

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Engenharia Elétrica

Reverse Engineering of VLSI Chips

Trabalho de Conclusão de Curso apresentado à Universidade Federal de
Uberlândia por Guilherme de Lima Fernandes para obtenção do título
de Engenheiro Elétrico – Ênfase Engenharia de Computação

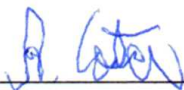
Uberlândia — 2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Faculdade de Engenharia Elétrica

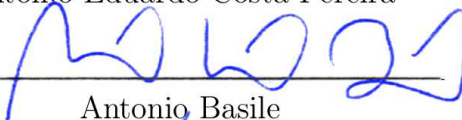
Reverse Engineering of VLSI Chips

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Uberlândia por Guilherme de Lima Fernandes para obtenção do título de Engenheiro Elétrico – Ênfase Engenharia de Computação

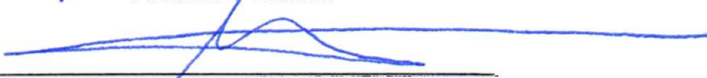
Banca Examinadora



Antonio Eduardo Costa Pereira



Antonio Basile



Noelio Oliveira Dantas

Uberlândia — 2017

À minha esposa, filhos.

Agradecimentos

Agradeço a Deus pela vida, por desfrutar de oportunidades, por conhecer e sentir a cada dia mais sua criação.

Ao Prof. Dr. Noelio Oliveira Dantas pelo incentivo e por disponibilizar o LNMIS (Laboratório de Novos Materiais Isolantes e Semicondutores), sem o qual esse trabalho não seria possível.

A Profa. Dra. Anielle Christine Almeida Silva pela relevante contribuição científica, quanto as análises de espectros de fotoluminescência, no âmbito do LNMIS, e a colaboração em artigos científicos.

Aos Professores e Funcionários da Faculdade de Engenharia Elétrica (FEELT), pela oportunidade de aprender ainda mais sobre conceitos da engenharia. Além disso agradeço também aos Professores e Funcionários do Instituto de Física e do Instituto de Química da UFU pelos auxílios prestados.

Aos meus pais, Ney e Artemisia, ao meu irmão Bruno, a minha esposa Daniela, aos meus filhos, Gabriel e Rafael, aos meus sogros, sogras, cunhados, concunhados, cunhadas, concunhadas e toda minha família pelo carinho, paciência, compreensão, apoio e companheirismo. Agradeço também ao meu irmão Leonardo pelos dias em que esteve presente em vida, transmitindo muita alegria e afeto, demonstrando também que a persistência é o caminho do êxito, pois obstáculos sempre existirão e nós sempre poderemos superá-los.

Ao CNPq, a CAPES e a FAPEMIG quanto às colaborações oferecidas a pesquisa, além do financiamento de equipamentos para UFU.

Abstract

This work deals with reverse engineering, which is a set of methods for obtaining the design and fabrication details of an object from a working specimen.

Reverse Engineering in Computer Hardware Design requires skills that few people can boast of having, those being solid state physics, atomic force and Raman microscopy, programming, cryptography, electronics, VLSI design (Very-large-scale integration design), Lambda Calculus, Functional Languages, Artificial Intelligence, etc. However, the present work is based on the premise that a person who is willing to learn a functional programming language and a few open source libraries will be able to ascertain useful information for the design and construction of a working microcontroller.

Reverse engineering technics used to inspect a microchip can also be used for quality control. The following techniques are worth mentioning: Atomic Force Microscopy, Raman Spectroscopy, Photoluminescence Spectroscopy, Scanning Electron Microscopy coupled with Energy Dispersive X-ray detector (EDX). The author of the present work had access to all equipment and tools necessary for performing advanced microscopy and spectroscopy. To make a long story short, microscopy is used to survey the structure, component shapes and material used in the building of integrated circuits.

Quantum dots are very small semiconductor particles, a few nanometers in size. They have a central role in nanotechnology, since they emit

light in specific frequencies, when submitted to an electrical current.

A quantum dot is a semiconductor nanostructure that confines the motion of conduction band electrons, valence band holes, or excitons in all three spatial directions. A quantum dot has a discrete quantized energy spectrum. That is the reason for the name.

The author believes that the main original contributions of the present work is the measurements necessary to verify the control of CdTe quantum dot growth in a silicate glass matrix. In the experiment, the author and his collaborators have used Absorption Spectroscopy and Atomic Force Microscopy. In the same experiment, Photoluminescence Spectroscopy has shown that the cadmium vacancies diffused away from the quantum dots.

In another experiment, the author used the same kind of tools, to wit, Atomic Force Microscopy and spectroscopy, in order to control densities of manganese ions and cadmium vacancies in $\text{Cd}_{1-x}\text{Mn}_x\text{Te}$ quantum dots that grew in a glass matrix. In this second experiment, manganese ions replaced cadmium. In consequence, the quantum dots have magnetic properties that arises from the spin of manganese ions.

The author and his collaborators succeeded in publishing the results of their experiments involving quantum dots in high impact periodicals in the field of Solid State Physics.

The present review also propose methods for quality control of VLSI circuits. These methods use the same technology that the author introduced in the study of quantum dots: Atomic Force Microscopy, Raman Spectroscopy, Photoluminescence Spectroscopy and Scanning Electron Microscopy coupled with Energy Dispersive X-ray detector.

Besides the original contributions, this document present an easy to follow review of Solid State Device Physics. The knowledge necessary for reading this work is very basic, and can be found in an introductory text like Arthur Beiser's Concepts of Modern Physics.

Resumo

Este trabalho trata de um tema polêmico, que é a engenharia reversa. Evidentemente, vamos estudar um conjunto de métodos para obter detalhes sobre o projeto e fabricação de circuitos integrados a partir de um SPECIMEN. A finalidade desse estudo é aprender Computer Hardware Engineering. Assim, não vamos discutir engenharia reversa PER SE.

Engenharia Reversa em Computer Hardware Engineering exige habilidades que pouca gente pode orgulhar-se de ter: Física de Estado Sólido, Microscopia de Força Atômica, Espectroscopia de Raman, Programação, Criptografia, Eletrônica, VSLI Design (Very-large-scale integration design), Calculo Lambda, Linguagens Funcionais, Inteligência Artificial, etc. Entretanto, o presente trabalho é baseado na premissa de que uma pessoa que queira aprender uma linguagem funcional, mecânica quântica, Física de Estado Sólido e algumas open source libraries será capaz de conseguir informação para projetar e construir um microcontrolador.

Técnicas de Engenharia Reversa usadas para inspecionar um microchip podem também ser úteis em controle de qualidade. As seguintes técnicas são dignas de menção: Microscopia de Força Atômica, Espectroscopia de Raman, Programação, Criptografia, Eletrônica, VSLI Design (Very-large-scale integration design), Calculo Lambda, Linguagens Funcionais, Inteligência Artificial, etc. Entretanto, o presente trabalho é baseado na premissa de que uma pessoa que queira aprender uma linguagem funcional, mecânica quântica, Física de Estado Sólido e algumas open source libraries será capaz de conseguir informação para projetar e

construir um microcontrolador.

Técnicas de Engenharia Reversa usadas para inspecionar um microchip podem também ser úteis em controle de qualidade. As seguintes técnicas são dignas de menção: Microscopia de Força Atômica, Espectroscopia de Raman, Programação, Criptografia, Eletrônica, VSLI Design (Very-large-scale integration design), Calculo Lambda, Linguagens Funcionais, Inteligência Artificial, etc. Entretanto, o presente trabalho é baseado na premissa de que uma pessoa que queira aprender uma linguagem funcional, mecânica quântica, Física de Estado Sólido e algumas open source libraries será capaz de conseguir informação para projetar e construir um microcontrolador.

Técnicas de Engenharia Reversa usadas para inspecionar um microchip podem também ser úteis em controle de qualidade. As seguintes técnicas são dignas de menção: Microscopia de Força Atômica, Espectroscopia Raman, Espectroscopia de Fotoluminescência, Microscopia Eletrônica de Varredura acoplado com detector de Energia Dispersiva de Raios-X (EDX). O autor do presente trabalho teve acesso à maioria dos equipamentos e ferramentas necessárias para realizar avançadas operações de microscopia e espectroscopia. Deixando os entretantos e chegando ao finalmente, microscopia é utilizado para investigar estruturas e formatos de componentes. Espectroscopia determina o material usado para construir circuitos integrados. O autor utilizou principalmente Microscopia de Força Atômica e Eletrônica. Em outros países, contudo, usa-se Microscopia de Tunelamento. O autor desse trabalho já entrou em contato com os professores B. Fejer e T. C. Shen, da Utah State University, para conseguir acesso a um microscópio de tunelamento, assim como de outros equipamentos necessários para fazer Computer Hardware Engineering.

Pontos quânticos são partículas de semicondutores muito pequenas, com dimensões de poucos nanômetros. Pontos quânticos têm um papel central na nanotecnologia, já que emitem luz em frequências específicas,

quando submetidas a uma corrente elétrica.

Um ponto quântico é uma nanoestrutura de semicondutor que confina o movimento de condução na faixa de energia permitida para elétrons, lacunas de valência ou excitons em todas as três direções espaciais. O ponto quântico tem um espectro de energia discreto e quantizado. Essa é a razão para o nome.

O autor acredita que a principal contribuição original do presente trabalho seja as medidas necessárias para verificar o controle do crescimento de pontos quânticos de CdTe em uma matriz vítrea silicato. Em um experimento, o autor e seus colaboradores usaram Espectroscopia de Absorção e Microscopia de Força Atômica. No mesmo experimento, Espectroscopia de Fotoluminescência mostrou que as vacâncias de cadmium difundem-se para fora dos pontos quânticos.

Em outro experimento, o autor usou as mesmas ferramentas, a saber, Microscopia de Força Atômica e Espectroscopia, para verificar o controle das densidades de íons de manganês e vacâncias de cadmium em pontos quânticos de $\text{Cd}_{1-x}\text{Mn}_x\text{Te}$ que cresceram em uma matriz vítrea. Nesse segundo experimento, os íons de manganês substituíram o cadmium. Em consequência, os pontos quânticos têm propriedades magnéticas que emergem do spin de íons de manganês.

O autor e seus colaboradores conseguiram publicar os resultados de seus experimentos envolvendo pontos quânticos em periódicos de alto impacto no campo da Física de Estado Sólido.

O presente review também propõe métodos para controle de qualidade de circuitos VLSI. Esses métodos usam a mesma tecnologia que o autor introduziu no estudo de pontos quânticos: Microscopia de Força Atômica, Espectroscopia Raman, Espectroscopia de Fotoluminescência, Microscopia Eletrônica de Varredura acoplado com detector de Energia Dispersiva de Raios-X.

O autor e seus colaboradores pretendem publicar os métodos de controle de qualidade de circuitos VLSI em uma revista de alto impacto em

Engineering Education. Além disso, o autor contatou alguns centros de excelência no exterior para iniciar uma colaboração na área de Computer Hardware Engineering. O autor acredita que as entusiásticas respostas que obteve a essas propostas de colaboração mostram que está no caminho certo.

Além das contribuições originais, esse documento apresenta uma revisão fácil de seguir da Física de Estado Sólido. O conhecimento necessário para ler esse trabalho é bastante básico, e pode ser encontrado em um texto introdutório, como o *Concepts of Modern Physics* de Arthur Beiser.

