UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA E DE COMPUTAÇÃO
DEPARTAMENTO DE MICROONDAS E ÓPTICA

SOASIM: SIMULADOR DO CONTROLE DINÂMICO DE GANHO EM AMPLIFICADORES ÓPTICOS A SEMICONDUTOR, INCLUINDO VALIDAÇÃO EXPERIMENTAL E APLICAÇÕES

Candidato: Cristiano de Mello Gallep

Orientador: Prof. Dr. Evandro Conforti

Banca examinadora:

Prof. Dr. Rui Fragassi Souza
DMO - FEEC - Unicamp

Prof. Dr. Carlos C. Brito Cruz
DEQ - IFGW - Unica mp

Esta dissertação é requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica, na área de concentração de Eletrônica e Comunicações, do Departamento de Microonda e Óptica, da Faculdade de Engenharia Elétrica e de Computação.

05/08/1999

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

G136s

Gallep, Cristiano de Mello
SOASim: simulador do controle dinâmico de ganho em amplificadores ópticos a semicondutor, incluindo validação experimental e aplicações / Cristiano de Mello Gallep.--Campinas, SP: [s.n.], 1999.

Orientador: Evandro Conforti
Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Dispositivos optoeletrônicos. 2. Simulação (Computadores). I. Conforti, Evandro. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Resumo – Apresenta-se o *SOASim*, programa de simulação para amplificadores ópticos a semicondutor (*SOA*) com controle dinâmico de ganho. Tal programa foi escrito em linguagem C++ utilizando-se de ambiente com interface amigável. São descritos o modelamento matemático utilizado e sua implementação computacional. Resultados de simulação referentes a cascatas de SOAs operados em regime de saturação profunda, com modulação da corrente injetada no SOA e a operação com controle dinâmico de ganho são apresentados em conjunto com resultados experimentais correspondentes, observando-se boa concordância. Resultados exclusivamente teóricos também são apresentados, mostrando as capacidades de regeneração/apagamento do sinal óptico desse tipo de configuração do SOA. Devido, porém, ao tempo de vida finito dos portadores elétricos, essa funções ficam limitadas a taxas médias de transmissão (~1Gb/s).

Abstract *- The SOASim, a semiconductor optical amplifier (SOA) with feed-forward gain (FFG) control simulation program, is presented. Such program was written in C++ computer language, using an environment with friendly interfaces. The mathematical model and its implementation are described. Simulation results of deep-saturated cascaded SOAs, with modulated injected current and operations with FFG control are presented, with good agreement with their respective experimental results. Exclusive theoretical results are presented too, showing the capabilities of optical signal regeneration/erasing with this kind of SOA configuration. However, the finite electronic carrier lifetime limits those functions to middle bit rates (~1 Gb/s).*

Agentes finaciadores: CAPES
MCT - Pronex

Agradecimentos: ao infinito Deus,
aos meus pais e mães,
ao meu orientador,

Muito Obrigado!

CONTEÚDO:

CAPÍTULO 1 - Introdução Geral

1.1 - Motivação

A transmissão de informação através de feixes luminosos guiados por fibras ópticas é, atualmente, usual em muitas redes de telecomunicação, com o processamento do sinal sendo feito eletronicamente após conversão optoeletrônica. É natural, devido ao grande sucesso da transmissão óptica, que se investigue a possibilidade de se realizarem processos necessários como demultiplexação, recuperação de relógio, regeneração e roteamento inteiramente no domínio óptico, eliminando a necessidade de conversão para o domínio elétrico e vice-versa. A utilização do processamento totalmente óptico da informação poderia, eventualmente, eliminar o gargalo atual das redes por fibras ópticas o qual se encontra nos nós eletrônicos de processamento.

Demonstrações de processamento com grande sucesso utilizando a fibra de sílica como meio não linear de processamento em configurações interferométricas foram apresentadas durante a década de 1980, pois a resposta não-linear da sílica atinge regime de fentosegundos. Porém, a dependência do seu índice de refração com a intensidade do sinal é bem pequena, requerendo então grandes potências e grandes comprimentos de fibra para realizar a diferença de fase necessária ao processamento. Por essa razão, o foco da pesquisa científica atual é encontrar novas formas de processamento óptico do sinal. Nesse contexto, o amplificador óptico a semicondutor (*SOA - semiconductor optical amplifier*) é elemento promissor para substituir a fibra de sílica em circuitos totalmente ópticos, devido às características da sua dinâmica de portadores, possibilidade de compactação, grande largura de banda e integração com outros dispositivos [**1, 2**].

Atualmente, porém, os *SOA*s disponíveis no mercado são muito caros, fazendo com que a utilização de ferramentas computacionais de simulação sejam essenciais para a diminuição dos custos de desenvolvimento assim como para a otimização da estrutura física de sistemas de alta velocidade, baseados em amplificadores ópticos a semicondutor. Essa é a principal motivação desta tese de mestrado, onde é apresentado o ***SOASim*** [**3**], programa de simulação para amplificadores ópticos a semicondutor com controle dinâmico de ganho, construído em linguagem C++ a partir do trabalho de mestrado do colega Juliano P. B. Custódio.

---

1 - R. J. Manning, A. D. Ellis, A. J. Poustie e K. J. Blew - *"Semiconductor laser amplifier for ultrafast all-optical signal processing"* - J. Optical Society of America B, Vol. 14, N° 11, pag. 3204-3216, nov./1997.

2 - Y. Shibata, Y. Yamada, K. Habara e N. Yoshimoto - *"Semiconductor laser diode optical amplifiers/gates in photonic packet switching"* - J. Lightwave Technology, Vol. 16, N° 12, pag. 2228-2235, dez./1998.

3 - C. M. Gallep, J. P. B. Custódio, A. C. Bordonalli e E. Conforti - *"SOASim: a simulator for semiconductor optical amplifier with feed gain control"* - Anais do *IMOC'99* (*Intenational Microwave and Optoelectronics Conference* -1999), reproduzido no APÊNDICE B.

1.2 - O AMPLIFICADOR ÓPTICO A SEMICONDUTOR

Um amplificador óptico a semicondutor é construído a partir de um laser, polarizando-o abaixo do limiar (tipo *FP - Fabry-Perot*) ou modificando-o de maneira que passe a ter baixa reflexão em suas facetas (tipo *TW - Traveling-Wave*), classificações que serão detalhadas mais adiante. É um dispositivo que promete ser muito utilizado em sistemas ópticos [4], como amplificador e processador de sinais ópticos. Possui grande banda de amplificação (~50 nm) e pode ser construído de maneira a operar em uma das três janelas de comunicação óptica [5]. Nesse contexto, concorre diretamente com os amplificadores à fibra dopada com Érbio (*EDFA - Erbium doped fiber amplifiers*), que operam apenas na terceira janela (1550 nm).

Os SOAs podem ser classificados basicamente em dois tipos, como já mencionado anteriormente: o Fabry-Perot (*FP-SOA*) e o de Onda Caminhante (*TW-SOA*) [5]. O *FP-SOA* é formado por um laser polarizado abaixo do seu limiar de oscilação. O sinal acoplado ao dispositivo é amplificado nas sucessivas passagens pela cavidade, já que as facetas do laser agem como espelhos semi-refletivos (~ 32%), formando uma cavidade ressonante tipo Fabry-Perot. Seu fator de amplificação é dado por:

$$G_{FP} = \frac{(1-R_1)(1-R_2)G(\nu)}{\left(1-G\sqrt{R_1R_2}\right)^2 + 4G\sqrt{R_1R_2}\,sen\left[\pi(\nu-\nu_m)/\Delta\nu_L\right]} \qquad (1.2.1)$$

onde R_1 e R_2 são as refletividades das facetas, ν_m é a freqüência de ressonância da cavidade, $\Delta\nu_L$ é o espaçamento entre modos longitudinais e G é o ganho de uma única passagem pela cavidade, o que ocorre idealmente para o amplificador TW, desprezando-se a saturação. De fato, se $R_1 = R_2 = 0$ tem-se $G_{FP}=G$. Tipicamente, $G(R_1 R_2)^{0,5}$ é próximo de 1 e $\Delta\nu_L$ da ordem de 100 GHz, o que resulta numa banda de 3dB menor que 10GHz para o amplificador FP. Essa banda estreita faz com que esse tipo de amplificador não tenha muita utilidade para aplicações em sistemas ópticos, sendo utilizado em processamento de sinais. Outra limitação do *FP-SOA* é a potência de operação baixa, limitada pelo limiar de oscilação do laser. Uma visão simplificada do funcionamento e do espectro de emissão do *FP-SOA* é vista na Figura 1.2.1, onde *ASE* (*amplified spontaneous emission*) é o ruído de emissão espontânea amplificada.

O *TW-SOA*, também denominado Onda Caminhante (OC), é obtido através da minimização da realimentação interna com um recobrimento das facetas por material dielétrico anti-refletivo (AR), de modo que o sinal óptico atravesse o dispositivo uma única vez, sendo amplificado e emitido pela outra extremidade. Para que o laser funcione como *TW-SOA*, as refletividades devem ser extremamente pequenas: menores que 0,1 %. A refletividade mínima depende do ganho do amplificador, portanto para manter a razão

---

4 - M. J. O'Mahony - *"Semiconductor laser optical amplifiers for use in future fiber systems"* - J. Lightwave Tech., Vol. 6, N°. 4, abril/1988.

5 - G. P. Agrawal - *"Fiber-Optic Comunication Systens"* - *Chapter 3: Optical Sources and Transmiters , Chapter 8: Optical Amplifiers* - John Wiley & Sons, Inc - 1992.

$\Delta G = G_{FP}^{max} / G_{FP}^{min} = \left(1 + G\sqrt{R_1 R_2} / 1 - G\sqrt{R_1 R_2}\right)^2$ menor que 2, as refletividades das facetas devem ser tais que $G(R_1 R_2)^{1/2}<0,17$. Para um ganho de 30 dB, por exemplo, deve-se obter $(R_1 R_2)^{1/2}< 1,7\text{x}10^{-4}$.

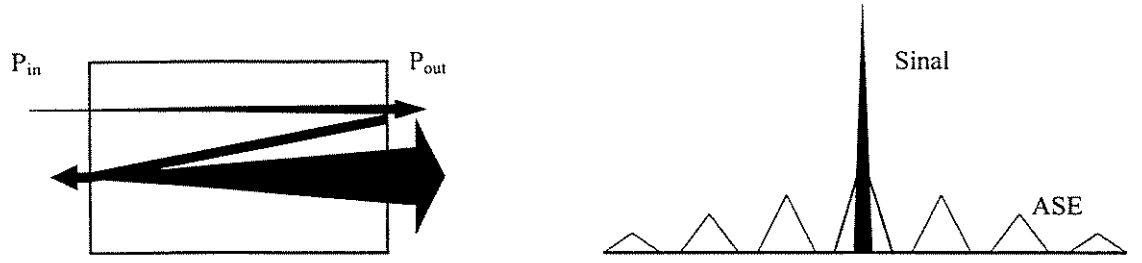

Figura 1.2.1: características do SOA-FP: amplificação e espectro de emissão

Esse dispositivo pode ser polarizado com correntes altas sem apresentar oscilação, permitindo grande coeficiente de ganho, alem de banda óptica larga (~ 50nm) limitada apenas pela curva de ganho do material, tendo maior atrativo para utilização em sistemas ópticos. A Figura 1.2.2 ilustra simplificadamente o mecanismo de amplificação e o espectro do TW.

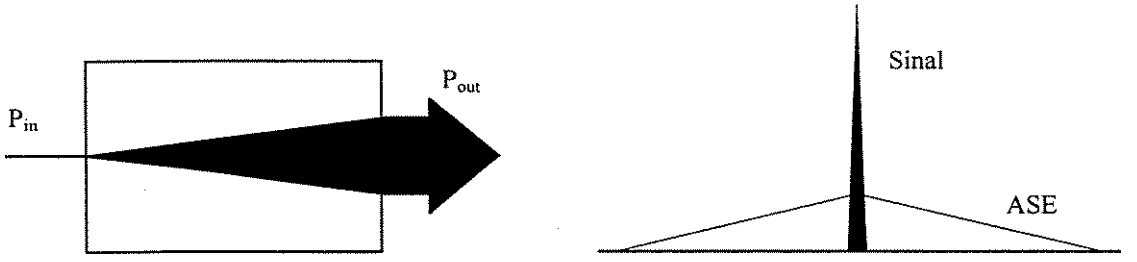

Figura 1.2.2: características do SOA-TW: amplificação e espectro de emissão

O fator de amplificação do SOA foi dado na eq. (1.2.1). Sua dependência em freqüência é resultado de G como função de ν, quando a condição $G(R_1 R_2)^{1/2}<0,17$ é satisfeita. Uma medida de ganho, então, apresentará pequenas oscilações devido às refletividades residuais das facetas.

Outro parâmetro importante é a *figura de ruído* de um SOA: alta, geralmente maior que 3 dB (tipicamente 5-7 dB). A maior contribuição se deve à emissão espontânea, que depende do fator de inversão da população de portadores elétricos. Outros fatores que contribuem para a alta figura de ruído são as perdas internas, como absorção de portadores ópticos, ou perdas por espalhamento. Abaixo relatamos uma expressão para F_n **[5]**:

$$F_n = 2\left(\frac{N}{N - N_0}\right)\left(\frac{g}{g - \alpha_{\text{int}}}\right) \qquad (1.2.2)$$

Para altos ganhos ($N>>N_0$) nota-se que F_n atinge seu valor mínimo de 3 dB ($F_n=2$).

Uma característica muito indesejável dos SOA é sua *sensitividade à polarização*: devido a diferença do fator de confinamento Γ e do ganho diferencial σ_e para os modos TE e TM, o ganho pode diferir de até 5 a 8 dB para as duas polarizações. Uma forma de reduzir esse efeito seria a construção de dispositivo com dimensões de largura e espessura da região ativa bem pequenas e próximas, requerendo baixa tolerância na fabricação. Outra forma é a utilização de SOAs construídos com estruturas de poços quânticos múltiplos com camadas tensionadas, desenvolvidas recentemente [**6**], que conseguem dependência com polarização bem baixa, da ordem de 0,3 dB.

Outra característica de forte evidência nos SOA são as *não-linearidades*. Principalmente em sistemas multicanal FDM (frequency division multiplexing) e WDM (wavelength division multiplexing), o comportamento não-linear dos SOA deve ser considerado quando operam perto da saturação e com canais sem espaçamento adequado. A combinação entre várias componentes do sinal de entrada resulta em saturação de ganho, interferência por saturação induzida (*saturation-induced 'SI' crostalk*) e por modulação induzida da densidade de portadores (*carrier- modulation-induced - CMI*) e inter-modulação [**7**].

A saturação da amplificação de um canal devido a presença de outros canais degrada a relação sinal-ruído e ocorre independente da separação entre canais. Ocorre principalmente em modulação tipo PSK (phase-shift keying) ou FSK (frequency-shift keying) e pode ser evitada operando-se o amplificador longe da saturação.

Já as intermodulações ocorrem em todos os sistemas multicanal, independente da modulação, e deve-se ao mecanismo já bem estudado da *'mistura de quatro ondas'* (four-wave mixing)[**8**]. Quando dois pulsos de freqüência ω_0 e $\omega_0+\Omega$ são injetados num meio de ganho de um SOA ocorre uma modulação da densidade de portadores na freqüência de batimento Ω se a intensidade dos sinais de entrada é alta o suficiente para induzir saturação. Como conseqüência do desvio do número de portadores aparece uma polarização não-linear do meio da forma [**9**]:

$$P_{NL} = \varepsilon_0 \frac{\partial \chi}{\partial N} \delta N \cdot E \qquad (1.2.3)$$

onde ε_0 é a constante dielétrica do vácuo, χ é a susceptibilidade do meio, N é a densidade de portadores, δN é o desvio de N e E é o campo elétrico total. Assim, δN oscila com freqüência Ω e interage com os campos $E(\omega_0)$ e $E(\omega_0+\Omega)$, resultando em novas componentes nas freqüências harmônicas, como por exemplo $\omega_0-\Omega$ e $\omega_0+2\Omega$..

---

6 - T. Ito, N. Yoshimoto, K. Magari e H. Sugiura - *"Wide-band polarization-independent tensile-stained InGaAs MQW-SOA gate"* - IEEE Photonics Lett., Vol. 10, No. 5, maio/1998.

7 - T. E. Darcie e R. M. Jopson - *"Nonlinear Interactions in Optical Amplifiers for Multifrequency Lightwave Systems"* - Eletronics Letters, Vol. 24, No. 10, maio/1988.

8 - G. P. Agrawal - *"Four-Wave Mixing and Phase Conjugation in Semiconductor Laser Media"* - Optics Letters, Vol. 12, No. 4, abril/1987.

9 - K. Inoue, T. Mukai e T. Saitoh - *"Nearly Degenerated Four-Wave Mixing in a Traveling-Wave Semiconductor Laser Amplifier"* - Applied Physics Letters, Vol. 51, No. 14, outubro/1987

Para imaginarmos a amplitude dos problemas decorrentes desse mecanismo em sistemas multicanal, imaginemos agora a interação entre três sinais nas freqüências ω_1, ω_2 e ω_3 com a modulação da densidade de portadores nas freqüências de batimento Ω_1 (ω_2-ω_1), Ω_2 (ω_3-ω_2) e Ω_3 (ω_3-ω_1) [7]. Como resultado, temos o aparecimento de mais nove freqüências, seis a partir dos produtos entre dois sinais e mais três a partir dos produtos entre três sinais, além da alteração da amplitude e fase dos sinais originais devido à interferência coerente (vide Figura 1.2.3, onde os produtos gerados $\omega_i + \omega_j - \omega_k$ são notados de forma abreviada ij-k).

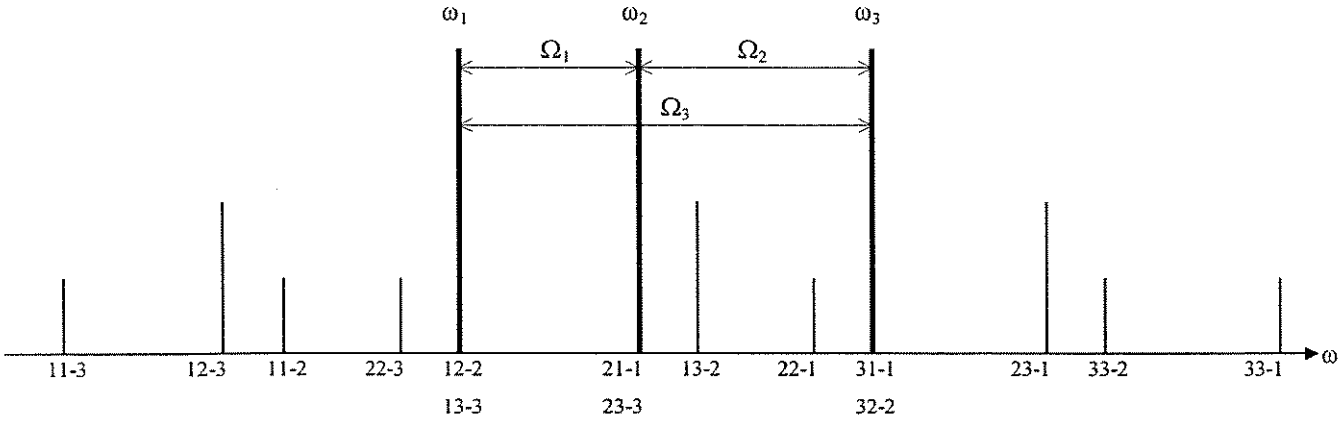

Figura 1.2.3: amplificação não-linear com os produtos gerados pela interação de ω_1, ω_2 e ω_3 com Ω_1, Ω_2 e Ω_3

Uma forma de se obter a operação linear do SOA mesmo em níveis de potência de entrada altos é através da adição de uma componente, dependente do sinal óptico de entrada, à corrente de polarização do SOA, mantendo-se então o ganho óptico praticamente constante, conseguindo-se redução de 10 a 20 dB na distorção devida à intermodulação [**10**]. A partir dessa idéia de inserção, na corrente polarização do SOA, de uma componente proporcional ao sinal óptico de entrada é que desenvolveu-se o trabalho dessa tese de mestrado.

Outra solução para operação linear do SOA em sistemas multicanal é a utilização da injeção na região ativa de um sinal óptico de bombeio com comprimento do onda menor (maior energia) que o das portadoras de informação, caracterizando o chamado SOA de ganho grampeado (*Gain-clamped SOA*) [**11**]. Nesse tipo de SOA a modulação da densidade de portadores elétricos é minimizada, pois o sinal de bombeio fornece a inversão de população necessária para suprimir a queima de portadores excessiva devido à amplificação de múltiplos canais, fazendo o ganho permanecer praticamente constante.

---

10 - T. E. Darcie , R. M. Jopson e A. A. M. Saleh - *"Compensation of Nonlinearity in Semiconductor Optical Amplifiers"* - Eletronics Letters, Vol. 24, No. 24, julho/1988.

11 - L. F. Tiemeijer, G. N. Hoven, P. J. A. Thijs, T. Dongen, J. J. M. Binsma e E J. Jansen - *"1310-nm DBR-Type MQW Gain-clamped Semiconductor optical amplifiers with AM-CATV-Grade linearity"* - IEEE Photonics Lett., Vol. 8, N°. 11, nov./1996.

1.3 - SOA NO PROCESSAMENTO TOTALMENTE ÓPTICO DE SINAIS

O amplificador óptico a semicondutor é um dispositivo promissor para a implementação das futuras redes ópticas [4]. Segundo essa referência, pode ser utilizado tanto em regime de operação linear, como repetidor não-regenerativo e pré-amplificador de recepção, como em regime não-linear no processamento óptico de sinais. Pode ainda, sob certas condições específicas, trabalhar como elemento bi-estável, como reconstrutor de pulsos e chave óptica.

Pesquisas recentes procuram desenvolver os aspectos não-lineares do SOA para implementação de multiplexação no tempo [12] e conversão de comprimento de onda [13] para altas taxas em sistemas totalmente ópticos.

Nas operações de chaveamento, a dinâmica de portadores do SOA permite que ele seja usado na recuperação de relógio (*clock recovery*), demultiplexação temporal e regeneração [1]. Para operar em altas taxas de transmissão, comumente utilizam-se montagens interferométricas. A primeira delas foi o desenvolvimento da utilização do SOA em uma configuração tipo *"loop mirror"* [14]. Outros tipos de montagens interferométricas foram desenvolvidas para implementação de remodulação *2R* (*regeneration & reshape*, com estrutura tipo Michelson) [15] e chaveamento para (de)multiplexação no domínio do tempo [16] utilizando pulso de controle na região de ganho do dispositivo [17] e a portadora de informação fora da região de ganho, de forma a obter-se uma chave transparente, com baixo ruído e banda larga.

Utilizando-se uma montagem mais simples, através do controle da corrente de alimentação do SOA, pode-se tanto implementar operações de chaveamento [18] como de remodulação para reconstrução do sinal óptico [19] e reutilização de portadora [20]. É

---

12 - S. Kawanishi, K Okamoto, M. Ishii, O. Katamani, H. Takara e K Uchiyama - *"All-optical time-division-multiplexing of 100 Gbit/s signal based on four-wave mixing in travelling-wave semiconductor laser amplifier"* - Electronics Lett., Vol. 33, N°. 11, maio/1997.

13 - A. E. Kelly, A. D. Ellis, D. Nesset, R. Kashyap e D. G. Moodie, - *"100 Gbit/s wavelength conversion using FWM in a MQW semiconductor optical amplifier"* - Electronics Lett., Vol. 34, N°. 20, out./1998.

14 - M. Eiselt, W. Pieper e H. G. Weber - *"SLALOM: semiconductor laser amplifier in a loop mirror"* - J. Lightwave Tech., Vol. 13, N°. 10, out/1995.

15 - D. Wolfson, P. B. Hasen, A. Kioch e K. E. Stubkjaer - *"All- optical regeneration based on interferometric structure incorporating optical amplifiers"* - Electronic Lett., Vol. 35, N°. 1, jan./1999.

16 - S. Diez, R. Ludwig e H. G. Weber - *"Gain transparent SOA-switch for high-bitrate OTDM Add/Drop multiplexing"* - IEEE Photonics Tech. Lett., Vol. 11, N°. 1, jan./1999.

17 - R. J. Manning e D. A. O. Davies - *"Three-wavelength device for all-optical signal processing"* - Optics Lett., Vol. 19, N°. 12, junho/1994.

18 - A. Ehhardt, M. Eiselt, G. Großkopf, L. Kuller, R. Ludwig, W. Pieper, R. Schnabel e H. G. Weber - *"Semiconductor Optical Amplifier as Optical Switching Gate"* - J. of Lightwave Technology, Vol.11, No.8, agosto/1993.

19 - E. Conforti, A. C. Bordonalli, S. H. Ho e S. M. Kang - *"Optical 2R remodulator using feedforward control of semiconductor optical amplifier gain"* - Microw. Optical Techn. Letters, Vol. 21, N° 1, abril/1999.

20 - E. Conforti, S. H. Ho, A. C. Bordonalli, C. M. Gallep, J. P. B. Custódio, R. Simão e S. M. Kang - *"Optical carrier reuse with gain compression and feedforward semiconductor optical amplifier"*- Anais do *IMOC'99* (*Intenational Microwave and Optoelectronics Conference* -1999), reproduzido no APÊNDICE B.

nesse sentido que se orienta esse trabalho de mestrado, através da implementação computacional de um simulador para esse tipo de utilização do SOA como elemento de processamento de sinais ópticos.

1.4 - *SOASIM OVERVIEW*

O programa SOASim [3] é um simulador para amplificadores ópticos a semicondutor com controle dinâmico de ganho (*feed-forward gain control - FFG*) construído com as facilidades de programação do ambiente *C++Builder3* (*Borland Inc.*) e baseado no modelamento matemático desenvolvido por Agrawal e Olsson [**21**]. O programa simula o formato temporal do sinal óptico de saída para um dado sinal de entrada, fornecendo também as formas do ganho, dinâmica do tempo de vida dos portadores elétricos e diferença de fase resultante das alterações no índice de refração da região ativa. A implementação do controle dinâmico de ganho é realizado através da adição à corrente de polarização do dispositivo de uma componente proporcional ao sinal de entrada, como ilustrado na Figura 1.3.1. Nesse tipo de montagem, uma amostra do sinal de entrada é coletada, convertida para o domínio elétrico, passa por amplificador e uma linha de atraso, convenientemente ajustada de modo que o sinal elétrico resultante, somado à corrente de polarização I_{DC}, chegue no amplificador juntamente com o sinal óptico, que foi devidamente atrasado por um trecho ajustável de fibra.

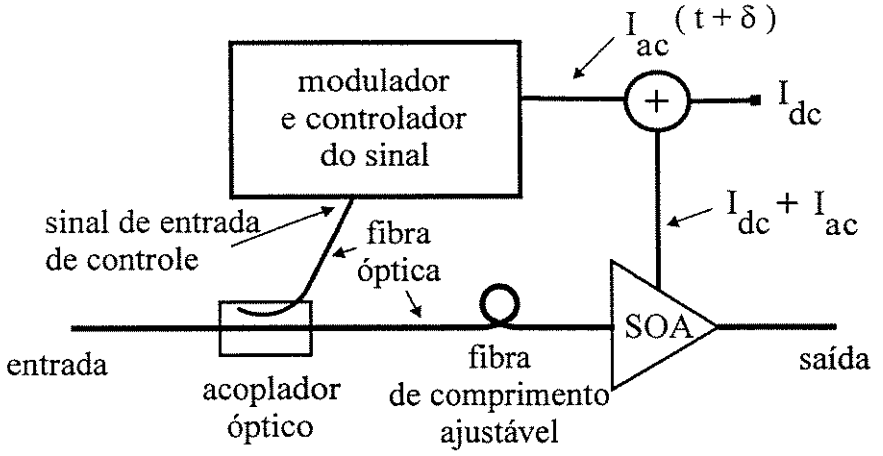

Figura 1.3.1: esquema de implementação do controle dinâmico de ganho (*FFG*).

Tal controle dinâmico de ganho [**19**] pode ser utilizado para a implementação de remoduladores tipo 2R [**22**] (*regeneration & reshape*), utilizados para melhorar a razão de extinção do sinal óptico e suprimir o excesso de ruído, ou para o apagamento da informação numa portadora a ser reutilizada [**20**].

---

21 - G. P. Agrawall e N. A. Olsson - *"Self-phase modulation and spectral broadening of pulses in semiconductor laser amplifiers"* - J. Quantum Electr., Vol. 25, N° 11, pag. 2297-2306, nov./1989

22 - P. E. Green, Jr., F. J. Janiello e R. Ramaswami - *"WDM protocol-transparent distance extension using R2 remodulation"* - IEEE Select Areas Commun., Vol. 14, pag. 962-967, 1996.

O modelamento utilizado, bem como as modificações introduzidas nesse trabalho, são apresentados detalhadamente no Cap. 2 dessa tese. No Cap. 3 é apresentada a implementação computacional do modelo e no Cap. 4 o programa e instruções de utilização. No Cap. 5 são apresentados resultados teóricos comparados com seus equivalentes experimentais, no intuito de analisar a fidelidade do programa. Finalmente, no Cap. 6 são apresentadas as conclusões relativas a esses resultados, bem como caminhos de melhoramento do *software* em questão.

CAPÍTULO 2 - Modelamento

2.1 - INTRODUÇÃO

Nesse capítulo é apresentado o modelamento matemático desenvolvido por Agrawal e Olsson [**21**], também contido na referência [**23**], para a amplificação de pulsos em amplificadores ópticos a semicondutor. São apresentadas, também, as expressões introduzidas no modelamento de modo a considerar a atenuação intrínsica do material e as perdas de acoplamento do sinal óptico ao passar da fibra para o *chip* do SOA, a variação do tempo de vida dos portadores elétricos com sua própria densidade, e implementar o controle dinâmico de ganho (*FFG - feed-forward gain control*) do SOA.

Para detalhes relativos à implementação computacional e ao programa final, consultar Capítulos 3 e 4.

2.2 - MODELAMENTO MATEMÁTICO

A teoria de propagação de pulsos em amplificadores ópticos geralmente trata esse dispositivo como um sistema de dois níveis (banda de valência e banda de condução, esta de estados com maior energia que a anterior), aproximação adequada para o caso de dispositivos sólidos como também para dispositivos a gás. Essa aproximação pode ser estendida para amplificadores ópticos a semicondutor se a região ativa for modelada como uma coleção de sistemas de dois níveis sem interação entre sí, com energias de transição estendendo-se sobre todo o intervalo das bandas de valência e condução. Todavia, esse modelamento ainda é difícil de ser implementado. Simplificações consideráveis podem ser feitas, porém, se a largura de pulso τ_{pulse} é assumida como muito maior que o tempo de relaxação intra-bandas $\tau_{intr.}$, que governa a dinâmica da polarização induzida. Como $\tau_{intr.}$ é igual ou menor a 0,1 ps, todo pulso com $\tau_{pulse} \geq 0{,}1$ ps pode ser considerado nesse modelamento simplificado. Nesse tipo de aproximação, o meio de ganho óptico responde ao campo óptico E como descrito pela equação de taxa:

$$\frac{\partial N}{\partial t} = D\nabla^2 N + \frac{I}{qV} - \frac{N}{\tau_c} - \frac{a\cdot(N-N_0)}{h\nu_0}|E|^2 \qquad (2.2.1)$$

onde N é a densidade de portadores elétricos, D o coeficiente de difusão, I a corrente injetada, q a carga do elétron, V o volume ativo, τ_c o tempo de vida dos portadores elétricos, antes que sejam consumidos por emissão espontânea, a o coeficiente de ganho da seção transversal, N_0 a densidade de portadores necessária à transparência e $h\nu_0$ a energia do fóton. Dessa maneira, a variação temporal da densidade de portadores elétricos depende dos quatro termos do lado direito da expressão (2.2.1), sendo que os dois primeiros se referem

---

23 - G. P. Agrawal e N. K. Dutta- *"Semiconductor Lasers"* - 2ª edição/1993, Van Nostrand Reinhold - NY.

ao acréscimo de portadores devido, respectivamente, à difusão de portadores no interior da cavidade e à injeção de portadores pela corrente de alimentação, e os dois últimos ao consumo de portadores devido, respectivamente, à emissão espontânea de fótons e à emissão estimulada pelo campo E, onde ocorre a amplificação do sinal.

A propagação do campo eletromagnético no interior da cavidade de amplificação é descrita pela equação de onda

$$\nabla^2 E - \frac{\varepsilon}{c^2}\frac{\partial^2 E}{\partial t^2} = 0 \qquad (2.2.2)$$

onde c é a velocidade da luz no vácuo e ε é constante dielétrica dada por

$$\varepsilon = n_b^2 + \chi \qquad (2.2.3)$$

onde n_b^2 é o índice de refração de fundo, sendo geralmente uma função das coordenadas transversais x e y, levando em conta o guiamento por índice dentro do SOA, e χ é a susceptibilidade, função da densidade de portadores N e que representa a contribuição da carga do portadores dentro da região ativa do SOA. A dependência exata de χ com N é bem complicada, dependendo, entre outras coisas, dos detalhes da estrutura de bandas do dispositivo. Num modelo simplificado, χ é assumido como sendo uma função linear de N, dado por

$$\chi(N) = -\frac{n_{efc} c}{\omega_0}(\beta_c + i)\cdot a(N - N_0) \qquad (2.2.4)$$

onde n_{efc} é o índice de modo efetivo, ω_0 a freqüência do sinal óptico e β_c é chamado de fator de ênfase de largura de linha (*linewidth enhancement factor*), com valores típicos entre 3 e 8 para SOAs.

As equações 2.2.1 a 2.2.4 fornecem um quadro teórico geral para a propagação de pulsos em amplificadores ópticos a semicondutor. Na prática, porém, são necessárias diversas aproximações para simplificar o problema. Será aqui considerado um TW-SOA ideal e assumindo que as dimensões de sua região ativa são tais que promovem guiamento monomodo na freqüência utilizada. É assumindo também o campo de entrada como linearmente polarizado, mantendo-se assim durante toda a propagação no interior do amplificador. Com essas considerações, pode-se dizer que dentro do SOA o campo pode ser descrito como

$$\vec{E}(x, y, z, t) = \{F(x, y)\, A(z, t) \exp[i(k_0 z - \omega_0 t)]\}\frac{\hat{x}}{2} + \text{c.c.} \qquad (2.2.5)$$

onde $\hat{x}$ é o vetor unidade de polarização, $F(x,y)$ é a distribuição de modos no guia de onda, k_0 o número de onda definido como $k_0 = n_{efc}\omega_0 / c$ e $A(z,t)$ é a envoltória de variação

temporal lenta, associada ao pulso óptico. Substituindo (2.2.5) em (2.2.2), desprezando as derivadas segundas de A com relação a ambas variáveis z e t e integrando nas dimensões transversais, obtêm-se:

$$\frac{\partial^2 F(x,y)}{\partial x^2}+\frac{\partial^2 F(x,y)}{\partial y^2}+\left(n_b^2-n_{efc}^2\right)\cdot\frac{\omega_0^2}{c^2}F(x,y)=0 \tag{2.2.6}$$

$$\frac{\partial A(z,t)}{\partial z}+\frac{1}{v_g}\frac{\partial A(z,t)}{\partial t}=\frac{i\omega_0\Gamma}{2n_{efc}c}\chi A(z,t)-\frac{1}{2}\alpha_{int}A(z,t) \tag{2.2.7}$$

onde v_g é a velocidade de grupo definida como c/n_g, com o índice de grupo definido como $n_g=n_{efc}+\omega_0\frac{\partial n_{efc}}{\partial\omega}$, e o fator de confinamento Γ como

$$\Gamma=\frac{\int_0^w\int_0^d|F(x,y)|^2dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}|F(x,y)|^2dx\,dy} \tag{2.2.8}$$

A solução de (2.2.6) fornece a distribuição transversal de campo $F(x,y)$ e o índice de modo efetivo n_{efc}. A equação (2.2.7) descreve a evolução da amplitude do pulso ao longo do comprimento do amplificador. Os efeitos transversais são incluídos pelo fator de confinamento Γ. O último termo em (2.2.7) se refere às perdas internas α_{int} experimentadas pelo modo quando $\chi = 0$. A dispersão de velocidade de grupo devida a derivada segunda do índice de refração com a freqüência do campo é desprezível para a pequena dimensão do comprimento do SOA e para a largura de pulso considerada, mas podem ser incluídas através de um termo contendo a segunda derivada de A com o tempo no lado esquerdo de (2.2.7).

