UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

LUCAS MACHADO

Estudo e Implementação de Lógica Adiabática para Circuitos Integrados com Baixo Consumo

Trabalho de Diplomação.

Prof. Dr. Renato Perez Ribas
Orientador

Porto Alegre, junho de 2010

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Machado, Lucas

Estudo e Implementação de Lógica Adiabática para Circuitos Integrados com Baixo Consumo / Lucas Machado. – Porto Alegre: UFRGS, 2010.

80 f.: il.

Trabalho de Conclusão – Universidade Federal do Rio Grande do Sul. Curso de Engenharia de Computação, Porto Alegre, BR–RS, 2010. Orientador: Renato Perez Ribas.

1. Lógica adiabática. 2. Circuitos integrados. 3. Baixo consumo. 4. Microeletrônica. I. Ribas, Renato Perez. II. Título.

Para meus pais...

AGRADECIMENTOS

Agradeço a meus pais Carson e Gisele por terem me incentivado e me ajudado em tudo, possibilitando fazer a faculdade, intercâmbio e tudo que esteve ao alcance. Agradeço a meu irmão Jonas por ser um cara parceiro que sempre me ajudou da forma que pode. Agradeço a todos meus familiares, que me apoiaram e me incentivaram. Agradeço a minha musa Rafaela, que me deu muita inspiração e alegria e teve paciência para entender os dias longe. Agradeço a meus colegas e amigos que muitas vezes foram fonte de conhecimento, mas na maior parte do tempo foram companheiros e parceiros para tudo. E agradeço a todos meus professores que foram grandes responsáveis pelo meu conhecimento e fontes de bons conselhos.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

MOS	Metal Oxyde Semiconductor
PMOS	Positive Metal-Oxyde Semiconductor
NMOS	Negative Metal-Oxyde Semiconductor
CMOS	Complementary Metal-Oxyde Semiconductor
CI	Circuito Integrado
PAL	Pass-Transistor Adiabatic Logic
CAL	Clocked-CMOS adiabatic logic
ECRL	Efficient Charge Recovery Logic
PFAL	Positive Feedback Adiabatic Logic
TSEL	True Single-Phase Energy-Recovering Logic
SCAL	Single-Phase Source-Coupled Adiabatic Logic
SCRL	Split Charge Recovery Logic
VDD	Tensão do circuito
GND	Terra do circuito
PTM	Predictive Technology Model
ASU	Arizona State University
CLA	Carry Look-ahead Adder
SRAM	Static Random-Access Memory
ASB	Adiabatic Super Buffer
OID	Intelligent Output Driver
MOSIS	Metal Oxide Semiconductor Implementation Service
TSMC	Taiwan Semiconductor Manufacturing Company

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho está inserido no ramo de microeletrônica, mais especificamente em circuitos integrados de baixo consumo. Trata-se de um campo importante visando diminuir o consumo de energia mundial, possibilitar a utilização de equipamentos com fonte de baterias por mais tempo e diminuir o custo de resfriamento de circuitos, além do consumo de energia do circuito propriamente dito. Existem diversos tipos de lógica que implementam circuitos integrados, sendo a lógica CMOS a mais utilizada por sua robustez, desempenho e eficiência. O tema do trabalho é a lógica adiabática, que tenta reaproveitar a energia descarregada do circuito, diminuindo o consumo. Há outras formas de conseguir diminuir o consumo do circuito como diminuir a tensão de alimentação, mas isso leva a diminuição de desempenho. Foram desenvolvidas nas últimas décadas diversas formas de implementação da lógica adiabática, criando-se várias famílias dessa lógica. Juntamente com essas famílias foram implementados circuitos de maior complexidade para comparação de consumo, sempre com vantagens em relação ao CMOS na tecnologia da época. Além disso, existe uma empresa em Cambridge que desenvolve *buffers*, dispositivo que armazena dados temporariamente e muito necessário em circuitos integrados, utilizando lógica adiabática e reduzindo em muito o consumo. Utilizando o simulador elétrico SpiceOpus, com modelos de processo de fabricação de transistores PTM, foram implementadas nesse trabalho as famílias lógicas adiabáticas estudadas avaliando o funcionamento e o consumo de um inversor. Faz-se também um exercício avaliando a possibilidade de criação de leiautes para uma biblioteca de células utilizando-se leiaute simbólico para representação das famílias lógicas estudadas. Por fim compara-se um somador completo entre um circuito CMOS, um circuito funcionando em nível sub-limiar, ou *subthresold*, e outro adiabático e analisa-se os resultados.

Palavras-chave: Lógica adiabática, circuitos integrados, baixo consumo, microeletrônica.

Study and Implementation of Adiabatic Logic for Low-power Integrated Circuits

ABSTRACT

This work is inserted in the field of microelectronics, more specifically, low-power integrated circuits. It is an important branch that tries to decrease the world power consumption, enable the use of battery sourced gadgets for a longer time and decrease the costs of cooling circuits. There are several types of logic that implement integrated circuits, but CMOS logis is the most used for its strength, performance and efficiency. This work's theme is adiabatic logic, which tries to recover the discharged energy of the circuit, decreasing power consumption. There are other ways to decrease power consumption in an integrated circuit like decrease the voltage, but it leads to performance loss. It has been developed in last decades several ways to implement adiabatic logic, creating lots of adiabatic logic families. Also it was developed more complex circuits for comparisons, always with several advantages over CMOS. Besides, there is an enterprise in Cambridge called Adiabatic Logic, which develops adiabatic logic buffers, decreasing a lot the power consumption. Using the electric simulator SpiceOpus and PTM transistor manufacturing process models, the studied adiabatic logic families are implemented and evaluated in this work the operation and power consumption of an inverter. There is also an evaluation of the possibility of layout creation for a standard cell library using symbolic layout for representation of the studied adiabatic logic families. In the end, there is a comparison of a full adder implementation between a CMOS circuit, a subthreshold operating CMOS circuit and an adiabatic circuit, analysing the results.

Keywords: Adiabatic logic, integrated circuits, low-power, microelectronics.

1 INTRODUÇÃO

Nesse Capítulo será mostrado onde esse trabalho se insere, qual a motivação e objetivos, além de uma introdução à redes de transistores e à lógica adiabática.

1.1 Contexto

No ramo da microeletrônica, um circuito integrado é um circuito eletrônico miniaturizado, fabricado com semicondutores (normalmente silício), que pode efetuar uma função simples (um multiplicador de matrizes) até uma função mais complexa (um processador de um computador pessoal). E não há como negar que os circuitos integrados, os chips, entraram na vida de todos seres humanos deste planeta. Desde o celular, o carro, a televisão, tocadores de música, até dispositivos de aplicação médica. Até mesmo no meio rural já existem equipamentos para medir a quantidade de nutrientes no solo, produzindo um alimento melhor e mais barato. Isso sem falar no computador, item indispensável tanto para o trabalho quanto para tarefas pessoais. O advento da Internet nos colocou em um mundo que gira em torno da rede mundial de computadores. Um exemplo de um provedor de Internet, fonte de informações buscadas todos os dias, pode ser visto na Figura 1.1.

Figura 1.1: Um provedor de Internet consome muita energia elétrica, inclusive para mantê-lo frio o suficiente para funcionar corretamente.

Um circuito integrado, por mais simples que seja, gasta energia elétrica. E não é pouco. Estamos cercados por transistores carregando, descarregando e gastando energia elétrica. O gasto de energia por transistor vem diminuindo nas últimas décadas devido a diminuição do tamanho da tecnologia, mas, ao mesmo tempo, a densidade de transistores e a quantidade de circuitos aumentaram absurdamente, gerando um gasto de energia considerável. Logo, apesar de ter diminuído o consumo por transistor, o consumo por circuito continua a aumentar e produzir circuitos que tenham menos consumo é um ramo

de grande importância e interesse. Isso vai de encontro a ideias atuais conhecidas como verdes, por exemplo. Um consumo menor de energia elétrica reduziria a necessidade de produção de energia elétrica não-renovável, como termoelétricas e usinas nucleares, diminuindo a produção de lixo nuclear e de monóxido de carbono no ar, contribuindo para a diminuição do aquecimento global, como visto em (COX et al., 2000). Ter um menor consumo também tem um nicho comercial importante: produzir um celular que seja recarregado somente uma vez por mês ou um marca-passos que dure até cem anos funcionando corretamente, sem ter a necessidade de serem recarregadas suas baterias, certamente teriam um valor de mercado excelente. E a quantidade de dispositivos com circuitos integrados só tende a crescer. Da máquina de lavar roupas e a geladeira de casa até o air-bag e o controle de faróis do carro. Onde for possível colocar um computador para fazer o trabalho do ser humano, tornando a vida mais fácil, eficiente e segura, será feito. E deve ser feito. A tecnologia está sendo desenvolvida para melhorar a qualidade de vida das pessoas. Sabendo que a população que utiliza equipamentos eletrônicos tende a crescer geometricamente nos próximos anos com o barateamento dos circuitos eletrônicos, o crescimento de gastos de energia elétrica também irá crescer. Sendo assim, poderemos ter um problema de crise energética se não tivermos um crescimento na geração igual ou maior ao crescimento no consumo de energia elétrica. Sabendo que os recursos são limitados, temos que nos preocupar em criar circuitos integrados mais eficientes. Além do problema do consumo de energia propriamente dito, existe o problema de aquecimento dos circuitos integrados, que acontece exatamente por causa do consumo: os transistores são carregados e descarregados, produzindo calor cada vez que isso acontece, devido ao efeito Joule. E a temperatura muito elevada pode levar ao mal-funcionamento do circuito ou até mesmo queimá-lo. Isso pode impossibilitar a utilização de um circuito integrado em um lugar que não haja dissipação desse calor através de dissipadores e ventoinhas. Além disso, o gasto para alimentar a ventoinha pode ser maior que o gasto do circuito integrado em si, sendo mais um fator de importante relevância.

1.2 Motivação e objetivos

Nesse contexto, a possibilidade de se usufruir de tudo que a tecnologia da informação pode nos oferecer e ainda ter um gasto de energia elétrica praticamente zero, torna-se uma ideia bem interessante. Se pudéssemos reaproveitar a energia de descarga dos capacitores, por exemplo, o gasto do circuito se resumiria a dissipações térmicas menores e o gasto estático devido a correntes de fuga. Hoje existem diversos estilos lógicos para a concepção de circuitos integrados digitais (RABAEY; CHANDRAKASAN; NIKOLIĆ, 1996), mas todos tem um compromisso entre desempenho e consumo. Circuitos de pré-carga, por exemplo, tem uma velocidade alta e um consumo também alto. Esse tipo de circuito se vale de carregar a saída do circuito sempre, e então dependendo se o resultado da lógica é zero, descarregar a saída, o que em geral é mais rápido devido a resistência do circuito de descarga ser menor. Existem circuitos que operam em nível sub-limiar (*subthreshold*), que tem um consumo baixo e um desempenho igualmente baixo. A idéia é trabalhar com uma tensão mais baixa, tendo-se assim um consumo menor. Existe mercado para esse tipo de lógica, onde o foco não é desempenho por não necessitar disso. A motivação para esse trabalho é estudar um estilo lógico CMOS alternativo, que consiga aliar baixo consumo e consiga manter desempenho. Na ideia inicial do trabalho, a lógica adiabática (DENKER, 1994). Um exemplo de implementação de uma porta lógica inversora utilizando lógica adiabática pode ser visto na Figura 1.2.

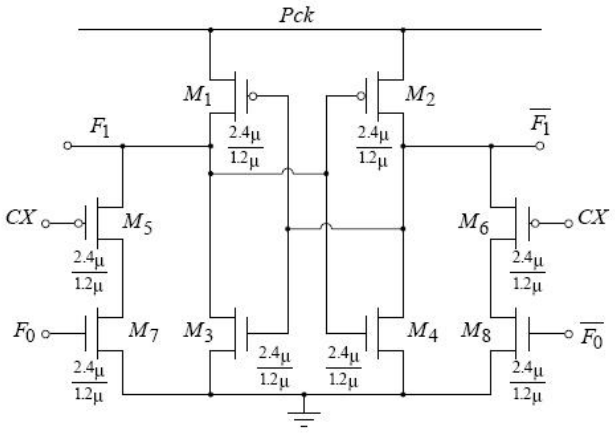

Figura 1.2: Inversor utilizando lógica adiabática - CAL.
(MAKSIMOVIC et al., 2000)

A lógica adiabática tem como idéia base reaproveitar a energia elétrica que é descarregada das capacitâncias no momento em que a saída do circuito é levado a zero, que normalmente é desperdiçada em lógicas CMOS convencionais, simplesmente gerando calor. Esse tipo de lógica é relevante, visto que existem vários estudos atuais (ANUAR; TAKAHASHI; SEKINE, 2009), (PENGJUN; KUNPENG; FENGNA, 2009) e (SU et al., 2010), além de ter interesse comercial de empresas, como (LOGIC, 2010) que desenvolve essa tecnologia. O objetivo principal do trabalho é compreender a utilização e funcionamento da lógica adiabática, verificando complexidade, aplicabilidade e eficácia de tal técnica em relação a circuitos CMOS convencionais e alternativos. Objetivo secundário é projetar portas lógicas e blocos funcionais para testar e avaliar o uso de lógica adiabática em termos de consumo, desempenho e área, utilizando programas de simulação elétrica. Por fim, com os resultados obtidos, o objetivo é comparar com outras lógicas existentes e avaliar em termos de gasto de energia elétrica, velocidade e área, identificando as melhores lógicas para determinadas aplicações. Como meta é estudar a fundo esse estilo lógico, tendo conhecimento suficiente para criar portas lógicas nesse estilo e avaliar se é algo que deve ser investido mais estudo ou não.

1.3 Estilos lógicos digitais

Existem diversos estilos lógicos que podem implementar um circuito integrado, todos com suas vantagens e desvantagens, mas com sua importância histórica. Em uma ordem cronológica, cita-se alguns desses estilos lógicos.

1.3.1 Circuitos PMOS

Um dos primeiros estilos lógicos utilizando transistores MOS (acrônimo de metal-óxido-semicondutor), esse estilo se vale somente de transistores MOS positivos, ou seja, transistor com base em um semicondutor negativo (com mais elétrons) que tem a região ativa dopada de forma a ficar mais positiva (com mais espaços vazios). No caso do silício, normalmente se dopa a região ativa com boro ou alumínio para torná-la mais positiva. Se utiliza ainda dióxido de silício como isolante e o polissilício, substituindo o metal, para o contato de comporta (*gate*), que controla o contato entre a fonte e dreno. Na Figura 1.3 mostra-se um transistor PMOS em três dimensões e o símbolo respectivo utilizado

nos esquemáticos. O tamanho da tecnologia de fabricação (como o que vai ser utilizado nesse trabalho, de 45nm) significa o tamanho mínimo das dimensões de comprimento (L) e largura (W).