Sumário

1	Quantum dots	1
1.1	Engenharia no Brasil	2
1.2	Prova de conceito	3
1.3	Quantum dots	4
2	Estrutura de bandas	7
3	Shell	13
—	shell	13
—	bash	13
—	terminal	13
—	emacs – instalação	13
—	shell de dentro do <i>Emacs</i>	14
—	mudando o <i>prompt</i> do terminal	15
—	‘exit’ fecha o terminal	16
—	comandos do <i>Emacs</i>	16
—	<i>Emacs</i> – teclas modificadoras	16
3.1	Emacs e engenharia	18
3.2	Comandos do Emacs	18
3.3	Engenharia eletrônica.	19
3.4	Configurando emacs	20
3.5	Simulador	21
3.6	Flymake	22

3.7	emacs –daemon	23
—	evince veritedium.pdf – view veritedium.pdf	28
—	cd ~ – change to home folder	29
—	cp vauto.el tags.el	29
—	mv vauto.el features.el – troca nome	29
—	rm features.el – remove arquivo	29
—	cp -r -f \$HOME/ia . – copy \$HOME/ia para cá	30
—	mv \$Home/ia/lshort.pdf . – move arquivo para cá	30
—	rm -r -f ia – apaga diretório ia	30
—	instalação de pacotes	30
—	sudo apt-get install vim-gtk – instala vim-gtk	31
—	tab completa comando	31
4	Reverse Engineering	33
4.1	Examinando um circuito	35
4.2	Process Engineer	37
4.3	Silicon	38
4.4	MOS gates	39
4.5	Tabelas e sequenciadores	40
4.6	Stick diagrams	43
4.7	Considerações gerais sobre PUN e PDN.	45
4.8	Stick diagram de uma porta lógica	49

Capítulo 1

Quantum dots

Sei que todos estão ansiosos para ler um material acessível sobre quantum dots. Meus leitores ficariam frustrados se, antes de falar sobre *quantum dots* e os video-games maravilhosos que essa tecnologia possibilita, esse trabalho contivesse vários capítulos sobre *Quantum Mechanics*, *Solid State Physics*, poço de potencial, X-Ray Chrystallography, EDX, *van der Waals force*, *Kroning-Penney model* e coisas tais. Então, vamos combinar o seguinte: Leremos um capítulo sobre a tecnologia de quantum dots. Depois, falaremos brevemente sobre *Quantum Mechanics* e outros assuntos difíceis. O leitor casual pode parar no presente capítulo, para entender a contribuição do autor.

É costume escrever palavras do latim e do grego em itálico, sempre que for usada a ortografia dessas línguas. Mas palavras desse tipo, em um trabalho como este, são tantas que o autor do presente trabalho não vai seguir o costume. Ficaria cansativo. Então termos como *van der Waals force*, *silicon foundry*, *X-Ray Chrystallography* e *Solid State Physics* aparecerão em tipo normal. A nova ortografia, ao reintroduzir as letras Y e K no alfabeto português, permite isso.



Enquanto você faz relatórios,
há um chinês trabalhando e outro estudando

Figura 1.1: Competindo com os chineses

1.1 Engenharia no Brasil

Para trabalhar com alta tecnologia, muita gente acha que são necessárias as seguintes condições:

1. Ser chinês, de preferência nascido na província de Guangdong.
2. Fluência em cantonês e russo.
3. Ter estudado na Bauman University.
4. Ter uma Silicon foundry de 10 bilhões de dólares.

Deixemos para outro trabalho a discussão das três primeiras condições. Quanto ao quarto item, a silicon foundry de 10 bilhões de dólares

só é necessária se você quiser fabricar 1 milhão de Systems on Chip (SoC) por ano, com rigoroso controle de qualidade e a um custo de 2 dólares cada um. Mas se você só quiser uma prova de conceito da tecnologia, o preço de um boa casa é suficiente para começar.

Mas como conseguir o dinheiro? Se você perder tempo escrevendo projetos, administrando o dinheiro e fazendo relatórios, não produzirá nada de útil. A ideia é unir forças com pessoas interessadas em ver a coisa funcionando, em vez de preocupar-se com a avaliação da CAPES ou do MEC.

1.2 Prova de conceito

Vamos tomar um exemplo do que estamos falando. Nas universidades brasileiras, em especial nos cursos de engenharia, os professores tentam preparar os estudantes para criar produtos, inovar e abrir empresas. Ao fim de cada disciplina e durante o estágio, o estudante é obrigado a fazer produtos. Durante todo o aprendizado, o futuro engenheiro é doutrinado a criar uma startup.

A primeira crítica a tais projetos estudantis é que os produtos propostos são toscos, coisa que ninguém compraria. Inovação só existe nos nomes dos grupos de trabalho e nos títulos de relatório. Geralmente a tecnologia utilizada é arcaica e copiada de páginas da Internet sobre DIY (Do It Yourself). Em suma, nossos jovens engenheiros querem abrir startups sem produto.

Por circunstâncias que não vamos discutir aqui, os autores desse trabalho tiveram a seguinte proposta: Estudar a nanotecnologia de quantum dots, utilizada para construir televisões e monitores de Spectacular Ultra High Definition (SUHD) e obtenção de cores precisas. Por cores precisas, queremos dizer exatamente iguais àquelas vistas na natureza.

Evidentemente, televisores de Spectacular Ultra High Definition interessam ao mercado. Mas o autor desse trabalho pede ao leitor que

examine a questão sob o seguinte prisma: Para abrir uma startup, o autor teria que concorrer com a SAMSUNG e com a Amazon, que já estão vendendo equipamentos com a tecnologia de quantum dots. Essa proposta é, obviamente, absurda. Então, esse trabalho de conclusão de curso vai trilhar um caminho bem mais modesto. O autor vai tentar contribuir para a nanotecnologia em geral e para quantum dots em particular. A prova de que a contribuição é válida são artigos em revistas internacionais de alto impacto[26], [27], [28]. Um pouco de modéstia e humildade aproximam as expectativas da realidade e evitam o choque de acordar de um sonho inalcançável.

1.3 Quantum dots

Dizem os experts que quantum dots têm o potencial de revitalizar a tecnologia de display em nossos celulares, tablets, TVs, consoles de games e laptops. De fato, a tecnologia já está no mercado, e várias empresas estão vendendo tablets que usam quantum dots em seus displays. Evidentemente, todos os fabricantes de televisão adotaram a tecnologia por volta do ano de 2015.

Afinal de contas, o que são quantum dots? Basicamente, são partículas incrivelmente pequenas, contendo aproximadamente 50 átomos. O autor viu-se impossibilitado de medir essas coisas com uma régua comprada na papelaria da esquina. Além disso, não pode vê-las com uma lupa de plástico made in China. Por isso, precisou de experimentar algumas tecnologias alternativas: High Resolution Optical Microscopy, Scanning Electron Microscopy coupled with Energy Dispersive X-ray detector (EDX), Atomic Force Microscopy, Raman Spectroscopy e Photoluminescence Spectroscopy.

Na figura 1.2, rotulada com (a), podemos ver a matriz vítrea contendo Boron, Silicon, Aluminium e Natrium. No rótulo (a), a matriz vítrea é mostrada antes do autor fazer a dopagem com os quantum dots. No

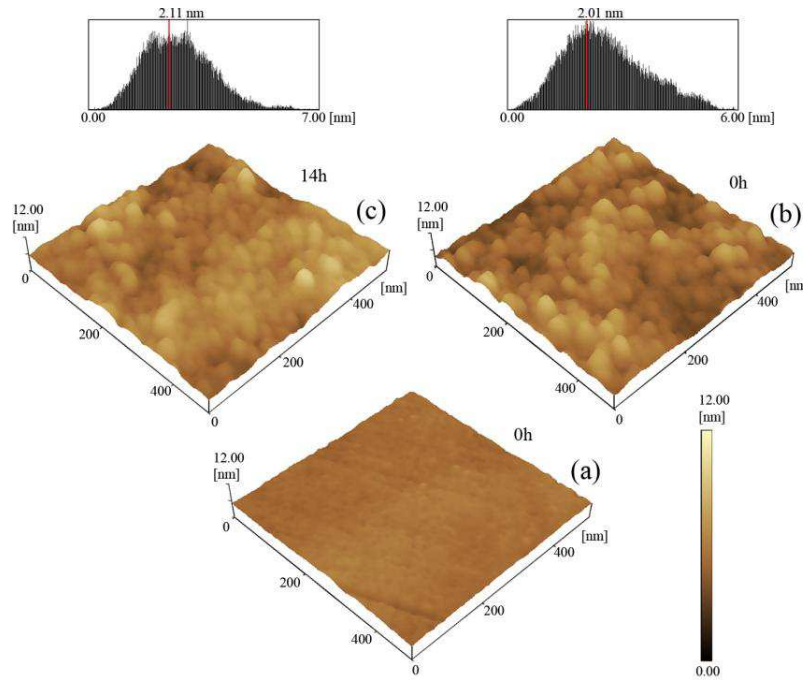


Figura 1.2: Quantum dots em uma matriz de Borosilicato

rótulo (b), podemos ver a matriz dopada com quantum dots antes do tratamento térmico utilizado pelo autor para controlar o crescimento dos quantum dots. No rótulo (c), o leitor pode ver os quantum dots depois do tratamento térmico.

Uma propriedade importante das quantum dots é que os portadores de carga, os elétrons e lacunas, são confinados nas três dimensões em níveis de energia quantizados. Por esse motivo, os quantum dots têm espectros bem definidos e são chamados de átomos artificiais.

Na figura 1.3, podemos ver os Energy Gaps de vários semicondutores. A bolinha na extremidade inferior do segmento representa o gap para o material sem confinamento (bulk). A setinha para baixo indica o gap para o ponto quântico com maior energia em relação ao bulk. Entre eles, a seta para cima mostra o gap para o ponto quântico de menor energia.

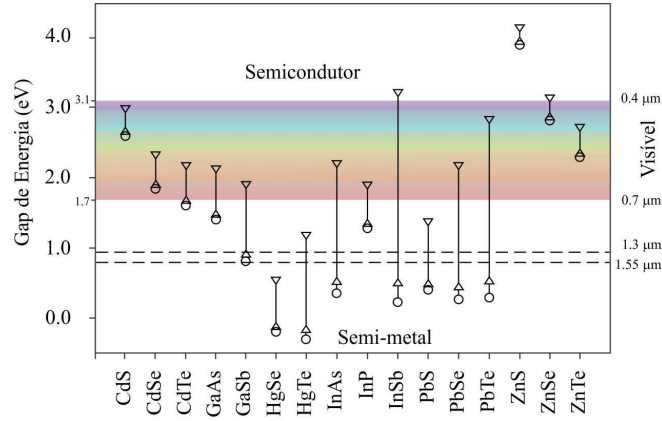


Figura 1.3: Energy Gaps

Explicando melhor a figura 1.3: Um sólido cristalino ou cristal é um material sólido cujos átomos, moléculas ou íons são dispostos num padrão ordenado que se estende nas três dimensões espaciais. Conforme vamos ver, a mecânica quântica mostra que um material com essas características forma bandas de energia, onde partículas reais ou virtuais que conduzem eletricidade podem habitar. Por isso essas bandas de energia permitida são ditas condutoras.

Para os materiais semicondutores, um parâmetro importante é o intervalo de energia que separa a banda de condução da banda de valência. Esse intervalo é conhecido como gap de energia.

Em semicondutores bulk, o gap de energia é um parâmetro fixo que identifica o material. Já em nanopartículas semicondutoras, conforme mostra a figura 1.3, os portadores de carga podem estar em um regime de confinamento quântico, em que as excitações eletrônicas sofrem influências das restrições aplicadas pelas leis da física sobre a partícula e respondem à diversidade de tamanho. Esse fenômeno é conhecido como efeito de confinamento quântico. As partículas em nanoescala que exibem o efeito de confinamento são denominadas quantum dots [29].

Capítulo 2

Estrutura de bandas

A estrutura de bandas eletrônicas de um semicondutor contém informações importantes sobre uma série de parâmetros essenciais para a caracterização do material e para a fabricação e aplicação de dispositivos feitos a partir dele. O conhecimento dos extremos de energia da banda de condução e da banda de valência, assim como os valores das massas efetivas de elétrons e das lacunas, são grandezas fundamentais para o estudo de transições óticas, além de outras aplicações. Podemos afirmar também que o conhecimento da estrutura de bandas é, principalmente, do gap de energia de cada material é *CONDITIO SINE QUA NON*.