A equação de densidade de portadores N (2.2.1) também pode ser simplificada notando-se que a largura w e a espessura d da região ativa são geralmente menores e seu comprimento L bem maior que o comprimento de difusão. Tomando então a distribuição de portadores como praticamente uniforme ao longo das dimensões transversais, pode-se utilizar um valor médio como boa aproximação e desprezar a difusão de portadores. Calculando a média sobre as dimensões da região ativa, obtém-se

$$\frac{dN}{dt}=\frac{I}{qV}-\frac{N}{\tau_c}-\frac{g(N)}{h\nu_0}\cdot|A|^2 \tag{2.2.9}$$

onde o ganho g é definido por

$$g=\Gamma a(N-N_0) \tag{2.2.10}$$

Combinando (2.2.9) e (2.2.10), obtém-se a equação de taxa para o ganho:

$$\frac{\partial g}{\partial t} = \frac{g_0 - g}{\tau_c} - \frac{g|A|^2}{E_{sat}} \tag{2.2.11}$$

onde E_{sat} é a energia de saturação do amplificador, representando a energia de pulso a partir da qual o amplificador se encontra profundamente saturado, não amplificando significativamente o sinal, definida como

$$E_{sat} = \frac{h\upsilon_0 \cdot wd}{\Gamma a} \tag{2.2.12}$$

e o ganho para pequenos sinais g_0, onde a saturação da região ativa é desprezada, é definido como

$$g_0 = \Gamma a N_0 (I / I_0 - 1) \tag{2.2.13}$$

onde $I_0 = qN_0V/\tau_c$ é a corrente requerida para a transparência, com V sendo o volume da cavidade ativa.

As equações (2.2.7) e (2.2.11) descrevem a propagação de pulsos em amplificadores ópticos a semicondutor e podem ser bem simplificadas se for feita a transformação de variável

$$\tau = t - z/v_g \tag{2.2.14}$$

onde o tempo reduzido τ é medido com referência a um quadro movendo-se juntamente com o pulso. Fazendo também a separação da amplitude e da fase do pulso, na forma

$$A = \sqrt{P} \cdot \exp(i\phi) \tag{2.2.15}$$

onde $P(z,t)$ e $\phi(z,t)$ são, respectivamente a potência e a fase do sinal, chega-se ao seguinte conjunto de equações, conseguidas a partir das equações (2.2.7) e (2.2.11) em conjunto com (2.2.4) e (2.2.10):

$$\frac{\partial P}{\partial z} = (g - \alpha_{int})P \tag{2.2.16}$$

$$\frac{\partial \phi}{\partial z} = -\frac{1}{2}\beta_c g \tag{2.2.17}$$

$$\frac{\partial g}{\partial \tau} = \frac{g_0 - g}{\tau_c} - \frac{gP}{E_{sat}} \tag{2.2.18}$$

onde (2.2.16) descreve a amplitude do sinal à medida que ele se propaga na direção z, de acordo com o ganho saturado (2.2.18), que também promove uma modulação de fase temporal em (2.2.17), que mostra a origem da auto-modulação de fase (*SPM - self-phase modulation*) [**21**].

Para se obter uma solução simples de (2.2.16) a (2.2.18), desprezam-se as perdas internas α_{int} (consideradas isoladamente, como apresentado mais adiante) e integra-se sobre o comprimento do amplificador, fornecendo

$$P_{out}(\tau) = P_{in}(\tau)\exp[h(\tau)] \tag{2.2.19}$$

$$\phi_{out}(\tau) = \phi_{in}(\tau) - \frac{1}{2}\beta_c\, h(\tau) \tag{2.2.20}$$

onde P_{in} e ϕ_{in} são, respectivamente, a potência e a fase do sinal de entrada, sendo a função $h(\tau)$ definida como

$$h(\tau) = \int_0^L g(z,\tau)dz \tag{2.2.21}$$

Numa visão do acontecimento físico, (2.2.21) representa o ganho da cavidade integrado por toda sua extensão, para cada ponto do pulso. Integrando (2.2.18) e utilizando (2.2.16), temos $h(\tau)$ como a solução da equação diferencial ordinária

$$\frac{\partial h}{\partial \tau} = \frac{g_0 L - h}{\tau_c} - \frac{P_{in}}{E_{sat}}[\exp(h) - 1] \tag{2.2.22}$$

Para um dado sinal de entrada P_{in} e um ganho de pequenos sinais g_0 obtém-se então a potência do sinal de saída através de (2.2.19) e sua fase através de (2.2.20). No intuito de introduzir também as perdas internas e por acoplamento α_{acp}, utilizou-se na implementação realizada nesse trabalho a expressão

$$P_{out}(\tau) = \alpha_{acp} \cdot P_{in}(\tau) \cdot \exp[h(\tau) - \alpha_{int} L] \tag{2.2.23}$$

onde, de maneira simplificada, as perdas internas são consideras como uma atenuação em um dispositivo passivo de mesmo comprimento que o da cavidade ativa.

Para a implementação do controle dinâmico de ganho do amplificador a semicondutor, utilizou-se uma corrente de injeção da seguinte forma

$$I(t) = I_{DC} + K\left(P_{in,nomal.}(t+\delta) - rms(P_{in,nomal.})\right) \qquad (2.2.24)$$

onde I_{DC} é a componente contínua da corrente total $I(t)$, K é o ganho de modulação, que multiplica o valor normalizado do sinal óptico de entrada decrescido de seu valor eficaz no período e deslocado no tempo de um ângulo δ. Esse ângulo é escolhido de forma a proporcionar um adiantamento ($\delta < 0$) ou atraso ($\delta > 0$) dessa componente modulada da corrente com relação ao sinal óptico de entrada. O ganho K, tem dimensão de corrente, e também pode ser escolhido de forma a somar ou subtrair a componente modulada na componente contínua. Essa corrente é utilizada no cálculo do ganho de pequenos sinais g_0 em (2.2.13), de forma que o ganho total a ser calculado em (2.2.22) passe a ser proporcional ao sinal de entrada.

Para que o modelo também considere a dinâmica do tempo de vida dos portadores elétricos, essa variável também passa a ser uma função temporal, dependente da concentração dos portadores na região ativa, sendo calculada a cada ciclo de resolução de (2.2.22) (ver Cap. 3 e 4). Desprezando as recombinações devido a defeitos na estrutura do dispositivo, utilizou-se a expressão [**23-24**]:

$$\tau_c(t) = 1/\left(BN(t) + CN(t)^2\right) \qquad (2.2.25)$$

onde B é o fator de recombinação radioativa e C o fator de recombinação Auger. A determinação desses fatores é geralmente experimental e complicada. Para maiores detalhes, consultar referências [**23-24**].

Para uma boa aproximação no cálculo de (2.2.25), a variação dinâmica da densidade de portadores N é calculada pela expressão [**23-24**]:

$$N = \frac{V}{\Gamma a} g + N_0 \qquad (2.2.26)$$

Outra expressão utilizada em algoritmo do *SOASim* é o tempo médio de vida dos fótons na cavidade ativa, antes da emissão espontânea:

$$\tau_p = \frac{1}{v_g \cdot \alpha_{cav}} \qquad (2.2.27)$$

onde $v_g = c / n_{efc}$ [m/s] é a velocidade de grupo α_{cav} [m^{-1}] é a taxa de perda no percurso completo na cavidade (ida e volta), dada por

$$\alpha_{cav} = \alpha_{int} + \frac{1}{2L} \cdot \log\frac{1}{R_1 R_2} \qquad (2.2.28)$$

---

24 - H. Ghafouri-Shiraz - *"Fundamentals of laser diode amplifiers"* - 1996, John Wiley & Sons, NY.

2.3 - CONCLUSÃO

Foi apresentado, neste capítulo, o equacionamento do modelo utilizado, bem como suas aproximações, para a construção do *SOASim*, simulador de amplificadores ópticos a semicondutor com controle dinâmico de ganho, sendo que os detalhes de implementação computacional e o programa em si são apresentados nos Capítulos 3 e 4.

A discussão sobre o funcionamento do SOA não foi estendida em demasia, pois já foi devidamente detalhada na tese de mestrado do colega Juliano P. B. Custódio, em trabalho anterior no mesmo grupo de pesquisa, chefiado pelo Prof. Evandro Conforti, do Departamento de Microonda e Óptica (DMO) da Faculdade de Engenharia Elétrica e de Computação (FEEC), na Universidade Estadual de Campinas (Unicamp), campus de Campinas.

CAPÍTULO 3 - Implementação Computacional

3.1 - INTRODUÇÃO

Nesse capítulo é apresentada a implementação computacional do simulador de amplificadores ópticos a semicondutor a partir do equacionamento proposto por Agrawal e Olsson [**21**] para descrever seu funcionamento, bem como as modificações propostas no intuito de incorporar o controle dinâmico do ganho e a variação do tempo de vida dos portadores com a densidade dos mesmos, discutidos no Capítulo 2. Essa implementação computacional, realizada com as facilidades de depuração e objetos gráficos de interface oferecidas pelo ambiente de desenvolvimento *C++Builder3* (*Borland*) teve como resultado no programa de simulação ***SOASim*** [**3**], que tem sua interface com o usuário e algoritmos apresentados no Capítulo 4.

Na Seção 3.2 é apresentada a sequencia lógica de funcionamento do programa, em forma sucinta.

Nas Seções 3.3 à 3.7 são apresentados em detalhes os passos dessa lógica computacional.

Na Seção 3.8 são apresentados os métodos numéricos utilizados na implementação descrita nas seções anteriores.

3.2 - SEQÜÊNCIA LÓGICA

Nessa seção é apresentada a seqüência lógica de realização das tarefas pelo algoritmo a ser apresentado no Capítulo 4, bem como os detalhes de implementação necessárias ao seu entendimento.

Basicamente, a simulação da amplificação óptica ocorre como ilustrado na Figura 3.2.1. A partir de um sinal de entrada $P_{in}[i]$, correspondente ao um sinal óptico discreto no tempo com dez mil amostras (discretização temporal utilizada como padrão) é criado um sinal de corrente $I[i]$ (caminho ①), composto por uma componente contínua mais uma componente modulada proporcional ao conteúdo do sinal normalizado de entrada diminuído de seu valor eficaz no período. Para a obtenção da condição inicial $h[1]$ necessária ao algoritmo de cálculo do ganho do SOA, um ganho inicial ao longo do eixo z é calculado e integrado nessa direção (caminho ②). Com posse da condição inicial, do sinal de entrada e da corrente de alimentação, o programa calcula o ganho total $h[i]$ do dispositivo para todos os pontos do tempo. Caso a opção de variação dinâmica do tempo de vida dos portadores com sua densidade esteja ativada, em cada ciclo de cálculo do ganho $h[i]$, a densidade de portadores $N[i]$ e o tempo de vida $\tau_c[i]$ são calculados (caminho ④, setas brancas) e utilizados no próximo ciclo, juntamente com a corrente total $I[i]$, para o cálculo do ganho de pequenos sinais $g_0[i]$ utilizado para o cálculo do ganho total $h[i]$. Se a opção não estiver ativada, apenas a corrente de alimentação entra como variável para o cálculo do ganho de pequenos sinais. Com posse do ganho total $h[i]$, juntamente com as perdas internas e de acoplamento, é calculado o sinal de saída do amplificador e seu ganho fibra-fibra.

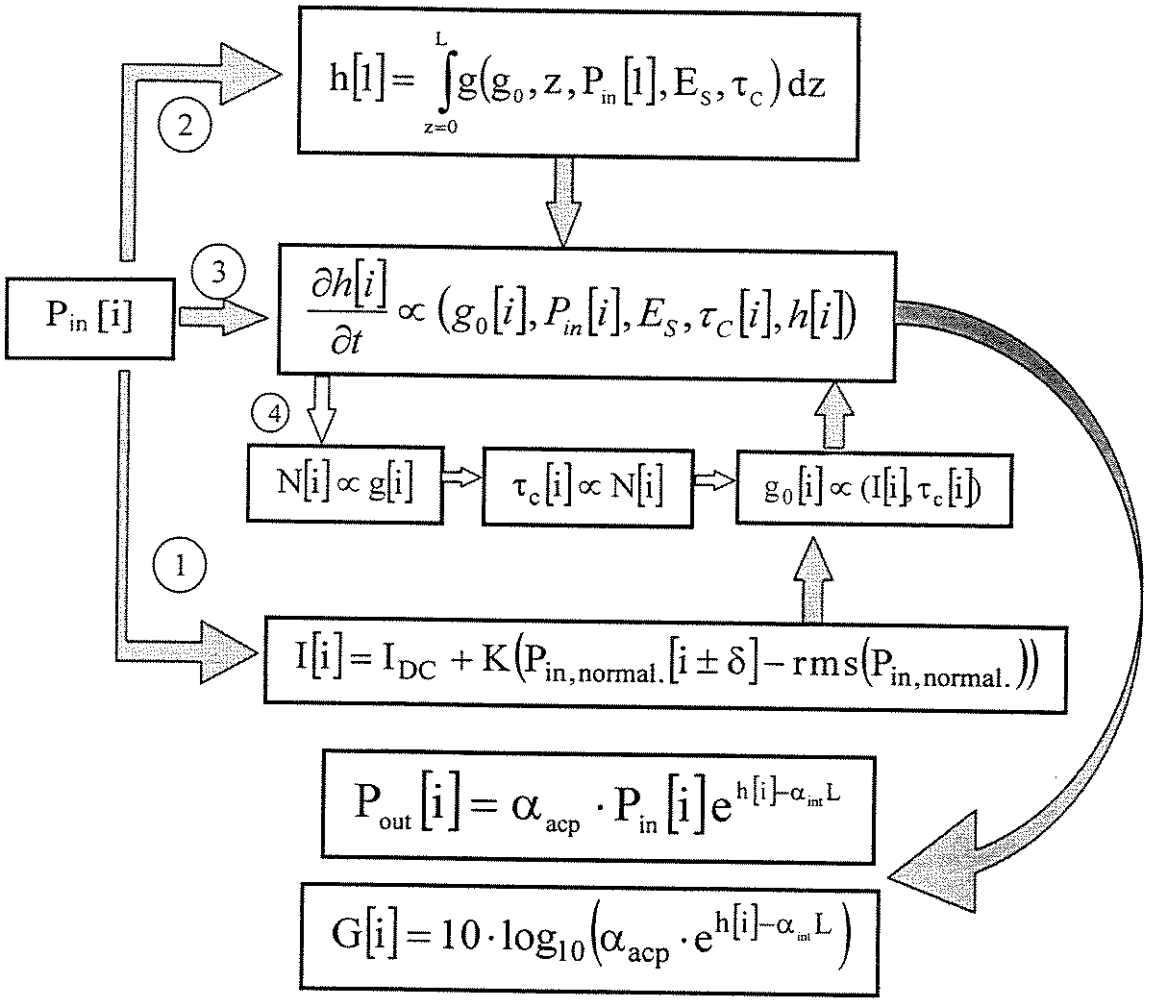

Figura 3.2.1: diagramação da estrutura de funcionamento do SOASim

Nas seções que seguem, cada um dos passos descritos resumidamente acima serão detalhados para o perfeito entendimento do algoritmo computacional.

3.3 - Sinal de Entrada

O sinal de potência óptica de entrada para a amplificação simulada pode ser tanto gerado pelo programa ou carregado a partir de um arquivo tipo ASCII, de extensão *.dat*, gerado pelo usuário com o auxílio de um software gráfico (ver Seção 4.2).

A geração do pulso de entrada $P_{in}[i]$ pelo simulador é feita por um algoritmo de criação de pulsos super-gaussianos (função *"create_pulse( )"*, Seção 4.3), que gera o vetor de discretização temporal *t[i]* de acordo com a taxa de transmissão escolhida e utiliza a eq. (3.3.1) em sua forma discreta (função *"pul_temp( )"*) para gerar o pulso de formato super-gaussiano. Se o sinal possui mais de um pulso, o número de amostras de cada pulso é reduzido proporcionalmente.

$$P(t) = \min \cdot \max + (1 - \min) \cdot \max \cdot \exp\left[-\left(\frac{t - T/2}{2T/10} \right)^{2 \cdot form} \right] \qquad (3.3.1)$$

onde *P(t)* é a potência óptica do sinal com relação ao tempo *t*, *min* e *max* são o nível mínimo e máximo de potência óptica, respectivamente, *T* o período do pulso, igual ao inverso da taxa de transmissão, e *form* o formato do pulso super-gaussiano.

O número *form* é fator importante para adequação de um sinal simulado ao experimental (ver Cap. 5), pois influi na forma mais ou menos abrupta de subida e descida do pulso, como ilustrado na Figura 3.3.1. Quanto maior o *form*, mais retangular o pulso: para *form* = 1 (curva de traço cheio mais grosso) o pulso é o gaussiano normal e a medida que aumenta-se esse valor passando para 2, 4, 16 e 40 o pulso tende a se ajustar a forma retangular. Nota-se pouca diferença do traçado da curva com *form* = 16 (pontilhada grossa) para a curva com *form* = 40 (pontilhada fina).

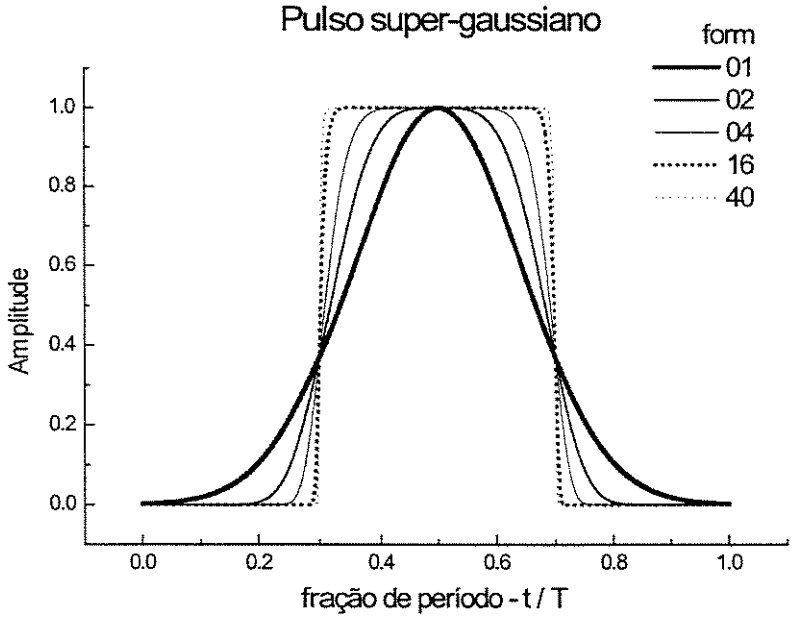

Figura 3.3.1: dependência do formato do pulso super-gaussiano com o número *form*

Com relação à taxa de transmissão a ser escolhida pelo usuário do simulador, convém observar que cada pulso aqui corresponde à forma da Figura 3.3.1, onde um nível lógico alto de duração T/2 apresenta em ambos os lados (antes e depois) nível lógico baixo de duração T/4, aproximadamente . Portanto, para gerar taxa de A bits por segundo, tomando um bit como apenas um símbolo lógico, deve-se usar uma taxa no programa de A/2 Hz, pois no simulador um ciclo é igual a um pulso super-gaussiano. Tal fato é novamente comentado no Cap. 5, referente aos resultados de simulação.

Para importar uma entrada já pronta, no formato conveniente, o usuário deve salvá-la em "*C:\SOAsimulator\temp\inputL.dat*" (ver seções 4.2 e 4.3, função *"loadinputpulse( )"*), de forma a ser reconhecida pelo programa . Esse tipo de

procedimento é utilizado quando deseja-se que o formato do trem de pulsos seja outro que não uma seqüência de pulsos super-gaussianos, como ilustrado nas simulações de experimentos no Cap. 5. Também pode ser utilizado para inserir sinal similar ao utilizado em experimento, facilitando a obtenção de uma resposta ideal para o simulador.

3.4 - CORRENTE DE ALIMENTAÇÃO

A corrente de alimentação *I[i]* do SOA é obtida utilizando-se a eq. (2.2.24), na sua forma discreta:

$$I[i] = I_{DC} + K\left(P_{in,normal.}[i \pm \delta] - rms\left(P_{in,normal.}\right)\right) \tag{3.4.1}$$

onde I_{DC} [mA] é a componente contínua da corrente; K [mA] é o ganho de modulação, positivo ou negativo, que multiplica a norma do sinal de entrada $P_{in,normal.}$ descontada de seu valor eficaz (*rms - root mean square*) e deslocada convenientemente de um ângulo δ, que define a diferença de fase entre a modulação da corrente e o sinal de entrada. Defini-se como valor eficaz de uma função como a raiz quadrada da integral no período escolhido do quadrado da função, dividido pelo período, como na eq. (3.4.2). Para verificar o algoritmo, ver Seção 4.3, função *"current( )"*.

$$rms(f) = \sqrt{\frac{1}{T}\int_0^T f^2(\tau)d\tau} \tag{3.4.2}$$

O sinal da corrente de alimentação também pode ser carregado a partir de um arquivo *ASCII*, de extensão *.dat*, como feito para o sinal de entrada. Para tanto, esse arquivo deve ser salvo como *"C:\SOAsimulator\temp\IinL.dat"* (função *"loadcurrent()"*, Seção 4.2 e 4.3). Dessa forma, podem-se também utilizar outras formas, escolhidas de acordo com o objetivo do usuário, para o sinal de corrente utilizado na simulação.

3.5 - GANHO INICIAL

O cálculo do ganho inicial fornece a condição inicial *h[1]* necessária ao algoritmo de implementação do método numérico de resolução de equações diferenciais ordinárias conhecido como Runge-Kutta de quarta ordem utilizado para resolver a eq. (2.2.22) na sua forma discreta, comentado mais adiante (RK4 - ver Seção 3.8).

Para tanto, o programa utiliza o valor inicial da potência de entrada $P_{in}[1]$ e a equação

$$g = \frac{g_0}{1 + P/P_s} \tag{3.5.1}$$

onde $P_s=E_s/\tau_c$ é a potência de saturação do SOA, E_s sua energia de saturação, e τ_c o tempo de vida dos portadores (ver Cap. 1 e 2) juntamente com o algoritmo RK4, para encontrar a variação do ganho *g[j]* na direção *z* para o sinal de entrada $P_{in}[1]$ (função *"g_0_z( )"*). Feito isso, o programa integra, com a utilização do algoritmo da regra de Simpson 3/8 (ver Seção 3.8), esse ganho espacial e obtém a condição inicial *h[1]* (função *"integral( )"*), utilizada no próximo passo. Para conferir a implementação dos algoritmos, ver Seção 4.3.

3.6 - Ganho Total

Com posse da condição inicial de ganho *h[1]*, o programa pode calcular o ganho total, para todos os pontos da discretização temporal, utilizando novamente o método de Runge-Kutta de quarta ordem, como mostrado na função *"calculo( )"* do programa (Seção 4.3).

Quando acionada a opção de variação dinâmica do tempo de vida dos portadores elétricos (ver Seção 4.2), o algoritmo calcula o valor inicial da densidade de portadores *N[1]* (eq. (2.2.26)) a partir de *h[1]*. Calculado *N[1]*, o programa utiliza-o para o cálculo do valor inicial para o tempo de vida dos portadores elétricos $\tau_c[1]$ que é utilizado, juntamente com o valor discreto da corrente de alimentação *I[1]*, para o cálculo do ganho de pequenos sinais inicial *g0[1]*. Então inicia-se a série de ciclos para o cálculo de *h[i]* utilizando RK4, sendo que, após incrementar o contador *i* em cada ciclo, os valores de *N[i]*, $\tau_c[i]$ e *g0[i]* (eq. (2.2.26), (2.2.25) e (2.2.13)) são calculados para serem utilizados no próximo ciclo de cálculo.

Quando essa opção não é ativada, o valor do tempo de vida dos portadores elétricos τ_c permanece constante no tempo e é determinado pela escolha do usuário em sua janela correspondente (Seção 4.2). Então, antes de se iniciar os ciclos para o cálculo do ganho total *h[i]*, o vetor do ganho para pequenos sinais *g0[i]* é calculado com base apenas nas variações contidas no vetor de corrente *I[i]* (eq. (2.2.13)).

3.7 - Sinal de Saída

Com o ganho total *h[i]* calculado, o programa calcula, então, o sinal de saída a partir do sinal de entrada e das perdas de inserção e acoplamento, como na eq. 2.2.23. São calculados, também, o sinal normalizado de saída, o ganho fibra-a-fibra e a diferença de fase entre os sinais ópticos de entrada e de saída do SOA (ver Seção 4.2).

3.8 – MÉTODOS NUMÉRICOS

Nessa seção são apresentados, resumidamente, os métodos numéricos [25] utilizados na implementação descrita nas seções anteriores. Para analisar a sua utilização durante a execução do programa, consultar Seção 4.3.

3.8.1 - Integração de equações diferenciais ordinárias

A resolução de problemas envolvendo equações diferenciais ordinárias (ODE - "*ordinary diferencial equations*") podem sempre ser reduzidos ao estudo de conjuntos de equações diferenciais de primeira ordem. Como exemplo, tome-se a equação de segunda ordem abaixo:

$$\frac{d^2y}{d^2x} + q(x)\frac{dy}{dx} = r(x) \tag{3.8.1}$$

onde x é a variável independente e y, q e r funções de x, que pode ser reescrita como duas equações de primeira ordem:

$$\begin{aligned} \frac{dy}{dx} &= z(x) \\ \frac{dz}{dx} &= r(x) - q(x)z(x) \end{aligned} \tag{3.8.2}$$

onde z é uma nova variável do problema, introduzida com o intuito de facilitar a integração e escolhida usualmente como uma derivada das variáveis originais. De acordo com o problema a ser resolvido utilizam-se alguns fatores ou variáveis independentes com o propósito de evitar singularidades, que poderiam resultar em *"overflow"* ou aumento do erro numérico de cada passo.

O problema genérico que tem-se que enfrentar na resolução de equações diferenciais ordinárias é reduzi-las a conjuntos de N equações de primeira ordem acopladas para a função y_i, com i =1, 2,...,N, com a forma genérica:

$$\frac{dy_i(x)}{dx} = f_i(x, y_1, ..., y_n), i = 1, ..., N \tag{3.8.3}$$

sendo f_i funções conhecidas de x e y.

O problema crucial, entretanto, é determinar como atacar o problema numericamente de acordo com as condições de contorno disponíveis, isto é, condições algébricas sobre valores y_i, que podem ser desde simples valores numéricos de alguma variável até complicados conjuntos de equações não-lineares envolvendo algumas ou mesmo todas as variáveis. Usualmente, é a natureza da condição de contorno que

---

25 - *Numerical Recipes in C: the art of scientific computing* - Cambridge University Press - 1992

determina o método numérico factível no caso, podendo ser divididas em duas grandes categorias:

1 - Problemas de valor inicial, onde os valores de todos y_i são conhecidos em algum ponto inicial x_s e deseja-se conhecer o valor final dos y_i em algum x_f final;

2 - Problemas de contorno em duas fronteiras, onde tipicamente algumas condições são especificadas para o ponto inicial x_s e as demais para o final x_f. Este tipo de problema é de resolução bem mais difícil que o anterior e, portanto, não será considerado nesse texto.

A idéia básica para a resolução de problemas de valor inicial pode ser resumida da seguinte maneira: reescrever dy's e dx's em (3.8.3) como passos finitos Δy e Δx e multiplicar a equação por Δx, conseguindo relações algébricas para as mudanças nas funções a cada mudança de x, incrementado em passos Δx. Fazendo Δx bem pequeno, obtêm-se boas aproximações para a equação diferencial analisada. A implementação literal desse procedimento é conhecida como Método de Euler, que, entretanto, não é recomendado para uso prático. Porém, esse método é de grande importância conceitual e sua idéia básica é utilizada para a confecção de métodos práticos.

São três os melhores tipos de métodos numéricos para a resolução de problemas de valor inicial para *ODE*s. Neste trabalho concentra-se no chamado Método de Runge-Kutta, que propaga a solução em um intervalo através da combinação das informações fornecidas por vários passos do tipo Euler, cada um envolvendo uma avaliação das f's em (3.8.3). Este é o método utilizado para resolução da maioria dos problemas.

Os outros dois métodos alternativos são:

- Extrapolação de Richardson, que usa a idéia de extrapolar um resultado para um valor que seria obtido se o passo fosse bem menor que o utilizado, desejando-se que a extrapolação chegue ao passo de dimensão zero. Esse método é freqüentemente chamado de Burlisch-Stoer; já que esses dois autores foram os primeiros a realizar uma implementação prática desta idéia;

- Método Preditor-Corretor, que armezena a solução ao longo do caminho e usa esses resultados para extrapolar a solução do passo seguinte, corrigindo então essa extrapolação usando a informação derivada no novo ponto. Este método é tido como o melhor método para funções bem regulares.

Método Runge-Kutta

Consideremos a fórmula do Método de Euler:

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{3.8.4}$$

onde y_i é o valor da função no ponto x_i, que avança da solução de x_n para $x_{n+1} \equiv x_n + h$. A presente fórmula é assimétrica, pois avança a solução em um intervalo h mas usa informações do início do intervalo apenas, como mostra a Figura 3.8.1, onde as setas representam a estimativa da derivada da função nos pontos, que, extrapoladas no

intervalo h, fornecem o próximo ponto. Isto significa que o erro do passo é apenas uma potência de h menor do que a correção, isto é, $O(h^2)$ somado a (3.8.4), o que pode ser verificado numa expansão em séries de potência. Entre as diversas razões de não se utilizar esse método na pratica, é que ele não é de grande precisão quando comparado a outros com o mesmo tamanho de passo e nem tampouco possui grande estabilidade.

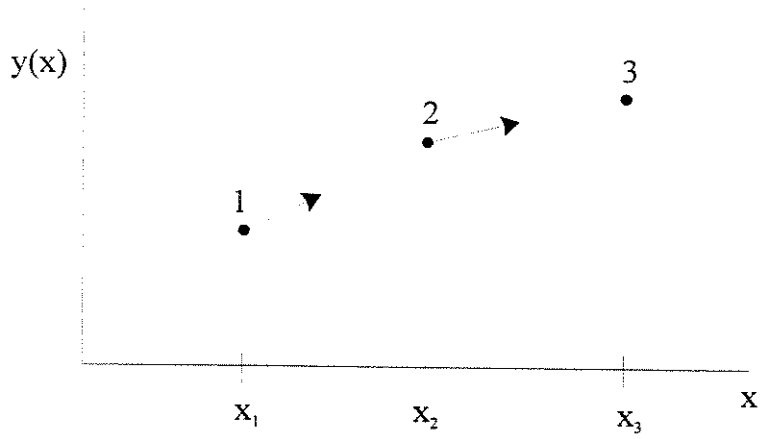

Figura 3.8.1: Método de Euler, onde a derivada do ponto inicial de cada intervalo é extrapolada para encontrar-se o próximo valor da função, com precisão de primeira ordem

Considera-se, agora, o uso de um passo como em (3.8.4) porém tomando um ponto no meio do intervalo e usando os valores de x e y nesse ponto mediano para calcular o passo total Δy através do intervalo completo $\Delta x=h$, ilustrado na Figura 3.8.2 e equacionado da seguinte forma:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ y_{n+1} &= y_n + k_2 + O(h^3) \end{aligned} \qquad (3.8.5)$$

onde k_1 é coeficiente obtido com a derivada no ponto inicial e k_2 obtido no meio do intervalo, obtendo-se precisão de segunda ordem com a utilização da derivada do ponto inicial de cada intervalo para se estimar o valor da função no meio do passo, utilizando-o então para o cálculo do passo todo. Os pontos intermediários 2 e 4 tem seus valores descartados assim que suas derivadas são calculadas e utilizadas.

Com essa simetrização o erro de primeira ordem é cancelado, e o método é classificado como de segunda ordem de precisão, como indica o termo do erro O, proporcional a terceira potência da discretização. Esse método é chamado então de Runge-Kutta de segunda ordem ou Método do Ponto Mediano.

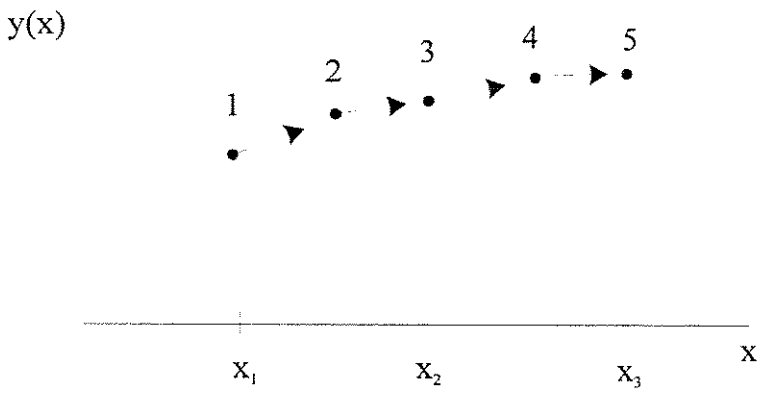

Figura 3.8.2: Método do Ponto Mediano.

Existem muitas maneiras de se avaliar a função *f(x,y)* que, uma vez combinadas corretamente, podem eliminar ordem por ordem os termos relativos ao erro. Esta é a idéia básica do método de Runge-Kutta. A clássica fórmula de Runge-Kutta de quarta-ordem é a mais utilizada e é mostrada a seguir:

$$\begin{aligned} k_1 &= hf(x_n, y_n) \\ k_2 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}) \\ k_3 &= hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}) \\ k_4 &= hf(x_n + h, y_n + k_3) \\ y_{n+1} &= y_n + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} + O(h^5) \end{aligned} \tag{3.8.6}$$

O método de Runge-Kutta de quarta-ordem requer quatro avaliações de f para cada passo h, como na Figura 3.8.3, de maneira que a cada passo a derivada é estimada quatro vezes: uma no ponto inicial (k_1), duas em pontos medianos de teste (k_2 e k_3) e uma num ponto final de teste (k_4). A partir dessas derivadas é calculado o valor final da função y_{n+1}.

