerciais. Ele foi projetado desde o início para ser uma ferramenta de simulação de circuitos integrados.

Dispositivos a Onda Acústica de Superfície, OAS (em inglês, SAW= "Surface Acoustic Wave") têm diversas vantagens, tais como: baixo custo, leveza, reduzido tamanho e operação passiva. Modelos de transdutores acústicos têm sido propostos, mas apenas um deles pode ser utilizado em algumas versões do SPICE (PSPICE® e HSPICE®). Todos têm dificuldade para modelar transdutores longos. Nosso modelo é baseado nas equações dos modos acoplados. É compacto e tem sido utilizado, com sucesso, em diversas versões do SPICE, desde o SPICE 3f4, disponível gratuitamente, até o ELDO® da Mentor Graphics Corp.

Abstract of Dissertation presented to UFPE as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Compact SPICE Model for Devices and Sensors Based on Surface Acoustic Wave

Alberto de Moraes Barbosa

May/2002

Supervisor: Edval J. P. Santos, Ph.D.
Area of Concentration: Electronic
Keywords: SPICE, SAW, Smart Sensors
Number of pages: xv+155

Recent developments on sensors aim at a higher integration between the sensing element and the electronic circuitry for detection, processing, and communication, so that, all components can be manufactured on the same chip. This kind of sensor is commonly called integrated smart sensor. The simulation of such sensors requires models that can be used in circuit simulators. SPICE is a general purpose simulator, free of charge and its engine is used by several commercial simulators. Since the beginning, it was developed as a simulation tool for integrated circuits.

Surface acoustic wave devices, SAW, have a large application range, due to its advantages, such as: low cost, light weight, small size, passive component, electric isolation. Acoustic transducers models have been proposed, but only one can be applied in some SPICE versions (PSPICE® and HSPICE®). All of them have difficulty to model long transducers. Our model is derived from coupled-mode equations. It is a compact model and has been used successfully in several SPICE versions, since SPICE 3f4, that can be downloaded free of charge, up to ELDO® from Mentor Graphics Corp.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

Conhecer previamente o comportamento do circuito é extremamente útil no desenvolvimento de dispositivos, já que representa economia com montagens de protótipos. Não se pode pensar em simulação sem modelamento, pois o resultado obtido estará o mais próximo da realidade quanto mais eficiente for o modelo adotado, podendo este ser funcional, analítico ou físico. No caso dos modelos funcionais o seu comportamento pode ser representado por um circuito elétrico cuja funcionalidade se assemelha ao do dispositivo real. No analítico, uma expressão matemática reproduz o comportamento físico. Já no modelo físico, o seu comportamento é definido através de fenômenos da natureza. No caso do simulador SPICE, modelos funcionais podem ser implementados através de macromodelos e em algumas versões comerciais os parâmetros do modelo podem ser representados por equações matemáticas. Já os físicos necessitam que o código fonte do SPICE sofra alterações, de forma que as equações pertinentes ao modelo sejam explicitadas. Estas técnicas de implementação serão detalhadas oportunamente.

O nosso objetivo é desenvolver um modelo de um transdutor de Onda Acústica de Superfície (OAS) ou *Surface Acoustic Wave (SAW)* simples e capaz de ser empregável no simulador SPICE. Um modelo de um OAS para o SPICE foi proposto em 1995 [1], porém este modelo representa cada par de eletrodo através de um circuito independente, o que dificulta o modelamento de trandutores longos.

Dispositivos OAS são encontrados hoje em televisores para seleção de canais, como filtros em celulares e sistemas de comunicação [2] e em radares. Como sensores estão presentes em análises de gases [3], líquidos e também na biomedicina. Neste caso,

o dispositivo OAS é normalmente utilizado como linha de atraso ou ressonador. Na sua construção como linha de atraso são empregados dois transdutores interdigitados, figura 1.1. O primeiro é utilizado para excitar a onda acústica que se propagará sobre um cristal piezoelétrico até o segundo transdutor que converterá este sinal acústico em elétrico. Durante esta propagação o sinal irá sofrer um atraso e será atenuado. Este atraso e atenuação é decorrente da interação com o meio e podemos utilizar a variação da freqüência de oscilação como parâmetro de monitoramento da grandeza de interesse, caracterizando-o como um sensor. A integração entre o elemento acústico e o circuito eletrônico está presente no desenvolvimento de sensores inteligentes de baixo custo e portáteis.

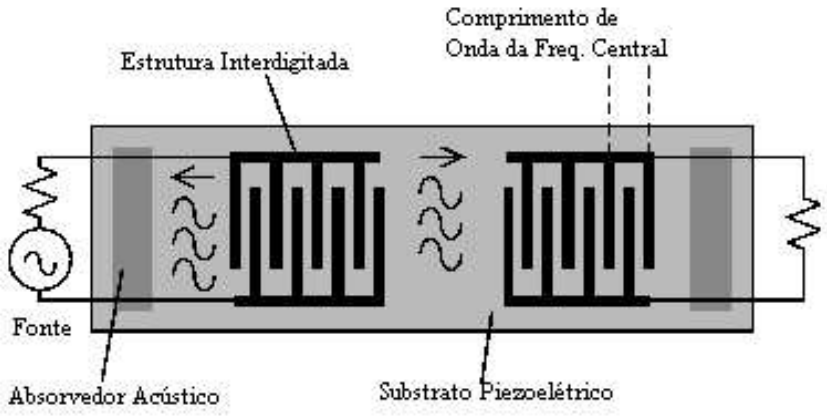

Figura 1.1: OAS como Linha de Atraso

Vimos que os simuladores são uma ferramenta de grande valia no desenvolvimento de circuitos. Um simulador de propósito geral capaz de utilizar componentes ativos e passivos, de forma a atender a necessidade de integração sensor x eletrônica, não está disponível comercialmente com fácil acesso e baixo custo, o que dificulta a sua utilização na elaboração de projetos. Esta foi a motivação para o desenvolvimento de um modelo para um dispositivo OAS capaz de ser incorporado por um simulador de propósito geral, de livre acesso e de uso disseminado no meio acadêmico. O simulador ideal para este propósito é o SPICE (*Simulation Program with Integrated Circuit Emphasis*), que possui versões gratuitas distribuídas na Internet, amplamente utilizado e base de muitos simuladores comerciais. Outros modelos para simulação de dispositivos OAS foram desenvolvidos [4], mas nenhum deles tem a simplicidade do apresentado nesse trabalho ou não permitem uma integração com o simulador.

1.1 Organização da tese

Esta dissertação está divida em seis capítulos, o primeiro é esta introdução, onde é apresentado o dispositivo OAS, algumas aplicações e qual o objetivo e motivação para elaborar um modelo para sua simulação no SPICE.

No segundo capítulo é dada uma visão do comportamento do dispositivo OAS, como podem ser classificados quanto ao modo de propagação e os tipos de ondas mecânicas de superfície. Também são apresentados alguns modelos já existentes e suas desvantagens.

No terceiro é abordado sobre como modela-se um dispositivo no SPICE, tanto com a implementação de macromodelos, utilizando a estrutura de subcircuitos do SPICE, como a implementação de um novo componente alterando-se o código-fonte do SPICE 3f4.

No quarto capítulo é apresentado uma proposta de um macromodelo para um dispositivo OAS e a implementação de um novo modelo físico para uma linha acústica.

No quinto capítulo, é demonstrado uma aplicação do modelo como um sensor de espessura de filme fino, comparando os resultados da simulação com valores experimentais obtidos da literatura.

No sexto e último capítulo são apresentadas as conclusões e sugestões para trabalhos futuros.

Esta dissertação também contém três apêndices, para consultas sobre o simulador SPICE3f4 e os seus códigos fontes que sofreram alterações.

Capítulo 2

Uma visão do dispositivo OAS

A aplicação do fenômeno da Onda Acústica de Superfície em dispositivos eletrônicos requer a utilização de materiais piezoelétricos. A piezoeletricidade refere-se a polarização elétrica do substrato pela imposição de uma força mecânica, este fenômeno é recíproco e responsável pela conversão do sinal eletromagnético em acústico e vice-versa. A constante que expressa este acoplamento é representada por $K^2(0 \leq K^2 < 1)$ e o seu valor está associado ao tipo do substrato.

De uma forma simplista, um dispositivo OAS pode ser considerado como uma sucessão de eletrodos metalizados depositados sobre um substrato piezoelétrico altamente polido, como o quartzo. Dispostos com polaridades alternadas, de forma que, ao ser aplicado um sinal de tensão RF com uma freqüência propícia causam uma contração e uma expansão na superfície do cristal. Isso gera uma onda acústica de superfície, onde metade de sua energia se propaga para a direita e a outra metade para a esquerda (bidirecional). Uma metade da onda caminha em direção ao transdutor de saída e a outra caminha em direção ao final do cristal, podendo ser refletida na borda imediatamente atrás do transdutor. Isto acarreta um efeito conhecido como *Triple Transit Echo* que resulta em distorções no sinal de recepção. Este efeito pode ser eliminado pelo uso de absorvedores acústicos nos finais da linha de atraso ou até mesmo pela disposição dos transdutores de modo a formarem um ângulo com as bordas do substrato, figura 1.1.

O fato de apenas uma metade ser transferida já implica em uma perda de 3dB, inerente à estrutura. Outras fontes de perdas de inserção são as perdas resistivas e o próprio substrato piezoelétrico, que introduz atenuações na propagação da onda e

nas suas conversões entre mecânica e elétrica. Esta perda de inserção em dispositivos práticos, como filtros, é da ordem de 15 a 30dB. A velocidade de propagação e a dependência com a temperatura está relacionada também com o tipo e a orientação do material cristalino usado na fabricação. A tabela 2.1 mostra alguns parâmetros relevantes para cada material.

Material	Orientação	Veloc. (m/s)	Coef.de temp. (ppm/^{o}C)	Atenuação a 1GHz(dB/μS)
Quartzo	Y, X	3159	-24	2.6
Quartzo	ST, X	3158	0	3.1
Tantalato de lítio	Y, Z	3230	35	1.14
Tantalato de lítio	rotação 167^o	3394	64	-
Niobato de lítio	Y, Z	3488	94	1.07
Niobato de lítio	rotação 128^o	3992	75	-

Tabela 2.1: Parâmetros de alguns materiais

A resposta em freqüência do transdutor é a transformada de Fourier da resposta ao impulso, sendo esta a mesma coisa que a distribuição de carga ao longo do comprimento. Considerando um transdutor formado por eletrodos distribuidos uniformementes e de mesmo comprimento, o envelope da resposta ao impulso é um pulso, o que é de se esperar, já que a soma dos impulsos existentes em cada par de eletrodo acarreta uma distribuição de carga que tem a mesma forma retangular do transdutor e a transformada de Fourier de um pulso é a função sen(x)/x, como mostrado na figura 2.1. A largura da banda passante, portanto, é definida pelo número de pares de eletrodos existentes.

A capacitância de um par de eletrodo na superfície com fator de metalização de 50%, figura 2.2, é calculada através da técnica de mapeamento conforme e expressa por:

$$C = \frac{1}{2}(\epsilon_0 + \epsilon_{sup})W \tag{2.1}$$

onde ϵ_{sup} e ϵ_0 são a permissividade da superfície e do ar ou vácuo, respectivamente.

Numa faixa de freqüência de RF e microondas (acima de 1GHz) é mais interessante utilizar os parâmetros S, os quais representam a passagem de ondas viajantes, isto é, a_1 e a_2 representam ondas incidentes nas portas 1 e 2 respectivamente, enquanto que b_1 e b_2 representam ondas emergentes nas portas 1 e 2 respectivamente.

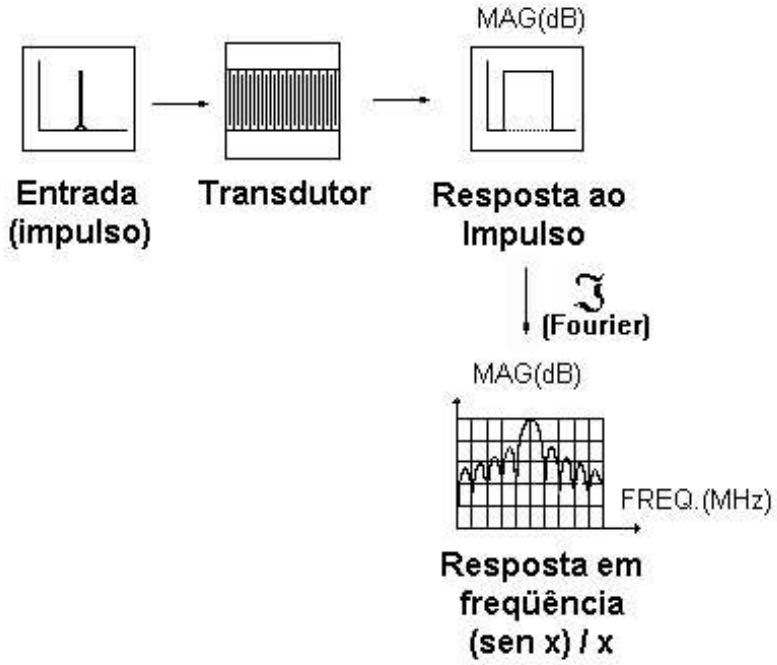

Figura 2.1: Resposta do transdutor ao Impulso

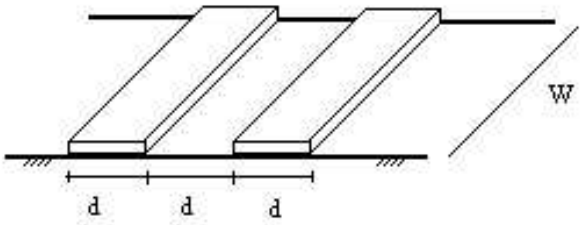

Figura 2.2: Par de eletrodos

$$b_1 = S_{11}a_1 + S_{12}a_2$$

$$b_2 = S_{21}a_1 + S_{22}a_2$$

$S_{11} = \Gamma_{11}$ representa o coeficiente de reflexão, com a porta 2 casada.

$S_{22} = \Gamma_{22}$ representa o coeficiente de reflexão, com a porta 1 casada.

$S_{12} = T_{12}$ representa o coeficiente de transmissão direto.

$S_{21} = T_{21}$ representa o coeficiente de transmissão reverso.

Em termos de matriz pode-se escrever: $\mathbf{b} = S\mathbf{a}$, onde $\mathbf{b}$ e $\mathbf{a}$ são matrizes colunas

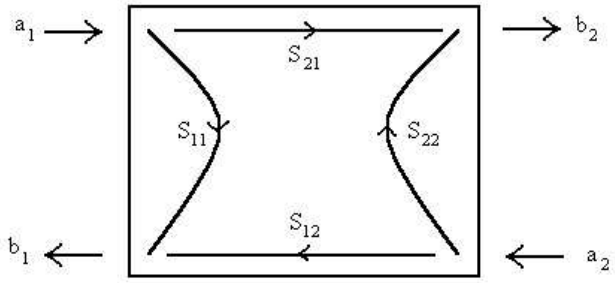

Figura 2.3: Rede de dois acessos com parâmetros S

e S é a matriz simétrica denominada de matriz de espalhamento. Alguns parâmetros podem ser definidos:

1- Perda de Inserção (IL=*Insertion Loss*) = S_{21}

2- TTS (*Triple Transit Echo*) = $\frac{|S_{21}|_{max}}{|S_{21}|_{min}}$

A largura de banda de um transdutor é inversamente proporcional ao número de pares de eletrodos (N) e obtida através da equação 2.2.

$$\Delta f = \frac{2f_0}{N} \tag{2.2}$$

Um transdutor típico com 50 pares de eletrodos operando numa freqüência de 30MHz terá uma largura de banda de cerca de 1,2MHz, o que implica em dizer que o sinal de RF deverá ser de 30MHz $\pm$ 0,6MHz. O comprimento de onda é determinado pelo espaçamento entre os eletrodos. A impedância elétrica do transdutor em um dado substrato é determinada pelo número de eletrodos e a distância entre eles. Para um típico substrato piezoelétrico como o quartzo, que possui uma velocidade de propagação da onda de cerca de 3100m/s, a uma freqüência de operação de 31MHz, teremos um comprimento de onda de 100 micrometros ($f = v/\lambda$). Como a distância entre dois eletrodos consecutivos de mesmo potencial representa um comprimento de onda, e assumindo que a largura do eletrodo é a mesma do seu espaçamento, teremos uma estrutura interdigitada com um espaçamento entre os eletrodos de 25 micrometros.

A Onda Acústica de Superfície é útil na construção de linhas de atraso e ressonadores miniaturizados devido ao comprimento da onda acústica ser 10^5 vezes menor

que uma onda eletromagnética de mesma freqüência. Na prática a microfabricação, através da litografia, limita a largura dos eletrodos ao "estado da arte", cerca de 0,25 micrometros (uma freqüência de operação de 3GHz) e o limite da freqüência inferior é imposto pelo seu tamanho físico, este valor é da ordem de 10MHz, o que resultaria num dispositivo de alguns cm^2.

2.1 Classificação dos Dispositivos

Dispositivos Acústicos são classificados pelo modo de propagação da onda através ou sobre um substrato piezoelétrico. A propagação pode se dar de forma paralela ou normal à superfície do substrato, dependendo da polarização imposta.

Se a onda se propaga através do substrato é chamada de *Bulk Acustic Wave (BAW)* ou Onda Acústica de Volume. Os dispositivos BAW mais comuns são o ressonador de modo transversal, ou *Thickness Shear Mode (TSM)* e o sensor de modo de ondas transversais horizontais ou *shear-horizontal acoustic plate mode (SH-APM).*

2.1.1 Ressonador TSM de Onda Acústica de Volume

O ressonador TSM consiste em um disco fino de quartzo com dois eletrodos de placas circulares depositados em seus lados. A aplicação de uma tensão entre estes eletrodos causa uma deformação transversal do cristal, figura 2.4.

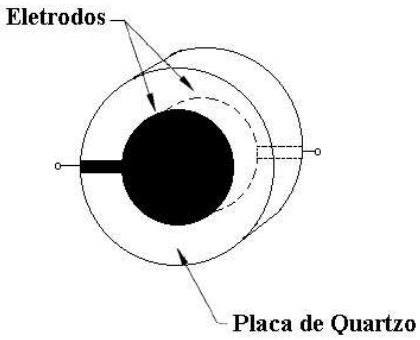

Figura 2.4: Ressonador TSM

2.1.2 Sensor SH-APM de Onda Acústica de Volume

Já o sensor SH-APM utiliza um cristal piezoelétrico em forma de placa fina que serve como guia de onda acústica, confinando a energia entre a superfície superior e inferior da placa, figura 2.5.

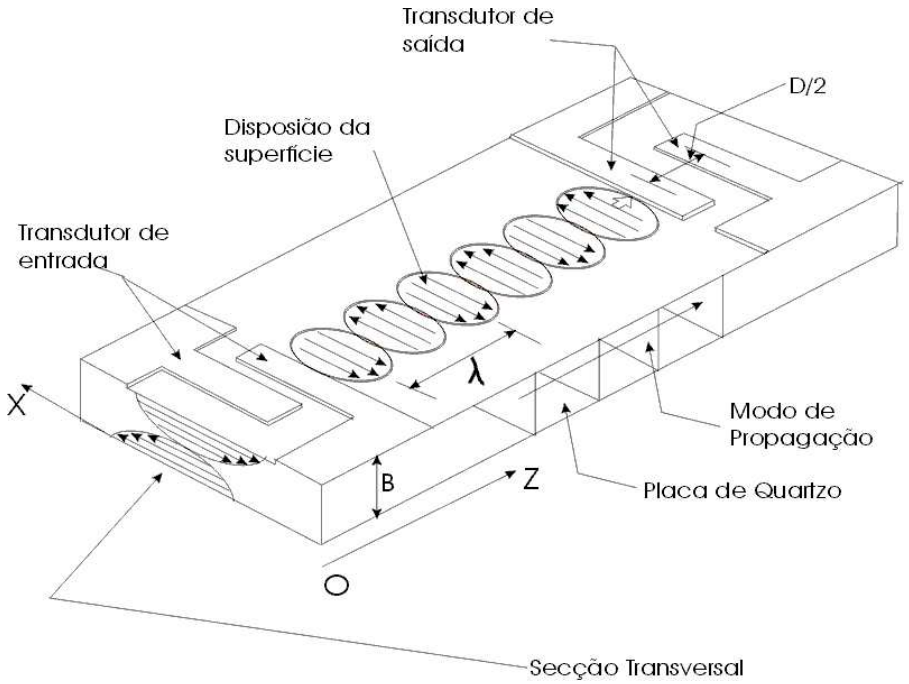

Figura 2.5: Sensor SH-APM

2.1.3 Dispositivos de Onda Acústica de Superfície

Se a onda se propaga pela superfície do substrato é conhecida como Onda de Superfície. Os dispositivos mais comuns são os sensores OAS, figura 2.6(a) e o sensor de onda acústica transversal-horizontal ou *shear-horizontal surface acoustic wave (SH-SAW)*, figura 2.6(b). O corte do cristal piezoelétrico é que determina o modo de propagação (horizontal ou vertical).

2.2 Classificação das Ondas Mecânicas de Superfície

Como o foco é a onda que se propaga na superfície cabe mostrar como podem ser classificadas quanto à maneira que esta propagação ocorre na superfície dos sólidos.

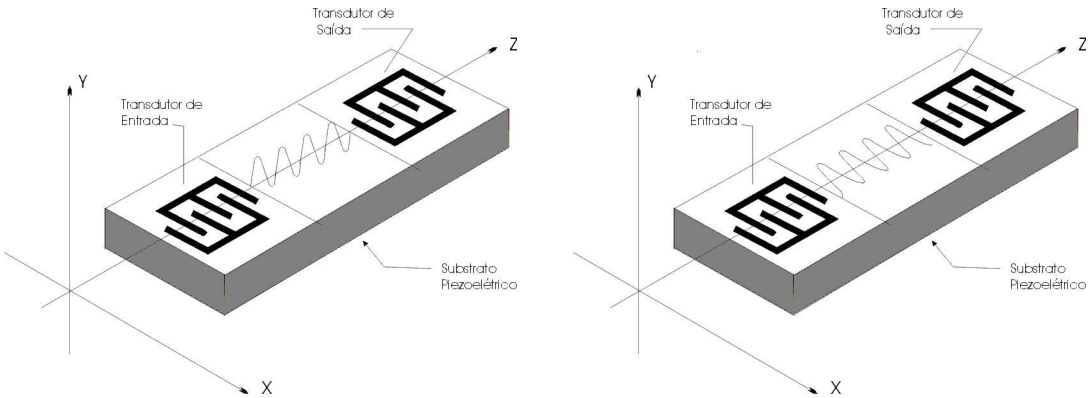

Figura 2.6: (a) Sensor OAS (b) Sensor SH-SAW

2.2.1 Ondas de Rayleigh

São ondas que se propagam na interface entre um meio elástico e o vácuo. Este tipo de onda será considerado quando daqui por diante nos referenciarmos a ondas acústicas de superfície sem a explicitação do seu tipo.

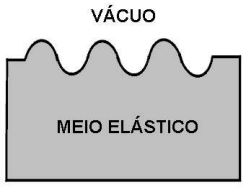

Figura 2.7: Ondas de Rayleigh

2.2.2 Ondas de Lamb

São ondas que se propagam na interface entre um meio elástico de pouca espessura e o vácuo, de forma que a deformação do meio ocorre de maneira uniforme.

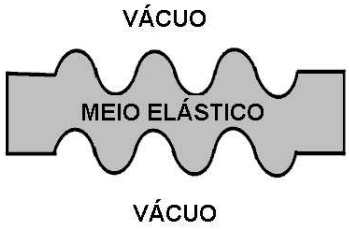

Figura 2.8: Ondas de Lamb

2.2.3 Ondas de Love

São ondas que se propagam na superfície de um meio elástico sobreposto a outro.

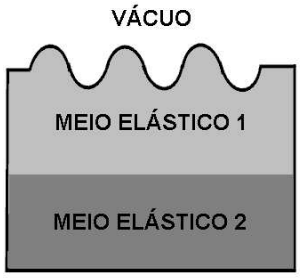

Figura 2.9: Ondas de Love

2.2.4 Ondas de Stoneley

São ondas que se propagam na interface entre um meio elástico e outro.

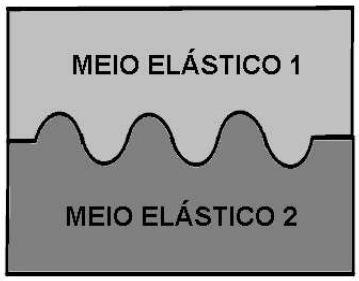

Figura 2.10: Ondas de Stoneley

Todos os dispositivos acústicos são sensores intrínsecos, já que são sensíveis a pertubações de vários parâmetros físicos. O modo de propagação (TSM, SH-APM ou SH-SAW) deve ser escolhido de modo que se torne mais eficaz no monitoramento da grandeza de interesse. Neste trabalho é dedicado um capítulo ao modelamento de um sensor de ondas acústicas de volume(TSM) para uso no SPICE apresentado por Alf Püttmer *et al* em seu artigo [5] e no decorrer deste trabalho será proposto um modelo para o caso de um sensor OAS.

2.3 Equações do dispositivo OAS, um par de eletrodos

Como não se pode representar o OAS completamente através de uma simples função de transferência, é lançado mão de algumas equações que, levando em consideração o tipo de propagação da onda, explicam de forma aproximada o seu comportamento físico. As equações demonstradas levam em conta um transdutor com eletrodos alinhados, de mesmas dimensões, distribuídos uniformemente e propagando ondas de Rayleigh (na interface entre um meio elástico e o vácuo ou ar) e com índice de metalização de 50%.

Um par de eletrodos paralelos depositados em um substrato piezoelétrico constitui um dipolo complexo com impedância dada por:

$$Z(j\omega) = \frac{1}{j\omega C} + R_0(\omega) \tag{2.3}$$

A capacitância eletrostática C é obtida por:

$$C = \frac{1}{2}(\epsilon_{sup} + \epsilon_0)W$$

onde W é o comprimento dos eletrodos e ϵ_{sup}, ϵ_0 a permissividade da superfície e do ar ou vácuo, respectivamente. A impedância característica $R_0(\omega)$ é dada por [6]:

$$R_0 \approx \frac{2k^2\alpha^2(K_\infty)}{(\epsilon_{sup} + \epsilon_0)W\omega} \tag{2.4}$$

onde

$$\alpha(K) = \frac{W}{2jQ_0}\int_{-\infty}^{+\infty} \sigma(x)e^{jkx}dx$$

$\sigma(x)$ é a densidade de carga superficial e Q_0 é a carga acumulada no capacitor de comprimento W.

$$K_\infty = \frac{\omega}{v_\infty}$$

onde v_∞ é a velocidade de propagação da onda de Rayleigh em circuito aberto. Na prática $\alpha^2 \simeq 0.7$ perto da freqüência de ressonância, de forma que podemos representar a impedância característica como:

$$R_0 \approx \frac{1.4k^2}{(\epsilon_{sup} + \epsilon_0)W\omega} \tag{2.5}$$

2.3.1 Um sistema de duas capacitâncias

Um sistema de duas capacitâncias consiste em um dispositivo de duas portas formadas por um par de eletrodos, um na entrada e outro na saída, como ilustrado na figura 2.11.

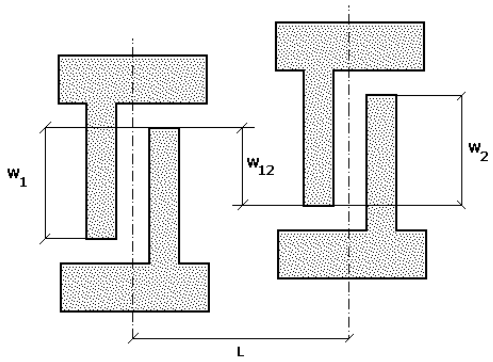

Figura 2.11: Par de transdutores

A matriz impedância representa a relação entre as correntes e tensões de entrada.

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{j\omega C_1} + R_1 & Z_{12} \\ Z_{21} & \frac{1}{j\omega C_2} + R_2 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$$

Neste caso, temos que a transimpedância é dada por:

$$Z_{21} = \frac{2k^2\alpha_1\alpha_2 W_{12}}{(\epsilon_0 + \epsilon_{sup})W_1 W_2 \omega} e^{-j\omega\tau_{12}}$$

onde:

$$\tau_{12} = \frac{L}{v_\infty}$$

No caso em interesse, temos que: $R_1 = R_2 = R_0$; $C_1 = C_2 = C$; $W_{12} = W_1 = W_2 = W$. Como a piezoeletricidade é um fenômeno recíproco, temos que:

$$Z_{12} = Z_{21} = R_0 e^{-j\omega\tau_{12}}$$

2.4 Tansdutores com $N + 1$ eletrodos idênticos

Neste caso teremos N capacitores planares. Consideraremos um espaçamento regular de D e um comprimento de W.

A matriz de impedância é representada da seguinte forma:

$$\mathbf{V_l} = Z_{li}\mathbf{I_i}(i, l = 1, 2, ..., N) \tag{2.6}$$

onde:

$$Z_{ll} \approx \frac{1}{jC\omega} + R_0$$
$$Z_{li} \approx R_0 e^{-j\omega|l-i|\tau}$$
$$\tau = \frac{D}{v_\infty}$$

Sendo V a tensão aplicada ao transdutor e I a corrente total, temos que:

$$V_l = (-1)^l \frac{V}{2} \tag{2.7}$$

$$I = \sum_{l=1}^{N} I_l \tag{2.8}$$

2.4.1 Solução exata

A solução exata da equação 2.6, considerando as equações 2.7 e 2.8 é dada por [6]:

$$Z(\omega) = \frac{V}{I} = \frac{1}{jC\omega}\left\{\frac{\eta G(\xi) + rF(\xi)}{[N-(N-2)r]G(\xi) + 2(1-r)^2 H(\xi)}\right\} \tag{2.9}$$

onde

$$r = -e^{-j\omega\frac{D}{v_\infty}} = e^{-j\theta}$$
$$\eta = (1-r) + \gamma(1+r)$$
$$F(\xi) = \frac{(\xi^{N-1}+1)}{(\xi+1)}$$
$$G(\xi) = \frac{(\xi^N - 1)}{(\xi^2-1)}$$
$$H(\xi) = [(1/2)NF(\xi) - G(\xi)]\xi/(\xi-1)^2$$
$$= \sum_{n=1}^{(N/2)-1} n[(N/2)-n]\xi^{2n-2} + n[(N/2)-n-1]\xi^{2n-1}$$

O número complexo ξ é uma das raízes da equação:

$$\xi + \frac{1}{\xi} = r(1-\gamma) + \frac{1}{r}(1+\gamma)$$

onde

$$\gamma = jR_0C\omega \approx j0.7k^2$$

A impedância característica calculada através da equação 2.9 é mostrada na figura 2.12 para diferentes valores do produto Nk^2.

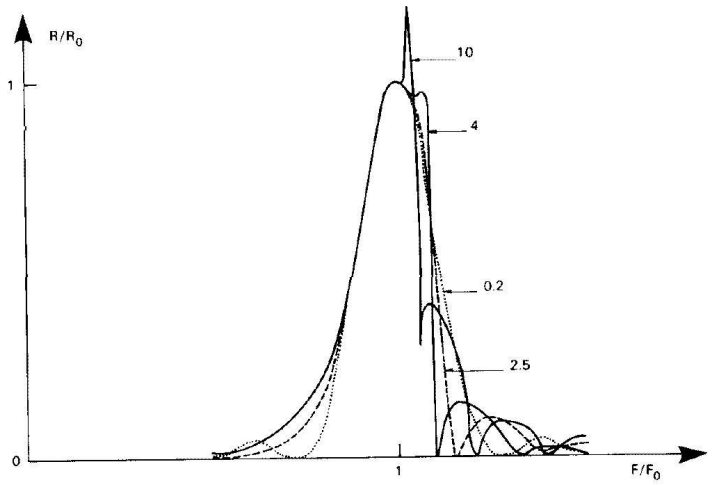

Figura 2.12: Impedância característica. Fonte: M. Feldman, Surface Acoustic Waves for Signal Processing.