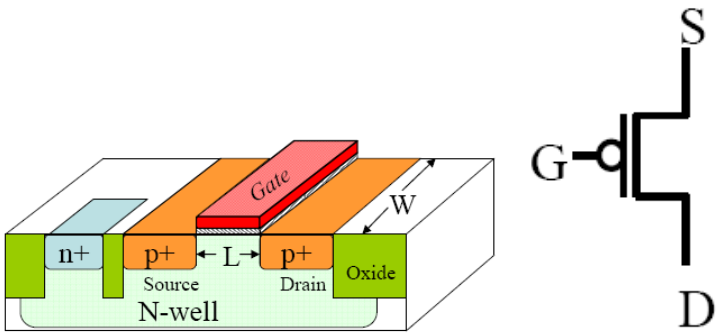

Figura 1.3: Transistor PMOS e o símbolo utilizado para representação.
(RABAEY; CHANDRAKASAN; NIKOLIĆ, 1996)

Para fazer portas lógicas com esses transistores basicamente se utiliza um deles sempre ligado (com o *gate* ligado no terra), com a fonte (*source*) no terra e o dreno (*drain*) na saída e a lógica propriamente dita é feita utilizando uma árvore de transistores PMOS ligada na saída e na fonte (tensão do sistema). Há ainda o contato do substrato (*bulk*), que precisa ser ligado na tensão do sistema para que não altere a tensão de limiar (*threshold*) do PMOS, devido ao efeito de corpo (*body effect*). Para fazer um inversor, por exemplo, podemos fazer como na Figura 1.4.

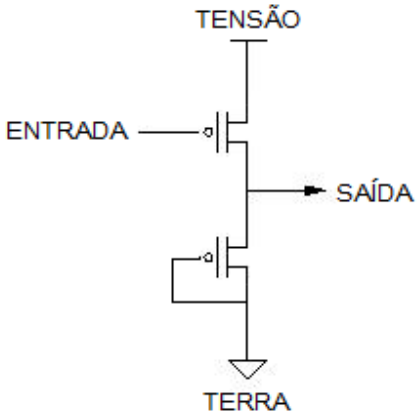

Figura 1.4: Inversor PMOS.

Uma lógica assim não passa com qualidade o nível baixo, o ’0’ lógico, deficiência do PMOS, e tem um alto consumo quando o transistor de controle está ativo, visto que a tensão vai ser ligada diretamente na saída e ser dissipada para o terra através do transistor que funciona como um resistor.

1.3.2 Circuitos NMOS

A lógica subsequente à lógica PMOS foi exatamente a sua complementar, a lógica NMOS. A vantagem em relação a PMOS foi o seu menor tamanho e a melhor capacidade

de transferir o nível baixo, o ’0’. Esse estilo lógico se vale somente de transistores negativos, ou seja, transistor com base em um semicondutor positivo (com mais espaços vazios) que tem a região ativa dopada de forma a ficar mais negativa (com mais elétrons). No caso do silício, normalmente se dopa a região ativa com fósforo ou arsênio para torná-la mais negativa. Também se utiliza dióxido de silício e polissilício com as mesmas finalidades que para o PMOS. Na Figura 1.5 mostra-se um transistor NMOS em três dimensões e o símbolo respectivo utilizado nos esquemáticos.

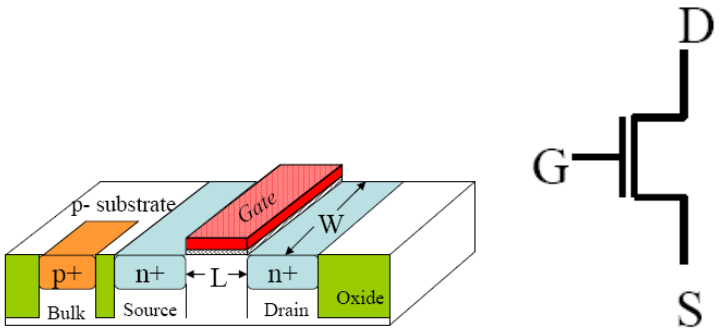

Figura 1.5: Transistor NMOS e o símbolo utilizado para representação.
(RABAEY; CHANDRAKASAN; NIKOLIĆ, 1996)

Para fazer portas lógicas com esses transistores também se utiliza um deles sempre ligado (com o *gate* ligado na tensão do sistema), com a fonte na tensão e o dreno na saída e a lógica propriamente dita é feita utilizando uma árvore de transistores NMOS ligada na saída e no terra. Há ainda o contato do substrato (*bulk*), que precisa ser ligado no terra do sistema para que não altere a tensão de *threshold* do NMOS, devido ao efeito de corpo (*body effect*). Para fazer um inversor, por exemplo, podemos fazer como na Figura 1.6. Uma lógica assim não passa com qualidade o nível alto, o ’1’ lógico, deficiência do NMOS, e tem um alto consumo quando o transistor de controle está ativo, assim como o inversor PMOS.

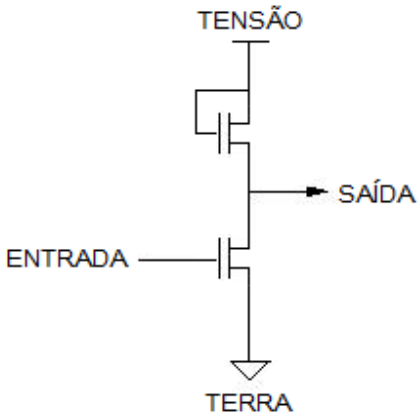

Figura 1.6: Inversor NMOS.

1.3.3 Circuitos CMOS

O CMOS clássico, o mais utilizado até hoje, foi a evolução das duas lógicas anteriores. Visto que o PMOS é um bom condutor de nível '1' e o NMOS é um bom condutor de nível '0', se juntou os dois, criando-se a lógica complementar. A desvantagem em relação às lógicas anteriores é que o CMOS necessita de mais área para lógicas de maior complexidade, visto que a lógica é duplicada, são duas árvores de transistores, uma de PMOS e outra de NMOS. No processo de fabricação então são necessários processos a mais para dopagem do semicondutor. Além disso é necessário organizar o leiaute de maneira que as entradas entrem tanto na rede PMOS quanto na NMOS, o que nem sempre é trivial. Na Figura 1.7 mostra-se um inversor utilizando CMOS padrão.

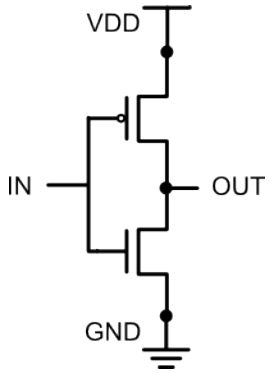

Figura 1.7: Inversor CMOS.

No entanto, dessa forma se obteve um sistema muito robusto, conduzindo bem tanto o nível alto quanto o nível baixo e não tendo um consumo excessivo (somente há um pequeno curto-circuito quando há troca do valor da saída), tornando-se a lógica padrão para os circuitos integrados.

1.3.4 Lógicas de transistores de passagem

As lógicas de transistores de passagem se valem da capacidade de chaveamento dos transistores, deixando passar um caminho ou outro. A mais robusta, a complementar, se vale de transistores de passagem duplicados (*transmission gates*), com PMOS e NMOS, precisando de sinais de controle normais e negados, mas garantindo a qualidade do sinal em valor '0' e '1'. São de bom funcionamento e muito utilizadas em algumas portas lógicas de seleção, como registradores e multiplexadores (como o visto na Figura 1.8). Porém, quando sua lógica aumenta, torna-se muito complexa e há uma degradação do sinal, dependendo muito da qualidade do sinal das entradas e sua capacidade de carga.

1.3.5 Lógicas Dinâmicas

Conforme citada anteriormente, a lógica dinâmica se vale de uma pré-carga da saída, se tornando muito eficiente, porém com muito consumo. A lógica é feita com uma árvore NMOS e há dois transistores extras de controle, gerando duas fases: pré carga, quando o PMOS é ativado e carrega a saída em '1'; e avaliação, quando o PMOS é desativado e, conforme a lógica das entradas, o circuito descarrega a saída para '0' ou mantém em '1'. Na fase de avaliação é quando as entradas precisam estar estáveis e a saída está correta. A característica da lógica dinâmica pode ser vista na Figura 1.9.

Apesar de ter um desempenho alto, uma área menor que as outras e um controle relativamente simples, pode ter um consumo muito grande. Isso acontece quando a lógica

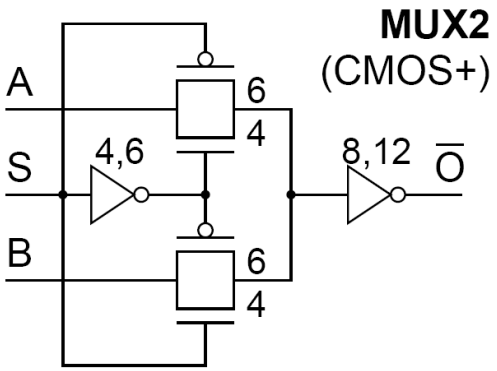

Figura 1.8: Multiplexador utilizando lógica de transistor de passagem complementar. (ZIMMERMANN; FICHTNER, 1997)

das entradas resulta em descarga da saída, o circuito vai ficar carregando e descarregando a saída até que a lógica das entradas mude de forma a deixar a saída em '1'. O controle necessário é devido a saída estar correta somente na fase de avaliação, precisando de um certo cuidado no cascateamento entre portas lógicas.

1.4 Apresentação da Lógica Adiabática

Nessa seção será apresentada a lógica adiabática ou lógica reversível, colocando seu conceito teórico e algumas regras de como implementá-la. Maiores detalhes serão tratados nos Capítulos subsequentes.

1.4.1 O que é lógica adiabática?

Lógica adiabática é o termo dado a circuitos de baixo consumo de potência que implementam lógica reversível. O termo vem da física, onde um processo adiabático é aquele que o calor ou energia total do sistema permanece constante, não entrando nem saindo calor. Com os circuitos eletrônicos ficando menores e mais rápidos, a dissipação de energia deles também aumenta. Utilizando uma técnica que reaproveite essa energia, teremos circuitos com menos problemas relacionados a resfriamento e principalmente, a energia elétrica será poupada. Existem diversas técnicas de lógica adiabática que serão descritas no Capítulo 2, a maioria delas se baseia em circuitos CMOS, que já são muito eficientes em questões de consumo quando comparados a técnicas similares, como foi mostrado nas seções anteriores. Para entender o que se pretende fazer em lógica, pode-se utilizar uma analogia: existem duas banheiras, uma quente, com 50 graus e outra fria com 5 graus Celsius, por exemplo. Para que a fria chegue à temperatura da quente, de uma vez só, é necessário muita energia. Porém, se colocarmos banheiras com temperaturas menores entre elas, será gasto menos energia entre cada banheira. Agora, se forem infinitas banheiras entre as duas, a diferença de energia entre cada banheira tenderá a zero. É mais ou menos essa a ideia.

1.4.2 Como fazer lógica adiabática?

Na lógica CMOS, a maior parte da dissipação de energia se deve à carga e à descarga da capacitância de *gate* C através de uma resistência R. A energia dissipada E pode ser definida como na Equação 1.1, onde V é a tensão do circuito e T é o período de tempo

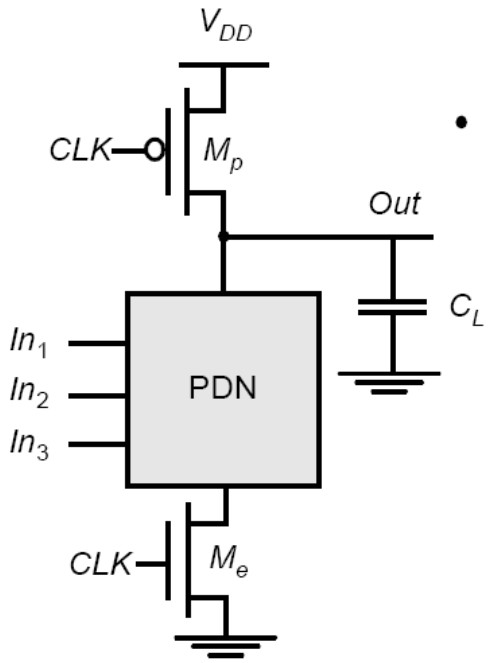

Figura 1.9: Característica da lógica dinâmica.
(RABAEY; CHANDRAKASAN; NIKOLIĆ, 1996)

que o *gate* leva para carregar ou descarregar.

$$E = \frac{RC}{T}(CV^2) \tag{1.1}$$

Em circuitos convencionais, o T é proporcional a RC. Em lógicas reversíveis, se usa o fato de que T é muito maior que RC e tenta-se espalhar a carga por todo ciclo de clock e então reduzir a energia dissipada. Para aumentar o tempo de carga da capacitância *gate*, deve-se garantir que o transistor não irá ligar se houver uma diferença de potencial entre a fonte e o dreno e, uma vez ligado, a energia deve percorrer por ele de forma gradual e controlada. Essa é a primeira regra da lógica adiabática, segundo (FRANK, 2003). A segunda regra é nunca desligar um transistor quando existir corrente entre dreno e fonte, se deve ao fato de que transistores não fazem chaveamentos perfeitos indo de ligado para desligado instantaneamente. Ao invés disso, ele varia gradualmente de ligado para desligado quando a tensão de *gate* muda. Além disso, a mudança de ligado para desligado é proporcional à velocidade de variação da tensão de *gate*. Este fato, combinado com a primeira regra, implica que o transistor está em um estado transiente por um longo período de tempo. Durante esse tempo, a tensão cai muito, apesar de a resistência não ser grande o suficiente para trazer a dissipação de energia a zero. Essas regras são importantes para desenvolvimento de novas lógicas. Os estilos apresentados nesse trabalho obedecem essas regras.

1.5 Estrutura do texto

No Capítulo 2 fala-se um pouco sobre técnicas existentes para diminuir o consumo de circuitos CMOS, comparando-as e discutindo a viabilidade de cada uma. No fim do Capítulo, apresenta-se a lógica adiabática e suas diversas famílias lógicas, bem como trabalhos já feitos com esse estilo lógico. Além disso, discute-se sobre a implementação das fontes

necessárias. No Capítulo 3, descreve-se o funcionamento elétrico dos estilos lógicos adiabáticos estudados, bem como a implementação e as dificuldades de cada um. Após isso, faz-se uma análise de leiaute, prevendo-se dificuldades e facilidades em uma possível implementação em hardware, no Capítulo 4. Por fim, no Capítulo 5, implementa-se um circuito de maior complexidade com CMOS clássico, um circuito CMOS funcionando em modo de operação *subthreshold* e um utilizando lógica adiabática, analisando as vantagens e desvantagens de cada um.

