UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RICARDO CARVALHO DE MELOS

TRAPSIMULATOR – UM SIMULADOR DIDÁTICO DE RUIDO RTN

Porto Alegre
2018

RICARDO CARVALHO DE MELOS

TRAPSIMULATOR – UM SIMULADOR DIDÁTICO DE RUIDO RTN

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, da Universidade Federal do Rio Grande do Sul, como parte dos requisitos para a obtenção do título de Mestre em Engenharia Elétrica.

Área de concentração: Engenharia da Computação

ORIENTADOR: Prof. Dr. Gilson Inácio Wirth

Porto Alegre

2018

RICARDO CARVALHO DE MELOS

TRAPSIMULATOR – UM SIMULADOR DIDÁTICO DE RUIDO RTN

Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Engenharia Elétrica e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____

Prof. Dr. Gilson Inácio Wirth, PPGEE - UFRGS

Doutor pela Universitaet Dortmund, Dortmund, Alemanha.

Banca Examinadora:

Prof. Dr. Altamiro Amadeu Susin, UFRGS

Doutor pelo Institut National Polytechnique de Grenoble – Grenoble, França

Prof. Dr. Tiago Roberto Balen, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Marcelo de Oliveira Johann, UFRGS

Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Coordenador do PPGEE: _____

Prof. Dr. Valner João Brusamarello

Porto Alegre, março de 2018.

DEDICATÓRIA

Dedico este trabalho a Deus, que sempre esteve ao meu lado.

AGRADECIMENTOS

À minha esposa e filha que foram razão de todo o meu esforço, e por abdicarem de momentos em família para que eu pudesse alcançar minhas metas.

Aos meus pais, que formaram a base do meu caráter, disciplina e determinação, e pelo seu apoio incondicional.

Aos meus colegas do Grupo de Estudos de Micro e Nano Eletrônica do PPGEE, que me ajudaram compartilhando seus conhecimentos pelo período que estive lá, em especial aos mestrandos, naquela época, Gabriela Firpo Furtado e Thales Becker.

Ao meu orientador, Prof. Dr. Gilson Inácio Wirth, pelo apoio.

Ao Programa de Pós-Graduação em Engenharia Elétrica, PPGEE, pela oportunidade de realização dos trabalhos em minha área de pesquisa.

Aos demais colegas do PPGEE pelo seu auxílio nas tarefas desenvolvidas durante o curso e apoio na revisão deste trabalho.

À CAPES pela provisão da bolsa de mestrado.

RESUMO

TrapSimulator é uma ferramenta didática de simulação de corrente elétrica sob efeito do ruído *Random Telegraph Noise*, presente nos dispositivos eletrônicos semicondutores, mais precisamente em transistores de efeito de campo. Esta ferramenta possibilita a execução de simulações rápidas, de fácil visualização e usabilidade, isto porque, além de ser um software de código aberto, disponibilizado sob licença *Gnu Generic Public License*, é acessível através de um simples navegador de internet. Seu propósito principal é levar ao ambiente de sala de aula, uma forma de visualização do ruído no domínio do tempo e frequência, utilizando uma infraestrutura básica, presente na maioria das universidades. TrapSimulator implementa em seu algoritmo, métodos de simulação de um vetor que representa a corrente *Id(t)* sob efeito de ruído citado anteriormente. Este ruído é modelo conforme o conceito Trapping De-Trapping, o qual afirma que a causa da flutuação do nível de corrente é devida a captura e emissão de portadores por defeitos localizados no dielétrico dos transistores, próximo à interface entre este e o canal ativo do dispositivo. A motivação por executar este trabalho se dá pelo interesse em contribuir com o aprendizado de estudantes de graduação da área de Engenharia Elétrica e Eletrônica. Baseando-se numa metodologia empregada no ensino tecnológico, chamada de Aprendizagem Baseada em Projetos, este trabalho propõe um processo cognitivo mais atuante, priorizando a busca mais ativa pelo conhecimento.

Palavras-chave: RTN, TrapSimulator, Simuladores, ABP, Aprendizagem Baseada em Projetos.

ABSTRACT

TrapSimulator is a current simulation didactic tool under Random Telegraph Noise effect, present in electronic semiconductor devices, more precisely in field effect transistors. This tool makes it possible to perform fast, easy-to-view and usability simulations, because it is open source, made available under a Gnu Generic Public License, accessible through a simple web browser. Its main purpose is to take the classroom environment, a way of noise visualizing in the time and frequency domain, using a basic infrastructure, present in most universities. TrapSimulator implements in its algorithm, simulation methods that represents the $I_D(t)$ current vector under noise effect quoted above. Random Telegraph Noise is modeled according to the Trapping De-Trapping concept, which establishes that current level fluctuation is due to carriers capture and emission by defects located at dielectric region, next to active channel on semi-conductor devices. The work motivation is due to the desire to contribute to the undergraduate students learning in Electrical and Electronic Engineering area. Based on methodology used in technological teaching, called Project-Based Learning, this work proposes a cognitive process more directed to making hands on, the most active search for knowledge.

Keywords: RTN, TrapSimulator, Simulators, PBL, Project-Based Learning.

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ABP: Aprendizagem Baseada em Projetos

CI: Circuito Integrado

CMOS: *Complementary Metal-Oxide Semiconductor*

DFT: *Discrete Fourier Transform*

FFT: *Fast Fourier Transform*

GPL: *Generic Public License*

MOSFET: *Metal-Oxide Semiconductor Field Effect Transistor*

NCN*: Network for Computational Nanotechnology*

PDF*: Probability Density Function*

PPGEE: Programa de Pós-Graduação em Engenharia Elétrica

RAPPTURE: *Rapid Application Infrastructure*

RTN: *Random Telegraph Noise*

RTS: *Random Telegraph Signal*

1. INTRODUÇÃO

Nas últimas décadas, têm se percebido uma mudança comportamental nas unidades acadêmicas, que pode ser atribuída, em sua maior parte, pelo avanço tecnológico e pela informação em tempo real.

Atualmente, é possível se armazenar, em um smartphone, todo o conteúdo digitalizado de uma biblioteca de médio porte. Aplicativos móveis podem registrar a experiência de seus usuários, e com rapidez, encontrar as suas preferências, localização geográfica, entre outras informações. Ainda, podem sugerir novas tendências e atrativos baseados em inteligência artificial e aprendizado de máquina. Boa parte dessa tecnologia, com apelo consumista, tem transformado a forma de vida das pessoas, afetando inclusive, o universo acadêmico escolar.

As metodologias pedagógicas tradicionais não têm se mostrado tão eficientes perante esta evolução da sociedade. Neste novo cenário, globalizado, flexível e de estímulo ao aprendizado contínuo, estas não se sustentam e já se mostram menos adequadas (WAGNER, 2012).

Como alternativa à esta evolução que atinge amplamente o ensino, estudos têm sido feitos sobre uma metodologia ativa de aprendizado chamado de Aprendizagem Baseada em Projetos (ABP). Cientificamente é comprovado que alunos têm sua capacidade cognitiva aumentada quando exposto a um ensino mais voltado a atividades práticas (WAGNER, 2012) (LEITE, 2017).

A ABP tem como base educacional, não mais unicamente a transferência de conhecimento, e sim o estímulo à sua busca. Agora, o professor não é mais um canal de fornecimento de informação, ele deixa de ter uma postura passiva e passa a ter uma postura mais ativa e dinâmica, agindo mais como orientador do que como avaliador (BRAGA, 2002).

No Relatório da Comissão Internacional sobre a Educação no Século XXI, a UNESCO descreve a educação “como desenvolvimento humano por meio da construção, pelas pessoas, de competências e habilidades que lhes permitam alcançar seu desenvolvimento pleno e integral”. “Aprender a conhecer”, “aprender a fazer”, “aprender a conviver” e “aprender a ser”, sendo estas as dimensões que devem ser atingidas no decorrer da vida acadêmica *(The Four Pillars of Learning*).

Muitas áreas de conhecimento já adotam esta visão curricular em suas grades, com estágios profissionalizantes e cursos de pós-graduação residentes (KRISHNAN, 2013). Na área da engenharia não poderia ser diferente. Estudos demonstram que a adoção da ABP (BRAGA, 2002) tem proporcionado o desenvolvimento de pensamento crítico, condizente com a realidade, através da relação teoria-prática, maior desenvoltura dos estudantes quando expostos à tomada de decisões, desenvolvimento de habilidades, atitudes e postura ética. Diante dessas constatações, tende-se a formação de um engenheiro mais adaptável e flexível, com uma visão sistêmica e holística.

Motivado por este novo paradigma, este trabalho propõe uma contribuição voltada para a aprendizagem mais ativa – a construção de uma ferramenta computacional de simulação para ser adotada nesta nova abordagem educacional que visa estimular a busca pelo conhecimento por parte do estudante.

O escopo do projeto é trazer uma visão qualitativa (de primeira ordem) da concepção de ruído *Random Telegraph Noise* (RTN), possibilitando a visualização da relação entre causa e efeito. O estudante e o professor poderão fazer uso deste simulador em seus estudos, bem como em ambiente de sala de aula, já que a plataforma que hospeda o software, que faz parte de um programa acadêmico voltado para a Micro e Nano Tecnologia, Nanohub.org, pode ser acessada através de um simples navegador de internet. Tal plataforma é mantida pela *Purdue*

University, West Lafayette, Indiana, EUA e administrada pelo grupo de estudo *Network for Computational Nanotechnology* (*NCN*) (LUNDSTROM e KLIMECK, 2006).

O algoritmo utilizado na simulação da corrente de dreno em um dispositivo semicondutor MOSFET sob o efeito de ruído é baseado no conceito *Trapping De-Trapping* (GRASSER, 2014) que atribui sua causa à captura e emissão de portadores de carga por armadilhas, localizadas no dielétrico do dispositivo, ao longo de toda a interface deste dielétrico e a sua região ativa.

Este trabalho abordará no Capítulo 2 um estudo feito sobre a Aprendizagem Baseada em Projetos, no Capítulo 3 a fundamentação teórica do ruído RTN. O Capítulo 4 será dedicado exclusivamente a descrição do software de simulação e o algoritmo utilizado e finalmente no Capítulo 5 conterá as conclusões finais e sugestões para novos trabalhos.

2. APRENDIZAGEM BASEADA EM PROJETOS

As estruturas educacionais, bem como, seus métodos de ensino, têm sido alvo de frequente evolução devido às novas abordagens pedagógicas oferecidas pelos avanços tecnológicos e também pelas mudanças comportamentais tanto do corpo discente como do docente (WAGNER, 2012 e LEITE, 2017).

A velocidade com que a informação se propaga atualmente é responsável pelo novo modelo de aprendizado onde se busca a aquisição rápida e direta de conhecimento, muitas vezes um conhecimento local, sem base teórica, para resolver um problema pontual. Tal modelo tem sido observado nos últimos anos na maioria dos estudantes.

Estudos na área da psicologia, voltados para a análise comportamental de estudantes, mostram que o aprendizado obteve melhores resultados quando metodologias práticas foram adotadas como ferramenta de ensino (LEITE, 2017).

A grande questão aqui é que a atividade prática leva à visualização em tempo real da implementação de conteúdos teóricos, que muitas vezes são maçantes e meramente expositivos. Neste tipo de atividade, o interessado pode obter respostas na prática sobre o comportamento, no caso da área da Engenharia Elétrica, de circuitos e componentes ao alterar seus principais parâmetros, por exemplo.

Outras áreas de conhecimento muito importantes, como a medicina e a odontologia, por exemplo, já utilizam de “metodologia ativa de aprendizagem” com estágios práticos e cursos de pós-graduação residentes. Já nas áreas tecnológicas, como as Engenharias, de um modo geral, esse formato vem sendo discutido e aplicado sob a nomenclatura de Aprendizagem Baseada em Projetos (ABP) (WAGNER, 2012 e LEITE,2017).

O desenvolvimento da metodologia da aprendizagem baseada em projetos teve suas origens em 1900, quando o filósofo americano John Dewey (1859 – 1952) comprovou o

“aprender mediante o fazer”, valorizando, questionando e contextualizando a capacidade de pensar dos alunos numa forma gradativa de aquisição de um conhecimento relativo para resolver situações reais em projetos referentes aos conteúdos na área de estudos, que tinha como meta o desenvolvimento dos mesmos, no aspecto físico, emocional e intelectual, por meio de métodos experimentais (MCDERMOTT, 1981).

A eficiência da ABP pode ser comprovada por meio da aferição dos conceitos apreendidos durante as aulas práticas, na área das engenharias, onde se pode constatar que a aprendizagem é mais efetiva que nas aulas teóricas. A busca constante pelo desenvolvimento de novas metodologias de ensino é a razão básica do crescimento e da popularidade da Aprendizagem Baseada em Projetos, objetivando uma transformação constante. Assim existe a necessidade de se incorporar o recente pensamento sobre padrões e avaliação, com a finalidade do delineamento do processo de planejamento para projetos focados em padrões (STEPIEN e GALLAGHER, 1998).

Esta estratégia de ensino pode exigir muito mais empenho de alunos e professores. Uma mudança na postura em classe – que antes era de um especialista – se faz necessária, trazendo o docente mais para um âmbito de treinamento. A ABP tem sido foco de discussões e é tratada como a Estratégia de Cognição do século (BELL, 2010).

Os estudos acerca da metodologia ABP têm se enriquecido com os conhecimentos sobre a gênese do processo cognitivo, da aprendizagem do aluno e da fisiologia da memória, ressaltando-se a importância da experiência prévia e da participação ativa como pontos fundamentais para a motivação e aquisição de conhecimentos. O objetivo, primeiramente, é conscientizá-lo do que ele sabe e do que precisa aprender e o motivar a buscar informações relevantes. Enfim, esta metodologia busca estimular no aluno a capacidade de “aprender a aprender”, de trabalhar em equipe, de ouvir outras opiniões (mesmo que contrárias às suas),

induzindo-o a assumir um papel ativo e responsável pelo seu próprio aprendizado (WAGNER, 2012 e LEITE, 2017).

Nesta metodologia, o professor deixa de ser o meio de transmissão do saber e passa a ser o estimulador de busca por aprendizagem e descoberta de conhecimento. Agora, no papel de orientador, cabe ao docente criar pontos de discussão referente à dinâmica do assunto a ser tratado, bem como, definir a direção certa para o alcance do objetivo proposto. Então o aluno consegue produzir um exame analítico e apurado de um dado problema, de forma a traçar metas e buscar objetivos sob um olhar mais de crítico e menos de mero espectador.

Uma forma de abordagem da ABP é a inserção dos alunos em plataformas virtuais, podendo-se integrar novas ferramentas de disseminação de conhecimento, tornando possível que a metodologia de ensino acompanhe a evolução tecnológica (PSOTKA, 2013). A exposição dos jovens a tais tecnologias diminui a barreira que existe entre seu novo aspecto cognitivo e o formato curricular ultrapassado. É fato que a grade curricular atual não gera mais os mesmos resultados, e que esta não se adapta totalmente as novas maneiras de se construir conhecimento.