Nota-se que para valores pequenos de Nk^2, a função tende para uma forma assintótica $\left[\frac{senx}{x}\right]^2$, onde: $x = N\omega/\omega_0$.

2.4.2 Expressão aproximada

Queremos simplificar a equação 2.9 considerando a forma assintótica para valores pequenos de k^2. Temos que:

$$[Z] = \frac{1}{j\omega C} + R_0[M] \tag{2.10}$$

A matriz [M] é dada por:

$$M_{ij} = r^{|i-j|}(i, j = 1, 2, ...N)$$

Considerando as equações 2.7 e 2.8, temos que:

$$Z = \frac{1}{\sum_{il} Y_{il}} \tag{2.11}$$

Da equação 2.11 podemos concluir que:

$$Z(\omega) = \frac{1}{j\omega NC} + \frac{R_0}{N^2}\left[N + 2\frac{(N-1)r - Nr^2 + r^{N+1}}{(1-r)^2}\right] \tag{2.12}$$

Esta equação pode ser simplificada considerando-se a freqüência próxima da ressonância pela substituição:

$$x = \frac{N\theta}{2} = \frac{N\tau}{2}(\omega - \omega_0)$$

onde:

$$\tau = \frac{D}{v_\infty}$$

$$\omega_0 = \pi/\tau$$

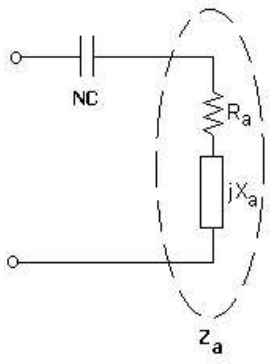

Figura 2.13: Circuito equivalente série

Para $x \ll 1$, temos finalmente que:

$$Z(\omega) = \frac{1}{j\omega NC} + R_0\left[\left(\frac{\text{sen}x}{x}\right)^2 + j\frac{\text{sen}2x - 2x}{2x^2}\right] \tag{2.13}$$

Esta é a equação simplificada da impedância de um transdutor com N+1 eletrodos, e mostra uma capacitância NC em série com uma impedância Z_a, como ilustrado na figura 2.13, da forma:

$$Z_a = 2R_0\frac{e^{-z} + z - 1}{z^2}$$

onde:

$$z = 2jx = jN\tau(\omega - \omega_0)$$

Considerando a representação em paralelo mostrada na figura 2.14, temos:

$$Y(\omega) = j\omega NC + N^2C^2\omega^2 Z_a \tag{2.14}$$

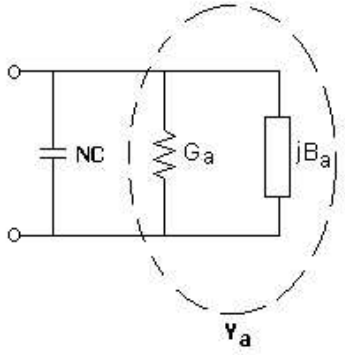

Figura 2.14: Circuito equivalente paralelo

2.5 Modelos já propostos

Um modelo de circuito equivalente para transdutores a Ondas Acústicas de Superfície foi primeiramente proposto por Smith et al [4]. Este circuito, que representa uma seção de um par de eletrodos, foi obtido através da analogia com o modelo de Mason para transdutores de ondas de volume. Modificações e melhoramentos deste circuito foram feitos por outros autores, como Krimholtz et al [7], que introduziu componentes dependentes da freqüência conectados ao centro de uma linha de transmissão, Redwood et al [8], que utilizou o circuito para modelar propagações transversais ou horizontais e sem a geração de um terceiro harmônico em sua resposta em freqüência, Steven et al [9], que utilizou uma linha acústica para modelar a transmissão da onda acústica de volume, simplificando o modelo de Mason, Yasuo et al [10], que simula os efeitos não lineares do dispositivo acústico e Bhattacharyya [1]. Este, em particular, trata-se de uma última proposta de modelo para um transdutor OAS, baseado no modelo de Mason, e portanto, é pertinente um maior detalhamento. Já Kiyoshi Nakamura et al [11] fez uso das equações dos modos acoplados para representar a propagação da onda acústica em um OAS, não utilizando, com isto, o modelo de Mason. Desta maneira, o modelamento do transdutor é feita de forma conjunta e não apenas um par de eletrodo, porém, o modelo de Kiyoshi não apresenta implementação direta no simulador SPICE3f4 que não permite expressões matemáticas para os parâmetros. Como trata-se de um modelo mais compacto e com os parâmetros físicos mais explícitos, resolvemos nos basear nesta proposta para desenvolvermos o nosso modelo para um dispositivo OAS, implementável num simulador SPICE.

2.6 Circuito equivalente de Bhattacharyya

O modelo é baseado no circuito proposto por Redwood [8] que fez uso do circuito equivalente de Mason, figura 2.15.

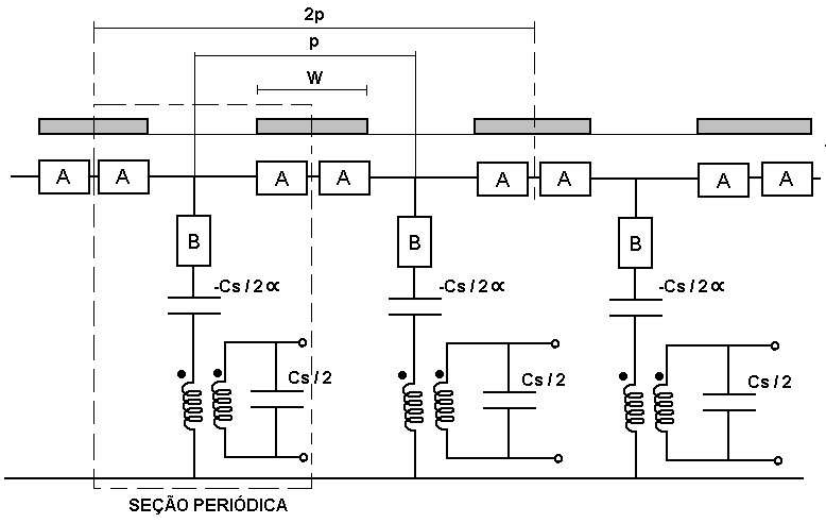

Figura 2.15: Modelo para uma seção periódica.

Este circuito representa fisicamente o intervalo entre os centros de dois eletrodos consecutivos, cada um com comprimento "W", compondo um período "p". Os elementos "A" e "B" de uma seção periódica são definidos como $jR_0 \tan(\phi/2)$ e $jR_0 \csc\phi$, respectivamente. ϕ representa o ângulo de passagem por uma seção periódica, obtido pela equação 2.15.

$$\phi = \frac{\pi\omega}{\omega_0} \tag{2.15}$$

ω_0 é a frequência central. A impedância característica (R_0), tal como em uma linha de transmissão, é dependente da capacitância estática de um par de eletrodos por unidade de comprimento (C_S). O parâmetro α fornece um valor apropriado para o modelo de propagação transversal ($\alpha = 0$) e o horizontal ($\alpha = 1$).

Para especificar o comportamento aproximado dos elementos "A" e "B" através de um circuito, foi utilizado o método de Foster que é baseado na teoria de que a localização e a natureza dos polos e zeros, acrescido do conhecimento da impedância em uma outra freqüência é suficiente para obtenção do circuito. Os respectivos circuitos dos elementos estão mostrados nas figuras 2.16 e 2.17.

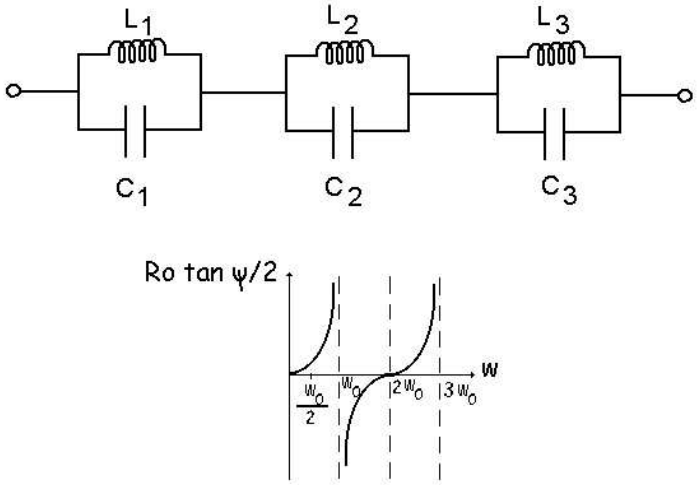

Figura 2.16: Circuito equivalente do comportamento funcional de $jR_0 \tan(\phi/2)$.

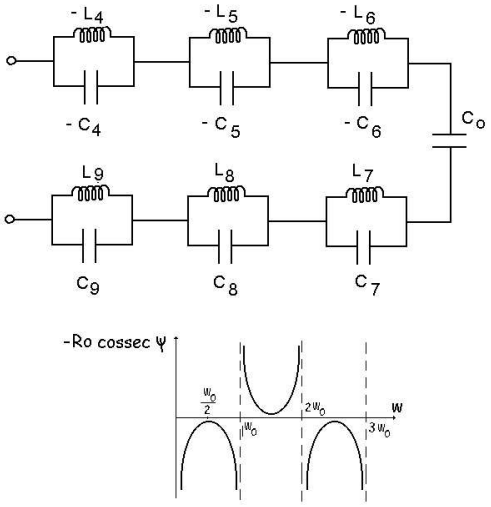

Figura 2.17: Circuito equivalente do comportamento funcional de $jR_0 \csc \phi$.

Baseado nestas aproximações foi desenvolvido um circuito para aplicação em simuladores do tipo SPICE que podem representar diretamente uma capacitância e uma indutância negativa, tais como o PSPICE® e o HSPICE®. O circuito está apresentado na figura 2.18.

O problema inerente ao modelo de Mason é a singularidade na freqüência central, este efeito pode ser diminuído utilizando o HSPICE para sua simulação e fazendo uso do *double sweep* na análise AC. Como a relação de transformação é de 1:1 não é preciso representá-lo no programa de descrição.

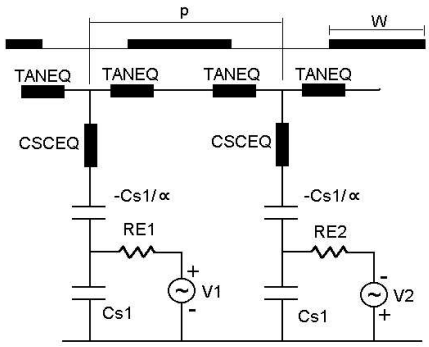

Figura 2.18: Circuito SPICE equivalente

2.6.1 Simulando um ressonador

Para a simulação de um ressonador foi considerado um transdutor com 9 pares de eletrodos em um substrato Y-Z $LiNbO_3$, com largura e espaçamento entre eles de 0,5mm. O comprimento dos eletrodos é de 2,5mm (W=2,5mm), operando numa freqüência de 74,4MHz. Nesta simulação foi considerada uma propagação transversal, e por isso, apenas os capacitores positivos foram representados. Os elementos RE1 e RE2 representam as resistências dos eletrodos. De forma a construir um ressonador, as resistências R01 e R02 foram colocadas nos terminais do transdutor, com valor igual ao da impedância característica.

2.6.2 Resultados

Na curva da resposta em freqüência do ressonador, figura 2.19, fica evidenciada a singularidade existente na freqüência de ressonância, assim como, a existência de um terceiro harmônico (223,2MHz).

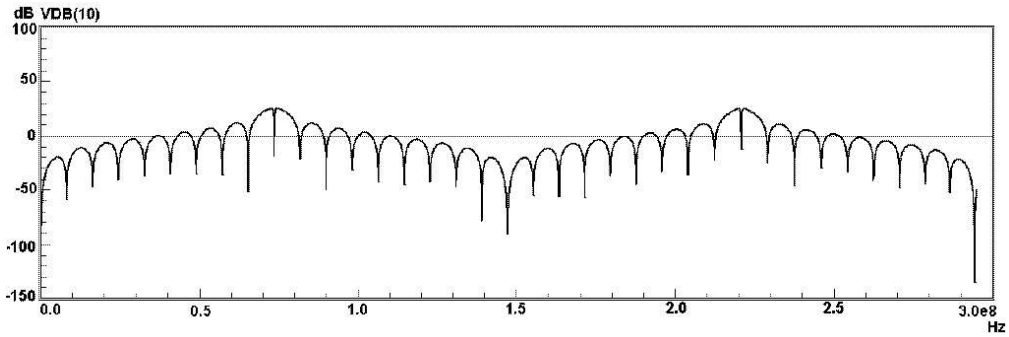

Figura 2.19: Resposta em freqüência do circuito

Programa 2.1 Descrição do circuito

```
Simulation of 74MHz,9f.p. IDT on YZLiNbO3
X1 10 11 FING
X2 11 12 FING
X3 12 13 FING
X4 13 14 FING
X5 14 15 FING
X6 15 16 FING
X7 16 17 FING
X8 17 18 FING
X9 18 19 FING
RO1 10 0 241.721K
RO2 0 19 241.721K
.SUBCKT FING 10 11
X1 10 3 TANEQ
X2 3 22 CSCEQ
X3 3 4 TANEQ
X4 4 5 TANEQ
X5 5 26 CSCEQ
X6 5 11 TANEQ
CS1 22 0 0.553703557PF
CS2 0 26 0.553703557PF
V1 2 0 AC 1
RE1 2 22 0.01
V2 0 6 AC 1
RE2 6 26 0.01
.ENDS
.SUBCKT TANEQ 30 34
L5 30 32 673.786uH
C5 30 32 6.93978E-03pF
L6 32 33 87.3427uH
C6 32 33 5.94839E-03pF
L7 33 34 56.5980uH
C7 33 34 3.30466E-03pF
.ENDS
.SUBCKT CSCEQ 50 57
L5 50 51 -336.893uH
C5 50 51 -1.38796E-02pF
L6 51 52 -43.6713uH
C6 51 52 -1.18968E-02pF
L7 52 53 -28.2990uH
C7 52 53 -6.60932E-03pF
C14 53 54 2.81862E-02pF
L17 54 55 87.0945uH
C17 54 55 1.34220E-02pF
L18 55 56 26.1284uH
C18 55 56 1.11850E-02pF
L19 56 57 21.2898uH
C19 56 57 6.10091E-03pF
.ENDS
.AC LIN 295 1 295MEG
.END
```

2.7 Conclusão

Vimos que um dispositivo OAS pode ser considerado como uma sucessão de eletrodos metalizados, dispostos com polaridades alternadas sobre um material piezoelétrico, de forma que, ao ser aplicado um sinal de RF com uma freqüência propícia, é gerada uma onda acústica de superfície, onde metade de sua energia se propaga para a direita e a outra metade para a esquerda. Esta propagação sofre atenuações devido a perdas resistivas, a conversão eletro-acústica e ao material piezoelétrico. Esta perda de inserção, como são chamadas estas atenuações, são da ordem de 15 a 30dB, em dispositivos práticos, como filtros. A distância entre dois eletrodos consecutivos de mesmo potencial representa um comprimento de onda, sendo a largura de banda de um transdutor inversamente proporcional ao número de pares de eletrodos.

Modelos para transdutores a Ondas Acústicas de Superfície têm sido propostos desde Smith, em artigo publicado em 1969 [4], até recentemente, em 1995, onde um modelo para um par de eletrodos foi proposto por Bhattacharyya [1]. Todos os modelos, até então apresentados, modelam apenas um par de eletrodos, o que dificulta a representação de transdutores longos. Além disso, parâmetros físicos, como: acoplamento, freqüência central, banda passante, etc., não são modelados diretamente pelos componentes do modelo. Uma peculiaridade inerente ao modelo de Bhattacharyya é a existência de um terceito harmônico na sua resposta em freqüência, fato este que discorda do comportamento do dispositivo real.

A necessidade de simular sensores inteligentes a Onda Acústica de Superfície, onde existe a necessidade de integração sensor x eletrônica, foi a motivação para habilitar o simulador SPICE para tal tarefa. Desta forma, o desenvolvimento de de um macromodelo que represente de forma simples um transdutor OAS, capaz de ser implementado desde a versão gratuita até o ELDO® da mentor graphics, é o objetivo. Para isso, conhecer as formas de modelamento do SPICE, seja através de um macromodelo ou até mesmo de um modelo físico incorporado ao código fonte, torna-se imperioso.

Capítulo 3

Modelamento no SPICE

A simulação está sendo aceita e empregada, cada vez mais, como uma ferramenta que permite simular soluções de seus problemas. O modelo descreve a aparência do elemento e a simulação mostra o seu comportamento. Neste capítulo vamos explicitar como podemos representar um modelo no simulador SPICE.

3.1 Uso de um Macromodelo

Um mecanismo para simularmos um dispositivo no SPICE é descrever o seu comportamento como um circuito eletrônico. Desta maneira, chamamos este modelo de macromodelo. Para que este circuito represente o comportamento do dispositivo a ser simulado é preciso descrevê-lo no SPICE como um subcircuito, assim poderemos referencia-lo de modo semelhante a um modelo físico pré-existente. O trecho que conterá a descrição, atendendo aos formatos do SPICE, terá início com `.SUBCKT SUBNOME N1 <N2 N3 ...>` e encerrado com `.ENDS <SUBNOME>`. Caso o nome do subcircuito, após `.ENDS`, seja explicitado, indicará qual subcircuito estará sendo finalizado, já que aninhamentos de subcircuitos são permitidos. Se o nome for omitido, todos os subcircuitos serão finalizados. O apêndice A pode ser consultado para maiores detalhes sobre a estrutura do SPICE.

Esta é uma forma bem prática para implementar um modelo, caso o seu comportamento possa ser bem representado por um circuito eletrônico. Conhecer então este circuito torna-se o pré-requisito para efetivarmos a simulação. A qualidade deste modelo torna-se tão melhor quanto a resposta do circuito seja semelhante ao dispositivo

real que se quer simular.

3.2 Exemplo de um modelamento através de um macromodelo

Trata-se de um modelo de transdutor TSM proposto por Alf Püttmer [5], baseado no modelo de W. Marshall [12]. Este exemplo é especialmente relevante já que se fez uso da linha de transmissão elétrica como modelamento da propagação da onda acústica.

Na figura 3.1 está representado um transdutor TSM, onde assumimos que a onda se propaga na direção z e que um campo elétrico (E) e a densidade de fluxo elétrico (D) também estão na direção z. Considerando ζ a posição da partícula, u a velocidade e f a força, as equações que governam a onda são:

$$\frac{df}{dz} = -\rho A_z s u \tag{3.1}$$

$$c\frac{d\zeta}{dz} = -\frac{1}{A_z} f + hD \tag{3.2}$$

$$E = -h\frac{d\zeta}{dz} + \frac{1}{\epsilon} D \tag{3.3}$$

sendo as equações 3.1, 3.2 e 3.3 a Lei de Newton, Lei de Hook e o Acoplamento Piezoelétrico, respectivamente, onde s é a freqüência complexa, ρ é a densidade, A_z é a área da secção transversal ao eixo z do cristal, c é a constante elástica relativa (N/m^2), h é a constante piezoelétrica ($N \cdot m^4/C$) e ϵ a permissividade (F/m).

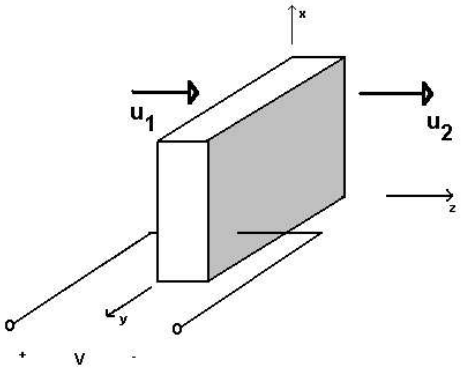

Figura 3.1: Transdutor TSM

Considerando que uma corrente i percorra o cristal, uma carga q estará presente

nos eletrodos de forma que $q = i/s$. Como $D = q/A_z$, temos que:

$$D = \frac{i}{sA_z} \tag{3.4}$$

Como a densidade do fluxo elétrico não varia na direção do eixo z, ou seja, $dD/dz = 0$, podemos considerar também que $\frac{d(hi/s)}{dz} = 0$, logo podemos subtrair este termo da parte esquerda da equação 3.1 sem comprometermos a sua igualdade.

$$\frac{d}{dz}\left[f - \frac{h}{s}i\right] = -\rho A_z s u \tag{3.5}$$

Sabemos que $\zeta = u/s$, então aplicando estas relações na equação 3.2, temos que:

$$\frac{du}{dz} = -\frac{s}{A_z c}\left[f - \frac{h}{s}i\right] \tag{3.6}$$

A tensão aplicada ao transdutor é obtida integrando-se a equação 3.3 em z entre os limites $z = 0$ e $z = l$, onde l é a espessura do transdutor, ou seja:.

$$v = \frac{h}{s}[u_1 - u2] + \frac{1}{C_0 s}i \tag{3.7}$$

onde:

$$u_1 = u(0)$$

$$u_2 = u(l)$$

$$C_0 = \epsilon A_z/l$$

sendo C_0 a capacitância entre os eletrodos.

3.2.1 O emprego da Linha de Transmissão

As equações da linha de transmissão para a tensão e corrente são:

$$\frac{dV}{dz} = -(R + sL)I \tag{3.8}$$

$$\frac{dI}{dz} = -(G + sC)V \tag{3.9}$$

onde L e C representam a indutância por unidade de comprimento e a capacitância por unidade de comprimento, respectivamente.

Considerando o caso $R = G = 0$, podemos fazer uma analogia entre as equações 3.5 e 3.8, como também entre as equações 3.6 e 3.9, onde V é análogo a $[f - (h/s)i]$ e I é análogo a u, temos então que os parâmetros da linha de transmissão seriam:

$$
\begin{aligned}
L &= \rho A_z \\
C &= \frac{1}{A_z c} \\
u_p &= \frac{1}{LC^{1/2}} = (c/\rho)^{1/2} \\
Z_0 &= (L/C)^2 = A_z(\rho c)^{1/2} = \rho A_z u_p
\end{aligned}
$$

sendo u_p e Z_0 a velocidade de fase e a impedância característica, respectivamente.

O circuito proposto por W. Marshall [12] que representa as equações 3.5, 3.6 e 3.7 está mostrado na figura 3.2.

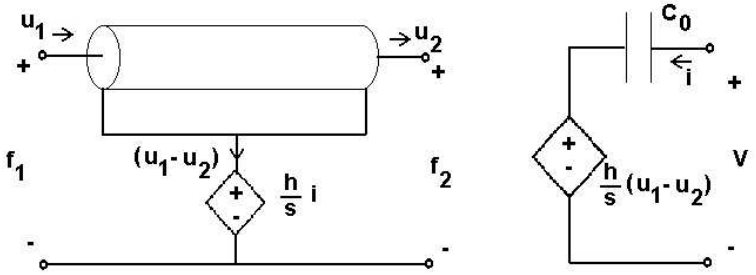

Figura 3.2: Circuito análogo do transdutor TSM

3.2.2 Representação das perdas

Como o modelo apresentado não possui uma boa precisão para transdutores com baixo fator de qualidade (Q), onde as perdas acústicas são significativas, Alf püttmer propôs uma nova maneira de representar os parâmetros da linha de transmissão, de forma que as perdas fossem consideradas.

Em uma linha de transmissão com perdas temos que a impedância característica é dada por [13]:

$$
Z_0 = \sqrt{\frac{R + j\omega L}{G + j\omega C}} \tag{3.10}
$$

sendo a constante de propagação definida como:

$$\gamma = \alpha + j\beta = \sqrt{(R + j\omega L)(G + j\omega C)} \tag{3.11}$$

onde α é a constante de atenuação e β a constante de fase, de modo que a velocidade de fase é dada por:

$$v_p = \frac{\omega}{\beta} \tag{3.12}$$

Assumindo $G = 0$ e $R \ll \omega L$ (alta freqüência) as equações 3.10 e 3.11 podem ser aproximadas como:

$$Z_0 \approx \sqrt{\frac{L}{C}} \tag{3.13}$$

$$\gamma \approx \frac{R}{2}\sqrt{\frac{C}{L}} + j\omega\sqrt{LC} \tag{3.14}$$

sendo a velocidade de fase:

$$v_p \approx \frac{1}{\sqrt{LC}} \tag{3.15}$$

Pela equação 3.14 temos que a constante de atenuação é da forma:

$$\alpha \approx \frac{R}{2}\sqrt{\frac{C}{L}} \tag{3.16}$$

ainda podemos rearrumar as equações 3.15 e 3.16 da seguinte maneira:

$$\alpha \approx \frac{\omega}{2v_p}.\frac{R}{\omega L} = \frac{\omega}{2v_p}\sigma \tag{3.17}$$

sendo σ o fator de perdas ($\sigma = \frac{1}{Q}$)

Para sólidos sem separações, Q é constante e α aumenta proporcionalmente a freqüência, logo, R deve também crescer com a freqüência. Para isso temos que:

$$R = R^* \cdot \omega \tag{3.18}$$

3.2.3 Implementação do Macromodelo SPICE

O circuito a ser implementado é o descrito na figura 3.2, tendo os parâmetros da linha de transmissão considerando as perdas de propagação da onda pelo transdutor. Vimos que o modelo possui o operador de Laplace (s) e uma resistência por unidade

de comprimento da linha de transmissão em função da freqüência. O PSPICE® não apresenta dificuldades para implementarmos este circuito tal qual está representado, porém o SPICE de Berkeley não permite que representemos R em função de ω, nem tão pouco podemos fazer o uso de s diretamente na descrição do circuito. Para isso, temos que utilizar algumas técnicas para a representação de h/s e calcularmos R para a freqüência de ressonância do transdutor ($R = \omega L/Q$).

O trecho de código da implementação do macromodelo no PSPICE® está mostrado abaixo e o circuito representado na figura 3.3

Programa 3.1 Macromodelo PSPICE®

```
.SUBCKT PZ35 E B F
T1 B 1 F 1 LEN="l" R={"R*"*SQRT(-S*S)} L="L" G=0 C="C"
V1 1 2
E1 2 0 LAPLACE {I(V2)}={"h"/S}
V2 E 3
C0 3 0 "C0"
F1 0 3 V1 "h*C0"
.ENDS
```

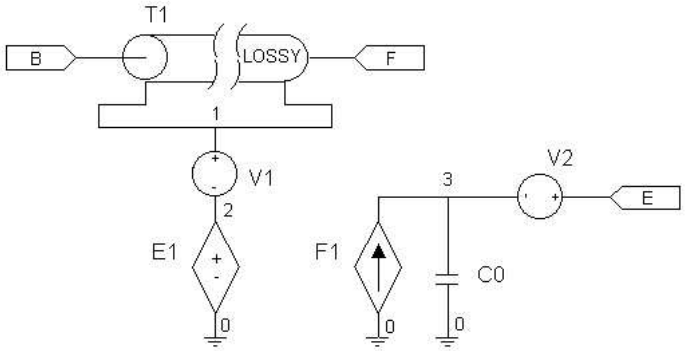

Figura 3.3: Subcircuito PSPICE®

3.2.4 Implementação de h/s no SPICE

Uma maneira de representarmos h/s é fazermos uso de um circuito RC paralelo em conjunto com uma fonte controlada de corrente e de ganho h de forma que a tensão represente $(h/s)i$, figura 3.4

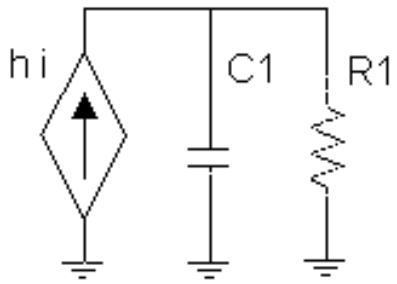

Figura 3.4: Circuito RC paralelo

Para que a fonte controlada de tensão $E1$ seja igual a $(h/s)i$, como exposto na figura 3.2, é preciso que $\frac{1}{R_1 C_1} \ll$ freqüência central (ω_0) e que $C_1 = 1$, pois:

$$hi = C_1 s V_c + \frac{V_c}{R_1}$$

$$V_c = \frac{R_1}{R_1 C_1 s + 1} hi$$

Desta forma o trecho de código é mostrado no programa 3.2 e o circuito completo está representado na figura 3.5. Os resultados da magnitude da impedância e da fase estão apresentados nas figuras 3.6 e 3.7, respectivamente.

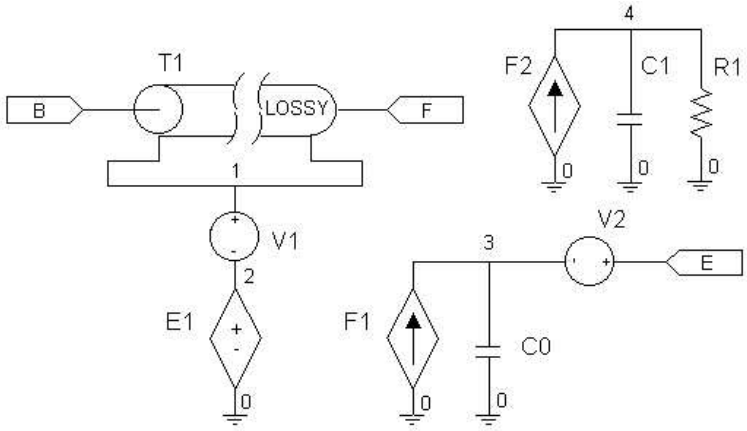

Figura 3.5: Subcircuito SPICE

3.3 Uso de um dispositivo físico

Uma outra forma de simularmos um dispositivo é explicitar o seu comportamento através de equações que definam os relacionamentos matemáticos entre suas variáveis

Programa 3.2 Macromodelo SPICE

```
.SUBCKT PZ35 E B F
T1 B 1 F 1 LEN="l" R="wL/Q" L="L" G=0 C="C"
V1 1 2
E1 2 0 4 0 1
V2 E 3
C0 3 0 "C0"
F1 0 3 V1 "h*C0"
F2 0 4 V2 "h"
R1 4 0 1E3
C1 4 0 1
.ENDS
```

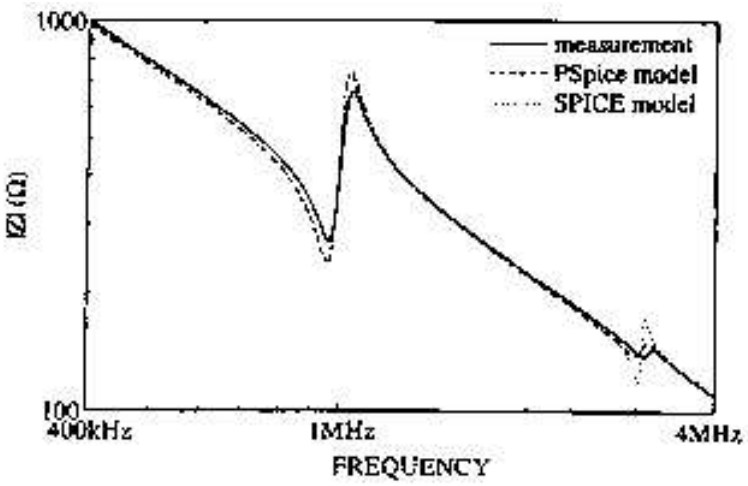

Figura 3.6: Magnitude da Impedância. Fonte: Alf Pütmer et al. SPICE Model for Lossy Piezoceramic Transducers.

físicas. Desta maneira estaremos implementando um modelo físico, que no SPICE terá a necessidade de ter o seu código fonte alterado, para que possamos adicionar este novo modelo aos pré-existentes. Isto não é uma tarefa fácil, pois necessita conhecimento de programação e limita o seu uso ao SPICE que tenha o código fonte disponível. A versão 3f4 de Berkeley foi escrita em linguagem C e tem o seu código fonte aberto, que pode ser obtido de vários sites na Internet. Um deles é o: `ftpsunsite.unc.edu` que contém o arquivo `spice3f4.tar.gz`, no diretório `pub/Linux/apps/circuits`.