2 ESTILOS LÓGICOS DE BAIXO CONSUMO

O baixo consumo sempre é um dos objetivos em qualquer ramo de produtos. Seja um produto eletrônico ou que utiliza combustíveis fósseis, a eficiência é muito importante para a viabilidade do produto. E isso não é diferente nos circuitos integrados. Para tanto, foram desenvolvidas diversas técnicas no nível lógico para diminuir o consumo dos circuitos e isso é o que será tratado nesse Capítulo. Conforme pesquisas feitas em (ZIMMERMANN; FICHTNER, 1997), não será tratado aqui lógicas com transistores de passagem, visto que não apresentam vantagens em relação a circuitos CMOS convencionais na questão de consumo de energia. Além do que será mostrado aqui, existem alternativas para diminuir o consumo e aumentar o desempenho, como em (NAVI et al., 2008), onde foi rearranjada as funções lógicas de um somador completo (*full-adder*) reorganizando os transistores e conseguiu-se uma melhora tanto em desempenho como em consumo. No fim do Capítulo, apresenta-se o funcionamento das diversas famílias lógicas adiabáticas, seguido de implementações acadêmicas em lógica adiabática e o que já está sendo feito comercialmente nesse sentido.

2.1 Escala de tensão

Diminuir a tensão é algo que afeta diretamente o gasto de energia do sistema, visto que se diminui a fonte de energia, diminuindo diretamente o gasto. Revendo a equação 1.1, notamos que diminuindo a tensão, o gasto energético vai diminuir quadraticamente. Dessa forma, não é surpresa nenhuma que uma das primeiras formas de explorar a diminuição do consumo de um circuito tenha sido a escala de tensão. O motivo pelo qual a tensão menor irá afetar o desempenho do circuito é que, com uma tensão mais alta, a determinada para a tecnologia, a carga e descarga das capacitâncias de *gate* serão mais rápidas, de acordo com o que o fabricante determinou. Diminuindo essa tensão, mais lentas ficarão essas transições pela maior dificuldade de carregar a capacitância de saída.

2.1.1 Escala de tensão dinâmica

Já aplicado em computadores portáteis atuais, a ideia básica é diminuir a tensão de funcionamento do sistema quando for necessário um menor consumo. Apesar de diminuir o desempenho do computador, isso se torna muito útil para aumentar o tempo que se está utilizando a bateria, por exemplo. Para implementar isso em lógica é necessário um certo controle, além de fontes com valores de tensão diferentes ou uma fonte controlável. Por exemplo, podemos ter três fontes, uma com tensão máxima, outra menor e outra mínima para o funcionamento. Quando for executar uma função de baixo desempenho se utiliza uma tensão menor e vice-versa. Nesse caso teríamos um ganho em consumo

sem perder desempenho, mas haveria um gasto extra de lógica para controlar a fonte ou o chaveamento das fontes, que precisariam detectar se o atual processo é *CPU-bound*, ou seja, que necessita mais do circuito do que de entradas e saídas, ou se é *IO-bound*, necessitando mais das entradas e saídas do que do processamento.

2.1.2 Funcionamento em nível *subthreshold*

Apesar de tecnicamente os transistores só começarem a conduzir da fonte para o dreno quando a tensão entre o *gate* e a fonte for maior que a tensão de *threshold*, essa não é uma verdade absoluta. Os transistores MOS não deixam de funcionar em tensões abaixo da tensão de *threshold*, daí o termo *subthresold logic*. Os transistores não só continuam funcionando como tem um consumo energético muito menor que circuitos CMOS convencionais. O seu grande problema, no entanto, é o desempenho, pois só funcionam com frequências muito baixas. O que pode não ser um problema para certas aplicações, desde que não necessitem de um maior desempenho.

2.2 Lógica Adiabática

Conforme explicado no Capítulo anterior, a lógica adiabática se vale do tempo de carga para não ter um gasto energético alto. Para tanto a fonte precisa gerar essa rampa de carga. Além disso, a descarga dos capacitores de *gate* é levada de volta para a fonte. Para tanto, a fonte necessita de um projeto especial, com capacitores e indutores, mas esse não é o objetivo do trabalho, como vai ser dito mais adiante. Nessa seção o objetivo é descrever as famílias lógicas existentes e seu funcionamento lógico, deixando para a análise elétrica e de implementação para os próximos Capítulos. As famílias lógicas adiabáticas podem ser divididas em quantidade de fontes, sendo necessário um cuidado no cascateamento, determinando a fonte a ser utilizada em cada fase. Para simplificar a explicação das famílias lógicas, define-se VDD como a tensão do sistema, GND como o terra do sistema, '1' como nível lógico '1' e '0' como nível lógico '0'.

2.2.1 Pass-Transistor Adiabatic Logic (PAL)

Uma porta lógica PAL, visto em (OKLOBDZIJA; MAKSIMOVIC; LIN, 1997), consiste de dois blocos funcionais, um real e um complementar, de uma função lógica implementada com NMOS, com um par de transistores PMOS cruzados para memorização. Um exemplo de uma porta lógica pode ser visto na Figura 2.1, onde foi implementada uma função AND-OR: $Q = (A * B) + C$.

A energia do sistema é dada por um relógio (*clock*) senoidal (PC), que quando vai de '0' para '1', dependendo das entradas, ele segue um caminho por um dos blocos, real ou complementar, setando a saída correspondente em '1'. O outro nodo vai ficar em alta impedância e mantido próximo a zero pela capacitância de carga. Dessa forma, um dos transistores PMOS irá conduzir e carregará a saída. O estado é válido quando o valor de PC é próximo de '1'. Por fim, o *clock* senoidal irá voltar a '0', recuperando a energia armazenada no nodo de saída. Por ter somente duas fases, uma de avaliação e outra de recuperação, a lógica PAL necessita de somente duas fontes para um correto cascateamento.

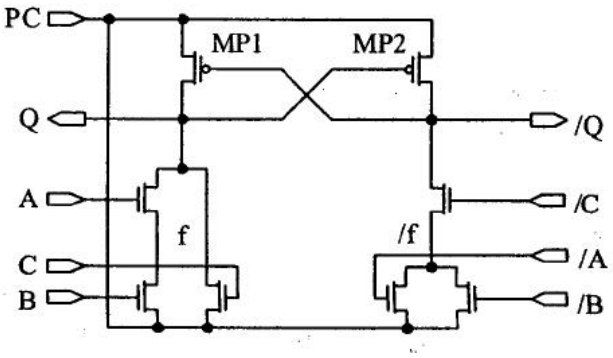

Figura 2.1: Porta lógica AND-OR usando PAL.
(OKLOBDZIJA; MAKSIMOVIC; LIN, 1997)

2.2.2 Clocked-CMOS adiabatic logic (CAL)

Um dos componentes mais simples e mais utilizados em um circuito integrado é o inversor, que é mostrado na Figura 2.2. A lógica CAL, desenvolvida por (MAKSIMOVIC et al., 2000), mostra um circuito um pouco mais complexo, mas com uma boa eficiência e baixo consumo de energia. Os transistores M1 a M4 são inversores cruzados que dão a função de memorização. Os transistores M7 e M8 podem ser substituídos por uma rede de transistores NMOS que efetuem uma função binária qualquer. A diferença básica dos circuitos 2N-2N2P é o uso dos transistores M5 e M6, podendo ser utilizado então somente uma fonte trapezoidal (Pck) ao invés de quatro.

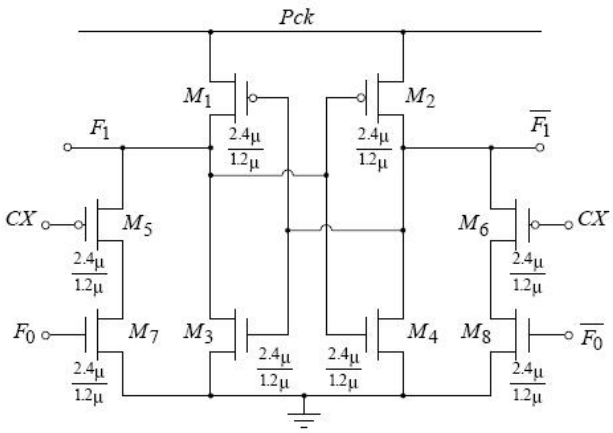

Figura 2.2: Inversor usando CAL.
(MAKSIMOVIC et al., 2000)

O resultado de uma simulação do inversor CAL é mostrado na Figura 2.3. A avaliação da entrada é feita pelo *clock* CX. Quando CX for '1' é feita a avaliação: se F0='0', a saída /F1 é levada a '0' e a saída F1 recebe Pck. Quando a avaliação é '0', as saídas recebem o valor armazenado. Um pulso de Pck é desperdiçado dessa forma, pois só é utilizado quando a avaliação está ativa. Para reduzir consumo de energia nos *clocks* de avaliação, pode-se utilizar a variação entre eles, sem alterar os níveis de sinais da lógica. CAL pode também operar como um sistema convencional, com VDD ligado à Pck.

A energia do sistema é dada por um *clock* senoidal (PC), que quando vai de '0' para '1', dependendo das entradas, ele segue um caminho por um dos blocos, real ou comple-

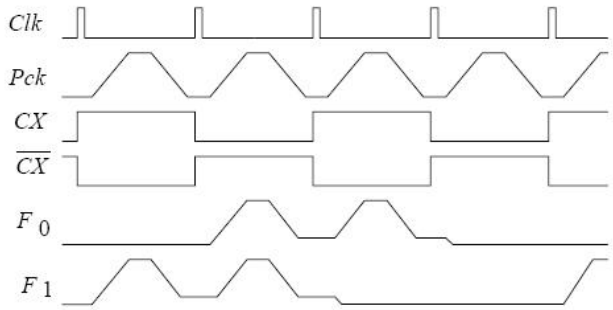

Figura 2.3: Sinais necessários para o sistema CAL.
(MAKSIMOVIC et al., 2000)

mentar, setando a saída correspondente em '1'. O outro nodo vai ficar em alta impedância e mantido próximo a zero pela capacitância de carga. Dessa forma, um dos transistores PMOS irá conduzir e carregará a saída. O estado é válido quando o valor de PC é próximo de '1'. Por fim, o *clock* senoidal irá voltar a '0', recuperando a energia armazenada no nodo de saída.

2.2.3 Efficient Charge Recovery Logic (ECRL) ou 2N2P

Outra técnica que se vale de *clocks* trapezoidais é a ECRL, vista em (MOON; JEONG, 1996), ou 2N2P como visto em (KRAMER et al., 1995). As funções lógicas também são definidas por redes de transistores NMOS e há dois transistores PMOS para carregar a saída, devido ao fato de transistores PMOS passarem o '1' com mais qualidade. Um exemplo de inversor e como seria o *clock fonte* do circuito estão na Figura 2.4.

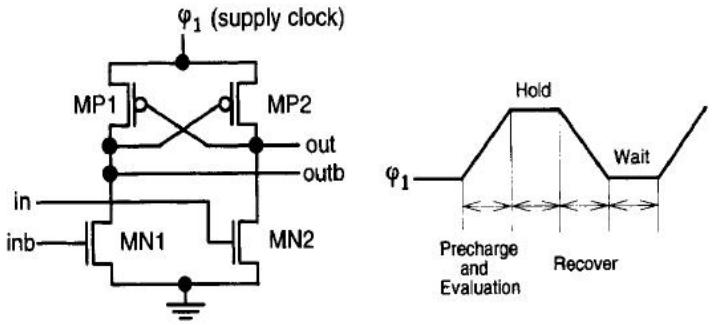

Figura 2.4: Inversor usando ECRL ou 2N2P e exemplo de fonte.
(MOON; JEONG, 1996)

A lógica se vale da ideia de uma fonte de energia dada por um *clock* trapezoidal, duas redes de transistores NMOS, um com lógica normal e outro com lógica negada e um registro feito por um par de transistores PMOS. Tem-se quatro fases distintas: fase de recuperação, fase de espera, fase de avaliação e fase de retenção. Logo, serão necessários quatro fontes de *clock* diferentes, com diferença de fase de um quarto de período entre eles. Na fase de recuperação, as entradas estão em '0' e as saídas são complementares. Nesse momento, a saída que está em '1', acompanhará a fonte e irá a '0', devolvendo sua carga à fonte. Durante a fase de espera, o *clock* está em '0' e as entradas devem ser alteradas. Por exemplo, se tivermos a entrada *in* em '1' e *inb* em '0', ou seja, a saída *outb* ficará em aberto e a saída *out* ficará em '0'.

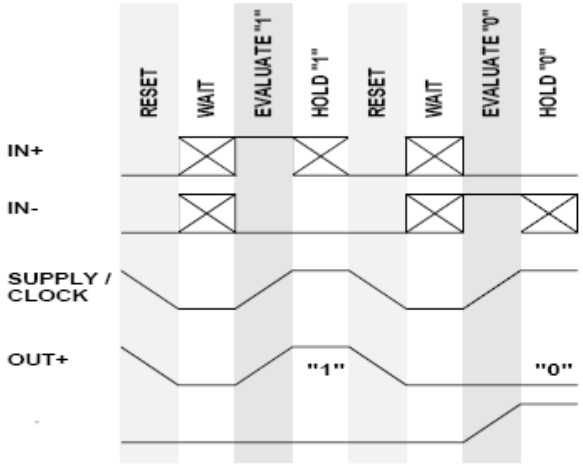

Figura 2.5: Simulação das entradas e saídas.
(KRAMER et al., 1995)

Após isso, na fase de avaliação, ambas saídas estão em ’0’, como pode ser visto na Figura 2.5. Porém, com o *clock fonte* indo para ’1’, carrega-se a saída *outb*. Ao mesmo tempo, a saída *out* permanece em ’0’, já que está ligada em ’0’ através do bloco funcional. Durante essa fase, as entradas precisam se manter constantes. Na fase de retenção, as entradas são levadas à ’0’ e ambos blocos funcionais NMOS são desligados. Dessa forma, as saídas tem seus valores mantidos pelos transistores PMOS cruzados e durante essa fase eles podem ser usados em um próximo estágio.

2.2.4 2N-2N2P

A lógica 2N-2N2P, também feita em (KRAMER et al., 1995), é uma variação do mesmo tema: fonte de energia dada por um *clock* trapezoidal, lógica normal e negada e um registro feito por um par de inversores CMOS. A diferença de 2N2P é a inserção de dois NMOS, MN1 e MN2. Tem-se 4 fases diferentes: fase de entrada, fase de avaliação, fase de retenção e fase de recuperação. Logo, serão necessários quatro fontes de *clock* diferentes. Segue a idéia das lógicas anteriores e o buffer/inversor é também de relativa simplicidade. Durante a fase de entrada, o *clock* é colocado em ’0’ e as entradas podem ser alteradas. Por exemplo, se tivermos a entrada *in* em ’1’ e */in* em ’0’, o transistor MN3 está fechado e o MN4 está aberto. Ou seja, temos que o circuito que define a saída *out* funcionando e o circuito que define a saída */out* em aberto.

Após isso, no início da fase de avaliação, ambas saídas estão em ’0’, como pode ser visto na Figura 2.6. Porém, com o *clock fonte* indo para ’1’, carrega-se a saída */out* através do transistor MP2. Ao mesmo tempo, a saída *out* permanece em ’0’, já que está ligada em ’0’ através do bloco funcional. No fim, MN1 é fechado e as saídas são salvas pelos transistores PMOS cruzados. Durante essa fase, as entradas precisam se manter constantes Na fase de retenção, as entradas são levadas à ’0’ e ambos blocos funcionais NMOS são desligados. Dessa forma, as saídas tem seus valores mantidos pelos transistores PMOS cruzados e durante essa fase eles podem ser usados em um próximo estágio. Por fim, a fase de recuperação diminui o *clock* de carga até zero. Quando o *clock* diminui, */out* vai para ’0’ via MP2 até atingir a tensão de *threshold*, quando MP2 é aberto e a saída */out* permanece igual. A carga que ainda está em */out* é dissipada sem ser reaproveitada se no próximo ciclo o bloco funcional NMOS complementar for ligado.