Esse método será superior ao do Ponto Mediano se pudermos ter um passo maior que o dobro com a mesma precisão, isto é, mesmo número de avaliações da função para um dado passo, fato que acontece na maioria dos problemas, mas não em todos. Isso quer dizer que nem sempre uma ordem maior significa uma melhor precisão.

Entretanto, para a maioria dos usuários científicos, o método de Runge-Kutta de quarta-ordem é a melhor ferramenta para a resolução de problemas de integração de equações diferenciais de primeira ordem, principalmente se associado a um algoritmo de passo adaptativo, que otimiza o tamanho do passo a partir da análise do erro de truncamento.

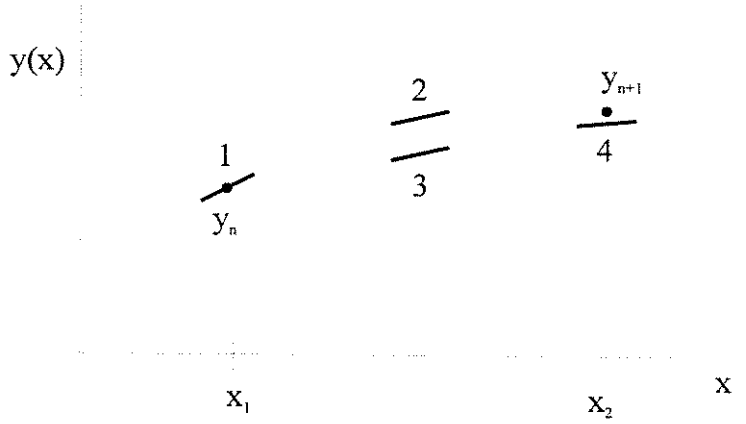

Figura 3.8.3: Método de Runge-Kutta de Quarta-Ordem.

3.8.2 - Integração de funções

A integração numérica, também chamada de quadratura, tem história anterior à invenção do cálculo. O fato de que integrais de funções elementares geralmente não podiam ser computadas analiticamente, ao contrário das suas derivadas, serviu para a criação de campo vasto para o desenvolvimento da análise numérica durante os séculos XVIII e XIX.

Com o advento da computação, a integração numérica tornou-se apenas mais uma questão numérica entre tantas, e nem sempre das mais interessantes. A computação trouxe a oportunidade de realização de um campo mais rico, o da integração numérica de equações diferenciais. A quadratura é simplesmente o caso especial da determinação do valor da integral

$$I = \int_a^b f(x)dx \tag{3.8.7}$$

que é equivalente à resolução, para $I \equiv y(b)$, da equação diferencial

$$\frac{dy}{dx} = f(x) \tag{3.8.8}$$

com a condição de contorno

$$y(a) = 0 \tag{3.8.9}$$

A Seção 3.8.1 trata de integração numérica de equações diferenciais, enfatizando os conceitos de variável e escolhas adaptativas de tamanho de passo. Neste capítulo o método de quadratura é baseado no conceito óbvio de somar o valor do integrando numa seqüência de abcissas dentro do intervalo de integração. O trunfo no caso é obter a integral mais precisa possível, com o menor número de determinações do valor do integrando.

3.8.2.1 - Fórmulas clássicas para abcissas igualmente espaçadas

As fórmulas clássicas, para integração de funções cujos valores são conhecidos em passos igualmente espaçados, apresentam certa elegância matemática. Entretanto quase todas entraram em desuso com o tempo devido a sua baixa eficiência computacional.

Partindo de uma seqüência de abcissas, notadas $x_0, x_1, ..., x_N, x_{N+1}$ e espaçadas por um passo constante h,

$$x_i = x_0 + ih \tag{3.8.10}$$

A função $f(x)$ tem valores conhecidos nos pontos x_i's

$$f(x_i) = f_i \tag{3.8.11}$$

Querendo integrar a função $f(x)$ entre um limite inferior a e um limite superior b, onde a e b são valores de x_i, podem ser utilizadas fórmulas chamadas *fechadas*, que usam os valores dos limites do intervalo $f(a)$ e $f(b)$. Ocasionalmente, os limites do intervalo de integração podem apresentar singularidades. Então, fórmulas *abertas* devem ser usadas, estimando a integração usando valores de x_i estritamente entre a e b.

Os blocos básicos de construção das fórmulas clássicas são regras para integrar uma função em um número pequeno de intervalos. Com o crescimento desse número, pode-se encontrar regras exatas para polinomiais de ordem crescente, lembrando porém que nem sempre maior ordem implica em maior precisão em casos práticos. Algumas dessas fórmulas fechadas são apresentadas a seguir.

3.8.2.2 - Fórmulas Fechadas de Newtom-Cotes

Regra Trapezoidal

A Regra do Trapézio, ou Trapezoidal, pode ser escrita na forma:

$$\int_{x_1}^{x_2} f(x)dx = h\left[\frac{f_1}{2} + \frac{f_2}{2}\right] + O(h^3 f''') \tag{3.8.12}$$

onde f_1 e f_2 são os valores assumidos pela função f em x_1 e x_2 ($x_1 + h$), respectivamente.

O termo de erro $O(\ )$ mostra que o valor verdadeiro da integral difere do valor estimado por um montante que é o produto de algum coeficiente numérico pelo h^3 pelo valor da segunda derivada da função em algum ponto dentro do intervalo de integração. Esse coeficiente pode ser conhecido, porém o ponto da segunda derivada não, de forma que apenas se notará aqui o erro como $O(\ )$, independente do coeficiente.

A eq. (3.8.12), da Regra do Trapézio, é uma fórmula de dois pontos (x_1 e x_2), exata para polinomiais de grau um ou maior, como $f(x) = x$. Pode-se antecipar que existe uma fórmula de três pontos exata para polinomiais de grau 2; porém, devido à

simetria entre os lados direito e esquerdo desta fórmula, pode-se cancelar coeficientes e esta fórmula de três pontos passa a ser exata para polinômios de grau 3 ou superior, $f(x) = x^3$, como abaixo:

Regra de Simpson:

$$\int_{x_1}^{x_3} f(x)dx = h\left[\frac{f_1}{3} + \frac{4f_2}{3} + \frac{f_3}{3}\right] + O(h^5 f^{(4)}) \qquad (3.8.13)$$

onde f_1, f_2 e f_3 são os valores assumidos pela função f em x_1, x_2 ($x_1 + h$) e x_3 ($x_2 + h$), respectivamente, e $f^{(4)}$ é a derivada quarta da função, calculada em um ponto desconhecido do intervalo. Nota-se que a fórmula fornece a integral sobre um intervalo de tamanho $2h$.

Regra de Simpson 3/8:

Não há cancelamento fortuito na fórmula de quatro pontos (eq. 3.8.14), portanto ela é exata para polinomiais de grau maior ou igual a 3:

$$\int_{x_1}^{x_4} f(x)dx = h\left[\frac{3f_1}{8} + \frac{9f_2}{8} + \frac{9f_3}{8} + \frac{3f_4}{8}\right] + O(h^5 f^{(4)}) \qquad (3.8.14)$$

Regra de Bode:

A fórmula de cinco pontos se beneficia de cancelamento:

$$\int_{x_1}^{x_5} f(x)dx = h\left[\frac{14f_1}{45} + \frac{64f_2}{45} + \frac{24f_3}{45} + \frac{64f_4}{45} + \frac{14f_5}{45}\right] + O(h^5 f^{(4)}) \qquad (3.8.15)$$

sendo exata para polinomiais de grau maior ou igual a cinco.

3.9 – Conclusão

Foram apresentados, neste capítulo, as soluções encontradas para a implementação computacional do modelamento descrito no Capítulo 2. Foram também apresentados os métodos numéricos utilizados para a resolução das equações propostas.

CAPÍTULO 4 - O Programa SOASim

4.1 - INTRODUÇÃO

Nesse capítulo são apresentados o arquivo principal, em linguagem C++, chamado *Unidade1.cpp*, do projeto *SOASim1*, desenvolvido utilizando-se o software *C++Builder3*, da empresa *Borland* (*Inprise Corp.*), e sua interface gráfica correspondente, que provê acesso aos parâmetros de simulação. Os demais arquivos relativos ao projeto não serão apresentados por não terem ligação direta com as funções do programa, funcionando apenas como arquivos de gerenciamento no funcionamento do projeto.

Na Seção 4.2, são apresentadas as janelas (painéis) da interface amigável com o usuário e o procedimento de utilização destas.

Na Seção 4.3 é apresentada a conclusão referente a este capítulo

O programa em C++ *Unidade1.cpp* é apresentado no Apêndice A, onde estão contidas as rotinas de programação, juntamente com comentários relativos à execução de ações pelo programa.

Para uma descrição mais detalhada dessas rotinas, consultar o Capítulo 3.

4.2 - INTERFACES AO USUÁRIO
PROCEDIMENTOS DE UTILIZAÇÃO

Nessa seção são apresentadas as interfaces tipo *windows* utilizadas no projeto para facilitar a utilização do programa pelo usuário. Os objetos gráficos, como caixas de texto, botões e menus, são pré-construídos e fornecidos pelo ambiente amigável do *C++Builder3*. Toda a interface utiliza o idioma Inglês, no intuito de facilitar sua possível utilização em outras partes do globo. O diretório utilizado pelo programa para estocar os resultados de simulação é "*C:\SOAsimulator*", devendo pré-existir à execução do programa. Para analisar esses resultados, o usuário deve utilizar um software gráfico de auxílio, como por exemplo o *Origin4.1*.

Na Figura 4.2.1 é apresentada a tela inicial do programa. Para iniciar-se uma simulação é necessário selecionar, no menu principal, o comando "*Simulation -> New*", que habilitará o restante do menu e também os painéis auxiliares onde podem ser modificados os parâmetros de simulação.

Após esse comando, o painel correspondente ao sinal de entrada do SOA aparece, como na Figura 4.2.2. Nela são mostrados os parâmetros de um trem de pulsos super-gaussiano a ser criado pelo programa. Tais parâmetros são: comprimento de onda (seleção entre 1,3 e 1,55 μm), máxima e mínima potência do pulso óptico [μW], taxa de transmissão [MHz], número de pulsos do trem e formato super-gaussiano; e, inicialmente, aparecem com seus valores *default*, podendo ser todos editados pelo usuário.

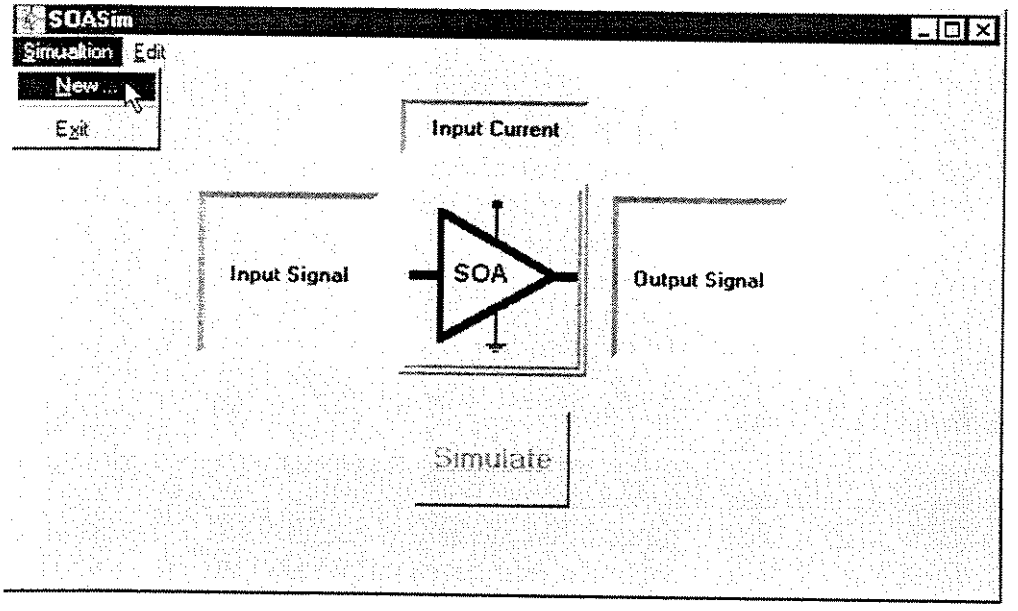

Figura 4.2.1: tela inicial de apresentação do programa SOASim

Inicialmente, apenas o botão "*Create input*" encontra-se habilitado. Ao acionar-se esse botão (com o *mouse* ou selecionando com a tecla *TAB* e ativando com *Enter*) são criados um vetor temporal *tj[i]*, contendo a discretização do tempo, e um vetor *bj_in[i]*

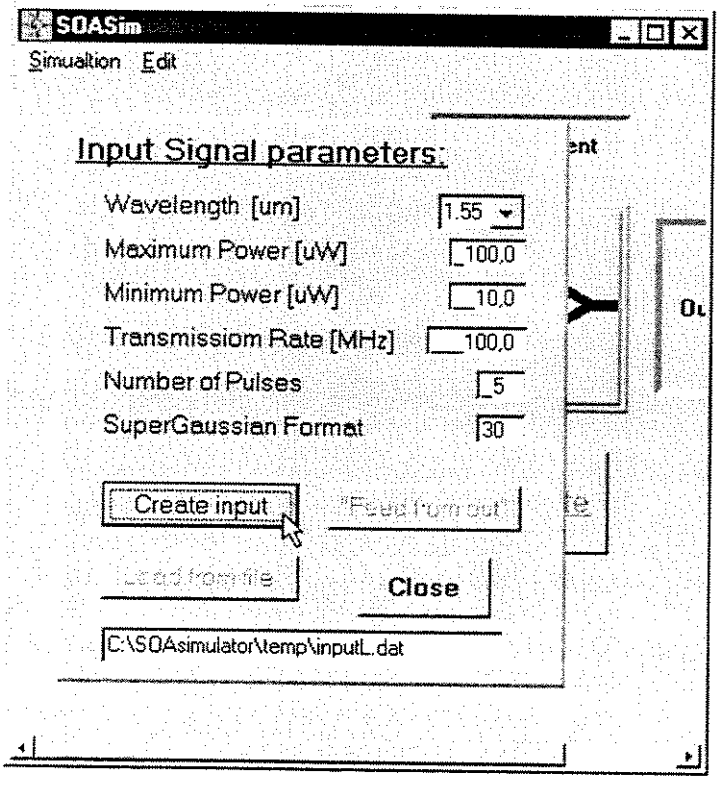

Figura 4.2.2: painel do sinal de entrada

com os correspondentes valores de potência para um trem de pulsos super-gaussianos, com os parâmetros definidos anteriormente (função *"create_pulse( )"*). Um vetor com a norma desse sinal de entrada também é criado (*norma_in[i]*) e ambos, sinal e sua norma, são arquivados em forma de arquivos ASCII, com extensão ".dat", no diretório "*C:\SOAsimulator*", em conjunto com o vetor temporal, sob os nomes respectivos de *input1* e *normain1*. O número no final do nome do arquivo corresponde ao número do amplificador numa possível cascata, como será visto mais adiante.

Completada essa função, esse painel é ocultado e aparece o painel de edição da corrente de alimentação (fig. 4.2.3), comentada adiante. Após essa primeira simulação, o painel de edição do sinal de entrada pode ser novamente ativada como feito anteriormente ou clicando-se duas vezes sobre o quadrado *Input Signal* ou selecionando-se no menu principal o comando "*Edit -> Input signal*". Feito isso, o botão "*Load from file*" aparecerá habilitado, podendo então o sinal de entrada ser carregado do arquivo "*C:\SOAsimulator\temp\inputL.dat*", calculada sua norma e gravados no diretório de resultados como *inputLd* e *normaLd*, respectivamente (função *"loadinputpulse( )"*).

O último botão ainda desabilitado nesse painel, "*Feed from out*", só será habilitado depois de uma primeira simulação e conseqüente geração do sinal de saída. Então, ao apertar esse botão o contador da cascata, inicialmente de valor 1, é incrementado e o resultado de saída da última simulação é copiado para entrada e são gravados os arquivos *input2* e *normain2* (função *"refeed( )"*). Esse procedimento é utilizado quando se deseja simular uma cascata de SOAs. Exemplos são apresentados no Cap. 5.

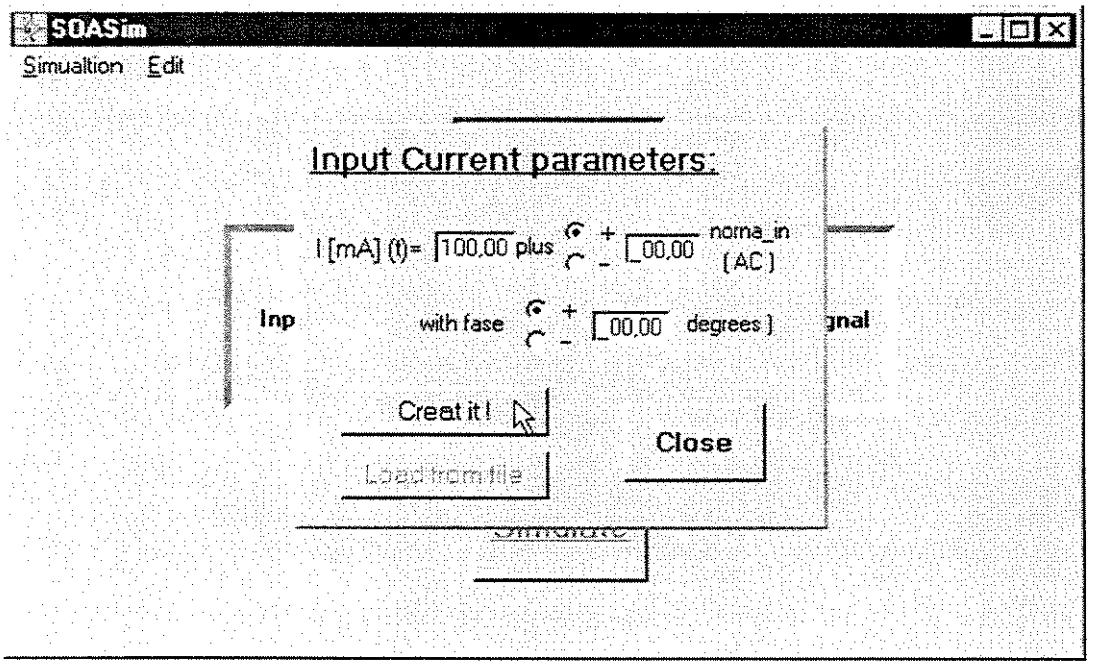

Figura 4.2.3: painel da corrente de alimentação

Na Figura 4.2.3 é apresentado o painel da corrente de alimentação do SOA, onde aparecem janelas com valores correspondes à sua componente DC [mA], ao ganho K [mA] que multiplicará a norma do sinal de entrada diminuída do seu valor eficaz no período, e à defasagem δ [graus] que o sinal de corrente terá com relação ao sinal de entrada. Esses dois últimos parâmetros podem ter seus valores selecionados como positivos ou negativos, através de um "click" na opção desejada. O botão "*Creat it* ", que ao ser apertado gera a criação do sinal de corrente *Ij[i]* de acordo com os parâmetros escolhidos e arquiva sob o nome de *Iin1* (função *"current( )"*), somente aparece habilitado depois da criação de um sinal de entrada. Depois de criado o primeiro sinal de corrente, o botão "*Load from file*" do painel de corrente também é habilitado e pode-se, acionando-o, carregar do arquivo "*C:\SOAsimulator\temp\IinL.dat*" um sinal de corrente criado/modificado de acordo com a necessidade do usuário (função *"loadcurrent( )"*), como pôde ser feito também para o sinal de entrada.

Criado o sinal de corrente, seu respectivo painel é ocultado e aparece o painel correspondente aos parâmetros do SOA, como na Figura 4.2.4. Na Tabela I são apresentados os parâmetros do amplificador a ser simulado que podem ser modificados nesse painel; seus símbolos correspondentes (utilizados nas notações matemáticas do Cap. 2); assim como suas variáveis no programa (apresentado no Apêndice A), com respectivos valores de inicialização (*default*).

TABELA I: parâmetros modificáveis

Parâmetro	**Símbolo**	**Variável correspondente**	**Valor default**
densidade de portadores na transparência	N_0	Nt	$2.10^{18}/cm^3$
ganho da seção transversal	a	a	2.10^{-16} cm^2
coeficiente de atenuação do material	α_{in}	alfa	20 /cm
Refletividade das facetas	R_1, R_2	R	0.0001
fator de aumento de largura de linha	β_c	beta	5
índice de refração efetivo	n_{efc}	nsc	3.4
fator de confinamento	Γ	gama	0.4
perda de inserção	α_{acp}	inloss	5 dB
Coeficientes de recombinação de portadores	*B (radioativo)* *C (Auger)*	B C	$10^{-10}cm^3/s$ $10^{-29}cm^6/s$
dimensões da cavidade ativa	*w (largura)* *d (altura)* *l (comprimento)*	w d l	1.40 μm 0.20 μm 350 μm

De acordo com a escolha destes parâmetros resultam os valores das características (função *"data( )"*, Apêndice A) para o dispositivo (equacionamento na Seção 2.2), apresentados na Tabela II.

TABELA II: parâmetros resultantes

Parâmetro	Símbolo	Variável Correspondente	Valor default
volume da cavidade ativa	V	v	98 μm^3
energia de saturação	E_{sat}	Es	4.491 pJ
ganho estático de pequenos sinais	g_0	g0	35.34 dB
perda interna na cavidade	α_{int}	Loss	3.04 dB
corrente de transparência	I_0	i0	40.78 mA
tempo de vida dos fótons	τ_p	Talp	0.40 ps

A variável tempo de vida dos portadores elétricos τ_c [ps] tem seu valor default igual a 750 ps e opção de variação dinâmica não ativada de modo a permanecer fixa durante a simulação. Esse valor também pode ser modificado pelo usuário e, através da marcação no "*checkbox*" correspondente, calculado a partir dos valores escolhidos (eq. 2.2.25) no painel, passando a ser uma variável dinâmica na simulação, possuindo variação temporal de acordo com a densidade de portadores. Para os valores *default* acima, seu valor inicial é de 769,23 ps, apresentado na janela correspondente no painel de parâmetros.

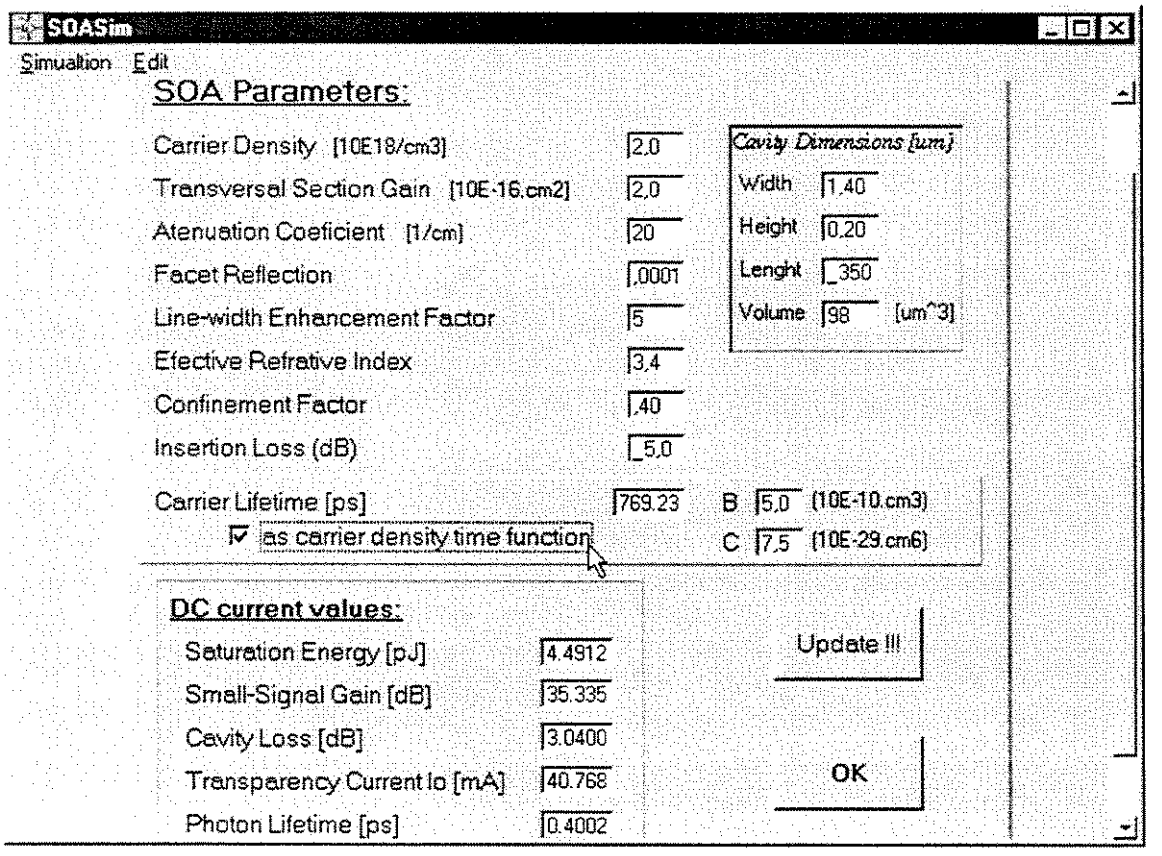

Figura 4.2.4: painel dos parâmetros do SOA

Após serem devidamente escolhidos os parâmetros do SOA, o botão "*Update!!!*" deve ser acionado para que os dados sejam atualizados e os novos parâmetros mostrados ao usuário. Estando de acordo com os novos parâmetros, deve-se fechar o painel utilizando o botão "*OK*".

Para realizar a simulação com o sinal de entrada, de corrente e parâmetros escolhidos, deve-se acionar o botão "*Simulate*". Feito isso, o programa calcula o sinal de saída resultante *bj_out[i]*, sua norma *norma_out[i]*, o ganho do SOA *gain[i]*, a diferença de fase entre sinal de entrada e saída *phshift[i]* e, se escolhida a opção de variação dinâmica do tempo de vida dos portadores elétricos, sua variação com o tempo *talcj[i]*. Esses resultados são armazenados no diretório de resultados com os respectivos nomes: *output1, normaout1, gain1, phaseshift1* e *tal1* (chamada das funções *"g_0_z( )"*, *"integral( )"* e *"calculo( )"*). A rotina de simulação também arquiva em um arquivo *.txt* todos os dados referentes à simulação, de nome *data1* (função *"savedata( )"*). Novamente, o número 1 corresponde ao número do amplificador na cascata, sendo incrementado cada vez que se utiliza o sinal de saída de uma simulação como entrada de outra. Realizadas tais funções, aparece uma janela informando que todos os resultados foram calculados e armazenados no diretório de resultados.

Para se fechar o programa, basta acionar o *"X"* (Alt + F4) no canto superior direito da tela do simulador ou selecionar o comando *Simulation -> Exit...* do menu principal.

4.3 - CONCLUSÃO

Neste capítulo foram apresentadas as interfaces de utilização do *SOASim*, bem como suas rotinas de programação em C++. Procurou-se explicar o procedimento de utilização da maneira mais clara possível, de forma que o usuário possa facilmente operar o simulador.

Apesar de bem amigável, a interface com o usuário pode ainda ser melhorada, acrescentando-se janelas gráficas para exposição dos resultados no próprio programa, otimizando o processo de análise de resultados e escolha de parâmetros. Pode-se também utilizar-se janelas de diálogo para escolha de caminhos para o arquivamento dos arquivos de saída e carregamento de arquivos de entrada. Um futuro trabalho poderia também implementar a possibilidade de simulação de vários amplificadores, nas mais diversas configurações, de uma só vez.

Tais melhoramentos tiveram sua implementação pretendida pelo candidato, porém não foram feitas devido à pouca familiaridade com o ambiente de programação, bem como de suas estruturas de classes de objetos, ocorrendo incompatibilidades entre as variáveis funções pré-implementadas no *C++Builder3* e a linguagem C++ básica utilizada nas rotinas. Para possibilitar maiores facilidades de programação há a necessidade do conhecimento das estruturas pré-contruídas e da geração, a partir destas, de novas estruturas dedicadas aos interesses de simulação.

CAPÍTULO 5 - RESULTADOS SIMULAÇÃO E EXPERIMENTO

5.1 - INTRODUÇÃO

Nesse capítulo são apresentados resultados obtidos em simulações utilizando-se o SOASim, com auxílio suplementar do software gráfico Origin4.1. São também apresentados os resultados obtidos em experimentos realizados no Laboratório de Comunicações Ópticas e Celulares (LAPCOM) "Prof. Dr. Atílio J. Giarolla", do Departamento de Microondas e Óptica - FEEC - Unicamp. Esses resultados experimentais são utilizados para a convalidação do modelo utilizado e investigação de suas limitações, bem como para calibração dos parâmetros físicos do dispositivo simulado. Para evitar repetições, todos os resultados se referem a portadoras ópticas em 1550 nm, na terceira janela de transmissão da fibra de sílica, mais utilizada em comunicações a longa distância por apresentar as menores perdas por atenuação.

5.2 - AMPLIFICAÇÃO EM CASCATA

Nessa seção são apresentados resultados experimentais de amplificação em uma cascata de amplificadores ópticos a semicondutor, comparados com suas respectivas simulações, no intuito de convalidar o modelo utilizado no programa SOASim, bem como averiguar suas limitações.

No experimento apresentado foi utilizada a montagem experimental da Figura 5.2.1, onde uma cascata de três amplificadores ópticos a semicondutor (*ETEK-HSOA-1550*) sob controle de temperatura e polarizados de maneira a otimizar o ganho de pequenos sinais, são utilizados na tentativa de apagamento de informação da portadora óptica através do fenômeno de compressão de ganho [**20**].

Nessa montagem, cada amplificador é seguido de um isolador óptico (47 dB de isolação, 1 dB de perda de inserção) e de um filtro óptico passa-faixa (4 dB de perda de inserção, 30 dB de isolação máxima), além de um controle de polarização (não indicado na figura). A portadora óptica foi gerada por um laser de cavidade externa (*Photonetics/Nanotunics*) e modulada em amplitude por modulador de amplitude Mach-Zender (*UTP-APE MZM-3.0GHz*) alimentado por gerador de bits em seqüência pseudo-aleatória e por uma fonte de DC (para controle da razão de extinção). Os pulsos de saída foram analisados em um analisador óptico para sinais de comunicação com 30 GHz de banda (*HP-83480A/HP83482A*).

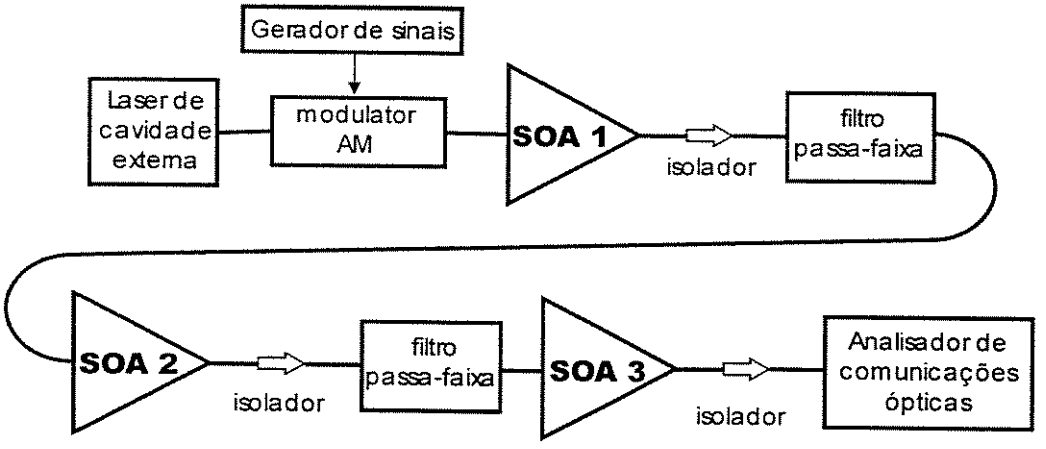

Figura 5.2.1: Montagem experimental para teste de apagamento de informação da portadora óptica através da compressão de ganho do SOA

A tentativa do apagamento da informação presente numa portadora óptica por compressão de ganho utiliza o fenômeno da saturação do ganho em amplificadores ópticos a semicondutor [**20**], isto é, a diminuição do ganho de amplificação de um sinal óptico quando a potência deste excede ao que se chama potência de saturação. Um exemplo desse fato pode ser observado na Figura 5.2.2, onde temos curvas teóricas do ganho de um SOA versus potência óptica de entrada, em escala logarítmica, de um trem de pulsos em 100 MHz, para quatro correntes injetadas. Esse fato ocorre devido a energia finita, determinada pela quantidade de portadores elétricos em estados energéticos mais elevados que o do fóton do sinal, que o amplificador pode acrescentar ao sinal durante a sua passagem pela cavidade ativa.

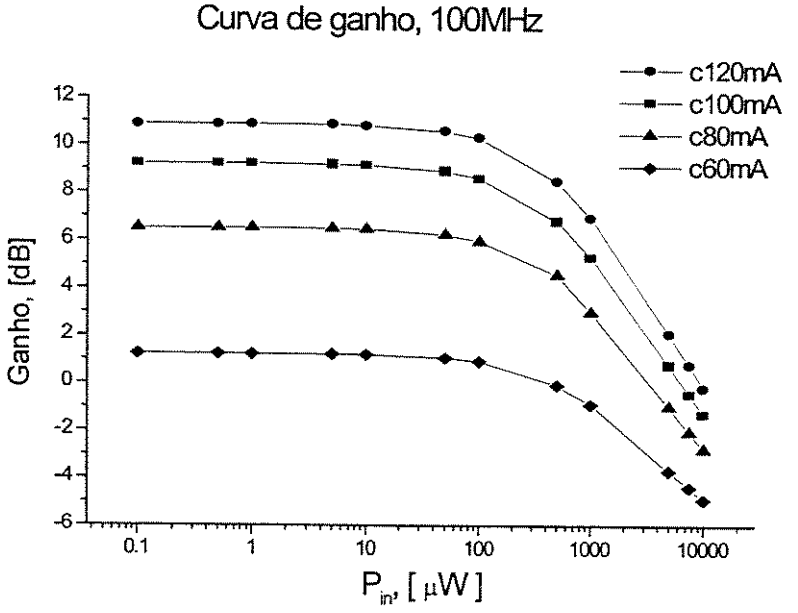

Figura 5.2.2: Saturação de ganho.

Transformando os dados da Figura 5.2.2 de forma a termos potência de saída versus potência de entrada, construiu-se a curvas da Figura 5.2.3. No gráfico dessa figura são apresentadas também três retas correspondentes a amplificação com ganho igual a 10 (10 dB), igual a 1 (0 dB) e igual a 0.2 (-7 dB), de forma que podemos comparar a evolução do ganho das curvas teóricas para as diversas correntes de injeção, a medida que se aumenta a potência do sinal de entrada.