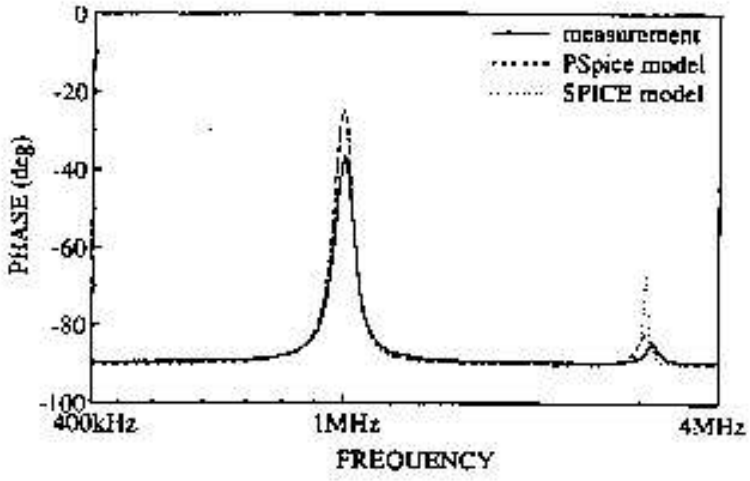

Figura 3.7: Fase da Impedância. Fonte: Alf Pütmer et al. SPICE Model for Lossy Piezoceramic Transducers.

3.4 Conclusão

No SPICE, pode-se descrever o comportamento de um dispositivo como um circuito eletrônico. Esta forma bem prática para implementar um modelo é chamada de macromodelo. A qualidade deste modelo é proporcional a fidelidade da resposta do circuito ao comportamento físico do dispositivo. Conhecer então este circuito é requesito básico para a simulação. Técnicas de desenvolvimento de circuitos para dispositivos de Onda Acústica de Volume foram praticadas neste capítulo.

Uma outra forma de modelamento é acrescentar ao código fonte do SPICE, um novo modelo físico. Para isso é preciso um versão de código aberto, como o SPICE3f4 de Berkeley. Não se trata de uma tarefa simples pois requer conhecimento e habilidade em programação C. O que restringe também a sua aplicação ao SPICE de Berkeley. É necessário, então, entender o mecanismo de inclusão de um novo componente no SPICE3f4.

Uma vez entendido o processo de modelamento no SPICE, um novo macromodelo para um transdutor OAS pode ser desenvolvido, bem como uma linha de transmissão acústica pode ser acrescentada ao SPICE3f4, de forma que, os parâmetros inerentes a linha sejam explicitados em termos de seu comprimento (LEN), resistência (R) e velocidade de propagação (VEL). Este desenvolvimento é o objetivo deste trabalho e está demonstrado no próximo capítulo.

Capítulo 4

Um modelo de um OAS para o SPICE

Neste capítulo é proposto um modelo para um OAS fazendo uso das equações dos modos acoplados [16] e como podemos representá-lo como um macromodelo no SPICE.

4.1 As equações dos modos acoplados para um OAS

As equações dos modos acoplados têm sido bastante utilizadas em análises de guias de ondas ópticos, óptica não linear e mais recentemente em transdutores OAS [17]. Consideraremos a onda de superfície se propagando nas direções $+x$ e $-x$, em uma região sem a presença de eletrodos. Podemos representá-la por suas amplitudes $v^+(x)$ e $v^-(x)$, respectivamente, sendo a sua dependencia implícita com o tempo da forma: $e^{j\omega t}$.

$$\begin{aligned} v^+(x) &= A^+(x)e^{-jk_R x} \\ v^-(x) &= A^-(x)e^{jk_R x} \end{aligned} \tag{4.1}$$

Como estamos desprezando as perdas podemos escrever da forma:

$$\begin{aligned} \frac{dv^+(x)}{dx} &= -jk_R v^+(x) \\ \frac{dv^-(x)}{dx} &= jk_R v^-(x) \end{aligned} \tag{4.2}$$

Uma vez na presença do transdutor interdigitado com espaçamento entre os eletrodos de $\Lambda = \lambda/2$, comprimento total L e com N pares de dentes, a onda sofre uma alteração na sua constante de propagação de Δk e um acoplamento entre elas deve ser considerado. Incluindo a ação da tensão V aplicada ao transdutor e assumindo que a sua variação por comprimento dos eletrodos é pequena, temos:

$$\begin{aligned} \frac{dv^+(x)}{dx} &= -j(k_R + \Delta k)v^+(x) + j\kappa_{21}v^-(x) + j\alpha V \\ \frac{dv^-(x)}{dx} &= j(k_R + \Delta k)v^-(x) + j\kappa_{12}v^+(x) + j\beta V \end{aligned} \tag{4.3}$$

onde, pela conservação da energia e reciprocidade do sistema, temos: $\kappa_{12} = -\kappa_{21}^*$ e $\beta = -\alpha^*$

Na estrutura periódica formada pelos eletrodos (transdutor) κ_{12} é função de x. A decomposição de Fourier de κ_{12} para uma estrutura periódica [18] de período Λ é da forma:

$$\kappa_{12} = \sum_n \kappa(n) e^{jnk_g x} \tag{4.4}$$

onde $k_g = \frac{2\pi}{\Lambda}$.

Explicitando as variáveis na equação 4.2, temos:

$$\begin{aligned} v^+(x) &= A_1(x)e^{-j\frac{k_g}{2}x} \\ v^-(x) &= A_2(x)e^{j\frac{k_g}{2}x} \end{aligned}$$

substituindo na equação 4.3, obtemos:

$$\begin{aligned} \frac{dA_1}{dx} &= -j(k_R - \frac{k_g}{2} + \Delta k)A_1 + j\sum_n \kappa(n)A_2 e^{j(n+1)k_g x} + j\alpha V \\ \frac{dA_2}{dx} &= j(k_R - \frac{k_g}{2} + \Delta k)A_2 + j\sum_n \kappa^*(n)A_1 e^{-j(n+1)k_g x} - j\alpha^* V \end{aligned}$$

O coeficiente de acoplamento entre as ondas que se propagam nas direções $+x$ e $-x$ não variam ao longo do transdutor. De todos os componentes de Fourier somente o caso $n = -1$ produz um o coeficiente independente da posição. De forma a simplificar as equações, definiremos:

$$\delta \equiv k_R - \frac{k_g}{2} + \Delta k$$

Omitindo os parêntesis para κ, temos equações de modos acoplados, da forma:

$$\begin{aligned}\frac{dA_1}{dx} &= -j\delta A_1 + j\kappa A_2 + j\alpha V \\ \frac{dA_2}{dx} &= j\delta A_2 - j\kappa^* A_1 - j\alpha^* V\end{aligned} \tag{4.5}$$

4.2 Proposta de solução

Kiyoshi Nakamura, em seu artigo [16], propôs uma equação de modos acoplados escrita da forma:

$$\begin{aligned}\frac{dA^+(x)}{dx} &= -j\kappa_{11}A^+(x) - j\kappa_{12}e^{j2\delta x}A^-(x) + j\zeta e^{j\delta x}V \\ \frac{dA^-(x)}{dx} &= j\kappa_{12}e^{-j2\delta x}A^+(x) + j\kappa_{11}A^-(x) - j\zeta e^{-j\delta x}V\end{aligned} \tag{4.6}$$

onde V é a tensão aplicada ao transdutor interdigitado (IDT), ζ é a constante associada à conversão da grandeza elétrica para a acústica e δ é definido como:

$$\delta = k - k_0 \tag{4.7}$$

onde:

$$k_0 = \frac{2\pi}{\lambda} \quad , \quad k = \frac{\omega}{V_f}$$

Uma solução para a equação 4.6 é expressa como:

$$\begin{aligned}v^+(x) &= \{h_1 e^{-j\beta_1 x} + p h_2 e^{-j\beta_2 x} + q\zeta V\}e^{-jk_0 x} \\ v^-(x) &= \{p h_1 e^{-j\beta_1 x} + h_2 e^{-j\beta_2 x} + q\zeta V\}e^{jk_0 x}\end{aligned} \tag{4.8}$$

onde β_1,β_2, p e q são dados por:

$$\begin{aligned}\beta_1, \beta_2 &= \pm\sqrt{(\delta + \kappa_{11})^2 - \kappa_{12}^2} \\ p &= (\beta_1 - \delta - \kappa_{11})/\kappa_{12} \\ q &= 1/(\delta + \kappa_{11} + \kappa_{12})\end{aligned} \tag{4.9}$$

É importante salientar que a propagação da onda na direção $+x$ se dá agora com um número de onda $k_0 + \beta_1$ e $k_0 + \beta_2$ e com magnitudes h_1 e h_2, respectivamente. Considerando o casamento acústico das portas ou um transdutor com infinitos pares

de eletrodos ($\beta_1 = \beta_2 = 0$), as freqüências das bordas da *stopband* são obtidas. Neste caso, temos: $\delta + k_{11} = \pm k_{12}$ e $\delta = (\omega_s/V_f) - k_0$, logo:

$$\begin{aligned} \frac{\omega_{s1}}{V_f} &= k_0 - \kappa_{11} - \kappa_{12} \\ \frac{\omega_{s2}}{V_f} &= k_0 - \kappa_{11} + \kappa_{12} \end{aligned} \tag{4.10}$$

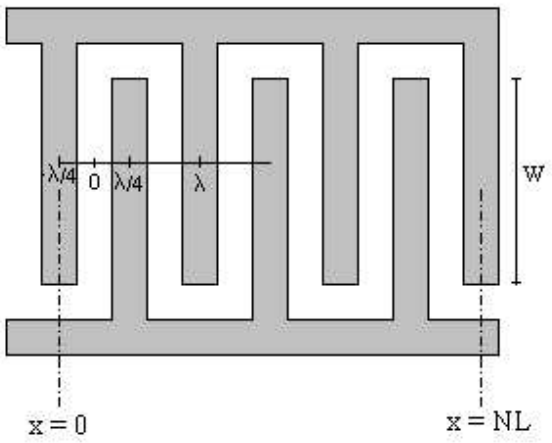

Figura 4.1: Transdutor interdigitado

Considere as portas, 1 e 2, no centro dos eletrodos das extremidades do transdutor interdigitado, $x = 0$ e $x = NL$, como mostrado na figura 4.1. Da equação 4.8, as amplitudes nas portas acústicas são escritas como:

$$\begin{aligned} v^+(0) &= h_1 + ph_2 + q\zeta V \\ v^-(0) &= ph_1 + h_2 + q\zeta V \\ v^+(NL) &= \pm(e_1^{-1}h_1 + e_1 ph_2 + q\zeta V) \\ v^-(NL) &= \pm(e_1^{-1}ph_1 + e_1 h_2 + q\zeta V) \end{aligned} \tag{4.11}$$

onde:

$$e_1 = e^{j\beta_1 NL}$$

Os sinais positivo e negativo correspondem aos casos $N = i$ (número inteiro de pares) e $N = i + 0,5$ (apresenta o último par pela metade), respectivamente. As velocidades totais nas duas portas acústicas podem ser expressas como:

$$\begin{aligned} v_1 &= v^+(0) + v^-(0) = (1+p)(h_1 + h_2) + 2q\zeta V \\ v_2 &= -\{v^+(NL) + v^-(NL)\} = \mp[(1+p)\{e_1^{-1}h_1 + e_1 h_2\} + 2q\zeta V] \end{aligned} \tag{4.12}$$

As forças nas portas acústicas são consideradas proporcionais à diferença entre v^+ e v^- e podem ser definidas convenientemente como:

$$\begin{aligned} F_1 &= v^+(0) - v^-(0) = (1-p)(h_1 - h_2) \\ F_2 &= v^+(NL) - v^-(NL) = \pm[(1-p)\{e_1^{-1}h_1 - e_1 h_2\}] \end{aligned} \tag{4.13}$$

Como podemos fazer a analogia de que a corrente é proporcional à velocidade $(v^+(x) + v^-(x))$ ao longo de x, calculamos a corrente total I integrando ao longo de todo o transdutor acrescentando a corrente relativa aos capacitores formados pelos pares de eletrodos, ou seja:

$$\begin{aligned} I &= \eta \int_0^{NL} [(1+p)\{h_1 e^{-j\beta_1 x} + h_2 e^{-j\beta_2 x}\} + 2q\zeta V]dx + j\omega N C_s V \\ &= j\eta[(1+p)\{\frac{h_1}{\beta_1}(e_1^{-1} - 1) + \frac{h_2}{\beta_2}(e_1 - 1)\}] + 2q\zeta\eta NLV + j\omega N C_s V \end{aligned} \tag{4.14}$$

Onde η é a constante associada à conversão da grandeza acústica para a grandeza elétrica e C_s é a capacitância de um par de eletrodos, ou seja:

$$C_s = \frac{1}{2}(\epsilon_p + \epsilon_0)W \tag{4.15}$$

Expressando h_1 e h_2 em termos de F_1 e F_2, as equações 4.14 e 4.12 ficam da forma:

$$\begin{aligned} I &= (j\omega N C_s + 2q\zeta\eta NL)V + \frac{\eta(1+p)}{j\beta(1-p)}F_1 \mp \frac{\eta(1+p)}{j\beta(1-p)}F_2 \\ v_1 &= 2q\zeta V + \frac{1+p}{1-p}\frac{1}{j\tan 2\theta}F_1 \mp \frac{1+p}{1-p}\frac{1}{j\text{sen}2\theta}F_2 \\ v_2 &= \mp 2q\zeta V \mp \frac{1+p}{1-p}\frac{1}{j\text{sen}2\theta}F_1 + \frac{1+p}{1-p}\frac{1}{j\tan 2\theta}F_2 \end{aligned} \tag{4.16}$$

onde:

$$\begin{aligned} \theta &= \beta NL/2 \\ \beta &\triangleq \beta_1 = -\beta_2 \end{aligned} \tag{4.17}$$

Um transdutor de onda acústica usando o efeito piezoelétrico pode ser expresso por um circuito recíproco se a analogia força-tensão for empregada. Então o teorema da reciprocidade estabelece a seguinte relação entre η e ζ:

$$2q\zeta = \frac{\eta(1+p)}{j\beta(1-p)} \qquad \therefore \qquad \eta = 2j\zeta \tag{4.18}$$

Com isso a equação 4.14 pode ser escrita da forma:

$$\frac{dI(x)}{dx} = j2\zeta e^{-j\delta x}A^{+}(x) + j2\zeta e^{j\delta x}A^{-}(x) + j\omega CV \tag{4.19}$$

onde $I(x)$ é a corrente que percorre o transdutor e C é a capacitância por unidade de comprimento. Utilizando a equação 4.18, podemos representar a equação 4.16 da seguinte forma:

$$\begin{bmatrix} I \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} j\omega C_T + \frac{\phi^2}{j2\theta Z_0} & \frac{\phi}{j2\theta Z_0} & \frac{\mp\phi}{j2\theta Z_0} \\ \frac{\phi}{j2\theta Z_0} & \frac{1}{jZ_0\tan 2\theta} & \frac{\mp 1}{jZ_0\mathrm{sen}2\theta} \\ \frac{\mp\phi}{j2\theta z_0} & \frac{\mp 1}{jZ_0\mathrm{sen}2\theta} & \frac{1}{jZ_0\tan 2\theta} \end{bmatrix} \begin{bmatrix} V \\ F_1 \\ F_2 \end{bmatrix} \tag{4.20}$$

onde:

$$\begin{aligned} Z_0 &= \frac{1-p}{1+p} = \frac{1}{q\beta} = \sqrt{\frac{\omega-\omega_{s1}}{\omega-\omega_{s2}}} \\ \phi &= \eta NL = 2j\zeta NL \\ C_T &= NC_s \end{aligned} \tag{4.21}$$

O circuito equivalente para $N = i$ é mostrado na figura 4.2. Z_0 é a impedância característica acústica. No caso de $N = i+1/2$, a equação 4.20 será da mesma forma, apenas troca-se F_2 por $-F_2$ e v_2 por $-v_2$.

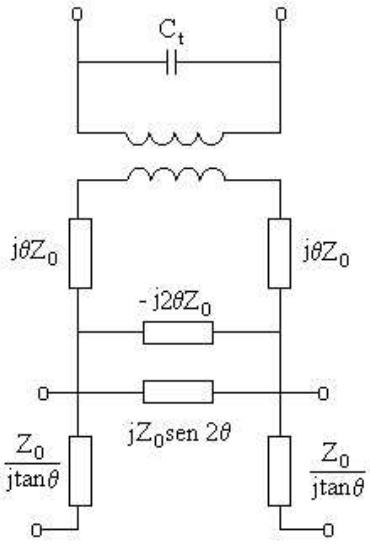

Figura 4.2: Circuito equivalente

4.3 Representando como um macromodelo SPICE

De posse da matriz admitância do OAS, o objetivo agora é construir um circuito que represente tal comportamento e que possa ser simulado pelo SPICE.

A impedância $j2\theta Z_0$ pode ser escrita da forma:

$$j2\theta Z_0 = j\frac{NL}{q} = j\frac{NL}{V_f}(\omega - \omega_{s1}) = j\frac{N}{f_0}(\omega - \omega_{s1}) \tag{4.22}$$

Em um circuito série LC, temos:

$$Z(\omega) = jL(\omega - \frac{1}{\sqrt{LC}})\frac{(\omega + \frac{1}{\sqrt{LC}})}{\omega}$$

Próximo à freqüência central, podemos utilizar a seguinte aproximação:

$$Z(\omega) \sim j2L(\omega - \frac{1}{\sqrt{LC}})$$

Efetuando a comparação direta com a equação 4.22, de forma a representá-la através de um circuito série LC, temos que:

$$\begin{aligned} L &= \frac{N}{2f_0} \\ C &= \frac{1}{L\omega_{s1}^2} \sim \frac{1}{L\omega_0^2} \end{aligned} \tag{4.23}$$

A constante η para um transdutor com uma razão de metalização de 0.5 é dada por [19]:

$$\eta^2 = \frac{\sqrt{2.268\omega_o C_s K^2}}{L} \tag{4.24}$$

onde K^2 é o acoplamento piezoelétrico do OAS e ω_0 expresso por:

$$\omega_0 = k_0 V_f = 2\pi V_f / L$$

Vimos que $\phi = \eta NL$, então:

$$\phi = N\sqrt{2.268\omega_0 C_s K^2}$$

A matriz 2x2 da parte direita inferior da equação 4.20 representa a parte acústica do modelo, e possui a seguinte forma:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{jZ_0 \tan 2\theta} & \frac{\mp 1}{jZ_0 \mathrm{sen} 2\theta} \\ \frac{\mp 1}{jZ_0 \mathrm{sen} 2\theta} & \frac{1}{jZ_0 \tan 2\theta} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \end{bmatrix}$$

Esta matriz pode ser representada por um circuito-π, figura 4.3, onde:

$$Z_1 = jZ_0 \text{sen} 2\theta$$
$$Z_2 = \frac{Z_0}{j \tan \theta}$$

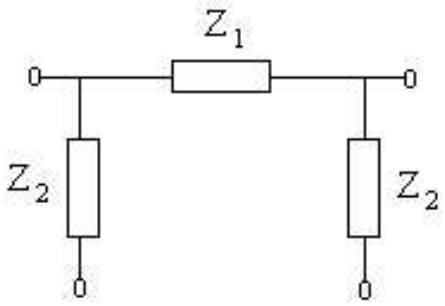

Figura 4.3: Circuito-π de uma Linha de Transmissão

De uma forma geral, uma linha de transmissão pode ser representada por um circuito-π [20] da seguinte forma:

$$Z_1 = Z_0 \text{senh} 2\phi$$
$$Z_2 = \frac{Z_0}{\tanh \phi}$$

onde: $2\phi = (\alpha + j\beta)x$.

No caso sem perdas($\alpha = 0$), temos que:

$$\text{senh} j\beta x = j\text{sen}\beta x$$
$$\tanh j\beta x = j \tan \beta x$$

O SPICE possui dois modelos para linha de transmissão. Um para linha com perdas (LTRA) e outra para sem perdas (TRA). Podemos representar a porção acústica por uma linha de transmissão com perdas que possui os seguintes parâmetros:

LEN - Comprimento da linha de transmissão

R - Resistência por unidade de comprimento

G - Condutância por unidade de comprimento

L - Indutância por unidade de comprimento

C - Capacitância por unidade de comprimento

Para isso, devemos considerar:

$$LEN = \frac{1}{\sqrt{LC}} \times \frac{N}{f_0}$$

$$R = G = 0$$

$$\sqrt{\frac{L}{C}} = Z_0 \sim 1$$

Para o caso exclusivo do SPICE de Berkeley, o comprimento da linha foi determinado empiricamente, da forma:

$$LEN = \frac{1}{\sqrt{LC}} \times \sqrt{\frac{N}{f_0}} \times 10^5$$

Para uma linha sem perdas, seus parâmetros são:

TD - Tempo de atraso

Z0 - Impedância característica da linha

E neste caso, teremos:

$$TD = \frac{N}{f_0} \tag{4.25}$$

A região entre os transdutores pode ser também modelada por uma linha de transmissão, tendo como parâmetros:

$$TD = \frac{d}{V_f}$$

$$Z_0 = 1$$

sendo d a distância entre os transdutores e V_f a velocidade com que a onda se propaga.

Para representarmos a impedância negativa $-j2\theta Z_0$ no modelo SPICE, devemos proceder da seguinte maneira:

$$Z_{neg} = \frac{V}{(1-\alpha)I_z} = \frac{Z}{1-\alpha}$$

Para $\alpha = 2$ temos que $Z_{neg} = -Z$. Isto pode ser implementado colocando-se uma fonte de corrente controlada por corrente com ganho α em paralelo com a impedância em referência, figura 4.4.

Com isso, podemos construir o macromodelo SPICE que representa a equação 4.20 como ilustrado na figura 4.5.

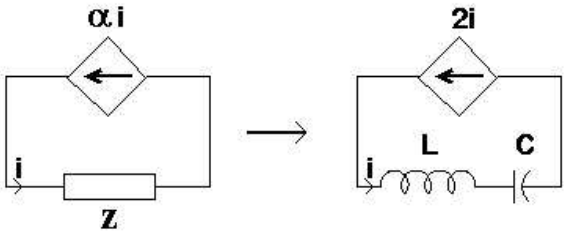

Figura 4.4: Impedância negativa

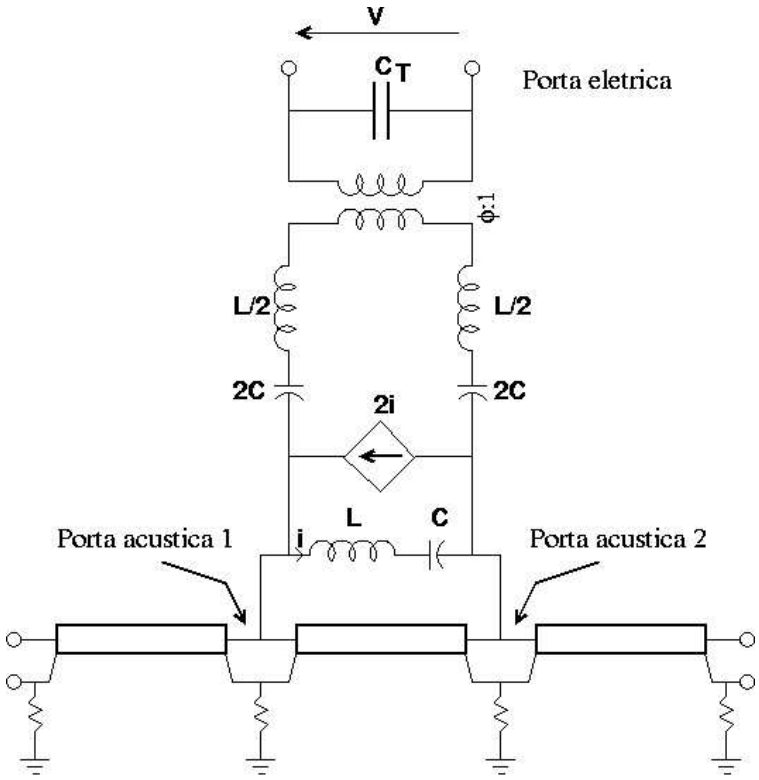

Figura 4.5: Macromodelo SPICE

4.4 Simulando um exemplo

Como teste do nosso modelo foi simulado um circuito ressonador que se trata de um transdutor contendo uma resistência, em cada porta acústica, de valor igual a impedância característica da linha (R=Z0=1), figura 4.6.

A sua alimentaçao foi feita através de uma fonte de freqüência igual a 50MHz, sendo esta a freqüência central de projeto deste dispositivo OAS. Os parâmetros do modelo OAS para uma freqüência central de 50MHz é mostrado através da tabela 4.1

A resposta em freqüência deste circuito é mostrado na figura 4.7, o que revela o modelo eficaz como ferramenta de conhecimento do comportamento do dispositivo

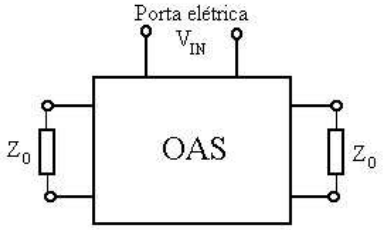

Figura 4.6: Circuito ressonador

Parâmetro	Equação	$f_0 = 50MHz$
C_T	$\frac{N}{2}\epsilon_0(1+\epsilon_{sup})W$	$3.267pF$
L	$\frac{N}{2f_0}$	$200nH$
C	$\frac{1}{L\omega_0^2}$	$50pF$
ϕ^2	$2.268N^2\omega_0 NC_T K^2$	47μ
$Lpri$	1	1
$Rpri$	0.0001	0.0001
$Lsec$	ϕ^2	47μ
$Rsec$	0.0001	0.0001
$Ktrans$	1	1
$Tidt$	$\frac{N}{f_0}$	$0.4\mu s$
LEN	$\frac{1}{\sqrt{LC}} \times \sqrt{\frac{N}{f_0}} \times 10^5$	63.25
Z_0	1	1

Tabela 4.1: Parâmetros do modelo

antes de ser criado [21].

O trecho de código que representa o subcircuito do exemplo anterior está apresentado no programa 4.1.

4.5 Um modelo físico de linha de transmissão acústica

Os arquivos da versão 3f4 do SPICE de Berkeley estão organizados segundo a estrutura de diretórios apresentada na figura 4.8 e podem ser descompactados através da linha de comando: `tar -zxvf spice3f4.tar.gz`.

Como não dispomos de nenhuma literatura que explicite todos os passos detalhadamente para que um novo dispositivo seja incluído no código fonte da versão 3f4 (linguagem C) [14] [15], tivemos que estabelecer um método para determinarmos

Programa 4.1 Subcircuito SPICE

```
.SUBCKT SAW 1 2 3 4 5 6
CT 1 2 3.267P
RPRI 1 11 0.0001
LPRI 11 2 1
RSEC 21 22 0.0001
LSEC 22 24 47U
KTRANS LPRI LSEC 1

R100 21 210 0.1
C1 210 31 100P
L1 31 26 100N

R200 24 240 0.1
C2 240 32 100P
L2 32 25 100N

R300 25 250 0.2
C3 250 33 50P
L3 33 34 200N
V3 34 26 0
FNEG 26 25 V3 2

R4 27 0 0.01
R5 28 0 0.01

ODENTES 25 27 26 28 DENTES
TLATERAL1 25 27 3 4 Z0=1 TD=1U
TLATERAL2 26 28 5 6 Z0=1 TD=1U
R8 4 0 100MEG
R9 6 0 100MEG
.MODEL DENTES LTRA LEN=63.245 R=0 L=100P G=0 C=100P
.ENDS
```

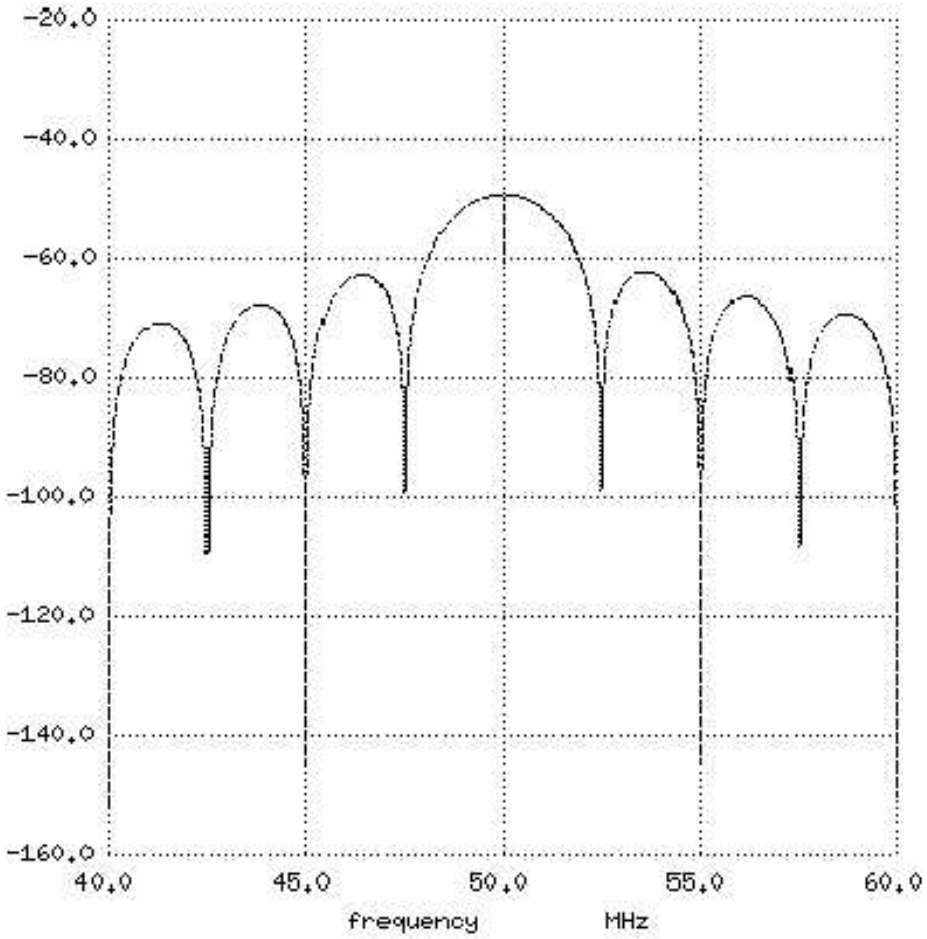

Figura 4.7: Resposta em freqüência do circuito ressonador

quais arquivos e o que alterar para que um novo modelo fosse disponibilizado pelo simulador.

A idéia é criar um novo componente sendo a cópia de um já existente. Para isso, utilizamos a linha de transmissão com perdas ("O") e seu modelo associado ("LTRA"), por se tratar de um modelo de duas entradas e ter como função o transporte de uma onda, que no caso é a corrente elétrica. Características que podem ter analogia com um meio de transmissão acústico ou até mesmo com um dispositivo OAS. Caso a duplicação obtenha sucesso fica esclarecido o processo de inclusão de um novo dispositivo, necessitando apenas a modificação das funções pertinentes ao modelo para que um, com novas características, seja criado.