O resultado de estudos com a utilização de ambientes virtuais combinados com ambientes reais oferece um suporte educacional que contribui para melhor aprendizagem e absorção de conteúdos didáticos por abordar uma maneira diferenciada de ensino.

O uso de ambientes virtuais é crescente e tem trazido muitas contribuições para diversas áreas do conhecimento. Algumas disciplinas, da área de computação, apresentam altas taxas de abandono e retenção do aluno, causadas pela dificuldade em compreender conceitos abstratos pelo método de ensino tradicional, que são baseados em aulas expositivas sem interações entre os alunos, gerando baixa motivação e, consequentemente, a falta de interesse em aprender.

Finalmente, pode-se afirmar que o perfil do estudante mudou severamente nos últimos anos. Esta nova geração é mais dinâmica, atuante, interessada em interagir mais do que em assistir, desde que, estimulada corretamente. Jovens nascidos em meio a informação em tempo

real e ao acesso ao conhecimento em qualquer hora e lugar não devem ter desperdiçado todo este dinamismo com estruturas pedagógicas antigas que formavam estudantes carregados de conceitos abstratos sem vivência prática daquilo que aprenderam. Na mesma linha evolutiva, o ensino, bem como sua estrutura, deve seguir na inovação que a situação atual exige (TEREZINHA, MIRANDA, MUNHOZ e CASTANHEIRA, 2012).

3. MECANISMOS FUNDAMENTAIS DO RUÍDO

Ruídos são perturbações em circuitos ou dispositivos, com comportamento aleatório, devidos à natureza física do material que os compõem, ou por ação externa, como por exemplo, vibração, incidência de luz, interferência causada por circuitos adjacentes, entre outros. A corrente *I(t)* que circula em um dispositivo , pode ser definida pela soma da corrente média $\bar{I}$e a flutuação aleatória causada pelos ruídos no instante *t*.

Atualmente, estudos relacionados à modelagem de ruídos têm ganhado grande interesse, à medida que transistores, com dimensões cada vez menores, têm sido desenvolvidos. A variabilidade dos processos e materiais envolvidos na construção de Circuitos Integrados (CIs) são fatores determinantes para seus desenvolvimentos, o que torna ainda mais importante conhecer as causas e efeitos dos ruídos nestes dispositivos. A Figura 1 mostra a composição de um ruído presente em um dispositivo MOS (HAARTMAN, 2007).

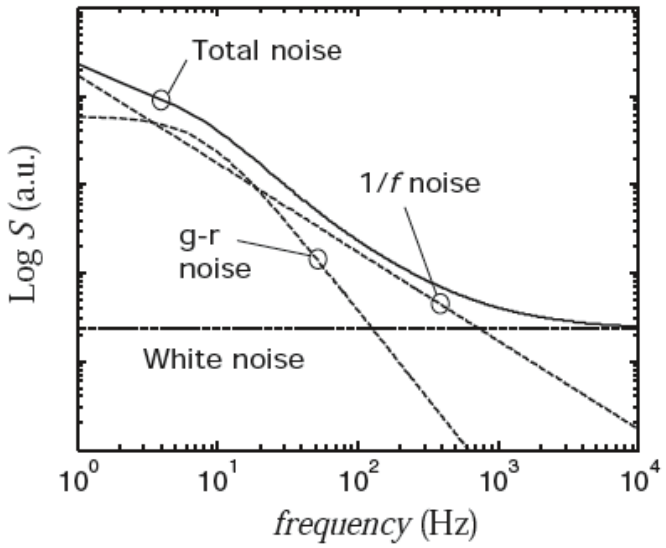

Figura 1 - Composição do Ruído em um transistor MOS

Quando elétrons em movimento, devido a sua energia térmica, colidem com outros elétrons, ambos se espalham em direções aleatórias, causando uma flutuação na corrente. Este ruído, presente em toda a matéria resistiva de um componente é chamado de **Ruído Térmico**, ou Ruído Branco, pois sua Densidade Espectral se mantém constante para qualquer frequência.

Outro tipo de ruído, o **Ruído Shot**, também chamado de Ruído Quântico, ocorre devido o fluxo de elétrons, cuja carga é uma grandeza discreta, quando estes atravessam uma barreira de potencial, causando a flutuação na corrente (HAARTMAN, 2007).

O **Ruído de Geração e Recombinação** (GR) é devido às flutuações no número de portadores livres no interior do dispositivo, associadas às transições randômicas de cargas livres entre estados com diferentes níveis de energia. Exemplos típicos de transições são os que acontecem entre a banda de condução e níveis localizados dentro da banda proibida, e entre a banda de condução e valência. Entretanto, o ruído GR é devido inerentemente às flutuações no número dos portadores, normalmente mantendo a neutralidade de carga da amostra (KANDIAH e WHITING, 1978).

Uma atenção especial será dada neste trabalho ao ruído chamado ***Random Telegraph Noise*** ou *Random Telegraph Signal* (RTN ou RTS), visto que este é o ruído simulado pelo software desenvolvido neste trabalho. Diferentemente do ruído GR, que ocorre no corpo do dispositivo, este ruído é exclusivamente gerado na interface entre o dielétrico e o canal ativo dos transistores, portanto, exclusivo de dispositivos de efeito de campo. A seção 3.1 explicará os fundamentos deste tipo de ruído.

3.1. RANDOM TELEGRAPH NOISE

Em dispositivos eletrônicos muito pequenos, da ordem de poucas dezenas de nanômetros, a captura e emissão aleatória de portadores por uma armadilha gera uma flutuação discreta na resistência do dispositivo semicondutor como um sinal de telégrafo, e a este

fenômeno se dá o nome de ruído *Random Telegraph Noise* (RTN). Algumas literaturas demonstram que o RTN é a origem do ruído de baixa frequência (*1/f*) nestes dispositivos, e fornecem uma nova visão sobre a natureza dos defeitos em uma interface (WIRTH, SILVA e BREDERLOW, 2007). Vale salientar que essa afirmação não é unânime dentre os pesquisadores da área.

Este fenômeno de emissão e captura de portadores, definido pelo conceito *Trapping De-trapping* (GRASSER, 2014), gera variações na tensão de limiar e na mobilidade de portadores de cargas no canal ativo dos dispositivos. Estas variações se refletem em uma flutuação no nível da corrente de dreno. O nome do conceito que descreve este fenômeno origina do fato que armadilhas, ou do inglês "*traps*" aprisionam os portadores de carga no dielétrico dos dispositivos CMOS, junto à interface entre o dielétrico e o seu canal ativo. A Figura 2 demonstra um evento de captura e emissão de um portador causado por uma armadilha, bem como sua localização no dielétrico junto à interface.

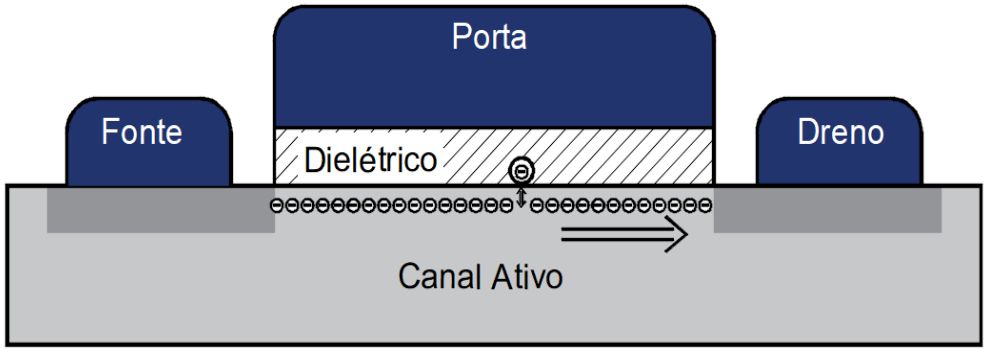

Figura 2 - Representação de evento de captura de portador por uma armadilha presente em um transistor de efeito de campo

Seus parâmetros mais relevantes são o tempo τ_c de captura e τ_e de emissão, relacionados com a sua energia de aprisionamento. A média harmônica entre estes tempos geram τ, que por

sua vez define a frequência de corte (frequência em que a curva de Densidade Espectral de Potência (do inglês *PSD*) do ruído decai de $1/f^2$), conforme mostra as equações:

$$\frac{1}{\tau} = \frac{1}{\tau_c} + \frac{1}{\tau_e} \tag{1}$$

$$f_C = \frac{1}{\tau} \tag{2}$$

Tanto a energia de aprisionamento, a quantidade de armadilhas, quanto os tempos de captura e emissão de portadores são variáveis aleatórias, sendo a sua energia definida em uma distribuição *U-Shape*, a quantidade por uma distribuição de Poisson e os tempos definidos em uma distribuição logarítmica uniforme, todos tidos como parâmetros descorrelacionados (MONTGOMERY, 2003).

A probabilidade de ocupação de uma armadilha é tida como independente do tempo, ou seja, os valores de captura e emissão são constantes, não mudam ao longo do tempo. Assim, as armadilhas que mais contribuem para a ocorrência do RTN são aquelas que se mantém permutando constantemente de estado de ocupação, logo, armadilhas quase sempre ocupadas, ou desocupadas, pouco contribuem para o ruído.

A Figura 3 mostra uma visão qualitativa do ruído aleatório (RTN) medido na corrente de dreno de um MOSFET no domínio tempo e da frequência. Os tempos nos estados de alta e baixa corrente correspondem aos tempos de captura τ_c e emissão τ_e de portadores, respectivamente.

No Capítulo próprio para demonstração do algoritmo utilizado na ferramenta de simulação, objeto deste trabalho, este assunto será retomado, elucidando-se o comportamento matemático dos parâmetros que envolvem a modelagem deste tipo de ruído. A Figura 3a mostra

a curva PSD deste tipo de ruído, que é caraterizada por uma região de potência constante, chamada de platô e uma região de declínio com comportamento próximo a $1/f^2$, sendo esta, uma curva Lorentziana (WIRTH, 2007 e PAVELKA, 2009), que descreve o ruído causado por uma única armadilha. A curva Lorentziana pode ser obtida analiticamente pela equação (HAARTMAN, 2007):

$$S(f) = \frac{1}{f}\frac{A^2}{\left[1+\left(\frac{fc}{f}\right)^2\right]} \tag{3}$$

Onde A é a amplitude (ΔI_D) do pulso da corrente $I_D(t)$, f é a frequência de análise e f_c a frequência de corte definida na equação (2).

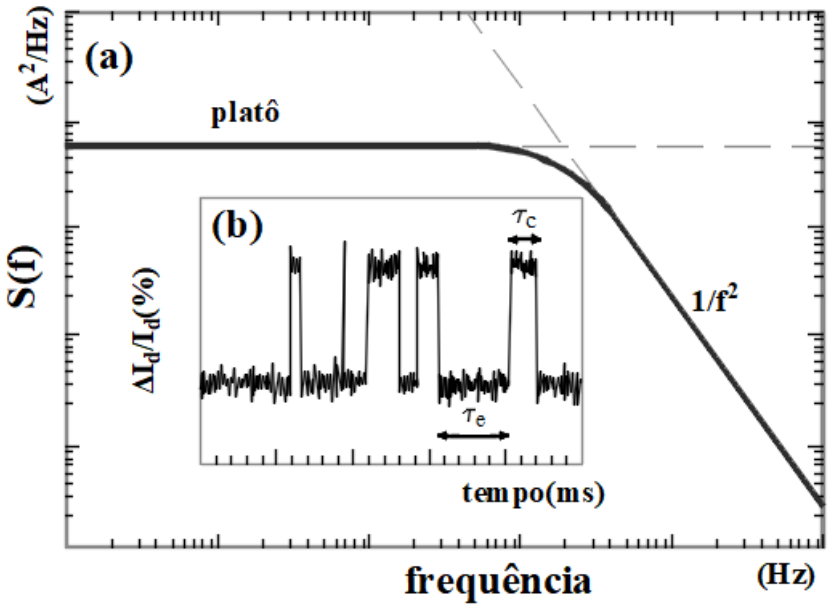

Figura 3- RTN causado por uma única armadilha a) no domínio da frequência e b) no domínio do tempo.

4. TRAPSIMULATOR

TrapSimulator, é o nome dado a ferramenta didática de simulação de ruído RTN em MOSFET de dimensões pequenas (da ordem de poucas dezenas de nanômetros). Seu propósito é contribuir com o aprendizado de alunos de graduação em Engenharia Elétrica e Eletrônica, possibilitando a visualização do comportamento do ruído.

Como uma ferramenta de uso pedagógico, ela se baseia no contexto de ABP, mencionado anteriormente, já que ferramentas de simulação podem tornar um aprendizado mais ativo, uma premissa da metodologia mencionada. Este software de simulação pretende fornecer resultados qualitativos, somente. Neste trabalho, se prezou por construir uma ferramenta de compreensão visual e de uso rápido. Para isto, em alguns parâmetros internos da simulação, os valores adotados foram os típicos para a estrutura de MOSFETs dos nós tecnológicos mais atuais, encontrados na literatura (REISINGER, 2014). Ainda, algumas aproximações foram assumidas para minimizar o tempo computacional.

Sabe-se que o tempo computacional de uma simulação com valores mais apurados e sem muitas aproximações pode chegar à ordem de dezenas de horas, em contrapartida, TrapSimulator consegue fornecer uma visão aproximada de um ruído, contemplando mais o comportamento qualitativo em detrimento da quantificação exata de valores numéricos.

Sua estrutura foi pensada de tal modo que possa ser utilizada em estudos e pesquisas, bem como ferramenta de cognição para uso docente. É uma ferramenta de código aberto, distribuída sob licença *Gnu Generic Public License (GPL)*, desenvolvida e publicada na plataforma voltada para a área da Nano Ciência, Nanohub.org.

4.1. AMBIENTE DE DESENVOLVIMENTO NANOHUB.ORG

NanoHub.org é uma plataforma online totalmente gratuita que tem como motivação facilitar e promover pesquisa e desenvolvimento de conteúdo didático direcionado a nanotecnologia, seja ela da área eletrônica, eletromecânica ou biológica. É, também, um grande repositório de artigos e trabalhos científicos, bem como, uma plataforma de desenvolvimento de ferramentas de análises computacionais, simuladores, que podem ser utilizadas em tempo real e em qualquer ambiente com acesso a um simples navegador de internet (LUNDSTROM e KLIMECK, 2006).

Atualmente, NanoHub.org conta com mais de 400 ferramentas de simulação, mais de 1,4 milhões de usuários ativos sendo que destes, mais de 5600 são pesquisadores.

Esta plataforma, que armazena conteúdo proveniente de mais de 170 países, é considerada um portal da ciência utilizado para pesquisa, educação e colaboração em rede mundial para a nanotecnologia. No momento da escrita deste trabalho, o portal já tinha computado mais de 1,4 milhões de simulações feitas, e ainda cerca de 400 itens classificados como material didático, bem como outros conteúdos, com novos materiais sendo adicionados continuamente (*Nanohub.org Usage*).