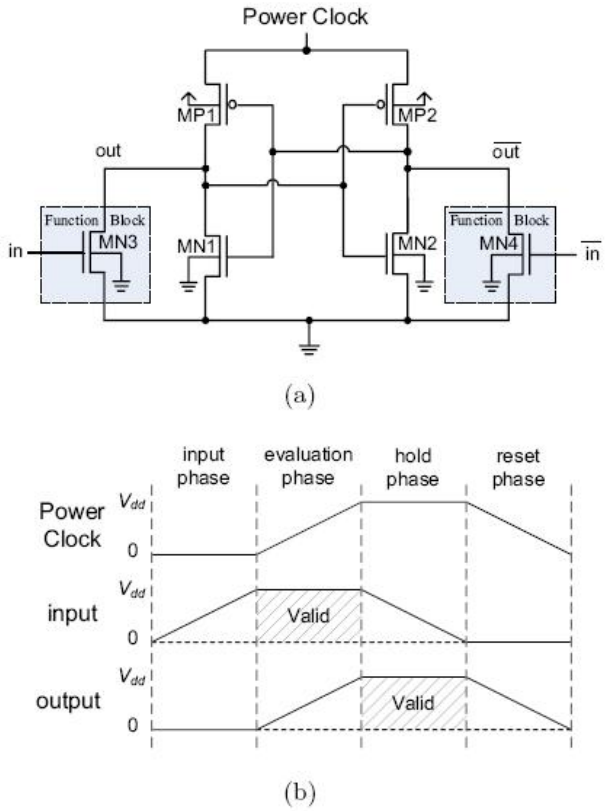

Figura 2.6: (a) Inversor utilizando 2N-2N2P. (b) Simulação das entradas e saídas. (KRAMER et al., 1995)

2.2.5 Positive Feedback Adiabatic Logic (PFAL)

A estrutura básica da família lógica PFAL, visto em (VETULI; PASCOLI; REYNERI, 1996), é mostrada na Figura 2.7. Dois blocos funcionais NMOS executam a função e a complementar. Existe a saída e o seu complementar. A diferença em relação aos anteriores é que a rede lógica N é ligada diretamente na fonte e na saída e o uso de capacitores para dar uma melhor qualidade na saída.

A razão entre a energia necessária em um ciclo e a dissipada pode ser vista na Figura 2.8. Durante a fase de recuperação de energia, a capacitância de saída devolve a energia para a fonte, com a queda da energia.

2.2.6 True Single-Phase Energy-Recovering Logic (TSEL)

TSEL, visto em (KIM; PAPAEFTHYMIOU, 1998), é uma lógica parcialmente adiabática como as outras, com algumas diferenças. A energia é dada por somente uma fonte de *clock* para PMOS e uma para NMOS. Duas referências DC garantem operações de alto desempenho e eficiência. O funcionamento do TSEL é dividido em portas PMOS e NMOS com o cascateamento de circuitos feito alternando circuitos PMOS e NMOS. A estrutura básica de uma porta lógica PMOS e uma NMOS é mostrada na Figura 2.9. Esse inversor compreende um par de transistores cruzados (MP1 e MP2), um par de chaves de controle de corrente (MP3 e MP4) e dois blocos funcionais (MP5 e MP6). A porta PHI é a entrada do *clock fonte*. A porta PREF é para a tensão de referência, no caso do PMOS seria o '1'. No entanto, diminuindo esse valor, temos uma queda de desempenho e uma diminuição no consumo. Entradas e saídas tem seus valores normais e negados. A

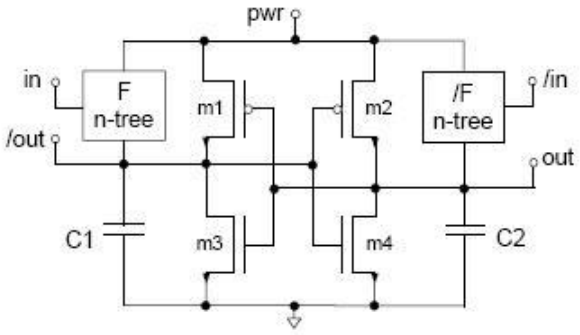

Figura 2.7: Família lógica PFAL.
(VETULI; PASCOLI; REYNERI, 1996)

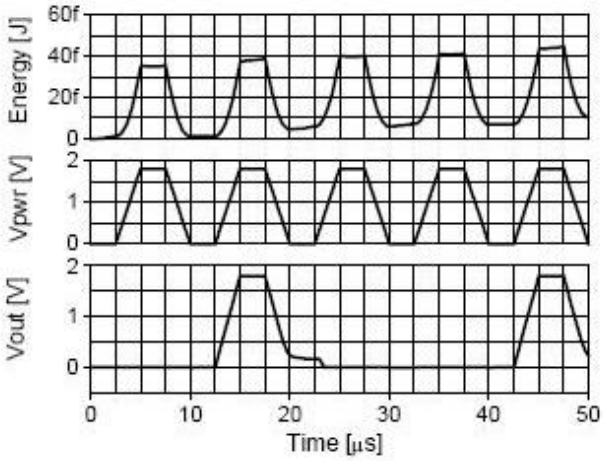

Figura 2.8: Simulação das ondas do circuito PFAL.
(VETULI; PASCOLI; REYNERI, 1996)

principal diferença entre outras famílias de lógica é a tensão de referência e as chaves de controle de corrente. A operação é dada em duas fases: descarga e avaliação. Durante a descarga, a energia armazenada nas capacitâncias de saída de *out* e */out* é recuperada. Inicialmente o *clock* está em '1', e quando começa a diminuir até '0', ele coloca ambas saídas em '0' através da tensão de *threshold* do PMOS. Esse evento é adiabático até que o *clock* fique abaixo da tensão de referência menos a tensão de *threshold*.

Durante a fase de avaliação as entradas são verificadas e as saídas começam a ser resolvidas. Assumindo que a entrada *in* está em '1' e */in* está em '0'. Inicialmente o *clock* está em '0'. Quando começa a aumentar a tensão, MP1 e MP2 ligam. Enquanto o *clock* se mantém abaixo da tensão de referência menos a tensão de *threshold*, MP3 e MP4 estão conduzindo. Quando PREF é maior que o *clock*, um caminho de *pull-up* é criado de PREF até */out*, que vai subindo até '1'. Os PMOS cruzados funcionam como um amplificador e aumentam a diferença entre os dois nós de saída. Assim que a diferença é maior que a tensão de *threshold*, MP1 desliga e a saída */out* é carregada adiabaticamente. Quando o *clock* excede a tensão de referência menos a tensão de *threshold*, MP3 e MP4 desligam e desconectam os blocos funcionais das saídas. A partir desse momento, mudanças nas entradas não seriam propagadas às saídas. No fim da fase de avaliação, /out é carregado pelo *clock fonte*. A saída deve ser amostrada no topo do *clock*. Idealmente, o nodo de

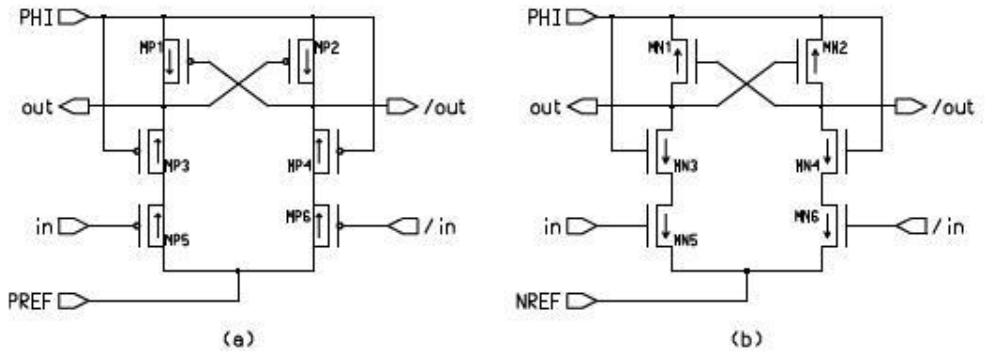

Figura 2.9: (a) TSEL PMOS. (b) TSEL NMOS.
(KIM; PAPAEFTHYMIOU, 1998)

saída *out* permanece na tensão de *threshold* durante a fase de avaliação. A operação da porta NMOS é complementar à do PMOS. As duas fases de operação são as mesmas. Durante a avaliação, um caminho é formado de uma saída para NREF, que seria '0' para o NMOS, até que o *clock* atinja a tensão da referência mais a tensão de *threshold*. No entanto, aumentando esse valor de referência, temos uma queda de desempenho e uma diminuição no consumo. Quando o *clock* está abaixo disso, as chaves de controle de corrente desconectam os blocos funcionais das saídas, e a energia armazenada em uma das saídas é iniciada.

2.2.7 Single-Phase Source-Coupled Adiabatic Logic (SCAL)

SCAL, visto em (KIM; PAPAEFTHYMIOU, 1999) é uma implementação melhorada de TSEL. Essa família lógica consegue uma menor dissipação de energia que TSEL para uma série de frequências de operação. Ela é uma família lógica parcialmente adiabática que utiliza somente uma fonte de *clock*. O baixo consumo é obtido dimensionando a fonte de corrente em cada *gate*. A estrutura de um inversor em SCAL pode ser visto na Figura 2.10.

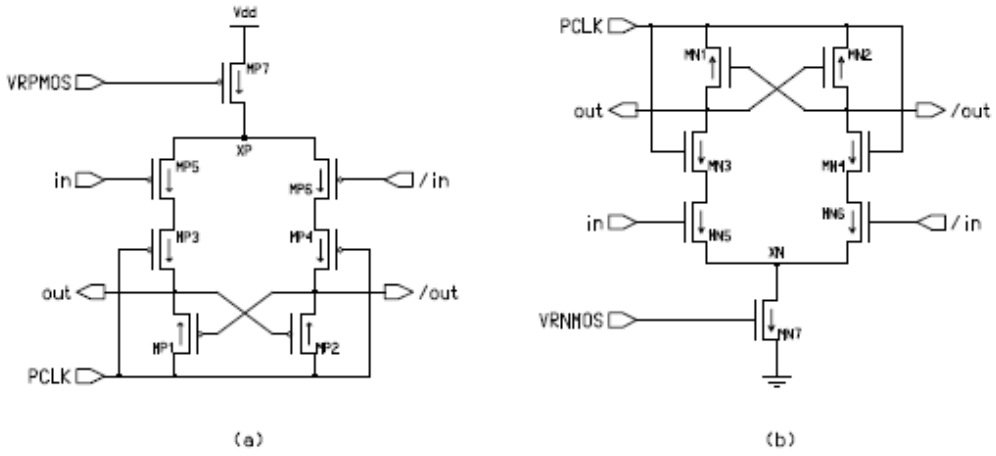

Figura 2.10: (a) SCAL PMOS. (b) SCAL NMOS.
(KIM; PAPAEFTHYMIOU, 1999)

O inversor PMOS tem um par de transistores para registro da saída (MP1 e MP2), um

par de transistores para controle da corrente (MP3 e MP4), dois blocos lógicos (MP5 e MP6) e um transistor para controle da fonte de corrente (MP7). Essa fonte de corrente é a característica que diferencia do TSEL. A quantidade de carga é controlado pela dimensionamento de MP7 (W/L). A fonte DC é necessária para ativar o registro em MP1 e MP2. A fonte de *clock* senoidal é PCLK. São duas fases de operação: descarga e avaliação. A energia armazenada em out ou /out é recuperada na fase de descarga. Nessa fase, a fonte de *clock* começa em '1' e vai até '0', colocando ambas saídas em '0'. A nova entrada é computada na fase de avaliação, quando a fonte de *clock* vai de '1' até '0', ligando MP1 e MP2. Quando MP7 é ligado, o nodo interno XP é ligado em VDD. Assumindo que *in* está em '1' e */in* está em '0', */out* é ligado ao nodo XP, levando a '1'. Os PMOS cruzados funcionam como um amplificador e aumentam a diferença entre os dois nós de saída. Assim que essa diferença excede a tensão de *threshold*, MP1 é desligado e a saída */out* é carregada adiabaticamente. Quando o *clock* excede a tensão de referência menos a tensão de *threshold*, MP3 e MP4 desligam e desconectam os blocos funcionais das saídas. A partir desse momento, mudanças nas entradas não seriam propagadas às saídas. No fim dessa fase, */out* é carregado até o topo da fonte de *clock*.

2.2.8 Split Charge Recovery Logic (SCRL)

No trabalho de doutorado (YOUNIS, 1994), foi desenvolvida uma família de circuitos adiabáticos conhecidos como Split Charge Recovery Logic (SCRL). A Figura 2.11 mostra como uma NAND é implementada utilizando esse tipo de lógica. O circuito é bastante parecido à uma porta lógica NAND convencional, porém uma das diferenças é que o circuito é regido por *clocks* trapezoidais $\phi 1$ e $/\phi 1$, ao invés de VDD e GND. Na saída da NAND tem um transistor de passagem regido por P1 e /P1. No começo, $\phi 1$ e $/\phi 1$ são setados em VDD/2, enquando P1 é ressetado para '0' para desligar o transistor de passagem. O próximo passo é gradualmente chavear o valor de P1 para VDD. Em seguida, $\phi 1$ é setado em VDD e $/\phi 1$ em ´0´. Nesse momento, o funcionamento é igual a uma NAND convencional. Após a saída ser capturada, o circuito deve gradualmente voltar ao estado anterior, não modificando as entradas até estabilizar para não violar a primeira regra.

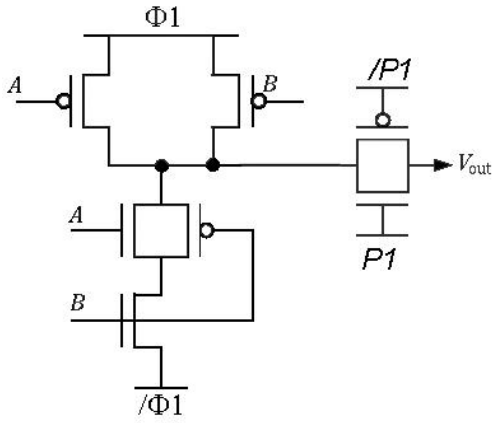

Figura 2.11: NAND utilizando SCRL.
(YOUNIS, 1994)

Por fim, explica-se porque tem um PMOS adicional conectado à entrada *b*. Quando esse transistor não está ali, se *a* assumir o valor '1' e *b* o valor '0', corrente vai passar de

VDD pelo PMOS controlado por *b* até o NMOS controlado por *a*, o que significa que há uma dissipação desnecessária de energia. Esse problema é resolvido com o PMOS extra, mas deve-se ter cuidado para que de alguma outra forma não exista dissipação de energia.