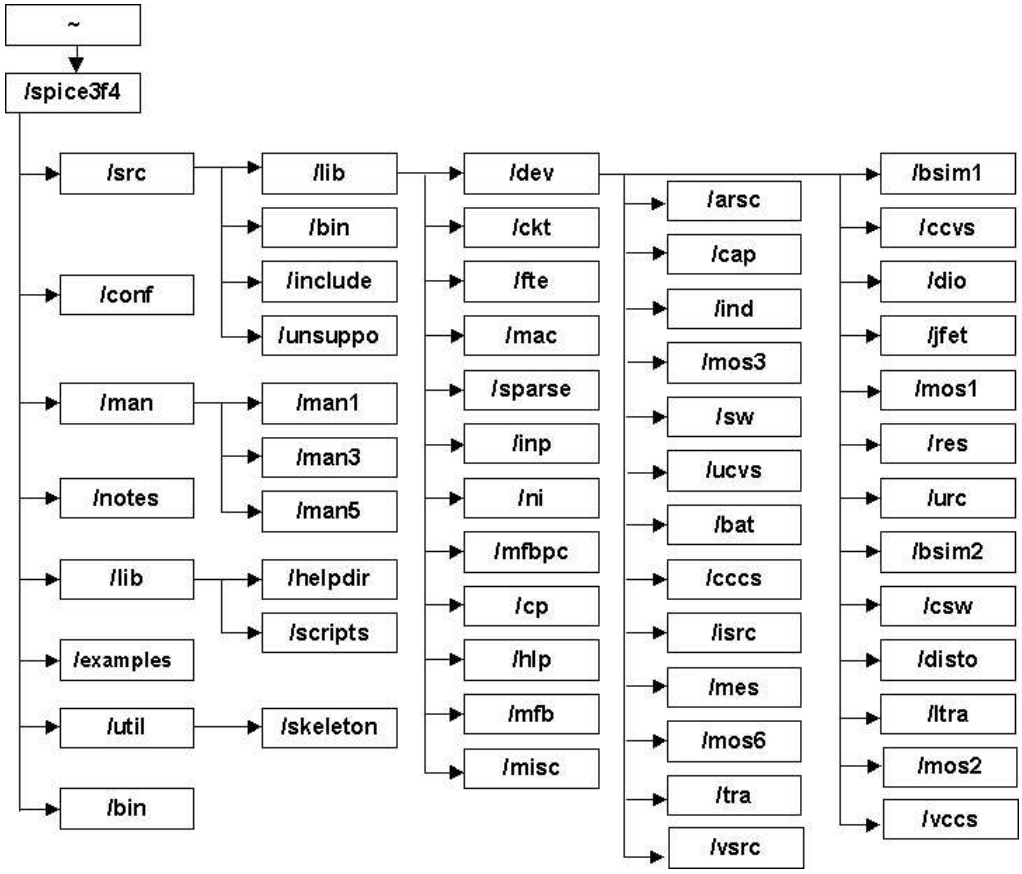

Figura 4.8: Estrutura de diretórios da versão 3f4

4.5.1 Gerando o programa executável

Antes de qualquer coisa, é preciso saber gerar o programa executável (spice3*) a partir do código fonte. Para isso é bastante utilizar o comando `./util/build linux` dentro do diretório `/spice3f4`. O programa executável é criado automaticamente e gravado no diretório `/spice3f4/src/bin`.

4.5.2 Criando uma cópia de um modelo existente

Primeiramente, é necessário verificar quais arquivos têm ligação com a linha de transmissão com perdas. Para isso, palavras como: trasmission line, LTRA, O, foram pesquisadas em todo o código fonte. Uma vez definidos os arquivos, temos condições de estudar o processo de definições de funções e passagem de parâmetros. Como a letra **A** não referencia nenhum componente, foi escolhida para o nosso novo modelo, que chamamos de **SAW**. Assim, teremos um novo componente (A) com seu modelo associado (SAW). Este novo componente será a cópia da nossa linha de transmissão

com perdas.

O diretório `/spice3f4/src/lib/dev/saw` foi adicionado à estrutura inicial. Nele, foram copiados os arquivos pertencentes ao diretório da linha de transmissão com perdas (`/spice3f4/src/lib/dev/ltra`) , tendo sido as letras **ltra** dos seus nomes e das linhas de código substituídas por **saw**, ficando assim com os seguintes arquivos:

```
makedefs
saw.c
sawacct.c
sawacld.c
sawask.c
sawdefs.h
sawdel.c
sawdest.c
sawext.h
sawitf.h
sawload.c
sawmask.c
sawmdel.c
sawmisc.c
sawmpar.c
sawpar.c
sawset.c
sawtemp.c
sawtrun.c
```

Cada um destes arquivos estão listados no apêndice C, com as respectivas alterações explicitadas.

Foi também acrescentado o arquivo `inp2a.c` no diretório:

```
~/spice3f4/src/lib/inp
```

Este arquivo contém a estrutura de entrada para o componente **A**. Também os arquivos relacionados abaixo sofreram alterações e estão listados no apêndice B.

```
~/spice3f4/conf/defaults
~/spice3f4/util/skeleton/make-def.bd
~/spice3f4/src/bin/bconf.c
~/spice3f4/src/bin/cconf.c
~/spice3f4/src/bin/config.c
~/spice3f4/src/unclude/inpdefs.h
~/spice3f4/src/lib/ckt/dctran.c
~/spice3f4/src/lib/ckt/pzan.c
~/spice3f4/src/lib/fte/subckt.c
~/spice3f4/src/lib/inp/inpdomod.c
~/spice3f4/src/lib/inp/inppas2.c
~/spice3f4/src/lib/inp/makedefs
~/spice3f4/src/lib/inp/sperror.c
```

Uma vez as modificações realizadas é preciso verificar se o novo componente está corretamente adicionado. Para isto foi feita uma simulação de um circuito simples contendo uma linha de transmissão com perdas, figura 4.9.

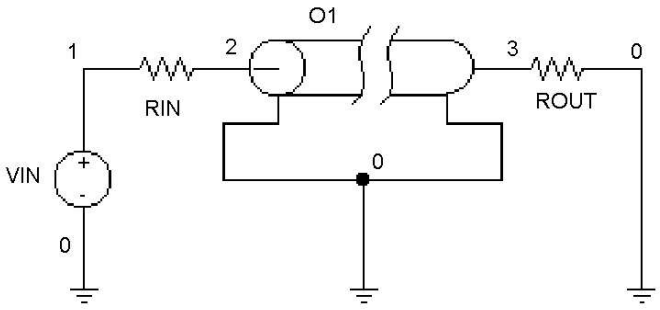

Figura 4.9: Circuito contendo o modelo LTRA

Outra simulação foi realizada utilizando o componente **A** e o modelo **SAW**, figura 4.10, a propósito de comparação.

Os resultados foram idênticos, figura 4.11, o que comprova a correta sistemática de inclusão de um novo modelo físico.

Programa 4.2 Arquivo de descrição do circuito - modelo LTRA

```
*DESCRICAO DO CIRCUITO

VIN 1 0 PULSE(0 4 30NS 0 0 40NS 0)
RIN 1 2 100
O1 2 0 3 0 LINHA
ROUT 3 0 100

.MODEL LINHA LTRA LEN=1M R=100 L=37M G=0 C=37N

.TRAN 0.1NS 150NS

.END
```

Programa 4.3 Arquivo de descrição do circuito - modelo SAW

```
*DESCRICAO DO CIRCUITO

VIN 1 0 PULSE(0 4 30NS 0 0 40NS 0)
RIN 1 2 100
A1 2 0 3 0 LINHA
ROUT 3 4 100

.MODEL LINHA SAW LEN=1M R=100 L=37M G=0 C=37N

.TRAN 0.1NS 150NS

.END
```

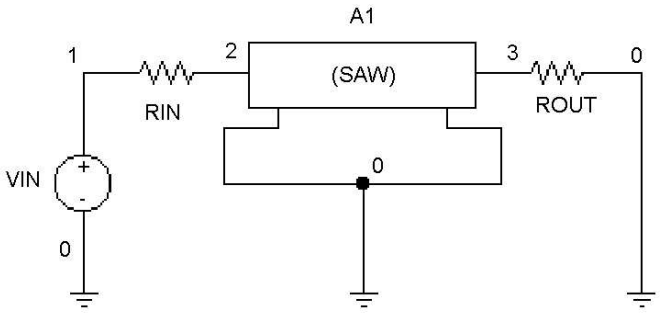

Figura 4.10: Circuito contendo o modelo SAW

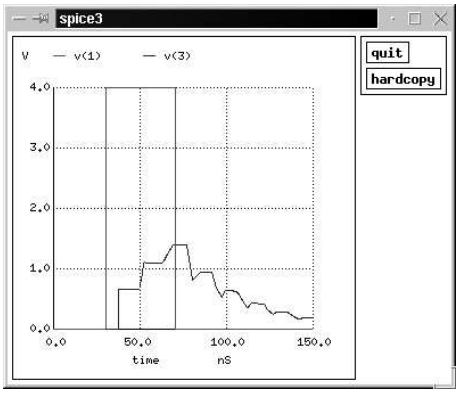

Figura 4.11: Resultado das simulações

4.5.3 Inclusão de um modelo de linha de transmissão acústica

Como obtivemos sucesso no processo de duplicação, passamos para a compreensão de como especificar novos parâmetros para o modelo e como fazer uso deles nas equações. Se tivermos como parâmetros para a linha de transmissão (A) a velocidade de fase e o comprimento, considerando a impedância característica igual a 1 ($Z0 = 1$), teremos um modelo de linha de transmissão acústica. Para isso, foi definido um novo parâmetro para o modelo, que representa a velocidade de fase, no caso o parâmetro "VEL", acarretando mudanças nos arquivos:

```
saw.c
sawdefs.h
sawmask.c
sawmpar.c
sawset.c
```

No arquivo `sawset.c` foram definidos os valores de L e C, que são iguais, em função da impedância característica unitária, a partir do valor informado da velocidade, através das equações para uma linha de transmissão do tipo RLC, considerando $G = 0$ e $R \ll \omega L$:

$$Z_0 = \sqrt{\frac{L}{C}}$$

$$V_P = \frac{1}{L}$$

onde L e C são a indutância e a capacitância por unidade de comprimento, respectivamente, Z_0 a impedância característica da linha e V_P a velocidade de fase. As alterações no código fonte estão explicitadas no apêndice C.

Para que tenhamos a certeza do correto funcionamento do modelo da linha acústica simulamos os mesmos circuitos apresentados nas figuras 4.9 e 4.10, apenas ajustando os valores da indutância e capacitância para que a linha tenha impedância característica unitária ($L = C$), como pode ser visto na descrição do circuito.

Programa 4.4 Arquivo de descrição do circuito - linha LTRA

```
*DESCRICAO DO CIRCUITO

VIN 1 0 PULSE(0 4 40NS 0 0 40NS 0)
RIN 1 2 0.001
RS 2 0 1
O1 2 0 3 0 LINHA
ROUT 3 0 1

.MODEL LINHA ltra LEN=60U R=100 L=333.33333U C=333.33333U

.TRAN 0.01NS 200NS

.END
```

A simulação usando o dispositivo (A) foi efetuada com o valor do parâmetro "VEL" igual ao inverso do valor da indutância da simulação anterior. De forma que a linha acústica tenha a mesma velocidade de propagação, cerca de 3000m/s, como pode ser visto na descrição do circuito.

Os resultados das simulações são idênticos, figura 4.12, como era de se esperar, comprovando a correta funcionalidade do modelo. A criação desta linha acústica

Programa 4.5 Arquivo de descrição do circuito - linha acústica (A)

```
*DESCRICAO DO CIRCUITO

VIN 1 0 PULSE(0 4 40NS 0 0 40NS 0)
RIN 1 2 0.001
RS 2 0 1
A1 2 0 3 0 LINHA
ROUT 3 0 1

.MODEL LINHA SAW LEN=60U R=100 vel=3000

.TRAN 0.01NS 200NS

.END
```

ajuda a entender melhor o mecanismo de inclusão de um modelo físico no SPICE, servindo de fonte de consulta para um desenvolvimento de um dispositivo mais complexo.

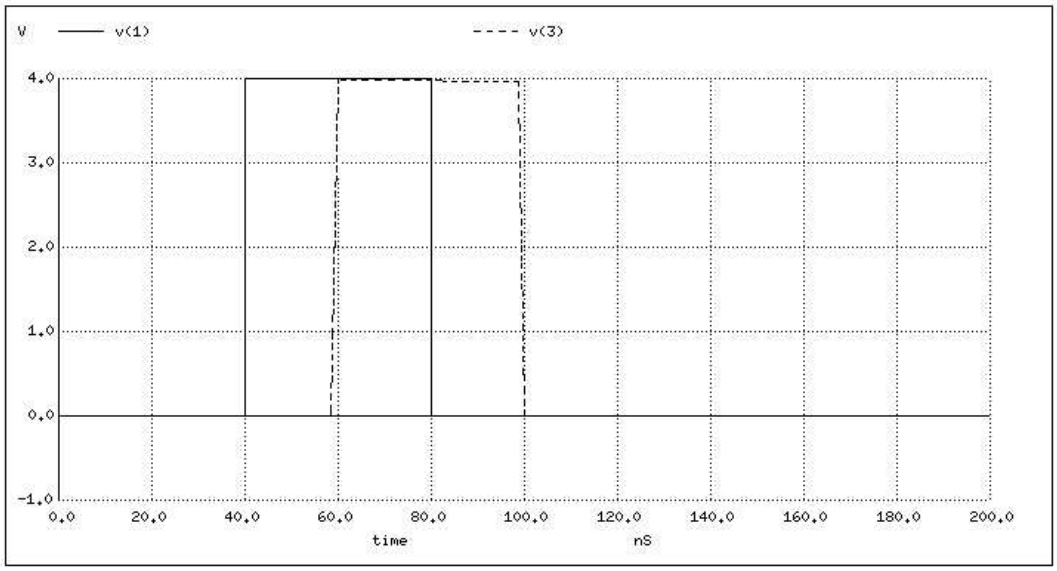

Figura 4.12: Resultado das simulações (Linha acústica e LTRA)

4.6 Conclusão

O emprego das equações dos modos acoplados possibilita o modelamento do comportamento de um transdutor OAS por inteiro e não apenas de um par de eletrodos.

Nakamura, em seu artigo [16], propõe uma solução para estas equações que representam o comportamento de um transdutor OAS, onde uma matriz admitância de um sistema de duas portas acústicas e uma elétrica, foi desenvolvida. O desafio, então, é a representação desta matriz em termo de um subcircuito, capaz de servir como um macromodelo implementável em qualquer versão SPICE, desde a gratuita, como o SPICE3f4 de Berkeley, até o ELDO® da Mentor Graphics.

O circuito foi obtido com a representação da parte acústica da matriz através de uma linha de transmissão. Desta maneira, o comprimento do transdutor é agora um parâmetro da linha, não mas precisando de vários circuitos em série para representar transdutores longos. Isto torna o modelo simples, compacto e de fácil implementação. Uma aplicação do modelo como sensor de espessura de filme fino, apresentada no próximo capítulo, mostra isso e também valida o modelo, já que resultados experimentais [22] são comparados com os obtidos nas simulações.

Neste capítulo, a inclusão de um modelo de linha de transmissão acústica também é demonstrada. Para isso, o modelo (LTRA) da linha de transmissão com perdas "O"foi escolhido para ser copiado para um novo modelo (SAW) associado a um novo componente "A"(linha de transmissão acústica). No caso, o modelo "LTRA"foi duplicado e representado como "SAW", tendo o componente "A"as mesmas características do componente "O". Esta foi uma forma de aprender sobre a estrutura do código fonte e de como um novo componente e modelo pode ser implementado. Para ter certeza do entendimento do processo, foram feitas simulações com ambos os componentes e comparados os resultados.

A duplicação do modelo da linha de transmissão com perdas esclareceu o processo de inclusão de um novo componente no simulador SPICE3f4. O passo agora foi alterar a estrutura de entrada dos parâmetros da linha, de forma que, a capacitância (C) e a indutância (L) da linha foram substituidas pela velocidade de propagação da onda acústica (VEL). Com isto, os parâmetros do novo modelo (SAW) para uma linha acústica (A) são agora o comprimento (LEN), a resistência (R) e a velocidade de propagação (VEL). Isto tudo foi obtido com alterações do código fonte em linguagem C. O emprego deste novo modelo torna mais direto o modelamento da superfície de propagação da onda acústica entre dois transdutores OAS.

Capítulo 5

Sensores OAS

Algumas aplicações de OAS como sensores serão apresentadas e também validado o nosso modelo SPICE, como ferramenta de simulação de um sensor de espessura de filmes finos, comparando os resultados com os obtidos por Hank Wohltjen em seu artigo [22].

5.1 Aplicações como Sensores

Sensores que utilizam o OAS como linha de atraso são facilmente construídos para monitoramento de mudanças de amplitude ou velocidade da onda acústica. Para a obtenção da variação da amplitude é utilizada uma fonte de rádio freqüência para excitar a onda que irá propagar-se no OAS e medida sua potência na extremidade final. Estes sistemas podem utilizar uma linha como referência para se obter uma maior precisão da medida desta variação, como utilizado pelo sensor de vapor químico ilustrado na figura 5.4. Medições de mudanças de velocidade são obtidas com grande precisão utilizando a linha de atraso como elemento de ressonância. Desta forma, um amplificador de RF é utilizado na realimentação do circuito, ou seja, a energia da saída é amplificada e realimenta o OAS. Desde que o seu ganho seja superior às perdas da linha, o circuito oscila na freqüência de ressonância do transdutor. Esta freqüência de ressonância é alterada pela mudança de velocidade da onda de Rayleigh. Esta técnica oferece uma maior precisão da medida comparada com a obtida pela variação da amplitude.

O número de fenômenos que podem ser monitorados pelo OAS pode ser ampliado

através do uso de materiais que alteram sua massa, elasticidade ou condutividade quando expostos a algum estímulo físico ou químico e que podem ser dispostos entre os transdutores do OAS, recobrindo a superfície de propagação da onda.

O OAS torna-se um sensor de pressão, torque ou força quando uma tensão mecânica é aplicada, mudando a dinâmica de propagação do meio. Torna-se um sensor de massa quando partículas são agregadas ao meio de propagação, alterando a tensão aplicada. Escolhendo um filme que absorva um específico vapor químico e recubra o meio de propagação, um sensor químico é construído, ou até mesmo um sensor biológico, caso o filme seja, por exemplo, sensível a um elemento biológico presente em líquidos.

Iremos entrar em alguns detalhes de cada um destes sensores.

5.1.1 Sensor de Temperatura

A temperatura afeta a velocidade de propagação da onda, e esta dependência pode ser maximizada pela orientação e tipo do material cristalino utilizado na fabricação do OAS. Sensores típicos de temperatura, baseados em osciladores construídos com um OAS como linha de atraso, possuem resolução de miligraus e boa linearidade [23]. Estes sensores são hermeticamente encapsulados.

5.1.2 Sensor de Pressão

O uso de um OAS como sensor de temperatura teve sua primeira publicação em 1975 [24]. A velocidade da onda é extremamente afetada pela pressão aplicada ao substrato piezoelétrico no qual está se propagando. Este sensor é construído formando um diafragma com o OAS, conforme podemos ver na figura 5.1.

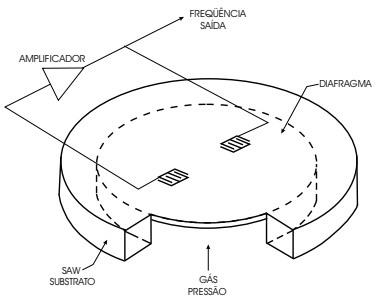

Figura 5.1: Sensor de pressão não compensado

Historicamente, estes sensores de pressão são construídos sem compensação da variação da temperatura. Contudo, um dispositivo OAS pode ser adicionado no mesmo substrato próximo a região do diafragma, como referência, de modo que os dois sinais possam ser misturados e a influência da temperatura minimizada. O sensor de temperatura que serve de referência não pode ser exposto à pressão do diafragma. Trata-se, portanto, de um sensor dual e pode ser visto na figura 5.2.

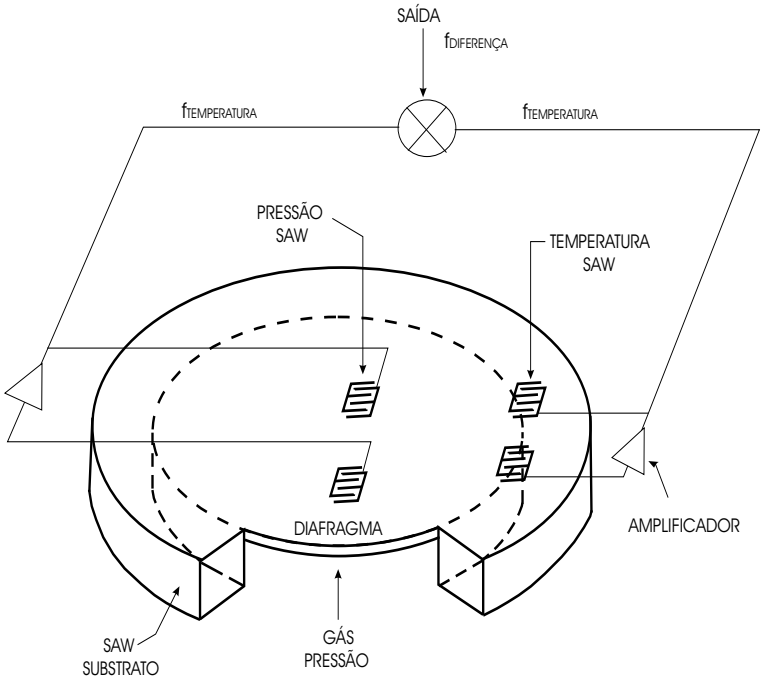

Figura 5.2: Sensor de pressão compensado

Os sensores de pressão, assim como o de temperatura, não requerem fonte de entrada para excitação da onda, o que os torna interessantes em aplicações sem fio.

5.1.3 Sensor de Torque

Uma vez um dispositivo OAS firmemente montado em um eixo liso, e este eixo seja submetido a um torque, esta mesma força irá atuar no OAS, tornando-se um efetivamente em um sensor passivo de torque. Caso este eixo gire em apenas uma direção, o OAS é disposto de forma tensionada, caso não, outro OAS deve ser disposto de forma comprimida, formando um ângulo reto entre as suas linhas centrais, como ilustrado na figura 5.3. Como os dois sensores são expostos à mesma temperatura, a soma destes dois sinais minimiza o efeito da temperatura.

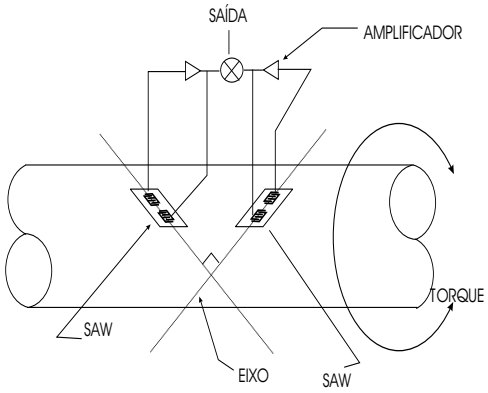

Figura 5.3: Sensor de torque compensado

Comparando este sensor com sensores do tipo *strain gauges*, transdutores ópticos e barras de torção, ele oferece menor custo, fácil construção e operação remota (sem fio).

5.1.4 Sensor de Massa

Sensores OAS são excelentes sensores de massa, sendo o mais sensível de todos os sensores mencionados. Isso abre uma gama de aplicações, que incluem sensores de partículas e de espessura de filmes.

Se o sensor for coberto com uma substância adesiva, qualquer partícula que caia sobre a superfície irá ficar retida e irá pertubar a propagação da onda, tornado-o um sensor de partículas. Um sensor com resolução de 3 picogramas para um OAS a 200MHz construído em Quartzo com corte ST foi apresentado [25]. Este tipo de sensor é utilizado em salas limpas e monitoramento atmosférico e da qualidade do ar.

Sensores de espessura, fundamentalmente, trabalham da mesma maneira, exceto porque a medida da variação da freqüência é proporcional à massa do filme depositado entre os transdutores, onde a densidade do filme e a impedância acústica informam a espessura do filme, que deve ser um pequeno percentual do comprimento da onda acústica.

5.1.5 Sensor de Vapor Químico

Estes sensores tiveram suas primeiras publicações em 1979 [26]. A maioria baseia-se na medição da massa através de um filme que absorve o vapor de interesse e que é

disposto na região de propagação da onda, onde a absorção do gás pelo filme implica num aumento de massa. Para compensar a variação da temperatura um outro OAS é utilizado como referência. Esta montagem pode ser vista na figura 5.4.

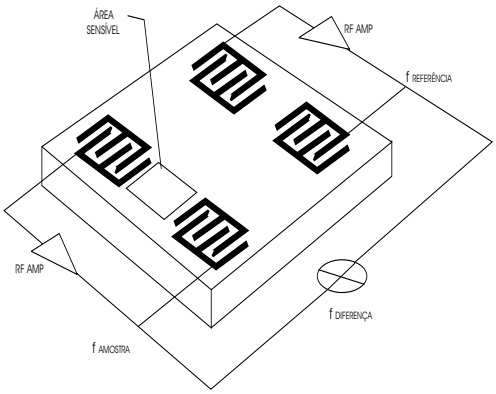

Figura 5.4: Sensor de vapor químico

Há algumas considerações que devem ser feitas na escolha do filme para a absorção deste vapor químico. Idealmente esta película deve ser completamente reutilizável, ou seja, absorver o gás de interesse e poder ser limpo com ar para estar disponível para a reabsorção. Este período no qual o filme absorve e desabsorve este gás deve ser inferior a 1 segundo, ao mesmo tempo deve ser robusto para não se deteriorar na presença de gases químicos e com grande seletividade para que outros gases, que não o de interesse, sejam absorvidos também.

Detecção de gases químicos e sua identificação é possível com o uso de uma matriz de sensores OAS, cada um com seletividade para um gás de interesse, e todos expostos a uma mesma amostra. Desta forma, um software de reconhecimento de padrões pode ser utilizado para identificar os diversos compostos voláteis presentes.

5.1.6 Biosensor

Similar ao sensor de vapores químicos, os biosensores, porém, detectam compostos químicos em líquidos e não em forma gasosa. Para este tipo de aplicação o OAS não é a melhor forma para a sua constituição, já que se trata de uma propagação de componente vertical que sofrerá atenuação pelo líquido. Estes sensores são geralmente fabricados utilizando ressoadores do tipo TSM, SH-APM e sensores SH-SAW, já mencionados anteriormente. De todos os sensores acústicos para líquidos, os sensores de Ondas de Love possuem a melhor sensibilidade [27].

5.2 Sensibilidade

A sensibilidade do sensor a uma propriedade, X_{prop}, é definida como sendo a derivada da freqüência em função da propriedade de interesse.

$$S_{prop} = \frac{\partial f}{\partial X_{prop}} \tag{5.1}$$

Para um sensor de massa OAS, a equação aproximada que descreve a variação de freqüência é dada por:

$$\Delta f = -1,26 \times 10^{-7} f_0^2 \frac{m}{A}$$

onde f_0 é a freqüência central e $(\frac{m}{A})$ a massa depositada sobre a área sensível. Neste caso, considerando a equação 5.1, a sensibilidade para este sensor é a seguinte:

$$S_m = \frac{\partial f}{\partial m} = -1,26 \times 10^{-7} \frac{f_0^2}{A}$$

A variação de massa mínima detectável dependerá da qualidade do amplificador em termos de estabilidade e de ruído.

$$\Delta m_{min} = \frac{1}{S_m} \Delta f_{min}$$

5.3 Simulação de um Sensor de Espessura de Filme Fino

O sensor de espessura de filme fino possui grande sensibilidade, já que fazemos uso de um OAS como linha de atraso, servindo como elemento ressonante em um circuito oscilador. Mudanças de velocidade de propagação do OAS são proporcionais à espessura do filme depositado entre os transdutores. Isto porque a freqüência de oscilação é dada por:

$$\omega = (2n\pi - \phi_e) V_R / L \tag{5.2}$$

ϕ_e é o deslocamento de fase causado pelo circuito, V_R é a velocidade de propagação da onda no OAS, L o comprimento da linha de atraso entre os transdutores e n um número inteiro. Aparentemente, várias freqüências de ressonância são possíveis dependendo do valor de n, porém na prática n é restrito devido à largura da banda

imposta pelos eletrodos interdigitados. Como ϕ_e e L são constantes significa dizer que variações da freqüência de oscilação irão ocorrer devido a pertubações da velocidade, a qual está intimamente ligada à sua interação com o filme depositado na superfície do dispositivo, como explicitamos no caso do sensor de massa.

A intenção é mostrar a funcionalidade do nosso modelo proposto para um OAS. Para isto, iremos simular um sensor de espessura de filme fino, no caso o Polimetilmetacrilato, que teve sua implemantação fisica e seus resultados experimentais publicados em artigo [22] e comparar com os obtidos em nossa simulação.

5.3.1 Montagem Física

O dispositivo foi fabricado em substrato ST-Quartzo com meia polegada de largura e duas de comprimento, tendo um lado polido. O transdutor interdigital consiste de 50 pares de eletrodos de 2000Å de ouro em 200Å de cromo. Cada eletrodo possui 25 micrometros de largura e espassados de igual valor e sobreposição(W) de 7250 micrometros. A estrutura interdigitada foi construída utilizando técnica de microfabricação com litografia óptica e metalização por evaporação. O espaçamento entre os centros dos transdutores é de $2.0cm$ e silicone foi aplicado em cada extremidade do dispositivo, de forma a funcionar como absorvedores acústicos com o propósito de eliminar reflexões. O dispositivo então foi concebido para operar em 31MHz e a sua montagem pode ser vista na figura 5.5.

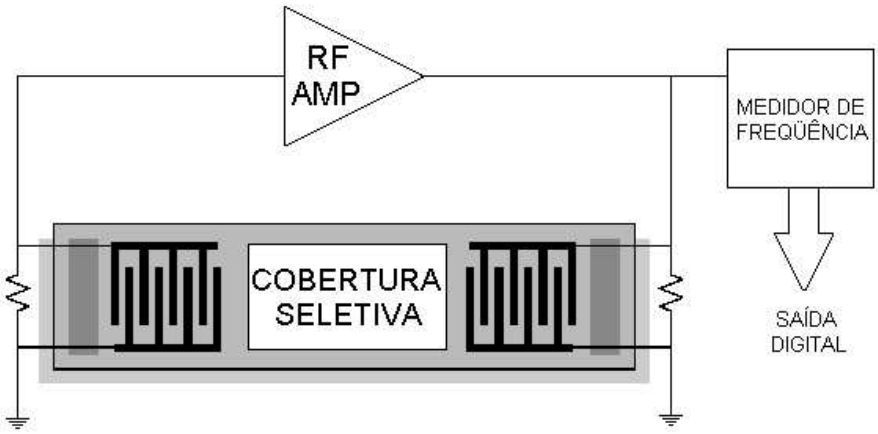

Figura 5.5: Montagem do Sensor Espessura de Filme Fino

Camadas de filme de Polimetilmetacrilato foram depositadas sobre a superfície

do dispositivo que teve sua espessura determinada por microscópio. Cada deposição foi feita sobre condições idênticas, tendo, para cada espessura, uma medição da freqüência efetuada. Para estas medidas foi utilizado um medidor de freqüência e um simples amplificador de RF foi implementado para ampliação do sinal e como buffer do medidor. O circuito do amplificador pode ser visto na figura 5.6.

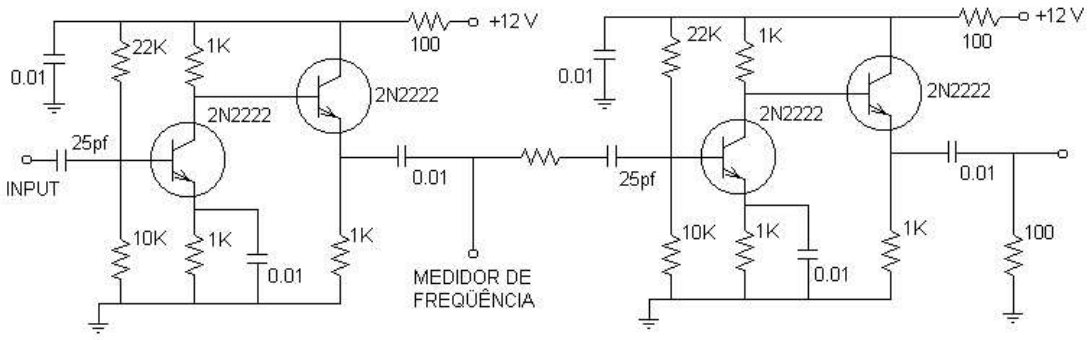

Figura 5.6: Amplificador de RF

5.3.2 Circuito para Simulação

Iremos agora representar o circuito físico através de blocos de subcircuitos capazes de serem implementados no SPICE. Desta forma, a linha de atraso composta do OAS e que representa o elemento de ressonância do circuito terá seu comportamento simulado pelo nosso modelo proposto para um OAS. O amplificador terá o seu circuito diretamente representado, pois como vimos, o SPICE é um simulador de propósito geral capaz de implementar modelos de transistores, capacitores, fontes e resistores. A deposição do filme será simulada através de uma linha de atraso (TFILME) colocada entre os transdutores de forma que a variação do tempo de atraso represente a variação da velocidade de propagação decorrente da espessura do filme. Desta maneira, teremos uma resposta da variação da freqüência de ressonância decorrente da variação do tempo de atraso da linha TFILME, com a qual podemos estabelecer uma proporção com a espessura do filme e levantarmos a curva da variação da freqüência versus a variação da espessura do filme. Esta curva pode ser comparada com a obtida de forma experimental por Wohltjen. Caso essa verificação mostre um comportamento semelhante entre as curvas podemos considerar o nosso modelo como viável para utilização em dispositivos OAS como sensores de massa ou espessura.