Como todo engenheiro de computação sabe, elétrons de valência que se movem nos cristais interagem com o potencial periódico gerado pela rede, conforme a expressão 2.1

$$U(\vec{r}) = U(\vec{r} + \vec{R}) \quad (2.1)$$

O potencial periódico é devido ao posicionamento fixo dos núcleos atômicos em uma estrutura de grade. Para resolver problemas de engenharia, muitas vezes é preciso trabalhar com modelos simplificados. Vamos imaginar um lattice unidimensional, com os núcleos regularmente espaçados, como mostrado na figura 2.1

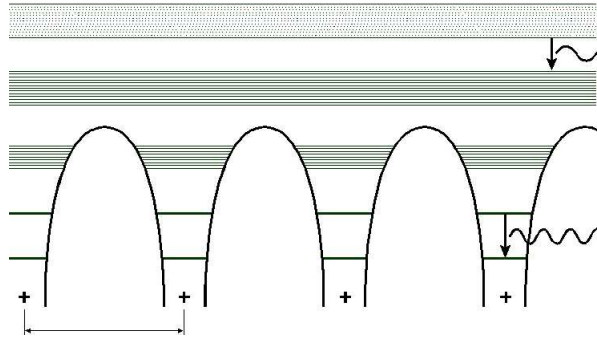


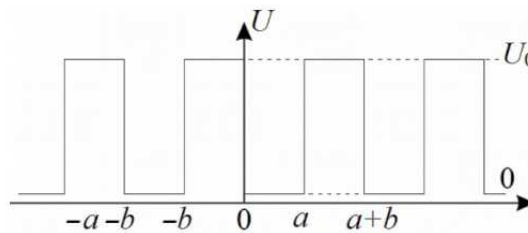
Figura 2.1: Potencial periódico unidimensional

Basicamente, o potencial elétrico de uma partícula carregada muito pequena é dado pela seguinte expressão:

$$U(r) = k_e \frac{qQ}{r}$$

A variável r é a distância entre a partícula e um dos núcleos. Quando r fica muito pequeno, $U(r)$ cresce. Quando r fica entre dois núcleos, $U(r)$ atinge um valor mínimo.

Com a simplificação discutida acima, poderíamos resolver o problema por métodos numéricos. Mas como só queremos mostrar como as bandas de condução aparecem, vamos usar o modelo de Kronig-Penney, que é ainda mais simplificado: o potencial é representado por uma onda quadrada, como mostrado abaixo.



No potencial de Kronig-Penney, a largura do poço é a e a largura

da barreira é b . O período é $P = a + b$. A equação de Schrödinger independente do tempo é dada por:

$$-\frac{\hbar}{2m} \frac{\partial^2 \psi}{\partial x^2} + U(x)\psi(x) = E\psi(x)$$

Essa equação reduz-se às seguintes formas:

Dentro dos poços.

$$\frac{d^2 \psi}{dx^2} = -\frac{2mE}{\hbar^2} \psi$$

para $0 < x < a$. Portanto, a solução é dada por

$$\psi_1 = Ae^{iKx} + Be^{-iKx} \quad (2.2)$$

com $K^2 = 2mE/\hbar$ definido por conveniência.

Nas barreiras.

$$\frac{d^2 \psi}{dx^2} = \frac{2m(U_0 - E)}{\hbar^2} \psi$$

A solução é

$$\psi_2 = Ce^{\kappa x} + De^{-\kappa x} \quad (2.3)$$

com $\kappa^2 = 2m(U_0 - E)/\hbar^2$ para obter-se uma notação mais suscinta.

Resumindo, a função de onda 2.2 corresponde ao poço entre 0 e a com $E = \hbar^2 K^2/2m$, enquanto a função de onda 2.3 descreve o comportamento das partículas entre a e $a + b$ com $U_0 - E = \hbar^2 \kappa^2/2m$.

Teorema de Bloch. O teorema de Bloch descreve o movimento de elétrons em um sólido. Foi enunciado pelo suíço Felix Bloch. Ele estudou na Eidgenössische Technische Hochschule Zürich (ETH). Recebeu o prêmio Nobel em 1952 por desenvolver métodos precisos de medir efeitos magnéticos nucleares. O que nos interessa, entretanto, é sua tese de doutoramento sobre a teoria quântica dos sólidos, onde demonstra o famoso

teorema de Bloch. A seguir vamos enunciar o teorema e dar um exemplo detalhado de aplicação.

Conforme vimos, a representação do potencial em um cristal é uma função periódica com período $P = a + b$. O teorema de Bloch estabelece que $\psi(x + r) = \exp(ikr)\psi(x)$. Logo, a função de onda para o ponto $x = -b$ tem a forma $\psi(a - P) = \exp(-ikP)\psi(a)$, ou seja:

$$\psi(-b) = \exp(-ikP)\psi(a)$$

O vetor de onda k da função de Bloch é uma nova quantidade e será cuidadosamente expresso a partir de K e κ . A seguir obtemos as primeiras equações para determinar as relações entre os vetores de onda, onde foram aplicadas as condições de contorno e a continuidade de ψ e ψ' (derivada de ψ) nos pontos $x = 0$ e $x = a$. Além disso, será utilizado também o teorema de Bloch sobre a função ψ_2 nos pontos a e $-b$.

$$\psi_1(0) = \psi_2(0) \quad A + B = C + D \quad (2.4)$$

$$\psi'_1(0) = \psi'_2(0) \quad iKA - iKB = \kappa C - \kappa D \quad (2.5)$$

$$\psi_1(a) = \psi_2(a) \quad Ae^{iKa} + Be^{-iKa} = e^{ikP}(Ce^{-\kappa b} + De^{\kappa b}) \quad (2.6)$$

$$\psi'_1(a) = \psi'_2(a) \quad iKAe^{iKa} - iKBe^{-iKa} = e^{ikP}(\kappa Ce^{-\kappa b} - \kappa De^{\kappa b}) \quad (2.7)$$

Forma matricial. As equações 2.4, 2.5, 2.6 e 2.7 podem ser colocadas na forma matricial, conforme mostrado na equação 2.8. A solução não trivial exige que o determinante da matriz seja 0. A pergunta é: Como fazer essa manipulação algébrica? Não que seja impossível resolver esse

problema a mão. Acontece que o pesquisador gasta muito tempo para obter a solução.

$$\begin{bmatrix} 1 & 1 & -1 & -1 \\ iK & -iK & -\kappa & \kappa \\ e^{iKa} & e^{-iKa} & -e^{-\kappa b + ikP} & -e^{\kappa b + ikP} \\ iKe^{iKa} & -iKe^{-iKa} & -\kappa e^{-\kappa b + ikP} & \kappa e^{\kappa b + ikP} \end{bmatrix} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.8)$$

Felizmente, existe para nós um programa chamado Maxima Computer Algebra, que permite realizar matemática simbólica. Esse indispensável programa é escrito em Common Lisp e vem sendo continuamente aperfeiçoado desde 1968. Isso é possível porque Lisp é uma linguagem fortemente fundamentada na matemática e, portanto, não fica obsoleta. Deixo, porém, a discussão de Computer Algebra para o trabalho de conclusão de curso de uma colaboradora, Cecília Carneiro. Assim, contento-me em apresentar a resposta do problema:

$$\begin{aligned} \cos(kP) = & \cos\left(a\sqrt{\frac{2mE}{\hbar^2}}\right) \cosh\left(b\sqrt{\frac{2m(U_0 - E)}{\hbar^2}}\right) \\ & - \frac{2E - U_0}{2\sqrt{E(U_0 - E)}} \sin\left(a\sqrt{\frac{2mE}{\hbar^2}}\right) \sinh\left(b\sqrt{\frac{2m(U_0 - E)}{\hbar^2}}\right) \end{aligned} \quad (2.9)$$

A equação 2.9 possui comportamentos distintos em cada lado da igualdade. O lado esquerdo da expressão está limitado ao intervalo $[-1, 1]$, pois trata-se de uma função cosseno de valores reais. Já o lado direito da expressão contém fatores de funções hiperbólicas, cujo valor pode ultrapassar de muito a unidade. Então, para certos valores da energia não existirá um correspondente valor real para k que torna a igualdade possível. Isso significa que há faixas de energia proibidas. O importante nessa discussão é o aparecimento de faixas de energia, onde as partículas condutoras podem circular, e gaps, onde a energia é proibida. Felix

Bloch fez seu doutorado sob a orientação de Werner Heisenberg. Conforme dissemos, Bloch é o autor do teorema que levou à conclusão de que existem gaps de energia, ou seja, regiões do espectro onde a energia é proibida. Outro estudante de Heisenberg, Sir Alan Herries Wilson, formulou a teoria explicando como surgem as faixas de energia.

Vamos resumir tudo que foi dito nesse capítulo. Em Solid State Physics, a band structure de um sólido descreve os intervalos de energia que um elétron pode ter e os intervalos que não pode ocupar. A teoria das bandas obtém essas faixas permitidas e esses gaps proibidos examinando a função de onda de um elétron em um lattice cristalino longo e periódico. A band theory teve sucesso em explicar muitas propriedades físicas dos sólidos, tais como resistividade elétrica e absorção ótica. Além disso, essa teoria é fundamental na compreensão de todos os dispositivos de estado sólido, tais como transistores, células solares, etc.

Capítulo 3

Shell

Shell é um interpretador de comandos que atua em um **terminal**. Os principais comandos do shell apresentador a seguir tem como base o trabalho da programadora canadense chamada Nia. Para abrir o terminal, Nia mantém a tecla **Alt** pressionada e aperta **F2**. No *prompt* do diálogo *Executar Aplicativo*, ela digita `gnome-terminal` **Enter**.

Emacs – A maneira mais fácil de instalar o *Emacs* é pelo *aptitude*.

```
~$ sudo apt-get install emacs Enter
```

Feita a instalação, Nia entra no *Emacs* com o comando abaixo.

```
nia@cdr~$ emacs & Enter
```

O comando ‘`emacs &`’ não prende o terminal; assim, é possível lançar processos concorrentes com o editor.

Shell de dentro do emacs – O ambiente inicial mostra o *buffer* de boas vindas e o *minibuffer*, que é uma janela de uma linha só. Nia tecla **Esc** **x** e o *minibuffer* apresenta o prefixo `M-x` para pedir um comando. Como mostrado abaixo, Nia digita `shell` para capturar um terminal.

```
M-x shell
```

Agora Nia conseguirá utilizar o terminal de dentro do *Emacs*.

O Emacs possui também seu próprio shell, escrito inteiramente em Emacs-Lisp. O shell, chamado *eshell*, é um substituto rico em recursos, como o *bash* que pode trabalhar em qualquer plataforma que o Emacs execute, incluindo o Microsoft Windows.

```
$ cd ~ Enter      # Nia vai para $HOME
~$ mkdir ia Enter    # cria pasta ia
~$ cd ia      # e entra na pasta ia
~/ia$ _
```

Nia vai criar um programa para inserir *AUTO comments* em arquivos verilog. Ela tecla Ctrl-x Ctrl-f (mantém a tecla Ctrl pressionada e aperta sucessivamente as teclas x e f). O *minibuffer* apresenta um *prompt* para o nome do arquivo (vide figura 3.1). Nia digita: `~/ .emacs`

```
~/ $ mkdir ia
~/ $ cd ia
~/ia$ _

*shell* _____
Find file: ~/ .emacs
```

Figura 3.1: minibuffer

Recordando: Nia apertou as teclas Ctrl-x Ctrl-f para abrir um novo arquivo. O *minibuffer* pediu o nome do arquivo, e Nia digitou

```
Find file:  ~/ .emacs
```

Emacs apresenta um *buffer* contendo o arquivo `.emacs` de configuração, ou seja, uma cópia do arquivo armazenada na memória do compu-

tador. Nia digita o seguinte programa:

```
;; Inserir em ~/.emacs
(defun veritedium-auto()
  (interactive)
  (insert "/*AUTO*/") (backward-char 2) )
(global-set-key (kbd "C-x t") 'veritedium-auto)
```

Nia aperta `Ctrl-x Ctrl-s` para salvar esse *buffer*. Em seguida, ela tecla `Alt-x` e digita `load-file` no M-x *prompt* do *minibuffer*:

M-x load-file `Enter`

O *minibuffer* pede o nome do arquivo, que Nia fornece e aperta `Enter`

Load file: ~/.emacs `Enter`

Quando quiser utilizar o programa, Nia apertará `Ctrl-x t` para inserir um *AUTO comment* em volta do cursor:

`Ctrl-x t` /*AUTO_*/

Finalmente, Nia aperta `Ctrl-x Ctrl-c` para sair do emacs e voltar para o prompt do Unix.

Mudando o prompt do terminal – A *shell* utilizada no Ubuntu chama-se *bash*. Para modificar o *prompt* da *bash* de modo que mostre apenas o diretório, é preciso editar o arquivo de configuração ~/.bashrc

nia@cdr~\$ emacs -nw ~/.bashrc `Enter`

Para ir até a última linha de ~/.bashrc, Nia aperta a tecla `Esc` e solta; em seguida, tecla `Shift >` e insere o texto abaixo.

PS1="\w\$ "

Agora Nia salva o arquivo .bashrc modificado. Para salvar, basta teclar `Ctrl-x Ctrl-s` conforme já vimos antes. Para sair do emacs, Nia tecla `Ctrl-x Ctrl-c` e volta ao prompt do Unix.