2.3 Implementação e lógica adiabática comercial

Nessa seção será feita uma análise de circuitos acadêmicos que já foram implementados em lógica adiabática e seus resultados. Além disso, serão mostradas soluções comerciais que já estão sendo vendidas que buscam reduzir o consumo em regiões críticas dos circuitos, como buffers de entrada e saída. Também faz-se uma análise das fontes em um circuito adiabático, trazendo-se problemas e possíveis soluções.

2.3.1 Implementações acadêmicas com lógica adiabática

Na literatura acadêmica encontraram-se diversas implementações em lógica adiabática, como somadores e multiplicadores, comparando o desempenho de famílias lógicas adiabáticas com circuitos CMOS convencionais. Nessa seção serão mostrados os resultados dessas implementações em relação a desempenho e consumo.

2.3.1.1 Somadores

Em geral, para o teste de funcionamento e comparação, os pesquisadores se valem da simulação de somadores. São circuitos mais simples, que agilizam o desenvolvimento e teste, mas que são muito utilizados e podem ser usados como referência por conterem lógica utilizada em todos outros circuitos integrados. No trabalho (MOON; JEONG, 1996), foi implementado um somador Carry Look-ahead (CLA) de 16 bits utilizando-se ECRL. Na Figura 2.12 são mostradas simulações feitas para diferentes frequências e tensões, para um CLA ECRL e para um CLA CMOS convencional, onde nota-se uma vantagem em consumo bem significativa para o circuito adiabático.

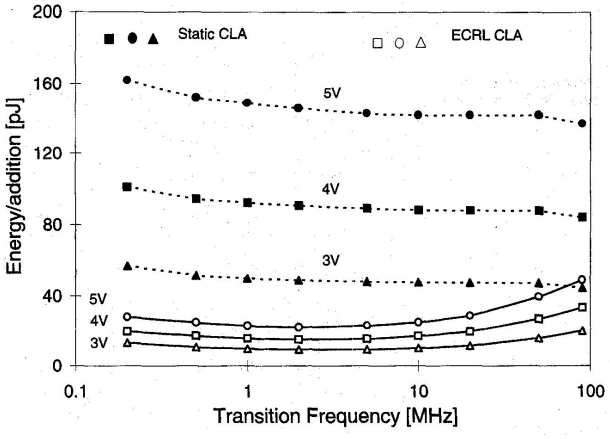

Figura 2.12: Gráfico comparativo CLA 16-bits - ECRL x CMOS.
(MOON; JEONG, 1996)

Na Figura 2.13 mostra-se uma comparação de CLA's funcionando a 10MHz. Ela inclui no cálculo o consumo do circuito ECRL mais o consumo da fonte senoidal com um conversor DC/AC com eficiência de 41%. Mesmo assim, o ganho em consumo é de 56% em relação a um CMOS convencional.

Power Dissipation of ECRL core	258uW
Power Dissipation of DC-to-AC converter	365uW
Total Delivered Power	623uW
Conversion Efficiency	41%
Power Dissipation of Conventional CMOS	1430uW
Power Gain of ECRL over Conventional CMOS	56%

Figura 2.13: Tabela comparativa CLA 16-bits - ECRL x CMOS.
(MOON; JEONG, 1996)

No trabalho (KIM; PAPAEFTHYMIOU, 1999), mais recente, foram implementados diversos CLA's de 4-bits com 4 estágios de pipeline, comparando lógicas adiabáticas entre si e com lógica CMOS convencional. Novamente, as famílias lógicas adiabáticas tiveram um consumo bem inferior ao CMOS padrão, como pode ser verificado na Figura 2.14.

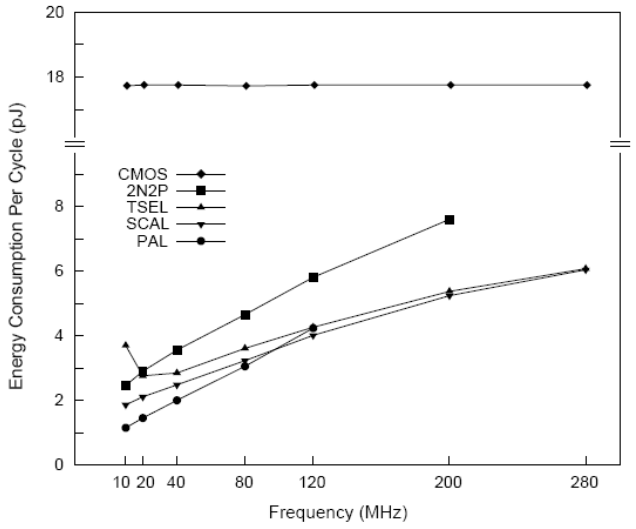

Figura 2.14: Gráfico comparativo CLA 4-bits - Lógica adiabática x CMOS.
(KIM; PAPAEFTHYMIOU, 1999)

Na Figura 2.14 é possível verificar que algumas famílias lógicas tem restrições de funcionamento conforme a frequência, mostrando que apesar de se obter um consumo baixo, tem-se uma limitação de desempenho. O leiaute do CLA implementado com SCAL e com o controle das diversas fontes é mostrado na Figura 2.15.

2.3.1.2 Memória SRAM

Foi implementada uma memória SRAM no trabalho (SOMASEKHAR; YE; ROY, 1995), largamente utilizada comercialmente e uma das principais fontes de consumo em um circuito. Avalia-se a possibilidade de isso ser feito com lógica reversível e quais ganhos que isso traria. E os resultados são positivos, mostrando um leiaute simples (conforme Figura 2.16) e boa relação desempenho versus consumo.

Existe o compromisso entre desempenho e consumo de energia: quanto mais veloz a memória, menos energia é possível ser recuperada, como pode ser visto na Figura 2.17. Mesmo assim, é um resultado interessante para a tecnologia empregada e isso já

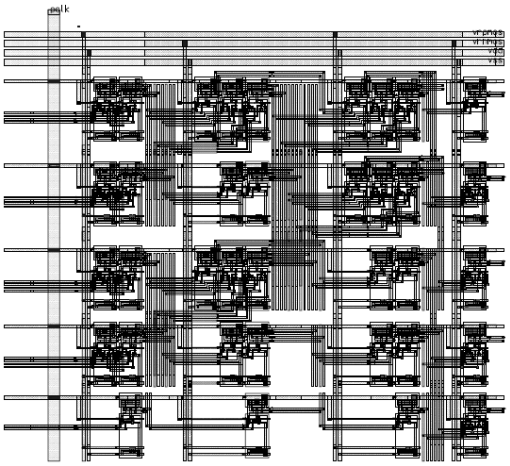

Figura 2.15: Leiaute de um CLA de 4-bits com 4 estágios de pipeline usando SCAL. (KIM; PAPAEFTHYMIOU, 1999)

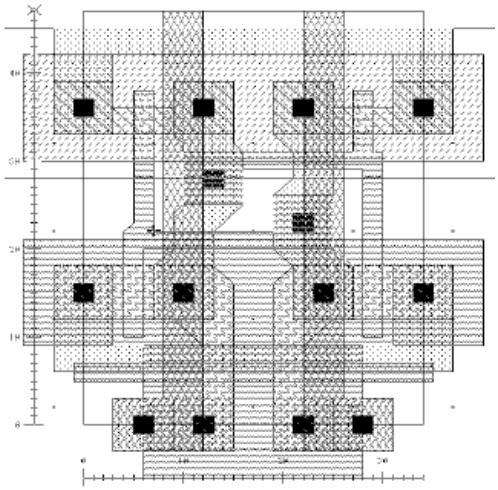

Figura 2.16: Leiaute de uma célula de memória SRAM utilizando lógica adiabática. (SOMASEKHAR; YE; ROY, 1995)

era esperado, visto que um maior consumo geralmente está relacionado com uma maior desempenho.

2.3.2 Lógica adiabática comercial

Já existem empresas no ramo de microeletrônica que estão vendendo tecnologia usando lógica adiabática. Uma delas é a empresa "Adiabatic Logic Power-Saving Electronics Solutions"(LOGIC, 2010) (símbolo da empresa pode ser visto na Figura 2.18), localizada na Universidade de Cambridge, no Reino Unido, dentro do Cambridge Technology Group. Essa empresa tem duas frentes de trabalho: o IOD e o ASB, que são basicamente buffers, grandes consumidores de energia elétrica dentro de um circuito integrado. Nas próximas seções será explicado o funcionamento de cada um desses buffers.

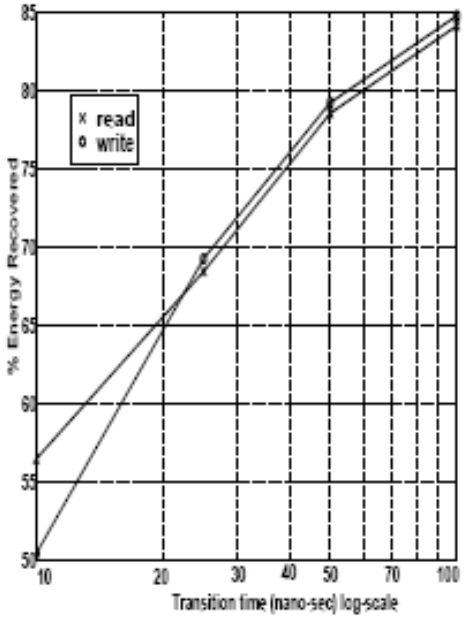

Figura 2.17: Gráfico de desempenho x Recuperação de energia para uma memória SRAM. (SOMASEKHAR; YE; ROY, 1995)

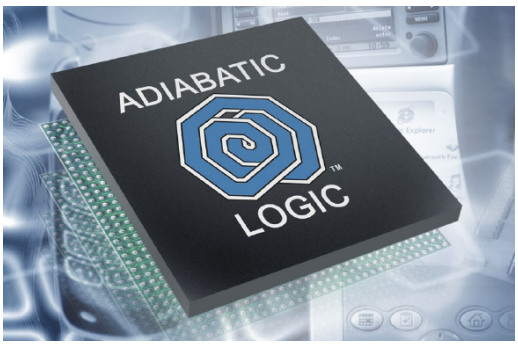

Figura 2.18: Símbolo da empresa Adiabatic Logic. (LOGIC, 2010)

2.3.2.1 Intelligent Output Driver (IOD)

O IOD (Intelligent Output Driver) é uma tecnologia patenteada que opera da mesma maneira que um buffer de entrada e saída convencional, com as mesmas características para o ambiente elétrico, sendo possível simplesmente colocá-lo no lugar de um buffer tradicional. O que ele faz é entregar a corrente sem resistência a partir de uma capacitância mantida na metade da tensão padrão do sistema, ajudada pela indutância inerente da carga. A capacitância entrega carga nas curvas de subida e recupera a carga nas curvas de descida, "reciclando"a energia. Essa tecnologia, segundo (LOGIC, 2010), economiza até 75% do consumo de energia, reduz o número de componentes no sistema, otimiza a integridade do sinal de saída, melhora a velocidade de chaveamento do sinal e simplifica o projeto e simulação de sistemas complexos. A ideia é evitar que a descarga de energia seja perdida para o ambiente em forma de calor por um resistor.

Quando o circuito está em zero, os dois tipos de buffer funcionam da mesma forma, ligados em zero, conforme visto na Figura 2.19. Os circuitos funcionam da mesma forma

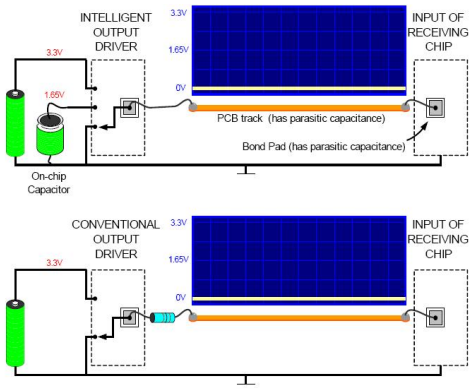

Figura 2.19: IOD - Ambos circuitos em GND.
(LOGIC, 2010)

também quando estão com tensão máxima na saída, pegando energia da fonte, como visto na Figura 2.20.

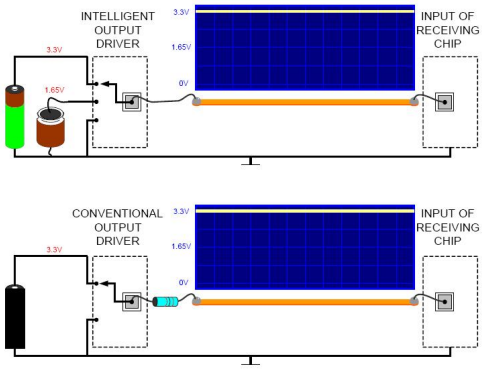

Figura 2.20: IOD - Ambos circuitos em VDD.
(LOGIC, 2010)

Se há uma carga parcial da saída, no buffer inteligente se pega a energia do capacitor, enquanto que no convencional se passa por um resistor em um circuito divisor de tensão, liberando calor para o ambiente e desperdiçando energia. Essa diferença pode ser visualizada na Figura 2.21.

Quando há uma descarga parcial da saída, no buffer inteligente se carrega o capacitor, enquanto que no convencional se passa por um resistor ligado em zero, liberando calor para o ambiente e desperdiçando energia, conforme Figura 2.22.

2.3.2.2 Adiabatic Super Buffer (ASB)

O ASB (Adiabatic Super Buffer) segue a mesma ideia, mas ao invés de focar na comunicação entre circuitos integrados, o foco é utilizar isso para dentro do circuito, em menor escala, em buffers de relógios, por exemplo. Assim como o IOD, o ASB tem como princípio manter e até melhorar o desempenho em relação às soluções atuais. As estimativas para essa tecnologia é de uma economia de 50% de energia em relação aos buffers convencionais.

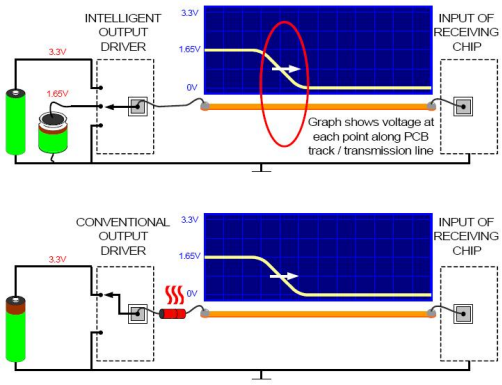

Figura 2.21: IOD - Carga Parcial.
(LOGIC, 2010)

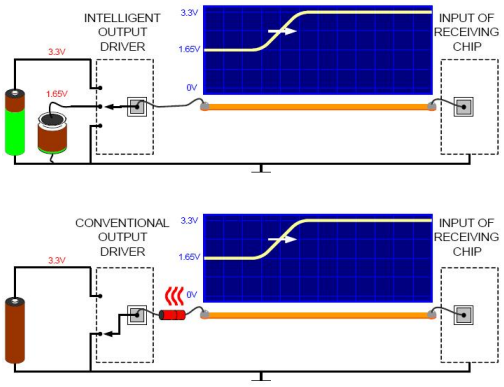

Figura 2.22: IOD - Descarga Parcial.
(LOGIC, 2010)

2.3.3 Os problemas das fontes

Existem diversos problemas referentes à alimentação em circuitos adiabáticos. Primeiro, deve-se ter uma fonte em formato de rampa ou senoidal, acarretando mais lógica ou um hardware específico. Segundo, deve-se ter fontes com diferença de fase para um correto cascateamento, dependendo do estilo lógico a ser utilizado, precisando de mais lógica e um maior controle do desenvolvedor. E por fim, para construir essas fontes corretamente, acarretará um gasto energético, e se esse gasto energético for alto, pode-se não ter ganhos em consumo como um todo. Porém, levando-se em conta que possa ser separada a fonte do circuito, existem aplicações que podem ser interessantes, visto que a necessidade de refrigeração ficaria somente na fonte, podendo se colocar o circuito em um local que não haja a possibilidade disso.