O macromodelo do transdutor OAS foi implementado no simulador ELDO® da Mentor Graphics de estrutura SPICE e desenvolvido para ambiente UNIX, a única mudança em relação ao arquivo de descrição apresentado em capítulos anteriores é a forma de representar as linhas de transmissão, como pode ser visto no programa 7.1. Isso comprova que o modelo proposto é versatil e de fácil implementação em simuladores de estrutura SPICE. Neste modelo foi implementada uma linha de transmissão com perdas para modelar a porção acústica do transdutor.

Programa 5.1 Macromodelo SPICE do Transdutor - Arquivo:SUBSAWP31

```
* SUBCIRCUITO DE UM TRANSDUTOR OAS DE 31MHz

.SUBCKT SAW 1 3 5
CT 1 0 19.729968P
RPRI 1 11 0.0001
LPRI 11 0 1
RSEC 21 22 0.0001
LSEC 22 24 479.373U
KTRANS LPRI LSEC 1

C1 21 31 65.36850558P
L1 31 25 403.2258065N
R1 21 25 100MEGA

C2 24 32 65.3685P
L2 32 26 403.2258065N
R2 24 26 100MEGA

C3 26 33 32.68425279P
L3 33 34 806.4516129n
R3 26 34 100MEGA
V3 34 25 0
FNEG 25 26 V3 2

.MODEL LDTL MACRO LANG=C
YDENTES LTDL 25 26 0 PARAM: LEVEL=3 LENGTH=5.0935485
+R=0 L=316.65611U C=316.65611U
TLATERAL1 5 0 25 0 Z0=1 TD=1.5u
* ESTA LINHA DE GRANDE COMPRIMENTO SIMULA O ABSORVEDOR ACUSTICO
TLATERAL2 26 0 3 0 Z0=1 TD=300u

.ENDS
```

Inicialmente fizemos a simulação utilizando um amplificador ideal para que a forma

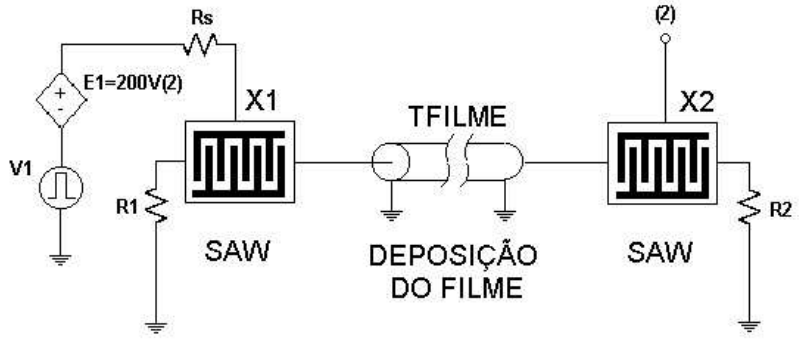

Figura 5.7: Diagrama do circuito oscilador - Amp.ideal

de onda não sofresse distorções oriundas da implementação do amplificador real e pudéssemos obter uma boa linearidade da curva da variação da freqüência, de forma que a descrição do circuito está apresentada no programa 7.2 e o diagrama do circuito pode ser visto na figura 5.7

Programa 5.2 Descrição SPICE do circuito com amp.ideal

```
* CIRCUITO DO SENSOR OAS - OSCILADOR

E1 7 8 2 0 200
V1 8 0 AC 1 PULSE(0 1 5n 1n 1n 10n)

RS 7 1 50
X1 1 3 4 SAW
X2 2 6 5 SAW
R1 3 0 1
R2 6 0 1
TLINHA 4 0 5 0 Z0=1 TD=0.011179297U
.TRAN 0.1N 8U
.DC
.PLOT TRAN V(2)
.INCLUDE SUBSAWP31
.END
```

Como uma condição para que o circuito oscile é que o ganho do amplificador seja maior que as perdas impostas pela linha, foi determinado um fator de ganho de 200, o que podemos visualizar na figura 5.8, do ganho AC em malha aberta, que esta necessidade foi atendida.

No caso do macromodelo do amplificador de RF foram utilizados os mesmos tran-

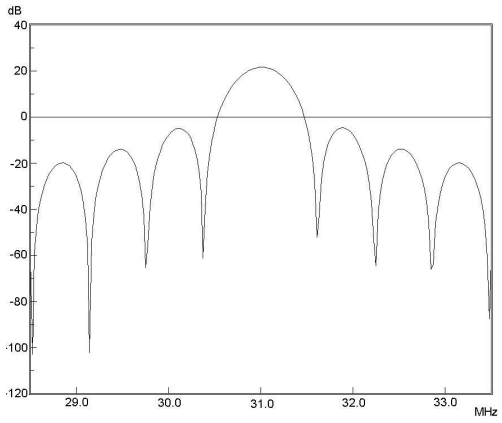

Figura 5.8: Resposta em freqüência de malha aberta (Amp.Ideal)

sistores do circuito original e a sua listagem também foi implementada no ELDO® . Desta maneira o circuito agora simulado é o mesmo que o implementado experimentalmente e está descrito no programa 7.3.

Diferente da simulação anterior uma linha de transmissão sem perdas foi implementada no macromodelo para o modelamento da porção acústica do transdutor, de forma que o seu arquivo de descrição sofreu pequenas alterações que podem ser vistas no programa 7.4.

A descrição do circuito oscilador que implementa o OAS, a área de deposição do filme representada pela linha TFILME e o amplificador de RF está apresentada no programa 7.5 e o diagrama do circuito pode ser visto na figura 5.9. A fonte de pulso presente no circuito fornece a energia inicial necessária para o funcionamento do oscilador.

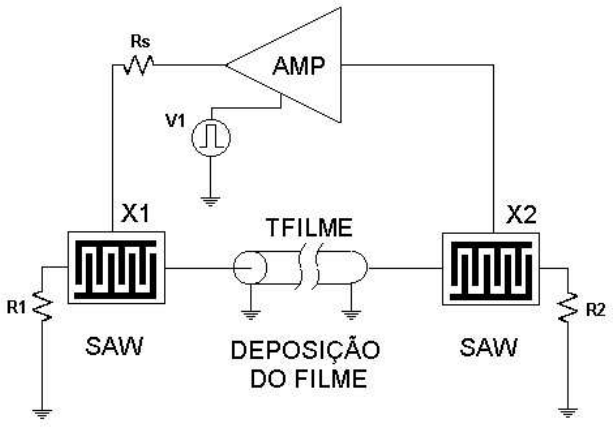

Figura 5.9: Diagrama do circuito - Amp. real

Programa 5.3 Macromodelo SPICE do Amp.de RF - Arquivo:AMPRF2

```
* SUBCIRCUITO DE UM AMPLIFICADOR DE RF

.SUBCKT AMPRF 1 2 500

C1 1 3 25P
R1 3 6 22K
R2 3 500 10K
C2 6 500 0.01
Q1 4 3 5 Q2N2222
.MODEL Q2N2222 NPN (IS=15.01F XTI=3 EG=1.11 VAF=90.7 BF=223.7 NE=2.348
+ ISE=70.78P IKF=3.837 XTB=1.5 BR=1 NC=2 ISC=0 IKR=0 RC=0 CJC=2P
+ MJC=0.3333 VJC=0.75 FC=0.5 CJE=5P MJE=0.3333 VJE=0.75 TR=10N TF=1N
+ ITF=0 VTF=0 XTF=0)
R3 6 4 1K
R4 5 500 1K
C3 5 500 0.01
Q2 6 4 7 Q2N2222
R5 15 6 100
V1 15 500 DC 12
R6 7 500 1K
C4 7 8 0.01

R70 8 9 0.001

C5 9 10 25P
R7 10 500 10K
R8 13 10 22K
C6 13 500 0.01
Q3 11 10 12 Q2N2222
R9 13 11 1K
R10 12 500 1K
C7 12 500 0.01
Q4 13 11 14 Q2N2222
R11 14 500 1K
R12 13 15 100
C8 14 2 0.01
R13 2 500 100

.ENDS
```

Programa 5.4 Macromodelo SPICE do Transdutor - Arquivo:SUBSAWP31S

```
* SUBCIRCUITO DE UM TRANSDUTOR OAS DE 31MHz

.SUBCKT SAW 1 3 5
CT 1 0 19.729968P
RPRI 1 11 0.0001
LPRI 11 0 1
RSEC 21 22 0.0001
LSEC 22 24 479.373U
KTRANS LPRI LSEC 1

C1 21 31 65.36850558P
L1 31 25 403.2258065N
R1 21 25 100MEGA

C2 24 32 65.3685P
L2 32 26 403.2258065N
R2 24 26 100MEGA

C3 26 33 32.68425279P
L3 33 34 806.4516129n
R3 26 34 100MEGA
V3 34 25 0
FNEG 25 26 V3 2

TDENTES 25 0 26 0 Z0=0 TD=1.6129032u
TLATERAL1 5 0 25 0 Z0=1 TD=1.5u
* ESTA LINHA DE GRANDE COMPRIMENTO SIMULA O ABSORVEDOR ACUSTICO
TLATERAL2 26 0 3 0 Z0=1 TD=300u

.ENDS
```

Programa 5.5 Descrição SPICE do circuito com amp.real

```
* CIRCUITO TESTE DO SENSOR OAS - OSCILADOR

V1 500 0 AC 1 PULSE(0 1m 5n 1n 1n 10n)

X3 2 7 500 AMPRF
RS 7 1 50
X1 1 3 4 SAW
X2 2 6 5 SAW
R1 3 0 1
R2 6 0 1
TLINHA 4 0 5 0 Z0=1 TD=0.0277U
.TRAN 3.2n 20u 0u 3.2n
.DC
.PLOT TRAN V(2)
.INCLUDE SUBSAWP31S
.INCLUDE AMPRF2
.END
```

Foi analisada a resposta em freqüência de malha aberta do circuito para verificar se o ganho do amplificador é maior que as perdas impostas pela linha, de forma que o circuito possa oscilar, o que podemos verificar pela figura 5.10.

5.3.3 Resultados Obtidos

Primeiramente, examinaremos o circuito com o amplificador ideal efetuando simulações e medindo as freqüências de ressonância para cada valor estipulado do tempo de atraso da linha TFILME. Os valores encontrados estão expostos na tabela 5.1. Os valores da variação do tempo de atraso (TD) foram multiplicados por um fator de proporcionalidade 165 de forma que representasse a espessura do filme e tivesse a sua curva ajustada para efeito de confronto. Desta forma, podemos levantar o comportamento da variação da freqüência de ressonância com a variação da espessura do filme e compará-la com os resultados obtidos experimentalmente.

Na figura 5.11 podemos confrontar o resultado gráfico das simulações e os valores obtidos experimentalmente. Os pontos levantados pela simulação apresentam uma baixa dispersão em relação a linha de tendência, o que comprova a boa funcionalidade do modelo para aplicações com sensores deste tipo [28] [29].

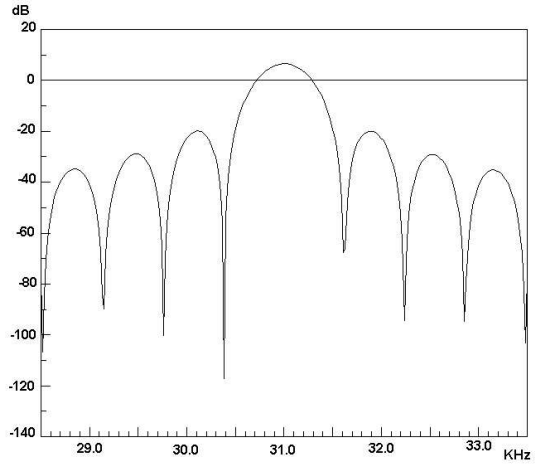

Figura 5.10: Resposta em freqüência de malha aberta (Amp.Real)

As mesmas simulações foram realizadas substituindo o amplificador ideal pelo utilizado na montagem experimental. Neste caso, o fator de proporcionalidade encontrado foi de 72, sendo elaborada a tabela de resultados 5.2.

Na figura 5.12 está o resultado gráfico das simulações. Neste caso, houve uma dispersão das variações obtidas em torno da reta de tendência, porém o comportamento da curva ainda permanece semelhante, comprovando a funcionalidade do modelo. A sensibilidade deste sensor, considerando a espessura como a propriedade de interesse, de acordo com a equação 5.1, é então, a inclinação da curva $\Delta f \times$ espessura do filme, que é da ordem de 100KHz por micron.

5.4 Conclusão

Sensores de temperatura, pressão, torque, massa, espessura, vapor químico e biosensores são alguns tipos de sensores que podem ser construídos com transdutores OAS, pois é grande o número de aplicações que podem ser desenvolvidas. Esta versatilidade, robustez, baixo custo e reduzidas dimensões motiva o seu emprego em sensores inteligentes. A possibilidade de efetuar uma simulação, de forma integrada, do sensor acústico com a eletrônica de tratamento e condicionamento do sinal, que foi oferecido pelo desenvolvimento do macromodelo SPICE constitui, então, uma ferramenta importante no desenvolvimento destes sensores. Este capítulo mostra exatamente o

TD (μ s)	Freq. (MHz)	Δ Freq. (KHz)	Δ TD (μ s)	Espessura (microns)
0,009582	30,885800	0,000000	0,000000	0,00
0,011179	30,864681	-21,119250	0,001597	0,26
0,012776	30,839873	-45,927200	0,003194	0,53
0,014373	30,811499	-74,300623	0,004791	0,79
0,015970	30,789262	-96,538200	0,006388	1,05
0,017567	30,765793	-120,00733	0,007985	1,32
0,019164	30,736711	-149,08910	0,009582	1,58
0,020761	30,698681	-187,11931	0,011179	1,84
0,021560	30,674342	-211,45820	0,011977	1,98
0,022358	30,647656	-238,14450	0,012776	2,11

Tabela 5.1: Tabela dos resultados das simulações - Amp.Ideal

emprego do modelo como um sensor de espessura de filme fino. Trata-se de uma simulação de uma montagem experimental que teve seus resultados publicados por Hank Wohltjen [22]. O amplificador utilizado na montagem experimental foi representado de duas maneiras. Em uma primeira representação do circuito para simulação, o amplificador foi considerado ideal e numa segunda, foram utilizados os modelos dos transistores da montagem de forma a reproduzi-lo com maior fidelidade. A idéia da primeira simulação é eliminar a influência do circuito do amplificador e mostrar a funcionalidade do modelo. Os resultados da simulação mostram uma variação da freqüência de ressonância com a espessura do filme com um comportamento bem semelhante aos obtidos experimentalmente. Já na segunda simulação, com o modelo do amplificador real, uma maior dispersão com os resultados experimentais foram obtidas para as medidas de menor espessura, o que comprova a preocupação com o modelamento do amplificador, mas que não invalida a sua implementação.

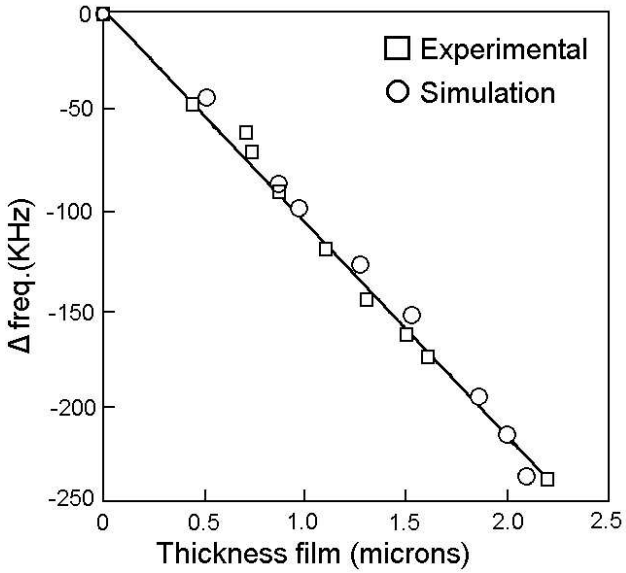

Figura 5.11: Gráfico dos resultados da simulação e experimental

TD (μ s)	Freq. (MHz)	Δ Freq. (KHz)	Δ TD (μ s)	Espessura (microns)
0,0283	30,970	0,0	0,0000	0,00
0,0291	30,945	-25,0	0,0008	0,05
0,0297	30,930	-40,0	0,0014	0,10
0,0306	30,923	-47,0	0,0023	0,16
0,0331	30,914	-56,0	0,0048	0,34
0,0356	30,900	-70,0	0,0073	0,52
0,0381	30,892	-78,0	0,0098	0,70
0,0406	30,884	-86,0	0,0123	0,88
0,0431	30,873	-97,0	0,0148	1,06
0,0456	30,852	-118,0	0,0173	1,24
0,0481	30,835	-135,0	0,0198	1,42
0,0506	30,814	-156,0	0,0223	1,60
0,0530	30,800	-170,0	0,0247	1,77
0,0545	30,790	-180,0	0,0262	1,88
0,0556	30,784	-186,0	0,0273	1,96

Tabela 5.2: Tabela dos resultados das simulações - Amp.Real

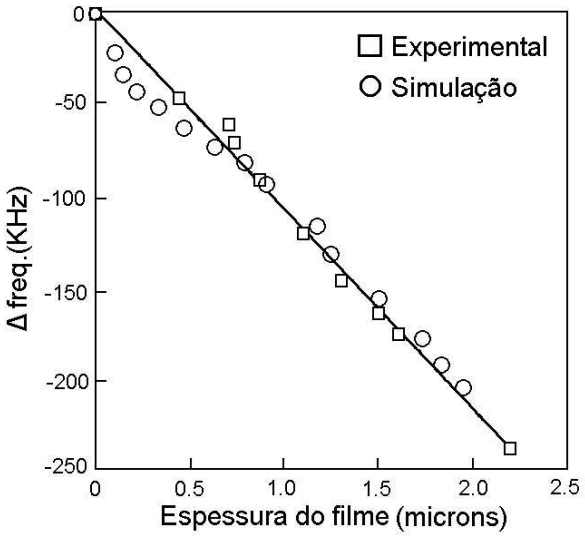

Figura 5.12: Resultado da simulação - Amp.Real

Capítulo 6

Conclusões e Trabalhos futuros

O modelo obtido com as equações de modos acoplados é compacto pois a estrutura interdigitada é modelada por uma linha de transmissão. Além disso, os parâmetros físicos do transdutor, tais como: acoplamento, freqüência central, banda passante, capacitância, velocidade de propagação, etc. são modelados diretamente pelos componentes do modelo. Diferente do que ocorre no de Bhattacharyya, onde parâmetros físicos como comprimento do transdutor e número de pares de eletrodos, não possuem componentes que os represente diretamente.

Uma vantagem em se utilizar uma linha de transmissão é que a simulação de transdutores longos não gera nós extras no modelo de circuito. Devido a sua simplicidade, pode ser implementado em qualquer simulador SPICE. Durante o nosso trabalho foram realizadas simulações com o SPICE3f4, PSPICE® e com o ELDO® da Mentor Graphics. Em todos eles, o nosso modelo demonstrou ser perfeitamente aplicável. Uma ressalva deve ser feita quanto à obtenção do comprimento da linha de transmissão com perdas, que para o caso do SPICE3f4, teve sua equação ajustada empiricamente. Este comportamento pode ser melhor esclarecido quando do maior aprofundamento do conhecimento do código fonte, no que se refere às equações que descrevem o comportamento do modelo LTRA e que, portanto, constitui sugestão para trabalhos futuros. Diferente do modelo anteriormente apresentado na literatura, nosso modelo não apresenta uma resposta em freqüência não-física na terceira harmônica.

Fizemos uso de resultados experimentais, publicados em artigos, para validarmos o modelo proposto, comparando estes resultados com os obtidos em nossas simulações.

A aplicação do modelo como sensor de espessura de filme fino de polimetilmetacrilato revelou resultados bem satisfatórios quando comparados com valores experimentais.

Além do macromodelo, também implementamos um modelo físico de linha de transmissão acústica. Esse modelo foi acrescentado ao SPICE3f4 com sucesso e foi batizado com a letra *A*. A nova linha de transmissão é definida informando o parâmetro "VEL", que representa a velocidade de propagação. Esse modelo da linha acústica serviu para entender a estrutura do simulador SPICE e o mecanismo de introdução de modelos físicos, pois exigiu um estudo detalhado da estrutura do código-fonte e a filosofia de comunicação das várias partes do programa.

Como trabalho futuro pode-se citar um estudo mais detalhado da implementação de novos modelos no código fonte do SPICE; procurar entender os detalhes da implementação da linha LTRA no SPICE 3f4; uso da linha de transmissão acústica e testes com SPICE 3f4; utilizar o modelo no projeto e modelamento de senores inteligentes integrados.

Apêndice A

O que é SPICE

Neste apêndice queremos fornecer uma visão geral sobre o software de simulação de circuitos analógicos, SPICE (*Simulation Program with Integrated Circuit Emphasis*). Sua origem, características, derivações, estrutura de dados e alguns comandos relevantes serão comentados aqui. O intuito é introduzir alguns conceitos, de forma a consolidar o entendimento desta dissertação, dando ênfase ao que lhe é pertinente, deixando o manual do software como fonte de consulta [30] para outras informações.

A.1 Introdução

O SPICE foi desenvolvido pela Universidade da Califórnia, Berkeley, a partir de 1972 [31], tendo como objetivo a simulação de circuitos contendo componentes ativos e passivos, tais como: transistores bipolares, transistores de efeito de campo, diodos, resistores, capacitores, indutores, entre outros. Três tipos de análises fundamentais são realizadas: análise DC não linear, transitório não linear e AC linear.

Em 1975 foi lançado o SPICE2 [32] escrito em Fortran e em 1985 o SPICE3 [32], agora desenvolvido em linguagem C, baseado na versão 2G6 do SPICE2, porém com uma outra formatação para a saída de dados e alguns melhoramentos. Neste trabalho foi utilizado o SPICE3f4.

A versão original do SPICE, que é distribuída gratuitamente, deu origem a muitas outras versões comerciais. Dentre elas podemos citar:

ESPICE® - Desenvolvida pela Philips Components em Hamburg.

HSPICE® - Desenvolvida pela Meta-Software, Inc. para ambiente UNIX, compatível com a formatação de entrada de dados da versão original do SPICE.

Intusoft® SPICE - Desenvolvida para ambiente PC, pela Intusoft.

PSPICE® - Desenvolvida para ambiente PC, pela MicroSim.

XSPICE® - Desenvolvida para ambiente UNIX, pela Georgia Tech Research Institute.

Circuit Maker® - Desenvolvida para ambiente Windows, pela Interactive Image Technologies.

Workbench® - Desenvolvida para ambiente Windows, pela Protel International.

A.2 Estrutura geral

Podemos agrupar o SPICE em quatro módulos principais: entrada, saída, configuração e análise. Na análise DC são usadas matrizes de equações, as quais são resolvidas através de métodos iterativos. Na AC o sistema é linearizado em torno do ponto de operação. E na do transitório são utilizadas análises DC e integrações numéricas.

A descrição do circuito a ser analizado é feita através de um arquivo contendo um conjunto de linhas de instruções, as quais se classificam em:

Linha de título: A primeira linha do arquivo é sempre o título que será usado para identificar a saída dos dados.

Linha de final: Cada arquivo de entrada deve terminar com uma linha de final, composta da expressão `.END`.

Linha de comentário: Qualquer linha iniciada com um "*" é tratada como um comentário e ignorada pelo SPICE.

Linha de elemento: O circuito a ser analizado é descrito usando um conjunto destas linhas. Inicia com o nome do dispositivo, o qual é identificado através da primeira letra (A-Z), ou seja, "R" para um resistor ou "C" para o caso de um capacitor. Depois são mencionados os seus nós de conexão, seguidos de alguns

parâmetros, como por exemplo o valor de um resistor. Os nós são representados por números, sendo que o nó de referência obrigatoriamente deve ser o zero. Na versão 3f4 do SPICE os nós podem ser representados por caracteres alfanuméricos, com excessão do nó de referência.

Linha de controle: Começa sempre com o símbolo ".". Especificam um conjunto de parâmetros para análise, tais como: temperatura dos componentes, modelo do dispositivo, subcircuitos e que tipo de análise deve ser realizada.

O SPICE trata letras maiúsculas e minúsculas da mesma maneira. Fatores de escala como "m" e "M" representa mili. Para indicar mega deve ser usado "meg" ou "MEG". Quaisquer caracteres colocados após o fator de escala será ignorado, ou seja, "MEG" é o mesmo que "MEGA". Não é aconselhável usar tabulações ou caracteres de controle, pois acarretarão em erros. Outros fatores de escala são: T=tera, G=giga, K=quilo, MIL= $25,4^{-6}$, U=micro, N=nano e P=pico

Caso seja necessário a continuação de uma linha em outra, pode-se acrescentar o símbolo "+" na coluna um da linha subsequente, a qual será interpretada como parte integrante da anterior.

A.3 Modelo físico do dispositivo

Cada dispositivo possui parâmetros que lhes são característicos. Alguns em comum para um conjunto de dispositivos, e outros bem peculiares. Sendo assim precisamos poder especificá-los através de uma forma distinta. Para isso, o SPICE possui uma linha de controle, que é iniciada com `.MODEL`, e possui a seguinte sintaxe:

```
.MODEL NOMEMOD TIPO (parâmetros)
```

onde:

`NOMEMOD` Nome escolhido para o modelo do dispositivo (alfanumérico).

`TIPO` Nome do dispositivo físico implementado no SPICE.

`parâmetros` Valores dos parâmetros pertinentes ao dispositivo em questão.

Alguns exemplos:

```
.MODEL MOD1 NPN BF=50 IS=1E-13 VBF=50

.MODEL LINHA LTRA LEN=22M R=27K L=370M G=0 C=370N
```

A.4 Modelo de subcircuito do dispositivo

Esta é uma outra forma de caracterização de um modelo de dispositivo. Sendo este implementado através de um circuito equivalente. Este subcircuito é definido dentro do arquivo de entrada por um grupo de linhas de elementos, o qual se inicia com a linha SUBCKT e termina com a linha ENDS. A referência a este subcircuito é feita através da linha de elemento "X". As suas formas serão descritas a seguir.

A.4.1 Linha SUBCKT

Sua forma geral é:

```
.SUBCKT SUBNOME N1 <N2 N3 ...>
```

onde:

`SUBNOME` Nome escolhido para o subcircuito (alfanumérico).

`N1` Número do primeiro nó do subcircuito.

`N2` Número do segundo nó do subcircuito, e assim por diante.

Alguns exemplos:

```
.SUBCKT OPAMP 1 2 3 4

.SUBCKT RETIFICADOR  10 11 12 13
```

A.4.2 Linha ENDS

Sua forma geral é:

```
.ENDS <SUBNOME>
```

onde:

`SUBNOME` Nome do subcircuito (opcional).

Alguns exemplos:

```
.ENDS OPAMP
```

```
.ENDS
```

A.4.3 Linha de chamada do subcircuito

Sua forma geral é:

```
Xxxxxxxx N1 <N2 N3 ...>
```

onde:

`xxxxxxx` Nome do dispositivo "X" que faz referência a um subcircuito.

`N1` Número do primeiro nó do circuito.

`N2` Número do segundo nó do circuito, e assim por diante.

Alguns exemplos:

```
.X1 2 4 17 3 1 MULTI
```

```
.XOPERAC 9 10 11 12 OPAMP
```

A.5 Definindo o tipo de análise

O SPICE é capaz de realisar análises AC, DC, do transitório, do ponto de operação e dos pólos e zeros. O ruído no circuito também pode ser analisado pelo SPICE. O tipo de análise é definido através de uma linha de controle, algumas delas serão descritas a seguir.

A.5.1 Análise AC de pequenos sinais

Sua forma geral é:

```
.AC DEC ND FREQINI FREQFIM

.AC OCT NO FREQINI FREQFIM

.AC LIN NP FREQINI FREQFIM
```

onde:

`DEC` Estabelece uma variação por década.

`OCT` Estabelece uma variação por oitava.

`LIN` Estabelece uma variação linear.

`ND` Número de pontos por década.

`NO` Número de pontos por oitava.

`NP` Número de pontos.

`FREQINI` Frequência inicial.

`FREQFIM` Frequência final.

Alguns exemplos:

```
.AC DEC 10 1 10K

.AC DEC 10 1K 100MEG

.AC LIN 100 1 100HZ
```

A.5.2 Análise da função de transferência DC

Sua forma geral é:

```
.DC NOMEFONTE VINI VFIM VINCR [NOMEFONTE2 VINI2
+   VFIM2 VINCR2]
```

onde:

`NOMEFONTE` Nome da fonte independente de tensão ou corrente.

`VINI` Valor da tensão ou corrente inicial da fonte.

`VFIM` Valor da tensão ou corrente final da fonte.

`VINCR` Valor de incremento para a grandeza da fonte.

`NOMEFONTE2` Especificação opcional de uma segunda fonte.

`VINI2` Valor inicial para a segunda fonte.

`VFIM2` Valor final para a segunda fonte.

`VINCR2` Valor do incremento para a segunda fonte.

Alguns exemplos:

```
.DC VIN 0.25 5.0 0.25

.DC VDS 0 10 .5 VGS 0 5 1

.DC VCE 0 10 .25 IB 0 10U 1U
```

A.5.3 Análise do transitório

Sua forma geral é:

```
.TRAN TPASSO TFIM <TINI <TMAX>>
```

onde:

`TPASSO` Tamanho do passo para análise.

`TFIM` Tempo final da análise.

`TINI` Tempo inicial da análise (opcional). O seu valor padrão é zero.

`TMAX` Tamanho máximo para o passso (opcional).

Alguns exemplos:

```
.TRAN 1NS 100NS

.TRAN 1NS 1000NS 500NS

.TRAN 10NS 1US
```

A.6 Dispositivos elementares

O SPICE possui alguns dispositivos elementares implementados, tais como: resistor, capacitor, indutor, fonte controlada e independente. Alguns deles serão descritos a seguir.

A.6.1 Resistor

Sua forma geral é:

```
Rxxxxxxx N1 N2 VALOR
```

onde:

`xxxxxxx` Nome designado para o resistor (alfanumérico).

`N1` Número do primeiro nó.

`N2` Número do segundo nó.

`VALOR` Valor da resistência.

Alguns exemplos:

```
RIN 1 0 10K

R2 4 5 100MEG
```

A.6.2 Capacitor

Sua forma geral é:

```
Cxxxxxxx N+ N- VALOR <IC=CONDINI>
```

onde:

`xxxxxxx` Nome designado para o capacitor (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`VALOR` Valor da capacitância em Farads.

`<IC=CONDINI>` Valor opcional da condição inicial em Volts.

Alguns exemplos:

```
COUT 3 0 10PICO IC=3V

C5 10 11 100N
```

A.6.3 Indutor

Sua forma geral é:

```
Lxxxxxxx N+ N- VALOR <IC=CONDINI>
```

onde:

`xxxxxxx` Nome designado para o indutor (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`VALOR` Valor da indutância em Henries.

`<IC=CONDINI>` Valor opcional da condição inicial em amperes.