Apesar do gráfico na Figura 5.2.3 referir-se a valores estáticos, pode-se utiliza-lo para exemplificar o apagamento por compressão da ganho. Passando um sinal com razão $(P_1/P_0)_0 = 10$ (pulso *1 in* na Figura 5.2.3) por um SOA teórico polarizado com I = 120 mA, obtêm-se na saída o sinal *1 out* (igual à *2 in*), com razão $(P_1/P_0)_1 = 2.23$, quase 4.5 vezes menor que anterior. Passando novamente por um SOA, agora polarizado com 100 mA, obtêm-se o sinal *2 out* (igual à *3 in*), já com razão $(P_1/P_0)_2 = 1.42$, 1.6 vezes menor que a anterior. Passando novamente o sinal por um SOA igual ao anterior, obtêm-se o sinal *3 out*, com razão $(P_1/P_0)_3 = 1.11$, quase dez vezes menor que a razão inicial.

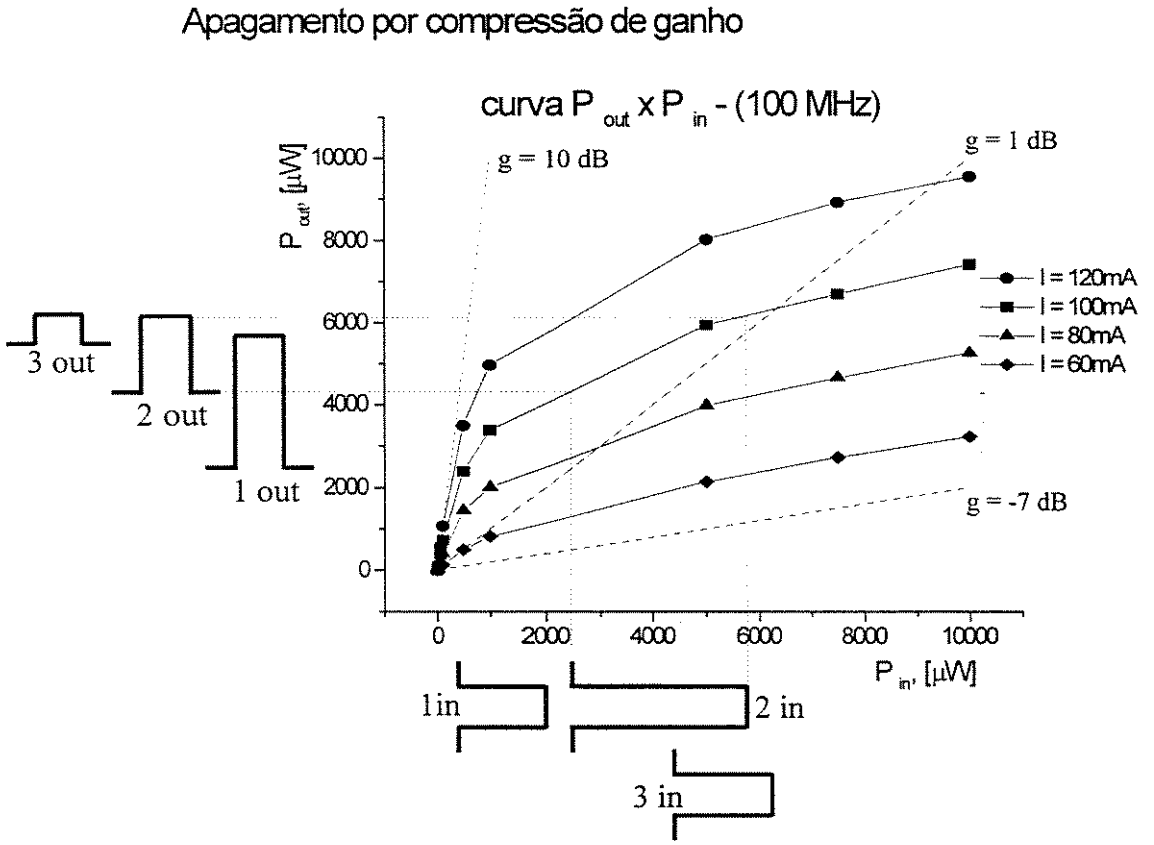

Figura 5.2.3: Tentativa de apagamento por compressão de ganho.

Pretendía-se dessa forma implementar o apagamento da informação presente numa portadora óptica. Porém, os níveis de potência necessários para operar os dispositivo no regime de saturação necessário são demasiadamente altos para a operação prática, podendo mesmo danificar os dispositivos. Outro problema é o comportamento dinâmico do ganho, que também limita esse tipo de apagamento no domínio da freqüência.

Na Figura 5.2.4 é apresentado o pulso óptico de entrada na cascata experimental apresentada na Figura 5.2.1, além da entrada teórica formada por um pulso super-gaussiano ajustado ao sinal experimental. Pode-se notar boa concordância entre teoria e experimento. Para uma maior clareza na visualização, o sinal experimental foi ligeiramente deslocado para cima e para a direita. Os parâmetros desse pulso de entrada são: taxa de 13Mb/s, nível lógico baixo (bit 0) em 40 μW, nível lógico alto (bit 1) em 138 μW, com uma razão de extinção (*extiontion ratio,* ER=10.log[P_0/P_1]) de -5,4 dB.

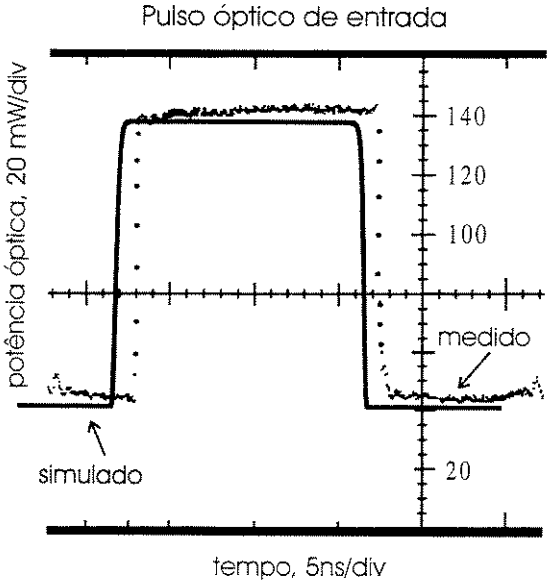

Figura 5.2.4: Pulsos ópticos de entrada em 13Mb/s, teórico e medido.

Para operarem-se os amplificadores em regime de saturação profunda, os mesmos foram polarizados em 160, 45 e 30 mA, apresentando ganhos de pequenos sinais iguais a 11,7, 3,7 e 1,3 dB, respectivamente. Desta forma, o primeiro SOA trabalhou com ganho elevado, de forma a fornecer um sinal de saída com potência elevada, que foi aplicado ao segundo amplificador da cascata. Este, por sua vez, foi polarizado com baixa corrente, de forma que operou em regime de saturação mais profunda. O mesmo ocorre no terceiro amplificador da cascata.

O pulso de saída da cascata é apresentado na Figura 5.2.5, bem como o resultado da simulação, onde os níveis lógicos do resultado experimental passam para 455 μW (bit 0) e 785 μW (bit 1), resultando numa razão de extinção ER= -2,4 dB, 3 dB maior que a anterior, diminuindo a diferença relativa entre os níveis de potências dos níveis lógicos do bit.

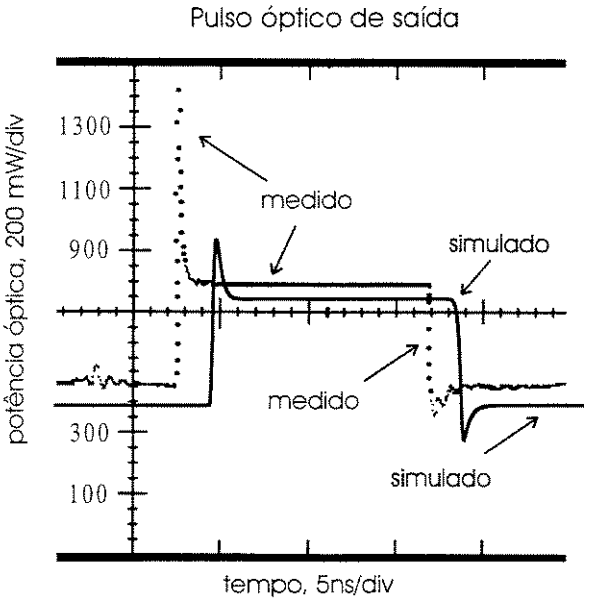

Figura 5.2.5: Pulso de saída, simulado e medido, em 13Mb/s, após cascata de amplificadores .

O resultado da simulação, observado na Figura 5.2.5, apresenta boa concordância com os resultados experimentais no que tange a níveis lógicos e formato do sinal, mas diverge significativamente no que diz respeito à amplitude do ruído de *overshoot*, que no experimento apresenta valores duas vezes maiores que na simulação. Tal divergência poderia ser explicada por fatores relativos ao ruído de emissão espontânea (*ASE*), não considerados pelo *SOASim*. Outra hipótese, apresentada nos comentários referentes à figura 5.5.10, sugere que a amplitude do ruído de *overshoot* estaria ligada ao tempo de relaxação do laser.

No mesmo experimento, com sinal agora com taxa de 35Mb/s, tem-se o diagrama de olho do sinal de entrada na cascata na Figura 5.2.6, onde o nível lógico baixo (P_0) fica em torno de 20 μW e o nível alto (P_1)em torno de 172 μW.

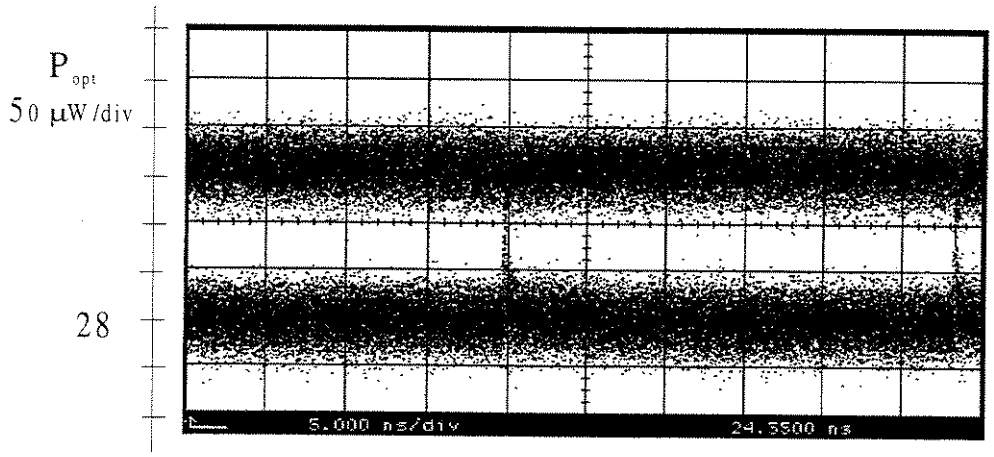

Figura 5.2.6: Diagrama de olho para trem de pulsos de entrada em 35Mb/s
(offset em 28 μW, escala de 50 μW/div, P_0 = 20 μW, P_1 = 172 μW)

Na saída da cascata, operada nas mesmas condições de polarização para operação em saturação profunda, obteve-se o resultado ilustrado por diagrama de olho da Figura 5.2.7.

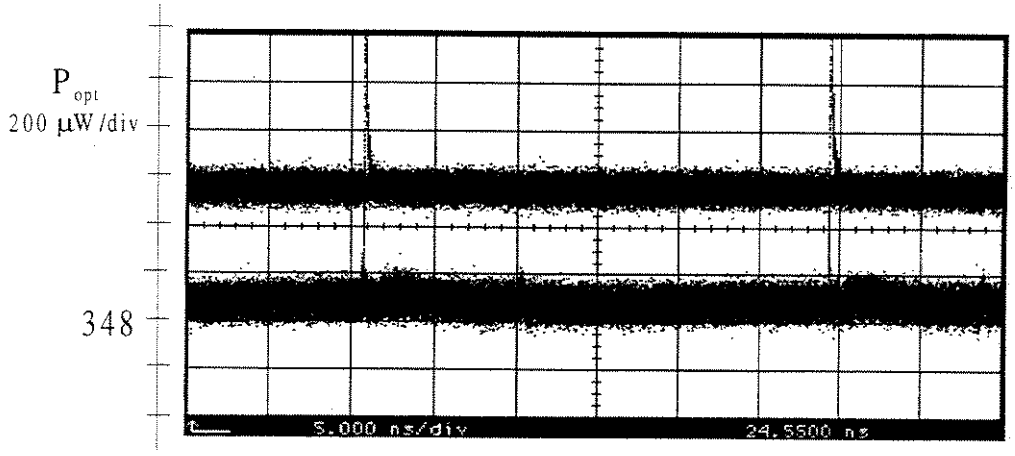

Figura 5.2.7: Diagrama de olho para trem de pulsos de saída em 35Mb/s
(offset em 348 μW, escala de 200 μW/div, P_0 = 423 μW, P_1 = 898 μW)

Para simular a entrada com seqüência de pulso pseudo-aleatória, utilizou-se um trem de pulsos como ilustrado na Figura 5.2.8. Essa seqüência foi formada a partir de 10 pulsos super-gaussianos de grau 40 (form = 40, para possibilitar tempo de subída equivalente ao experimental), com taxa de 17,5 MHz (equivalente a 35 Mb/s), e trabalhados no software

Origin4.1, de forma a aproximar o tempo de duração dos níveis alto e baixo dos pulsos, na forma mais simétrica possível. Esta simulação da seqüência pseudo-aleatória possibilita a observação de possíveis efeitos de acúmulo ou gasto excessivo de portadores devido a uma seqüência repetitiva de "zeros" e "uns" lógicos. A partir da seqüência da Figura 5.2.8, com gráfico construído apenas pelos pontos de discretização do SOASim (sem linhas de união) dividido em períodos de um bit (28,571428 ns) e colados um sobre os outros, construiu-se o diagrama de olho teórico, apresentado na Figura 5.2.9.

O diagrama de olho teórico de entrada (Figura 5.2.9) representa de maneira satisfatória o obtido experimentalmente (Figura 5.2.6), com tempo de subida e descida coincidentes, além dos níveis de potência equivalentes, desprezando-se aqui o ruído. Porém, devido à utilização de pulsos super-gaussianos, as derivadas temporais no início e no fim dos níveis lógicos altos (bits 1) divergem das experimentais (Figura 5.2.6), fato que influenciará no resultado da simulação, apresentados mais adiante, na Figura 5.2.10.

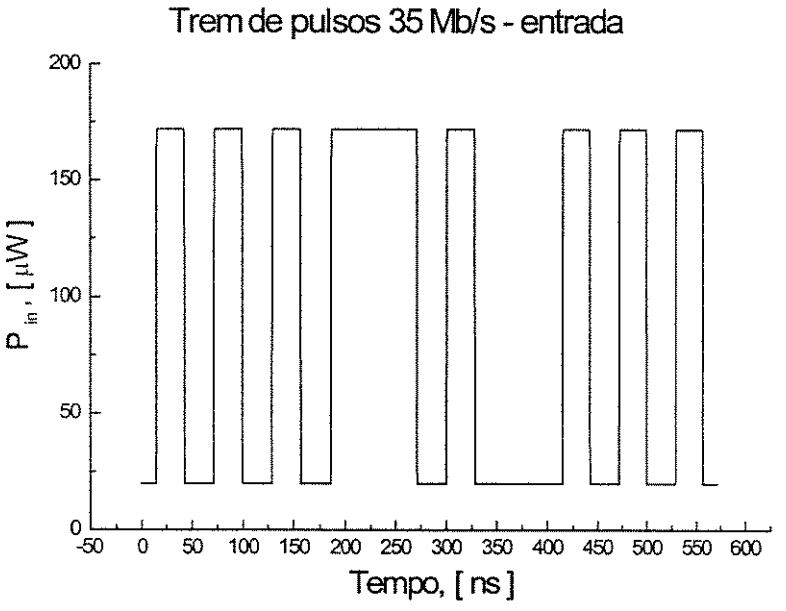

Figura 5.2.8: Trem de pulsos teórico de entrada em 35Mb/s

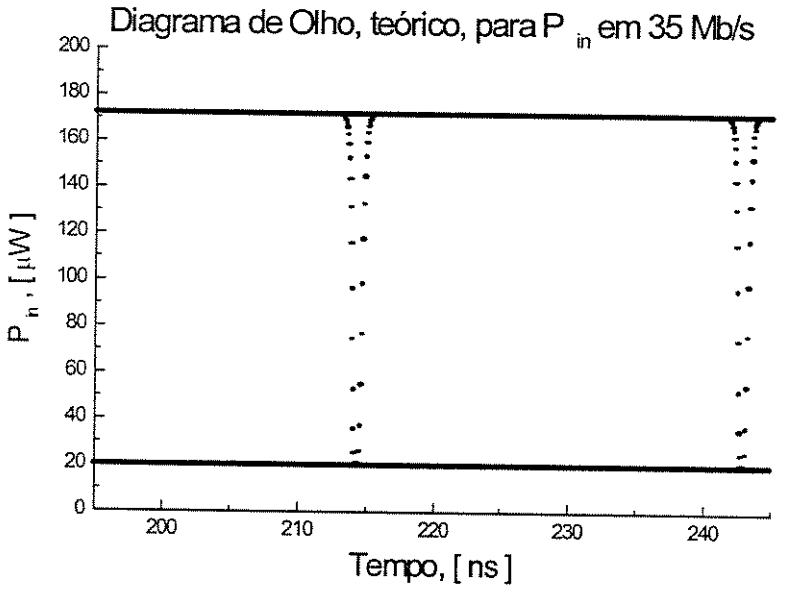

Figura 5.2.9: Diagrama de olho teórico de entrada em 35Mb/s

Utilizando, então, o sinal da Figura 5.2.8 como entrada de uma cascata de três SOAs teóricos com as seguintes características:

- dimensões da cavidade ativa:

largura (w) = 1,4 μm; altura (d) = 0,2 μm;
comprimento (l)= 350 μm, com volume equivalente de 98 μm^3;

- Características do dispositivo ativo:

densidade de portadores na transparência (N_0) = $2x10^{18}/cm^3$;
ganho da seção transversal (σ) = $2x10^{-16}$ cm^2;
coeficiente de atenuação do material (α) = 20 /cm;
refletividade das facetas (R_1=R_2) = 0,0001;
fator de aumento de largura de linha (β) = 5
índice de refração efetivo (n_{efc}) = 3,4;
fator de confinamento (Γ) = 0,4;
perda de inserção = 4 dB;
coeficientes de recombinação de portadores:
radioativo (B) = $5x10^{-10}cm^3/s$;
Auger (C) = $7,5x10^{-29}cm^6/s$;

Estes parâmetros resultam nas seguintes características para o dispositivo: tempo de vida dos portadores elétricos (τ_c) em torno de 770 ps, corrente de transparência (I_0) de 40,8 mA, perda interna na cavidade de 3,05 dB, energia de saturação em 4,5 pJ e tempo de vida dos fótons de 0,4 ps.

Para obter-se um sinal de saída teórico (Figura 5.2.9) bem próximo ao conseguido experimentalmente, foi necessário simular a alimentação dos amplificadores com correntes DC de 120, 70 e 60 mA, respectivamente, para poder conseguir o regime de operação em saturação profunda, além de aumentar a perda por inserção no terceiro e último amplificador de 4 para 7 dB para conseguir-se atingir os valores de potência óptica experimentais. Tais disparidades com os experimentos podem ser atribuídas novamente à não incorporação no modelo dos fenômenos referentes ao ruído de emissão espontânea, bem como do desconhecimento dos parâmetros do dispositivo real. Os parâmetros utilizados na simulação foram calibrados com ajuda de dados experimentais de ganho e tempo de subida do pulso, bem como de valores comumente utilizados na literatura.

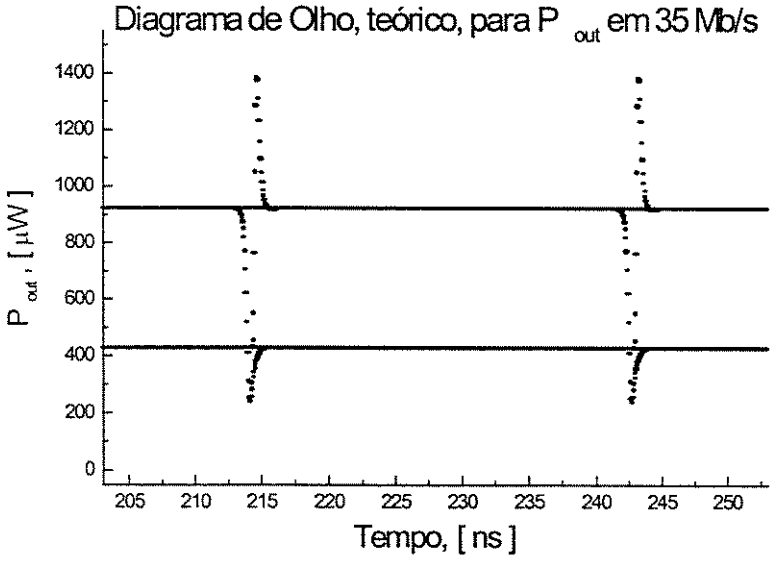

Figura 5.2.10: diagrama de olho teórico de saída para 35Mb/s

Comparando-se os diagramas experimental (Figura 5.2.7) e simulado (Figura 5.2.10) para a saída da cascata de amplificadores, pode-se verificar boa coincidência no que se refere aos tempos de subida e descida do pulso amplificado, além dos patamares de potência óptica coincidentes. Verifica-se, também, que o resultado de simulação, apesar de apresentar o *overshoot* bem próximo do teórico, difere ao apresentar um *undershoot* não notado experimentalmente. Isto ocorre devido, talvez, à grande faixa de ruído presente no diagrama experimental, que impossibilita uma análise mais precisa.

Nessa mesma configuração experimental, utilizou-se uma taxa maior em 2Gb/s, com diagrama de olho do sinal de entrada apresentado na Figura 5.2.11. Após passar pela cascata de amplificadores operados sob saturação profunda, obteve-se na saída o diagrama de olho apresentado na Figura 5.2.12, onde observa-se claramente *overshoots* sobrepostos com amplitude diferenciada, devido ao efeito na dinâmica de ganho devido ao consumo maior ou menor de portadores elétricos por uma seqüência aleatória de "uns" ou "zeros" antes do pulso observado, provocando maior ou menor saturação do ganho do SOA.

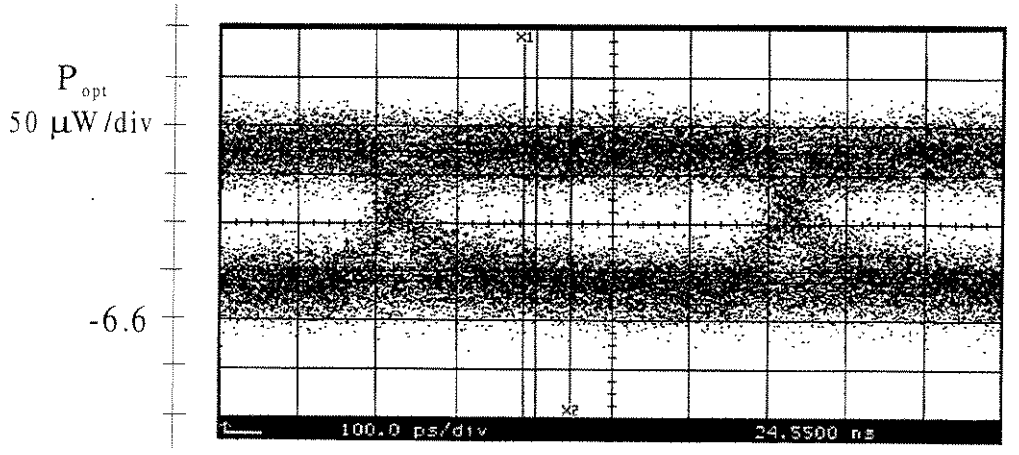

Figura 5.2.11: diagrama de olho para trem de pulsos de entrada em 2Gb/s;
(offset em -6.6 μW, escala de 50 μW/div, P_0 = 32.4 μW, P_1 = 164.6 μW)

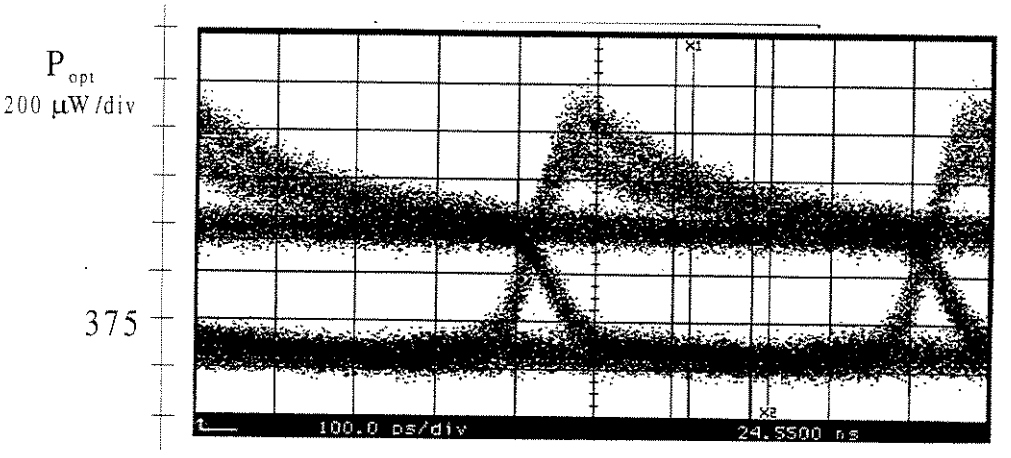

Figura 5.2.12: diagrama de olho para trem de pulsos de saída em 2Gb/s;
(offset em 375.4 μW, escala de 200 μW/div, P_0 = 224 μW, P_1 = 805 μW)

Para realizar a simulação desse experimento, utilizou-se uma seqüência de dez pulsos super-gaussianos em 1GHz (equivalente a 20 bits a 2Gb/s), reformatados individualmente, como feito para o caso anterior (35Mb/s), de forma que a duração do nível alto do pulso seja equivalente à do nível baixo, e trabalhando a duração do símbolo lógico de forma a simular uma seqüência pseudo-aleatória. Esse trem de pulsos de entrada é apresentado na Figura 5.2.13.

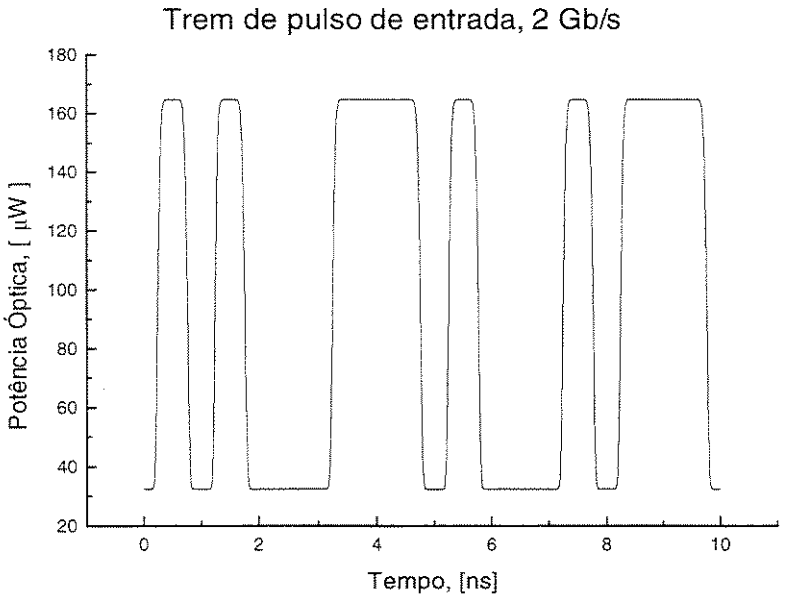

Figura 5.2.13: trem de pulsos teórico de entrada em 2Gb/s

A partir do trem de pulsos ilustrado na Figura 5.2.13, recortados um a um e sobrepostos, formou-se o diagrama de olho teórico da Figura 5.2.14 que, desprezando-se o ruído, ajusta-se precisamente ao seu equivalente experimental (Figura 5.2.11).

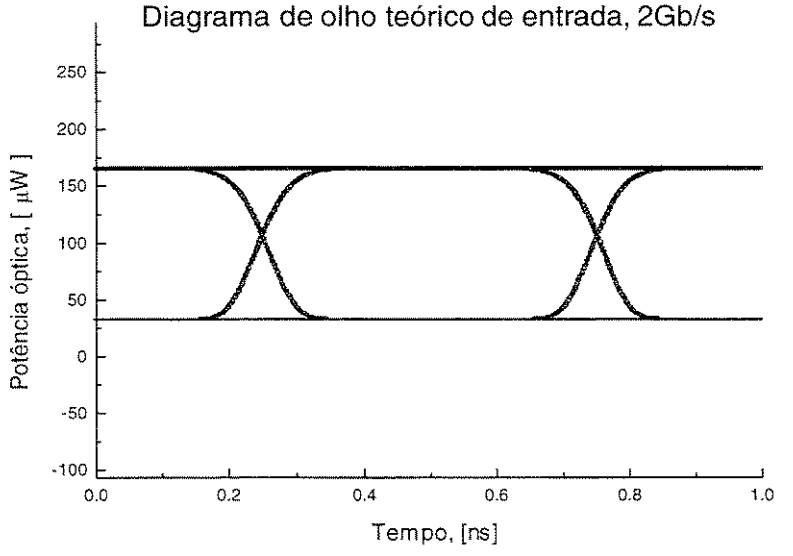

Figura 5.2.14: Diagrama de olho de entrada, 2 Gb/s

Utilizando, então, o sinal da Figura 5.2.13 na entrada da cascata de amplificadores da simulação, com os mesmos parâmetros utilizados no caso de 35 Mb/s e alimentados com correntes de 120, 65 e 49 mA (ajustadas de forma a obter os níveis de potência equivalentes aos conseguidos experimentalmente), obteve-se a resposta ilustrada pelo diagrama da Figura 5.2.15, onde observa-se grande semelhança com o equivalente experimental (Figura 5.2.12) quanto aos níveis de potência, tempos de subída e descida e presença de dois tipos de *overshoot*, dependentes da sequência de bits anteriores ao ilustrado. Nota-se novamente o aparecimento de *undershoot* não notado na resposta experimental, único fato que difere a resposta simulada da experimental, como já observado anteriormente.

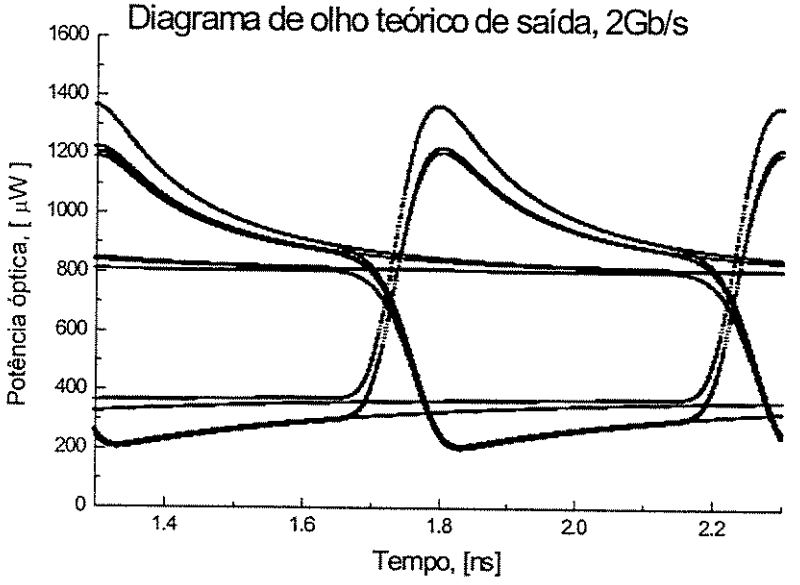

Figura 5.2.15: diagrama de olho de saída teórico, 2Gb/s

A grande quantidade de ruído, devido a esse *overshoot* positivo, introduzido no sinal óptico amplificado por um SOA operado na saturação tem uma solução relativamente simples sugerida na literatura [**26**], onde um interferômetro Mach-Zehnder é utilizado como um discriminador de freqüência e suprime grande parte da distorção introduzida pela amplificação.

Com base nos dados dessa seção, conclui-se que o simulador tem resposta com boa coincidência para os experimentos apresentados, mostrando sua capacidade de considerar o comportamento dinâmico da densidade de portadores no interior da cavidade, quando operada em regime de saturação profunda.

---

26 - Watanabe, T., Yasaka, H., Skaida, N. e Koga, M. - *"Waveform shaping of chirp-controlled signal by semiconductor optical amplifier using Mach-Zehnder frequency discriminator"* - IEEE Photonics Tech. Letters, Vol. 10, N° 10, out./1998.

5.3 - MODULAÇÃO DA CORRENTE INJETADA

Nessa seção são apresentados resultados experimentais, bem como suas respectivas simulações, para o amplificador óptico a semicondutor com modulação da corrente injetada.

O primeiro resultado apresentado refere-se à modulação com onda quadrada em 509 MHz da corrente de alimentação de um SOA com sinal de entrada CW com 800 μW de potência óptica. Na Figura 5.3.1 são apresentados o sinal correspondente à corrente injetada, na parte superior da figura, com escala original com *offset* em -424 mV e escala de 100 mV/div. (equivalente à escala transformada para unidade de corrente utilizada na figura), tempo de subida 753,79 ps; e o sinal óptico de saída na parte inferior da figura, com offset em 596 μW e escala de 100 μW/div.

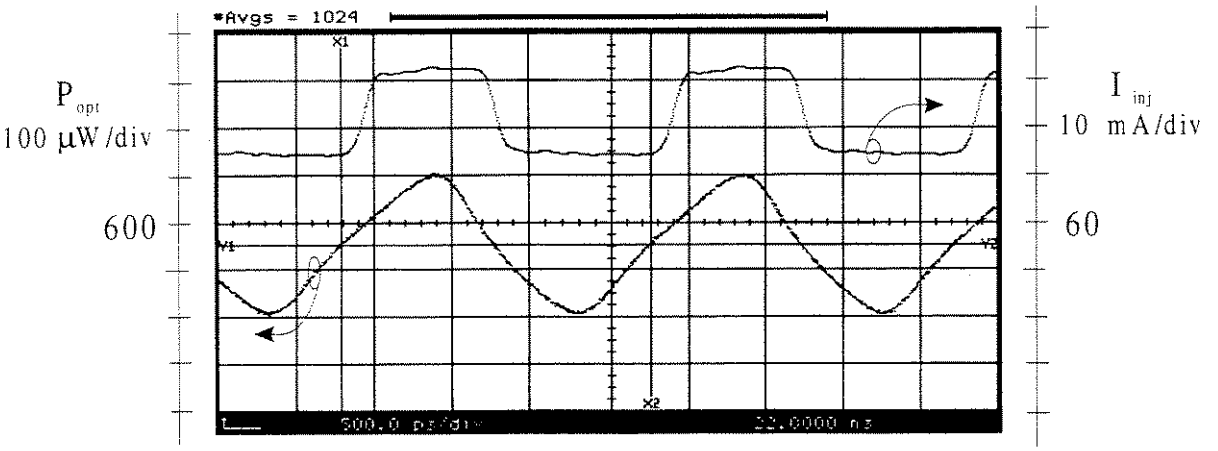

Figura 5.3.1: resultado experimental, modulação da corrente (acima) e sinal de saída (abaixo)

Para simular esse experimento, utilizou-se um SOA teórico com os seguintes parâmetros: $N_t = 2\text{x}10^{18}/\text{cm}^3$, $\sigma_g = 2\text{x}10^{-16}$ cm^2, $\alpha = 20$ cm^{-1}, R_1=R_2=0,0001, β_c=5, n_{eff}= 3,4, $\Gamma = 0.4$, $w = 2$ μm, $d = 0.2$ μm e l = 350 μm, perda de inserção de 5 dB, com fatores de recombinação $B = 5\text{x}10^{-10}\text{cm}^3/\text{s}$ e $C = 7{,}5\text{x}10^{-29}\text{cm}^6/\text{s}$, resultando em um tempo de vida estático dos portadores elétricos (τ_c) de 770 ps e corrente de transparência I_0 = 58.24 mA. Utilizou-se corrente de polarização com componente contínuo Idc = 80 mA e ganho de modulação K = 13 mA com diferença de fase zero ($\delta = 0^o$) (ver eq. (2.2.24)). Com isso, obteve-se a resposta apresentada na Figura 5.3.2, apresentando boa concordância com os dados experimentais (Figura 5.3.1) no que se refere a tempo de subida e níveis de potência óptica, apesar de apresentar derivada temporal do sinal óptico ligeiramente superior à obtida no experimento.