Toda essa infraestrutura, mantida por *Network for Computational Nanotechnology (NCN)*, é motivada pelo desenvolvimento de uma sociedade virtual que compartilha software de simulação, dados e outros conteúdos inovadores e que fornecem aos engenheiros e cientistas o conhecimento fundamental necessário para promover a nanociência na nanotecnologia. A plataforma cibernética *NCN* é liderada por Gerhard Klimeck, diretor e professor de engenharia elétrica e informática na *Purdue University*, West Lafayette, Indiana, EUA (LUNDSTROM e KLIMECK, 2006).

Dentre as facilidades que NanoHu.org oferece, uma atenção especial será dada à interface de produção de simuladores computacionais chamada *Rapid Application Infrastructure* (*Rappture*). *Rappture* é um conjunto de ferramentas que torna rápido e fácil o desenvolvimento de aplicações científicas poderosas. O pesquisador pode desenvolver seu simulador paralelamente, podendo ser escrito em *C / C ++*, *Fortran, Octave, R, Java, Perl, Python* ou *Tcl*. Depois de descrever as entradas e saídas para o seu simulador, *Rappture* gera uma interface gráfica automaticamente com base na descrição feita (Nanohub.org Usage). O aplicativo resultante é fácil de implementar no portal Nanohub.org entre outros baseados na plataforma *HUBzero*, de modo que uma grande comunidade de usuários pode acessar o aplicativo através de um navegador da Web convencional.

Rappture foi a plataforma utilizada, então, para servir como parte do desenvolvimento e publicação da ferramenta de simulação, objeto deste trabalho. Além da comodidade na produção do software, Nanohub.org oferece todo aparato necessário, bem como, poder computacional robusto e suficiente, ao ponto de atender a demanda, de maneira gratuita, trazendo ainda, uma maior visibilidade ao grupo de estudo de Micro e Nano Eletrônica deste Programa de Pós-Graduação para a Nanociência mundial.

4.2. ARQUITETURA DO ALGORITMO

A arquitetura utilizada no algoritmo deste trabalho segue o modelo estrutural de ciclo de programação, tendo um único fluxo, sem processamento paralelo ou ciclos alternativos. O algoritmo é não determinístico, o que, na linguagem formal da Ciência da Computação, significa dizer que, mesmo que os parâmetros de entrada submetidos sejam sempre os mesmos, este não gera um mesmo resultado. Isto acontece porque, em dados momentos, são feitas

análises probabilísticas baseadas em números aleatórios gerados pelo software, refletindo na variabilidade que o comportamento simulado oferece.

O código fonte é descrito em linguagem C, cujo paradigma de programação é composto por uma estrutura imperativa procedural, onde cada passo do algoritmo sempre é executado. As responsabilidades funcionais são divididas entre arquivos de classes, conforme a Tabela 1.

No Apêndice A é possível saber mais sobre os parâmetros de entrada e o retorno de cada função da Tabela 1.

Este paradigma de programação especifica os passos que um programa deve seguir para alcançar um estado desejado, iniciando e finalizando sempre nos mesmos pontos. Os Procedimentos, também conhecidos como rotinas, sub-rotinas, métodos ou funções (que não devem ser confundidas com funções matemáticas, mas são similares àquelas usadas na programação funcional) simplesmente contêm um conjunto de passos computacionais a serem executados.

A Figura 4 mostra um diagrama imperativo procedural de classes contendo o fluxo e a dependência entre elas.

Tabela 1-Tabela de funções e responsabilidades de cada classe

Classe	Responsabilidades	Funções
Rappture	Interface gráfica;	Não se aplica.
Main.c	Contempla o fluxo inicial e principal da aplicação. Armazena, em tempo de execução, os parâmetros de entrada da execução do algoritmo. Confere status durante execução do algoritmo para interface gráfica.	*main()*
Trapsimulator.c	Define estruturas de dados (*struct*). Estabelece o comportamento das *traps*. Estabelece o comportamento da corrente. Gera histogramas.	*GetK()* *GetCurrent()* *GetTrapState()* *GetNumTrapped()* *GetHistogramVector()* *GetNoiseAmplitudeByOne()* *GetTrappedEventHist()* *GetInitialTrapState()*
Randomics.c	Gera valores aleatórios baseados nas distribuições de Poisson, Log-Uniforme e Uniforme.	*RndPoisson()* *PdfPoisson()* *Fat()* *CreateTreeProb()* *GetTotalProb()* *AddNode()* *DefineProbIntervals()* *GetRandomLogScale()* *GetRandom()*
Fft.c	Executa a Transformada Discreta de Fourier no vetor de corrente elétrica gerando um vetor de números complexos. Gera o vetor de Densidade Espectral de Potência, aplicando cálculo de amplitude do vetor de números complexos da FFT.	*CreateComplexVectorWn()* *Fft()*
Run*.xml	Repositório de dados gerados pelo algoritmo. Onde * representa um identificador de cada simulação. Persistência de informação (Base de Dados).	Não se aplica
Rappture.h	Coleta os parâmetros de entrada. Expõe os resultados.	*rpLibrary()* *rpGetString()* *rpUtilsProgress()* *rpPutString* *rpResult()*

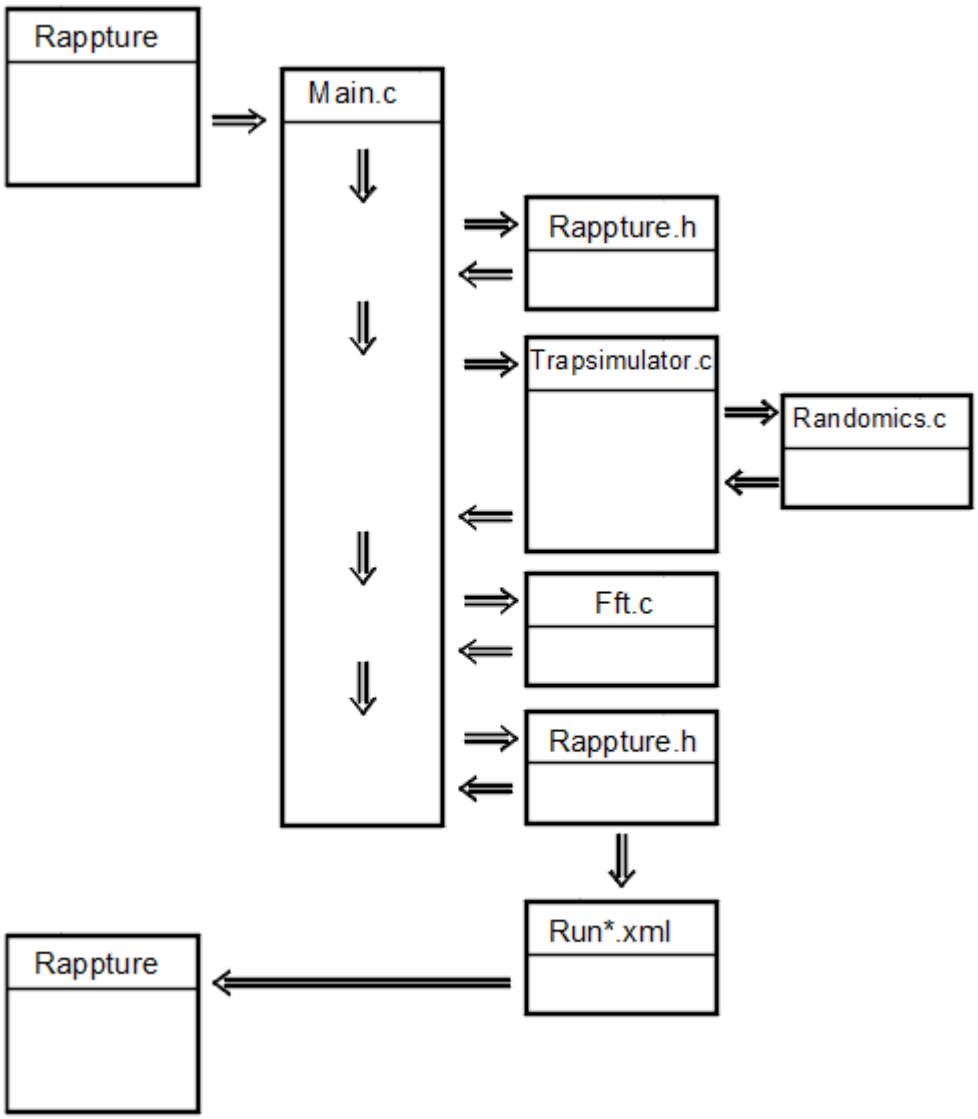

Figura 4 - Diagrama de fluxo imperativo procedural de classes e suas dependências

4.3. ARQUITETURA DO SOFTWARE

TrapSimulator é constituído de uma arquitetura simples, mas eficaz, baseada em três camadas conforme mostra a Figura 5. Esta é uma arquitetura proposta pela plataforma que hospeda o software. A Camada de Interface e a Camada de Dados, bem como as funções de interação entre todas as três camadas, são fornecidas pela plataforma Nanohub.org. O

desenvolvimento deste trabalho refere-se unicamente à Camada *Business*, que contempla a execução do algoritmo de simulação.

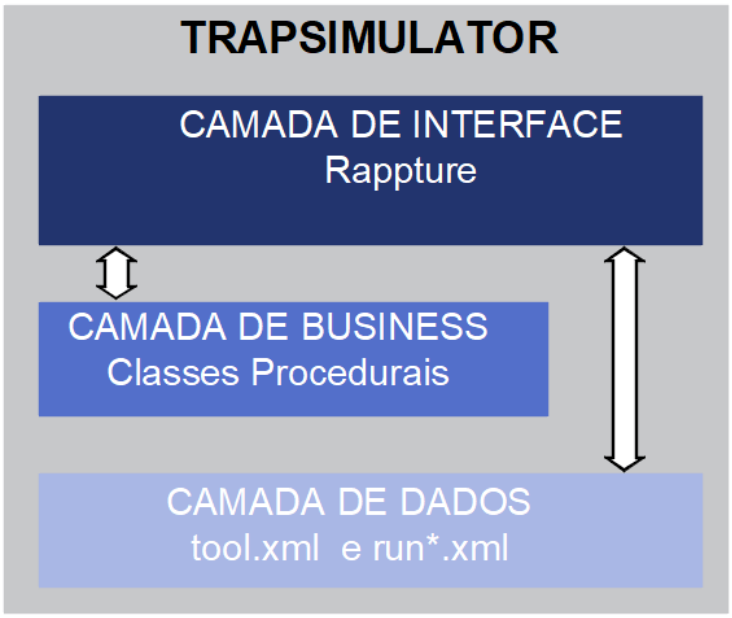

Figura 5 - Diagrama de arquitetura de camadas

4.3.1. CAMADA DE INTERFACE

Neste tipo de arquitetura, a camada gerente da aplicação é a Camada Interface, ficando à sua responsabilidade, a interação entre as demais camadas. Logicamente, a Camada de Interface é a que interage com o usuário final, apresentado telas de seleção de parâmetros para execução da simulação, bem como as telas de resultados. Cada tela de seleção de parâmetros é classificada quanto a sua natureza dentro do fenômeno físico simulado. A Figura 6 mostra a tela de apresentação da ferramenta, contendo uma descrição sucinta do seu propósito e motivação. Além de conter uma indicação sobre o autor e orientador do trabalho.

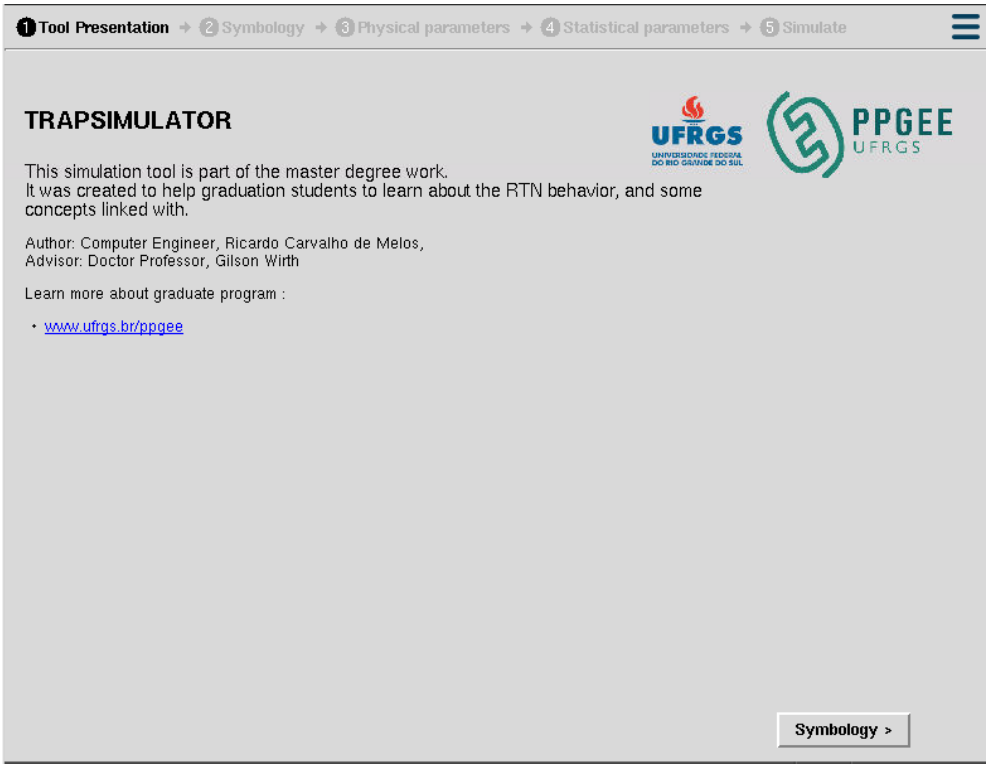

Figura 6 - Tela de apresentação

A Figura 7 mostra a primeira tela de seleção de parâmetros, onde o usuário seleciona a quantidade do vetor de corrente $I_D(t)$ a ser mostrada no gráfico. Vale ressaltar que o vetor de corrente sempre é calculado em sua totalidade, este parâmetro apenas define a porção de amostra do gráfico. Uma amostra menor torna a execução do algoritmo mais rápida sem oferecer de perda de informação, uma vez que a baixa resolução de imagem não torna a análise deste vetor mais apurada, mesmo que este fosse amostrado por completo. Para fins didáticos, esta tela também apresenta a simbologia de um transistor CMOS utilizada em diagramas de projetos eletrônicos.

A Figura 8 contempla a tela de seleção dos parâmetros físicos relacionados ao dispositivo simulado, são eles: Nó Tecnológico; Largura do Canal Ativo; e Material do dispositivo.