Alguns exemplos:

```
LLINK 42 69 1UH

LSHUNT 23 51 10U IC=15.7MA
```

A.6.4 Fonte independente de tensão

Sua forma geral é:

```
Vxxxxxxx N+ N- <<DC> DC/TRAN VALOR> <AC<ACMAG <ACFASE>>>
+    <DISTOF1<F1MAG <F1FASE>>> <DISTOF2<F2MAG <F2FASE>>>
```

onde:

`xxxxxxx` Nome designado para a fonte de tensão (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`<<DC> DC/TRAN VALOR>` Valores da fonte em DC e no transitório, em Volts. Caso a fonte tenha valores DC e TRAN invariáveis, apenas o DC pode ser usado.

`<AC<ACMAG <ACFASE>>>` `ACMAG` é a magnitude AC e ACFASE é a sua fase.

`<DISTOF1<F1MAG <F1FASE>>>` `DISTOF1` é a palavra chave que indica que a fonte possui distorção de entrada na frequência F1, sendo `F1MAG` e `F1FASE`, a magnitude e a fase, respectivamente.

`<DISTOF2<F2MAG <F2FASE>>>` `DISTOF2` é a palavra chave que indica que a fonte possui distorção de entrada na frequência F2, sendo `F2MAG` e `F2FASE`, a magnitude e a fase, respectivamente.

Alguns exemplos:

```
VCC 10 0 DC 6

VIN 13 2 AC 0.333 45.0

VCARRIER 1 0 DISTOF1 0.1 -90.0

VMODULATOR 2 0 DISTOF2 0.01
```

A.6.5 Fonte linear de corrente dependente da tensão

Sua forma geral é:

```
Gxxxxxxx N+ N- NC+ NC- VALOR
```

onde:

`xxxxxxx` Nome designado para a fonte (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`NC+` Número do nó positivo da tensão de controle.

`NC-` Número do nó negativo da tensão de controle.

`VALOR` Valor da transcondutância em mhos.

Alguns exemplos:

```
G1 2 0 5 0 0.1MMHO

GFONTE 5 0 10 0 10
```

A.6.6 Fonte linear de corrente dependente da corrente

Sua forma geral é:

```
Fxxxxxxx N+ N- VNOME VALOR
```

onde:

`xxxxxxx` Nome designado para a fonte (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`VNOME` Nome da fonte de tensão que representa a corrente de controle.

`VALOR` Valor do ganho de corrente.

Alguns exemplos:

```
F1 13 5 VSENS 5
```

```
F2 5 0 V1 100
```

A.6.7 Fonte linear de tensão dependente da tensão

Sua forma geral é:

```
Exxxxxxx N+ N- NC+ NC- VALOR
```

onde:

`xxxxxxx` Nome designado para a fonte (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`NC+` Número do nó positivo da tensão de controle.

`NC-` Número do nó negativo da tensão de controle.

`VALOR` Valor do ganho de tensão.

Alguns exemplos:

```
E1 2 3 14 1 2.0
```

```
EFONTE 5 0 10 0 10
```

A.6.8 Fonte linear de tensão dependente da corrente

Sua forma geral é:

```
Hxxxxxxx N+ N- VNOME VALOR
```

onde:

`xxxxxxx` Nome designado para a fonte (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`VNOME` Nome da fonte de tensão que representa a corrente de controle.

`VALOR` Valor da transresistência em ohms.

Alguns exemplos:

```
HX 5 17 VZ 0.5K

H1 3 0 V2 100
```

A.6.9 Fonte dependente não linear

Sua forma geral é:

```
Bxxxxxxx N+ N- <I=EXPR> <V=EXPR>
```

onde:

`xxxxxxx` Nome designado para a fonte (alfanumérico).

`N+` Número do nó positivo.

`N-` Número do nó negativo.

`EXPR` Expressão de relação para a corrente ou tensão de controle, se a expressão for fornecida para a tensão (V=EXPR) a fonte não linear será também uma fonte de tensão, caso contrário a fonte será de corrente.

Alguns exemplos:

```
B1 0 1 I=COS(V(1))+SIN(V(2))

B1 0 1 V=LN(COS(LOG(V(1,2)^2)))-V(3)^4+V(2)^V(1)
```

A.7 O Dispositivo LTRA

Este dispositivo refere-se à uma linha de transmissão com perdas, do tipo RLC, RC, LC ou RG. A sua forma geral é:

```
Oxxxxxxx N1 N2 N3 N4 NOMEMOD
```

onde:

`xxxxxxx` Nome designado para a linha de transmissão com perdas (alfanumérico).

`N1` Número do nó superior do lado esquerdo da linha.

`N2` Número do nó inferior do lado esquerdo da linha.

`N3` Número do nó superior do lado direito da linha.

`N4` Número do nó inferior do lado direito da linha.

`NOMEMOD` Nome do modelo associado.

Alguns exemplos:

```
O1 3 0 4 0 LINHA1

O5 2 3 4 5 LINHACOMPERDA
```

O modelo do LTRA possui vários parâmetros, uns obrigatórios, outros opcionais. A definição do modelo possui a seguinte forma:

```
.MODEL NOMEMOD LTRA LEN=valor <parâmetros>
```

onde:

`NOMEMOD` Nome escolhido para o modelo.

`LTRA` Associa o dispositivo LTRA (Linha de Transmissão) ao modelo.

`LEN=valor` Estabelece o comprimento da linha. Este valor deve, obrigatoriamente, ser especificado.

os parâmetros são:

`R=valor` Estabelece o valor da resistência nos casos de linhas do tipo RLC, RC ou RG. O seu valor padrão é zero.

`L=valor` Estabelece o valor da indutância nos casos de linhas do tipo RLC ou LC. O seu valor padrão é zero.

`G=valor` Estabelece o valor da condutância nos casos da linha do tipo RG. O seu valor padrão é zero.

`C=valor` Estabelece o valor da capacitância nos casos de linhas do tipo RLC, RC ou LC. O seu valor padrão é zero.

Outros parâmetros estão listados no apêndice.

Alguns exemplos:

```
.MODEL LINHA1 LTRA LEN=39M R=27K G=0 L=370M C=370N

.MODEL LINHACOMPERDA  LTRA  LEN=1.45M  R=737K  G=0
+                           L=1.76 C=65N
```

Outros parâmetros do dispositivo são:

`REL=valor` Estabelece um valor para o erro relativo utilizado nas iterações, o seu valor padrão é um.

`ABS=valor` Estabelece um valor para o erro absoluto utilizado nas iterações, o seu valor padrão é um.

`COMPACTREL` Valor de tolerância do erro relativo quando utilizado o modo de condensação do histórico dos valores de tensão e corrente de entrada da linha de transmissão com perdas.

`COMPACTABS=valor` Valor de tolerância do erro absoluto quando utilizado o modo de condensação do histórico dos valores de tensão e corrente de entrada da linha de transmissão com perdas.

`NOSTEPLIMIT` Indica que não será usada a restrição padrão que estabelece que o passo de iteração seja menor do que o atraso causado pela linha do tipo RLC.

`NOCONTROL` Indica que um critério, baseado em um limite de erro, para a escolha do passo utilizado nas convoluções não será usado.

`LININTERP` Indica que será usada interpolação linear ao invés da interpolação quadrática para o cálculo dos sinais com atraso.

`MIXEDINTERP` Indica que será usada a interpolação linear caso a quadrática não for o melhor método.

`TRUNCNR` Indica que será utilizado o método de Newton-Raphson pelas rotinas de controle do incremento.

`TRUNCDONTCUT` Indica que não será usado um critério para a escolha do passo de forma a diminuir o erro nos cálculos da resposta ao impulso.

A.8 Configurando o simulador

Várias variáveis de simulação estão disponíveis no SPICE3, e podem ser alteradas para controlar a sua precisão, velocidade ou valores padrão de alguns dispositivos. Estes valores podem ser alterados utilizando a linha de controle na forma:

```
.OPTIONS [variáveis]
```

onde as variáveis podem ser:

`ABSTOL=valor` Estabelece um valor de tolerância para o erro absoluto.

`BADMOS3` Indica que o modelo para o MOS3 que utiliza a descontinuidade "kappa" deverá ser empregado.

`CHGTOL=valor` Estabelece a tolerância de carga do simulador. O valor padrão é 1,0e-14.

`DEFAD=valor` Estabelece o valor da área de difusão do dreno do transistor MOS. O valor padrão é 0,0.

`DEFAS=valor` Estabelece o valor da área de difusão do fonte do transistor MOS. O valor padrão é 0,0.

`DEFL=valor` Estabelece o valor do comprimento do canal do transistor MOS. O valor padrão é 1000,0 micrometros.

`DEFW=valor` Estabelece o valor da largura do canal do transistor MOS. O valor padrão é 100,0 micrometros.

`GMIN=valor` Estabelece o valor da mínima condutância permitida pelo simulador. O valor padrão é 1,0e-12.

`ITL1=valor` Estabelece o valor limite de iterações para a análise DC. O valor padrão é 100.

`ITL2=valor` Estabelece o valor limite de iterações para a análise da curva de transferência DC. O valor padrão é 50.

`ITL4=valor` Estabelece o valor limite de iterações para a análise do transitório. O valor padrão é 10.

`KEEPOPINFO` Retém a informação do ponto de operação quando uma análise AC, DC, de distorção ou de Pólo-Zero está sendo executada. Isto é particularmente utilizado se o circuito é grande e você não quer executar uma análise redundante do ponto de operação.

`METHOD=nome` Estabelece o método usado para a integração numérica. Os possíveis nomes são: "Gear" ou "Trapezoidal". O nome padrão é trapezoidal.

`PIVREL=valor` Estabelece a taxa relativa entre o maior valor por coluna da matriz de entrada e um valor para pivot aceitável. O valor padrão é 1,0e-13. No algorítmo numérico de pivoteamento o valor mínimo permitido para o pivot é determinado por EPSREL = AMAX1(PIVREL*MAXVAL, PIVTOL), onde MAXVAL é o maior elemento da coluna aonde um pivoteamento é aplicado (pivoteamento parcial).

`PIVTOL=valor` Estabelece o valor mínimo absoluto para um valor da matriz de entrada ser aceito como um pivot. O valor padrão é 1,0e-13.

`RELTOL=valor` Estabelece o valor da tolerância do erro relativo do programa. O valor padrão é 0,001 (0,1%).

`TEMP=valor` Estabelece o valor da temperatura de operação do circuito. O valor padrão é 27 graus Celsius.

`TNOM=valor` Estabelece o valor da temperatura nominal em que os parâmetros dos dispositivos foram medidos. O valor padrão é 27 graus Celsius.

`TRTOL=valor` Estabelece a tolerância do erro transitório. O valor padrão é 7,0. Este parâmetro é uma estimativa do fator pelo qual o SPICE superestima o erro de truncamento atual.

`TRYTOCOMPACT` Aplicável somente ao modelo LTRA. Quando especificado, o simulador tenta condensar a história passada dos valores de corrente e tensão de entrada das linhas de transmissão com perdas.

`VNTOL=valor` Estabelece a tolerância do erro absoluto de tensão do programa. O valor padrão é 1 microvolt.

A.9 Interface de Linha

O Spice quando disponibiliza um *prompt* do tipo "Spice 4->" (o número 4 só indica que outros três *prompts* foram mostrados anteriormente) significa que está no modo de interface de linha, onde comandos digitados diretamente via console são imediatamente executados após o "ENTER". Os comandos devem ser escritos em letras minúsculas. Alguns deles são:

A.9.1 Help

Apresenta um texto de ajuda sobre os comandos solicitados ou sobre todos eles.

Sua forma geral é:

```
help [all] [comando] ...
```

onde:

`all` Indica que a ajuda se refere a todos os comandos do interpretador.

`comando` Estabele uma lista de comandos sobre os quais se deseja obter ajuda.

A.9.2 Edit

Este comando inicia um editor de texto para que o arquivo de descrição do circuito seja modificado ou criado, caso já exista ou não, respectivamente.

Sua forma geral é:

```
edit [arquivo]
```

onde:

`arquivo` Nome do arquivo a ser editado.

A.9.3 Load

Carrega arquivos de descrição de circuito para serem analisados.

Sua forma geral é:

```
load [nome] ...
```

onde:

`nome` Nomes dos arquivos que se deseja carregar.

A.9.4 Run

Executa a simulação do circuito descrito no arquivo.

Sua forma geral é:

```
run [arquivo]
```

onde:

`arquivo` Nome do arquivo de descrição do circuito.

A.9.5 Print

Mostra os valores dos vetores solicitados.

Sua forma geral é:

```
print [coluna] [linha] expr ...
```

onde:

`coluna` Indica que os valores do vetores serão dispostos em colunas.

`linha` Indica que os valores dos vetores serão dispostos em linhas.

`expr` Nome dos vetores ou relações matemáticas entre eles, por exemplo, V(1)-V(2).

A.9.6 Plot

Traça o gráfico dos vetores solicitados.

Sua forma geral é:

```
plot exprs [ylimit yinf ysup] [xlimit xinf xsup] [xindices
+    xini xfim]  [xcompress comp]  [xdelta xespa]  [ydelta
+    yespa] [xlog] [ylog]  [loglog]  [vs xescala]  [xlabel
+    nome] [ylabel nome] [title titulo]  [samep]  [linear]
```

onde:

`exprs` Nome dos vetores ou relações matemáticas entre eles, por exemplo, V(1)-V(2).

`ylimit` Indica que os valores limites do eixo y serão informados.

`yinf` Valor limite inferior do eixo y.

`ysup` Valor limite superior do eixo y.

`xlimit` Indica que os valores limites do eixo x serão informados.

`xinf` Valor limite inferior do eixo x.

`xsup` Valor limite superior do eixo x.

`xindices` Indica que apenas um intervalo do gráfico será traçado.

`xini` Valor inicial do intervalo.

`xfim` Valor final do intervalo.

`xcompress` Indica que apenas um número de pontos especificado será usado na construção do gráfico.

`comp` Número de pontos usados como intervalo na construção do gráfico.

`xdelta` Indica que um espaçamento entre as linhas de grade do eixo x será fornecido.

`xespa` Valor do espaçamento entre as linhas de grade do eixo x.

`ydelta` Indica que um espaçamento entre as linhas de grade do eixo y será fornecido.

`yespa` Valor do espaçamento entre as linhas de grade do eixo y.

`xlog` Indica que a escala do eixo x é logarítmica.

`ylog` Indica que a escala do eixo y é logarítmica.

`loglog` Indica que ambos os eixos possuem escala logarítmica.

`vs` Indica que um valor de escala para o eixo x será fornecido.

`xescala` Valor da escala do eixo x.

`xlabel` Indica que um nome será fornecido para o eixo x.

`nome` Nome especificado para o eixo x.

`ylabel` Indica que um nome será fornecido para o eixo y.

`nome` Nome especificado para o eixo y.

`title` Indica que um título será especificado para o gráfico.

`titulo` Título do gráfico.

`samep` Indica que os parâmetros estabelecidos em gráficos anteriores serão válidos para este.

`linear` Define um gráfico padrão com escala logarítmica.

A.9.7 Quit

Sai do SPICE.

Sua forma geral é:

```
quit
```

A.10 Simulando um exemplo

Para consolidar o que foi apresentado, será descrito uma simulação de um circuito simples com alguns dos componentes já vistos. Primeiramente os nós são numerados de forma que cada um possua um número distinto, e nomes são atribuídos aos componentes, respeitando a sua forma geral. O esquema do circuito é mostrado na figura A.1.

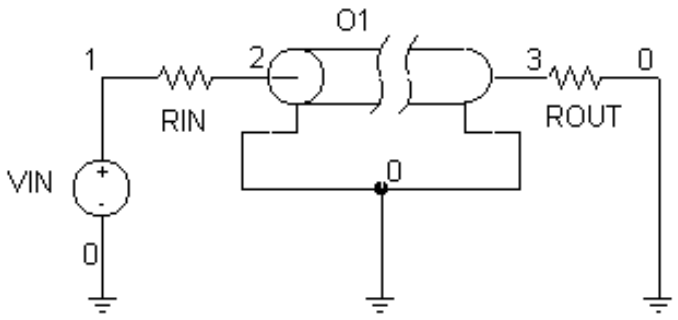

Figura A.1: Circuito exemplo1

O primeiro passo é criar o arquivo de descrição do circuito. Podemos criá-lo dentro do SPICE. Para carregarmos o SPICE devemos digitar o nome do seu arquivo executável, no nosso caso em especial digitamos "spice3", e uma linha de comando com o *prompt* "Spice 1->" surgirá. Agora o comando "edit exemplo1" pode ser digitado. Com isso o editor de textos associado ao SPICE criará um arquivo chamado exemplo1 e ficará em modo de edição. No nosso caso o editor em questão é o JOVE e o conteúdo deste arquivo é mostrado na figura A.2.

Pode ser observado que os nós foram numerados como 1, 2 , 3 e 0 para o nó de referência, e que cada componente teve um nome designado de acordo com a sua forma geral. Após a digitação da linha de final `.END` devemos gravar o arquivo e sair do editor. Feito isto o SPICE perguntará se deseja iniciar a simulação, caso responda "y" os cálculos serão executados e ao seu término estará disponível um novo *prompt* para que os comandos de construção de gráficos, medidas de tensão entre nós, ou seja, para que o comportamento do circuito submetido à análise seja investigado. Outra forma de fazermos isto é a criação do arquivo em qualquer editor de textos sem a necessidade de carregarmos o SPICE. Uma vez no modo de interface de linha do

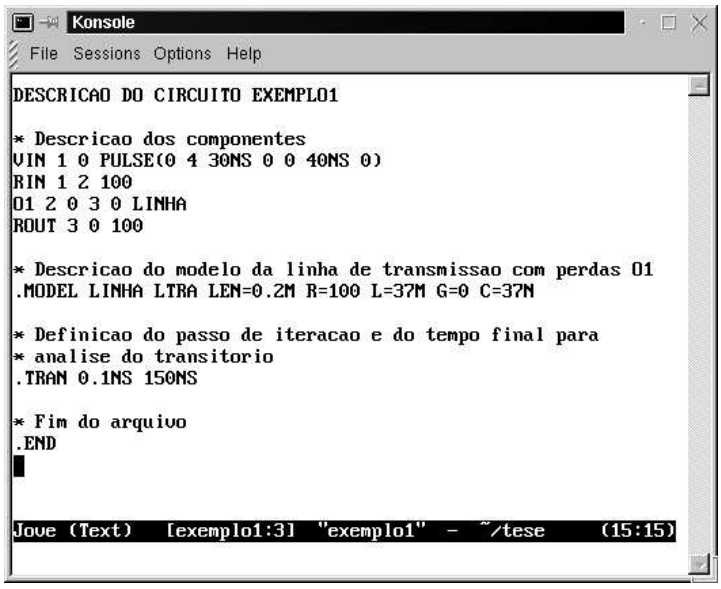

Figura A.2: Arquivo exemplo1

SPICE basta usarmos o comando `run exemplo1` para que a simulação seja realizada.

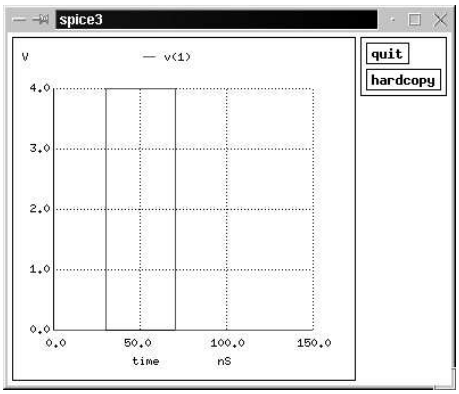

Figura A.3: Gráfico de V(1)

Na figura A.3 é visto o gráfico da tensão de entrada V(1) obtido a partir do comando `plot v(1)`, já na figura A.4 é mostrado o gráfico da tensão de saída V(3).

Para que o uso de um modelo de subcircuito seja melhor compreendido, especificaremos a linha de transmissão com perdas, presente no circuito exemplo1, através de um subcircuito. O novo arquivo de descrição está contido na figura A.5.

Como se trata do mesmo circuito, o resultado da simulação é o mesmo. O que pode ser comprovado através da figura A.6 que mostra o gráfico das tensões V(1) e V(3) simultaneamente, obtido através do comando `plot V(1) V(3)`.

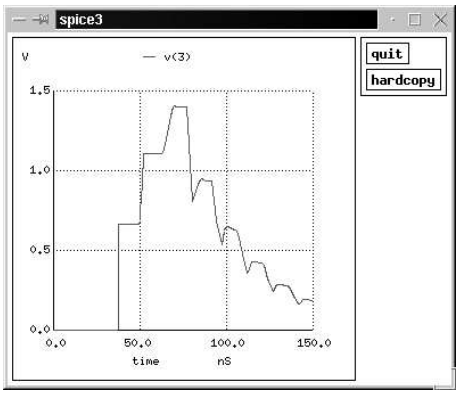

Figura A.4: Gráfico de V(3)

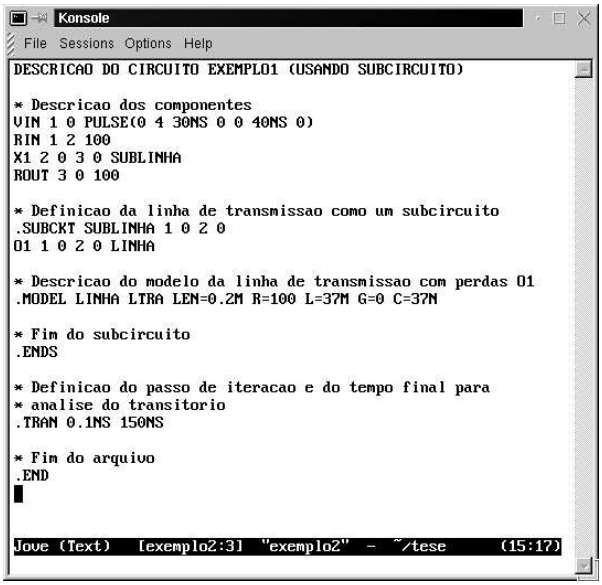

Figura A.5: Arquivo exemplo2

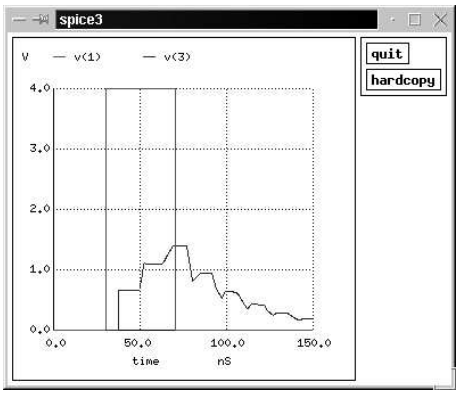

Figura A.6: Gráfico de V(1) e V(3)

Apêndice B

Alterações do código fonte do SPICE3f4

Neste apêndice estão listados trechos dos códigos dos arquivos fonte do SPICE3f4 que sofereram modificações ou foram criados. Todas as alterações do código estão sinalizadas com comentários.

B.1 Diretório: /spice3f4/conf

B.1.1 Arquivo: defaults

Neste arquivo estão definidos algumas variáveis utilizadas para a geração do arquivo executável. Foi acrescentado o dispositivo SAW na variável DEVICES.

```
# Default definitions
### To Edit This File:The contents of this file are used by the "make"
### command, and follow the syntax rules for Unix "make".  Long lines
### can be split by placing a '\' at the end of one line and
.
. <trecho do programa omitido>
.
# DEVICES lists the types of devices that you want to use in your
# copy of Spice3.  For a smaller binary, reduce the number of devices.
# Some systems may have trouble building the "bsim1" and/or "bsim2".
```

```
# It does not make sense to remove any of cap, cccs, ccvs, ind, isrc,
# res, vccs, vcvs, or vsrc.
# asrc: arbitrary voltage/current source
# bjt: bipolar junction transistor
.
. <trecho do programa omitido>
.
# mos6: MOS, fast analytic, short-channel
# res: resistor
# saw:  Dispositivo saw (Alberto) /* acrescentado */
# sw: switch
# tra: lossless transmission line
# urc: uniform RC line
# vccs: voltage-controlled current source
# vcvs: voltage-controlled voltage source
# vsrc: voltage source
#
# /* acrescentado o dispositivo saw em DEVICES  */
DEVICES = asrc bjt bsim1 bsim2 cap cccs ccvs csw dio ind isrc \
  jfet ltra mes mos1 mos2 mos3 mos6 res saw sw tra urc \
  vccs vcvs vsrc
# ANALYSES list the analysis types that you want to have available in
# Spice3.  As with DEVICES, this can reduce the size of the resulting
.
. <trecho do programa omitido>
.
###
### The End.   Use the 'build' command in util/ to build spice.
###
```

B.2 Diretório: /spice3f4/util/skeleton

B.2.1 Arquivo: make_def_bd

Neste arquivo estão definidos algumas variáveis utilizadas para a geração do arquivo executável. Foi acrescentado o dispositivo SAW na variável ALL_DEVICES.

```
###########
# Copyright 1991 Regents of the University of California.
# All rights reserved.
###########
#
# Standard definitions.It occurs to me that too much has been left in.
SHELL = /bin/sh
PATH_SEP = /
PS = /
REVISION =
VERSION = 3f4
VERSION_REVISION= $(VERSION)$(REVISION)
NOTICE =
.
.  <trecho do programa omitido>
.
INSTALL_SUBDIRS = $(SUBDIRS)
MSC_SUBDIRS = $(SUBDIRS)
MAKE_SUBDIRS = $(SUBDIRS)
MSC_ERRS = msc.out
DEV_SUBDIRS = $(DEVICES)

# Alberto - LDN/DES/UFPE
# alterado ALL_DEVICES

ALL_DEVICES = asrc bjt bsim1 bsim2 cap cccs ccvs csw dio ind isrc \
  jfet ltra mes mos1 mos2 mos3 mos6 res saw sw tra urc \
  vccs vcvs vsrc
```

```
ASM_HACK =
SRC_TOP = $(DIST_DIR)/src/
DEV_TOP = $(SRC_TOP)/lib/dev/
SRC_DIR = $(SRC_TOP)$(SUBDIR)
OBJ_TOP = $(SYS_DIR)/obj/
OBJ_DIR = $(OBJ_TOP)/$(SUBDIR)
CURR_DIR = $(DIST_DIR)/$(DIR)/$(SUBDIR)
DISTLIB_DIR = $(DIST_DIR)/lib
OBJBIN_DIR = $(OBJ_DIR)/bin
OUTPUT =
PARENT =
REAL_CC_OPT = $(CC_OPT)
CC_OPT_SAFE = $(CC_OPT)
MKDIR = mkdir
ARQ = q
```

B.3 Diretório: /spice3f4/src/bin

B.3.1 Programa: bconf.c

Neste arquivo constam algumas definições de cabeçalhos (.h) e dispositivos. Foi acrescentado o arquivo "sawitf.h" e o dispositivo "saw".

```
/*
 * Analyses
 */
#ifndef TABLES_ONLY
#define AN_op
#define AN_dc
#define AN_tf
#define AN_ac
#define AN_tran
#define AN_pz
#define AN_noise
#endif
```

```
/*
 * Devices
 */
#define DEV_asrc
#define DEV_bjt
#define DEV_cap
#define DEV_cccs
#define DEV_ccvs
#define DEV_csw
#define DEV_dio
#define DEV_ind
#define DEV_isrc
#define DEV_mos1
#define DEV_mos2
#define DEV_res

/* alterado em 15/09/99  */
/* Alberto - LDN/DES/UFPE */

#define DEV_saw
#define DEV_vccs
#define DEV_vcvs
#define DEV_vsrc
.
.  <trecho do programa omitido>
.
#ifdef HAS_FLAT_INCLUDES
#include "asrcitf.h"
#include "bjtitf.h"
#include "capitf.h"
#include "cccsitf.h"
#include "ccvsitf.h"
#include "cswitf.h"
#include "dioitf.h"
```

```
#include "inditf.h"
#include "isrcitf.h"
#include "mos1itf.h"
#include "mos6itf.h"
#include "resitf.h"
#include "sawitf.h"      /* alterado em 15/09/99 */
#include "switf.h"       /* Alberto - LDN/DES/UFPE */
#include "vccsitf.h"
#include "vcvsitf.h"
#include "vsrcitf.h"
.
.  <trecho do programa omitido>
.
#else
#include "asrc/asrcitf.h"
#include "bjt/bjtitf.h"
#include "cap/capitf.h"
#include "cccs/cccsitf.h"
#include "ccvs/ccvsitf.h"
#include "csw/cswitf.h"
#include "dio/dioitf.h"
#include "ind/inditf.h"
#include "isrc/isrcitf.h"
#include "mos1/mos1itf.h"
#include "mos6/mos6itf.h"
#include "res/resitf.h"
#include "saw/sawitf.h"        /* alterado em 15/09/99 */
#include "sw/switf.h"        /* Alberto - LDN/DES/UFPE */
#include "vccs/vccsitf.h"
#include "vcvs/vcvsitf.h"
#include "vsrc/vsrcitf.h"
.
.  <trecho do programa omitido>
.
```

```
#ifdef DEV_mos6
        &MOS6info,
#endif
#ifdef DEV_res
        &RESinfo,
#endif
#ifdef DEV_saw          /* alterado em 15/09/99  */
        &SAWinfo,/* Alberto - LDN/DES/UFPE */
#endif
#ifdef DEV_sw
        &SWinfo,
#endif
.
.  <trecho do programa omitido>
.

    CKTdoJob,       /* doAnalyses function */
    CKTtrouble,    /* non-convergence message function */
    sizeof(DEVices)/sizeof(SPICEdev *),
    (IFdevice**)DEVices,
    sizeof(analInfo)/sizeof(SPICEanalysis *),
    (IFanalysis **)analInfo,
    sizeof(nodeParms)/sizeof(IFparm),
    nodeParms,
    sizeof(specSigList)/sizeof(char *),
    specSigList,
#endif
};
```

B.3.2 Programa: cconf.c

Neste arquivo constam algumas definições de cabeçalhos (.h) e dispositivos. Foi acrescentado o arquivo "sawitf.h" e o dispositivo "saw".

```
/*
 * Analyses
 */
#ifndef TABLES_ONLY
#define AN_op
#define AN_dc
#define AN_tf
#define AN_ac
#define AN_tran
#define AN_pz
/* add the following three (NJ) */
#define AN_disto
#define AN_noise
#define AN_sense
#endif
/*
 * Devices
 */
#define DEV_asrc
#define DEV_bjt
/*add these (NJ) */
#define DEV_bsim1
#define DEV_bsim2
#define DEV_ltra
#define DEV_jfet
#define DEV_mes
#define DEV_mos2
#define DEV_mos3
#define DEV_mos6
#define DEV_sw
#define DEV_tra
#define DEV_urc
#define DEV_cap
#define DEV_cccs
```

```
#define DEV_ccvs
#define DEV_csw
#define DEV_dio
#define DEV_ind
#define DEV_isrc
#define DEV_mos1
#define DEV_res
#define DEV_saw      /*  alterado em 15/09/99  */
#define DEV_vccs     /* Alberto - LDN/DES/UFPE */
#define DEV_vcvs
#define DEV_vsrc
.
.  <trecho do programa omitido>
.
#ifdef HAS_FLAT_INCLUDES
#include "asrcitf.h"
#include "bjtitf.h"
#include "capitf.h"
#include "cccsitf.h"
#include "ccvsitf.h"
#include "cswitf.h"
#include "dioitf.h"
#include "inditf.h"
#include "isrcitf.h"
#include "mos1itf.h"
#include "mos6itf.h"
#include "resitf.h"
#include "sawitf.h"     /*  alterado em 15/09/99   */
#include "switf.h" /* Alberto - LDN/DES/UFPE */
#include "vccsitf.h"
#include "vcvsitf.h"
#include "vsrcitf.h"
.
.  <trecho do programa omitido>
```

```
.
#else
#include "asrc/asrcitf.h"
#include "bjt/bjtitf.h"
#include "cap/capitf.h"
#include "cccs/cccsitf.h"
#include "ccvs/ccvsitf.h"
#include "csw/cswitf.h"
#include "dio/dioitf.h"
#include "ind/inditf.h"
#include "isrc/isrcitf.h"
#include "mos1/mos1itf.h"
#include "mos6/mos6itf.h"
#include "res/resitf.h"
#include "saw/sawitf.h"      /*  alterado em 15/09/99 */
#include "sw/switf.h"      /* Alberto - LDN/DES/UFPE */
#include "vccs/vccsitf.h"
#include "vcvs/vcvsitf.h"
#include "vsrc/vsrcitf.h"
.
.  <trecho do programa omitido>
.
#ifdef DEV_mos6
        &MOS6info,
#endif
#ifdef DEV_res
        &RESinfo,
#endif
#ifdef DEV_saw              /*  alterado em 15/09/99  */
        &SAWinfo,           /*  Alberto - LDN/DES/UFPE */
#endif                      /*                         */
#ifdef DEV_sw
        &SWinfo,
#endif
```

```
.
.  <trecho do programa omitido>
.
    CKTdoJob,       /* doAnalyses function */
    CKTtrouble,    /* non-convergence message function */
    sizeof(DEVices)/sizeof(SPICEdev *),
    (IFdevice**)DEVices,
    sizeof(analInfo)/sizeof(SPICEanalysis *),
    (IFanalysis **)analInfo,
    sizeof(nodeParms)/sizeof(IFparm),
    nodeParms,
    sizeof(specSigList)/sizeof(char *),
    specSigList,
#endif
};
```

B.3.3 Programa: config.c

Assim como nos arquivos `bconf.c` e `cconf.c`, neste arquivo constam algumas definições de cabeçalhos (.h) e dispositivos. Foi acrescentado o arquivo "sawitf.h" e o dispositivo "saw", o uso de um ou outro arquivo depende para qual sistema operacional o código executável do SPICE será gerado.