A aparente diferença de fase entre o sinal óptico experimental e o simulado deve-se ao atraso no posicionamento temporal do mesmo quando da realização do experimento, devido ao comprimento óptico adicional dos cabos de fibra óptica do SOA até o analisador

óptico. A fase correta é representada no resultado teórico (Figura 5.3.2), onde o sinal óptico tem sua mudança de sinal da derivada temporal (subida e descida) coincidente com o início da subida e da descida do sinal de corrente.

Esses resultados mostram claramente a resposta aproximadamente triangular do amplificador à modulação em onda quadrada da corrente injetada, já nessa taxa de 0,5 GHz. Isto ocorre devido ao tempo de vida finito dos portadores elétricos no SOA ser grande em relação ao período da onda quadrada utilizada.

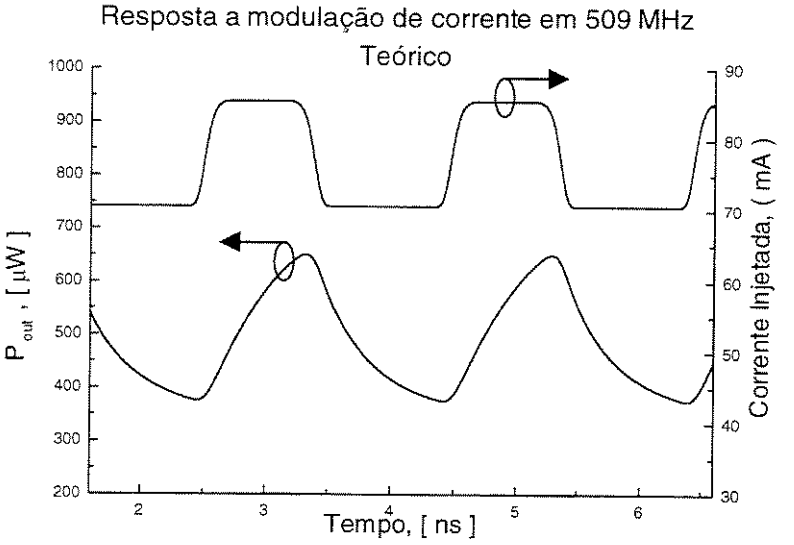

Figura 5.3.2: resposta do SOA a modulação de corrente em 509 MHz, teórico

Devido a essa dinâmica dos portadores, uma melhor forma de utilizar-se o SOA no processamento totalmente óptico de sinais em altas taxas seria em montagem interferométrica (ver Seção 1.3). Neste caso, utiliza-se a diferença de fase, proporcional ao ganho da cavidade, que aparece no sinal óptico (devido à modulação do índice de refração complexo da região ativa com concentração de portadores), na forma da eq. (2.2.20).

O programa *SOASim* tem como um dos resultados de simulação essa diferença de fase entre sinal de entrada e de saída devido ao índice de refração complexo da cavidade ativa do SOA. Aproveita-se, então, o caso em questão para ilustrar capacidades do SOASim em prever tal ocorrência, ilustrando-se, na Figura 5.3.3, a diferença de fase entre o sinal de saída e de entrada para a modulação de corrente em 509 MHz. Nessa Figura observa-se a variação da diferença de fase do sinal em torno de $-\pi$, valor suficiente para se realizar processamento do sinal óptico em arranjos interferométricos [**1**].

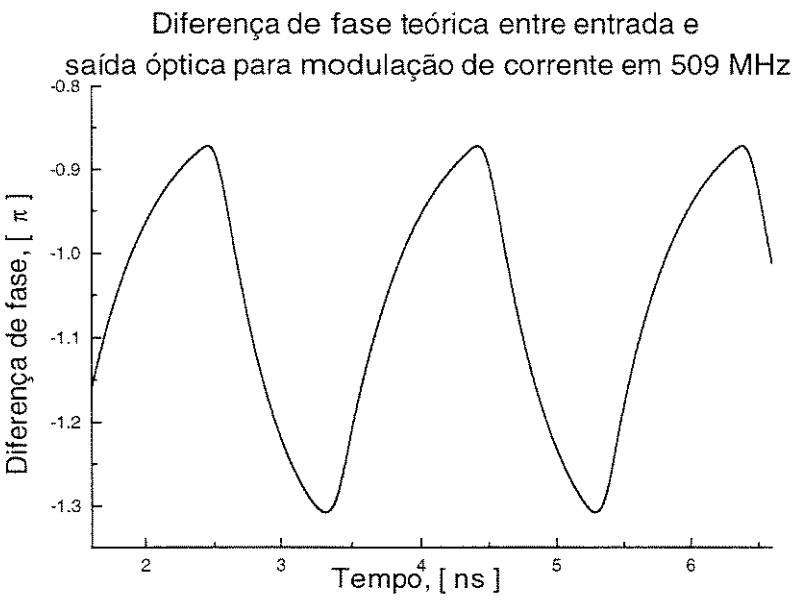

Figura 5.3.3: diferença de fase entre sinal óptico de entrada e de saída, modulação de corrente em 509 MHz

Aproveitando essa mesma simulação, a dinâmica do tempo de vida dos portadores elétricos τ_c dentro da cavidade (eq. (2.2.25)) é ilustrada na Figura 5.3.4, variando de 752 a 740 ps, sempre menor que o valor estático calculado (~770 ps). Apesar de ser pequena (<1,5%), a dinâmica dessa variação é importante dentro do contexto de amplificação com e sem controle dinâmico de ganho, aumentando com os níveis de modulação da corrente injetada e intensidade da saturação do SOA em questão.

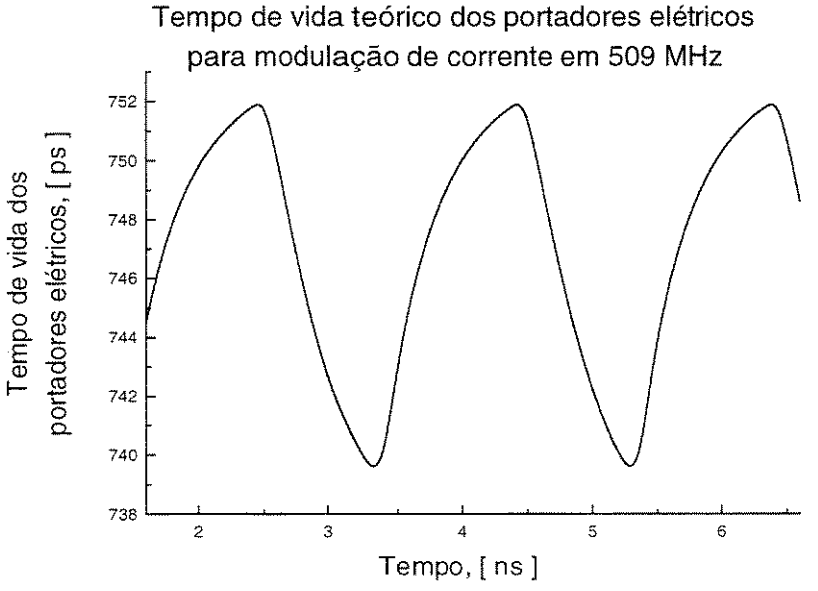

Figura 5.3.4: tempo de vida dos portadores elétricos, modulação de corrente em 509 MHz.

5.4 - CONTROLE DINÂMICO DE GANHO

O controle dinâmico de ganho (*FFG - feed-forward gain control*) do SOA é obtido através da modulação da corrente injetada na cavidade ativa de amplificação. Isso é feito através da introdução de uma componente modulada à corrente de injeção total do SOA, sendo essa modulação feita por um sinal de controle proporcional à componente alternada do módulo do sinal de entrada, de modo que o ganho do SOA passa a variar conjuntamente com o sinal óptico de entrada. Simplificadamente, pode-se dizer que esse sinal de controle impõe um variação positiva ($K>0$ na eq. (2.2.24)) ou negativa ($K<0$) em relação ao nível contínuo da corrente injetada, com uma diferença de fase em relação ao sinal óptico de entrada devidamente escolhida de acordo com o propósito de utilização do dispositivo.

O esquema equivalente de simulação é apresentado na Seção 1.4.

Para implementar o *FFG* na prática, realizou-se experimento com a montagem ilustrada na Figura 5.4.1.

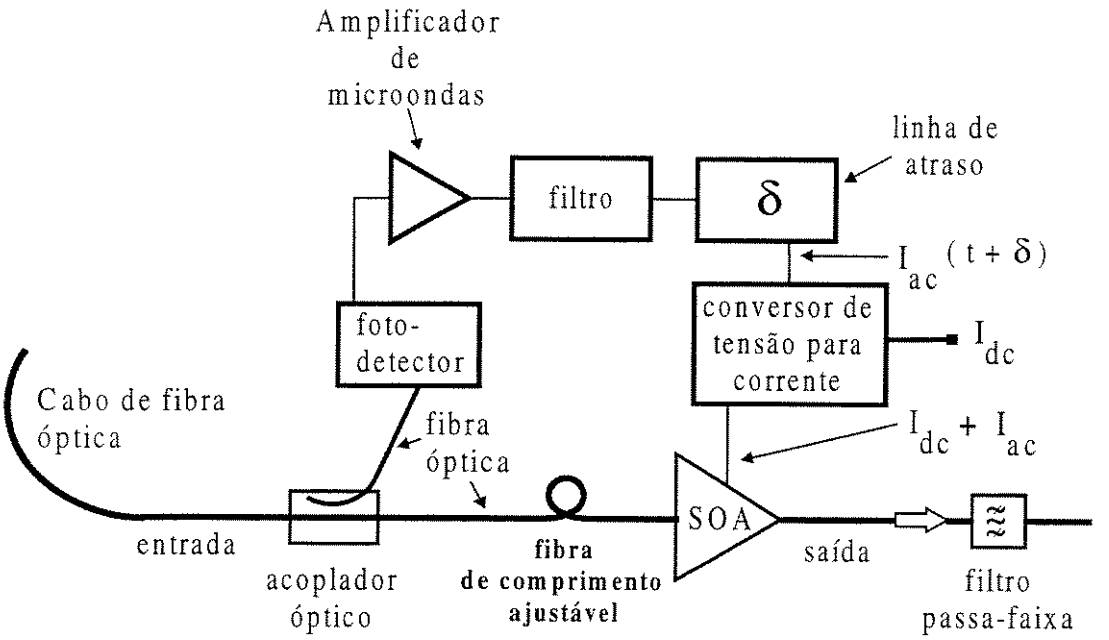

Figura 5.4.1: montagem experimental implementando FFG.

Nessa configuração, o sinal óptico de entrada do SOA é dividido, utilizando-se um acoplador óptico, convertido para sinal elétrico (em um fotodetector) e, posteriormente amplificado. Esse sinal passa, então, por uma linha de atraso e soma-se à corrente de polarização (I_{DC}) em um *bias-T*, com a corrente total sendo injetada no amplificador. O sinal óptico de entrada passa, também, por um fibra de comprimento ajustado adequadamente antes de ser acoplada ao amplificador, para que chegue simultaneamente com o sinal de corrente correspondente. O controle da diferença de fase entre corrente modulada (I_{AC}) e sinal de entrada é conseguido pelo ajuste da linha de atraso da parte elétrica do experimento.

Utilizou-se como sinal de entrada o trem de pulsos em 600 MHz, ilustrado na Figura 5.4.2. Observa-se nessa figura a boa concordância entre o sinal experimental e teórico, exceto pelo *overshoot* presente no sinal do experimento.

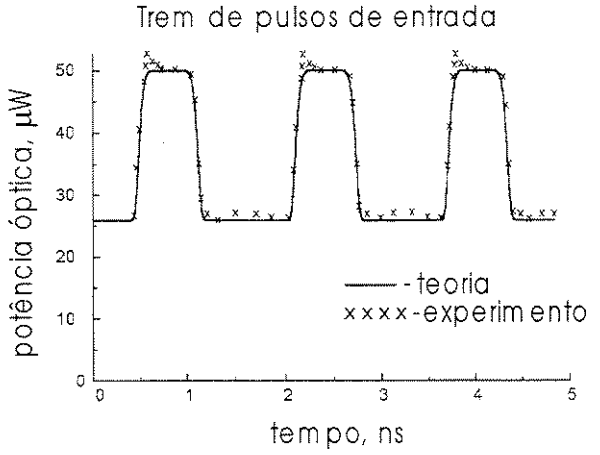

Figura 5.4.2: sinal de entrada em 600 MHz, teoria e experimento.

Após a amplificação com a utilização de FFG positivo ($K > 0$) em fase ($\delta = 0°$), obteve-se o resultado ilustrado na Figura 5.4.3 (experimento e simulação), onde a melhor concordância entre teoria e experimento ocorre quando se assume o tempo de vida dos portadores elétricos τ_c (aqui assumido como o tempo de recuperação de ganho) igual a 400 ps, obtendo boa precisão, exceto por pequena diferença no nível de *overshoot*, devido à sua diferença já presente no sinal de entrada. Obteve-se uma melhora de 3 dB na razão de extinção *ER*, que passou de -2,84 para -5,88 dB.

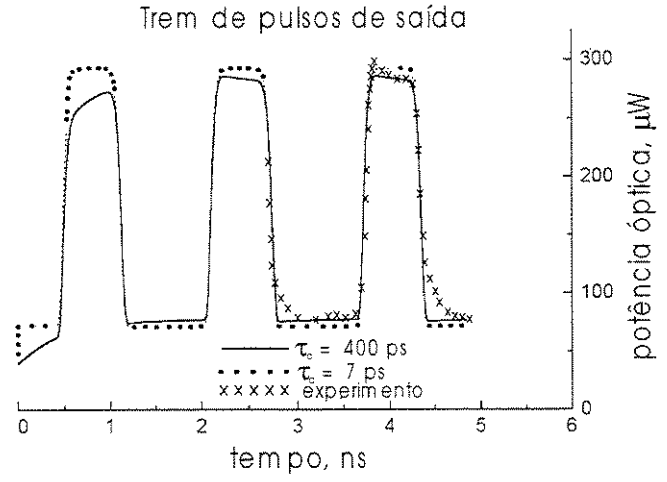

Figura 5.4.3: sinal de saída em 600 MHz, teoria e experimento.

Tal melhora deve-se ao fato do ganho do SOA com essa configuração de *FFG* ser aumentado à medida que também aumenta a intensidade do sinal óptico de entrada. Portanto, ao invés do sinal ter um ganho menor à medida que sua potência aumenta, como comumente ocorre nos SOA devido a saturação do ganho, o sinal passa a ter um ganho maior quanto maior for sua intensidade, devido ao arranjo dessa configuração de *FFG*, com o nível alto do sinal (bit 1) recebendo maior ganho de amplificação e o nível baixo um ganho menor, aumentando então a distinção relativa entre entes níveis.

Para ilustrar os efeitos da diferença de fase δ entre corrente de alimentação e sinal óptico de entrada, utilizou-se um sinal de 400 MHz, como ilustrado na Figura 5.4.4, onde novamente o sinal teórico não apresenta o *overshoot* presente no sinal experimental.

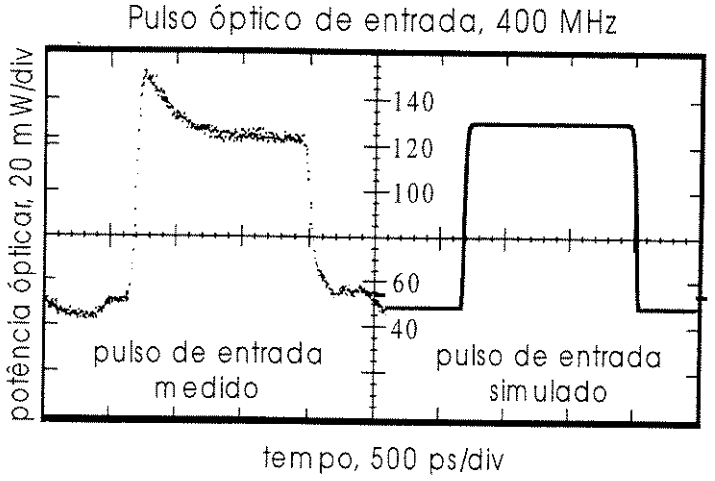

Figura 5.4.4: sinal de entrada em 400 MHz, teoria e experimento.

Após a amplificação, teórica e experimental, com corrente de alimentação do SOA otimizada de modo a tentar apagar a informação e com diferença de fase $\delta = 217.8°$, obteve-se a resposta ilustrada na Figura 5.4.5.

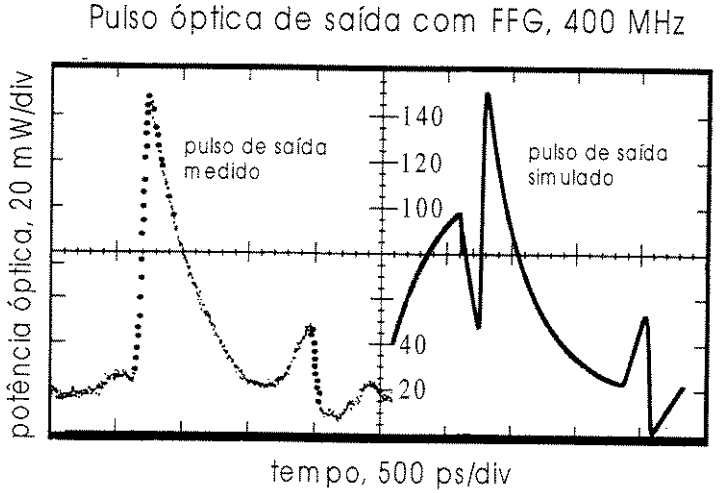

Figura 5.4.5: sinal de saída em 400 MHz utilizando FFG, teoria e experimento.

Nota-se grande concordância entre os resultados experimental e simulado quanto às derivadas temporais e *overshoots*, exceto pelo primeiro, que aparece demasiadamente maior na resposta simulada. Esse fato pode ser explicado pela ausência de *overshoot* nos sinal teórico de entrada, que traria, se fosse considerado, uma maior modulação da corrente de alimentação na simulação.

Com base nos resultados dessa seção pode-se afirmar que o SOASim funciona razoavelmente bem na simulação de FFG, prevendo quase todos os efeitos da modulação de corrente na dinâmica de ganho do amplificador óptico a semicondutor.

5.5 - RESULTADOS DE SIMULAÇÃO

Nessa seção são apresentados resultados de simulação sem suas devidas comprovações experimentais, tentando mostrar as capacidades do controle dinâmico de ganho (*FFG*) do SOA através da adição de uma componente modulada à corrente de alimentação do dispositivo. Essa componente modulada é proporcional à norma da potência do sinal de entrada no SOA multiplicada por um ganho K [mA] e defasada desse sinal de um ângulo δ [graus], como já explicado na Seção 1.4.

O primeiro caso apresentado utiliza taxa de 5 MHz, onde cada ciclo é formado por um pulso super-gaussiano de *form* = 30 (equivalente a uma taxa de 10 Mb/s com um bit igual a um símbolo lógico) de valor lógico alto (bit 1) em 100 μW e nível baixo (bit 0) em 10 μW. O sinal passa por um único SOA, com os seguintes parâmetros: $N_t = 2{,}5\text{x}10^{18}/\text{cm}^3$, $\sigma_g = 2\text{x}10^{-16}$ cm^2, $\alpha = 40$ cm^{-1}, R_1=R_2=0,0001, β_c=5, n_{eff}=3,4, Γ= 0.4, $w = 2$ μm, $d = 0{,}2$ μm e $l = 250$ μm, com fatores de recombinação $B = 1{,}1\text{x}10^{-10}\text{cm}^3/\text{s}$ e $C = 5\text{x}10^{-29}\text{cm}^6/\text{s}$, o que resulta em um tempo de vida dos portadores ópticos (τ_c) de 1,616 ns, valor praticamente instantâneo para a taxa utilizada.

O primeiro resultado, apresentado na Figura 5.5.1, é o de amplificação normal, isto é, sem controle dinâmico de ganho, com corrente de polarização em 42 mA, fornecendo um pulso de saída com um pequeno *overshoot* e alguma compressão de ganho, o que faz com que a razão de extinção piore, passando de -10 dB para -8,75 dB.

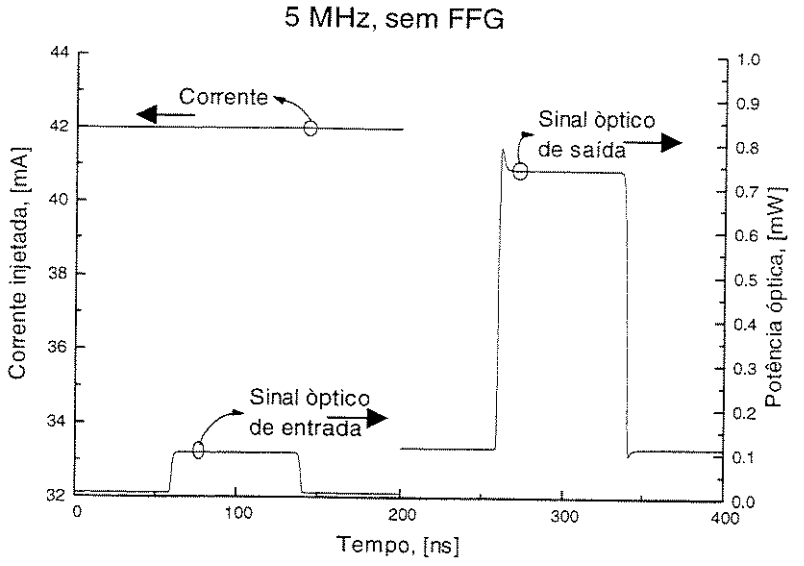

Figura 5.5.1: sinal em 5 MHz, entrada e saída, amplificação normal

Utilizando controle dinâmico de ganho com FFG positiva, isto é, somando-se uma parcela proporcional à norma do pulso de entrada à corrente de polarização do dispositivo, fazendo com que esta corrente e, consequentemente, o ganho do SOA, aumentem à medida que também aumenta a potência do pulso óptico de entrada, obteve-se a resposta ilustrada na Figura 5.5.2. Tem-se, então, um pulso de saída sem *overshoots* e com uma grande melhoria da razão de extinção, que passa dos -10 dB originais para cerca de -24 dB, mostrando as possibilidades de regeneração do sinal conseguidas com o FFG.

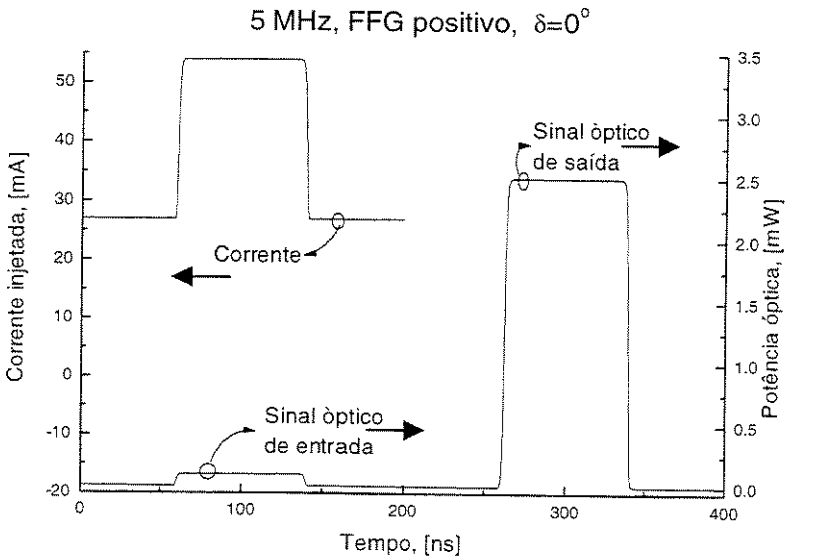

Figura 5.5.2: amplificação com FFG positivo, $\delta = 0°$

Passando ao FFG negativo, onde a corrente de polarização é subtraída de uma componente proporcional à norma do sinal de entrada, obteve-se a resposta ilustrada na Figura 5.5.3.

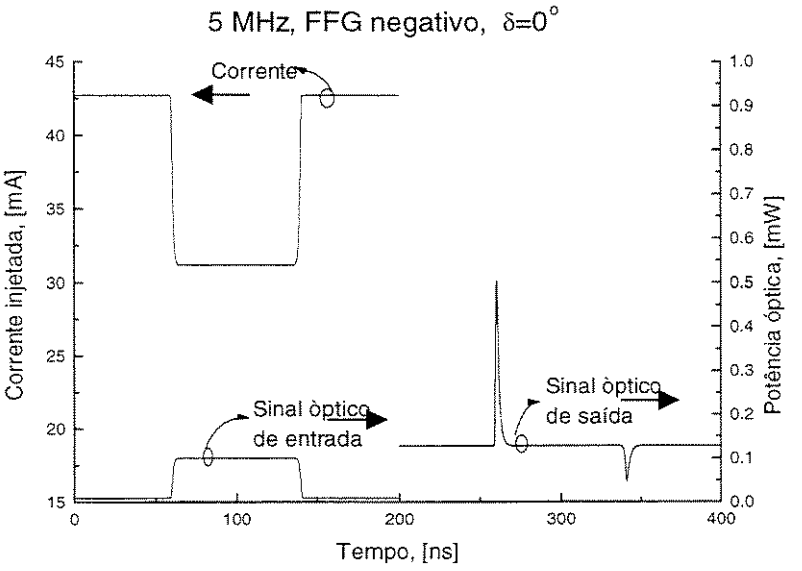

Figura 5.5.3: amplificação com FFG negativo, $\delta = 0°$

Nessa configuração, sem diferença de fase entre a componente modulada da corrente injetada e o sinal óptico de entrada, obteve-se um sinal de saída amplificado e sem informação. Neste caso, porém, existe um elevado *overshoot* (aproximadamente 400 μW) no local onde havia a transição entre o nível lógico "0" e o "1", e um *undershoot* onde havia a transição inversa. Esse maior ruído na antiga transição 0-1 óptica pode ser explicado pela dinâmica de variação do tempo de vida dos portadores elétricos, pois sua concentração é abruptamente diminuída pela modulação da corrente injetada, fazendo com que esse tempo de vida passe de um valor pequeno (alta densidade de portadores) para um maior (menor densidade de portadores) abruptamente **[10]**.

Na Figura 5.5.4, observa-se esse *overshoot* reduzido a menos da terça parte do seu valor original, devido a uma diferença de fase de 2,5 graus entre a modulação da corrente e o sinal óptico de entrada. Em compensação, essa diferença de fase faz com que um *overshoot* de mesma amplitude apareça na posição temporal onde havia a transição óptica 1-0, já que a corrente volta ao seu valor alto antes do nível óptico alto acabar.

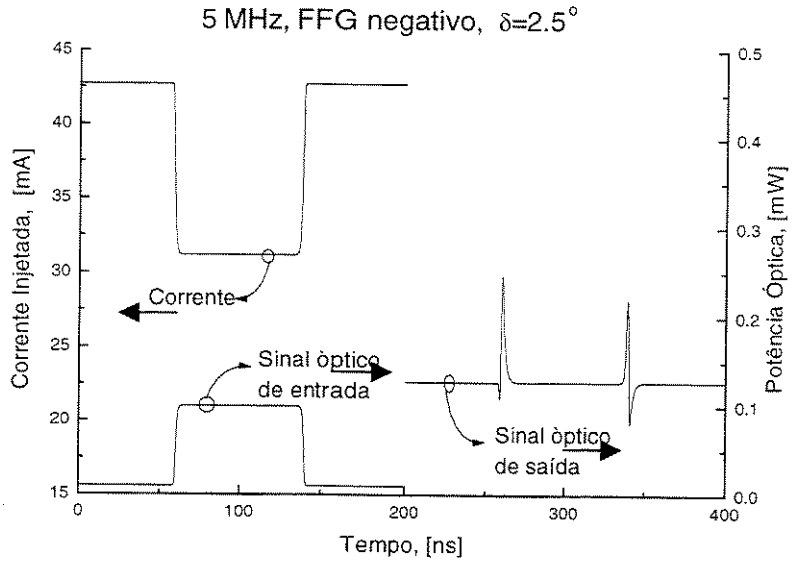

Figura 5.5.4: amplificação com FFG negativo, $\delta = 2.5°$

Tentando solucionar esse problema, modificou-se a largura do pulso da corrente de modulação. A simulação foi feita no dobro da taxa (10 MHz), com o período de nível baixo da corrente alargado temporalmente. O resultado obtido mostra redução significativa dos níveis de *overshoot* (40 μW), porém com *undershoots* ainda altos (até 80 μW). Esse resultado é apresentado na Figura 5.5.5.

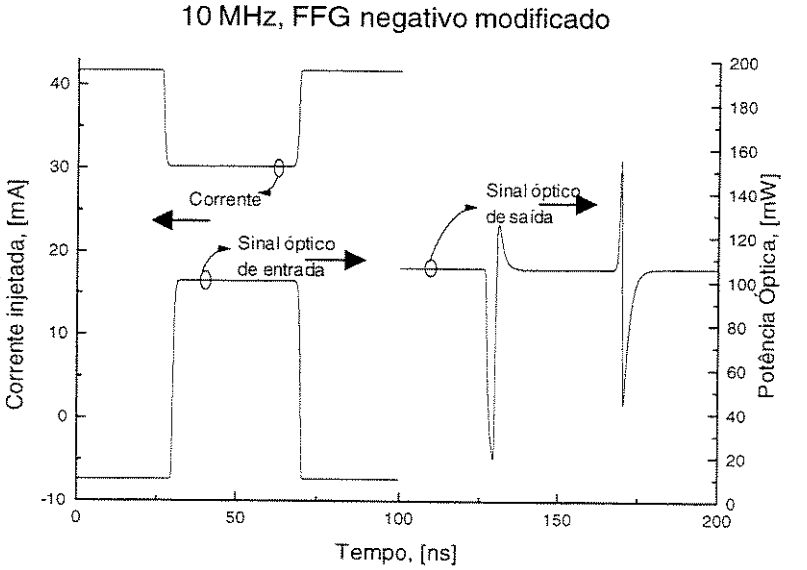

Figura 5.5.5: amplificação com FFG modificado, 10MHz

Pretende-se solucionar esse problema de altos transitórios presentes no sinal que deveria apresentar apagamento da informação no esquema FFG através da passagem em um ou mais SOAs com FFG. Outra forma seria incorporar na parte modulada da corrente de alimentação componentes proporcionais à derivadas temporais do sinal de entrada, de modo a compensar altas quedas na densidade de portadores elétricos energizados.

Voltando à taxa anterior (5MHz), utilizou-se uma diferença de fase acentuada de $\delta = 90°$, visando observar os efeitos sobre o pulso de entrada. Os resultados são apresentados na Figura 5.5.6. Neste caso, o sinal resultante da amplificação na saída do SOA passa a ter três níveis distintos de estabilização: um no antigo "0" óptico (em torno de 10 μW), outro um pouco acima do antigo "1" (130 μW) e outro bem acima (800 μW). Nesse tipo de configuração observa-se as capacidades do SOA com FFG de reformatar um sinal óptico através do controle dinâmico de corrente.

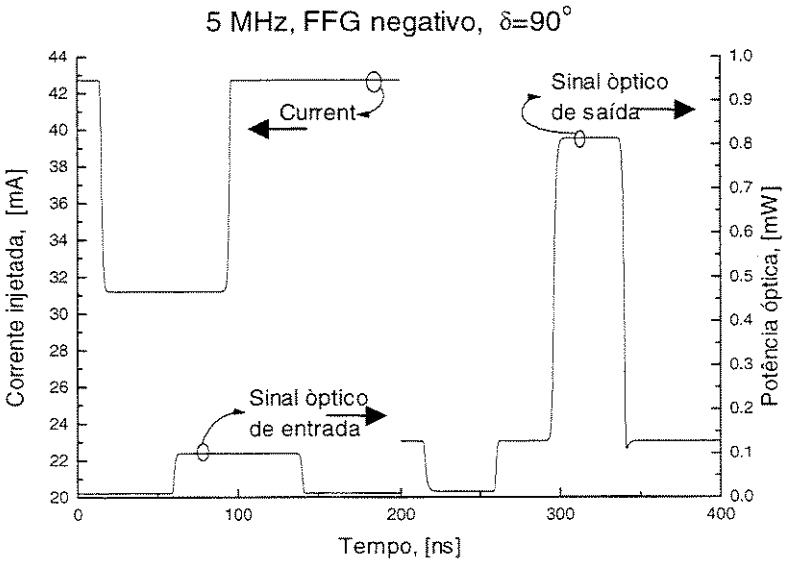

Figura 5.6.6: amplificação com FFG negativo, $\delta = 90°$

Passando para uma onda quadrada com taxa mais alta, de 1GHz de pulsos super-gaussianos (equivalente a 2Gb/s com bit de duração igual a um símbolo lógico), tem-se, na Figura 5.5.7, o resultado de amplificação sem FFG, para um SOA com as mesmas características do utilizado anteriormente. Observa-se claramente o tempo de resposta lento do dispositivo, que é alimentado com corrente a partir do instante t = 0, necessitando de 5 pulsos (5 ns) para estabilizar o ganho, fato já comentado na seção anterior.