Existem diversas implementações de fontes, como senoidais, em formato de escada (como na Figura 2.24), utilizando indutores e com chaveamento elétrico, todas com boa eficiência energética. No trabalho (YOUNIS, 1994) se faz uma análise teórica dessas fontes, demonstrando algebricamente a eficiência de cada fonte. No trabalho (ZIESLER; KIM; PAPAEFTHYMIOU, 2001) foi implementada uma fonte com boa eficiência, visando ser utilizada com lógica adiabática. Como o foco do trabalho não é a fonte e sim o gasto energético do circuito integrado, não será detalhado a implementação e a eficiência dessas fontes, podendo-se analisar os trabalhos referenciados.

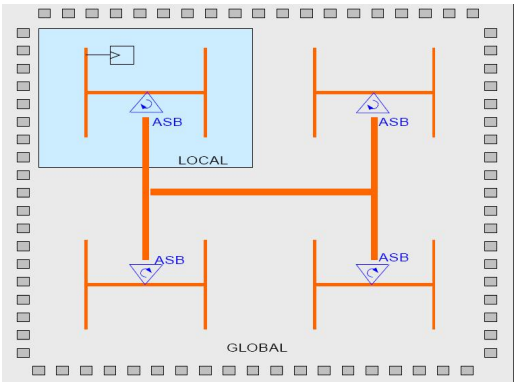

Figura 2.23: ASB's dentro do circuito integrado.
(LOGIC, 2010)

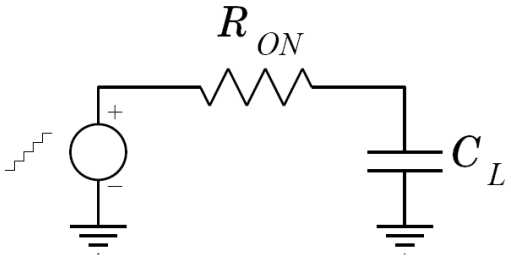

Figura 2.24: Exemplo de fonte em formato de escada.
(YOUNIS, 1994)

3 ANÁLISE ELÉTRICA

Neste Capítulo apresenta-se as ferramentas utilizadas para a simulação, bem como o resultado da análise elétrica de cada família lógica. Apresenta-se resultados de funcionamento, desempenho e consumo para portas lógicas simples.

3.1 Ferramentas utilizadas

A ferramenta utilizada para simulação elétrica foi o SpiceOpus (LJUBLJANA, 2010), enquanto o modelo de produção dos transistores foi obtido do projeto acadêmico PTM (ASU, 2010). As origens e motivos para utilização dessas ferramentas serão explicadas a seguir.

3.1.1 SpiceOpus

O SpiceOpus é uma ferramenta de simulação de circuitos elétricos gratuita. O projeto foi desenvolvido pela Faculdade da Engenharia Elétrica da University of Ljubljana, Slovenia e é uma junção dos códigos do SPICE 3f4 da University of California, Berkeley com o código do Xpice do Georgia Tech Research Institute, além de terem sido feitas diversas melhorias e correção de problemas (*bugs*). Escolheu-se essa ferramenta por ser de fácil acesso e ser multiplataforma, além de ser uma ferramenta confiável, com muitos anos de desenvolvimento. Para um melhor desenvolvimento e entendimento da ferramenta, fez-se necessária a leitura do livro (TUMA; BURMEN, 2009), que mostra com riqueza de detalhes todas as funções de simulação e comandos aceitos pelo SpiceOpus.

3.1.2 Predictive Technology Model (PTM)

O Predictive Technology Model (PTM) (ASU, 2010) é um estudo realizado na Arizona State University (ASU), nos Estados Unidos, que faz modelos de transistores CMOS que estão por vir, ou seja, pode-se desenvolver projetos eletricamente com transistores que serão fabricados, antevendo-se dificuldades e soluções para as mais diversas aplicações. A escolha desse modelo foi basicamente por isso, para obter um resultado que pode ser utilizado no futuro. Para a simulação dos circuitos, utilizou-se os modelos de 45nm para alto desempenho.

3.2 Avaliação do consumo de energia

Para a avaliação do consumo de energia foi utilizada a fórmula de cálculo de consumo médio para a fonte do sistema. A potência é dada pela Equação 3.1, onde P é a potência, T é o período, i é a corrente, v é a tensão e t é o tempo.

$$P = \frac{1}{T}\int_0^T i(t) * v(t)\, dt \tag{3.1}$$

Como as fontes utilizadas são as disponíveis na ferramenta, se pressupõe que são ideais e que aceitem entrada de corrente de volta do circuito. Será calculado o consumo de uma porta lógica inversora por um período na frequência de 250 MHz. Como anunciado anteriormente, não será tratado o consumo das fontes. Não serão feitas comparações com CMOS padrão nessa etapa, por se tratar de uma porta lógica muito simples, podendo haver distorções. Para todas simulações se define: a carga da sáida será um inversor CMOS padrão com tamanho quatro vezes maior que o tamanho mínimo e todos transistores dos circuitos serão utilizados com tamanho mínimo.

3.3 Análise de funcionamento e consumo

Nessa seção serão apresentados os resultados obtidos das simulações das famílias lógicas adiabáticas apresentadas no Capítulo 2.

3.3.1 Famílias lógicas não vistas nesse Capítulo

Como pode ser observado a seguir, nem todas famílias lógicas são cobertas nesse Capítulo. Quanto às famílias TSEL e SCAL, bem parecidas entre si, não foi possível reproduzi-las na tecnologia utilizada do PTM, devido à tensão de funcionamento e a tensão de *threshold* serem muito próximas (0.5V e 1.0V), dando pouca margem para um correto funcionamento do circuito.

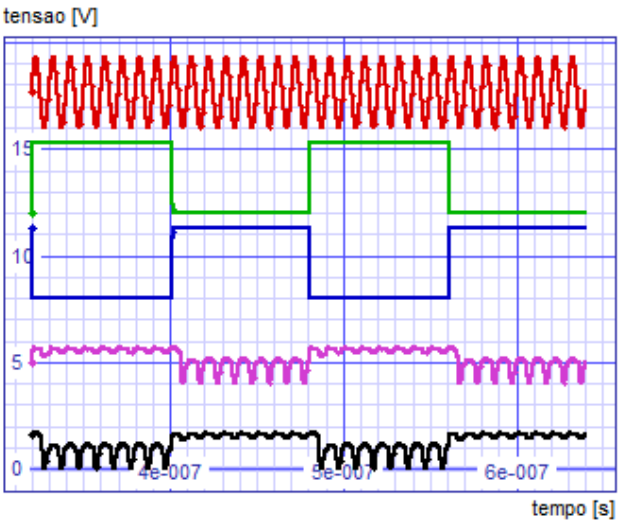

Figura 3.1: Funcionamento de um inversor TSEL NMOS.

Conforme mostrado anteriormente, o circuito passa um valor de diferença entre a tensão do circuito e a tensão de *threshold* para a saída, sendo esse valor aumentado pelos transistores cruzados. Como esse valor é muito pequeno nesse caso, isso não funciona. Para provar o funcionamento do circuito, foi implementado um inversor em TSEL PMOS e NMOS utilizando um modelo MOSIS de fabricação TSMC 0.18n, com tensão do circuito de 3.3V e tensão de *threshold* de 0.4V, dando bastante margem para um correto funcionamento.

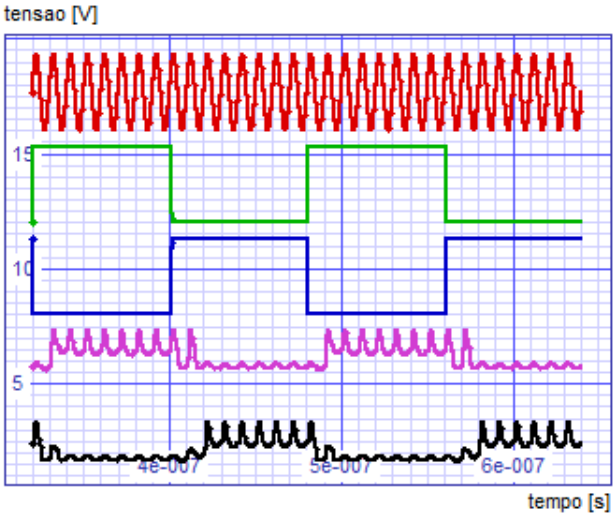

Figura 3.2: Funcionamento de um inversor TSEL PMOS.

Nas Figuras 3.1 e 3.2, os sinais são, de cima para baixo: fonte do sistema, entrada e entrada negada, saída e saída negada. Como pode-se ver, o circuito PMOS tem sua saída estável quando a fonte está próxima de '1' e o circuito NMOS tem sua saída estável quando a fonte está próxima de '0', possibilitando o cascateamento entre eles com somente uma fonte. No entanto, como o funcionamento não acontece para o modelo escolhido, essas famílias lógicas não serão avaliadas em questão de desempenho, consumo e área. Quanto à família SCRL não foi desejado o teste e implementação devido à grande dificuldade de cascateamento e controle como pode ser visto na Figura 3.3, com muitas penalidades em nível de lógica para controle e nenhuma vantagem de consumo em relação a outras famílias lógicas.

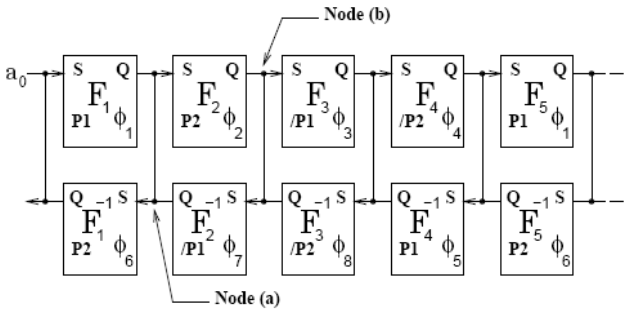

Figura 3.3: Cascateamento de portas lógicas SCRL com suas diversas fontes.
(YOUNIS, 1994)

3.3.2 PAL

PAL é uma das famílias lógicas mais simples, mas que exige alguns cuidados na hora de implementá-la. É necessária a existência de capacitância na saída para o correto funcionamento e a fonte de *clock* precisa ser senoidal e mais rápido que a frequência de alteração das entradas (se o *clock* for mais rápido vai acontecer mais cargas e descargas por ciclo, mas se for mais lento não conseguirá alterar a saída). Utilizando entradas como as que o PAL utiliza na saída (que seriam utilizadas no próximo estágio), esse estilo lógico

não funcionou na frequência exigida. Somente para referência, será avaliado o circuito funcionando a 25 MHz.

Pela ordem, de cima para baixo, podemos ver os seguintes sinais na Figura 3.4: fonte do circuito, entrada normal, entrada negada, saída normal e saída negada. Como se pode ver, o circuito funciona conforme o esperado, invertendo o sinal de entrada.

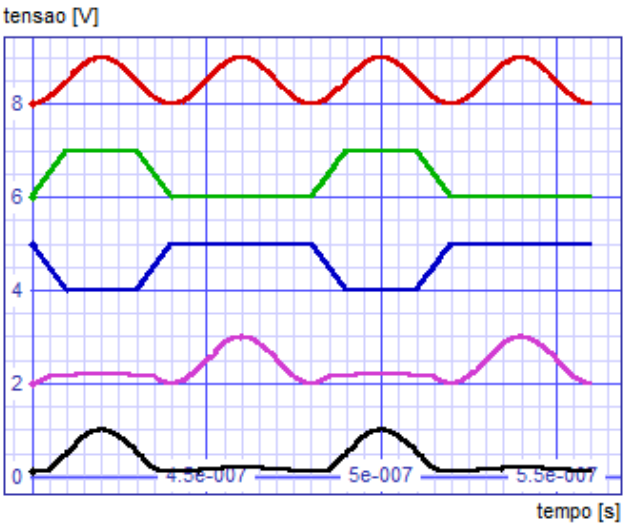

Figura 3.4: Gráfico com o funcionamento do inversor PAL.

A recuperação de energia também não é ideal, gastando-se energia. O gasto calculado foi $4.677e^{-16}$W, sendo superior a outras famílias lógicas, funcionando em uma frequência inferior. A medida foi feita para o período entre 440ns e 480ns que pode ser visto na Figura 3.5.

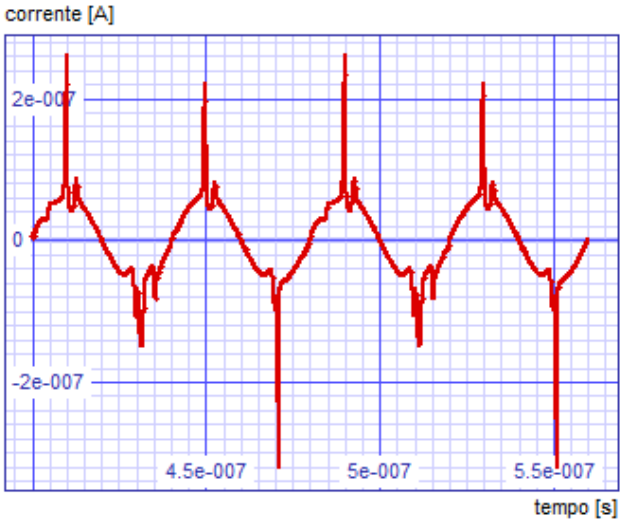

Figura 3.5: Gráfico com o corrente utilizada pelo inversor PAL.

3.3.3 CAL

CAL é uma família lógica pouco mais complexa, exigindo um controle de um *clock* para o cascateamento, que por causa dele, se necessita somente de uma fonte. Pela ordem, de cima para baixo, podemos ver os seguintes sinais na Figura 3.6: *clock* de controle,

fonte do circuito, entrada normal, entrada negada, saída normal e saída negada. Quando o *clock* de controle não está ativo, a saída é mantida pelos inversores de memorização, funcionando conforme o esperado.