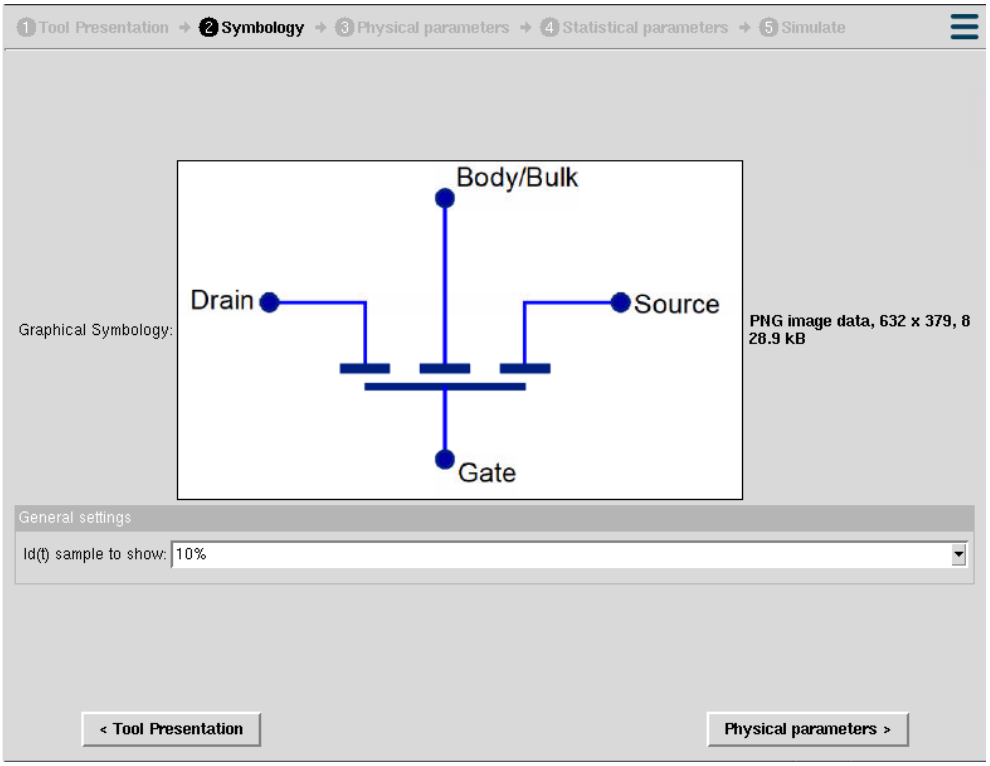

Figura 7 - Tela de seleção de parâmetro "Porcentagem de amostra do vetor $I_D(t)$ no gráfico".

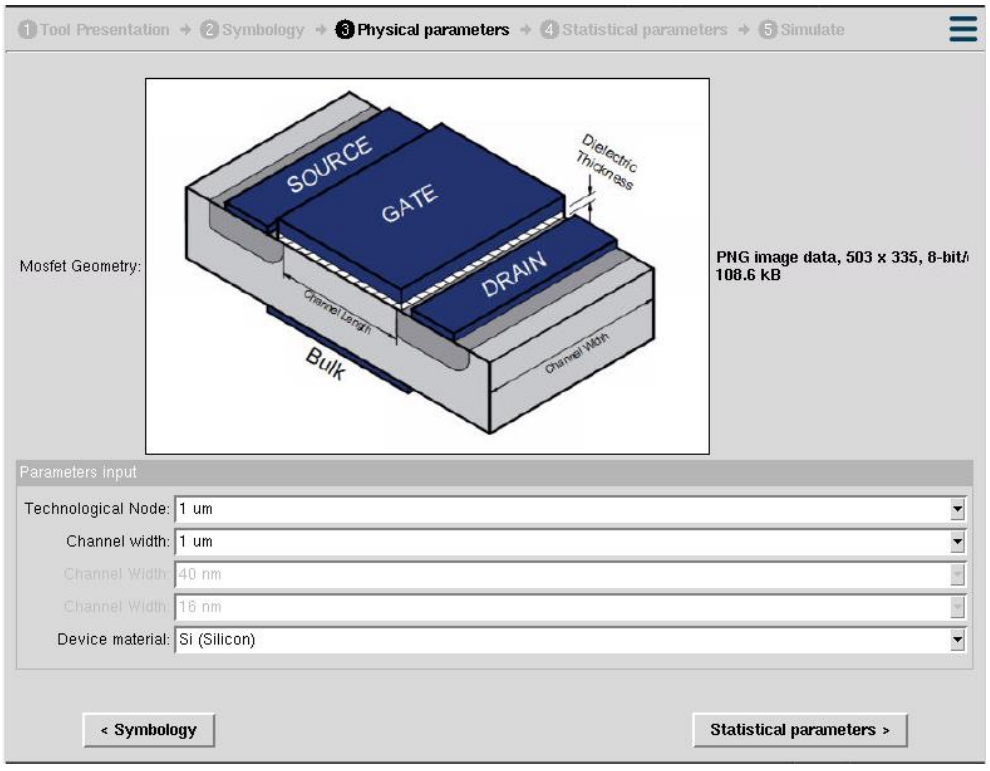

Figura 8 - Tela de seleção de parâmetros relacionados a constituição física do dispositivo

A Largura do Canal ativo é um parâmetro que muda de intervalo de possibilidade conforme o Nó Tecnológico selecionado. Obviamente que a largura (W) mínima não pode ser menor que o comprimento (L) mínimo, que neste trabalho foi predefinido como medida fixa de tamanho dado pelo Nó Tecnológico (para tecnologia 1µm, L=1µm, por exemplo), nem grande o suficiente para tornar-se um valor não realista. Esta tela ainda oferece uma visão estrutural de um dispositivo CMOS genérico para simples reconhecimento de quais pontos deste são afetados pelas alterações nas dimensões.

A Figura 9 apresenta a tela de seleção de parâmetros relacionados ao comportamento estatístico da simulação.

Nela, o usuário opta por uma simulação com a quantidade de armadilhas definidas aleatoriamente em uma distribuição de Poisson com a média também definida por ele, ou, pela quantidade definida diretamente.

O intervalo de escolha da quantidade direta ou média de quantidade de armadilhas também depende do Nó Tecnológico selecionado na tela anterior. Certamente que a quantidade de ocorrência de armadilhas em um dispositivo é proporcional a sua área.

Igualmente, a primeira e a segunda tela (Figura 7 e 8), esta contém uma apresentação didática, neste caso, do evento descrito pelo conceito *Trapping De-Trapping* de captura e emissão de portadores por armadilhas.

Todos os parâmetros iniciais de simulação e os valores padrões, que o usuário pode escolher para a execução de uma simulação, podem ser observados na Tabela 2.

Tabela 2 - Parâmetros de entrada da Simulação

Nome	Valores possíveis	Valores Padrão
Nó Tecnológico	1µm, 0.1µm, 40nm and 16nm	1µm
Porcentagem de amostra de ID No gráfico	10%, 25% ou 50%	10%
Usa Poisson	Sim ou Não	Sim
τ_C e τ_E iguais	Sim ou Não	Sim
Média de Armadilhas	De 1 até 1000	---
Largura do canal ativo	De 16nm até 10µm	---
Material do dispositivo	Si, Ge ou GaAs	Si

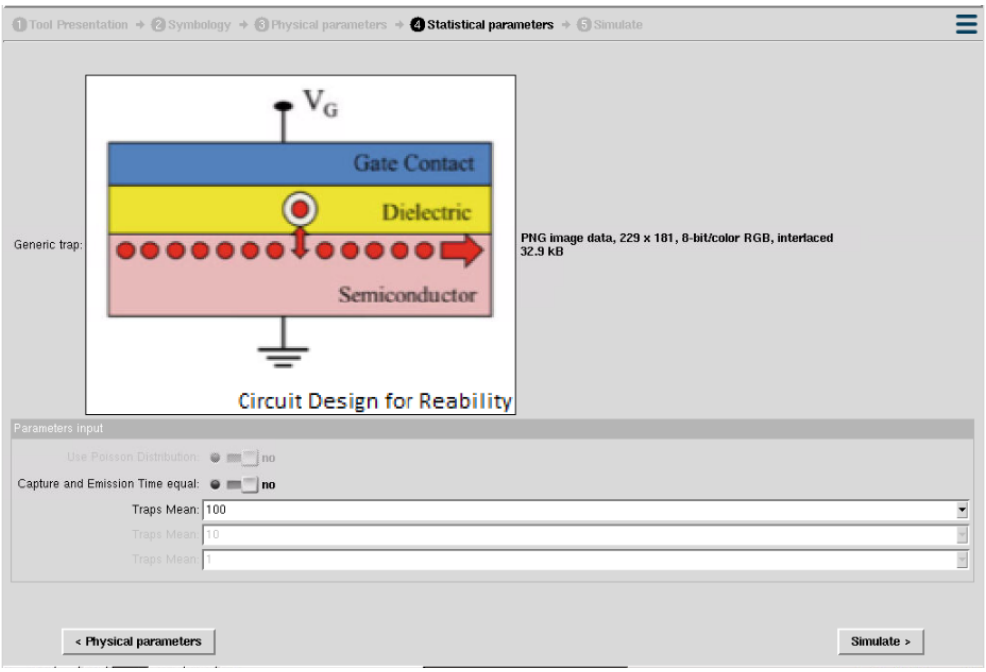

Figura 9 - Tela de seleção de parâmetros estatístico da simulação

A Figura 10 apresenta a tela de resultados da simulação contendo uma parte do vetor de corrente elétrica no domínio do tempo, a Figura 11, no domínio da frequência.

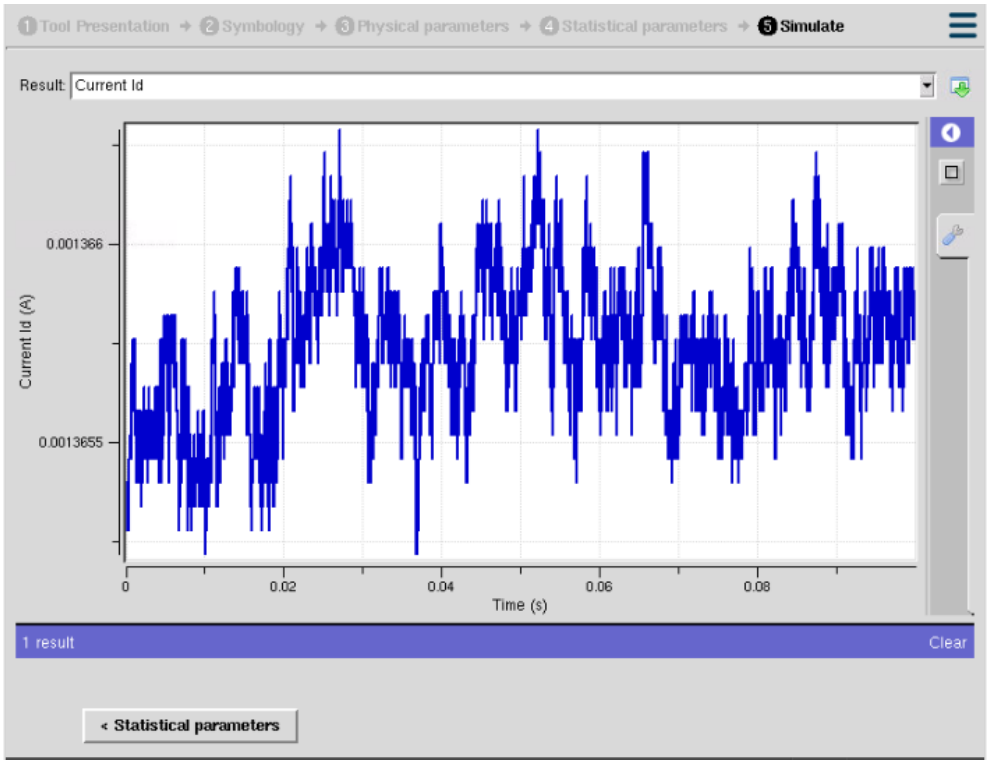

Figura 10- Tela de resultados. Corrente no domínio do tempo

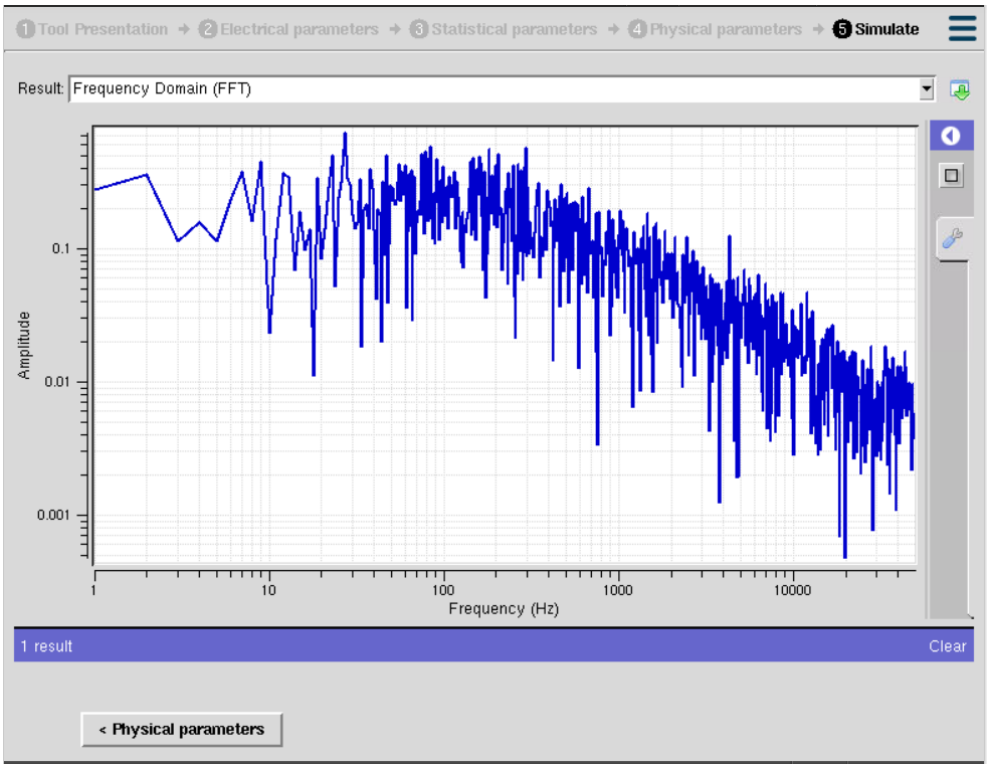

Figura 11 - Tela de resultados. Densidade Espectral de Potência

A Figura 12 mostra um histograma de quantidade de armadilhas em estado de captura a cada interação de cálculo de corrente elétrica, e a Figura 13 mostra um histograma do tempo de duração dos eventos de captura, analisando todos eventos de captura de todas as armadilhas presentes na simulação.