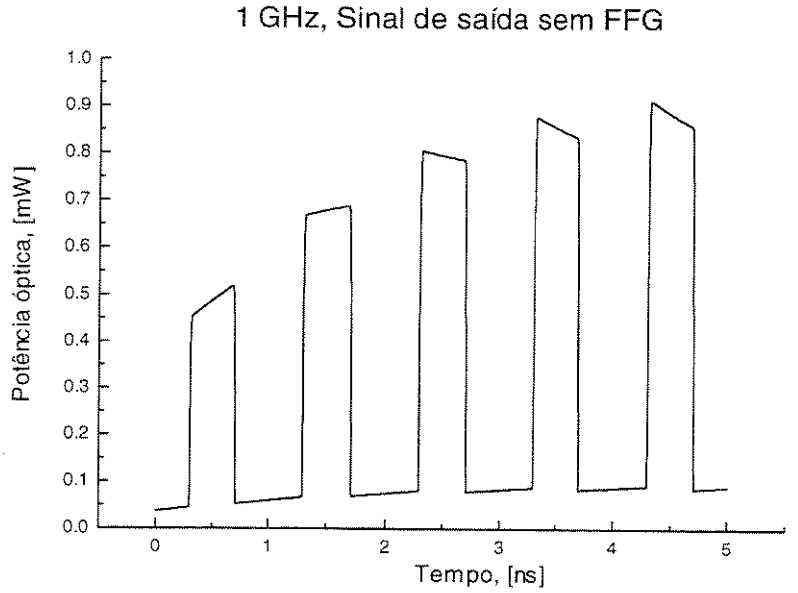

Figura 5.5.7: amplificação normal em 1GHz

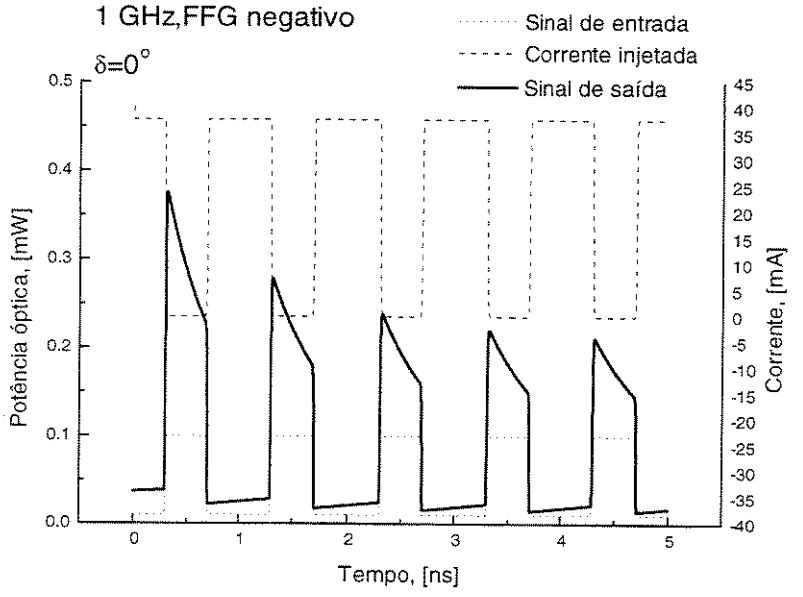

Figura 5.5.8: 1GHz, FFG negativo, $\delta = 0°$

Devido a essa dinâmica de ganho lenta, em altas taxas a reconstrução do pulso óptico fica comprometido, tanto para a regeneração como para o apagamento da informação presente na portadora óptica. Para exemplificar, apresentam-se os dois casos. Primeiro, realizou-se FFG negativo (Figura 5.5.8), observando-se que o sinal, apesar de decrescente no transitório, nunca tem seu nível reduzido significativamente, de modo que pudesse ser realizado um apagamento de informação da portadora óptica.

Em seguida, simulou-se FFG positivo, tentando melhorar a razão de extinção do sinal óptico (Figura 5.5.9), o que também não acontece em taxa dessa grandeza. Observa-se claramente a dinâmica lenta de ganho, com o ganho do SOA comportando-se aproximadamente como um integrador da corrente injetada, não conseguindo estabilizar o ganho dentro do relativamente curto período do sinal óptico.

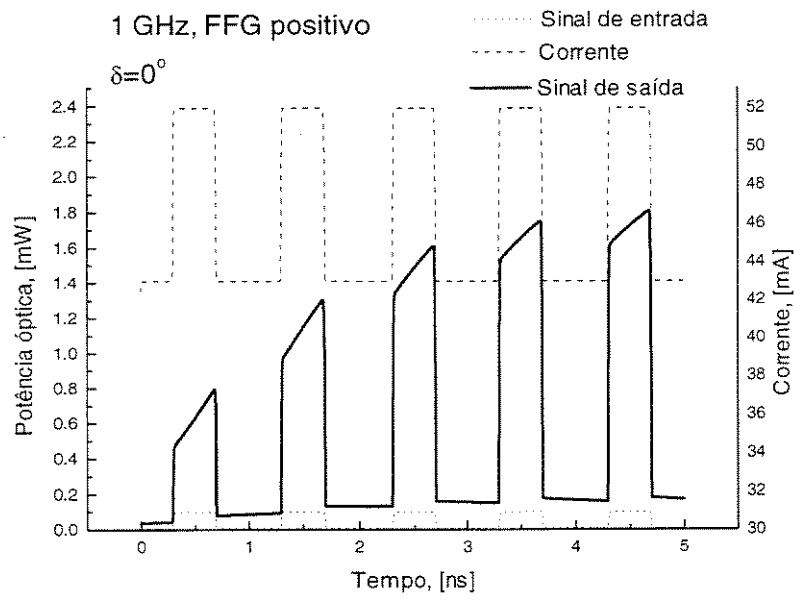

Figura 5.5.9: 1GHz, FFG positivo, $\delta = 0°$

Como pode-se observar nos exemplos acima, a resposta dinâmica do ganho do SOA com a modulação da corrente é lenta quando trabalha-se com taxas da ordem de Gigabits por segundo. Devido a este fato, as operações de processamento óptico em altas taxas utilizando SOA como elemento ativo utilizam o deslocamento de fase inserido no sinal que é amplificado devido ao índice de refração complexo da cavidade, modulado pela densidade de portadores elétricos [**1**, **2**, **8-9**, **10**, **12-15**, **21**]. Tal deslocamento de fase foi anteriormente comentado na Seção 5.3, com resultados teóricos.

Para ilustrar como a variação da resposta dinâmica do ganho do SOA varia com a taxa de transmissão, simulou-se a passagem de trem de pulsos super-gaussianos com nível baixo P_0 = 10 μW e nível alto P_1 = 100 μW (ER_{in} = -10 dB) com taxa variando de 5 MHz até 500 GHz (10 Mb/s a 1 Tb/s) por um SOA com parâmetros *default* (ver Seção 4.2) e alimentado por uma corrente composta por componente DC de 90 mA e componente modulada com ganho K = -76.7 mA, sem defasagem com o sinal de entrada (δ = 0°). Essa corrente foi ajustada de maneira a "apagar" o bit 1 do sinal em 5Mb/s (ER_{out} = 0 dB),

resultando em sinal CW com nível ligeiramente maior que o P_{bit1} anterior (141 μW) e altos *overshoots*, como exemplificado na Figura 5.5.3.

Para cada freqüência de transmissão foram coletados o valor do *overshoot* inicial e do nível de estabilização do sinal após o mesmo. Foi também calculada a razão de extinção de saída $ER_{saída}$, após a passagem do sinal pelo SOA. Com os dados de $ER_{saída}$ versus taxa de transmissão, foi construída a Figura 5.5.10, onde a condição $ER = 0$ define o apagamento. Observa-se, nessa figura, que apenas até taxas da ordem de 100 MHz o controle de ganho consegue fazer com que o nível alto do sinal (P_{bit1}) fique no mesmo patamar do nível baixo (P_{bit0}). A partir dessa taxa, a dinâmica de ganho torna-se lenta em relação ao período do bit, de forma que não se consegue mais fazer com que a razão de extinção atinja o valor ideal 0.

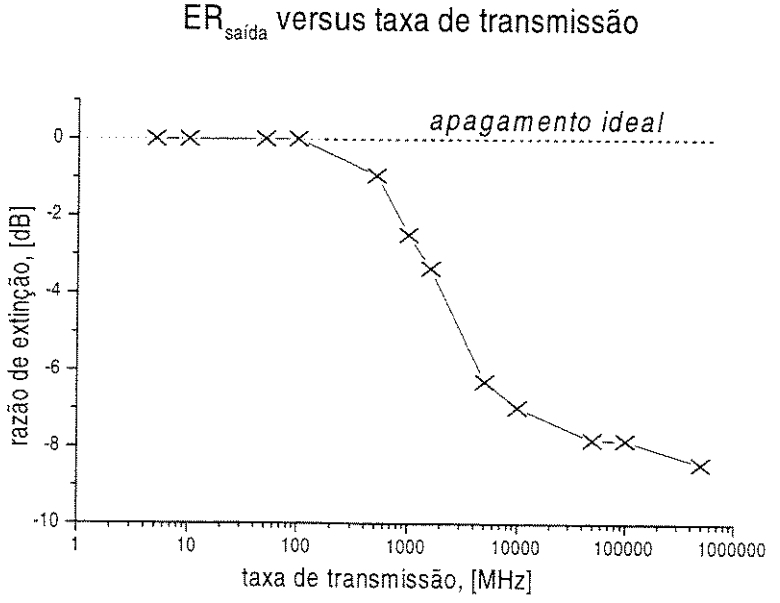

Figura 5.5.10: ER x taxa de transmissão para sinal óptico em forma de onda quadrada.

Com os dados de nível de *overshoot* versus taxa de transmissão, construiu-se a Figura 5.5.11, onde se observa a amplitude do *overshoot* crescendo até um máximo em 1600 MHz, posteriormente caindo e voltando a subir em 500 GHz, mostrando que em altas taxas a razão de extinção tende a voltar a seu valor inicial, antes da amplificação.

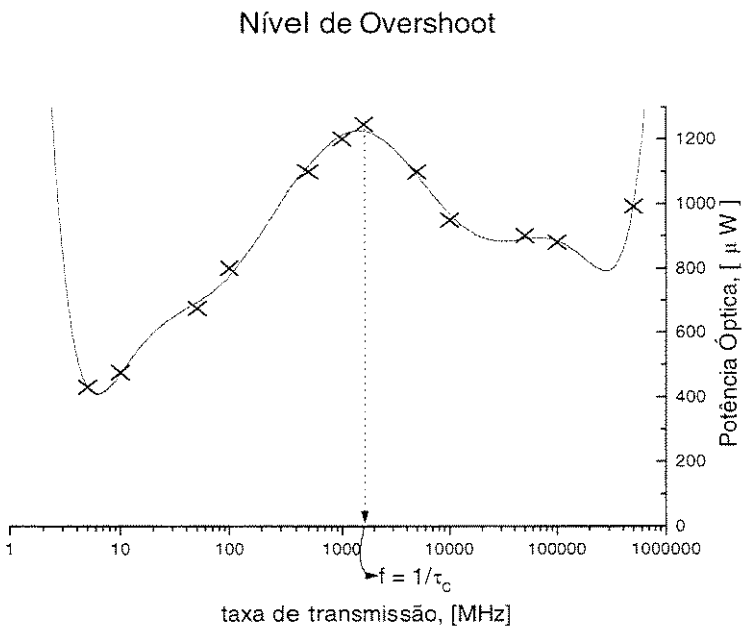

Figura 5.5.11: *overshoot* x taxa de transmissão, no apagamento para sinal de onda quadrada.

Através de aproximação numérica, ajustou-se um polinômio de grau 8 aos pontos teóricos da Figura 5.5.11 e obteve-se a seguinte equação para a relação Y (*overshoot*) versus X (taxa de transmissão):

$$Y = 7012.8764516 - 28199.0744252*X + 47921.368632*X^2 - 42451.9350209*X^3 + 21739.3105634*X^4 - 6621.6706085*X^5 + 1179.6944115*X^6 - 113.3152783*X^7 + 4.528659*X^8$$

O ponto de máximo de *overshoot* na Figura 5.5.11 foi obtido analisando-se os resultados de simulações em torno da freqüência igual ao inverso do tempo de vida estático dos portadores elétricos (1/770ps $\cong$ 1300 MHz, no caso de parâmetros default), pois já se pensava numa relação com o tempo de relaxação intrínsico da cavidade, como observado para o ruído de intensidade relativo (*RIN, relative intensity noise*) em lasers [**23**]. Pode-se notar nesta figura que o máximo ocorre em $f \cong 1/\tau_c$. Este valor de máximo ocorre em 1600 MHz, o que fornece um tempo de relaxação aproximado em 625 ps. Tal valor é próximo do valor mínimo alcançado pela variação dinâmica do tempo de vida dos portadores em amplificação nessa taxa de transmissão, como ilustrado na Figura 5.5.12.

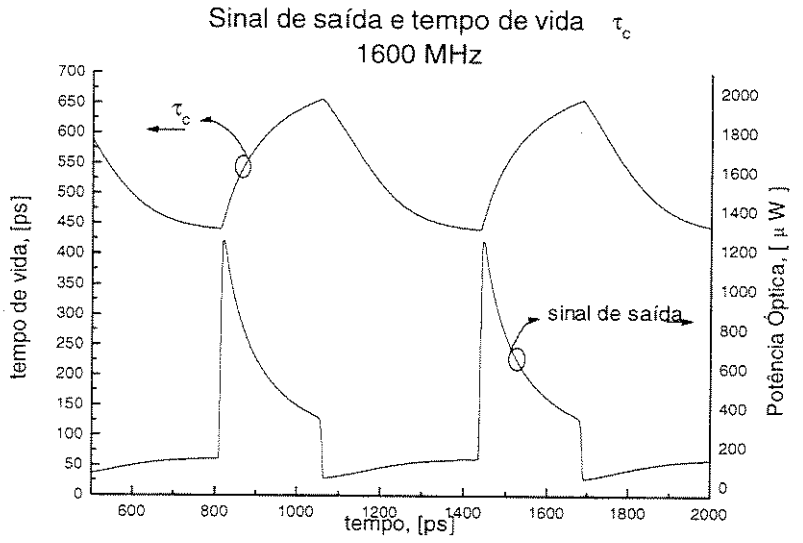

Figura 5.5.12: sinal de saída e tempo de vida dos portadores elétricos para taxa de 1600 MHz.

No processo de procura do máximo de *overshoot*, observou-se também que a amplitude da variação dinâmica do tempo de vida dos portadores aumenta com a diminuição da taxa de transmissão. Isto ocorre devido ao fato de que, em menores taxas, há tempo suficiente para um maior consumo de portadores durante a amplificação do sinal óptico. Isso indicaria uma relação dinâmica do tempo de relaxação na cavidade laser com a taxa de consumo de portadores.

Na Figura 5.5.13 são apresentados os dados referentes ao nível de estabilização do sinal após o *overshoot* inicial. Observa-se, como já discutido anteriormente, o apagamento

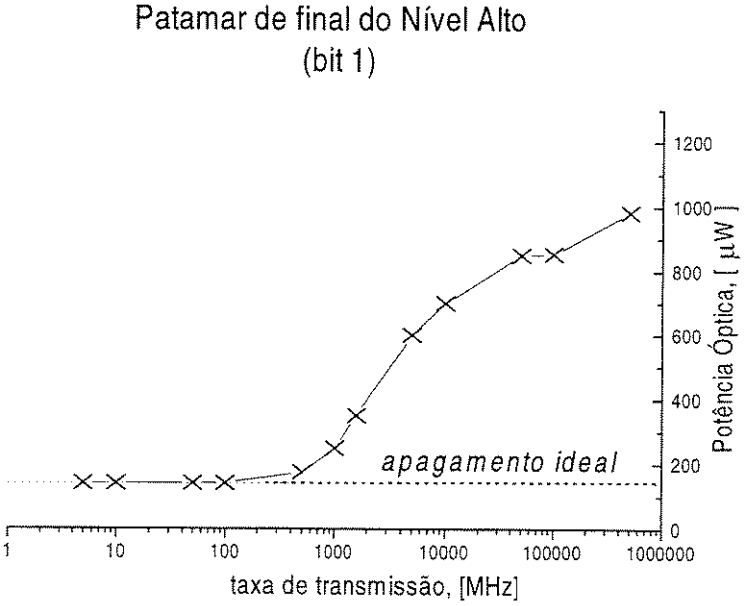

Figura 5.5.13: patamar de estabilização do bit 1, no apagamento para sinal de onda quadrada.

do sinal ($P_{bit1} = P_{bit0}$) para taxas de até 100 MHz. Com taxas mais altas, o SOA não consegue mais responder convenientemente à modulação da corrente injetada durante a descida do pico inicial, não sendo possível chegar a um valor estável antes do final do período do bit, como ilustrado na Figura 5.5.8. Tomou-se então o nível mínimo do pico (final do período do "bit 1") como valor para representar o patamar final de estabilização. A partir de algumas centenas de Mb/s o sinal já não consegue ter sua razão de extinção levada a zero e em altas taxas (centenas de Gb/s) praticamente o *overshoot* desaparece (analizar a Figura 5.5.11 em conjunto com 5.5.13), pois o nível máximo do sinal aproxima-se do patamar de mínimo do bit 1. Isto é ilustrado na Figura 5.5.14, onde apresenta-se o sinal de saída do SOA para taxa em 50 GHz, após a estabilização do ganho.

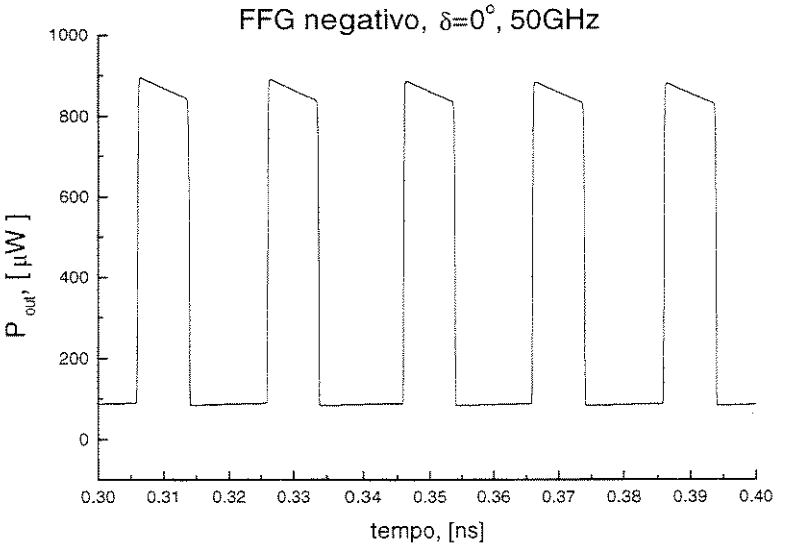

Figura 5.5.14: resposta em 50 GHz, no apagamento para sinal de onda quadrada.

5.6 - CONCLUSÃO

A partir dos resultados apresentados neste capítulo, pode-se concluir que o programa *SOASim* tem boa capacidade de prever os resultados experimentais, tanto para amplificação de sinal óptico em arranjo de cascata de SOAs operados na saturação, como para os casos em que a corrente injetada no dispositivo é modulada, a exemplo dos casos em FFG.

Mostram-se, também, as capacidades de utilização desse tipo de configuração para a regeneração para sinais ópticos de onda quadrada com taxas transmissão de até algumas centenas de MHz. Em taxas mais altas, a dinâmica da densidade de portadores impede a boa eficiência de um processamento de sinais ópticos para a configuração aqui utilizada.

CAPÍTULO 6 – CONCLUSÕES FINAIS

Foi desenvolvido um programa, baseado na solução das equações diferenciais de taxa, para simular amplificadores ópticos a semicondutor com controle dinâmico de ganho. A partir dos resultados apresentados no Cap. 5 pode-se concluir que este programa possui boa capacidade de previsão dos resultados experimentais. Entretanto, o programa ainda não contém simulações do ruído de emissão espontânea, parâmetro importante em SOAs. Exceto por este fato, os resultados teóricos apresentaram boa concordância com seu respectivos resultados experimentais. De forma geral, pôde-se mostrar claramente as possibilidades de utilização do SOA como chave óptica assim como na regeneração de sinais ópticos, com taxas de transmissão de até algumas centenas de MHz. Em taxas mais elevadas, a dinâmica da densidade de portadores do SOA, e consequentemente sua dinâmica de ganho, impossibilitam a regeneração do sinal óptico para as montagens aqui apresentadas.

Como sugestões para futuros trabalhos de aprimoramento desse programa, pode-se citar a adição de ruído ao modelamento utilizado, de forma a trazer maior confiabilidade para as simulações. Em adição, poder-se-iam utilizar modelos que possibilitassem o estudo de fenômenos de modulação cruzada, visando aplicações que envolvam a amplificação de mais de uma portadora de sinal óptico. Finalmente, quanto à interface gráfica, sugere-se o melhoramento das condições de simulação, através da criação de classes orientadas ao problema em questão, em consonância com os modernos programas em C^{++}.

APÊNDICES

APÊNDICE A - ROTINA C++

Nesse apêndice é apresentado na íntegra o arquivo ***Unidade1.cpp***, correspondente à interface e rotinas apresentadas na Seção 4.2, bem como comentários relevantes ao entendimento dos comandos e rotinas em grifo. Convém observar que, para facilitar a utilização de unidades comumente usadas na literatura, as unidades de comprimento das variáveis do programa são sempre em centímetros (cm).

Para uma descrição mais detalhada dessas rotinas, consultar o Cap. 3.

Unidade1.cpp

```
/*-------- Inclusão das bibliotecas necessárias ao funcionamento do programa----*/
#include <vcl.h>
#pragma hdrstop
#include <math.h>
#include <stdio.h>
#include <iostream.h>
#include <io.h>
#include <fstream.h>
#include <stdlib.h>
#include <complex.h>
#include <strstrea.h>
#include <string.h>
#include <conio.h>

                                Inclusão do arquivo .h correspondente.
#include "Unidade1.h"

#define n_space_points 1000     Definição de variáveis imutáveis: discretização espacial
#define n_time_points 10000     e temporal, velocidade da luz c, carga do elétron q, constante de
                                Plank h e o número e.
#define c 3*pow(10,10)
#define q 1.6*pow(10,-19)
#define h 6.63*pow(10,-34)

#define e 2.71828182845904
//---------------------------------------Inicializações do C++Builder3
#pragma package(smart_init)
#pragma resource "*.dfm"
TSimulform *Simulform;
//---------------------------------------------------------------------------

__fastcall TSimulform::TSimulform(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
/*----- Declaração das funções ----------*/

void data(void);
void create_pulse(void);
double pul_temp(double,double,double,double,int);
void current(void);
void g_0_z(void);
void calculo(void);
void integral(void);
void refeed(void);
```

```
void loadinputpulse(void);
void loadcurrent(void);
void savedata (void);
/*---------------------------------*/

/*---------Declaração das variáveis globais----*/

double bj_in[n_time_points];
double bj_out[n_time_points];
double hj[n_time_points];
double gj[n_time_points];
double dj[n_time_points];
double rj[n_time_points];
double tj[n_time_points];
double gain[n_time_points];
double Nj[n_time_points];
double Ij[n_time_points];
double norma_in[n_time_points];
double norma_out[n_time_points];
double g0j[n_time_points];
double talcj[n_time_points];
double phshift[n_time_points];
double zj[n_space_points];
double pot0[n_space_points];
double nsc,R1,R2,gama,talc,talp,i0,beta;
double a,Nt,N0,vg,alfa,alfacav,g0,G,es,Gn;//fator;
double maximo,ampli_per,maximo2;
double lamb,max1,min1,low,T,per,pin_f,pin_ini,rin;
double taxa,dt,foton,dz,ps,freq,delta_pot;
int num_amp,cont_amp=1,npul,i,j,form,count1,cont2,count3;
int cont_maximo,cont_onw,n_rin,n_pot,cont_pot,cont_rin;

bool error=false;

bool signalfase=true;
bool sign=false, talcxN=false;

double w,d,l,v;

double temponw1,difm,difmin,ampli_pat;
int ii,cont_m,cont_x,cont_der,tes;

double curr,K,fase,delta,sigma,loss,dbinsloss,insloss;

double B,C,Pdc;

/*-------------------------------------*/
/*------ Função Data ----------*/

void data(void)
{                       // calcula alguns valores importantes como
v=w*d*l;                // volume ativo, ganho de pequenos sinais estático, tempo de
N0=Nt*v;                // vida dos fótons, perda da cavidade, etc
vg=c/nsc;
i0=q*N0/talc;
g0=(gama*a*Nt)*(curr/i0-1);
G=10*log10(exp(g0*l));

alfacav=alfa+1/(2*l)*log(1/(R1*R2));
talp=1/(vg*alfacav);
```

```
loss=10*log10(exp(alfa*l));

freq=c/lamb;
foton=h*freq;
es=foton*w*d/(gama*a);

ps=es/talc;
}
/*-------------------------*/
```
/*------rotina de arquivamento dos parâmetros do sinal de entrada, corrente e do SOA em arquivo .txt no diretório especificado -*/

```
void savedata(void)
{
  strstream path_data;
  ofstream data_out;

 dataout:
   path_data<<"c:\\SOAsimulator\\data"<<cont_amp<<".txt"<<ends;
          data_out.open(path_data.str(),ios::out);

          if(data_out.fail())
          {     goto dataout;    };

   data_out<<"Input Signal Parameters:"<<endl<<endl;
   data_out<<"Wavelength = "<<lamb*pow(10,4)<<" um"<<endl;
   data_out<<"Maximum Power = "<<max1*pow(10,6)<<" uW"<<endl;
   data_out<<"Minimum Power = "<<low*pow(10,6)<<" uW"<<endl;
   data_out<<"Transmission Rate = "<<taxa*pow(10,-6)<<" Mb/s"<<endl;
   data_out<<"Number of Pulses = "<<npul<<endl;
   data_out<<"Supergaussian Form = "<<form<<endl<<endl;

   data_out<<"SOA cascade number "<<cont_amp<<endl<<endl;

   data_out<<"Input Current Parameters:"<<endl<<endl;
   data_out<<"DC current = "<<curr*pow(10,3)<<" mA"<<endl;
   data_out<<"Modulated current = "<<K*pow(10,3)<<" mA"<<endl;
 if (signalfase)
   data_out<<"Phase shift = +"<<fase<<" degrees"<<endl<<endl;
else data_out<<"Phase shift = -"<<fase<<" degrees"<<endl<<endl;

   data_out<<"SOA parameters:"<<endl<<endl;

if (talcxN){
   data_out<<"Carrier lifetime as N function"<<endl;
   data_out<<"Recombination Factors:"<<endl;
   data_out<<"B = "<<B*pow(10,10)<<" .10^-10.cm^3/s"<<endl;
   data_out<<"C = "<<C*pow(10,29)<<" .10^-29.cm^6/s"<<endl<<endl;
};

   data_out<<"Cavity Dimension:"<<endl;
   data_out<<"width = "<<w*pow(10,4)<<" um"<<endl;
   data_out<<"height = "<<d*pow(10,4)<<" um"<<endl;
   data_out<<"length = "<<l*pow(10,4)<<" um"<<endl;
   data_out<<"volume = "<<v*pow(10,12)<<" um^3"<<endl<<endl;

   data_out<<"Carrier Density = "<<Nt*pow(10,-18)<<" 10^18/cm^3"<<endl;
   data_out<<"Transversal Section Gain = "<<a*pow(10,16)<<" 10^-16.cm^2"<<endl;
   data_out<<"Atenuation Coeficient = "<<alfa<<" /cm"<<endl;
```

```
   data_out<<"Facet Refletion = "<<R1<<endl;
   data_out<<"Line-width Enhancement Factor = "<<beta<<endl;
   data_out<<"Efective Refrative Index = "<<nsc<<endl;
   data_out<<"Confinement Factor = "<<gama<<endl;
   data_out<<"Insertion Loss = "<<dbinsloss<<" dB"<<endl;
   data_out<<"Carrier Lifetime = "<<talc*pow(10,12)<<" ps"<<endl<<endl;
   data_out<<"Transparency Current = "<<i0*pow(10,3)<<" mA"<<endl;
   data_out<<"Short-signal Gain = "<<G<<" dB"<<endl;
   data_out<<"Cavity Loss = "<<loss<<" dB"<<endl;
   data_out<<"Saturation Energy = "<<es*pow(10,12)<<" pJ"<<endl;
   data_out<<"Photon Lifetime = "<<talp*pow(10,12)<<" ps"<<endl;

  data_out.close();
}
/*-------------------------*/
```