Fechando o terminal – Para fechar o terminal, Nia digita `exit` seguido de  conforme mostrado abaixo.

```
nia@cdr~$ exit 
```

Quando Nia abrir o terminal novamente, após editar `~/ .bashrc`, notará que o *prompt* passou a mostrar apenas o diretório.

```
~$ _
```

Comandos do emacs – Os comandos do *Emacs* são teclas prefixadas com os modificadores *Control* () e *Meta* ():

 - 	Nia mantém  pressionada e aperta 
 - 	Nia mantém  pressionada e aperta 

Teclas modificadores – As teclas  e  são *modificadoras* e devem ser mantidas pressionadas quando se aperta a tecla que modificam. Assim, em - Nia mantém  pressionada, enquanto aperta . Neste capítulo, teclas modificadoras são separadas por hífen das modificadas.

Teclas justapostas – Quando duas teclas estão diretamente justapostas, Nia solta a primeira antes de apertar a segunda. Para digitar   Nia aperta e solta  e só depois aperta 

Representação das teclas modificadoras – Na literatura, comandos como - são representados por `\C- $\tau$` , onde τ é uma tecla qualquer. Analogamente, - é denotado por `\M- $\tau$` . Por exemplo, o comando - - de abrir arquivo é representado por `\C-x\C-f`.

Tecla de escape –   faz a mesma coisa que -. Entretanto,  não é modificadora, pois é preciso soltá-la antes de apertar 

Admitamos que Nia está dentro do *buffer* do arquivo `~/ia/vauto.el` onde digitou um programa, conforme mostrado na figura 3.2.

```
(defun veritedium-auto()
  (interactive)
  (insert "/*AUTO*/")
  (backward-char 2) )

(global-set-key (kbd "C-x t") 'veritedium-auto)

U:--- vauto.el L11 [[[(Emacs-Lisp)]]]
Switch to buffer (default *eshell*): _
```

Figura 3.2: Um programa em Emacs Lisp

Volta ao eshell. No exemplo da figura 3.2, saímos do eshell para digitar um programa. Por várias razões, podemos necessitar de abandonar um buffer para trabalhar em outro. Quando fazemos isso, o buffer original não é perdido. No caso presente, o *buffer* da *shell* não foi perdido.

Para voltar ao *buffer* da *shell*, Nia aperta `Ctrl-x b`. O *minibuffer* sugere um *buffer* de destino. Nia pode selecionar outro *buffer* rolando as opções com as setas `↑` e `↓`. Feita a seleção, Nia aperta `Enter`. Uma alternativa mais rápida é digitar `*esh` e completar o nome `*eshell*` com `Tab`. De volta à *eshell*, Nia pode verificar a presença do arquivo `~/ia/vauto.el`

```
~/ia$ ls -lia vauto.el
8263197 -rw-r--r-- 1 nia nia 234 2017-06-03 15:02 vauto.el
~/ia$ _
```

A propósito, o comando `ls -lia teste.txt` produz várias informações sobre o arquivo, tais como a data de criação e o tamanho.

3.1 Emacs e engenharia

Muita gente acredita que a escolha do editor de texto é irrelevante. Não é. Para começar, Emacs pode ser configurado, programado e incrementado, permitindo que os engenheiros do mundo inteiro colaborem durante décadas para criar ferramentas de projeto e de produtividade. Graças a isso, Emacs tem vários macros indispensáveis ao engenheiro de computação, tais como: Orgmode, que permite organizar o trabalho e modos para diversas linguagens de programação.

3.2 Comandos do Emacs

C-x C-c fecha o editor; com [Ctrl] apertada, tecle [x] e [c]
C-x C-s salva arquivo; com [Ctrl] apertada, tecle [x] e [s]
C-x C-f abre um arquivo; com [Ctrl] apertada, tecle [x] e [f]
C-g desiste de um comando que já está no minibuffer

C-d apaga caractere debaixo do cursor
C-k elimina tudo entre o cursor e o fim da linha
C-/ defaz o último comando, como: apagar, colar, digitar...

C-f move o cursor um caractere para frente
C-b move o cursor um caractere para trás
C-n move o cursor para a linha de baixo
C-p move o cursor para a linha de cima
C-a move o cursor para o começo da linha
C-e move o cursor para o fim da linha
Esc < Começo do texto; aperte Esc e solte; tecle <
Esc > Fim do texto; aperte Esc e solte; tecle >

C-x 2 quebra janela em duas
C-x o vai para a outra janela
C-x 1 fecha a outra janela, quando quebrada em duas
C-x 0 fecha a janela do cursor

Comandos de busca — Comandos para procurar uma *string* no texto:

C-s Busca incremental para frente
 C-r Busca incremental para trás
 Enter Encerra busca quando a "string" é encontrada
 C-s C-s Repetir busca

Selecionar — Textos para apagar ou copiar para o *kill ring*:

C-SPC inicia seleção. Mova o cursor para selecionar
 C-w apaga região selecionada. Deixa cópia no kill-ring
 M-w copia região selecionada para o kill-ring
 C-y insere topo do kill-ring no texto
 M-y gira kill-ring; deve ser precedido por [Ctrl]-[y]

3.3 Engenharia eletrônica.

O hacker de eletrônica monta circuitos em Printed Circuit Boards, ou PCBs. Uma PCB tem, no máximo, 300 componentes. Mas o *hardware engineer* faz CPUs com milhões de componentes menores do que um vírus. Assim, não pode raciocinar em termos de portas lógicas. Então, o *hardware engineer* raciocina sobre programas, e utiliza um compilador especial para gerar máquinas em pequenos cristais de silício.

Há várias linguagens que podem ser utilizadas pelos *hardware engineers*. Uma delas é a verilog. Construir uma CPU em verilog é tedioso. Por isso, Wilson Snyder criou a ferramenta veritedium. Basicamente, veritedium é um programa, escrito em Lisp, que cuida da parte tediosa do projeto. Para instalar a *veritedium*, você deve fazer *download* do programa `verilog-mode.el` da Internet.

Instalando verilog-mode. A operação de *download* põe o arquivo `verilog-mode.el` na pasta Downloads. Você deve transferir esse arquivo para a pasta `~/ .emacs.d/verimode` conforme mostrado a seguir.

1. Verificar se `.emacs.d` existe com `ls`.

```
~$ cd ~ # Vá para HOME
~$ ls ~/.emacs.d
```

2. Se `.emacs.d` não existir, crie a pasta.

```
~$ mkdir .emacs.d
```

3. Crie o diretório `~/emacs.d/verimode`

```
~$ mkdir ~/.emacs.d/verimode
~$ mv ~/Downloads/verilog-mode.el ~/.emacs.d/verimode/
```

3.4 Configurando emacs

O último passo para começar a projetar hardware é introduzir os *snippets* abaixo no arquivo `.emacs` de configuração do emacs.

```
;; Acentos do português
(require 'iso-transl)

;; Mouse activation in terminal
(require 'mouse)
(xterm-mouse-mode t)
(defun track-mouse (e))
(setq mouse-sel-mode t)

;; wrap word at the end of line
(global-visual-line-mode t)
(setq inhibit-splash-screen t)
(setq inhibit-startup-message t)
(set-face-attribute 'default nil :height 180)

(transient-mark-mode t)
```

```
;; Informar onde está o arquivo verilog-mode.el
(add-to-list 'load-path "~/.emacs.d/verimode")
(autoload 'verilog-mode "verilog-mode" "Verilog mode" t)
(add-to-list 'auto-mode-alist
  '("\\\\. [ds]?vh?\\\\" . verilog-mode))
```

Para digitar o programa acima, você precisa abrir `.emacs` no editor. Para isso, abra um terminal: mantenha a tecla `[Alt]` abaixada e aperte a `[F2]` key. No *prompt* do diálogo *Executar Aplicativo*, digite `gnome-terminal` `[Enter]` e entrará do terminal Unix. Abra o arquivo de configuração e ative o `verilog-mode`.

```
~$ cd ~      # Só para ter certeza de estar em HOME
~$ emacs .emacs
```

Depois de introduzir as modificações sugeridas no arquivo `.emacs` de configuração, aperte `[Ctrl]-[x]` `[Ctrl]-[s]` para salvar e `[Ctrl]-[x]` `[Ctrl]-[c]` para sair do editor. Estamos quase prontos para a ação.

3.5 Simulador

Há várias maneiras de construir computadores. Uma delas é utilizar os serviços de uma *semiconductor fabrication plant*, instalação também conhecida como *semiconductor foundry* ou, simplesmente, *fab*. Uma fab custa bilhões de dólares e fabrica *Application-specific integrated circuit* (ASIC). Para a fabricação de ASICs, você pagará centenas de milhares de dólares. Esse tipo de investimento só vale a pena se você fabricar milhares de circuitos integrados.

Outra opção é usar uma placa de desenvolvimento de *Programmable System on Chip* (PSoC), que contém um computador e vários dispositivos reconfiguráveis. O investimento é baixo, mas a qualidade também é.

Qualquer das duas opções que você escolha, ASIC ou PSoC, não deixe de simular o circuito antes de construir o *hardware*. Para simular o circuito, aconselho instalar a icarus verilog, como mostrado abaixo.

```
~/src$ git clone git://github.com/steveicarus/iverilog.git
~/src$ cd iverilog
~/src/iverilog$ autoconf
~/src/iverilog$ ./configure
~/src/iverilog$ make
~/src/iverilog$ sudo make install
```

Finalmente, você terá de instalar a gtkwave para ver as formas de onda dos sinais produzidos pelo circuito. A gtkwave é distribuída no seguinte endereço:

<http://gtkwave.sourceforge.net>

3.6 Flymake

Clone o flymake dentro do diretório `~/ .emacs.d/` e siga os passos abaixo.

```
~$ cd ~
~$ cd .emacs.d
~/ .emacs.d$ git clone https://github.com/flymake/emacs-flymake
```

Agora, descubra na Internet o arquivo `flymake-verilog.el` e copie o conteúdo para o ficheiro `~/ .emacs` de configuração.

No começo do arquivo `~/ .emacs` de configuração, antes portanto de introduzir o conteúdo de `flymake-verilog.el` ou qualquer coisa referente a flymake, indique a *path* do emacs-flymake folder:

```
(require 'iso-transl) ;; Acentos em português
(add-to-list 'load-path "~/ .emacs.d/emacs-flymake")
```

3.7 emacs -daemon

Em vez de entrar no emacs diretamente, a melhor opção é disparar um daemon e usar um cliente. Isso tornará o editor muito mais ágil.

```
~$ cd ~           # Para ter certeza de estar no HOME folder
~$ mkdir wrk      # Crie um wrk folder, se ainda não existir
~$ cd wrk
~/wrk$ mkdir verilog
~/wrk$ cd verilog
~/wrk/verilog$ emacs --daemon
Starting Emacs daemon.
~/wrk/verilog$ emacsclient -nw flipflop.v
```

Se você precisar de matar o daemon, quando entrar no emacsclient, digite **Alt-x** e, do minibuffer, emita o comando que segue:

M-x kill-emacs

AUTO comments. Na editoração do arquivo `flipflop.v` que descreve o circuito de um flipflop D, vamos contar com três ferramentas. Flymake acompanha cada passo de nosso trabalho. Cada vez que há um erro no código, flymake vai indicá-lo. O verilog mode encarrega-se de colocar as indentações corretas e colorir a sintaxe. Finalmente, Wilson Snyder criou o veritedium, que fornece as chamadas *AUTO features*.

Essas *features* procuram ocorrências de *AUTO comments* no código e as expande em textos adequados. Enfim, as *AUTO features* evitam que os engenheiros tenham de digitar código que Lisp pode inferir a partir do resto do circuito.

Depois de digitado o código da figura 3.3, basta pressionar **Ctrl-c** e **Ctrl-a** para que veritedium complete o circuito para nós. Na figura 3.4, página 25, você encontrará o circuito de teste no `fftest.v` file.

```
// flipflop.v
module flipflop(*AUTOARG*);
    input clk;
    input reset;
    input d;
    output q;
    output qb;
    reg    q;

    assign qb= ~q;
    always @(*AS*/)
        begin
    if (reset) begin
        q <= 1'b0; // Asynchronous reset
    end else begin
        q <= d; // Assign D to Q
    end
        end
endmodule // flipflop
```

Figura 3.3: Flip Flop pattern

```
// fftest.v
module test;
    reg clk;
    reg reset;
    reg d;
    wire q;
    wire qb;

    // Instantiate design under test
    flipflop DFF(*AUTOINST*/);

    initial begin
        // Dump waves
        $dumpfile("ff.vcd");
        $dumpvars(1);
        $display("Reset flop.");
        clk= 0;
        reset= 1;
        d= 1'bx;
        display;

        $display("Release reset.");
        d= 1;
        reset= 0;
        display;

        $display("Toggle clk.");
        clk= 1;
        display;
    end // initial begin

    task display;
        #1 $display("d:%0h, q:%0h, qb:%0h",
        d, q, qb);
    endtask // display
endmodule // test
```

Figura 3.4: Flip Flop Test

Circuito de teste. Examinando o arquivo `fftest.v` da figura 3.4, página 25, vemos como utilizar uma ocorrência do flipflop que foi definido no `flipflop.v` *file*.

```
// Instantiate design under test
flipflop DFF(/*AUTOINST*/);
```

Note que o nome do arquivo deve ser idêntico ao nome do módulo, caso contrário veritedium não encontraria o dispositivo.