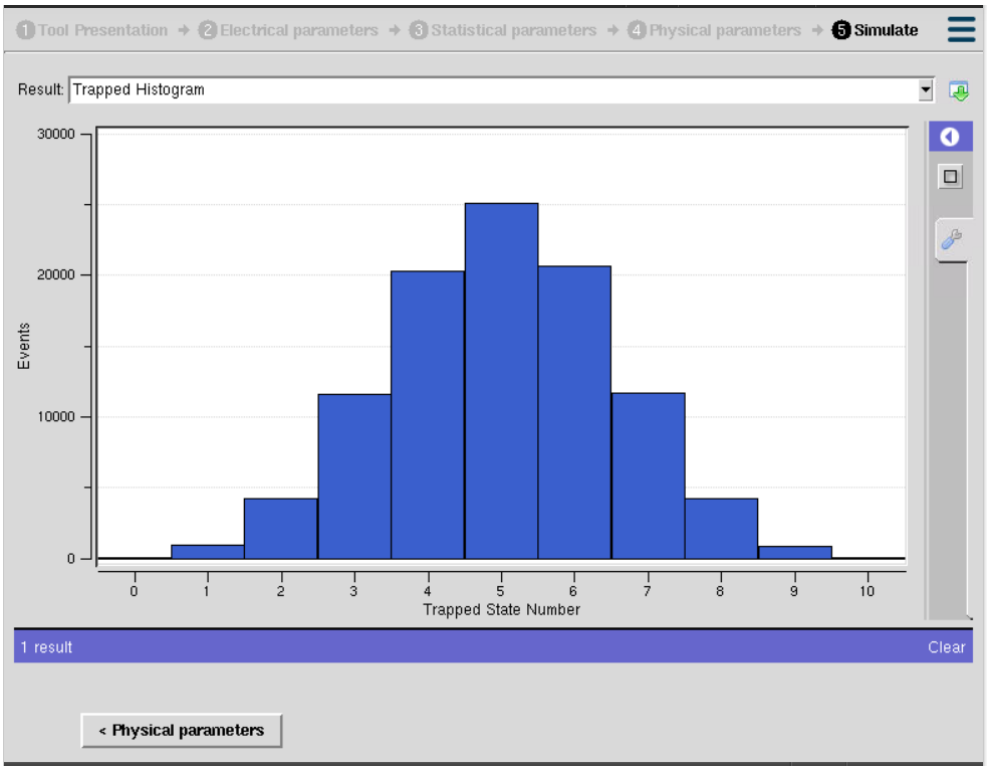

Figura 12 - Tela de resultados. Histograma de quantidade de armadilhas em estado de captura

Finalmente, o último componente da camada de interface é a tela de Relatório Textual. Esta tela oferece uma referência escrita dos parâmetros gerados internamente pelo software, tais como, tempo de captura e emissão da cada armadilha, quantidade estimada de armadilhas, caso o usuário opte por uma distribuição de Poisson. A Figura 14 mostra o formato desta tela, que igualmente a todas as telas que demonstram gráficos da simulação, pode ser carregada pelo usuário para que possa fazer uso em outras situações.

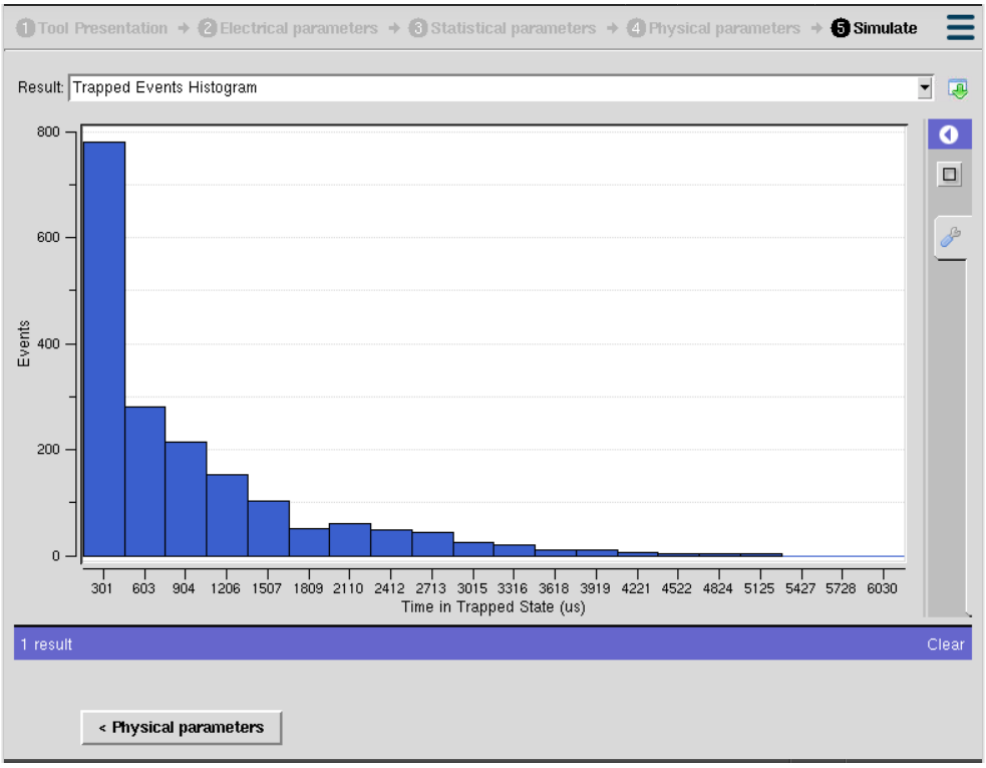

Figura 13 - Tela de resultados. Histograma de tempo de eventos de captura

Na camada de interface, a plataforma Rappture oferece ferramentas nativas para análise de múltiplas simulações. Isto quer dizer que, ao executar várias simulações, cada interface de saída, permite que sejam comparados os seus gráficos, ou separadamente, ou agrupados em uma mesma tela, facilitando assim a compreensão do estudante sobre as consequências de suas alterações nos parâmetros de simulação. A Figura 15 mostra três gráficos agrupados de diferentes vetores de Densidade Espectral de Potência.

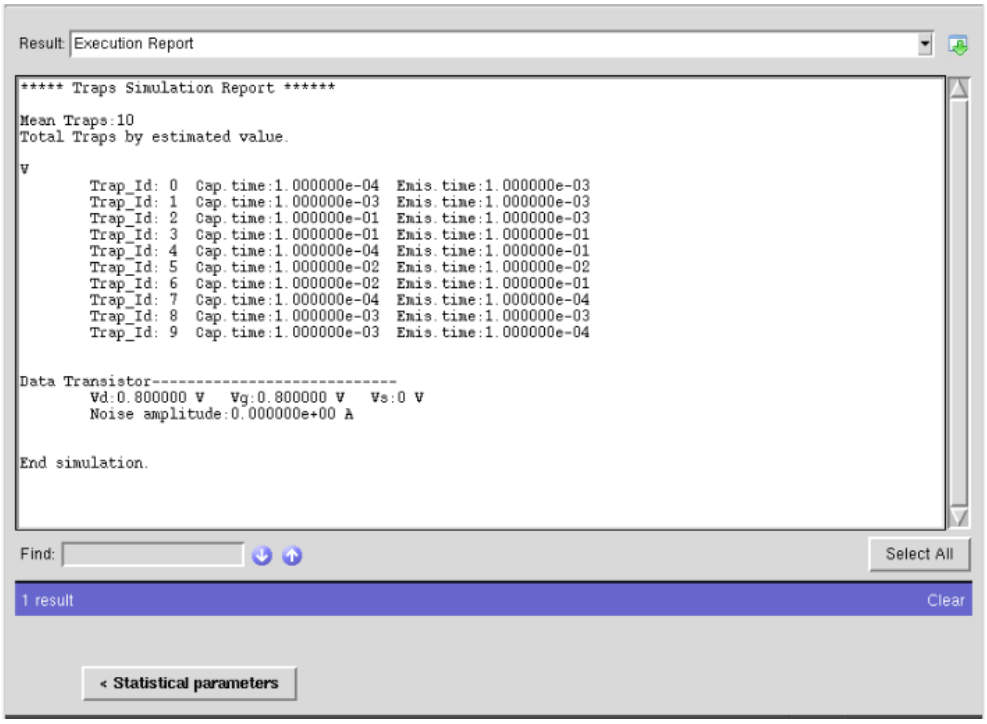

Figura 14- Tela de resultados. Relatório Textual

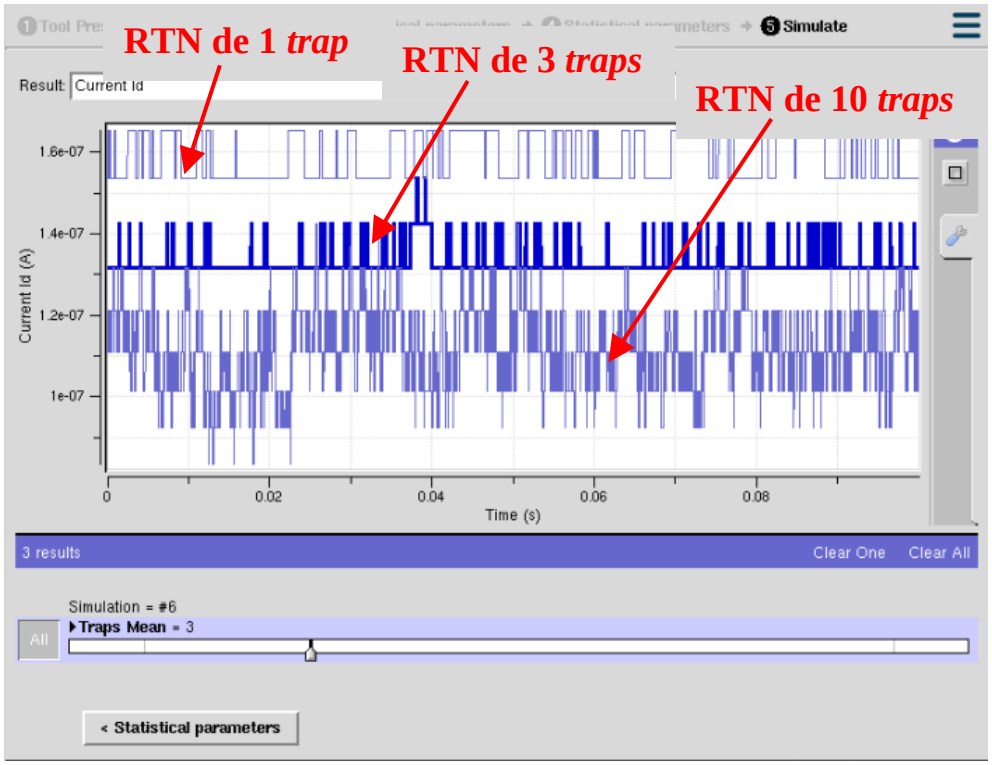

Figura 15 - Tela para análise de múltiplas simulações

4.3.2. CAMADA BUSINESS

Tipicamente, camadas de regras de negócio, denominadas Camada Business, executam as particularidades de cada aplicação. Elas contêm todos os modelos de cálculos, validações e verificações peculiares de um sistema de propósito especifico. Neste trabalho, a Camada Business é quem tem a responsabilidade de execução da simulação de corrente elétrica que percorre um dispositivo CMOS sob efeito do ruído RTN. Como dito anteriormente, este é um algoritmo imperativo estruturado divido em duas grandes rotinas, como mostra a Figura 16.

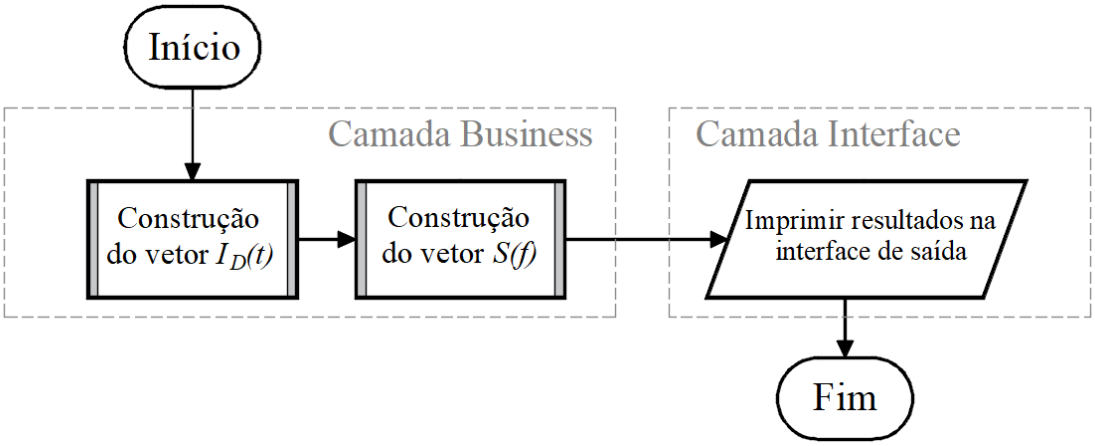

Figura 16 - Diagrama de blocos do algoritmo principal

4.3.2.1. ROTINA "CONSTRUÇÃO DO VETOR $I_D(t)$"

Nesta sub-rotina, utilizou-se para calcular a corrente de dreno, quando o dispositivo estiver operando na região de linear, a equação (SEDRA e SMITH, 1998):

$$I_D(t) = \frac{k'W}{L}\left[(V_{GS} - V_{TH})V_{DS} - \frac{V_{DS}^{\ 2}}{2}\right] \quad (4)$$

Caso o dispositivo estiver operando na região de saturação, a equação que modela a corrente de dreno é a seguinte (SEDRA e SMITH, 1998):

$$I_D(t) = \frac{k'W}{2L}(V_{GS} - V_{TH})^2 \tag{5}$$

Onde *W* e *L* são a largura e o comprimento do canal ativo do MOSFET, respectivamente. V_{GS}, V_{DS} e V_{TH} são a tensão de porta, tensão de dreno, ambas em relação a tensão de fonte, e tensão de limiar, respectivamente.

Onde *k'*, chamado de parâmetro de transcondutância, é dado por (6).

$$k' = \frac{\mu_n \varepsilon_{ox}}{t_{ox}} \tag{6}$$

Sendo μ_n a mobilidade dos portadores, t_{ox} a espessura do dielétrico e ε_{ox} a permissividade do material do dielétrico.

As equações (4) e (5) são modelagens simples que relacionam a corrente de dreno com as tensões dos terminais. Entre uma das suas limitações, está o fato de que a equação (5) não considera o fator de modulação de canal, da região de saturação. Ainda assim considera-se, para o uso a que se propõe este trabalho, aceitáveis tais aproximações feitas pelo autor.

Entretanto, as equações de modelagem da corrente de dreno (4) e (5), descritas anteriormente, não levam em consideração o efeito do RTN. Neste trabalho, se considerou apenas a alteração na tensão de limiar como causa da flutuação do nível de corrente de dreno. Assim, o efeito do RTN é incluído na simulação, fazendo-se em cada interação do cálculo de corrente, um incremento em V_{TH}, conforme (7).

$$V_{TH} = V_{TH0} + \Delta V_{TH} \tag{7}$$

Onde V_{TH0} é a tensão de limiar inicial e ΔV_{TH} é a variação de tensão de limiar, consequência do efeito *Trapping de-Trapping*, que é dada por (8) (FURTADO, 2017)

$$\Delta V_{TH} = \sum_{i=1}^{N_{tr}} \delta V_{THi} X_i(t) \tag{8}$$

Assim, δV_{TH} é o impacto individual de cada armadilha sobre a tensão de limiar e $X(t)$ representa o estado de cada armadilha no instante da interação, podendo assumir valor 1, para estado de ocupada e 0 para desocupada. A Tabela 3 relaciona o impacto individual de cada armadilha com o Nó Tecnológico do dispositivo (REISINGER, 2014).