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
**********/
#include "spice.h"
#ifdef DEVICES_USED
static char *devs = DEVICES_USED;
#endif
#ifdef ANALYSES_USED
static char *ans = ANALYSES_USED;
#endif
.
```

```
.  <trecho do programa omitido>
.
#ifdef HAS_FLAT_INCLUDES
#include "asrcitf.h"
#include "bjtitf.h"
#include "capitf.h"
#include "cccsitf.h"
#include "ccvsitf.h"
#include "cswitf.h"
#include "dioitf.h"
#include "inditf.h"
#include "isrcitf.h"
#include "mos1itf.h"
#include "mos6itf.h"
#include "resitf.h"
#include "sawitf.h"          /*  alterado em 16/09/99   */
#include "switf.h"      /* Alberto - LDN/DES/UFPE */
#include "vccsitf.h"
#include "vcvsitf.h"
#include "vsrcitf.h"
.
.  <trecho do programa omitido>
.
#else
#include "asrc/asrcitf.h"
#include "bjt/bjtitf.h"
#include "cap/capitf.h"
#include "cccs/cccsitf.h"
#include "ccvs/ccvsitf.h"
#include "csw/cswitf.h"
#include "dio/dioitf.h"
#include "ind/inditf.h"
#include "isrc/isrcitf.h"
#include "mos1/mos1itf.h"
```

```
#include "mos6/mos6itf.h"
#include "res/resitf.h"
#include "saw/sawitf.h"        /*   alterado em 16/09/99   */
#include "sw/switf.h"        /* Alberto - LDN/DES/UFPE */
#include "vccs/vccsitf.h"
#include "vcvs/vcvsitf.h"
#include "vsrc/vsrcitf.h"
.
.  <trecho do programa omitido>
.
#ifdef DEV_mos6
        &MOS6info,
#endif
#ifdef DEV_res
        &RESinfo,
#endif
#ifdef DEV_saw             /*     alterado  16/09/99   */
        &SAWinfo,    /* Alberto - LDN/DES/UFPE */
#endif
#ifdef DEV_sw
        &SWinfo,
#endif
.
.  <trecho do programa omitido>
.
CKTdoJob,       /* doAnalyses function */
    CKTtrouble,    /* non-convergence message function */
    sizeof(DEVices)/sizeof(SPICEdev *),
    (IFdevice**)DEVices,
    sizeof(analInfo)/sizeof(SPICEanalysis *),
    (IFanalysis **)analInfo,
    sizeof(nodeParms)/sizeof(IFparm),
    nodeParms,
    sizeof(specSigList)/sizeof(char *),
```

```
    specSigList,
#endif
};
```

B.4 Diretório: /spice3f4/src/include

B.4.1 Programa: inpdefs.h

Neste arquivo de cabeçalho estão definidas algumas estruturas de dados. Aqui foi acrescentada a estrutura "INP2A".

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
#ifndef INP
#define INP
    /* structure declarations used by either/both input package */
.
.  <trecho do programa omitido>
.
int INPmkTerm(GENERIC*,char**,INPtables*,GENERIC**);
int INPtypelook(char*);
void INP2A(GENERIC*,INPtables*,card*); /* criada em 27/09/99 */
void INP2B(GENERIC*,INPtables*,card*); /* Alberto - LDN/DES/UFPE */
void INP2C(GENERIC*,INPtables*,card*);
void INP2D(GENERIC*,INPtables*,card*);
.
.  <trecho do programa omitido>
.
void INPtabEnd();
void INPptPrint();
void INPgetTree();
void INP2A();    /*   criada em 27/9/99  */
```

```
void INP2B(); /* Alberto - LDN/DES/UFPE */
void INP2C();
void INP2D();
.
.  <trecho do programa omitido>
.
void INP2W();
void INP2Z();
int INP2dot();
#endif /* stdc */
#endif /*INP*/
```

B.5 Diretório: /spice3f4/src/lib/ckt

B.5.1 Programa: dctran.c

Este arquivo diz respeito a análise do transitório. Foram acrescentadas referências ao modelo SAW.

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
/* subroutine to do DC TRANSIENT analysis
        --- ONLY, unlike spice2 routine with the same name! */
#include "spice.h"
#include "misc.h"
#include <stdio.h>
.
.  <trecho do programa omitido>
.
#ifdef HAS_SHORTMACRO
    double mt;  /* temporary so macro call won't cross line boundry */
#endif
    int ltra_num;
```

```
    int saw_num;      /*  alterado em 08/02/2000 */
/* Alberto - LDN/DES/UFPE */
    if(restart || ckt->CKTtime == 0) {
        delta=MIN(ckt->CKTfinalTime/50,ckt->CKTstep)/10;
/* begin LTRA/SAW code addition */
if (ckt->CKTtimePoints != NULL)
    FREE(ckt->CKTtimePoints);
if (ckt->CKTstep >= ckt->CKTmaxStep)
    maxstepsize = ckt->CKTstep;
else
    maxstepsize = ckt->CKTmaxStep;
ckt->CKTsizeIncr = 10;
ckt->CKTtimeIndex = -1; /* before the DC soln has been stored */
ckt->CKTtimeListSize = ckt->CKTfinalTime / maxstepsize + 0.5;
ltra_num = CKTtypelook("LTRA");
        saw_num = CKTtypelook("SAW");   /* alterado em 08/02/2000 */
         /* Alberto - LDN/DES/UFPE */
if (ltra_num >= 0 && ckt->CKThead[ltra_num] != NULL)
    ckt->CKTtimePoints = NEWN(double, ckt->CKTtimeListSize);
        /* if colocado em 08/02/2000 */
/* Alberto - LDN/DES/UFPE */
        if (saw_num >= 0 && ckt->CKThead[saw_num] != NULL)
            ckt->CKTtimePoints = NEWN(double, ckt->CKTtimeListSize);
        /* end LTRA/SAW code addition */
if(ckt->CKTbreaks) FREE(ckt->CKTbreaks);
        ckt->CKTbreaks=(double *)MALLOC(2*sizeof(double));
        if(ckt->CKTbreaks == (double *)NULL) return(E_NOMEM);
        *(ckt->CKTbreaks)=0;
.
.  <trecho do programa omitido>
.
#ifdef STEPDEBUG
                (void)printf("delta at delmin\n");
#endif
```

```
            } else {
                ckt->CKTcurrentAnalysis = DOING_TRAN;
                ckt->CKTstat->STATtranTime +=
                        (*(SPfrontEnd->IFseconds))()-startTime;
                ckt->CKTstat->STATtranIter +=
                        ckt->CKTstat->STATnumIter - startIters;
                ckt->CKTstat->STATtranDecompTime +=
                        ckt->CKTstat->STATdecompTime - startdTime;
                ckt->CKTstat->STATtranSolveTime +=
                        ckt->CKTstat->STATsolveTime - startsTime;
errMsg = CKTtrouble((GENERIC *) ckt, "Timestep too small");
                return(E_TIMESTEP);
            }
        }
    }
    /* NOTREACHED */
}
```

B.5.2 Programa: pzan.c

Este arquivo refere-se a análise do polo zero. Foi acrescentada a restrição da não execução da análise para o modelo SAW assim como existe para o modelo LTRA.

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
**********/
#include "spice.h"
#include <stdio.h>
#include "complex.h"
#include "cktdefs.h"
#include "smpdefs.h"
.
.  <trecho do programa omitido>
.
```

```
/*
 * Perform error checking
 */
int
PZinit(ckt)
    CKTcircuit *ckt;
{
    PZAN *pzan = (PZAN *) ckt->CKTcurJob;
    int i;
    i = CKTtypelook("transmission line");
    if (i == -1) {
    i = CKTtypelook("Tranline");
    if (i == -1) {
i = CKTtypelook("LTRA");
                if (i == -1)                    /* alterado em 04/10/99 */
                    i = CKTtypelook("SAW");     /* Alberto - LDN/DES/UFPE */
            }                                   /*                        */
    }
    if (i != -1 && ckt->CKThead[i] != NULL)
/* ERROR(E_XMISSIONLINE, "Transmission lines not supported") */
        ERROR(E_XMISSIONLINE, "Transmission lines/SAW not supported")
/* Alberto - LDN/DES/UFPE */
    pzan->PZpoleList = (PZtrial *) NULL;
    pzan->PZzeroList = (PZtrial *) NULL;
    pzan->PZnPoles = 0;
    pzan->PZnZeros = 0;
.
.  <trecho do programa omitido>
.
    outData.v.numValue = pzan->PZnPoles + pzan->PZnZeros;
    outData.v.vec.cVec = out_list;
    (*SPfrontEnd->OUTpData)(pzPlotPtr, (IFvalue *) 0, &outData);
    (*(SPfrontEnd->OUTendPlot))(pzPlotPtr);
    return(OK);
```

```
}
```

B.6 Diretório: /spice3f4/src/lib/fte

B.6.1 Programa: subckt.c

Este arquivo refere-se ao tratamento dos subcircuitos declarados. Foi acrescentado a referência ao dispositivo (A).

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Wayne A. Christopher, U. C. Berkeley CAD Group
**********/
/*
 * Expand subcircuits. This is very spice-dependent. Bug fixes by Norbert
 * Jeske on 10/5/85.
 */
#include "spice.h"
#include "cpdefs.h"
#include "ftedefs.h"
.
.  <trecho do programa omitido>
.
        case 'o':
        case 'a': /*  alterado em 07/04/2000  */
case 's':
case 'm':
.
.  <trecho do programa omitido>
.
int
inp_numnodes(c)
    char c;
{
    if (isupper(c))
```

```
        c = tolower(c);
    switch (c) {
        case ' ':
        case '\t':
        case '.':
        case 'x':
        case '*':
        return (0);
        case 'a': return (4);     /*  alterado em 06/04/2000 */
        case 'b': return (2);
        case 'c': return (2);
        case 'd': return (2);
        case 'e': return (4);
.
.  <trecho do programa omitido>
.
        case 'w': return (3);
        case 'z': return (3);
        default:
        fprintf(cp_err, "Warning: unknown device type: %c\n", c);
            return (2);
    }
}
```

B.7 Diretório: /spice3f4/src/lib/inp

B.7.1 Programa: inpdomod.c

Neste arquivo foi acrescentado o reconhecimento do modelo "SAW".

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
#include "spice.h"
```

```
#include <stdio.h>
#include "iferrmsg.h"
.
.  <trecho do programa omitido>
.
INPmakeMod(modname,type,image);
    } else if(strcmp(typename,"r") == 0) {
        type = INPtypelook("Resistor");
        if(type < 0) {
            err = INPmkTemp(
                    "Device type Resistor not available in this binary\n");
        }
        INPmakeMod(modname,type,image);
    /*  alterado em 27/9/99   */
    /*  Alberto - LDN/DES/UFPE */
    } else if(strcmp(typename,"saw") == 0) {
        type = INPtypelook("SAW");
        if(type < 0) {
            err = INPmkTemp("Device type SAW not available in this binary\n");
        }
        INPmakeMod(modname,type,image);
    } else if(strcmp(typename,"c") == 0) {
        type = INPtypelook("Capacitor");
        if(type < 0) {
            err = INPmkTemp(
                    "Device type Capacitor not available in this binary\n");
        }
        INPmakeMod(modname,type,image);
    } else if(strcmp(typename,"sw") == 0) {
.
.  <trecho do programa omitido>
.
    INPmakeMod(modname,type,image);
    } else {
```

```
        type = -1;
        err = (char *)MALLOC(35 + strlen(typename));
        (void)sprintf(err,"unknown model type %s - ignored\n",typename);
    }
    return(err);
}
```

B.7.2 Programa: inppas2.c

Neste arquivo foi acrescentado a associação da letra "A" ao dispositivo linha de transmissão acústica.

```
**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
#include "spice.h"
#include "strext.h"
#include <ctype.h>
#include "ifsim.h"
.
.  <trecho do programa omitido>
.
switch(c) {
        case ' ':   /* blank line (space leading) */
        case '\t':   /* blank line (tab leading) */
            break;
        case 'R':   /* Rname <node> <node> [<val>][<mname>][w=<val>][l=<val>] */
            INP2R(ckt,tab,current);
            break;
        /*  alterado em 22/09/99   */
/* Alberto - LDN/DES/UFPE */
        case 'A':
            INP2A(ckt,tab,current);
            break;
```

```
        case 'C':   /* Cname <node> <node> <val> [IC=<val>] */
            INP2C(ckt,tab,current);
            break;
.
.  <trecho do programa omitido>
.
case 0:
    break;
        default:
            /* the un-implemented device */
            LITERR(" unknown device type - error \n")
            break;
        }
    }
end:
    return;
}
```

B.7.3 Programa: makedefs

B.7.4 Programa: sperror.c

Este programa trata de códigos de erro. Foi acrescentada a mensagem de erro caso haja uma tentativa de análise de polo zero para a linha acústica.

```
**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1985 Thomas L. Quarles
**********/
/*
 *  provide the error message appropriate for the given error code
 */
#include "spice.h"
```

```
#include <stdio.h>
#include "strext.h"
.
.  <trecho do programa omitido>
.
static char *order =        "Unsupported integration order";
static char *method =       "Unsupported integration method";
static char *timestep =     "Timestep too small";
/* static char *xmission = "transmission lines not supported by pole-zero"; */
/*  alterado em 07/04/2000 */
/* Alberto - LDN/DES/UFPE */
static char *xmission = "transmission lines/SAW not supported by pole-zero";
static char *toobig =       "magnitude overflow";
static char *isshort =      "input or output shorted";
.
.  <trecho do programa omitido>
.
#ifdef notdef
    else
(*(SPfrontEnd->IFerror))(ERR_PANIC,nomem,(IFuid *)NULL);
#endif
    return(val);
}
```

Apêndice C

Arquivos acrescentados ao código fonte do SPICE3f4

Neste apêndice estão listados os arquivos fonte do SPICE3f4 que foram acrescentados ao código original, que se fizeram necessários no momento da criação da linha de transmissão acústica. Lembrando que todas as palavras "LTRA" foram substituídas por "SAW".

C.1 Diretório: /spice3f4/src/lib/inp

C.1.1 Programa: inp2a.c

Efetua a leitura dos parâmetros passados pelo dispositivo "A" no arquivo de descrição e chama a subrotina específica..

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1990 Jaijeet S. Roychowdhury
**********/
#include "spice.h"
#include <stdio.h>
#include "ifsim.h"
#include "inpdefs.h"
#include "inpmacs.h"
```

```
#include "fteext.h"
#include "suffix.h"
void
INP2A(ckt,tab,current)
    GENERIC *ckt;
    INPtables *tab;
    card *current;
{
    /* Aname <node> <node> <node> <node> [IC=<val>,<val>,<val>,<val>] */
int type;   /* the type the model says it is */
char *line; /* the part of the current line left to parse */
char *name; /* the resistor's name */
char *nname1;   /* the first node's name */
char *nname2;   /* the second node's name */
char *nname3;   /* the third node's name */
char *nname4;   /* the fourth node's name */
GENERIC *node1; /* the first node's node pointer */
GENERIC *node2; /* the second node's node pointer */
GENERIC *node3; /* the third node's node pointer */
GENERIC *node4; /* the fourth node's node pointer */
int error;      /* error code temporary */
GENERIC *fast;  /* pointer to the actual instance */
int waslead;    /* flag to indicate that funny unlabeled number was found */
double leadval; /* actual value of unlabeled number */
char *model;    /* the name of the model */
INPmodel *thismodel;    /* pointer to model description for user's model */
GENERIC *mdfast;    /* pointer to the actual model */
IFuid uid;      /* uid for default model */
    type = INPtypelook("SAW");
    if(type < 0 ) {
        LITERR("Device type SAW not supported by this binary\n")
        return;
    }
    line = current->line;
```

```
    INPgetTok(&line,&name,1);
    INPinsert(&name,tab);
    INPgetTok(&line,&nname1,1);
    INPtermInsert(ckt,&nname1,tab,&node1);
    INPgetTok(&line,&nname2,1);
    INPtermInsert(ckt,&nname2,tab,&node2);
    INPgetTok(&line,&nname3,1);
    INPtermInsert(ckt,&nname3,tab,&node3);
    INPgetTok(&line,&nname4,1);
    INPtermInsert(ckt,&nname4,tab,&node4);
    INPgetTok(&line,&model,1);
    if( INPlookMod(model) ) {
        /* do nothing for now */
        /* no action required */
    } else {
/*
        nname4 = model;
        INPtermInsert(ckt,&nname4,tab,&node4);
        INPgetTok(&line,&model,1);
*/
    }
    INPinsert(&model,tab);
    current->error = INPgetMod(ckt,model,&thismodel,tab);
    if(thismodel != NULL) {
        if(type != thismodel->INPmodType) {
            LITERR("incorrect model type")
            return;
        }
        mdfast = (thismodel->INPmodfast);
    } else {
        if(!tab->defAmod) {
            /* create default A model */
            IFnewUid(ckt,&uid,(IFuid)NULL,"A",UID_MODEL,(GENERIC**)NULL);
            IFC(newModel,(ckt,type,&(tab->defAmod),uid))
```

```
        }
        mdfast = tab->defAmod;
    }
    IFC(newInstance,(ckt,mdfast,&fast,name))
    IFC(bindNode,(ckt,fast,1,node1))
    IFC(bindNode,(ckt,fast,2,node2))
    IFC(bindNode,(ckt,fast,3,node3))
    IFC(bindNode,(ckt,fast,4,node4))
    PARSECALL((&line,ckt,type,fast,&leadval,&waslead,tab))
}
```

C.2 Diretório: /spice3f4/src/lib/dev/saw

C.2.1 Programa: makedefs

C.2.2 Programa: saw.c

Define a estrutura de dados do modelo SAW. Foi definido também o novo parâmetro "VEL" para a linha de transmissão acústica.

```
/**********
Copyright 1990 Regents of the University of California.  All rights
reserved.
Author: 1990 Jaijeet S. Roychowdhury
**********/
/*
 * This file defines the SAW data structures that are available to the
 * next level(s) up the calling hierarchy
 */
#include "spice.h"
#include <stdio.h>
#include "devdefs.h"
#include "ifsim.h"
```

```
#include "sawdefs.h"
#include "suffix.h"
IFparm SAWpTable[] = { /* parameters */
 IOPAU( "v1", SAW_V1,   IF_REAL   , "Initial voltage at end 1"),
 IOPAU( "v2", SAW_V2,   IF_REAL   , "Initial voltage at end 2"),
 IOPAU( "i1", SAW_I1,   IF_REAL   , "Initial current at end 1"),
 IOPAU( "i2", SAW_I2,   IF_REAL   , "Initial current at end 2"),
 IP("ic", SAW_IC,   IF_REALVEC,"Initial condition vector:v1,i1,v2,i2"),
 OPU("pos_node1", SAW_POS_NODE1,IF_INTEGER,"Positive node of end 1 of t-line"),
 OPU("neg_node1", SAW_NEG_NODE1,IF_INTEGER,"Negative node of end 1 of t.line"),
 OPU("pos_node2", SAW_POS_NODE2,IF_INTEGER,"Positive node of end 2 of t-line"),
 OPU("neg_node2", SAW_NEG_NODE2,IF_INTEGER,"Negative node of end 2 of t-line")
};

IFparm SAWmPTable[] = { /* model parameters */
 IOP( "saw",SAW_MOD_SAW,IF_FLAG,"SAW model"),
 IOPU( "r", SAW_MOD_R,   IF_REAL   , "Resistance per metre"),
 IOPAU( "l", SAW_MOD_L,   IF_REAL   , "Inductance per metre"),
 IOPR( "g", SAW_MOD_G,   IF_REAL   , "Conductance per metre"),
 IOPAU( "c", SAW_MOD_C,   IF_REAL   , "Capacitance per metre"),
 IOPU( "len", SAW_MOD_LEN,   IF_REAL   , "length of line"), /* alteração feita */
 IOPU( "vel",  SAW_MOD_VEL, IF_REAL , "Phase Velocity"),    /*     (Alberto)   */
 OP( "rel", SAW_MOD_RELTOL, IF_REAL, "Rel. rate of change of deriv. for bkpt"),
 OP( "abs", SAW_MOD_ABSTOL, IF_REAL, "Abs. rate of change of deriv. for bkpt"),
 IOPU("nocontrol", SAW_MOD_NOCONTROL, IF_FLAG,"No timestep control"),
 IOPU( "steplimit", SAW_MOD_STEPLIMIT, IF_FLAG,
 "always limit timestep to 0.8*(delay of line)"),
 IOPU( "nosteplimit", SAW_MOD_NOSTEPLIMIT, IF_FLAG,
 "don't always limit timestep to 0.8*(delay of line)"),
 IOPU( "lininterp", SAW_MOD_LININTERP, IF_FLAG, "use linear interpolation"),
 IOPU("quadinterp", SAW_MOD_QUADINTERP, IF_FLAG, "use quadratic interpolation"),
 IOPU("mixedinterp", SAW_MOD_MIXEDINTERP, IF_FLAG,
 "use linear interpolation if quadratic results look unacceptable"),
 IOPU("truncnr", SAW_MOD_TRUNCNR, IF_FLAG,
```

```
 "use N-R iterations for step calculation in SAWtrunc"),
 IOPU( "truncdontcut", SAW_MOD_TRUNCDONTCUT, IF_FLAG,
    "don't limit timestep to keep impulse response calculation errors low"),
 IOPAU( "compactrel", SAW_MOD_STLINEREL, IF_REAL,
 "special reltol for straight line checking"),
 IOPAU( "compactabs", SAW_MOD_STLINEABS, IF_REAL,
 "special abstol for straight line checking")
#ifdef notdef
 IOP( "f",  SAW_MOD_FREQ, IF_REAL   , "Frequency"),
 IOP( "nl", SAW_MOD_NL,   IF_REAL   , "Normalized length at frequency given"),
 IOP("fullcontrol", SAW_MOD_FULLCONTROL, IF_FLAG, "rigorous timestep control"),
 IOP("halfcontrol", SAW_MOD_HALFCONTROL, IF_FLAG,
 "only the current step is considered for timestep control"),
 IOP( "print", SAW_MOD_PRINT, IF_FLAG, "printing of debugging info on"),
 IOP( "noprint", SAW_MOD_NOPRINT, IF_FLAG, "printing of debugging info off"),
 IOP( "ronly", SAW_MOD_RONLY, IF_FLAG, "use special load routines for G=0"),
 IOP( "choprel", SAW_MOD_CHOPREL, IF_REAL,
 "special reltol for truncation of impulse responses"),
 IOP( "chopabs", SAW_MOD_CHOPABS, IF_REAL,
 "special abstol for truncation of impulse responses "),
#endif
};

char *SAWnames[] = {
    "P1+",
    "P1-",
    "P2+",
    "P2-"
};

int SAWnSize = NUMELEMS(SAWnames);
int SAWpTSize = NUMELEMS(SAWpTable);
int SAWmPTSize = NUMELEMS(SAWmPTable);
int SAWiSize = sizeof(SAWinstance);
```

```
int SAWmSize = sizeof(SAWmodel);
```

C.2.3 Programa: sawdefs.h

Arquivo de cabeçalho que define as variáveis de armazenamento dos valores dos parâmetros do dispositivo. Foi acrescentado a definição do novo parâmetro para a velocidade.

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1990 Jaijeet S. Roychowdhury
Re-writen by Alberto Barbosa from UFPE in 2000
**********/
#ifndef SAW
#define SAW
#undef SAWLTEINFO
#undef SAWDEBUG
#include "ifsim.h"
#include "cktdefs.h"
#include "gendefs.h"
#include "complex.h"
/* structures used to describe SAW Device */
/* information used to describe a single instance */
typedef struct sSAWinstance {
    struct sSAWmodel *SAWmodPtr;    /* backpointer to model */
    struct sSAWinstance *SAWnextInstance;
        /* pointer to next instance of current model*/
    IFuid SAWname;      /* pointer to character string naming this instance */
    int SAWstate;  /* not used */
    int SAWposNode1;    /* number of positive node of end 1 of SAW */
    int SAWnegNode1;    /* number of negative node of end 1 of SAW */
    int SAWposNode2;    /* number of positive node of end 2 of SAW */
    int SAWnegNode2;    /* number of negative node of end 2 of SAW */
    int SAWbrEq1;       /* number of branch equation for end 1 of SAW */
    int SAWbrEq2;       /* number of branch equation for end 2 of SAW */
```

```
double SAWinput1;   /* accumulated excitation for port 1 */
double SAWinput2;   /* accumulated excitation for port 2 */
double SAWinitVolt1;    /* initial condition:  voltage on port 1 */
double SAWinitCur1;     /* initial condition:  current at port 1 */
double SAWinitVolt2;    /* initial condition:  voltage on port 2 */
double SAWinitCur2;     /* initial condition:  current at port 2 */
double *SAWv1;     /* past values of v1 */
double *SAWi1;     /* past values of i1 */
double *SAWv2;     /* past values of v2 */
double *SAWi2;     /* past values of i2 */
int SAWinstListSize; /* size of above lists */
double *SAWibr1Ibr1Ptr;     /* pointer to sparse matrix */
double *SAWibr1Ibr2Ptr;     /* pointer to sparse matrix */
double *SAWibr1Pos1Ptr;     /* pointer to sparse matrix */
double *SAWibr1Neg1Ptr;     /* pointer to sparse matrix */
double *SAWibr1Pos2Ptr;     /* pointer to sparse matrix */
double *SAWibr1Neg2Ptr;     /* pointer to sparse matrix */
double *SAWibr2Ibr1Ptr;     /* pointer to sparse matrix */
double *SAWibr2Ibr2Ptr;     /* pointer to sparse matrix */
double *SAWibr2Pos1Ptr;     /* pointer to sparse matrix */
double *SAWibr2Neg1Ptr;     /* pointer to sparse matrix */
double *SAWibr2Pos2Ptr;     /* pointer to sparse matrix */
double *SAWibr2Neg2Ptr;     /* pointer to sparse matrix */
double *SAWneg1Ibr1Ptr;     /* pointer to sparse matrix */
double *SAWneg2Ibr2Ptr;     /* pointer to sparse matrix */
double *SAWpos1Ibr1Ptr;     /* pointer to sparse matrix */
double *SAWpos2Ibr2Ptr;     /* pointer to sparse matrix */
double *SAWpos1Pos1Ptr;     /* pointer to sparse matrix */
double *SAWneg1Neg1Ptr;     /* pointer to sparse matrix */
double *SAWpos2Pos2Ptr;     /* pointer to sparse matrix */
double *SAWneg2Neg2Ptr;     /* pointer to sparse matrix */
unsigned SAWicV1Given : 1;  /* flag to ind. init. voltage at port 1 given */
unsigned SAWicC1Given : 1;  /* flag to ind. init. current at port 1 given */
unsigned SAWicV2Given : 1;  /* flag to ind. init. voltage at port 2 given */
```

```
    unsigned SAWicC2Given : 1;  /* flag to ind. init. current at port 2 given */
} SAWinstance ;
/* per model data */
typedef struct sSAWmodel {       /* model structure for a SAW device */
    int SAWmodType; /* type index of this device type */
    struct sSAWmodel *SAWnextModel; /* pointer to next possible model in
                                     * linked list */
    SAWinstance * SAWinstances; /* pointer to list of instances that have this
                                 * model */
    IFuid SAWmodName;       /* pointer to character string naming this model */
double SAWh1dashFirstVal; /* first needed value of h1dasg at current timepoint */
double SAWh2FirstVal;  /* first needed value of h2 at current timepoint */
double SAWh3dashFirstVal; /* first needed value of h3dash at current timepoint */
/*double *SAWh1dashValues;*/ /* values of h1dash for all previous times */
/*double *SAWh2Values;*/ /* values of h2 for all previous times */
/*double *SAWh3dashValues;*/ /* values of h3dash for all previous times */
/*double SAWh2FirstOthVal;/* needed for LTE calc; but their values */
/*double SAWh3dashFirstOthVal;*//*may depend on the current timepoint*/
/* double *SAWh1dashOthVals;*/ /* these lists of other values are */
/* double *SAWh2OthVals;*/    /* needed for truncation error  */
/* double *SAWh3dashOthVals;*/ /* calculation */
/* the OthVals do not depend on the current
 * timepoint; hence they are set up in SAWaccept.c.
 * They are used in SAWtrunc.c
*/
double SAWh1dashFirstCoeff;
  /* first needed coeff of h1dash for the current timepoint */
double SAWh2FirstCoeff;
/* first needed coeff of h2 for the current timepoint */
double SAWh3dashFirstCoeff;
/* first needed coeff of h3dash for the current timepoint */
    double *SAWh1dashCoeffs; /* list of other coefficients for h1dash */
    double *SAWh2Coeffs;     /* list of other coefficients for h2 */
    double *SAWh3dashCoeffs; /* list of other coefficients for h3dash */
```

```
    int SAWmodelListSize; /* size of above lists */
    double SAWconduct;  /* conductance G  - input */
    double SAWresist;   /* resistance R  - input */
    double SAWinduct;   /* inductance L - input */
    double SAWcapac;    /* capacitance C - input */
    double SAWlength;  /* length l - input */
    /* alterado em 26/11/2001 */
    double SAWvelocity;   /* Phase Velocity - input */
    /*                         */
    double SAWtd;       /* propagation delay T - calculated*/
    double SAWimped;    /* impedance Z - calculated*/
    double SAWadmit;    /* admittance Y - calculated*/
    double SAWalpha;   /* alpha - calculated */
    double SAWbeta;   /* beta - calculated */
    double SAWattenuation; /* e^(-beta T) - calculated */
double SAWcByR;     /* C/R - for the RC line - calculated */
double SAWrclsqr;   /* RCl^2 - for the RC line - calculated */
    double SAWintH1dash;/* \int_0^\inf h'_1(\tau) d \tau - calculated*/
    double SAWintH2;/* \int_0^\inf h_2(\tau) d \tau - calculated*/
    double SAWintH3dash;/* \int_0^\inf h'_3(\tau) d \tau - calculated*/
    double SAWnl;
 /* normalized length - historical significance only*/
    double SAWf;
        /* frequency at which nl is measured - historical significance only*/
double SAWcoshlrootGR; /* cosh(l*sqrt(G*R)), used for DC anal */
double SAWrRsLrGRorG; /* sqrt(R)*sinh(l*sqrt(G*R))/sqrt(G) */
double SAWrGsLrGRorR; /* sqrt(G)*sinh(l*sqrt(G*R))/sqrt(R) */
/*int SAWh1dashIndex;*/ /* index for h1dash that points to the
latest nonzero coefficient in the list */
/*int SAWh2Index;*/  /* ditto for h2 */
/*int SAWh3dashIndex;*/ /* ditto for h3dash */
int SAWauxIndex;    /* auxiliary index for h2 and h3dash */
double SAWstLineReltol;  /* separate reltol for checking st. lines */
double SAWchopReltol;  /* separate reltol for truncation of impulse responses*/
```

```
double SAWstLineAbstol;  /* separate abstol for checking st.  lines */
double SAWchopAbstol;  /* separate abstol for truncation of impulse responses */
    unsigned SAWreltolGiven:1;  /* flag to ind. relative deriv. tol. given */
    unsigned SAWabstolGiven:1;  /* flag to ind. absolute deriv. tol. given */
unsigned SAWtruncNR:1;
/* flag to ind. use N-R iterations for calculating step in SAWtrunc */
unsigned SAWtruncDontCut:1;
/* flag to ind. don't bother about errors in impulse response */
/* calculations due to large steps*/
double SAWmaxSafeStep; /* maximum safe step for impulse response calculations */
    unsigned SAWresistGiven : 1; /* flag to indicate R was specified */
    unsigned SAWconductGiven : 1; /* flag to indicate G was specified */
    unsigned SAWinductGiven : 1; /* flag to indicate L was specified */
    unsigned SAWcapacGiven : 1; /* flag to indicate C was specified */
    unsigned SAWlengthGiven : 1; /* flag to indicate length was specified */
    /*   ALTERADO EM 26/11/2001  */
    unsigned SAWvelocityGiven : 1;
/* flag to indicate Phase Veloc.was specified */
    /*                              */
    unsigned SAWnlGiven : 1;
/* flag to indicate norm length was specified */
int SAWlteConType;
/* indicates whether full control, half control or no control */
int SAWhowToInterp;
/* indicates how to interpolate for delayed timepoint */
unsigned SAWprintFlag: 1;
/* flag to indicate whether debugging output should be printed */
/*unsigned SAWrOnly: 1;*/
/* flag to indicate G=0, use known Bessel integrals for accuracy and speed */
int SAWstepLimit;
/* flag to indicate that the timestep should always be limited to 0.8*SAWtd */
    unsigned SAWfGiven : 1;
/* flag to indicate freq was specified */
    double SAWabstol;       /* absolute deriv. tol. for breakpoint setting */
```

```
    double SAWreltol;       /* relative deriv. tol. for breakpoint setting */
int SAWspecialCase; /* what kind of model (RC, RLC, RL, ...) */
} SAWmodel;
/* device parameters */
#define SAW_MOD_SAW 0
#define SAW_MOD_R 1
#define SAW_MOD_L 2
#define SAW_MOD_G 3
#define SAW_MOD_C 4
#define SAW_MOD_LEN 5
#define SAW_V1 6
#define SAW_I1 7
#define SAW_V2 8
#define SAW_I2 9
#define SAW_IC 10
#define SAW_MOD_RELTOL 11
#define SAW_MOD_ABSTOL 12
#define SAW_POS_NODE1 13
#define SAW_NEG_NODE1 14
#define SAW_POS_NODE2 15
#define SAW_NEG_NODE2 16
#define SAW_INPUT1 17
#define SAW_INPUT2 18
#define SAW_DELAY 19
#define SAW_BR_EQ1 20
#define SAW_BR_EQ2 21
#define SAW_MOD_NL 22
#define SAW_MOD_FREQ 23
#define SAW_MOD_Z0 24
#define SAW_MOD_TD 25
#define SAW_MOD_FULLCONTROL 26
#define SAW_MOD_HALFCONTROL 27
#define SAW_MOD_NOCONTROL 28
#define SAW_MOD_PRINT 29
```

```
#define SAW_MOD_NOPRINT 30
/*
#define SAW_MOD_RONLY 31
*/
#define SAW_MOD_STEPLIMIT 32
#define SAW_MOD_NOSTEPLIMIT 33
#define SAW_MOD_LININTERP 34
#define SAW_MOD_QUADINTERP 35
#define SAW_MOD_MIXEDINTERP 36
#define SAW_MOD_RLC 37
#define SAW_MOD_RC 38
#define SAW_MOD_RG 39
#define SAW_MOD_LC 40
#define SAW_MOD_RL 41
#define SAW_MOD_STLINEREL 42
#define SAW_MOD_STLINEABS 43
#define SAW_MOD_CHOPREL 44
#define SAW_MOD_CHOPABS 45
#define SAW_MOD_TRUNCNR 46
#define SAW_MOD_TRUNCDONTCUT 47
#define SAW_MOD_VEL 48   /* NOVO PARAMETRO VELOCIDADE DE FASE */
/* model parameters */
/* device questions */
/* model questions */
#include "sawext.h"
#endif /*SAW*/
```

C.2.4 Programa: sawmask.c

Este programa refere-se ao acesso aos valores dos parâmetros do modelo. Nele foi acrescentado a referência ao valor do parâmetro "VEL".