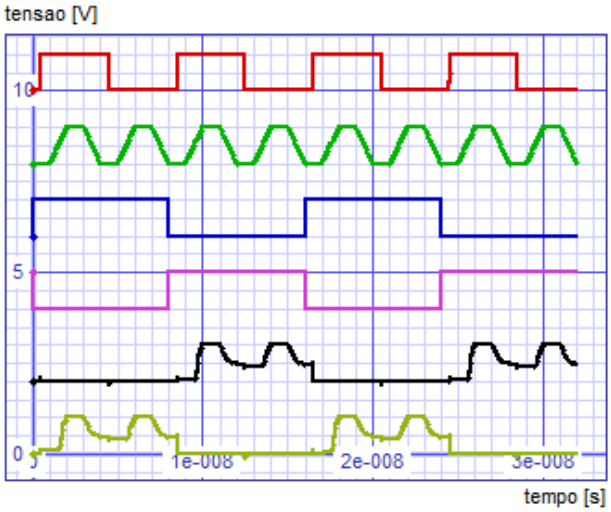

Figura 3.6: Gráfico com o funcionamento do inversor CAL - com *clock*.

No entanto, para a avaliação do consumo, será feita a situação em que o *clock* está ativo, conforme a Figura 3.7, onde o *clock* de controle está sempre em '1'.

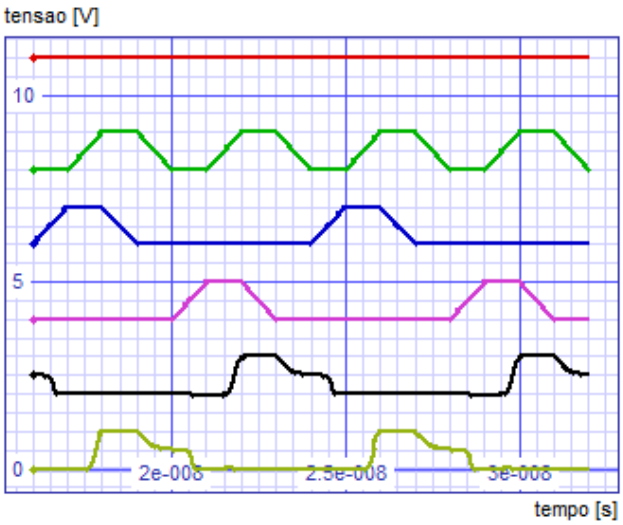

Figura 3.7: Gráfico com o funcionamento do inversor CAL - sem *clock*.

Conforme previsto, a transição não é completamente adiabática, retornando somente parte da carga de volta para a fonte. O cálculo do consumo ficou em $3.664e^{-16}$W para o período determinado, medido entre 20ns e 24ns, que pode ser visto na Figura 3.8.

3.3.4 ECRL (2N2P)

ECRL tem menos controle que CAL, mas em compensação necessita de quatro fontes para um correto cascateamento. No funcionamento da porta lógica, deve-se cuidar para que a entrada esteja estável quando a fonte está indo de '0' para '1'. Pela ordem, de cima para baixo, podemos ver os sinais do inversor: fonte do circuito, entrada normal, entrada

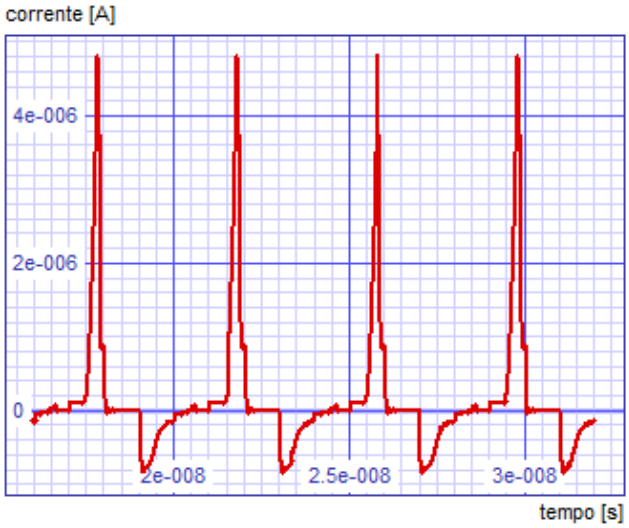

Figura 3.8: Corrente utilizada pela fonte para o inversor CAL.

negada, saída normal e saída negada. O funcionamento é conforme o esperado, visto que a saída está estável quando a fonte está em '1'.

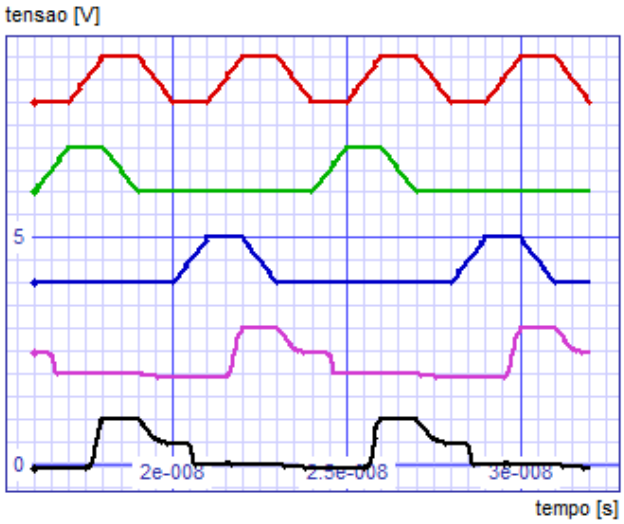

Figura 3.9: Gráfico com o funcionamento do inversor ECRL.

A transição também não é completamente adiabática, retornando somente parte da carga de volta para a fonte. O consumo de energia ficou em $3.883e^{-16}$W para o período determinado, medido entre 20ns e 24ns. Como pode ser visto na Figura 3.10, o resultado é bem parecido com CAL, inclusive em consumo.

3.3.5 2N-2N2P

Com um funcionamento muito parecido com ECRL, mas com um uso de um par de inversores ao invés de um par de transistores PMOS para a memorização, o 2N-2N2P teve um bom desempenho de consumo entre as famílias lógicas analisadas. De cima para baixo, temos os sinais: fonte do circuito, entrada normal, entrada negada, saída normal e saída negada. O funcionamento do circuito está dentro do esperado, invertendo o valor as entradas.

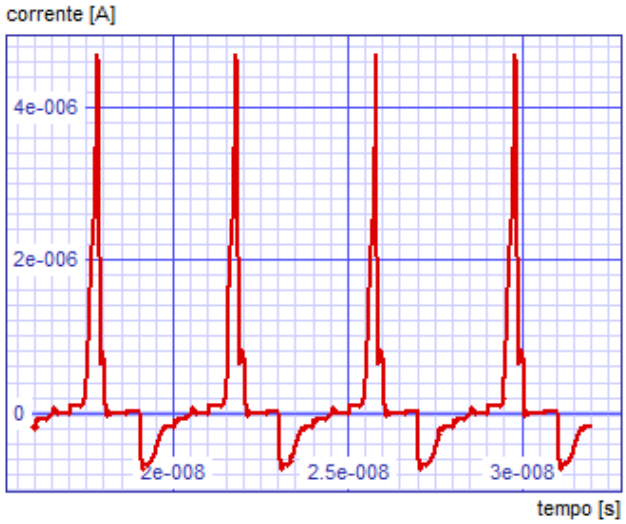

Figura 3.10: Corrente utilizada pela fonte para o inversor ECRL.

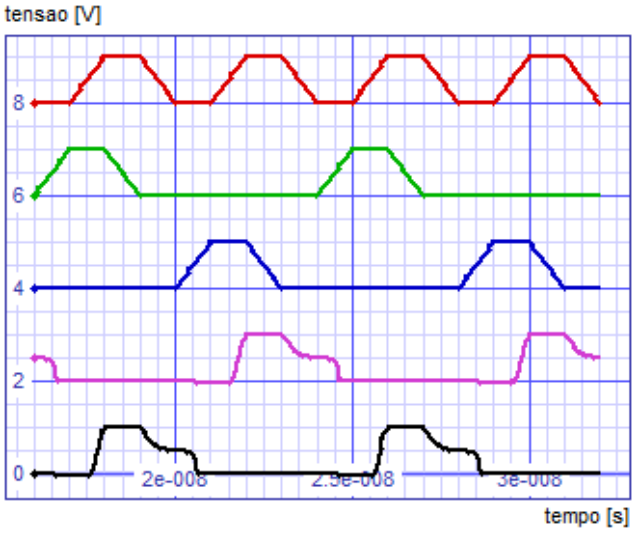

Figura 3.11: Gráfico com o funcionamento do inversor 2N-2N2P.

A transição também não é completamente adiabática, conforme a Figura 3.12 mostra, retornando somente parte da carga de volta para a fonte. Mesmo assim, obteve-se um bom resultado, ficando em torno de $3.719e^{-16}$W para o período determinado.

3.3.6 PFAL

PFAL é uma família lógica alternativa, usando a rede lógica ligada na fonte, e ao mesmo tempo muito parecida com 2N-2N2P, por usar um par de inversores. Todavia, o formato de funcionamento de PFAL deixa a saída por mais tempo em '1' do que necessário, gerando um gasto extra desnecessário. Isso torna essa família a que mais consome energia entre as avaliadas. Na Figura 3.13 temos o gráfico com o correto funcionamento do circuito, sendo os sinais, de cima para baixo: fonte do circuito, entrada normal, entrada negada, saída normal e saída negada.

A transição não é completamente adiabática, retornando somente parte da carga de volta para a fonte, sendo o pior resultado de consumo entre os circuitos medidos, ficando em $5.043e^{-16}$W para o período determinado entre 20ns e 24ns, que pode ser visto na

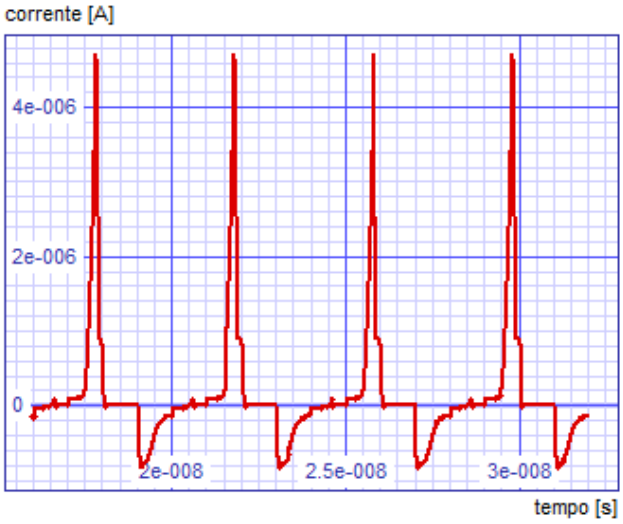

Figura 3.12: Corrente utilizada pela fonte para o inversor 2N-2N2P.

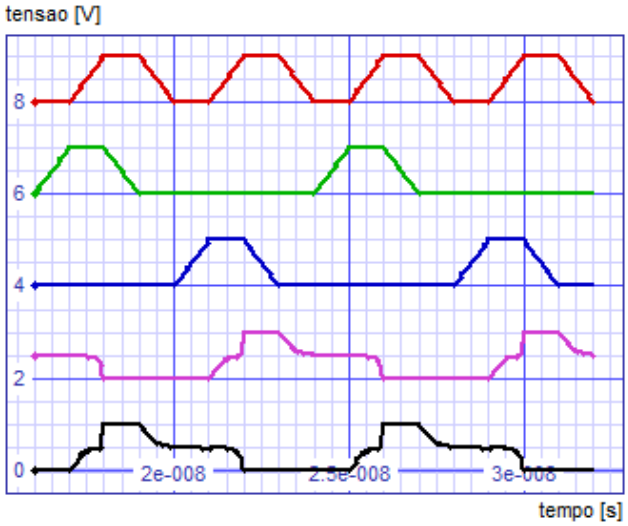

Figura 3.13: Gráfico com o funcionamento do inversor PFAL.

Figura 3.14.

3.3.7 Comparação entre famílias adiabáticas

Na Tabela 3.3.7 tem-se a comparação entre as famílias lógicas implementadas, com o consumo de um período de um inversor. Como pode-se ver, o melhor resultado foi obtido com CAL, com a ressalva de que o controle para cascateamento não foi implementado. O pior resultado foi obtido com PFAL, carregando a saída por mais tempo que o necessário e causando isso. PAL teve um consumo superior a outras famílias lógicas sem funcionar em frequências mais altas, tornando uma família lógica praticamente descartada. PFAL, ECRL e 2N-2N2P tem o mesmo problema de necessitar de quatro fontes para um correto cascateamento, embora 2N-2N2P tenha obtido o melhor resultado entre essas famílias.

Para avaliar o desempenho em relação ao atraso na saída, mostra-se na Figura 3.15 as saídas dos inversores CAL, ECRL, 2N-2N2P e PFAL. O circuito que tem o maior atraso, demorando tanto de ’0’ para ’1’ quanto de ’1’ para ’0’ é o PFAL. CAL e 2N-2N2P tem um resultado muito parecido, o que é esperado visto que, com o transistor de controle do

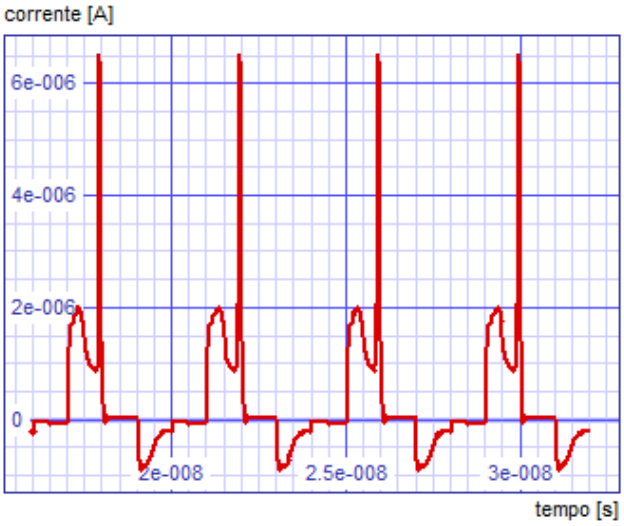

Figura 3.14: Corrente utilizada pela fonte para o inversor PFAL.

Família	Frequência	Consumo
PAL	25MHz	$4.677e^{-16}$W
CAL	250MHz	$3.664e^{-16}$W
ECRL	250MHz	$3.883e^{-16}$W
2N-2N2P	250MHz	$3.719e^{-16}$W
PFAL	250MHz	$5.043e^{-16}$W

Tabela 3.1: Comparação de consumo de um inversor entre famílias lógicas adiabáticas.

CAL ligado, o funcionamento dos dois circuitos deveria ser igual. Por fim o ECRL, que tem uma queda de tensão antes de subir o nível lógico muito rapidamente e descendo o nível lógico mais rápidamente que todos.