Tabela 3- Impacto individual de cada trap

Nó Tecnológico	δV_{TH} **(mV)**
1 um	0,05
0,1 um	1,57
40 nm	5,52
16nm	8,83

Cada armadilha tem seu estado inicial definido pela análise da probabilidade de estado ocupado $p_o(t)$ e um número aleatório *r* entre 0 e 1, gerado no instante da análise. Sendo,

$$X_i(t) = 1 \rightarrow p_0(t) >= r$$
$$X_i(t) = 0 \rightarrow p_0(t) < r$$

Onde $p_o(t)$ é dado por (9).

$$p_0(t) = \frac{1-\tau_c}{\tau_c + \tau_e} \tag{9}$$

Em todas as interações do cálculo da corrente, as análises dos estados de cada armadilha são refeitas, comparando a probabilidade de troca de estado $p(t)$ com um número aleatório r entre 0 e 1 gerado no instante da análise. Similar a definição do estado inicial:

$$Troca \rightarrow p(t) >= r$$
$$Permanece \rightarrow p(t) < r$$

Onde $p(t)$ é dado por (10), se o estado for ocupado, e por (11), se o estado for desocupado.

$$p(t) = \frac{dt}{\tau_e} \tag{10}$$

$$p(t) = \frac{dt}{\tau_c} \tag{11}$$

Considera-se dt o passo de análise com o intervalo mínimo de duração, neste trabalho, definido em $1x10^{-5}$ segundos.

Assim como mostrado na TABELA 3 para o impacto de cada armadilha em V_{TH}, a sua quantidade também é diretamente proporcional ao comprimento L do canal ativo, sendo que os transistores com L até 40nm, possuem, aproximadamente, uma ou duas unidades de armadilhas.

Já dispositivos com L≈40nm, possuem, aproximadamente uma dezena delas. Por fim, transistores de L≈1µm chegando a ordem de $1x10^3$ ocorrências de armadilhas (REISINGER, 2014).

A Figura 17 mostra o diagrama de blocos do pseudo-algoritmo referente a rotina de construção de $I_D(t)$. O laço *j* concentra as interações de análise dos estados das armadilhas e o laço *i*, as interações de cálculo de corrente elétrica do dispositivo.

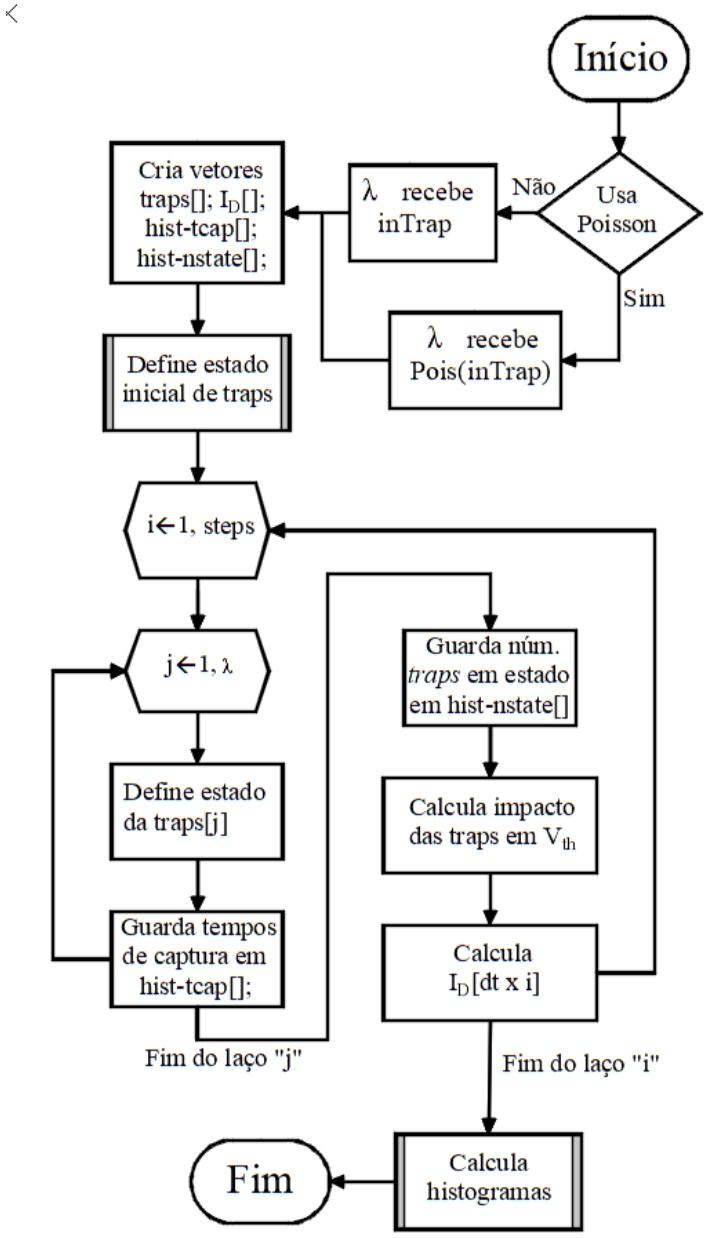

Figura 17- Diagrama de blocos da rotina de criação de $I_D(t)$

4.3.2.2. ROTINA "CONSTRUÇÃO DO VETOR *S(f)*"

O vetor *S(f)* corresponde a Densidade Espectral de Potência do ruído RTN. Este é gerado aplicando-se a Transformada Discreta de Fourier. Entretanto, neste trabalho foi implementado a Transformada Rápida de Fourier (12) e (13) ao vetor $I_D(t)$, o que reduziu a complexidade desta parte do algoritmo de n^3 para $n.log(n)$, onde n foi definido em $1x10^5$.

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x(2r) W_{\frac{N}{2}}^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} x(2r+1) W_{\frac{N}{2}}^{kr} \tag{12}$$

$$X(k) = \frac{1}{N} \sum_{k=0}^{N-1} x\left(\frac{2\pi}{N} k\right) W_N^{kn} \tag{13}$$

$$W_N^{kr} = e^{\frac{-j2\pi kr}{N}} \tag{14}$$

Assim, *W* é uma matriz de elementos rotacionais, *x* representa o vetor de corrente no domínio do tempo contendo *N* amostras, e *X* é o vetor de números complexos. Calculando-se a amplitude de cada número complexo contido em *X*, se obtém o vetor *S(f)* da Densidade Espectral de Potência.

4.3.2.3. NÚMEROS ALEATÓRIOS

Um dos fundamentos principais do RTN é seu comportamento aleatório. Esta natureza, que vem a ser evidenciada no próprio nome do ruído, gera variabilidade nos dispositivos CMOS em níveis que limitam os seus pontos de operação.

Cada variável aleatória envolvida na geração do ruído, tem sua função de densidade de probabilidade, entretanto, serão abordadas neste trabalho, somente as distribuições uniforme, logarítmica uniforme e de Poisson, que fazem parte do algoritmo de simulação.

4.3.2.3.1. DISTRIBUIÇÃO UNIFORME

Números aleatórios sob uma distribuição uniforme compõem o mais baixo nível de aleatoriedade produzidos por funções computacionais que geram números aleatórios sob uma dada distribuição. Seja *p(x, a, b)* a probabilidade de ocorrência de um número aleatório em uma distribuição uniforme, então:

$$p(x,a,b)=\begin{cases}\dfrac{1}{b-a} \rightarrow a \le x \le b \\ 0 \rightarrow x < a \\ 0 \rightarrow x > b\end{cases} \tag{15}$$

Como prova de conceito, a Figura 18 mostra um histograma contendo $1x10^6$ números aleatórios, contidos no intervalo entre 0 e 1, gerados em uma distribuição uniforme pelo algoritmo deste trabalho.

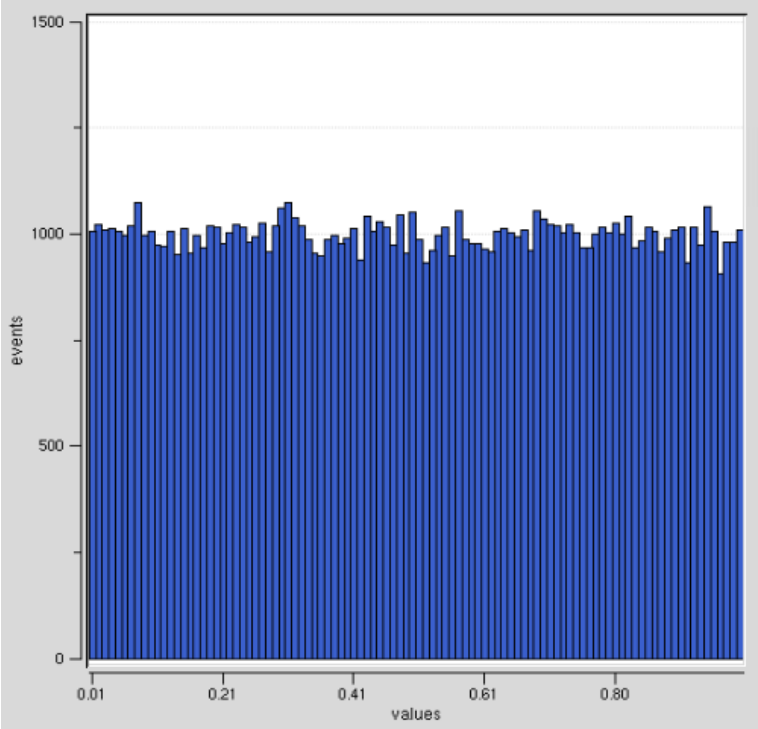

Figura 18 - Histograma de 1x10^6 números aleatórios, contidos no intervalo entre 0 e 1, para uma distribuição uniforme.

4.3.2.3.2. DISTRIBUIÇÃO LOGARÍTMICA UNIFORME

A composição de uma distribuição logarítmica uniforme de base decimal é originária de números elevados às potências, cujos valores contemplam números aleatórios de uma distribuição uniforme. Seja $p(10^x, 10^a, 10^b)$ a probabilidade de ocorrência de um número aleatório em uma distribuição logarítmica uniforme, então:

$$p\left(10^x,10^a,10^b\right)=\begin{cases}\dfrac{1}{b-a} \rightarrow 10^a \le 10^x \le 10^b \\ 0 \rightarrow 10^x < 10^a \\ 0 \rightarrow 10^x > 10^b\end{cases} \tag{16}$$

Como prova de conceito, a Figura 19 mostra um histograma contendo $1x10^6$ números aleatórios, contidos no intervalo entre 1 e $1x10^{13}$, gerados em uma distribuição logarítmica uniforme pelo algoritmo deste trabalho. O valor $1x10^{13}$ foi escolhido por ser o maior valor aceito pelo armazenamento computacional das variáveis do algoritmo.

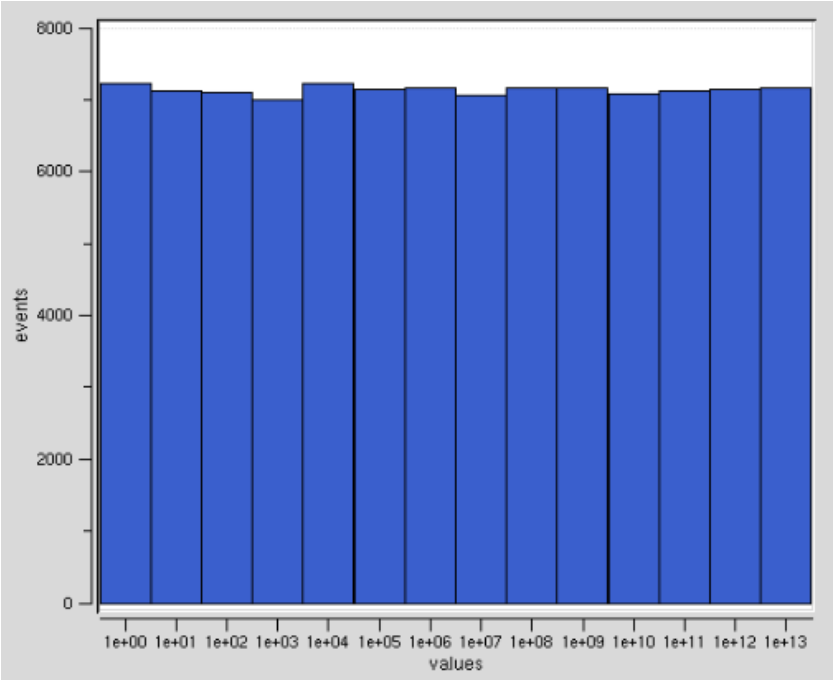

Figura 19 - Histograma de $1x10^6$ números aleatórios, contidos entre 1 e $1x10^{13}$, para uma distribuição logarítmica uniforme

4.3.2.3.3. DISTRIBUIÇÃO DE POISSON

A composição de uma distribuição de Poisson é formada pela probabilidade de ocorrências de eventos dado que já se conhece um valor estimado de ocorrências em um certo período de tempo. Seja $p(x, \lambda)$ a probabilidade de ocorrência de x eventos em um dado intervalo de tempo onde se estima que ocorra λ eventos, então:

$$p(x,\lambda) = \begin{cases} \dfrac{e^{-\lambda}\lambda^{x}}{x!} \end{cases} \tag{17}$$

Como prova de conceito, a Figura 20 mostra um histograma contendo $1x10^6$ números aleatórios, gerados em uma distribuição de Poisson de média igual a 10, pelo algoritmo deste trabalho em um intervalo de confiança de aproximadamente 99%.