Mostramos abaixo como compilar e rodar o programa.

```
~/wrk/verilog$ iverilog -o fliptest.x flipflop.v fftest.v
~/wrk/verilog$ ./fliptest.x
VCD info: dumpfile ff.vcd opened for output.
Reset flop.
d:x, q:0, qb:1
Release reset.
d:1, q:1, qb:0
Toggle clk.
d:1, q:1, qb:0
```

Pedimos que veritedium gerasse um arquivo `ff.vcd` que permite examinar a saída do circuito em uma aplicação chamada gtkwave.

No gtkwave, apresentado na figura 3.5 da página 27, clicando no ramo *test* do circuito, as variáveis `clk`, `d`, `q`, `qb` e `reset` aparecem em uma janela. Selecionando qualquer uma dessas variáveis com o mouse e apertando o botão , a onda correspondente é plotada em uma *canvas*, como se fosse a tela de um osciloscópio.

Lisp templates. Veritedium permite que você use um programa Lisp para determinar o nome de um fio conectado a um dado dispositivo. Voltemos ao exemplo do flipflop. Digamos que, em um dado circuito, sejam necessários mais de um flipflop.

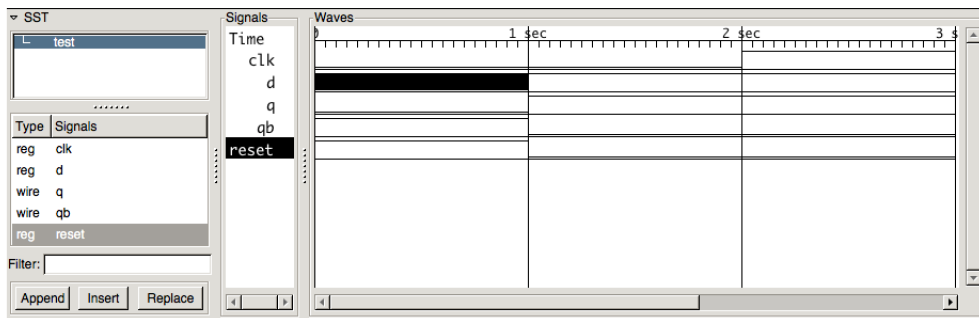


Figura 3.5: Interface da aplicação gtkwave

```

module test;
    reg clk, reset, d0, d1, d2;
    wire q, qb;

    /* flipflop AUTO_TEMPLATE (
        .d (d@"(+ @ 1)")); */
    flipflop ff0(*AUTOINST*/);
    flipflop ff1(*AUTOINST*/);
    initial begin
        $dumpfile("ff.vcd"); $dumpvars(1);
        $display("Reset flop.");
        clk= 0;
        reset= 1;
        d1= 1'bx; display;
        $display("Release reset.");
        d1= 1;
        reset= 0; display;
        $display("Toggle clk.");
        clk= 1; display;
    end // initial begin

    task display;
        #1 $display("d1:%0h, q:%0h, qb:%0h", d1, q, qb);
    endtask // display
endmodule // test

```

Na sintaxe `@"(...)"`, Veritedium avalia a Lisp *expression* entre aspas, com ocorrências de `@` substituídas pelo *instantiation number*. Qualquer aspas dentro da expressão deve ser prefixada com `\` *backslash*. No caso do flipflop `ff0`, temos que `@` vale 0. Isso significa que `(+ @ 1)` vale 1. Portanto, o fio `d1` está conectado com `ff0`.

A propósito, para desfazer o efeito das AUTO *features*, aperte - e digite `verilog-delete-auto` no M-x prompt.

evince — Para ler arquivos em pdf, Nia utiliza o visualizador *evince*. Assim, para ler este tutorial, Nia abriu-o no *evince*:

```
~/ia$ evince veritedium.pdf &
~/ia$ _
```

O comando `evince veritedium.pdf &` inicia o visualizador *evince*, sem prender o terminal. Já `evince veritedium.pdf` (sem `&`) deixa o terminal por conta da sessão do visualizador.

ls — O comando `ls` lista o conteúdo de uma pasta e tem várias opções. A opção `'ls -l'` fornece as permissões e a data de criação do arquivo.

```
~/ia$ ls -l
total 2100
-rw-r--r-- 1 rosa rosa      234 2017-03-03 15:02 html.el
-rw-r--r-- 1 rosa rosa 2139686 2017-04-06 07:15 lshort.pdf
```

A opção `'ls *.el'` vai listar apenas os arquivos com extensão `'.el'` conforme mostrado no exemplo abaixo.

```
~/ia$ ls *.el
vauto.el
~/ia$ _
```

Pasta home/pessoal – Nia tem dois modos de referir-se à sua pasta pessoal. Ela pode utilizar um `~` til, ou o valor da variável de ambiente `HOME`. Para obter o valor de uma variável de ambiente, basta prefixá-la com o símbolo de sestércio. A propósito, sestércio era o dinheiro romano, simbolizado por `$`. Logo, o valor de `HOME` é `$HOME`.

```
~/ia$ cd ~/Downloads/  
~/Downloads$ cd ~/ia  
~/ia$ cd $HOME/Downloads  
~/Downloads$ cd $HOME/ia  
~/ia$ _
```

cp – Nia quer fazer uma cópia do arquivo `vauto.el`.

```
~/ia$ cp vauto.el tags.el  
~/ia$ ls *.el  
vauto.el  tags.el
```

Renomear – Nia quer renomear o arquivo `tags.el`.

```
~/ia$ mv tags.el features.el  
~/ia$ ls *.el  
features.el  vauto.el
```

Remover – Nia removeu o arquivo `features.el`.

```
~/ia$ rm features.el  
~/ia$ ls *.el  
vauto.el
```

exit – Para fechar o *Emacs*, salve os arquivos modificados com `Ctrl-x` `Ctrl-s`. Em seguida, saia da *shell* com o comando `exit`. Finalmente, mate o *Emacs* com `Ctrl-x` `Ctrl-c`.

```
~/ia$ exit
```

Copiar diretórios — Nia está copiando ~/ia para a pasta atual, representada pelo ponto. Na falta do ponto, a *shell* emite um erro.

```
~$ cd Downloads/
~/Downloads$ cp -r -f ~/ia/ .
~/Downloads$ cd ia
~/Downloads/ia$ ls
vauto.el  vauto.el~  veritedium.pdf
~/Downloads/ia$ _
```

Movendo arquivos — O ponto representa a pasta atual, para onde o movimento está sendo realizado.

```
~/Downloads/ia$ cd ~/Documentos/
~/Documentos$ mv ~/Downloads/ia/veritedium.pdf .
~/Documentos$ ls
veritedium.pdf
~/Documentos$ ls ~/Downloads/ia/
vauto.el  vauto.el~
```

rm -r -f ia — Apaga um diretório inteiro. No exemplo, Nia está apagando o diretório ~/Downloads/ia, que copiou de ~/ia

```
~/Documentos$ cd ~/Downloads/
~/Downloads$ rm -r -f ia
~/Downloads$ ls ia
ls: impossível acessar ia: Diretório não encontrado
```

sudo apt-get update — Depois da instalação do Linux é preciso fazer a *update* (atualização) do *apt-get*. Comandos de gravação nos diretórios de *root* exigem senha para o *sudo* (abreviação inglesa de ‘*superuser do*’).

```
~$ sudo apt-get update
[sudo] password for nia:*****
```

sudo apt-get install emacs – A primeira coisa que Nia fez depois da *update* do `apt-get` foi instalar o `emacs`.

```
~$ sudo apt-get install emacs
[sudo] password for nia: *****
```

Tab – A tecla `Tab` completa um comando. Se há vários jeitos de completar o comando, `Tab` lista todos na tela. Na configuração da `bash`, Nia teclou `emacs ~/.ba` `Tab`, e o computador completou `emacs ~/.bashrc`

Pendrive – Quando Nia insere o *Pendrive* na *USB*, a pasta do dispositivo aparece no diretório `/media/`. Nia acabou de inserir um *pendrive* em uma das portas *USB* da máquina *cdr*. Veja só o que aconteceu:

```
~$ cd /media/ Enter
/media$
```

Agora Nia executa um comando `ls` para verificar se a pasta do *pendrive* está realmente dentro do diretório `/media/`.

```
~$ cd /media/
/media$ ls Enter
cdrom  KINGSTON
```

Nia sempre desmonta o *pendrive* antes de removê-lo do computador. Dessa forma, ela evita perder dados.

```
/media/$ umount KINGSTON/ Enter
```


Capítulo 4

Reverse Engineering

Há duas razões para examinar um *chip* ao microscópio. A primeira é estudar sua estrutura. A segunda razão é descobrir se o circuito não tem nenhuma falha.

A tecnologia que o autor pretende utilizar consiste em fazer um *scanning* do objeto que pretende examinar com uma van der Waals *probe*. A parte mais delicada do processo é a probe propriamente dita e a mecânica fina que faz o scanning. Mostramos abaixo um High-Dynamics Nanoscanner para Probe Microscopy. Esse nanoscanner tem um travel range de $5\text{ }\mu\text{m}$ em cada uma das direções cartesianas.



A eletrônica do microscanner consiste em um Phase Lock Loop Controller, que pode ser comprado ou construído pelo pesquisador. A microscopia resolve o problema de estudar shapes encontrados no circuito.

No caso do material, precisamos de espectroscopia.

No caso de estudar o *chip* projetado e construído por outras pessoas, o primeiro problema que temos é nos livrar do encapsulamento. Não é uma tarefa fácil. Entretanto, o autor desse Trabalho de Conclusão de Curso descobriu como remover as cápsulas de epoxy polimerizada que protegem os circuitos integrados. Para isso, o circuito é colocado em um forno a vácuo a uma temperatura de 400 graus durante 24 horas. O vácuo no interior do forno é necessário para evitar a oxidação dos contatos metálicos.

Depois de várias horas no forno, o polímero do encapsulamento acabou por degradar-se, deixando a pastilha pronta para ser examinada ao microscópio. Na figura 4.1, o circuito pode ser visto com grande aumento.

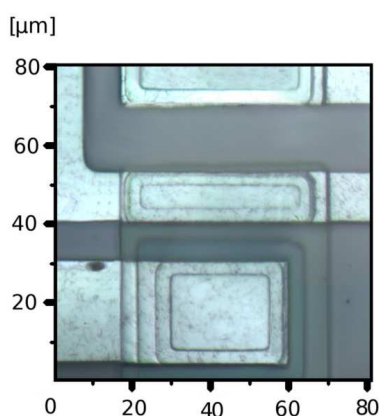


Figura 4.1: Circuito integrado com grande aumento

Na figura 4.2, você pode ver uma imagem em um scanner de força atômica, justamente aquele construído com uma probe de van der Waals e um High-Dynamics Nanoscanner. Na parte rotulada por (a) da imagem, temos uma pequena área de um *Medium Scale Integrated Circuit*. O rótulo (b) mostra um grupo de hemácias nas mesmas dimensões. O autor desse estudo escolheu esse circuito para desmontar, porque o referido

circuito é relativamente barato e tem poucos componentes. A título de comparação, a parte de cima da imagem mostra um grupo de hemácias.

Engenharia e Medicina. Não é por mero capricho que o autor mostra, na figura 4.2, hemácias e circuito eletrônico. Para conseguir fundos que possibilitassem a montagem do laboratório, o grupo de pesquisa o qual o autor desse estudo é integrante e outros interessados, alguns dos quais listados no agradecimento, reuniu recursos de vários programas de pós-graduação. Além disso, o autor prestou serviços de medição, levantamento de imagens e redução de dados para pesquisadores cujas contribuições tornaram possível a aquisição de equipamentos.