/*--rotina de carregar sinal de entrada de um arquivo-------*/

```
void loadinputpulse(void)
{
   ifstream arquivo_pin;
   ofstream arquivo_in,normal_out;
         int i;

 //loadin:
         arquivo_pin.open("c:\\SOAsimulator\\temp\\inputL.dat",ios::in);
         if(arquivo_pin.fail()) error=true;  //goto loadin;

    i=1;
                  do
                  { arquivo_pin>>tj[i]>>bj_in[i];
                    i++;
                  }while(!arquivo_pin.eof()); //}while(i<n_time_points-1);
                  arquivo_pin.close();    // error=true;

    dt=tj[1];

   storein:
         //error=false;
   arquivo_in.open("c:\\SOAsimulator\\inputLd.dat",ios::out);

         if(arquivo_in.fail())
         {  //error=true;
     goto storein;
   };

         i=1;
         do
         {       arquivo_in<<tj[i]<<" "<<bj_in[i]<<endl;
                  i++;
         }while(i<n_time_points);
   arquivo_in.close();


         i=1;                        // achando o máximo do sinal de entrada para cálculo da norma
         maximo2=0;
         do
         {        if(fabs(bj_in[i])>fabs(maximo2))
                  {        maximo2=fabs(bj_in[i]);       }
                  i=i+1;
```

```
        }while(i<n_time_points-1);

 pouta2:
  norma1_out.open("c:\\SOAsimulator\\normainLd.dat",ios::out);
  if(norma1_out.fail())
        { //error=true;
   goto pouta2;
  };
        i=1;
        do
        {   norma_in[i]= bj_in[i]/maximo2;
    norma1_out<<tj[i]<<" "<<norma_in[i]<<endl;
                i++;
        }while(i<n_time_points);
  norma1_out.close();
}
/*-----------------------------------------------*/

/*--rotina de carregar sinal de corrente de um arquivo-------*/

void loadcurrent(void)
{
  ifstream arqcurr_pin;
  ofstream arqcurr_in;
  double temp;
        int i;

        arqcurr_pin.open("c:\\SOAsimulator\\temp\\IinL.dat",ios::in);

   i=1;
                do
                { arqcurr_pin>>temp>>Ij[i];
                 i++;
                }while(!arqcurr_pin.eof()); //}while(i<n_time_points-1);
                arqcurr_pin.close();

  arqcurr_in.open("c:\\SOAsimulator\\IinLd.dat",ios::out);

        i=1;
        do
        {       arqcurr_in<<tj[i]<<" "<<Ij[i]<<endl;
                i++;
        }while(i<n_time_points);
  arqcurr_in.close();

}
/*-----------------------------------------------*/

/*-----gerador do pulso de entrada----------------*/

void create_pulse(void)
{                               //cria e salva o sinal de entrada, juntamente com sua norma
  double tempp;
  ofstream arquivo_pout,norma1_out;

        min1=low/max1;
        T=1/taxa;
        per=npul*T;
        dt=per/(n_time_points-1);
```

```
        i=1;
  do    {       tj[i]=i*dt;
                i++;
        }while(i<n_time_points);

  i=1;
        do  {   tempp=fmod(tj[i],T);
                bj_in[i]=pul_temp(min1,max1,tempp,T,form);
                norma_in[i]= bj_in[i]/max1;
                i++;
        }while(i<n_time_points);

        pouta:
        arquivo_pout.open("c:\\SOAsimulator\\input1.dat",ios::out);

        if(arquivo_pout.fail())
        { //error=true;
    goto pouta;
  };
        i=1;
        do
        {       arquivo_pout<<tj[i]<<" "<</*fabs*/(bj_in[i])<<endl;
                i++;
        }while(i<n_time_points);
  arquivo_pout.close();

        normal_out.open("c:\\SOAsimulator\\normain1.dat",ios::out);
  if(normal_out.fail())
        { //error=true;
    goto pouta;
  };
        i=1;
        do
        {       normal_out<<tj[i]<<" "<<norma_in[i]<<endl;
                i++;
        }while(i<n_time_points);
  normal_out.close();

}
/*-------------------------*/

/*------- gerador de pulso super-gaussiano--------*/

double pul_temp(double minimo,double maxi,double tempo,double periodo,int form)
{
        return(minimo*maxi+(1-minimo)*maxi*exp(-pow(((tempo-periodo*.5)/(.2*periodo)),2*form)));
}

/*-------------------------------------*/

/*-------arquivando o sinal de corrente ----------------------------*/

void current(void)
{                       // cria e grava o sinal de corrente
  double tempdc;
  strstream path_curr;
  ofstream curr_in;

  delta=floor(fase*n_time_points/(360*npul));
  sigma=n_time_points-delta;
```

```
 i=1;
 Pdc=0; tempdc=0;
do                                  // calcula o nível rms do sinal normalizado de entrada
{   tempdc=dt*norma_in[i]*norma_in[i];
   Pdc=Pdc+tempdc;
   i++;
}while(i<n_time_points);
   Pdc=Pdc/per;
   Pdc=pow(Pdc,0.5);
                              //faz o deslocamento de fase escolhido
if (signalfase){                    //se positivo...
   i=1;
   j=1;
          do    {   j=i+delta;
                Ij[i]= curr+K*(norma_in[j]-Pdc);
                    i++;
          }while(i<sigma);

   i=1;
   j=1;
          do    {   j=i+sigma;
                Ij[j]= curr+K*(norma_in[i]-Pdc);
                    i++;
          }while(i<delta);

}
else{     i=1;  j=1;                //se negativo...
          do    {   j=i+sigma;
                Ij[i]= curr+K*(norma_in[j]-Pdc);
                    i++;
          }while(i<delta);

   i=1;
   j=1;
          do    {   j=i+delta;
                Ij[j]= curr+K*(norma_in[i]-Pdc);
                    i++;
          }while(i<sigma);
}

  pouta:
   path_curr<<"c:\\SOAsimulator\\Iin"<<cont_amp<<".dat"<<ends;
          curr_in.open(path_curr.str(),ios::out);

          if(curr_in.fail())
          { //error=true;
     goto pouta;
    };


          i=1;
          do
          {         curr_in<<tj[i]<<" "<<Ij[i]<<endl;
                    i++;
          }while(i<n_time_points);

  curr_in.close();
}
```

```
/-------------------------/

/*--------- função de cálculo do ganho inicial ----------------*/

void g_0_z(void)
{                               // calcula o ganho inicial  gj[j] a partir de g0 e bj_in[1]
        int j=1;
        double k1f,k2f,k3f,k4f;

        dz=l/(n_space_points-1);

        do
        { zj[j]=(j-1)*dz;                       // discretização espacial zj[j]
                j++;
        }while(j<n_space_points);

        j=1;
        pot0[1]=bj_in[1];

  do                                                    // Runge-Kutta para pot0[j] ...
        {       k1f=dz*g0*pot0[j]/(1+pot0[j]/ps);
                k2f=dz*((pot0[j]+k1f/2)/(1+(pot0[j]+k1f/2)/ps));
                k3f=dz*((pot0[j]+k2f/2)/(1+(pot0[j]+k2f/2)/ps));
                k4f=dz*((pot0[j]+k3f)/(1+(pot0[j]+k3f)/ps));
                pot0[j+1]=pot0[j]+(k1f+2*k2f+2*k3f+k4f)/6;
                j++;
        }while(j<n_space_points);

        j=1;
        do
        {       gj[j]=g0/(1+pot0[j]/ps);
                j++;
        }while(j<n_space_points);
}
/*---------------------------------*/

/*----integração de  g(0,z) sobre z=[0,l]  ---*/

void integral(void)
{                               // integra gj para todo o dispositivo usando a regra de Simpson 3/8
          int i;                //encontrando hj[1] para a função calcul
        double tmp;
        i=1;
        hj[1]=0;
        do
        {       tmp=dz*(3/8*gj[i]+9/8*gj[i+1]+9/8*gj[i+2]+3/8*gj[i+3]);
                hj[1]=hj[1]+tmp;
                i=i+3;
         }while(i<=n_space_points-4);
}
/*---------------------------------*/

/*----- cálculo do pulso de saída ---*/

void calculo(void)
{                               // RK4 para achar hj[i];  gravação em arquivo
        int i;
        double k1,k2,k3,k4;
        char erroca[8];
        strstream path_x,path_norma,path_g,path_tal,path_phase;
```

```
        ofstream arquivo_out,n_out,ganho_out,tal_out,phase_out;

if (sign){                          //---- calculando com talcj[i]...

   Nj[1]= exp(hj[1])/(gama*a)+Nt;
   talcj[1]=1/(B*Nj[1]+C*Nj[1]*Nj[1]);
   g0j[1]=(gama*a*Nt)*(Ij[1]/(q*N0/talcj[1])-1);

   i=1;
        do                  // encontrando hj[i]...
        {       k1=dt*((g0j[i]*l-hj[i])/talcj[i]-(bj_in[i]/es*(exp(hj[i])-1)));
                k2=dt*((g0j[i]*l-(hj[i]+k1/2))/talcj[i]-(bj_in[i]/es*(exp(hj[i]+k1/2)-1)));
                k3=dt*((g0j[i]*l-(hj[i]+k2/2))/talcj[i]-(bj_in[i]/es*(exp(hj[i]+k2/2)-1)));
                k4=dt*((g0j[i]*l-(hj[i]+k3))/talcj[i]-(bj_in[i]/es*(exp(hj[i]+k3)-1)));

                hj[i+1]=hj[i]+(k1+2*k2+2*k3+k4)/6;
            i++;

     Nj[i]= exp(hj[i])/*(fabs(hj[i]))*//(gama*a)+Nt;
     talcj[i]=1/(B*Nj[i]+C*Nj[i]*Nj[i]);
     g0j[i]=(gama*a*Nt)*(Ij[i]/(q*N0/talcj[i])-1);

        }while(i<n_time_points-1);

 path_tal<<"c:\\SOAsimulator\\tal"<<cont_amp<<".dat"<<ends;
        tal_out.open(path_tal.str(),ios::out);
        if(tal_out.fail())
        {
    //error=true;
          exit(0);
        }
        i=1;
        do                                  // gravando talcj[i]...
        {       tal_out<<tj[i]<<" "<<talcj[i]<<endl;
                i++;
        }while(i<n_time_points-1);
  tal_out.close();

}
else {                          //---- calculando com talc constante...
        i=1;
        do
        {       g0j[i]=(gama*a*Nt)*(Ij[i]/i0-1);
                i++;
        }while(i<n_time_points);

        i=1;
        do                          // encontrando hj[i]...
        {
                k1=dt*((g0j[i]*l-hj[i])/talc-(bj_in[i]/es*(exp(hj[i])-1)));
                k2=dt*((g0j[i]*l-(hj[i]+k1/2))/talc-(bj_in[i]/es*(exp(hj[i]+k1/2)-1)));
                k3=dt*((g0j[i]*l-(hj[i]+k2/2))/talc-(bj_in[i]/es*(exp(hj[i]+k2/2)-1)));
                k4=dt*((g0j[i]*l-(hj[i]+k3))/talc-(bj_in[i]/es*(exp(hj[i]+k3)-1)));

                hj[i+1]=hj[i]+(k1+2*k2+2*k3+k4)/6;
                i++;
        }while(i<n_time_points-1);

}
```

```
        i=1;
        do                                          // encontrando bj_out[i]...
        {
                bj_out[i]=insloss*bj_in[i]*exp((hj[i])-alfa*l);

                sprintf(erroca,"%lf",fabs(bj_out[i]));
                if(!strcmp(erroca,"+NaN")||!strcmp(erroca,"-NaN"))
                {
      //error2=true;
                        exit(0);
                }
                                                    // calculando gain[i] ...
gain[i]=10*log10(insloss*exp((hj[i])-alfa*l);
    phshift[i]=(-0.5)*beta*hj[i]/3.141592;
    i++;
        }while(i<n_time_points);

      /*---------------------*/

        i=1;                            // encontrando o máximo de bj_out ...
        maximo=0;
        //minimo=1E7;
        do
        { if(fabs(bj_out[i])>fabs(maximo))
                {       maximo=fabs(bj_out[i]);
                        cont_maximo=i;
                }
                i=i+1;
        }while(i<n_time_points-1);

  i=1;                                  // calculando norma_out[i]...
  do
  { norma_out[i]=(bj_out[i]/maximo);
    i++;
   }while(i<n_time_points);

                                        //---arquivando os arquivos...

        path_x<<"c:\\SOAsimulator\\output"<<cont_amp<<".dat"<<ends;
        arquivo_out.open(path_x.str(),ios::out);
        if(arquivo_out.fail())
        {
    //error=true;
                exit(0);
        }

        i=1;                                    // arquivando bj_out[i]...
        do
        {   arquivo_out<<tj[i]<<" "<<(bj_out[i])<<endl;
                i++;
        }while(i<n_time_points-1);
  arquivo_out.close();

        path_norma<<"c:\\SOAsimulator\\normaout"<<cont_amp<<".dat"<<ends;
        n_out.open(path_norma.str(),ios::out);
        if(n_out.fail())
        {
    //error=true;
                exit(0);
        }
```

```
  i=1;
        do                                  // arquivando norma_out[i]...
        {
                n_out<<tj[i]<<" "<<norma_out[i]<<endl;
                i++;
        }while(i<n_time_points-1);
  n_out.close();

        path_g<<"c:\\SOAsimulator\\gain"<<cont_amp<<".dat"<<ends;
        ganho_out.open(path_g.str(),ios::out);
        if(ganho_out.fail())
        {
    //error=true;
          exit(0);
        }
        i=1;
        do                                  // arquivando  gain[i]...
        {
                ganho_out<<tj[i]<<" "<<gain[i]<<endl;
                i++;
        }while(i<n_time_points-1);
  ganho_out.close();

  path_phase<<"c:\\SOAsimulator\\phaseshift"<<cont_amp<<".dat"<<ends;
        phase_out.open(path_phase.str(),ios::out);
        if(phase_out.fail())
        {
    //error=true;
          exit(0);
        }
        i=1;
        do                          // arquivando phase shift[i]...
        {
                phase_out<<tj[i]<<" "<<phshift[i]<<endl;
                i++;
        }while(i<n_time_points-1);
  phase_out.close();

}
/*-------------------------*/

/*--- função Refeed ---*/

void refeed(void)                           // alimenta input com último output
{
        cont_amp=cont_amp+1;        //incrementa contador da cascata
        int count=1;
        strstream path_x,path_norma;
        ofstream arquivo_out,n_out;

        do
        {       bj_in[count]=bj_out[count];
                norma_in[count]=norma_out[count];
                count++;
        }while(count<n_time_points-1);

  maximo2=maximo;
        path_x<<"c:\\SOAsimulator\\input"<<cont_amp<<".dat"<<ends;
        arquivo_out.open(path_x.str(),ios::out);
```

```
        if(arquivo_out.fail())
        {
    //error=true;
                exit(0);
        }

        i=1;
        do
        {       arquivo_out<<tj[i]<<" "<<(bj_in[i])<<endl;
                i++;
        }while(i<n_time_points-1);
  arquivo_out.close();

  path_norma<<"c:\\SOAsimulator\\normain"<<cont_amp<<".dat"<<ends;
        n_out.open(path_norma.str(),ios::out);
        if(n_out.fail())
        {
    //error=true;
                exit(0);
        }

        i=1;
        do
        {       n_out<<tj[i]<<" "<<norma_in[i]<<endl;
                i++;
        }while(i<n_time_points-1);
  n_out.close();
}
/*-------------------------------------*/

void __fastcall TSimulform::FormCreate(TObject *Sender)
{
  IinPanel->Top=20;             //função chamada ao se criar o form (tela inicial)
  IinPanel->Left=140;           //inicialização do posicionamento das janelas dentro da principal

  simulPanel->Top=120;
  simulPanel->Left=160;

  InputPanel->Top=20;
  InputPanel->Left=23;

  SOAPanel->Top=30;
  SOAPanel->Left=60;

  OutputPanel->Top=80;
  OutputPanel->Left=280;

  Simulform->Top=30;
  Simulform->Left=60;

  SOAset->Height=115;
  SOAset->Width=110;

/*-------- inicialização das janelas de texto----------*/

tw->Text=140;
td->Text="020";
tl->Text=350;

tNt->Text=20;
```

```
ta->Text=20;
talfa->Text=20;
tR->Text="0001";
tbeta->Text=5;
tnsc->Text=34;
tgama->Text=40;
tinsloss->Text=50;
ttalc->Text=750;

tlamb->Text=1.55;
tmax->Text=1000;
tmin->Text=100;
trate->Text=1000;
tnpul->Text=5;
tform->Text=30;

tK->Text="0000";
tcurr->Text=10000;
tfase->Text="0000";

/*-----------------------------*/

/*------ inicialização das variáveis a partir das janelas de texto----------*/

w=tw->Text*pow(10,-6);
d=td->Text*pow(10,-6);
l=tl->Text*pow(10,-4);


Nt=tNt->Text*pow(10,17);
a=ta->Text*pow(10,-17);
alfa=talfa->Text*pow(10,0);
R1=tR->Text*pow(10,-4);
R2=R1;
beta=tbeta->Text*pow(10,0);
nsc=tnsc->Text*pow(10,-1);
gama=tgama->Text*pow(10,-2);
dbinsloss=tinsloss->Text*pow(10,-1);
insloss=pow(10,-dbinsloss/10);
talc=ttalc->Text*pow(10,-12);

lamb=tlamb->Text*pow(10,-4);
max1=tmax->Text*pow(10,-7);
low=tmin->Text*pow(10,-7);
taxa=trate->Text*pow(10,5);
npul=tnpul->Text*pow(10,0);
form=tform->Text*pow(10,0);

B=5*pow(10,-10);
C=7.5*pow(10,-29);

 tB->Text=B*pow(10,11);
 tC->Text=C*pow(10,30);

K = 0;
curr=tcurr->Text*pow(10,-5);
fase=0;

data();                          //chama função data, para calcular dados
```

```
tIo->Text=i0/pow(10,-3);          //apresentação dos dados calculados
tg0->Text=G*1;
tloss->Text=loss;
tes->Text=es*pow(10,12);
tv->Text=v*pow(10,12);
ttalp->Text=talp*pow(10,12);
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Panel2DblClick(TObject *Sender)
{
  OutputPanel->Show();  // mostra painel de saída quando duplamente clicada a caixa output
        }
//---------------------------------------------------------------------------

void __fastcall TSimulform::SOAsetDblClick(TObject *Sender)
{
  SOAPanel->Show(); //mostra o painel com parâmetros do SOA quando seu ícone é clicado
}
//---------------------------------------------------------------------------
 void __fastcall TSimulform::SOAparameters1Click(TObject *Sender)
{
  SOAPanel->Show();    //mesmo que acima, quando selecionado do menu principal
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Panel1DblClick(TObject *Sender)
{
  InputPanel->Show();    //mostra o painel do sinal de entrada quando clicado o quadrado input...
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Inputsignal1Click(TObject *Sender)
{
  InputPanel->Show();    //mesmo que acima, quando selecionada do menu inicial
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Button4Click(TObject *Sender)
{
   InputPanel->Hide();    //oculta o painel do sinal de entrada quando clicado CLOSE...
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Exit1Click(TObject *Sender)
{
   Application->Terminate();        //finaliza o aplicativo, quando selecionado do menu principal
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::New1Click(TObject *Sender)
{
  cont_amp=1;              // inicia o programa, habilitando paineis e mostrando o painel do sinal
                           // de entrada, quando selecionado do menu principal
  Current1->Enabled=true;
  Inputsignal1->Enabled=true;
  SOAparameters1->Enabled=true;
  Panel4->Enabled=true;
  Panel1->Enabled=true;
  Panel2->Enabled=true;
```

```
   SOAset->Enabled=true;
   SOAImage->Enabled=true;

   InputPanel->Show();
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Button5Click(TObject *Sender)
{
   SOAPanel->Hide();              //oculta o painel dos parâmetros do SOA, quando clicado OK
 }
//---------------------------------------------------------------------------

void __fastcall TSimulform::Button6Click(TObject *Sender)
{
    OutputPanel->Hide();          //oculta o painel de saída, quando clicado OK
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::createinButtonClick(TObject *Sender)
{
max1=tmax->Text*pow(10,-7);    // atualização das varáveis do sinal de entrada e sua criação...
low=tmin->Text*pow(10,-7);
taxa=trate->Text*pow(10,5);
npul=tnpul->Text*pow(10,0);
form=tform->Text*pow(10,0);

   cont_amp=1;

create_pulse();
InputPanel->Hide();
Button2->Enabled=true;
loadinput->Enabled=true;
IinPanel->Show();

}
//---------------------------------------------------------------------------

void __fastcall TSimulform::updatesoabuttonClick(TObject *Sender)
{
w=tw->Text*pow(10,-6);         // atualização dos parâmetros do SOA...
d=td->Text*pow(10,-6);
l=tl->Text*pow(10,-4);

Nt=tNt->Text*pow(10,17);
a=ta->Text*pow(10,-17);
alfa=talfa->Text*pow(10,0);
R1=tR->Text*pow(10,-4);
R2=R1;
beta=tbeta->Text*pow(10,0);
nsc=tnsc->Text*pow(10,-1);
gama=tgama->Text*pow(10,-2);
dbinsloss=tinsloss->Text*pow(10,-1);
insloss=pow(10,-dbinsloss/10);

B=tB->Text*pow(10,-11);
C=tC->Text*pow(10,-30);

if (talBox->Checked) talc=1/(Nt*(B+C*Nt));
else talc=ttalc->Text*pow(10,-12);
```

```
data();

tIo->Text=i0/pow(10,-3);
tg0->Text=G*1;
tloss->Text=loss;
tes->Text=es*pow(10,12);
 tv->Text=v*pow(10,12);
ttalp->Text=talp*pow(10,12);
ttalc->Text=talc*pow(10,12);

}
//---------------------------------------------------------------------------

void __fastcall TSimulform::simulatebuttonClick(TObject *Sender)
{
  if (talBox->Checked)sign=true;          // chamada das funções necessárias à simulação
  else sign=false;

   savedata();
   g_0_z();                               // cálculo e gravação do resultado da simulação
   integral();
   calculo();

   simulPanel->Show();                    //mostra painel declarando fim da simulação
   refeedButton->Enabled=true;

}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Button1Click(TObject *Sender)
{
  simulPanel->Hide();                     //oculta painel declarando fim da simulação
 }
//---------------------------------------------------------------------------

void __fastcall TSimulform::refeedButtonClick(TObject *Sender)
{
   refeed();                    //chamada para função de realimentação

   InputPanel->Hide();
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::viewiin(TObject *Sender)
{
   IinPanel->Show();                      //mostra painel da corrente
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Button2Click(TObject *Sender)
{
   K=tK->Text*pow(10,-5);                 // atualização das variáveis do sinal da corrente
   curr=tcurr->Text*pow(10,-5);
   fase=tfase->Text*pow(10,-2);

   if (minus->Checked)K=K*(-1);

  if (faseplus->Checked)signalfase=true;
  else signalfase=false;
```

```
  current();                    // chama função de criação do sinal de corrente

  data();
     tIo->Text=i0/pow(10,-3);
     tg0->Text=G*1;
     tloss->Text=loss;
     tes->Text=es*pow(10,12);
     tv->Text=v*pow(10,12);
     ttalp->Text=talp*pow(10,-9);

  IinPanel->Hide();
  SOAPanel->Show();
  simulatebutton->Enabled=true;
  loadcur->Enabled=true;
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::Button3Click(TObject *Sender)
{
  IinPanel->Hide();       // oculta painel da corrente, quando clicado Close
}
//---------------------------------------------------------------------------
void __fastcall TSimulform::loadinputClick(TObject *Sender)
{
  loadinputpulse();       // chama função de carregar sinal de entrada do arquivo
if (error) Edit2->Text="erro ambrindo arquivo!!!";
  InputPanel->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TSimulform::loadcurClick(TObject *Sender)
{
  loadcurrent();          // chama função de carregar sinal de corrente do arquivo
  IinPanel->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TSimulform::talBoxClick(TObject *Sender)
{
  if (talBox->Checked){           // verificação da escolha no checkbox e atualização dos valores
   talc=1/(Nt*(B+C*Nt));
   ttalc->Text=talc*pow(10,12);
   ttalc->ReadOnly=true;
   talcxN=true;
   data();

   tIo->Text=i0/pow(10,-3);
   tg0->Text=G*1;
   tes->Text=es*pow(10,12);

  }

  else { ttalc->ReadOnly=false;talcxN=false;}
}
//---------------------------------------------------------------------------

void __fastcall TSimulform::tlambChange(TObject *Sender)
{
 lamb=tlamb->Text*pow(10,-4);    //atualização do valor de λ
}
//---------------------------------------------------------------------------//
```

APÊNDICE B - ARTIGOS PUBLICADOS

Nesse apêndice são reproduzidos dois artigos, referentes ao trabalho desenvolvido e apresentado nessa dissertação, aceitos para apresentação oral no *IMOC-99* (*International Microwave and Optoeletrônics Conference* - 9 a 12 de agosto de 1999 - Rio de Janeiro - RJ , Brasil) onde o candidato é autor e co-autor.

SOASim: a Simulator for Semiconductor Optical Amplifier with Feed Gain Control

Cristiano M. Gallep, Evandro Conforti, Juliano B. P. Custódio, Aldário C. Bordonalli,
Steven H. Ho and Sung Mo (Steve) Kang

***Abstract* — A semiconductor optical amplifier simulator (SOASim) and its modeling background are presented. SOASim has the capability to consider semiconductor optical amplifiers under amplitude-modulated input current feeding. Experimental results including all-optical pulse reshaping applications are shown with good agreement with simulations. Several gigabit rate operation may be obtained if the SOA could achieve gain recovery time below 10 picoseconds.**

***Index Terms* — feed-forward semiconductor optical amplifier simulation, all-optical pulse reshaping.**

I. Introduction

As the technology of optoeletronic integrated circuits (OEIC) becomes mature and less expensive, the semiconductor optical amplifier (SOA) may be one of the main devices for optical signal processing. This is due to the multifunctional property of SOAs to being able to amplify, detect, modulate or attenuate an optical signal [1]. However, present day SOAs are expensive and the availability of a reliable and flexible computer-aided design tool is important to cuts designing costs. Also, a software analysis tool is critical to the design of more complex systems and to facilitate the performance prediction of very high speed based data links [2]. In this sense, SOASim could optimize the design of very fast SOA systems using OEIC circuits.

II. SOASim Overview

SOASim is a C++ program designed for time-efficient simulation and optimization of semiconductor optical amplifier (SOA) subsystems, using parameters that can be obtained from component manufacturers or experimental measurements.

This simulation program can be used to optimize the SOA internal parameters such as carrier population, gain, and cavity losses. Also, SOA output waveforms, including pulse output rise time and overshoot, can be computed for a given input optical data. SOASim includes simulation of SOAs with dynamically controlled current for use in all-optical pulse reshaping applications. One practical example of such a system is the SOA incorporating feed-forward gain (FFG) control [3, 4].

Manuscript received on March 23, 1999. This work was supported in part by FAPESP, CNPq, CAPES and MCT-Pronex from Brazil

C. M. Gallep, J. P. B. Custódio A. C. Bordonalli, and E. Conforti are with the Department of Microwaves and Optics (DMO), Faculty of Electrical and Computer Engineering (FEEC), Unicamp, Caixa Postal 6101, Campinas, SP, 13081-970, Brazil, tel 55-19-788-3796, fax 55-19-289-1395, conforti@dmo.fee.unicamp.br. S. Ho and S. M. Kang are with the Univ. of Illinois at Urbana-Champaign, USA

The FFG control [3] is obtained through the modulation of the SOA injected current, as shown in Fig. 1. In this approach, the SOA feeding current is composed by a DC and a modulation component .The modulation component (AC) is set by a control signal that has the same shape as that of the optical input signal, but with a chosen phase-shift. Alternatively, a processed portion of the optical input signal can be used as the control signal. As an overall result, the control signal will impose a SOA current variation in relation to its DC level, whose shape is equivalent to that of a phase-shifted sample of the optical input signal.

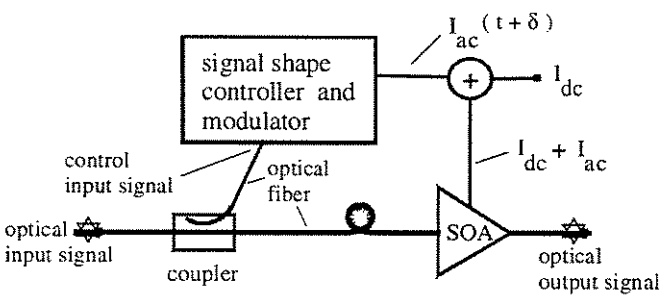

Fig. 1. Semiconductor optical amplifier with feed-forward injection current control.

The use of the simulation results for different SOA parameters allows an interactive design of a SOA based system. The program is designed to facilitate the inclusion of new models and simulation algorithms. The output files are generated in ASCII (*.dat files), containing the output signal, current injection, carrier lifetime and gain, for a given input signal. Auxiliary graphical software is necessary in order to visualize the data. The SOASim program runs over Windows95/98 and store results in a specified folder. The friendly user interface overview is shown in Fig. 2.

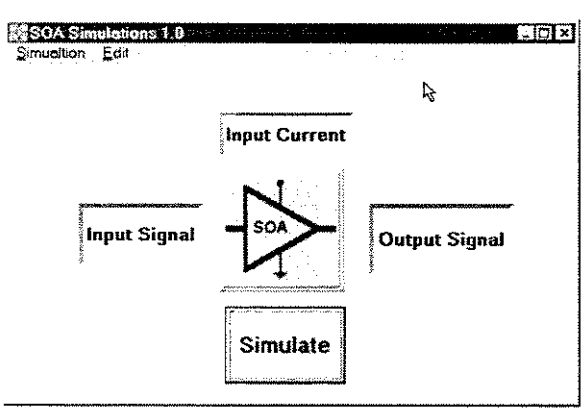

Fig. 2. SOASim interface overview.

III. SOA MODELING

By looking for a simple and accurate approach to describe the pulse amplification, the formulation of Agrawal and Olson [5] was used. A non-zero SOA active region longitudinal dimension and SI system of units are used. If the input signal is a $\hat{x}$ polarized plane wave propagating in the z direction, with optical angular frequency ω ,the electric field can be expressed as:

$$\vec{E}(x,y,z,t)=\{F(x,y)A(z,t)\exp\ j(k_0 z-\omega t)\}\frac{\hat{x}}{2} \quad (1)$$

where $A(z,t)$ is the slowly varying optical carrier envelope, $F(x,y)$ is the field transversal distribution in the active region and k_0 is the wave number. It is assumed that $F(x,y)$ is time independent and that the wave polarization is preserved during propagation in the SOA active region. Therefore, the time variation is included only in $A(z,t)$. By neglecting second derivative of $A(z,t)$ in time and space, and by using the Maxwell equations for wave propagation, the integration in the x and y directions can be performed and the envelope equation can be expressed as [5,6]:

$$\frac{\partial A(z,t)}{\partial z}+\frac{1}{v_g}\frac{\partial A(z,t)}{\partial t}=\frac{1}{2}(1+j\beta_c)g(z,t)A(z,t) \quad (2)$$

where v_g is the group velocity, β_c is the line-width enhancement factor and $g(z,t)$ is the semiconductor active region gain [5]:

$$\frac{\partial g(z,t)}{\partial t}=\frac{g_0-g(z,t)}{\tau_c}-\frac{g(z,t)P(z,t)}{E_s} \quad (3)$$

where E_s is the active region saturation energy, τ_c is the carrier lifetime and g_0 is the small-signal gain [5,6]:

$$g_0=\Gamma\sigma_g N_0(I/I_0-1) \quad (4)$$

where Γ is the guide confinement factor, σ_g is the active region transversal gain, N_0 is the carrier density required for transparency, I is the injected current in the active region and $I_0 = q\,V\,N_0/\tau_c$ is the current required for transparency.

By assuming that the pulse envelope can be divided into amplitude and phase terms, $A(z,t) = \Delta(z)\,e^{i\phi}$, where $\phi(z)$ represents the phase of the envelope and $\Delta(z) = \sqrt{P(z,t)}$, represents the square root of the optical pulse power $P(z,t)$, and by using adequate variable transformations [5,6], (2) can be rewritten as:

$$\frac{\partial P(z,t)}{\partial z}=g(z,t)\,P(z,t) \quad (5)$$

$$\frac{\partial \phi}{\partial z}=-\frac{1}{2}\beta_c\,g(z,t) \quad (6)$$

The output signal envelope power, P_{out} , and phase, ϕ_a , can be obtained by the integration of (5) over $0 \le z \le L$. After the integration, (5) and (6) become [5,6]:

$$P_{out}(t)=P_{in}(t)\exp[h(\tau)] \quad (7)$$

$$\phi_{out}(t)=\phi_{in}(t)-\frac{1}{2}\beta_C\,h(\tau) \quad (8)$$

where P_{in} and ϕ_{in} are the input signal envelope amplitude and phase, respectively. The amplifier total gain, $h(\tau)$, is given by

$$h(\tau)=\int_0^L g(z,\tau)dz \quad (9)$$

Therefore [5,6]:

$$\frac{\partial h(t)}{\partial t}=\frac{g_0L-h(t)}{\tau_c}-\frac{P_{in}(t)}{E_s}[\exp[h(t)]-1] \quad (10)$$

In order to consider the semiconductor attenuation, a term is phenomenologically introduced in (7):

$$P_{out}(t)=P_{in}(t)\exp[h(\tau)-\alpha L] \quad (11)$$

where α is the attenuation coefficient and L is the active region length.

In order to simulate the feed-forward gain control action, an injection current, with a DC level and a time-variant component is introduced here. The time-variant component is made proportional to the normalized optical input signal:

$$I(t)=I_{DC}+K\left(\frac{P_{in}(t+\delta)}{\max[P_{in}]}\right) \quad (12)$$

where $I(t)$ is the total current (modulation + DC), K is the feed-forward gain, and δ is the phase difference between the time-variant current component and the optical input signal. In order to calculate the small-signal gain as a function of the input signal, $I(t)$ is substituted in (4). In this way, the modulation depth (MD) of the intensity modulated optical output pulse can be altered by the feed-forward control of the SOA injection current, enabling the all-optical pulse reshaping. If the pulse output MD is increased, a signal enhancement is obtained [3] and if the MD is decreased to a very low level, the modulation of the optical carrier practically vanishes. In this way, the optical carrier would be ready to be modulated again in wavelength reuse applications [4]. In addition, the shape of the feed-forward injection current could be modified by fast Fourier transform techniques in order to implement the optimization of the pulse reshaping or carrier reuse process.

The carrier lifetime dependence in relation to the carrier concentration is given by [6,7]:

$$\tau_c=1/(BN+CN^2) \quad (13)$$

where B and C are, respectively, spontaneous and Auger recombination coefficients [6], and N can be calculated from [6,7]:

$$N=\frac{V}{\Gamma\sigma_g}g+N_0 \quad (14)$$

IV. Model Implementation

By using the approach shown in Section III, the SOASim algorithm was written using the software C++ Builder 3 (Borland Inc.). It works basically as shown in Fig. 3, where all time dependent variables (P_{in}, P_{out}, g, h, I) use discrete time variation $[i]$ in order to enable numerical calculation.

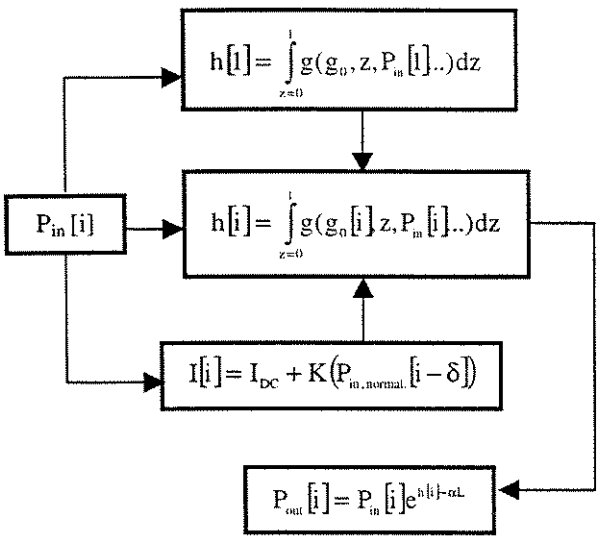

Fig. 3. Block structure of SOASim.

In order to find P_{out} from P_{in} , (10) has to be used to calculate $h(t)$ for each discrete time *"i"*. Initially, the optical input signal P_{in} is sampled for each *"i"*. Then, the program uses (12) to store the values of $I[i]$ relative to $P_{in}[i]$, for given values of I_{DC}, K and δ . After that, the calculation of $g(t=0, z)$ is performed within the interval $0 < z < L$, from the initial value of P_{in} ($P_{in}[1]$) and [6]:

$$g = \frac{g_0}{1 + P_{in}/P_s} \tag{15}$$

(where $P_s = E_s/\tau_c$ is the saturation power), by using fourth-order Runge-Kutta algorithm (RK4th) [8]. Equation (15) implements the SOA classical gain saturation effect. However, the program could also accept experimental curves of g x P in order to simulate particular SOAs, e.g., MQW (multi-quantum well) or strained layer devices.

By applying Simpson´s 3/8 rule [8], $g(t=0, z)$ is integrated over the interval $0 < z < L$. The result of the integration defines the initial condition $h[1]$ for $h[i]$. By using RK4th, $h[i]$ is determined for the entire range of time steps. Finally, (11) is used to obtain the discrete time output signal $P_{out}[i]$.

When the carrier lifetime dependence with carrier concentration is requested, the program uses $h[i]$ in each algorithm loop, (13) and (14) to calculate $\tau_c[i]$ for the next loop.

V. Program Utilization

When the program is started, the user is asked about the input signal: "*store from file or create it?* ". If the first option is chosen, the program stores the selected file in the input vector. If the second is chosen, an *input signal window* appears. This window has text boxes to edit the input signal parameters (Table I). Then the program algorithm is able to create the input signal with supergaussian pulses, using the C^{++} function:

$$P(t) = min * max + (1 - min) * max * exp\left[-\left(\frac{t - T/2}{3T/10} \right)^{2 * form} \right] \tag{16}$$

and store this vector in *.dat* (ASCII) files. Higher values of supergaussian *form* in (16) causes a faster rise-time (and fall-time) pulse.

TABLE I
Input Signal and SOA Parameters

Parameter	Unit
Wavelength	m
Maximum Power	W
Minimum Power	W
Transmission Rate	bit/s
Number of Pulses	-
Supergaussian Format	-
Carrier Density - N_t	$10^{24}/m^3$
Transversal Section Gain - σ_g	$10^{-20}.m^2$
Attenuation - α	m^{-1}
Facet Reflection – R_1,R_2	-
Line-width Enhancement Factor - β_c	-
Effective Refractive Index – n_{eff}	-
Confinement Factor - Γ	-
Width (w), Height (d) and Length (L)	m

The *input current window* appears as soon as the input signal is created, enabling the user to edit the constants K and δ of (10), and choose if they are positive or negative. After that, a current vector is created and stored in a file. The *SOA parameters window* then appears, and the user is able to change any parameter listed in Table I. Using these values, the program calculates the saturation power, the small-signal gain, the cavity loss, the SOA transparency and threshold currents, the carrier and photon lifetimes, and the active region volume. The results are presented in the same window, which also has a check box to activate the carrier lifetime dependence with carrier concentration (13). Finally, the program runs the simulation and stores the results in *.dat* files. The calculated output signal can be used as the input signal of a subsequent SOA in an array. In order to prevent file overwrite, an extension with the number of each amplifier is added to the files names.

VI. Simulation Results

In order to verify the accuracy of the simulation program, the experimental setup shown in Fig. 4 was used. After the fiber coupler, part of the optical input signal was injected in the SOA cavity. The other part was photo-detected and amplified. After amplification, the signal was processed by a filter and properly delayed in such a way to be synchronized to the SOA optical input signal. Then, after a phase-shift ϕ , with $|\delta| < \pi$) introduction, the signal was added to the DC SOA bias by using a precise voltage-to-current converter. The SOA output signal passes through an optical isolator and

a bandpass filter to maximize the SNR in the optical analyzer, where the signal arrives. The synchronism between the SOA input signal and the current injected FFG signal was obtained using a proper adjusted fiber length and a coaxial adjustable delay-line.

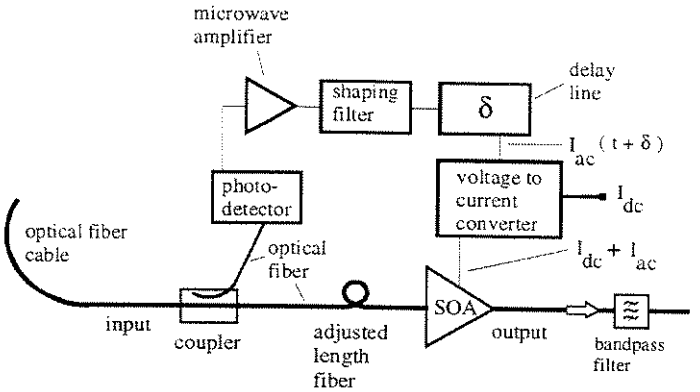

Fig. 4. Experimental setup

In order to compare the simulation with the experimental results, the FFG control by current modulation is used to enhance [3] an input signal of 600 Mb/s (Fig. 5). In this case, the current and the input signal have no phase-shift ($\delta = 0$) and K of (12) is positive. Also, the shaping filter is absent.

The output pulse-stream signals are presented in Fig. 6. The theoretical results show two curves. One is calculated for a typical SOA gain recovery time (GRT) of 400 ps. The second one is plotted for a hypothetical fast switching SOA with GRT of 7 ps. During the simulation, the SOA GRTs are supposed to be equal to τ_c of (13). Good agreement between experimental and simulation results is observed when τ_c = 400ps. Also, the pulse gradual built-up (note the pulse-shape difference from the first to third pulse in Fig. 6 for τ_c = 400ps) implies a pulse pattern dependence. The experimental existence of this dependence was initially confirmed by preliminary eye-diagrams examination techniques, but further measurements are needed to precisely check the pulse built up time span. After the pulse built up, the extinction ratio ($ER = 10.\log [(P_{base}-P_{dark})/(P_{top}-P_{dark})]$) improved from -2.84 dB to -5.88 dB.

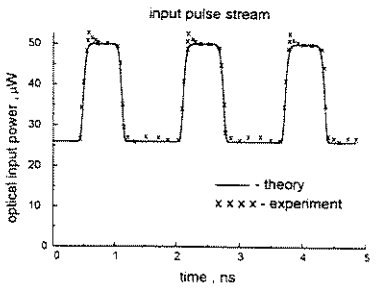

Fig. 5. Input signal envelope of 600 Mb/s, theory and experiment.