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1990 Jaijeet S. Roychowdhury
```

```
**********/
/*
 * This routine sets model parameters for
 * SAW lines in the circuit.
 */
#include "spice.h"
#include <stdio.h>
#include "const.h"
#include "ifsim.h"
#include "util.h"
#include "sawdefs.h"
#include "sperror.h"
#include "suffix.h"
int
SAWmAsk(ckt,inModel,param,value)
    CKTcircuit *ckt;
    GENmodel *inModel;
    int param;
    IFvalue *value;
{
    SAWmodel *mods = (SAWmodel*)inModel;
    switch(param) {
case SAW_MOD_SAW:
    value->iValue = 1;
    break;
        case SAW_MOD_RELTOL:
            value->rValue = mods->SAWreltol;
            break;
        case SAW_MOD_ABSTOL:
            value->rValue = mods->SAWabstol;
            break;
        case SAW_MOD_STLINEREL:
            value->rValue = mods->SAWstLineReltol;
            break;
```

```
        case SAW_MOD_STLINEABS:
            value->rValue = mods->SAWstLineAbstol;
            break;
        case SAW_MOD_CHOPREL:
            value->rValue = mods->SAWchopReltol;
            break;
        case SAW_MOD_CHOPABS:
            value->rValue = mods->SAWchopAbstol;
            break;
case SAW_MOD_TRUNCNR:
value->iValue = mods->SAWtruncNR;
break;
case SAW_MOD_TRUNCDONTCUT:
value->iValue = mods->SAWtruncDontCut;
break;
        case SAW_MOD_R:
            value->rValue = mods->SAWresist;
            break;
/*  alterado em 28/11/2001   */
        case SAW_MOD_VEL:
            value->rValue = mods->SAWvelocity;
            break;
/*                  */
        case SAW_MOD_L:
            value->rValue = mods->SAWinduct;
            break;
        case SAW_MOD_G:
            value->rValue = mods->SAWconduct;
            break;
        case SAW_MOD_C:
            value->rValue = mods->SAWcapac;
            break;
        case SAW_MOD_LEN:
            value->rValue = mods->SAWlength;
```

```
            break;
        case SAW_MOD_NL:
            value->rValue = mods->SAWnl;
            break;
        case SAW_MOD_FREQ:
            value->rValue = mods->SAWf;
            break;
        case SAW_MOD_FULLCONTROL:
            value->iValue = mods->SAWlteConType;
            break;
        case SAW_MOD_HALFCONTROL:
            value->iValue = mods->SAWlteConType;
            break;
        case SAW_MOD_NOCONTROL:
            value->iValue = mods->SAWlteConType;
            break;
        case SAW_MOD_PRINT:
            value->iValue = mods->SAWprintFlag;
            break;
        case SAW_MOD_NOPRINT:
            mods->SAWprintFlag= FALSE;
            break;
/*
        case SAW_MOD_RONLY:
            mods->SAWrOnly= TRUE;
            break;
*/
        case SAW_MOD_STEPLIMIT:
            value->iValue = mods->SAWstepLimit;
            break;
        case SAW_MOD_NOSTEPLIMIT:
            value->iValue = mods->SAWstepLimit;
            break;
        case SAW_MOD_LININTERP:
```

```
            value->iValue = mods->SAWhowToInterp;
            break;
        case SAW_MOD_QUADINTERP:
            value->iValue = mods->SAWhowToInterp;
            break;
        case SAW_MOD_MIXEDINTERP:
            value->iValue = mods->SAWhowToInterp;
            break;
        default:
            return(E_BADPARM);
    }
    return(OK);
}
```

C.2.5 Programa: sawmpar.c

Este programa armazena os valores dos parâmetros do modelo. Nele foi acrscentado o parâmetro "VEL".

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1990 Jaijeet S. Roychowdhury
**********/
/*
 * This routine sets model parameters for
 * SAW lines in the circuit.
 */
#include "spice.h"
#include <stdio.h>
#include "const.h"
#include "ifsim.h"
#include "util.h"
#include "sawdefs.h"
#include "sperror.h"
#include "suffix.h"
```

```
int
SAWmParam(param,value,inModel)
    int param;
    IFvalue *value;
    GENmodel *inModel;
{
    SAWmodel *mods = (SAWmodel*)inModel;
    switch(param) {
case SAW_MOD_SAW:
break;
        case SAW_MOD_RELTOL:
            mods->SAWreltol = value->rValue;
            mods->SAWreltolGiven = TRUE;
            break;
        case SAW_MOD_ABSTOL:
            mods->SAWabstol = value->rValue;
            mods->SAWabstolGiven = TRUE;
            break;
        case SAW_MOD_STLINEREL:
            mods->SAWstLineReltol = value->rValue;
            break;
        case SAW_MOD_STLINEABS:
            mods->SAWstLineAbstol= value->rValue;
            break;
        case SAW_MOD_CHOPREL:
            mods->SAWchopReltol = value->rValue;
            break;
        case SAW_MOD_CHOPABS:
            mods->SAWchopAbstol= value->rValue;
            break;
case SAW_MOD_TRUNCNR:
mods->SAWtruncNR = TRUE;
break;
case SAW_MOD_TRUNCDONTCUT:
```

```
mods->SAWtruncDontCut = TRUE;
break;
        case SAW_MOD_R:
            mods->SAWresist = value->rValue;
            mods->SAWresistGiven = TRUE;
            break;
     /*     ALTERADO EM 28/11/01            */
        case SAW_MOD_VEL:
            mods->SAWvelocity = value->rValue;
            mods->SAWvelocityGiven = TRUE;
            break;
/*                                  */
        case SAW_MOD_L:
            mods->SAWinduct = value->rValue;
            mods->SAWinductGiven = TRUE;
            break;
        case SAW_MOD_G:
            mods->SAWconduct = value->rValue;
            mods->SAWconductGiven = TRUE;
            break;
        case SAW_MOD_C:
            mods->SAWcapac = value->rValue;
            mods->SAWcapacGiven = TRUE;
            break;
        case SAW_MOD_LEN:
            mods->SAWlength = value->rValue;
            mods->SAWlengthGiven = TRUE;
            break;
        case SAW_MOD_NL:
            mods->SAWnl= value->rValue;
            mods->SAWnlGiven = TRUE;
            break;
        case SAW_MOD_FREQ:
            mods->SAWf= value->rValue;
```

```
            mods->SAWfGiven = TRUE;
            break;
        case SAW_MOD_FULLCONTROL:
            mods->SAWlteConType= SAW_MOD_FULLCONTROL;
            break;
        case SAW_MOD_HALFCONTROL:
            mods->SAWlteConType= SAW_MOD_HALFCONTROL;
            break;
        case SAW_MOD_NOCONTROL:
            mods->SAWlteConType= SAW_MOD_NOCONTROL;
            break;
        case SAW_MOD_PRINT:
            mods->SAWprintFlag= TRUE;
            break;
        case SAW_MOD_NOPRINT:
            mods->SAWprintFlag= FALSE;
            break;
/*
        case SAW_MOD_RONLY:
            mods->SAWrOnly= TRUE;
            break;
*/
        case SAW_MOD_STEPLIMIT:
            mods->SAWstepLimit = SAW_MOD_STEPLIMIT;
            break;
        case SAW_MOD_NOSTEPLIMIT:
            mods->SAWstepLimit= SAW_MOD_NOSTEPLIMIT;
            break;
        case SAW_MOD_LININTERP:
            mods->SAWhowToInterp= SAW_MOD_LININTERP;
            break;
        case SAW_MOD_QUADINTERP:
            mods->SAWhowToInterp= SAW_MOD_QUADINTERP;
            break;
```

```
        case SAW_MOD_MIXEDINTERP:
            mods->SAWhowToInterp= SAW_MOD_MIXEDINTERP;
            break;
        default:
            return(E_BADPARM);
    }
    return(OK);
}
```

C.2.6 Programa: sawset.c

Este programa é responsável por estabelecer os valores padrão, caso os parâmetros não sejam informados no arquivo de descrição. Aqui os valores de "L" e "C" foi definidos como função do parâmetro "VEL".

```
/**********
Copyright 1990 Regents of the University of California.  All rights reserved.
Author: 1990 Jaijeet S. Roychowdhury
**********/
#include "spice.h"
#include <stdio.h>
#include "util.h"
#include "smpdefs.h"
#include "cktdefs.h"
#include "sawdefs.h"
#include "sperror.h"
#include "suffix.h"
int
SAWsetup(matrix,inModel,ckt,state)
    register SMPmatrix *matrix;
    GENmodel *inModel;
    register CKTcircuit *ckt;
    int *state;
        /* load the transmission line structure with those pointers needed later
         * for fast matrix loading
```

```
        */
{
    register SAWmodel *model = (SAWmodel *)inModel;
    register SAWinstance *here;
    int error;
    CKTnode *tmp;
    /*  loop through all the transmission line models */
    for( ; model != NULL; model = model->SAWnextModel ) {
            if(!model->SAWnlGiven) {
                model->SAWnl = .25;
            }
            if(!model->SAWfGiven) {
                model->SAWf = 1e9;
            }
            if(!model->SAWreltolGiven) {
                model->SAWreltol = 1;
            }
            if(!model->SAWabstolGiven) {
                model->SAWabstol = 1;
            }
            if(!model->SAWresistGiven) {
                (*(SPfrontEnd->IFerror))(ERR_WARNING,
               "%s: lossy line series resistance not given, assumed zero",
                        &(model->SAWmodName));
model->SAWresist = 0.0;
                /*return(E_BADPARM);*/
            }
if (model->SAWstLineReltol == 0.0)
model->SAWstLineReltol = ckt->CKTreltol;
if (model->SAWstLineAbstol == 0.0)
model->SAWstLineAbstol = ckt->CKTabstol;
/* SAWchopReltol and SAWchopAbstol default zero */
if ((model->SAWhowToInterp != SAW_MOD_LININTERP) &&
(model->SAWhowToInterp != SAW_MOD_QUADINTERP) &&
```

```
(model->SAWhowToInterp != SAW_MOD_MIXEDINTERP)) {
/*
(*(SPfrontEnd->IFerror))(ERR_FATAL,
"%s: have to specify one of lininterp, quadinterp or mixedinterp",
&(model->SAWmodName));
return(E_BADPARM);
*/
if (ckt->CKTtryToCompact) {
model->SAWhowToInterp = SAW_MOD_LININTERP;
(*(SPfrontEnd->IFerror))(ERR_WARNING,
"%s: using linear interpolation because trytocompact option specified",
&(model->SAWmodName));
} else {
model->SAWhowToInterp = SAW_MOD_QUADINTERP;
}
}
if ((model->SAWstepLimit != SAW_MOD_NOSTEPLIMIT))
model->SAWstepLimit = SAW_MOD_STEPLIMIT;
#ifdef notdef
if ((model->SAWprintFlag != SAW_MOD_PRINT))
model->SAWprintFlag = SAW_MOD_NOPRINT;
#endif
if ((model->SAWlteConType != SAW_MOD_FULLCONTROL) &&
(model->SAWlteConType != SAW_MOD_HALFCONTROL))
model->SAWlteConType = SAW_MOD_NOCONTROL;
            if(!model->SAWconductGiven) {
/*
                (*(SPfrontEnd->IFerror))(ERR_WARNING,
           "%s: lossy line parallel conductance not given, assumed zero",
                        &(model->SAWmodName));
*/
model->SAWconduct = 0.0;
                /*return(E_BADPARM);*/
            }
```

```
/*  ALTERADO EM 28/11/2001    */
model->SAWinduct = 1/model->SAWvelocity;
model->SAWcapac = 1/model->SAWvelocity;
/* */
/*  ALTERADO EM 26/11/2001 - FORAM COMENTADAS  AS LINHAS ABAIXO
            if(!model->SAWinductGiven) {
        (*(SPfrontEnd->IFerror))(ERR_WARNING,
           "%s: lossy line series inductance not given, assumed zero",
                        &(model->SAWmodName));
model->SAWinduct = 0.0;
         */
         /*return(E_BADPARM);*/
         /*
            }
            if(!model->SAWcapacGiven) {
                (*(SPfrontEnd->IFerror))(ERR_FATAL,
           "%s: lossy line parallel capacitance not given, assumed zero",
                        &(model->SAWmodName));
model->SAWcapac = 0.0;
           */
                /*return(E_BADPARM);*/
       /*
           }
*/
           if(!model->SAWlengthGiven) {
                (*(SPfrontEnd->IFerror))(ERR_FATAL,
                        "%s: lossy line length must be given",
                        &(model->SAWmodName));
                return(E_BADPARM);
            }
/* ALTERADO EM 26/11/2001 */
            if(!model->SAWvelocityGiven) {
                (*(SPfrontEnd->IFerror))(ERR_FATAL,
```

```
                        "%s: SAW velocity must be given",
                        &(model->SAWmodName));
                return(E_BADPARM);
            }
if ((model->SAWresist == 0) && (model->SAWconduct == 0) &&
(model->SAWcapac != 0) && (model->SAWinduct != 0))
{
model->SAWspecialCase = SAW_MOD_LC;
#ifdef SAWDEBUG
                (*(SPfrontEnd->IFerror))(ERR_INFO,
                        "%s: lossless line",
                        &(model->SAWmodName));
#endif
}
if ((model->SAWresist != 0) && (model->SAWconduct == 0) &&
(model->SAWcapac != 0) && (model->SAWinduct != 0))
{
model->SAWspecialCase = SAW_MOD_RLC;
#ifdef SAWDEBUG
                (*(SPfrontEnd->IFerror))(ERR_INFO,
                        "%s: RLC line",
                        &(model->SAWmodName));
#endif
}
if ((model->SAWresist != 0) && (model->SAWconduct == 0) &&
(model->SAWcapac != 0) && (model->SAWinduct == 0))
{
model->SAWspecialCase = SAW_MOD_RC;
#ifdef SAWDEBUG
                (*(SPfrontEnd->IFerror))(ERR_INFO,
                        "%s: RC line",
                        &(model->SAWmodName));
#endif
}
```

```
if ((model->SAWresist != 0) && (model->SAWconduct == 0) &&
(model->SAWcapac == 0) && (model->SAWinduct != 0))
{
model->SAWspecialCase = SAW_MOD_RL;
            (*(SPfrontEnd->IFerror))(ERR_FATAL,
                        "%s: RL line not supported yet",
                        &(model->SAWmodName));
return(E_BADPARM);
#ifdef SAWDEBUG
#endif
}
if ((model->SAWresist != 0) && (model->SAWconduct != 0) &&
(model->SAWcapac == 0) && (model->SAWinduct == 0))
{
model->SAWspecialCase = SAW_MOD_RG;
#ifdef SAWDEBUG
            (*(SPfrontEnd->IFerror))(ERR_INFO,
                        "%s: RG line",
                        &(model->SAWmodName));
#endif
}
if ((model->SAWconduct != 0) && ( (model->SAWcapac != 0) ||
(model->SAWinduct != 0)))
{
model->SAWspecialCase = SAW_MOD_SAW;
            (*(SPfrontEnd->IFerror))(ERR_FATAL,
                     "%s: Nonzero G (except RG) line not supported yet",
                        &(model->SAWmodName));
return(E_BADPARM);
#ifdef SAWDEBUG
#endif
}
if ((model->SAWresist == 0.0? 0:1) + (model->SAWconduct
== 0.0? 0:1) + (model->SAWinduct == 0.0?0:1) +
```

```
(model->SAWcapac == 0.0? 0:1) <= 1) {
(*(SPfrontEnd->IFerror))(ERR_FATAL,
                "%s: At least two of R,L,G,C must be specified and nonzero",
&(model->SAWmodName));
return(E_BADPARM);
}
        /* loop through all the instances of the model */
        for (here = model->SAWinstances; here != NULL ;
                here=here->SAWnextInstance) {
            if(here->SAWbrEq1==0) {
                error = CKTmkVolt(ckt,&tmp,here->SAWname,"i1");
                if(error) return(error);
                here->SAWbrEq1 = tmp->number;
            }
            if(here->SAWbrEq2==0) {
                error = CKTmkVolt(ckt,&tmp,here->SAWname,"i2");
                if(error) return(error);
                here->SAWbrEq2 = tmp->number;
            }
/* macro to make elements with built in test for out of memory */
#define TSTALLOC(ptr,first,second) \
if((here->ptr = SMPmakeElt(matrix,here->first,here->second))==(double *)NULL){\
    return(E_NOMEM);\
}
            TSTALLOC(SAWibr1Pos1Ptr, SAWbrEq1, SAWposNode1)
            TSTALLOC(SAWibr1Neg1Ptr, SAWbrEq1, SAWnegNode1)
            TSTALLOC(SAWibr1Pos2Ptr, SAWbrEq1, SAWposNode2)
            TSTALLOC(SAWibr1Neg2Ptr, SAWbrEq1, SAWnegNode2)
            TSTALLOC(SAWibr1Ibr1Ptr, SAWbrEq1, SAWbrEq1)
            TSTALLOC(SAWibr1Ibr2Ptr, SAWbrEq1, SAWbrEq2)
            TSTALLOC(SAWibr2Pos1Ptr, SAWbrEq2, SAWposNode1)
            TSTALLOC(SAWibr2Neg1Ptr, SAWbrEq2, SAWnegNode1)
            TSTALLOC(SAWibr2Pos2Ptr, SAWbrEq2, SAWposNode2)
            TSTALLOC(SAWibr2Neg2Ptr, SAWbrEq2, SAWnegNode2)
```

```
            TSTALLOC(SAWibr2Ibr1Ptr, SAWbrEq2, SAWbrEq1)
            TSTALLOC(SAWibr2Ibr2Ptr, SAWbrEq2, SAWbrEq2)
            TSTALLOC(SAWpos1Ibr1Ptr, SAWposNode1, SAWbrEq1)
            TSTALLOC(SAWneg1Ibr1Ptr, SAWnegNode1, SAWbrEq1)
            TSTALLOC(SAWpos2Ibr2Ptr, SAWposNode2, SAWbrEq2)
            TSTALLOC(SAWneg2Ibr2Ptr, SAWnegNode2, SAWbrEq2)
/* the following are done so that SMPpreOrder does not
 * screw up on occasion - for example, when one end
 * of the lossy line is hanging
 */
TSTALLOC(SAWpos1Pos1Ptr, SAWposNode1, SAWposNode1)
TSTALLOC(SAWneg1Neg1Ptr, SAWnegNode1, SAWnegNode1)
TSTALLOC(SAWpos2Pos2Ptr, SAWposNode2, SAWposNode2)
TSTALLOC(SAWneg2Neg2Ptr, SAWnegNode2, SAWnegNode2)
        }
    }
    return(OK);
}
int
SAWunsetup(inModel,ckt)
    GENmodel *inModel;
    CKTcircuit *ckt;
{
#ifndef HAS_BATCHSIM
    SAWmodel *model;
    SAWinstance *here;
    for (model = (SAWmodel *)inModel; model != NULL;
    model = model->SAWnextModel)
    {
        for (here = model->SAWinstances; here != NULL;
                here=here->SAWnextInstance)
{
    if (here->SAWbrEq1) {
CKTdltNNum(ckt, (GENERIC *) here->SAWbrEq1);
```

```
here->SAWbrEq1 = 0;
    }
    if (here->SAWbrEq2) {
CKTdltNNum(ckt, (GENERIC *) here->SAWbrEq2);
here->SAWbrEq2 = 0;
    }
}
    }
#endif
    return OK;
}
```

C.2.7 Programa: sawacct.c

Este programa refere-se a análise AC do dispositivo. Neste programa e nos seguintes, apenas a palvra "LTRA" foi substituída por "SAW", por este motivo a exposição dos códigos não se faz necessária.

C.2.8 Programa: sawacld.c

Este programa também refere-se a análise AC do dispositivo.

C.2.9 Programa: sawask.c

Neste programa são armazenados em variáveis os valores dos parâmetros do dispositivo.

C.2.10 Programa: sawdel.c

Arquivo disponível para futuras implementações, é responsável por eliminar o dispositivo da simulação.

C.2.11 Programa: sawdest.c

Este programa é responsável por liberar memória e fazer referência a um dispositivo seguinte quando um for eliminado.

C.2.12 Programa: sawext.h

Arquivo de cabeçalho que define as funções utilizadas pelo dispositivo.

C.2.13 Programa: sawitf.h

Arquivo de cabeçalho que define as estruturas dos ponteiros.

C.2.14 Programa: sawload.c

Este programa se refere a análise DC e do transitório do dispositivo.

C.2.15 Programa: sawmdel.c

Arquivo disponível para futuras implementações, é responsável por eliminar o modelo utilizado na simulação.

C.2.16 Programa: sawmisc.c

Este arquivo contém várias funções específicas do dispositivo, que descrevem o seu comportamento.

C.2.17 Programa: sawpar.c

Este programa armazena os valores dos parâmetros do dispositivo.

C.2.18 Programa: sawtemp.c

Este programa trata da temperatura do circuito.

C.2.19 Programa: sawtrun.c

Este programa controla o passo de iteração para diminuir o erro.

Bibliografia

[1] A. B. Bhattacharyya and Swati Majumdar. SPICE Simulation of Surface Acoustic Wave Interdigital Transducers. *IEEE transactions on Ultrasonics, Ferroelectrics and frequency Control*, 42(4):784–786, Julho 1995.

[2] Mitsutaka Hikita, Toyoji Tabuchi, Yoshikatsu Ishida, Kazuhito Kurosawa, and Kunihiro Hamada. SAW Integrated Modules for 800-MHz Cellular Radio Portable Telephones with New Frequency Allocations. *IEEE transactions on Ultrasonics, Ferroelecctrics and frequency Control*, 36(5):531–539, Setembro 1989.

[3] H. Wohltjen and D. S. Ballantine. Surface Acoustic Wave Devices for Chemical Analysis. *Analytical Chemistry*, 61:704A.

[4] W. R. Smith, H. M. Gerard, J. H. Collins, T. M. Reeder, and H. J. Shaw. Analysis of Interdigital Surface Wave Transducers by use of an Equivalent Circuit Model. *IEEE Trans. Microwave Theory Tech.*, MTT-17:856–864, 1969.

[5] Alf Püttmer, Peter Hauptmann, Ralf Lucklum, Olaf Krause, and Bernd Henning. SPICE Model for Lossy Piezoceramic Transducers. *IEEE transactions on Ultrasonics, Ferroelecctrics and frequency Control*, 44(1), Janeiro 1997.

[6] Michel Feldmann and Jeannine Hénaff. *Surface Acoustic Waves for Signal Processing*. Artech House, 1989.

[7] R. Krimholtz, D. A. Leedom, and G. L. Matthaei. New Equivalent Circuits for Elementary Piezoelectric Transducers. *Electronics Letters*, 6(13), June 1970.

[8] R. F. Milson and M. Redwood. Interdigital Piezoelectric Rayleigh Wave Transducer: An Improved Equivalent Circuit. *Electron, Lett.*, 7(9), 1971.

[9] Steven A. Morris and Chriswell G. Hutchens. Implementation of Masont's Model on Circuit Analysis Programs. *IEEE Transaction on Ultrasonics, Ferroelectrics, and Frequency Control*, UFCC-33(3), May 1986.

[10] Junji Wakita Yasuo Cho and Nozomu Miyagawa. Nonlinear Equivalent Circuit Model Analysis of Acoustic Devices and Propagation of Surface Acoustic Wave. *Jpn. J. Appl. Phys.*, 32(5B), May 1993.

[11] Kiyoshi Nakamura and Kazuhiro Hirota. Considerations on SAW Coupled-Mode Equations and Equivalent Circuit Representation of Interdigital Transducers. *Ultrasonics Symposium*, pages 189–193, 1992.

[12] W. Marshall Leach Jr. Controlled-Source Analogous Circuits and SPICE Models for Piezoelectric Transducers. *IEEE transactions on Ultrasonics, Ferroelecctrics and frequency Control*, 41(1), Janeiro 1994.

[13] Ivan José de Albuquerque. *Teoria e Problemas de Linhas de Transmissão.* McGraw-Hill do Brasil Ltda, 1972.

[14] Thomas L. Quarles. *Adding Devices to SPICE3c1 - Memorandum UCB/ERL M89/45.* College of Engineering, Berkeley, 1989.

[15] Thomas L. Quarles. *The SPICE3c1 Implementation Guide - Memorandum UCB/ERL M89/44.* College of Engineering, Berkeley, 1989.

[16] Kiyoshi Nakamura. A Simple Equivalent Circuit for Interdigital Transducers Based on the Coupled-Mode Approach. *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, 40(6):763–767, Novembro 1993.

[17] Dong-Pei Chen and Herman A. Haus. Analysis of Metal-Strip SAW Gratings and Transducers. *IEEE Transactions on Sonics and Ultrasonics*, SU-32(3):395–408, Maio 1985.

[18] Charles Elachi. Waves in active and passive periodic structures: A review. In *Proceeding of the IEEE*, volume 40, pages 763–767, Dezembro 1976.

[19] Y. Koyamada, S. Yoshikawa, and F. Ishihara. Analysis of SAW resonators using long IDT's and their applications. *Trans.IECE Japan*, J60-A:805–812, 1977.

[20] Fernando Flammarion Vasconcelos. *Linhas de Transmissão e Circuitos*. Guanabara Dois, 1980.

[21] Alberto M. Barbosa and Edval J. P. Santos. SPICE Macromodel for SAW. *I Student Forum on Microelectronics Proceedings*, Setembro 2001.

[22] Hank Wohltjen. Mechanism of Operation and Design Considerations for Surface Acoustic Wave Device Vapour Sensors. *Sensors ans Actuators*, 5:307–325, 1984.

[23] Bill Drafts. Acoustic Wave Technology Sensors. *Microsensor systems Inc., a Sawtek Company*.

[24] D. Cullen and T. Reeder. Measurement of SAW Velocity Versus Strain for YX and ST Quartz. *Proc Ultrasonics Symposium*, pages 519–522, 1975.

[25] W. Bowers, R. Chuan, and T. Duong. A 200 MHz Surface Acoustic Wave Resonator Mass Microbalance. *Re Sci Instrum*, 62:1624–1629, 1991.

[26] H. Wohltjen and R. Dessy. Surface Acoustic Wave Probe for Chemical Analysis I. Introduction and Instrument Design. *Ana Chem*, 51:1458–1475, 1979.

[27] G. Kovacs. and M. Venema. Theoretical Comparison of Sensitivities of Acoustic Shear Wave Modes for Biochemical Sensing in Liquids. *Appl Phys Lett*, 61(6), 1992.

[28] Alberto M. Barbosa and Edval J. P. Santos. Compact SPICE Macromodel for SAW Based Smart Sensor Design. *Anais do SBMicro 2002, editores N. I. Morimoto, R. P. Ribas, P. Verdonck, 278-284.*, The Electrochemical Society, Inc. Pennington (2002).

[29] Edval J. P. Santos and Alberto de Moraes Barbosa. Novo Modelo SPICE para Dispositvos e Sensores OAS. *Instrumentation, Systems, and Automation Society Show*, South America, São Paulo (2002).

[30] Spice3f User's Manual.

[31] URL: http://nina.ecse.rpi.edu/shur/Ch1/tsld007.htm.

[32] URL: http://nina.ecse.rpi.edu/shur/Ch1/tsld008.htm.

[33] Leslie Lamport. *LaTeX User's Guide & Reference Manual.* Addison-Wesley Publishing Company, 1986.

[34] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion.* Addison-Wesley Publishing Company, 1994.