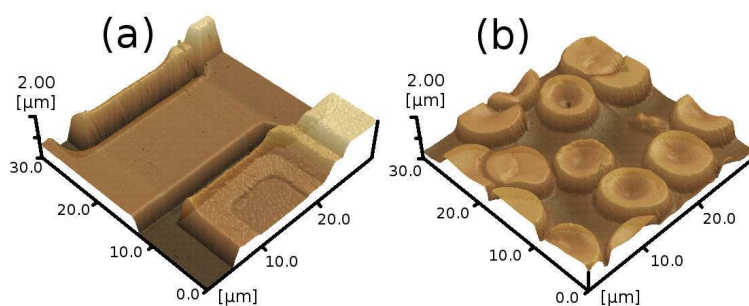


Figura 4.2: Hemácias e componente eletrônico comparados

4.1 Examinando um circuito

A figura 4.3 mostra a imagem de um circuito eletrônico já livre do encapsulamento. Da esquerda para a direita, temos o circuito ao lado de uma moeda para efeito de comparação, o circuito em uma escala de 0 a $1200\mu m$, um zoom para uma escala de 0 a $250\mu m$, e um zoom para 0 a $80\mu m$ em três dimensões.

Determinadas as formas, o autor tentou descobrir um método para levantar os materiais utilizados e as dopagens que transformam esses

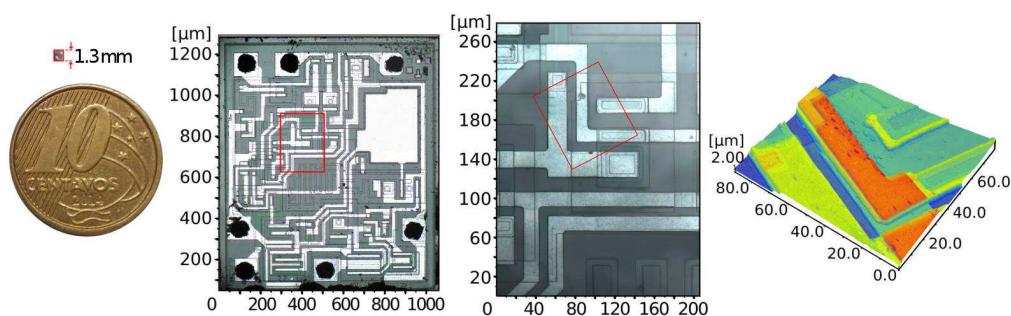
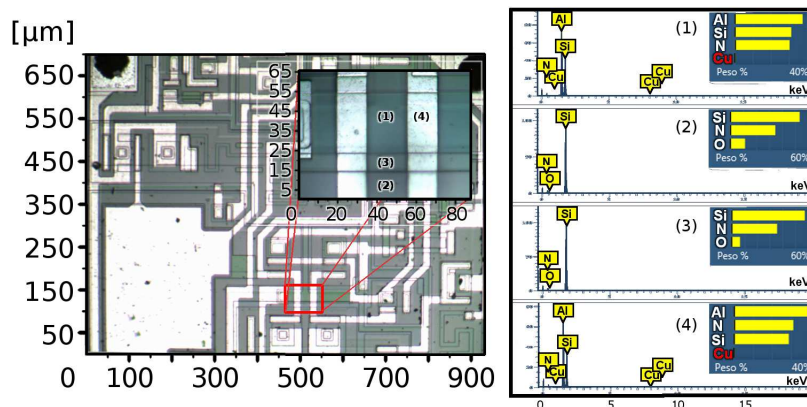


Figura 4.3: Ampliação do circuito eletrônico extraído do encapsulamento

materiais em semicondutores. O autor teve sucesso em encontrar Silicon (silício), Nitrogenium (Nitrogênio) e Aluminium (Alumínio). O silício é a base para a fabricação da maioria dos dispositivos semicondutores e de circuitos integrados. Assim, a presença de Silicon era esperada. O autor ficou espantado com a presença do Nitrogênio. Contudo, consultando a literatura, descobriu que o Nitrogênio é utilizado para aumentar a resistência mecânica do lattice. A presença do Aluminium nas trilhas de silício é, provavelmente, devida a uma unintentional purity, ou seja, alumínio de trilhas metálicas pode ter entrado em shapes de silício.



Os autores não conseguiram detectar dopantes de Phosphorum, Boron ou Arsenium em lugar algum. Há várias explicações para isso. A primeira é que não se procurou suficientemente. De fato, devido à premência de

tempo, uma vez que esse trabalho deveria estar pronto para apreciação da banca, foram realizados poucos experimentos. Outra possibilidade, porém, é que os dopantes estão em níveis mais profundos, não atingidos pela espectroscopia.

4.2 Process Engineer

Para construir um computador ou um componente de computador, precisamos de dois tipos de engenheiros. O primeiro deles é o *Computer Hardware Engineer*, que projeta e testa a máquina por meio de simuladores e tecnologia de baixo custo, como FPGA e PSoC. O *Computer Hardware Engineer* trabalha de terno e gravata, em escritório com ar condicionado ou em casa mesmo. Utiliza poderosos computadores e placas de desenvolvimento para projetar e testar o circuito. O projeto é uma série de desenhos bidimensionais, retângulos, polígonos e trilhas.

Para realmente construir o circuito, precisaremos de um *Process Engineer*, que trabalha coberto da cabeça aos pés por uma roupa especialmente projetada para evitar a contaminação do ambiente por partículas e fluidos gerados pela pele e corpo.

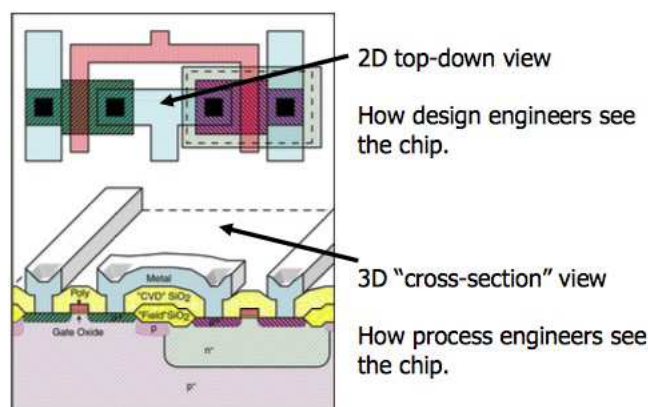


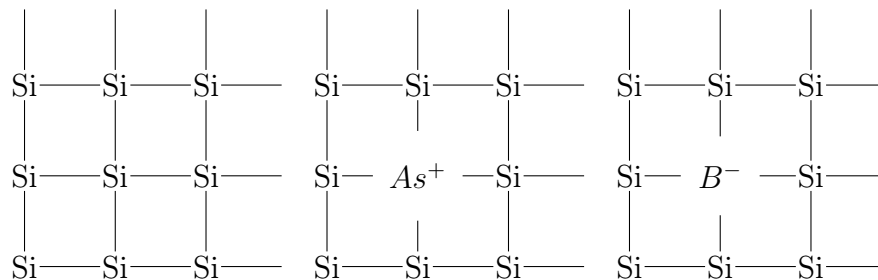
Figura 4.4: Transistor sob dois pontos de vista

Duas maneiras de ver um circuito. Na figura acima, você pode ver como os dois engenheiros veem um circuito integrado. O engenheiro de computação vê o circuito de cima, na forma de vários retângulos sobrepostos. O engenheiro de processos prefere ver os componentes da máquina em cortes transversais.

4.3 Silicon

Silicon (símbolo Si e número atômico 14) é um metalloide isolado em 1823 por Friherre Berzelius. O nome vem do latim SILEX/SILECIS, um tipo de rocha de onde o elemento foi isolado. Em inglês, esse metalloide recebeu o nome *silicon*, por comparação com *carbon* e *boron*.

Silicon é o material de base para a maioria dos circuitos integrados e consiste em um *lattice* tridimensional de átomos.



Dopantes. Como todos os seus electrons de valência estão participando de ligações químicas, silicon puro é um condutor ruim. Entretanto, a condutividade pode ser aumentada pela introdução de dopantes no *lattice*.

Arsenico (As) é um dopante que tem 5 electrons de valência. Quando As substitui um átomo de Si no *lattice*, o quinto electron fica fracamente ligado ao átomo. Vibração térmica basta para liberar o electron, que pode, então, carregar corrente e aumentar a condutividade do *lattice*. Uma região dopada com As é chamada de *nDiff* ($As^+ + \text{free electron}$). De maneira similar, Boron (B), com 3 electrons de valência, forma uma

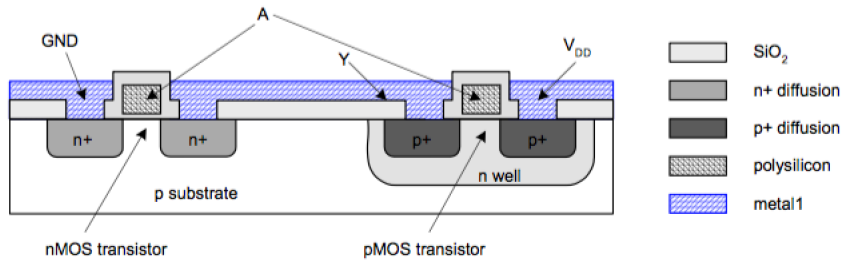


Figura 4.5: nMOS e pMOS transistors

pDiff atraindo um electron do átomo de Si vizinho ($B^- + \text{hole}$), que fica deprimido de um electron. O átomo de Si deprimido, por sua vez, pode pegar emprestado um electron, de forma que a depressão, ou *hole*, propaga-se pelo *lattice*.

4.4 MOS gates

Estruturas de *Metal-Oxide-Semiconductor* são criadas pela superposição de vários *layers* de material condutor e isolante. A tecnologia CMOS fornece dois tipos de transistores: nMOS e pMOS, respectivamente apresentados nos lados esquerdo e direito da figura 4.5. A operação desses transistores é controlada por campos elétricos, de forma que eles são ditos *Metal Oxide Semiconductor Field Effect Transistors* (MOSFET).

Switches. Um transistor tem três terminais: *gate*, *source*, *drain*. A voltagem aplicada à *gate* determina a corrente que passa entre a *source* e o *drain* do transistor. *Switches* estão na base do funcionamento de um computador. De fato, Konrad Zuse, quando construiu o computador Z3 em 1941, utilizou 2100 *relays* de telefonia que atuavam como *switches*. Assim sendo, examinemos como um transistor pode funcionar como *switch*.

O transistor nMOS consiste em n^+ *drain* e n^+ *source* assentadas em um *p-type substrate*. No transistor nMOS, uma voltagem maior do que o

valor V_T de *threshold* na *gate* gera um *conducting channel* entre a *source* e o *drain*, e corrente flui entre os dois terminais. A condutividade do *channel* é controlado pela *gate voltage*. Quando, porém, a *gate voltage* torna-se menor do que o *threshold*, o *channel* desaparece e a *switch* é considerada aberta. A corrente é causada por electrons movendo através de um *n-type channel* entre a *source* e o *drain*.

Já no transistor pMOS, temos um *n-type substrate* onde são inseridas uma p^+ *source* e um p^+ *drain*. No transistor pMOS, a corrente é gerada quando uma *hole* positiva move através do *p-channel*.

Na lógica nMOS, precisamos de dois valores Booleanos, **#t** (*true*) e **#f** (*false*). Podemos representar o valor **#t** por uma voltagem positiva que fecha uma *switch* implementada com um nMOS transistor. Com a *switch* fechada, uma corrente passa pelo transistor, da *source* ao *gate*. O valor **#f**, no caso de transistores nMOS, pode ser representado por uma voltagem zero ou negativa, que abre a *switch* e corta a corrente.

Uma voltagem negativa ou nula entre *gate* e *source* fecha uma pMOS *switch*, deixando a corrente passar. Uma voltagem positiva abre a pMOS *switch*, impedindo a corrente de passar.

4.5 Tabelas e sequenciadores

Um computador precisa de dois tipos de circuitos, um para fazer operações fundamentais e outro para construir sequências de operações.

Suponha que você queira fazer uma multiplicação de dois números de vários dígitos. As operações fundamentais, produto e soma de dígitos, são dadas por tabelas, também conhecidas como taboadas de mais e de vezes. Além das tabelas, o calculista precisa das regras de multiplicação, que permitem construir sequências dos estados do cálculo até chegar ao resultado final. Para resumir, o calculista precisa de tabelas e sequenciadores de operações.