In the following figures, only the simulation results are presented for a SOA operating below saturation and with the program default parameters (Table I): $N_t = 2.5\text{x}10^{24}/\text{m}^3$, $\sigma_g = 2\text{x}10^{-20}$ m^2, $\alpha = 4000$ m^{-1}, R_1=R_2=0.0001, β_c=5, n_{eff} =3.4, Γ= 0.4, $w = 2\text{x}10^{-6}$ m, $d = 2\text{x}10^{-7}$ m and $L = 2.5\text{x}10^{-4}$ m. The input pulses have a minimum power (bit 0) of 10 μW and a maximum power (bit 1) of 100 μW, with the supergaussian format, (*form*) , equals to 30.

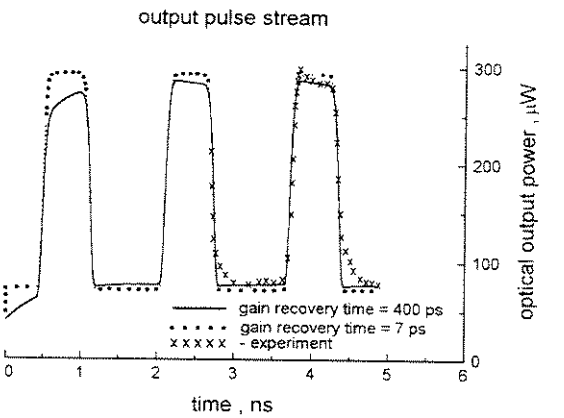

Fig. 6. Output signal envelope of 600 Mb/s, theory and experiment

Simulations are used to demonstrate enhancement (K>0) [3] and erasing (K<0) [4] possibilities of the FFG control. The bit rate of 5 Mb/s and a carrier lifetime of τ_c = 1,62 ns are used in order to avoid the pattern dependence shown in Fig. 6. SOA amplification with no FFG control is presented in Fig. 7. The output pulse presented some overshoots and an increase of the ER from -10 dB to -8.75 dB due to the SOA gain compression effects [4].

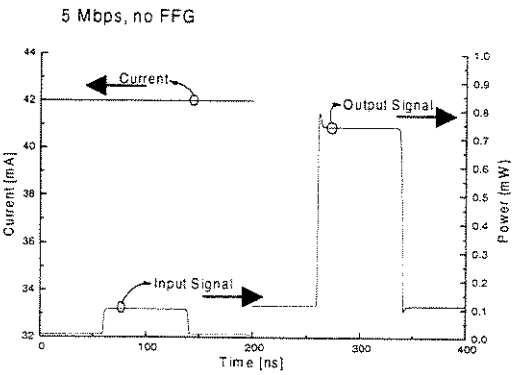

Fig. 7. Simulation result for 5 Mb/s: SOA without feed forward.

Using FFG control with no phase-shift (positive K and $\delta = 0$ in (10)), a signal enhancement is obtained, as a large ER decrease from -10 dB to -24 dB was observed with the absence of visible overshoot (Fig. 8). This result shows the possibilities of using FFG-SOAs for pulse regeneration [3] since the output pulse has a clearer distinction between the *zero* and the *one* logical levels.

The effects of a negative FFG (negative K) with $\delta = 2{,}5°$ is shown in Fig. 9. In this case, the output pulse has almost no distinction between the *zero* and the *one* logical levels (ER≈1). Therefore, an erasing amplification effect [4] could be obtained. However, the output pulse overshoots are very

large and further work is necessary to study ways of decreasing the effects of this overshoot optical noise.

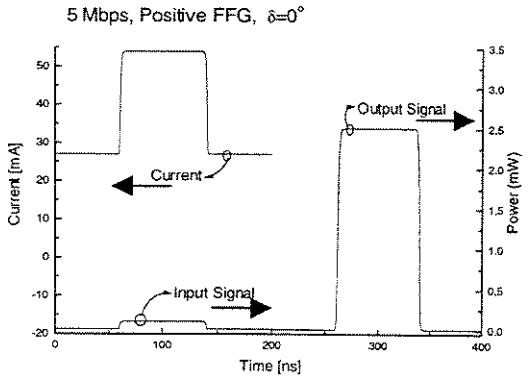

Fig. 8. Simulation result for a bit rate of 5 Mb/s: pulse enhancement (δ =0°, positive K).

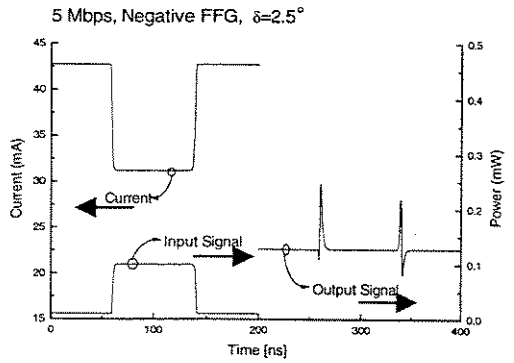

Fig. 9. Simulation result for 5 Mb/s: erasing with minimized overshoots (δ = 2.5°, negative FFG).

An example of an optical pulse shape modification is shown in Fig. 10. In this case, the FFG control parameter K is negative, with a phase-shift of δ = 90°. The output results present tree different pulse floor levels with a small overshoot.

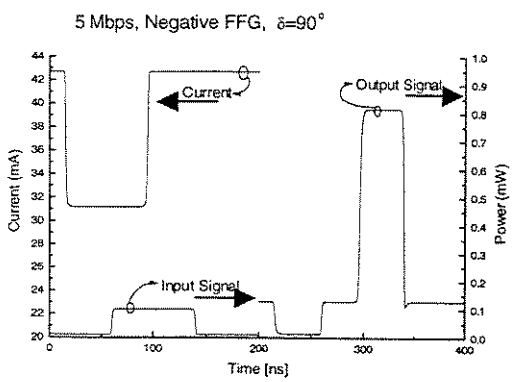

Fig. 10. Simulation result for 5 Mb/s: pulse reshaping (δ =90°, negative FFG).

In order to observe the pattern dependent phenomena, a 1 Gb/s simulation is presented in Fig. 11. This effect is due to the fact that the carrier lifetime is of the same order of magnitude as that of the bit period. The pulse built up provokes a gradual change in the upper pulse level with time, resulting in a pattern dependent pulse behavior.

Due to the slow gain recovery, no particular pulse reshape or enhancement/erasing is possible, since the amplifier is unable to answer in time to current variation. In order to obtain results like those shown for 5 Mb/s pulses, the carrier lifetime must be from five to six times as shorter as that of the bit period, that is, the carrier concentration in active region must be much higher than the standard $2.5 \times 10^{24} / m^3$.

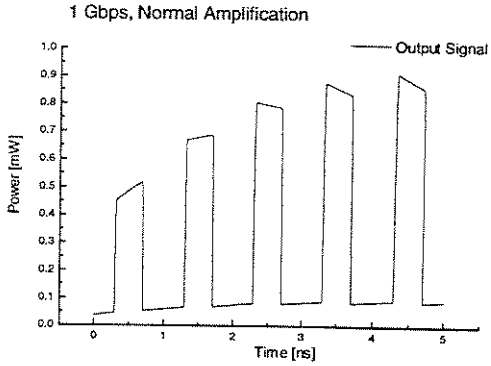

Fig. 12. Simulation result for 1 Gb/s input pulse: normal amplification.

VII. Conclusion

SOASim can be a useful tool for the design of new features based in semiconductor optical amplifiers for optical signal processing [1]. All optical pulse reshapping was simulated to illustrate the capabilities of feed-forwarding techniques [3, 4]. Future works should optimize SOA and feed-forward control parameters in order to achieve operations at signal bit rates of several Gb/s.

References

[1] –K. Bertilsson, R. Rorgren, P. A. Andrekson and S. T. Eng, "Characterization of an InGaAsP Semiconductor Laser Amplifier as a Multifunctional Device", *J. Lightwave Tech.*, Vol. 11, No. 7, pp. 1147-1150, July 1993.

[2] – J. J. Morikuni and S. M. Kang, *Computer-Aided Design of Optoeletronic Integrated Circuits and Systems*, SPIE Optical Engineering Press - 1997

[3] - E. Conforti, A. C. Bordonalli, S. Ho and S. M. Kang, "Optical 2R remodulator using feed-forward control of semiconductor optical amplifier gain", *Microwave and Optical Tech. Letters*, Vol. 21, No. 1, pp. 39-42, April, 5, 1999.

[4] E. Conforti , S. H. Ho, A. C. Bordonalli, C. M. Gallep, J. P. B Custodio, R. Simão and S. M. Kang, "Optical Carrier Reuse with Gain Compression and Feed-Forward Semiconductor Optical Amplifiers", submitted to IMOCC'99.

[5]- G. P. Agrawal and N. A. Olsson, "Self-Phase Modulation and Spectral Broadening of Optical Pulses in Semiconductor Laser Amplifiers", *J. Quantum Electr.*, Vol. 25, No. 11, pp. 2297 - 2306, November - 1989.

[6] - G. P. Agrawal and N. K. Dutta, *Semiconductor Lasers*, Van Nostrand Reinhold - 1993.

[7] - H. Ghafouri-Shiraz, *Fundamentals of Laser Diode Amplifiers*, John Willey & Sons - 1996.

[8] - *Numerical Recipes in C: The art of scientific computing*, Cambridge University Press, 1992.

Optical Carrier Reuse with Gain Compression and Feed-Forward Semiconductor Optical Amplifiers

Evandro Conforti, Steven H. Ho, Aldário C. Bordonalli, Cristiano M. Gallep, Juliano P. B Custodio, Rogério Simão and Sung Mo (Steve) Kang

Abstract — **Two techniques using semiconductor optical amplifiers (SOA) for optical carrier regeneration in networks with wavelength reuse were tested. The main objective was to recover an optical carrier by all-optical erasing its amplitude modulation. The first technique employs gain compression of deeply saturated SOAs. The second technique uses the feed-forward approach where the erasing effect is obtained by injecting a current signal with the inverse of the incoming optical pulse shape into the SOA. It was observed that the SOA gain recovery time limits the maximum usable bit rate for both techniques. Theoretical simulation showed good agreement with experimental results.**

Index Terms — **optical wavelength reuse, optical carrier regeneration, optical semiconductor amplifier, optical pulse reshaping.**

I. Introduction

The use of a cascade of semiconductor optical amplifiers (SOA) has been proposed [1] to support bypass routing in WDM optical networks. In addition to data bypassing, the amplifier cascade must also be capable of data receiving, extinction and modulation. The data extinction is the removal of the received data signals from the optical carrier and could be obtained by using the gain compression property of traveling wave optical amplifiers operating at saturation. In this way, the same wavelength channel could immediately be wavelength-reused after the data extinction process, with a possible improvement of the network switch efficiency. However, because of the large extinction ratio of the incoming pulses, the gain compression effect could not erase the data completely and the optical carrier would not have perfectly flat amplitude after data removal. Also, the dynamics of gain compression might cause transient effects on fast incoming pulses. Data extinction could alternatively be obtained by using an optical semiconductor amplifier (SOA) with feed-forward gain (FFG) control [2]. This work presents theoretical analises and experimental results for gain compressed FFG controlled SOAs for bit rates from 10 Mbit/s to 400 Mbit/s.

Manuscript received on March 08, 1999. This work was supported in part by FAPESP, CNPq, CAPES and MCT-Pronex from Brazil

E. Conforti , A. C. Bordonalli , C. M. Gallep, J. P. B Custodio, R. Simão, are with Department of Microwaves and Optics (DMO), Faculty of Electrical and Computer Engineering (FEEC), Unicamp, Caixa Postal 6101, Campinas, SP, 13083-970, Brazil, tel 55-197883796, fax 55-192891395, conforti@dmo.fee.unicamp.br. S. H. Ho and S. M. Kang are with Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1406 West Green St., Urbana, IL, 61801, USA

II. Gain Compression Approach

The digital amplitude-modulated (AM) optical carrier has two power logical levels: the low logic level and the high logic level. The optical pulse extinction ratio (ER) is defined as the ratio of the most prevalent low logic level (P_{base}-P_{dark}) to the most prevalent high logic level (P_{top}-P_{dark}):

$$ER = (P_{base} - P_{dark}) / (P_{top} - P_{dark}). \quad (1)$$

Data extinction by gain compression is based on the saturation property of an optical amplifier where the low logic level signal has a higher gain than that of the high logic level signal. Therefore, when an AM signal is amplified, the ER always increases. This is even more evident in heavily saturated SOAs. Therefore, if the AM signal crosses a sequence of heavily saturated optical amplifiers, the ER after the last amplifier is expected to be considerably close to one in such a way that it would be hard to distinguish between both the low and the high logic levels. A typical shape of the output pulse of a cascade of optical semiconductor amplifiers is shown in Fig.1. A pulse overshoot noise might be present and the associated parameters are named here: *Overshoot Amplitude Noise* (OAN) and *Overshoot Width Noise* (OWN). The OWN is measured at a level equal to P_{top} + 0.1 OAN. A completely data removal would have ER = 1 and OAN = 0. It is important to note that OWN limits the maximum bit rate since the OWN must be much narrower than the pulse width. The OWN measured value is around 500 ps for the SOA used here. Smaller OWN values could be achieved with the faster switching of Multi Quantum Well (MQW) SOAs [3]. In contrast, the measurements indicate that Erbium Doped Optical Amplifier would have a very large OWN value (in the order of 200 μs) due to its very long carrier lifetime.

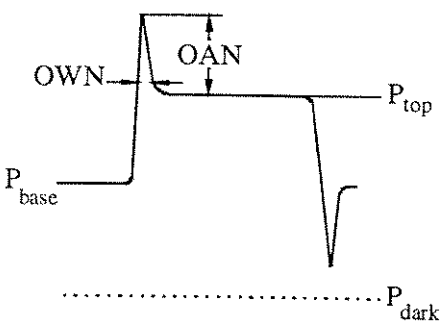

Fig. 1. Definitions of optical pulse levels and *Overshoot Amplitude Noise* (OAN) and *Overshoot Width Noise* (OWN).

The experimental setup (shown in Fig. 2) used a cascade of three semiconductor optical amplifiers (*E-TEK-HSOA-1550*), kept under temperature control (.05 K) and polarized with a precise DC current source for optimum small signal gain. Each amplifier was followed by a 47 dB fiber isolator (1 dB insertion loss) and an optical band-pass filter (4 dB insertion loss). Polarization control before each SOA was also provided. All connectors were FC/APC with a return loss better than 55 dB. The optical carrier (1550 nm, 2 mW) was generated by an external cavity laser (*Photonetics/Nanotunic*) and amplitude-modulated (AM) by a Mach-Zehnder intensity modulator (*UTP-APE MZM-3.0GHz*), driven by both a pulse generator with pseudo-random binary sequence (HP-8133A) and a dc source (ER adjustment). The output pulses arriving from the cascade of amplifiers were analyzed in a 30 GHz bandwidth optical-communication analyzer (HP-83480A/HP-83482A). The optical spectrum was also monitored. The bit rate was changed from few kHz until 1 Gbit/s.

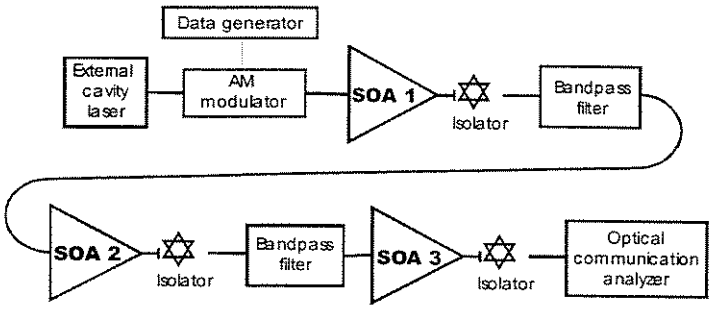

Fig. 2. Experimental setup for testing data erasing with SOA gain compression

In order to theoretically analyze the pulse propagation behavior of a SOA, the formulation of Agrawal and Olsson [4] has been used. Discrete numerical time steps were set for the time gain variation and the fourth-order Runge-Kutta method was applied to solve the rate equations with signal variation along the longitudinal direction of the SOA active region. A typical super gaussian input pulse (fitted to the experimental input pulse) is shown in Fig. 3.

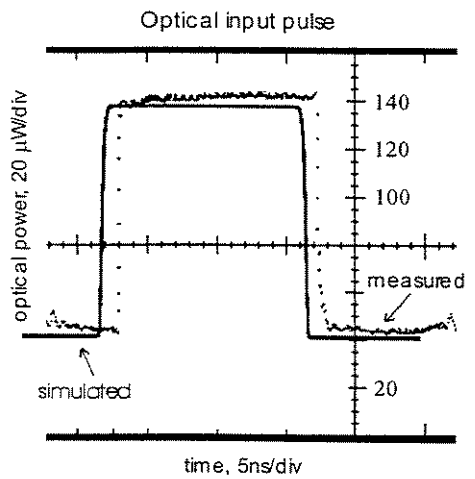

Fig. 3. Simulated and measured optical input pulses at 13 Mbit/s.

In order to test the data erasing with the gain compression scheme, the input pulse of Fig. 3 was applied to the circuit shown in Fig. 2 (in this experiment, the two filters between the SOAs were removed). The input pulse parameters were: bit rate of 13 Mbit/s, low logic level (P_{base}-P_{dark}) equal to 40 µW, high logic level (P_{top}-P_{dark}) equal to 138 µW and ER= –5.4 dB. In order to obtain deep SOA saturation, the SOAs in sequence (Fig. 2) were biased at 160 mA, 45 mA and 30 mA, leading to small signal gains of 11.7 , 3.7 and 1.3dB, respectively. The output pulse-stream after gain compression in the SOA cascade is shown in Fig. 4. The experimental output pulse parameters had a low logic level (P_{base}-P_{dark}) equal to 455 µW, a high logic level (P_{top}-P_{dark}) equal to 785 µW and ER = –2.4 dB. Note that the ER had a 3 dB increase. There is a good agreement between measured and calculated ER. The experimental and simulated OWN are similar, with a value of 440 ps. However, the measured OAN is two times as higher as the simulated OAN. The disagreement might be related to the SOA amplitude spontaneous emission (ASE) noise effects that were not considered in the simulation program.

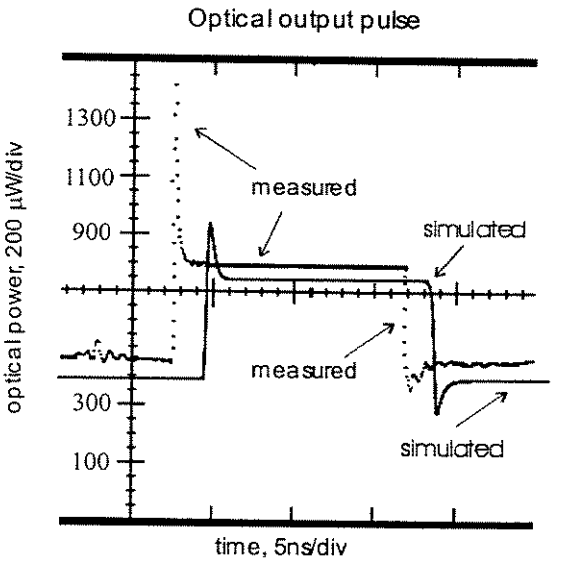

Fig. 4. Simulated and measured optical output pulses at 13Mbit/s after the SOA cascade shown in Fig. 1.

In order to analyze the limits of the ER increase as a function of the input power, theoretical results are presented in Fig. 5. A cascade of three SOAs is considered with individual gain of 12 dB, saturation power (P_{sat}) equal to 5.7 mW and a bit rate of 125 Mbit/s . Note that, for a given input pulse extinction ratio (ER_{in}) in the first SOA, the output pulse extinction ratio (ER_{out}) at the third SOA approaches the unit value when the optical input power P_{in} in the first SOA has a value around two times P_{sat}.

The SOA overshoot and undershoot are also important for data extinction since they appear as noise in the erased AM optical carrier. In order to analyze the behavior of the SOA output pulse overshoot, it is convenient to plot the normalized overshoot amplitude noise (OAN / [P_{top} - P_{dark}]). This

theoretical plot is presented in Fig. 6 for various levels of the input optical power, and having the ER_{in} as a parameter. The same conditions as those used in Fig. 5 were considered for the SOA cascade and it is possible to note that the OAN begins to decrease when the level of the input signal reaches the first SOA saturation level. In addition, theoretical results evince that the OWN puts a limit on the maximum usable bit rate. Also, the OWN is directly related to the spontaneous carrier lifetime of the SOA.

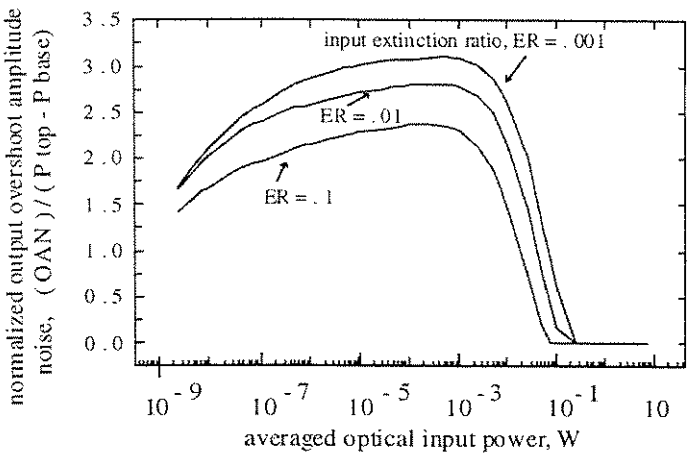

Fig. 5. Simulated pulse extinction ratio at the output of a cascade of 3 SOAs at 125 Mbit/s, as a function of the optical pulse input power.

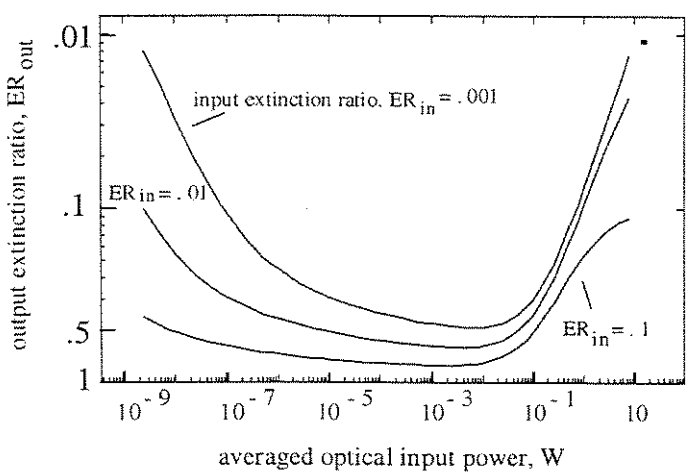

Fig. 6. Simulated normalized overshoot amplitude noise at the output of a cascade of 3 SOAs at 125 Mbit/s, as a function of the optical pulse input power.

III. Feed-forward approach

It was shown in Fig. 5 and 6 that very high levels of optical input signals are required for data extinction in AM modulated optical carriers. However, high levels of optical signals could not be a practical choice and to increase the output ER in a value close the unit might need some kind of synchronous active optical gain control. One way to implement such idea is to use an optical semiconductor amplifier (SOA) with feed-forward gain (FFG) control [2]. The technique consists of modulating the bias current of a SOA with an electrical current signal whose shape matches that of the amplitude-modulated optical carrier being simultaneously coupled into the active region of the SOA. However the electrical injected current should have a phase shift of 180° in relation to the optical signal. One way of providing the required shape matching and the 180° phase shift is by converting a sample of the incoming optical pulse into an electronic signal and inverting this signal. This current-converted electronic signal is then feed-forwarded into the SOA through its bias current, provided that the optical signal and the inverted current signal are synchronized in relation to each other. The signal synchronization leads to an interesting effect. As the pulse rises towards its high level, the current pulse moves towards its minimum. This results in a lower overall SOA gain as the carrier population decreases due to a smaller current pulse injection and higher photon population. However, for lower optical pulse levels, the opposite effect occurs and the SOA gain rises from its average value. Therefore, higher optical signal levels will be amplified with a lower optical gain, and lower level signals with higher SOA gain. In this way, the extinction ratio of the resulting optical pulse will be close to the unit if a convenient amount of feed-forward current is injected in the SOA terminal. Therefore, data extinction or equalization between the zero and the one logical levels could be obtained with the FFG scheme.

In the past, SOA with feed-forwarding current was employed for compensation of SOA nonlinearities [5]. The modulation of the SOA injected current has also been used to provide externally modulated optical signals [6]. Therefore, the SOA data extinction can be considered as a combination of the feed-forward current technique with the SOA external modulation characteristics as the synchronism between the inverted feed-forward current and optical pulse amplification is established [2]. The FFG scheme is presented in Fig. 7.

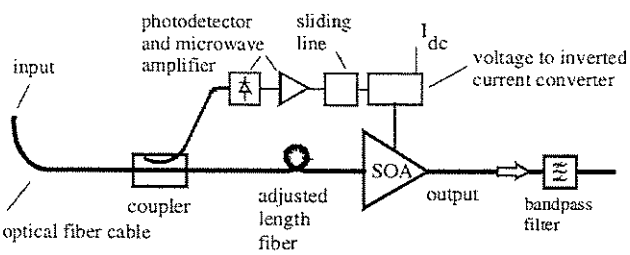

Fig. 7. Block diagram of the feed-forward gain-controlled semiconductor optical amplifier.

An optical AM modulated carrier (1550 nm, 2 mW) was generated as described in Section II. The carrier was applied to the feed-forward SOA circuit of Fig. 7. Part of the optical signal was photo-detected, generating an electronic signal that was amplified and fed into the SOA (*E-TEK, HSOA-1550*) by a high-speed voltage-to-inverted-current converter. It is important to note that the photo-detector circuit cuts off any dc component. Therefore, only the transitions of the detected optical signal are amplified. The other part of the incoming optical signal was coupled into the SOA active region. A proper time delay compensation was provided

(sliding line and adjusted fiber length) in order to obtain the synchronization between the feed-forward current and the optical pulse amplification process, ensuring that the SOA gain changes in accordance with the optical input signal intensity variation. The technique results in an increase (decrease) of the SOA gain in relation to the negative (positive) difference between the optical signal and its average component. The overall effect yields an equalization of the pulse transitions in the output of the feed-forward circuit output.

In order to test the FFG scheme, an input signal with very low ER was generated. The input pulse parameters were: bit rate of 13 Mbit/s, low logic level $(P_{base}\text{-}P_{dark})$ equal to 5 μW, high logic level $(P_{top}\text{-}P_{dark})$ equal to 174 μW and ER = –15.4 dB. The SOA polarization current and the microwave amplifier gain were adjusted for optimum FFG performance. The output pulse-stream after FFG is shown in Fig. 8. The experimental output pulse parameters had a low logic level $(P_{base}\text{-}P_{dark})$ equal to 6 μW, a high logic level $(P_{top}\text{-}P_{dark})$ equal to 9 μW and ER = –1.7 dB. Note that the ER had a 13 dB increase. Also, the FFG parameters can be adjusted for a ER very close to unit (0 dB). However, the overshoot noise is very high as can be seen in the FFG output pulses of Fig. 8. The simulated output pulses are very close to the measured pulses. It is interesting to note that the optimum FFG current is not exactly at a 180° phase-shift from the optical pulse. In reality, for various simulation tests, the optimum theoretical phase shift is around 220°. Also, the time lags between the overshoots are different for the simulated pulses of Fig. 8. Further work will be need to explain these results.

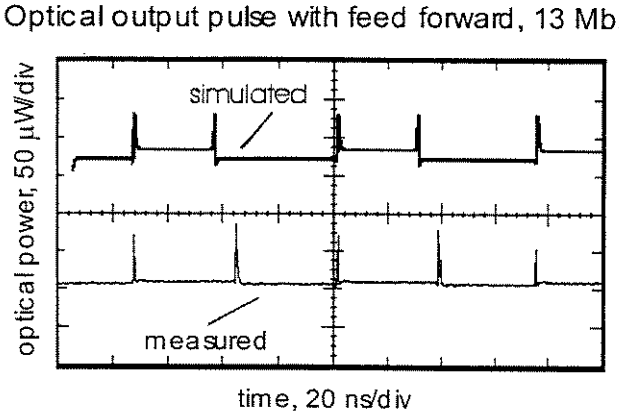

Fig. 8. Output pulses optimized for data extinction, using the feed-forward gain (FFG) controlled semiconductor optical amplifier at 13 Mbit/s.

In order to obtain a clear picture of the overshoot spikes of Fig. 8, the pulse stream data rate was increased to 400 Mbit/s. The super gaussian input pulse was fitted into the experimental input pulse. In this case, the employed simulation did not consider the experimental pulse overshoot, as shown in Fig. 9. The pulse input pulse parameters were: bit rate of 400 Mbit/s, low logic level $(P_{base}\text{-}P_{dark})$ equal to 49 μW, high logic level $(P_{top}\text{-}P_{dark})$ equal to 123 μW and ER = – 4 dB. The SOA polarization current and the microwave amplifier gain were adjusted for optimum FFG performance. The output pulse-stream after the FFG-control is shown in Fig. 10. The experimental output pulse parameters had a low logic level $(P_{base}\text{-}P_{dark})$ equal to 60 μW, a high logic level $(P_{top}\text{-}P_{dark})$ with the same value and ER ≈0 dB. However, the pulse overshoot is very large. The measured OWN was 750 ps. However the OWN measured in Fig. 3 without FFG is 440 ps. It may be possible that the OWN increase is a consequence of the dynamic FFG effects in the SOA carrier population. In fact, the simulated results confirms the 750 ps value of the OWN. Also, the simulated pulse has a similar format of the measured pulse but the transition before the OWN is much larger in the simulated waveform. The current pulse and optical pulse phase-delay in the simulation was 217.8 °.

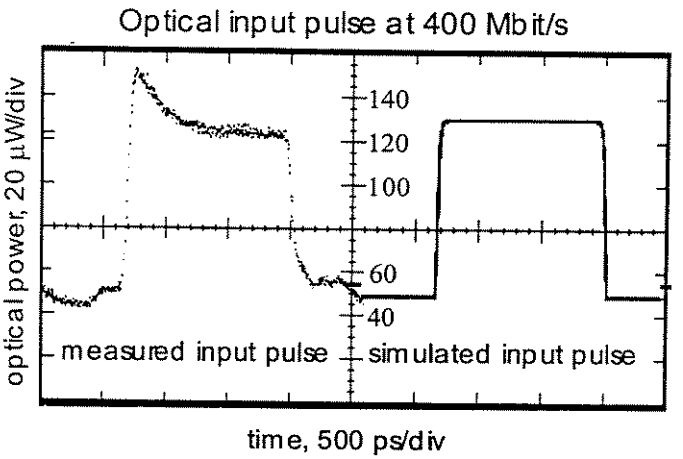

Fig. 9. Pulses at the input of the feed-forward gain (FFG) controlled semiconductor optical amplifier at 400 Mbit/s.

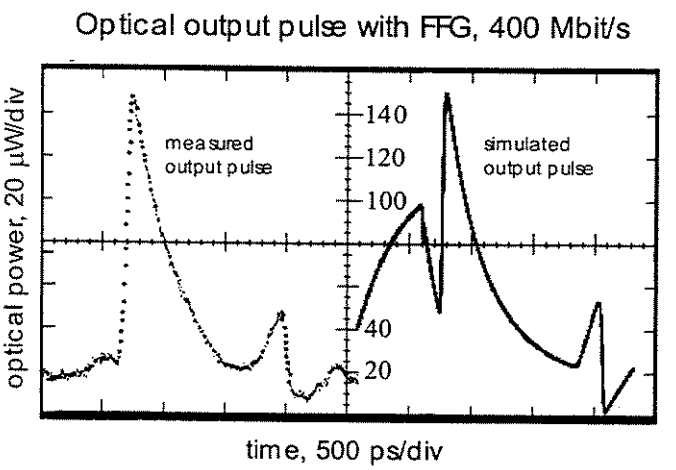

Fig. 10. Output pulses optimized for data extinction, using the feed-forward gain (FFG) controlled semiconductor optical amplifier at 400 Mbit/s.

It is interesting to note that in order to provide the desired extinction ratio, a higher bit rate requires a higher microwave amplifier gain due to the decrease of SOA response with higher input current frequencies. In addition, for the SOA used here, the OWN has a value around 750 ps. This is due to the finite rise time of the SOA gain switching action, which is limited by the carrier lifetime. For the SOA used in the

experiment, the estimated carrier life time was 400 ps, leading to an upper limit for the bit rate around 500 Mbit/s. However, to achieve data extinction using FFG in optical communication systems, operation at several GBit/s are needed with low OAN and very short OWN. It should be emphasized that the operation at those high speeds should wait for optical amplifiers with considerably lower carrier lifetimes since the lowest SOA carrier lifetime reported in the literature is 200 ps [3].

When several of these FFG circuits are cascaded in an all optical WDM network application, time jitter accumulation and amplitude spontaneous noise (ASE) should also be taken into consideration. In addition to that, data extinction can be obtained for any channel of a WDM system but not for all channels simultaneously, since WDM signals have to be demultiplexed to each individual wavelength and then sent through the FFG for data erasing.

IV. Conclusions

It was demonstrated that the gain compression effect and the switching action of a semiconductor optical amplifier in a feed-forward arragement can be used for optical data extinction of an AM optical carrier. However, pulse overshoot appear as a significant source of noises in the erased waveform. The current SOA gain dynamic is directly related to this noise and imposes a limit in the data extinction performance.

One of the advantages of the FFG technique is that the effect of the delay line slider could be obtained with a proper choice of RLC and varactor circuits. In this way, the FFG-SOA data extinction could be achieved by monolithic system integration with a possible cost reduction. Future works should incorporate overshoot noise reduction, pseudo-random bit streams, ASE noise, jitter accumulation and SOA generated chirping analyses in a cascaded system.

References

[1] S. H. Ho, E. Conforti and S. M. Kang , " LDOT - Life-range Delimitation Optical Transceiver for fast routing and multicasting in Wavelength Division Multiplex (WDM) Local Area Networks", *Optical Fiber Communication Conference,* Vol. 4, 1994 Technical Digest Series (Optical Society of America, Washington, D.C., 1994) , pp. 171-174.

[2] E. Conforti, A. C. Bordonalli, S. Ho and S. M. Kang, "Optical 2R remodulator using feed-forward control of semiconductor optical amplifier gain", *Microwave and Optical Tech. Letters*, Vol. 21, No. 1, pp. 39-42, April, 5, 1999.

[3] J. M. Wiesenfeld, A. H. Gnauck, G. Raybon, U. Koren, "High-speed multiple-quantum-well optical power amplifier," *IEEE Photon. Technol. Lett.*, **4**, pp. 708-711 (1992)

[4] G. P. Agrawal and N. A. Olsson, "Self-phase modulation and spectral broadening of optical pulses in semiconductor laser amplifiers," *IEEE J. Quantum Electron.* **25**, 2297- 2306 (1989)

[5] A. A. M. Saleh, R. M. Jopson, and T.E. Darcie, "Compensation of Nonlinearity in Semiconductor Optical Amplifiers," *Electron. Lett.*, **24**, no. 15, , pp. 950-952 (1988)

[6] U. Koren, et al. "High Frequency Modulation of Strained Layer Multiple Quantum Well Optical Amplifiers," *Electron. Lett.*, **27**, no.1, pp. 62-63 (1991)