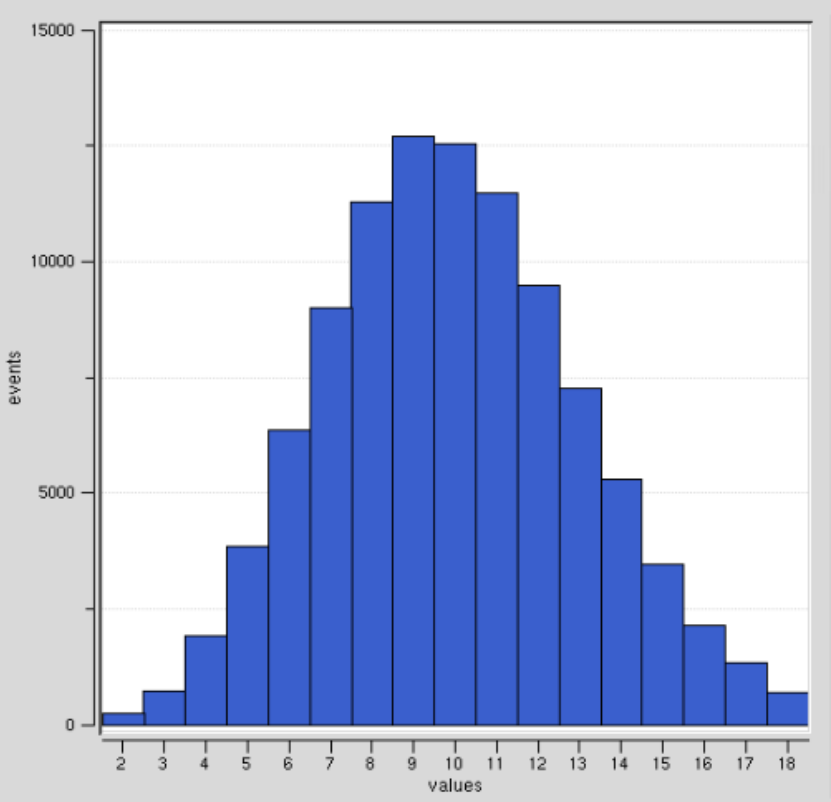

Figura 20 - Histograma de $1x10^6$ números aleatórios de uma distribuição de Poisson com média λ=10.

A escolha dos valores para comprovação dos conceitos de geração de números aleatórios desta distribuição também foi limitada pela máxima capacidade computacional de armazenamento de valores numéricos nas variáveis utilizadas.

4.3.3. CAMADA DE DADOS

Camadas de Dados são responsáveis por persistirem dados de uma aplicação. TrapSimulator necessita armazenar dois tipos de informação para garantir o sucesso nas simulações.

Inicialmente, a aplicação recupera os dados armazenados para construir, em tempo de execução, toda a interface gráfica. Cada componente contido no software é previamente definido, utilizando-se a plataforma Rappture para construção dessa interface, e posteriormente armazenado em um arquivo do tipo *Extensible Markup Language (XML)* conforme mostra a Figura 21.

```
<boolean id="equalTimes">
    <about>
        <label>Capture and Emission Time equal</label>
        <description>This checkbox indicates whether the algorithm
                should set the capture and emission equal.</description>
    </about>
    <default>no</default>
</boolean>
```

Figura 21 - Exemplo de armazenamento de componente de seleção booleana

Finalmente, após a execução bem-sucedida de uma simulação, todos os dados que irão compor os gráficos e histogramas são armazenados em outro arquivo, também do tipo *XML*. Isto se faz necessário para garantir que a aplicação possa retomar as simulações anteriores, bem como, otimizar comparações entre elas. A Figura 22 mostra um exemplo de dados armazenados após uma simulação.

```
<histogram id="Traps_histogram">
    <about>
        <label>Trapped Histogram</label>
        <description>Traps in trapped state histogram</description>
    </about>
    <xaxis>
        <label>Trapped State Number</label>
        <description>Trapped Number</description>
    </xaxis>
    <yaxis>
        <label>Events</label>
        <description>Events</description>
    </yaxis>
    <component>
        <xhw>
            0 927 1
            1 9316 1
            2 42004 1
            3 113846 1
            4 202215 1
            5 246823 1
            6 208431 1
            7 120046 1
            8 45609 1
            9 9804 1
            10 979 1
        </xhw>
    </component>
</histogram>
```

Figura 22 - Exemplo de dados de histograma armazenados

4.4. USABILIDADE DO SOFTWARE

O critério de usabilidade é de extrema importância para o desenvolvimento de sistemas, sejam eles complexos ou não. A experiência de usabilidade do usuário produz a aceitação do software por parte do cliente, neste caso a comunidade acadêmica. Não basta que o software em questão, produza os resultados esperados, conforme a modelagem matemática

determina, é preciso que este tenha uma interface amigável, e que seu uso seja simples e intuitivo. Aplicações que demandam um maior esforço de aprendizagem para seu uso, ou até mesmo, um alto nível de conhecimento de informática geral para sua instalação, tendem a obter uma menor aceitação por parte de seus usuários.

Atualmente, a quantidade de softwares disponíveis para os mais variados propósitos é inumerável, de forma que, a aceitação e usabilidade de cada um, o destaca em relação aos demais.

Priorizando a aceitação e usabilidade, este trabalho focou na visão qualitativa do ruído, utilizando em alguns de seus parâmetros, valores otimistas, não considerando, nem o pior caso, nem o melhor. Por exemplo, sabe-se que cada armadilha pode impactar diferentemente na geração do ruído, entretanto, considerou-se que todas as armadilhas presentes na simulação afetam da mesma forma.

A visão qualitativa, torna a simulação rápida o suficiente, ao ponto de possibilitar a execução de quantas simulações forem necessárias, para que o entendimento sobre o RTN possa ser adquirido. Em contrapartida, esta visão não permite atribuir conclusões acerca dos valores de densidade espectral de potência do RTN, por exemplo, já que não são valores tão apurados.

Ainda, a visão qualitativa, oferece uma abordagem conceitual que envolve a relação da frequência de corte e os tempos de captura e emissão de cada armadilha, bem como, o decaimento da densidade espectral de potência do ruído de $1/f^2$ a partir da frequência de corte estabelecida. A Figura 23 mostra uma curva de densidade espectral de potência gerada por um ruído contendo uma única armadilha, com tempo de captura e emissão de $1x10^{-3}$ segundos. A curva aproximada da média é equivalente a uma curva *Lorentziana*.

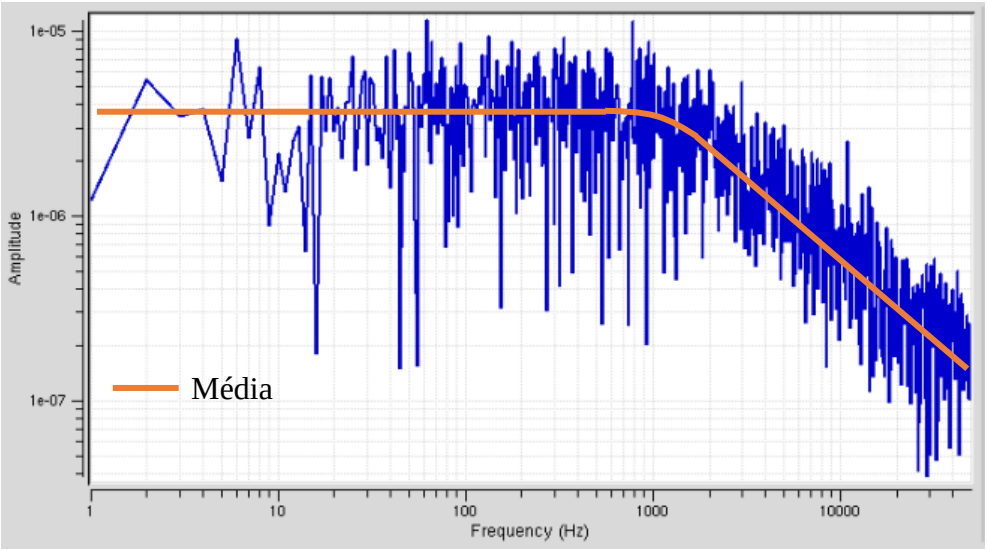

Figura 23- Curva *Lorentziana* gerada por armadilha com tempo de captura e emissão de $1x10^{-3}$ segundos

Finalmente, é possível observar que a sobreposição de várias curvas *lorentzianas* (curva de densidade espectral de potência de RTN causado por uma única armadilha) gera uma curva com decaimento *1/f*, o chamado Ruído de Baixa Frequência. A Figura 24 mostra uma simulação que imprimiu uma curva no domínio da frequência gerada por 20 armadilhas, com tempos de captura e emissão variando entre $1x10^{-4}$ e $1x10^{-1}$ segundos. A sobreposição das curvas que cada uma das armadilhas geraria, se estivesse atuando isoladamente, cria uma curva com comportamento aproximado a *1/f* como mostra a curva de média. Esta sobreposição de curvas é melhor explicada na Figura 25.

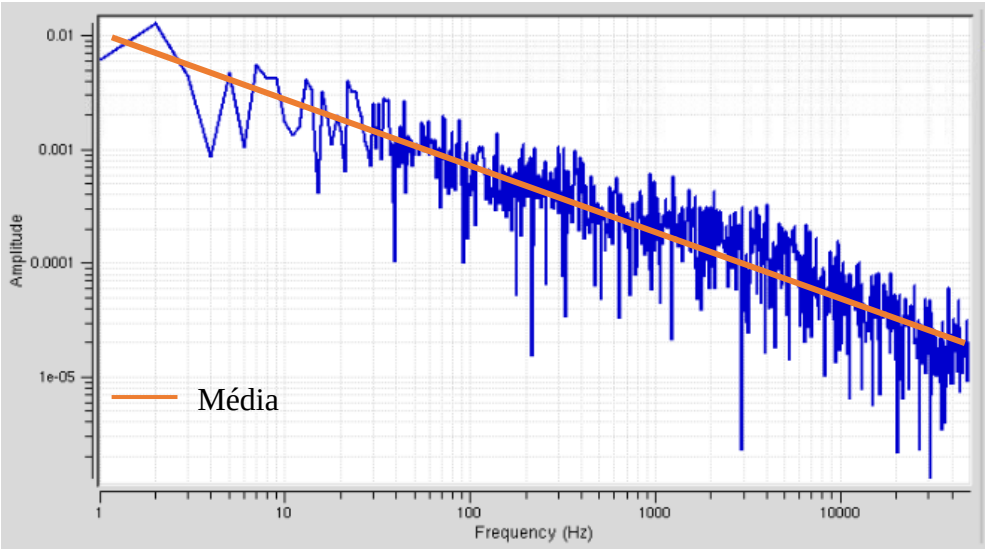

Figura 24 - Curva *1/f* gerada por 20 armadilhas de tempos de captura e emissão variados

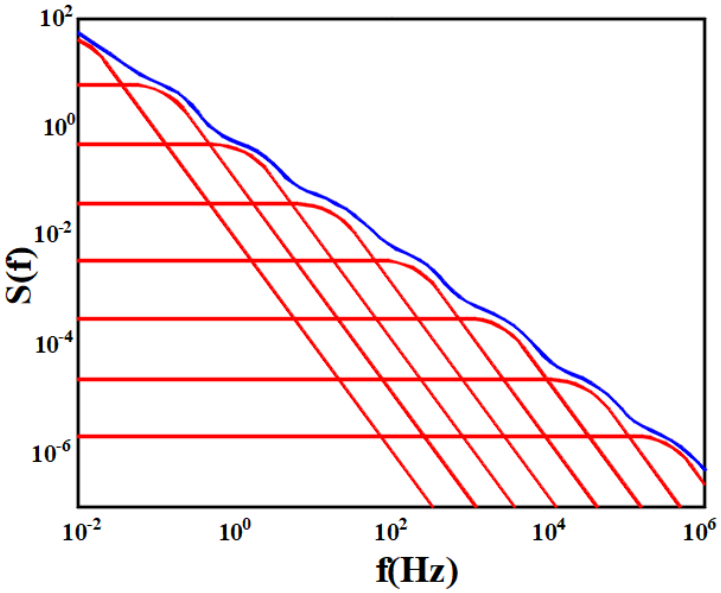

Figura 25- Sobreposição de *Lorentzianas*

Certamente, que a persistência de informação gerada também foi observada quando se definiu a arquitetura deste projeto, como forma de melhorar a experiência de usuário. Ambas,

imagem dos gráficos e arquivo de dados contendo os vetores calculados, são disponibilizados a cada simulação executada, para serem carregados e utilizados da maneira que o usuário desejar. As imagens são convertidas para extensões do tipo *ps*, *pdf*, *png* e *jpeg* ou para arquivos de texto separados por ponto e vírgula.

Quando se aborda análise de ocorrência de eventos, sendo estes independentes ou não, histogramas tem uma visão melhor de como e quando ocorrem tais eventos. Pensando nisso, este trabalho apresenta dois histogramas, o primeiro contendo a quantidade de armadilhas em estado de captura em cada instante de cálculo de corrente elétrica. Seja *p(x)* a probabilidade uniforme de um único evento de captura acontecer, a sobreposição da probabilidade de *n* eventos acontecer simultaneamente se aproxima de uma PDF normal, à medida que *n* se torna muito grande. Isto pode ser observado na Figura 26, onde é apresentado um histograma de armadilhas em estado de captura para um ruído composto por 20 armadilhas.

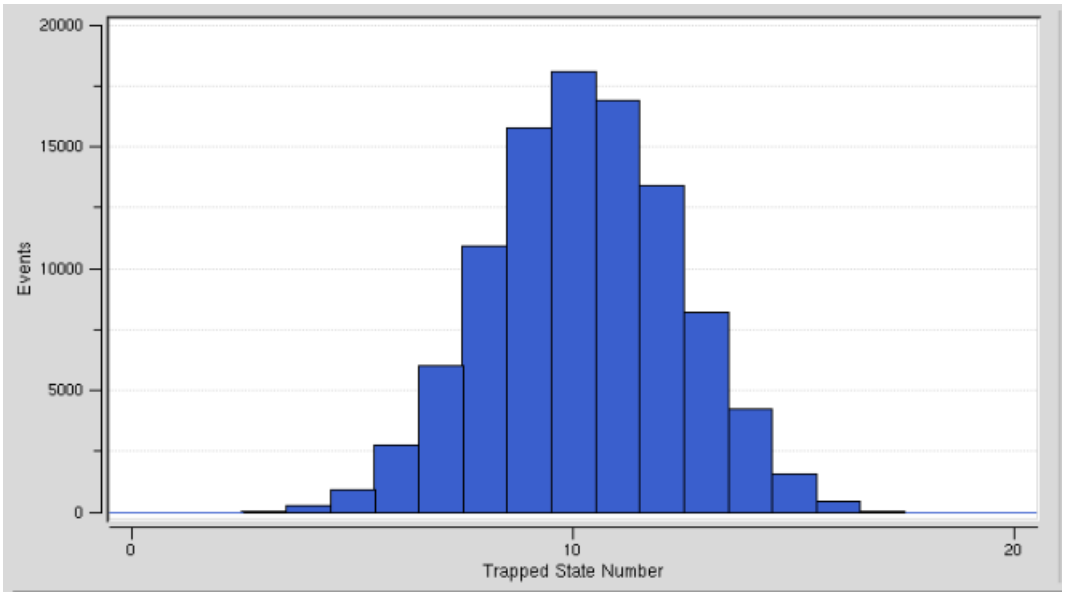

Figura 26 - Histograma de armadilhas em estado de captura de 20 armadilhas

O segundo histograma apresenta o tempo de duração dos eventos de captura, considerando o tempo que cada evento ocupou no estado. Com este tipo de gráfico, pode-se observar que a maior quantidade de eventos são aqueles de duram menos tempo, e que a maior parte dos tempos de captura estão contidos no intervalo proposto pela distribuição logarítmica uniforme entre $1x10^{-1}$ e $1x10^{-4}$ segundos, gerada automaticamente pelo algoritmo.