Os indianos inventaram um sistema de numeração que faz uso dos

dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9 para fazer cálculos. Nesse sistema, um inteiro positivo é expandido em uma soma de produtos, onde a posição do dígito mostra a potência de 10 pela qual deve ser multiplicado. Por exemplo, 1042 indica a soma $1 \times 10^3 + 0 \times 10^2 + 4 \times 10^1 + 2 \times 10^0$ de produtos, onde 10^0 é igual a 1 e 10^1 vale 10. Dizemos que os indianos usam a base 10 para construir numerais.

Na base 10, precisamos de enormes taboadas para fazer adições e multiplicações. Para facilitar a operação dos computadores, engenheiros como Konrad Zuse e Janos von Neumann decidiram usar apenas dois dígitos para representar números. Nesse esquema, os inteiros positivos são expandidos em somas de potências de 2. Assim, 1042 é representado pelo numeral 10000010010, que indica a soma $2^{10} + 2^4 + 2^1$.

Com apenas dois dígitos, as tabelas ou taboadas ficam muito menores. Considere, por exemplo, a tabela de adição.

A	B	vai 1	unidade
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Vamos chamar os dígitos que estão sendo somados de A e B. A quarta linha da tabela indica que $1+1$ produz 10, ou seja, 0 unidades e vai 1 para a coluna seguinte da soma. A segunda linha indica que $0+1$ é 1. Da mesma forma, a terceira linha mostra que $1+0$ é 1. Finalmente, a primeira linha estabelece que $0+0$ é 0.

Xor. Vamos construir uma tabela apenas das unidades da adição, sem nos preocupar com o dígito que é transportado para a coluna seguinte de uma soma de vários dígitos.

A	B	unidade
0	0	0
0	1	1
1	0	1
1	1	0

Gates. Vamos tentar obter essa tabela de adição pela combinação de três tipos de dispositivos conhecidos como portas lógicas (*gates*).

Desde os primórdios da Computação, Konrad Zuse, Janos von Neuman, Jack Kilby e Victor Glushkov tentaram construir circuitos capazes de implementar funções como (**and** *x y*), (**or** *x y*) e (**not** *x*). Em inglês, esses circuitos receberam o nome de *gates*, *ports* ou *procedures*.

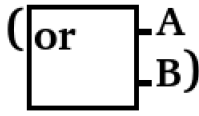
Em diagramas, um *gate* é representado por uma caixa rotulada com um identificador. O gate tem *inputs* ou argumentos e uma *output*, que é o resultado da tabela.

A figura abaixo mostra a representação de um **and** gate.



```
(and #f #f) → #f
(and #f #t) → #f
(and #t #f) → #f
(and #t #t) → #t
```

Para descrever o mapeamento dos argumentos de um predicado como (**and** *A B*) ao resultado, podemos utilizar uma tabela da verdade. No caso de dois argumentos, a tabela terá 4 linhas. Em geral, com um predicado de *n* argumentos, o número de linhas é dado por 2^n . O fato da tabela ter um número de linhas finito e conhecido, leva ao costume de fornecer apenas as linhas de resultado **#t**, deixando subentendido que todas as linhas não expressas têm resultado **#f**. Já dentro dessa convenção, mostramos abaixo a caixa e a tabela do predicado (**or** *A B*).



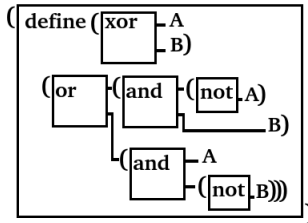
$$(\text{or } \#f \ \#t) \rightarrow \#t$$

$$(\text{or } \#t \ \#f) \rightarrow \#t$$

$$(\text{or } \#t \ \#t) \rightarrow \#t$$

Inversor, ou NOT-*gate*, é um circuito de uma entrada só que produz $\#t$ quando a entrada é $\#f$ e gera o resultado $\#f$ quando a entrada é $\#t$.

Definições. Podemos definir novos predicados escrevendo um circuito que combina *procedures* e *gates* primitivos ou definidos anteriormente. Mostramos abaixo a definição e a tabela da verdade de um $(\text{xor } A \ B)$ *gate*, que nada mais é do que a taboada de soma.



$$(\text{xor } \#f \ \#t) \rightarrow \#t$$

$$(\text{xor } \#t \ \#f) \rightarrow \#t$$

A definição tem dois componentes: a caixa $(\text{xor } A \ B)$, que mostra como o novo predicado será aplicado e a definição propriamente dita.

4.6 Stick diagrams

Podemos implementar um NOT *gate* com um único transistor nMOS. Uma voltagem alta representa tanto o dígito 1 quanto o valor booleano $\#t$. Por outro lado, voltagem nula representa o dígito 0 ou o valor booleano $\#f$. No figura acima, quando v_{in} é 1 (voltagem alta), o transistor conecta v_{out} ao terra, produzindo o valor 0. Quando a voltagem é 0, o transistor age como uma chave aberta, e v_{out} fica conectada à fonte de tensão V_S , que equivale ao dígito 1.

Complementary CMOS circuits são circuitos estáticos onde, em qualquer ponto do tempo, cada *gate output* está ligado por baixa resistên-

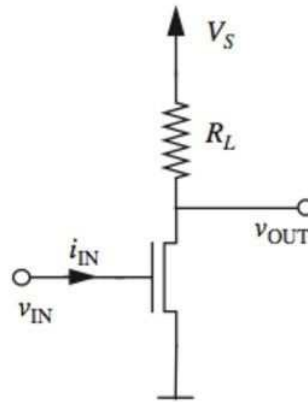


Figura 4.6: Inversor com um transistor

cia ou com V_{DD} ou com V_{ss} . Além disso, desconsiderando efeitos transitórios devido a chaveamento, as *outputs* sempre expressam os resultados da função Booleana que as *gates* correspondentes implementam. Esse comportamento contrasta com circuitos dinâmicos, que dependem de armazenamento temporário de valores de sinais em capacitores de nodos de alta impedância. Quando alguém perguntar o que é memória dinâmica ou qualquer outro circuito dinâmico, você já sabe o que responder.

PUN e PDN. Uma *static CMOS gate* é uma combinação de duas *networks*: *pull-up network* (PUN) e *pull down network* (PDN). A figura 4.7 mostra uma *gate* genérica onde todas as entradas estão igualmente distribuídas para as *pull-up* e *pull-down networks*. A função da PUN é providenciar a conexão entre a *output* e V_{DD} sempre que as *inputs* sinalizarem que a *gate* deve ser 1. Por outro lado, a PDN deve conectar a *output* quando as *inputs* indicarem *output* igual a 0, de acordo com a função lógica. PUN e PDN são mutuamente exclusivas, de modo que uma e apenas uma das *networks* está conectada em *steady state*. Dessa maneira, uma vez que os fenômenos transitórios desapareceram, uma *path* sempre existe entre V_{DD} e a *output* f para possibilitar a *output* 1. Ana-

logamente, uma *path* alternativa existe entre V_{SS} e f para providenciar a *output* 0.

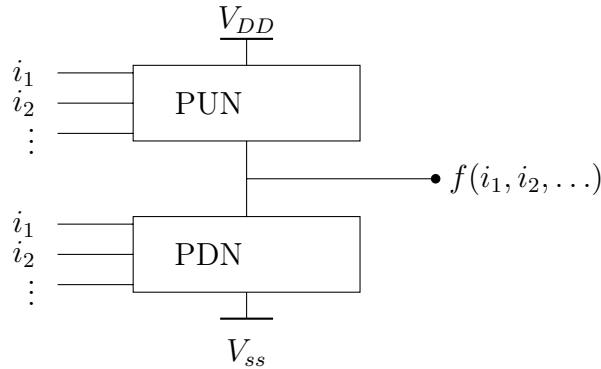


Figura 4.7: Pull-up/Pull-down Network

4.7 Considerações gerais sobre PUN e PDN.

Evidentemente, a lógica CMOS é mais cara e produz circuitos mais complexos, acho. Do lado positivo, esse tipo de tecnologia tem alta imunidade ao ruído e baixo consumo de energia em regime estático. Em consequência, dispositivos CMOS não produzem tanto calor quanto outras formas de lógica. Além disso, CMOS permite alta densidade de funções lógicas em um chip, o que talvez seja a razão principal para CMOS ser hoje a tecnologia mais utilizada na implementação de circuitos VLSI.

Alguns *hints* para quando você precisar de construir PDN e PUN *networks*:

- Um transistor pode operar como uma *switch* controlada pelo sinal de *gate*. Uma *nMOS switch* está *on* quando o sinal de controle está *high* e está *off*, quando está *low*. Já um *pMOS transistor*, está *on* quando o sinal de controle está *low* e *off* quando está *high*.

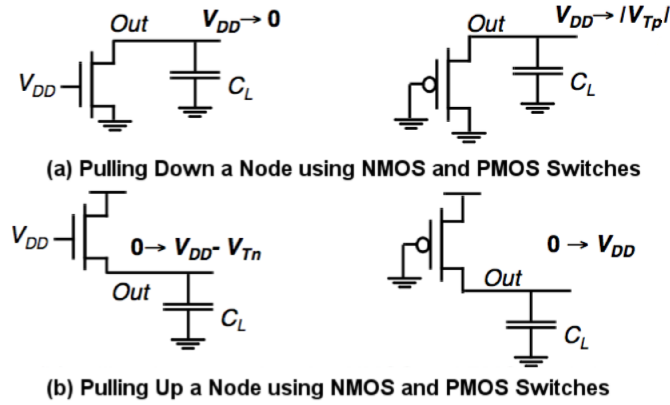


Figura 4.8: Strong zeros and strong ones

- A PDN é construída usando-se *nMOS devices*, pois esse tipo de transistor produz *strong zeros*. Considere a figura 4.8 a. O capacitor parasita está inicialmente carregado em V_{DD} . Admita que quero levar o valor para 0. Um nMOS leva a *output* para a *ground*, mas uma pMOS, se fosse utilizada na PDN, levaria apenas até V_{Tp} , o ponto de desligamento de dispositivos pMOS.
- *pMOS transistors* são utilizados na montagem da PUN, pois *pMOS devices* garantem *strong ones*. A figura 4.8 b ilustra que um transistor nMOS aplicaria, no capacitor parasita, a voltagem $V_{DD} - V_{Tn}$ para produzir 1, enquanto um dispositivo pMOS aplicaria praticamente V_{DD} inteira.

A figura 4.10 mostra o diagrama de um *static CMOS inverter* construído com dois transistores. A operação desse circuito pode ser compreendida através do modelo de chave do transistor MOS: o transistor nada mais é do que uma chave com *off-resistance* infinita, e *on-resistance* finita. Isso leva à seguinte interpretação do *inverter*: Quando V_{in} é alta e igual a V_{DD} , o transistor nMOS está *on* (R_n), enquanto o transistor pMOS está *off*. Isto pode ser visto na figura 4.9(a), que contém o diagrama elétrico equivalente a esta situação. Uma trajetória direta existe entre V_{out} e a terra, resultando em um valor próximo de zero.

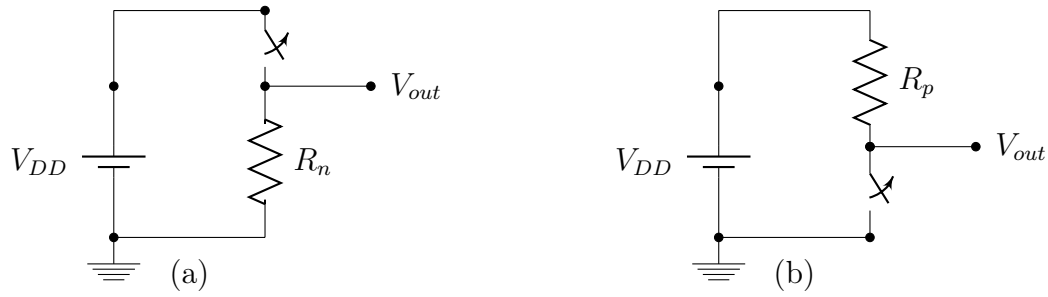


Figura 4.9: Static CMOS inverter

Por outro lado, quando a voltagem de entrada é baixa, os transistores nMOS e pMOS estão *off* e *on* (R_p), respectivamente. O circuito equivalente da figura 4.9 apresenta uma trajetória entre V_{DD} e V_{out} , produzindo uma voltagem de saída alta. A *gate* funciona claramente como um *inverter*.

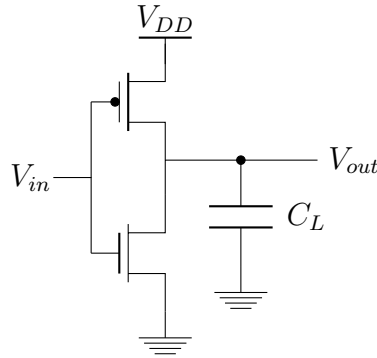


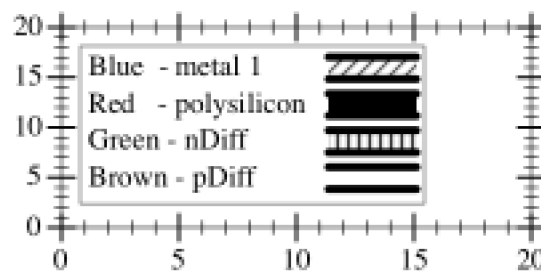
Figura 4.10: CMOS inverter com capacitor parasita

A figura 4.10 mostra que a tecnologia CMOS implementa *gates* com PUN e PDN, porque essa tecnologia providencia boas descargas para capacitâncias parasitas, tanto nas transições de $\#f \rightarrow \#t$ e de $\#t \rightarrow \#f$. Essa propriedade não é mostrada no circuito de um transistor da figura 4.6.

A figura 4.10 mostra que o nMOS transistor é representado por duas linhas paralelas, uma um pouco menor do que a outra. O pMOS tran-

sistor também é representado por duas linhas paralelas, mas há uma bolinha no ponto em que o *gate* entra em contato com o transistor.

Na construção de circuitos, precisamos basicamente de quatro tipos de material: polysilício (poly), silício com difusão de arsênico com excesso de elétrons (nDiff), silício com difusão de boro com falta de elétrons (pDiff) e metal. Eis como indicar esses materiais:



Stick diagram é uma vista aérea dos componentes, como mostrado na figura 4.4, página 37. A trilha de polysilicon cobre completamente o intervalo de *p substrate* entre a *source* e o *drain* do nMOS.

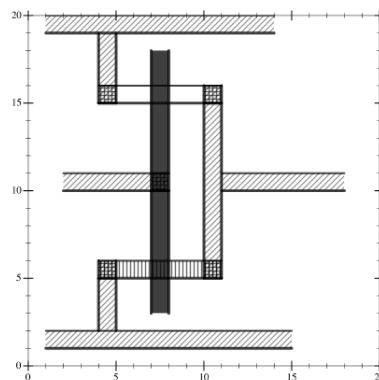


Figura 4.11: Stick diagram de um inversor

No caso do transistor pMOS, representamos a trilha da *source* e do *drain* por um retângulo sem hachuramento. Entretanto, não representamos a bacia de nDiff para não tornar o diagrama muito sobrecarregado.

O *process engineer*, quando encontrar polysilicon cortando trilha p, compreenderá que o transistor está dentro de uma bacia nDiff.

A figura 4.11 mostra um stick diagram de um inversor, como o mostrado na figura 4.10. Em um stick diagram, como o da figura 4.11, um transistor é representado apenas como o cruzamento de polysilicon com trilhas de difusão P ou difusão N.

AND/OR. Para completar nosso somador, faltam apenas dois circuitos, AND e OR, que são mostrados na figura 4.12, página 50. A figura 4.13, página 50, mostra o stick diagram de um circuito nand.

É interessante notar que circuitos NAND e NOR são de implementação muito simples. Por isso, o circuito AND é, em geral, implementado como um NAND seguido de um inversor (ver a parte de baixo da figura 4.12). Analogamente, o circuito OR é um NOR seguido de um inversor, como mostrado na parte de cima da figura 4.12. De fato, as portas NAND e NOR são tão fáceis de implementar que muitos engenheiros defendem que ASICs sejam construídos somente com portas NOR. Para isso, surgiram técnicas bem simples de gerar circuitos combinatórios complexos com portas NOR ou NAND. Uma dessas técnicas é denominada Weinberger arrays.

4.8 Stick diagram de uma porta lógica

O leitor deve ter desconfiado que o stick diagram ajuda o engenheiro a planejar rapidamente as shapes e trilhas que formam um circuito integrado. Um stick diagram não precisa estar em escala e pode ser traçado até com lápis de cor.

Para traçar o layout com finalidade de fabricar o circuito, é necessário traçar shapes de forma mais precisa. Na figura 4.13, mostramos as máscaras de um circuito NAND.

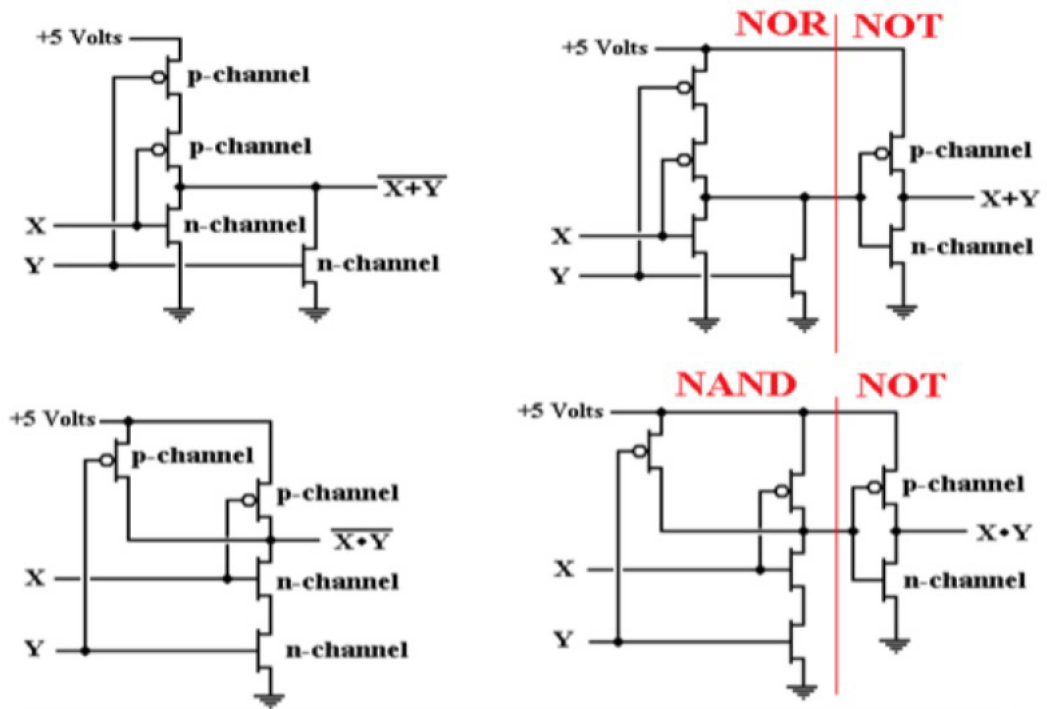


Figura 4.12: Circuitos and e or

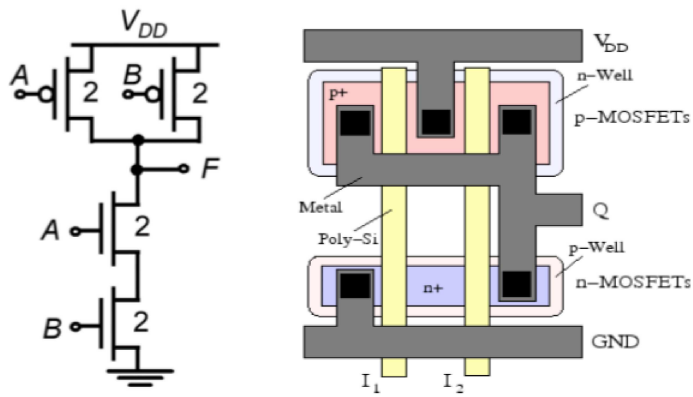


Figura 4.13: Circuito nand

Referências Bibliográficas

- [1] Marquis de Condorcet Essai de L'analyse à la probabilité des décisions. Paris, Imprimerie Royale.
- [2] Briscoe, T., Buttery, P., Carroll, J., Medlock, B., Watson, R. Open Source RASP 3.1 – Robust Parsing System for English. <http://ilexir.co.uk>
- [3] Nielsen, Jakob Information Pollution. Available in <http://www.nngroup.com/articles/information-pollution/> on August 11, 2003. Accessed at July 03, 2015.
- [4] Garside, R. The CLAWS Word-tagging System. In: R. Garside, G. Leech and G. Sampson (eds), The Computational Analysis of English: A Corpus-based Approach. 1987. London: Longman. <http://ucrel.lancs.ac.uk/claws/>
- [5] Tarantola, Albert Popper, Bayes and the inverse problem. Nature Physics, vol. 2 pp.492-494, 2006.
- [6] Koza, John. Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press, 1992.
- [7] List, Christian, GODIN, Robert E. Epistemic Democracy: Generalizing the Condorcet Jury Theorem. Journal of Political Philosophy, vol. 9, jan. 2001.

- [8] Wright, Benjamin D., MOK, Magdalena M. C.. An Overview of the Family of Rasch Measurement Models. In Introduction to Rasch Measurement, pp.1-24, 2004.
- [9] Seibel, Peter Practical Common Lisp. Apress, First Edition, 2005.
- [10] computer-and-hardware-engineers.htm, Bureau of Labor Statistics. Consulted in 18 May 2017.
<https://www.bls.gov/ooh/architecture-and-engineering>
- [11] Steel Bank Common Lisp. Available in www.sbcl.org.
- [12] Beane, Zach. quicklisp Beta. Available in www.quicklisp.org
- [13] SLIME: The Superior Lisp Interaction Mode for Emacs. Available from <https://common-lisp.net/project/slime/>
- [14] Fernando C. N. Pereira and Stuard M. Shieber. Prolog and Natural-Language Analysis.
- [15] Peter Norvig. Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp.
- [16] Stonebank, M. UNIX Tutorial for Beginners.
<http://www.ee.surrey.ac.uk/Teaching/Unix/>
- [17] Mark Kantrowitz. Infix. <http://www.cliki.net/infix>
- [18] Doug Hoyte. Let Over Lambda. Hoytech, 2008. ISBN: 978-1-4357-1275-1
- [19] A. Church, A set of postulates for the foundation of logic, Annals of Mathematics, Series 2, 33:346–366 (1932).
- [20] Alonzo Church. The Calculi of Lambda Conversion. Princeton University Press, 1986.

- [21] Arlo J. Miller; Martin C. Mihm Junior. Melanoma. The New England Journal of Medicine, Massachusetts, v. 355, p. 51-65, 2006.
- [22] Crayston Toby Charles William. Textrazor.
<https://www.textrazor.com>
- [23] Eric Schulte. Neutral Networks of Real-World Programs and their Application to Automated Software Evolution. University of New Mexico.
<https://github.com/eschulte/netgear-repair>
Retrieved on April, 13, 2017.
- [24] Rod Canion. Open: How Compaq Ended IBM's Domination and Helped Invent Modern Computing. Ben Bella Books, Inc. 2013. ISBN 978-1-936661-92-3
- [25] Boris Chertok. Rockets and People. There are three English editions of this book. Progressive Management has a carefully typeset edition for epub ereaders. NASA published a pdf edition, on which the Progressive Management edition is based. Finally, there is also a Kindle edition that is almost illegible due to layout and OCR errors. ISBN 0-16-073239-5.
- [26] Dantas, N. O., Fernandes, G. L., Anielle, C. A. Controlling the growth of ultrasmall CdTe quantum dots and the diffusion of cadmium vacancies: Thermal annealing. Journal of Alloys and Compounds, v. 637, p. 466-470, 2015.
- [27] Dantas, N. O., Fernandes, G. L., Baffa, O. Controlling Densities of Manganese Ions and Cadmium Vacancies in Cd 1-x Mn x Te Ultrasmall Quantum Dots in a Glass Matrix: x-Concentration and Thermal Annealing. Journal of Physical Chemistry, v. 119, p. 17416-17420, 2015.

- [28] Dantal, N. O., de Lima Fernandes, G. Cd_{1-x}Mn_xTe ultrasmall quantum dots growth in a silicate glass matrix by the fusion method. Applied Physics Letters, v. 105, p. 13410, 2014.
- [29] Victor Klimov. Nanocrystal quantum dots. Taylor & Francis Group, 2010.