A Figura 27 mostra um histograma de tempo em estado de captura para um RTN composto 20 armadilhas. Identicamente, um histograma de tempo em estado de emissão teria essa mesma distribuição exponencial.

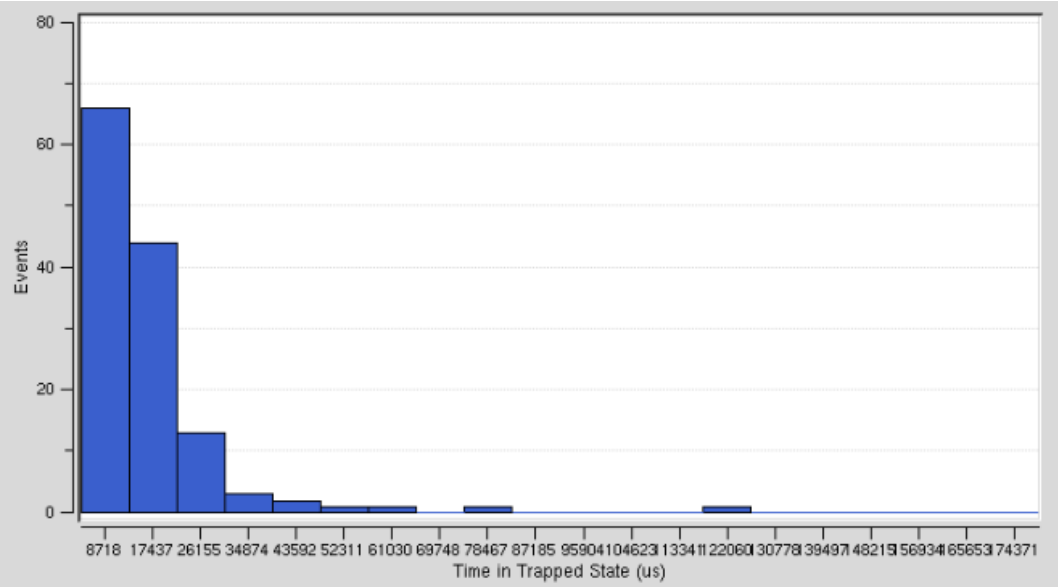

Figura 27 - Histograma de tempos em estado de captura para RTN contendo 20 armadilhas

5. CONCLUSÃO

De fato, espera-se uma maior eficiência no processo cognitivo de estudantes da área em questão utilizando-se de metodologias que seguem o conceito da ABP.

As ferramentas de simulação são grandes aliadas da metodologia de Aprendizagem Baseada em Projetos, pois com elas é possível colocar em prática conteúdos teóricos, como, por exemplo, as causas e efeitos do RTN.

O software, resultado deste trabalho, atingiu seu requisito principal e de maneira satisfatória, à medida que a proposta da ferramenta de simulação era proporcionar uma visão qualitativa e rápida do comportamento de RTN, permitindo a alteração de parâmetros de entrada, obtendo um resultado em tempo bem pequeno. TrapSimulator conseguiu produzir simulações consumindo, em média, alguns segundos.

A interface de construção de ferramentas de simulação fornecida por Nanohub.org trouxe a possibilidade de desenvolvimento de uma ferramenta de auxílio didático para uso em infraestruturas simples, como um computador contendo um navegador de internet.

Considerou-se que a motivação de produzir uma ferramenta de auxílio à comunidade acadêmica, foi um objetivo alcançado, já que, desde a sua implantação em setembro de 2017, TrapSimulator já foi utilizado em 7 países diferentes, executando mais de 400 simulações (Figura 28) (TrapSimulator, 2017).

Como trabalhos futuros, surge a necessidade de se melhorar algoritmo, a fim de fazê-lo ainda mais eficiente. Por exemplo, para tornar a curva de Densidade Espectral de Potência menos ruidosa, sugere-se a aplicação do Método de Welch, que utiliza de estimativas de espectro baseada em periodogramas.

Sugere-se, a criação de um segundo algoritmo que abrange todos os aspectos mais realistas da simulação, criando um vetor de corrente elétrica com valores mais apurados, para

atender à usuários de mais alto nível, possibilitando que este opte por uma simulação rápida e qualitativa ou por outra mais demorada, porém quantitativamente mais apurada.

Finalmente, sugere-se um terceiro algoritmo, que deve produzir um vetor de corrente e seu respectivo vetor de Densidade Espectral de Potência para cada armadilha individualmente, e agregando-os somente nos gráficos de saída. Este algoritmo será igualmente fruto da escolha do usuário, já que este também terá um custo computacional maior. A sobreposição das múltiplas curvas de Corrente Elétrica de Dreno e Densidade Espectral de Potência serão de grande utilização devido a possibilidade de melhor visualização destas, uma vez que cada curva pode ser apresentada em cor distinta em seu respectivo gráfico, o que torna melhora a sua análise.

Referente a ABP, sugere-se um estudo para demonstrar a sua eficiência, por meio da aferição dos conceitos apreendidos durante as aulas práticas que utilizam deste simulador, na área das engenharias, a fim se pode constatar que a aprendizagem é mais efetiva que nas aulas teóricas sem o seu uso.

Figura 28 - https://nanohub.org/resources/trapsimulator/usage

REFERÊNCIAS

BELL, S. Project-based learning for the 21st century: skills for the future. **The Clearing House**, London, v. 83, n.2, p.39-43, July 2010.

BRAGA, W. Evaluating Students on Internet Enhanced Engineering Courses. In: FRONTIERS IN EDUCATION CONFERENCE, 32., 2002, Boston, USA. **Proceedings . . .** Boston: IEEE, 2002. p. 1-6.

FURTADO, G. F. Deterministic methodology for electrical simulation of BTI induced pulse broadening. **IEEE Transactions on Device and Materials Reliability**, New York, v. 17, n. 3, p. 507-513, Dec. 2007.

GRASSER, T. **Bias temperature instability for devices and circuits**. New York: Springer, 2014.

HAARTMAN, M. V.; ÖSTLING, M. **Low-frequency noise in advanced MOS devices**. Dordrecht: Springer, 2007.

KANDIAH, K.; WHITING, F. B. Low frequency noise in junction field effect transistors. **Solid-State Electronics**, Amsterdam, v 21, p. 1079-1088, August 1978.

KRISHNAN, S. Promoting interdisciplinary project-based learning to build the skill sets for research and development of medical devices in academia. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE EMBS, 35., 2013, Osaka, Japan. **Proceedings . . .** Osaka: IEEE, 2013. p. 3142-3145.

LEITE, V. Innovative learning in engineering education: experimenting with short-term project-oriented research and project-based learning. In: INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS, 26., 2017, Edinburgh, UK. **Proceedings . . .** Edinburgh: IEEE, 2017. p. 1555-1560.

LEYRIS, C. et al. Impact of random telegraph signal in CMOS image sensors for low-light levels. In: SOLID-STATE CIRCUITS CONFERENCE, 2006, Montreux, Switzerland. **Proceedings . . .** Montreux: IEEE, 2006. p. 376-379.

LUNDSTROM, M.; KLIMECK, G. The NCN: Science, simulation, and cyber services. In: CONFERENCE ON EMERGING TECHNOLOGIES – NANO ELECTRONICS, 2006, Singapore, Singapore. **Proceedings . . .** Singapore: IEEE, 2006. p. 496-500.

MCDERMOTT, J. J. **The philosophy of John Dewey**. Chicago: University of Chicago Press, 1981.

MONTGOMERY, D. C.; RUNGER G. C. **Applied Statistics and Probability for Engineers**. New York: John Wiley & Sons, Inc, 2003

NANOHUB. **Nanohub.org usage**. Articles. Disponível em: <https://nanohub.org/usage>. Acessado em: 25 de nov. de 2017.

NANOHUB. **Trapsimulator:** Tools. Disponível em: <https://nanohub.org/tools/trapsimulator>. Acesso em: 20 de out. de 2017.

PAVELKA, J. et al. Amplitude of RTS noise in MOSFETs. In: IEEE INTERNATIONAL CONFERENCE ON MICROELECTRONICS - ICM, 2009, Marrakech, Morocco. **Proceedings . . .** Marrakech: IEEE, 2009. p. 346-349.

PSOTKA, J. Educational games and virtual reality as disruptive technologies. **Educational Technology & Society**, Valdosta, v.16, n.2, p.69-80, Apr. 2013.

REISINGER, H. The time-dependent defect spectroscopy. In: GRASSER, T. **Bias temperature instability for devices and circuits.** New York: Springer, 2014. p. 75-110.

SEDRA, A. S.; SMITH, K. C. **Microelectronic circuits**. New York: Oxford University Press, 1998.

STEPIEN, W.; GALLAGHER, S. Problem-based learning: as authentic as it gets. In: FOGARTY, R. **Problem-based learning:** a collection of articles. Arlington Heights: SkyLight, 1998.

UNESCO**. The four pillars of learning**. Education. Disponível em: <http://www.unesco.org/new/en/education/networks/global-networks/aspnet/about-us/strategy/the-four-pillars-of-learning/>. Acesso em: 05 de dez. de 2017.

WAGNER, T. **Creating innovators:** The making of young people who will change the world. New York: Scribner, 2012.

WIRTH, G. I.; REIS, R.; CAO, Y. **Circuit design for reliability**. New York: Springer, 2015.

WIRTH, G. I.; SILVA, R.; BREDERLOW, R. Statistical model for the circuit bandwidth Dependence of Low-Frequency Noise in Deep-Submicrometer MOSFETs. **IEEE Transactions on Electron Devices**, New York, v.54, n. 2, p. 340-345, Apr. 2007.

APÊNDICE A - Documentação de descrição das funções que compõem o Software TrapSimulator.

DESCRIÇÃO DE ESTRUTURAS

trapped_event_type: Utilizada para armazenar quantidades de armadilhas em estado de captura.

tqueue_histogram: Estrutura de fila utilizada para armazenar tempos em estado de captura.

trap_type: Utilizada que armazena os parâmetros de cada armadilha.

trap_hist: Utilizada como estrutura *wrapper* para a impressão de dados em tela do histograma de armadilhas em estado de captura.

current_type: Utilizada para armazenar parâmetros de corrente elétrica em cada interação.

tqueue_freq_type: Estrutura de fila utilizada para armazenar parâmetros de densidade espectral de potência em cada frequência.

tnode: Estrutura de árvore utilizada para calcular a função de densidade de probabilidade da função que gera número aleatório em uma distribuição de Poisson.

DESCRIÇÃO DE FUNÇÕES

main()

Entrada: *void*

Saída: *int*

Descrição: Executa o fluxo principal do algoritmo.

GetK()

Entrada: *long int mobility, float tox, int width, int length*

Saída: *float*

Descrição: Calcula o parâmetro K para posterior cálculo de corrente.

GetCurrent()

Entrada: *float vt0, float delta_vt0, float vg, float vd, float vs, float k, trap_type traps[], int traps_num*

Saída: *float*

Descrição: Calcula a corrente em cada interação e retorna o seu valor em Amperes.

GetTrapState()

Entrada: *trap_type traps[], int trap_num, float sample_step, tqueue_histogram *hist*

Saída: *int*

Descrição: Define o estado de todas armadilhas em cada interação e retorna um inteiro maior que zero se houver uma troca de estado de “ocupada” para “desocupada”.

GetNumTrapped()

Entrada: *trap_type traps[], int traps_num*

Saída: *int*

Descrição: Retorna a quantidade de armadilhas em estado de captura.

GetHistogramVector()

Entrada: *int v_aux[], trap_hist trap_events[], long int time_analysis*

Saída: *void*

Descrição: Retorna um vetor de densidade de ocorrências de eventos de captura que é passado por referência.

GetNoiseAmplitudeByOne()

Entrada: *float k, float vg, float vd, float vs*

Saída: *float*

Descrição: Retorna o valor da amplitude de um ruído causado por uma única armadilha.

GetTrappedEventHist()

Entrada: *tqueue_histogram *queue*

Saída: *float*

Descrição: Retorna um vetor de densidade de eventos considerando o seu tempo de duração.

GetInitialTrapState()

Entrada: *float tc, float te*

Saída: *bool*

Descrição: Retorna um valor booleano representando o estado inicial de cada armadilha.

RndPoisson()

Entrada: *int lambda*

Saída: *int*

Descrição: Retorna um número aleatório a partir de uma distribuição de Poisson com média *lambda*.

PdfPoisson()

Entrada: *int lambda, int x*

Saída: *double*

Descrição: Retorna a probabilidade de ocorrência de *x* em uma distribuição de Poisson com média *lambda*.

Fat()

Entrada: *long int n*

Saída: *long int*

Descrição: Retorna o fatorial de *n*.

CreateTreeProb()

Entrada: *int mean*

Saída: *tnode*

Descrição: Cria uma estrutura de árvore para armazenar a função de densidade de probabilidade de Poisson e retorna o nó raiz.

GetTotalProb()

Entrada: *tnode *root*

Saída: *double*

Descrição: Retorna o somatório das probabilidades da árvore de densidade de probabilidade de Poisson.

AddNode()

Entrada: *tnode *root, tnode *node*

Saída: *void*

Descrição: Adiciona o nó *node* na árvore com nó raiz *root*.

DefineProbIntervals()

Entrada: *tnode *node, double *temp_value*

Saída: *void*

Descrição: Define o comprimento de cada nó de probabilidade da arvore com nó *root*.

get_index()

Entrada: *tnode *root, double rnd, int *result*

Saída: *void*

Descrição: Retorna o índice por referência no ponteiro *result* do nó que corresponde a intervalo sorteado.

GetRandomLogScale()

Entrada: *int min_pow, int max_pow*

Saída: *float*

Descrição: Retorna um número aleatório em uma distribuição logarítmica uniforme localizado entre $1x10^{min\_pow}$ e $1x10^{max\_pow}$.

GetRandom()

Entrada: *void*

Saída: *float*

Descrição: Retorna um número aleatório em uma distribuição uniforme.

CreateComplexVectorWn()

Entrada: *long int N*

Saída: *complex_type **

Descrição: Retorna um vetor rotacional com *N* elementos.

Fft()

Entrada: *float *xn,long int N, float sample_freq*

Saída: *tqueue_freq_type **

Descrição: Retorna um vetor de densidade espectral de potência com *N* elementos.

rpLibrary()

Entrada: *char* path*

Saída: *RpLibrary**

Descrição: Retorna o arquivo de dados que correspondem a configuração inicial.

rpGetString()

Entrada: *RpLibrary* io, char* name, char* data*

Saída: *void*

Descrição: Retorna o valor correspondente a variável *name* no arquivo *io*.

rpUtilsProgress()

Entrada: *int progress, char* message*

Saída: *void*

Descrição: Atualiza a barra de progresso de simulação.

rpPutString()

Entrada: *RpLibrary* io, char* name, char* data, int operationMode*

Saída: *void*

Descrição: Armazena no arquivo *io* os dados da variável *name*.

rpResult()

Entrada: *RpLibrary* io*

Saída: *void*

Descrição: Encerra a inclusão de dados no arquivo *io*.