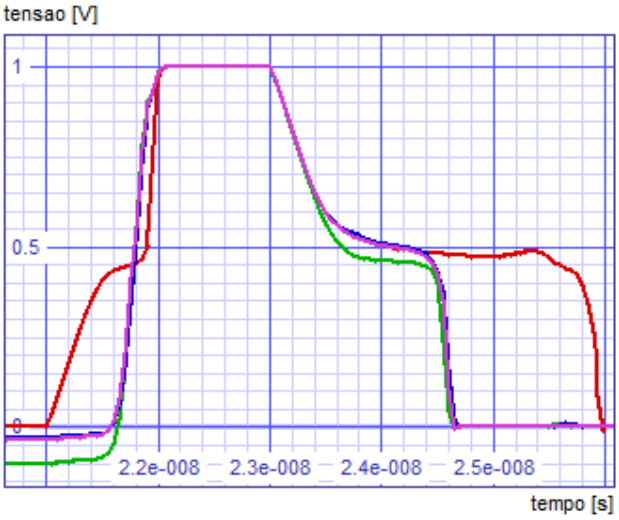

Figura 3.15: Comparação de atrasos entre inversores adiabáticos.

4 SOLUÇÕES DE LEIAUTE PARA LÓGICA ADIABÁTICA

Nesse Capítulo serão tratadas possíveis soluções de leiaute para criação de uma biblioteca de portas lógicas, visando um formato de desenvolvimento de circuitos integrados mais ágil, embora menos eficiente em questão de área, conhecido como *standard cell* (DUNLOP; KERNIGHAN, 1985), ou célula padrão. Nesse método de desenvolvimento, ao invés de se desenhar cada transistor do circuito, usa-se um arquivo de entrada em linguagem de descrição de hardware (como VHDL (ASHENDEN, 2002) ou Verilog (THOMAS; MOORBY, 2008)) e esse arquivo é sintetizado e processado por uma série de ferramentas para posicionar as portas lógicas, mapeá-las, rotear os sinais, entre outras etapas, dependendo do projeto. As portas lógicas utilizadas por essas ferramentas são descritas em uma biblioteca formato Liberty (SYNOPSYS, 1999), onde se descreve a porta lógica com sua função, capacidade de carga, área, tensão, etc. Também é necessário o leiaute da própria porta lógica, com suas de entradas e saídas, para a posterior colocação pela ferramenta. Porém, para facilitar a tarefa da ferramenta e ela poder colocar corretamente todas portas lógicas, se limita a altura da célula padrão pela porta lógica que mais ocupa essa dimensão na biblioteca. Isso causa um certo desperdício de área para portas lógicas mais simples, que não necessitariam de tanto espaço na altura.

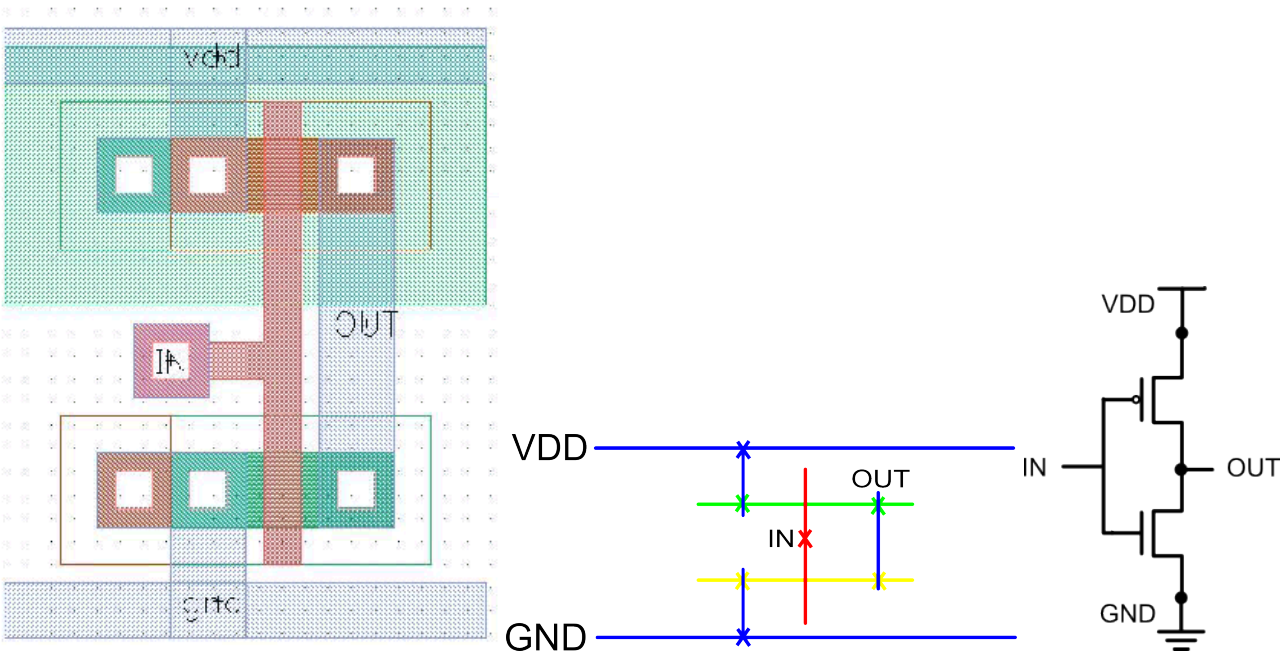

Figura 4.1: Exemplo de um leiaute de um inversor CMOS padrão e o leiaute simbólico e esquemático equivalente.

Portanto, dentro de cada família lógica será apresentada sua característica e o que pode ser explorado em nível de projeto do circuito para minimizar a altura de uma célula padrão. Para representação será feito um leiaute simbólico, com linhas para cada parte do processo, onde verde é a região ativa P, amarelo é a região ativa N, vermelho é o polissilício, azul é o metal e os 'x' são os contatos entre as camadass. Um exemplo de um leiaute simbólico com o leiaute real e o esquemático referente pode ser visto na Figura 4.1. A família lógica SCRL não será discutida por não apresentar um padrão de desenvolvimento como as outras famílias, tendo um formato de circuito muito parecido com um CMOS convencional. Em geral, a diferença de área em relação a um circuito CMOS foi pequena, visto que nas famílias lógicas adiabáticas vistas existem duas redes lógicas, sendo uma normal e uma negada, enquanto no CMOS padrão tem-se duas redes lógicas também, mas complementares em questão de transistores.

4.1 PAL e ECRL (2N2P)

Essas duas famílias lógicas adiabáticas, PAL e ECRL (2N2P), tem muito em comum e serão tratados como somente um caso. Eles se valem de redes lógicas NMOS e um par de transistores PMOS para efetuar a função de memorização das saídas.

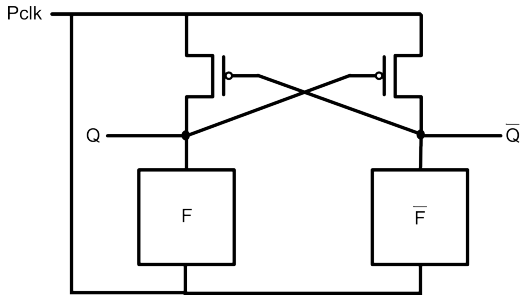

Figura 4.2: Característica da família lógica PAL.

A diferença básica entre PAL e ECRL (2N2P) é a utilização da fonte, visto que PAL se vale da fonte tanto para rede NMOS quanto para o par de transistores NMOS, como pode ser visto nas Figuras 4.2 e 4.3. A rede NMOS é facilmente construída utilizando uma técnica conhecida como caminho de Euler, que pode ser vista em (UEHARA; VAN-CLEEMPUT, 1981). Essa técnica acha o caminho ótimo para a elaboração do leiaute do lado PMOS ou NMOS. Nos circuitos CMOS existe uma dificuldade de encaixar as entradas devido aos leiautes PMOS e NMOS terem aspectos diferentes. Como nos circuitos adiabáticos apresentados aqui tem-se somente PMOS ou NMOS, o leiaute fica facilitado.

O leiaute em si fica muito simplificado, como pode ser visto na Figura 4.4, colocando-se o par de transistores PMOS no centro e uma rede lógica NMOS em cada lado. A altura mínima fica bastante reduzida, sendo que isto pode ser definido pela quantidade de faixas de uma mesma camada na horizontal, devido a regras de tamanho mínimo de camada e de distância mínima entre camadas. Nesse caso, a limitação é dada pela camada de metal, com quatro faixas. No entanto, é necessário uma entrada extra de tensão para o substrato dos transistores PMOS. Uma dificuldade extra encontrada no leiaute de todas famílias adiabáticas é a quantidade de alimentações necessária, tendo de ser revista a ideia de somente uma linha de alimentação para a célula padrão.

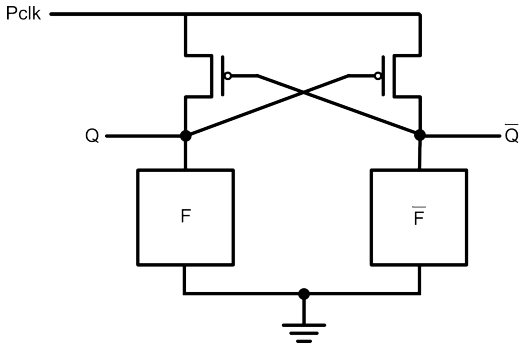

Figura 4.3: Característica da família lógica ECRL e 2N2P.

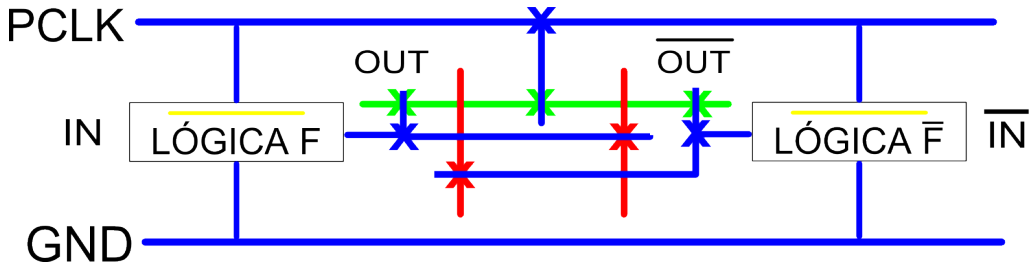

Figura 4.4: Solução de leiaute para PAL, ECRL e 2N2P.

4.2 CAL, 2N-2N2P e PFAL

Outras três famílias lógicas com leiaute muito parecido entre si são CAL, 2N-2N2P e PFAL. Também tem rede lógica NMOS, mas para memorizar o resultado utiliza-se um par de inversores.

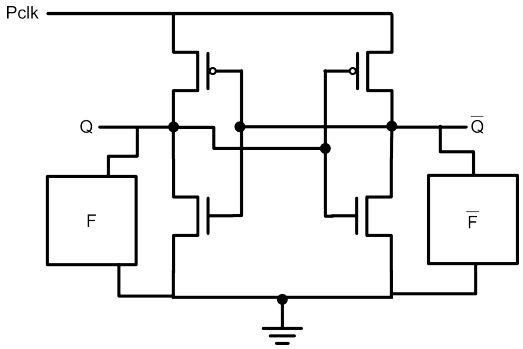

Figura 4.5: Característica da família lógica 2N-2N2P.

A diferença entre essas famílias lógicas está na posição das redes lógicas NMOS. Como pode ser visto nas Figuras 4.5 e 4.6, no 2N-2N2P as redes lógicas estão ligando as saídas com o terra, enquanto no PFAL as saídas passam pelas árvores NMOS até chegar na fonte.

Apesar dessas diferenças, o leiaute fica bem parecido, como pode ser visto na Figura 4.7. A diferença que pode ser colocada seria a não necessidade de ligar a fonte nas redes lógicas NMOS no caso do 2N-2N2P. Nesse caso a existência de duas linhas de transistores se faz necessária devido ao par de inversores. Contudo, é possível utilizar duas linhas de transistores nas redes lógicas NMOS, aproveitando essa área extra. O roteamento dos sinais tem uma entrada extra de tensão para o bulk dos transistores PMOS, ficando bem

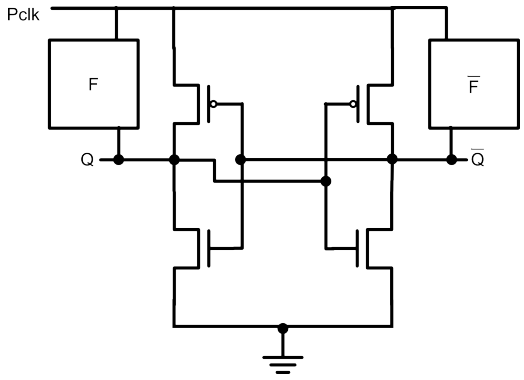

Figura 4.6: Característica da família lógica PFAL.

parecido com o do leiaute da seção anterior.

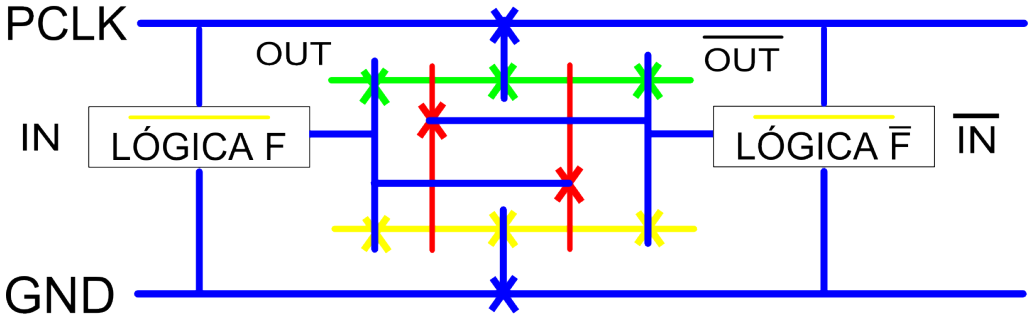

Figura 4.7: Solução de leiaute para 2N-2N2P e PFAL.

CAL pode ser incluído nessa seção porque também é bem parecido, tendo somente dois transistores a mais de controle, como pode ser visto na Figura 4.8.

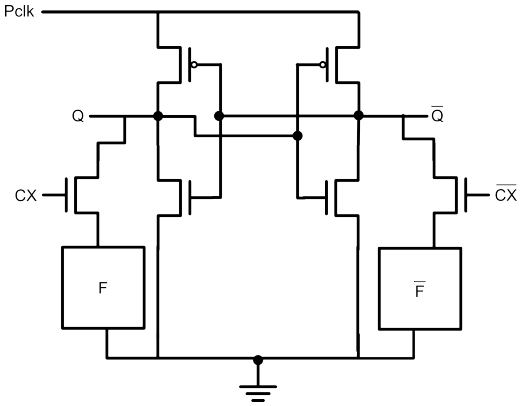

Figura 4.8: Característica da família lógica CAL.

Mesmo com dois transistores a mais, o leiaute fica parecido, pois os dois transistores a mais ocupam pouco espaço e podem compartilhar a mesma região ativa que os outros transistores NMOS, como se vê na Figura 4.9. Usando CAL dificultaria um pouco mais o roteamento, por necessitar de dois sinais a mais de controle por porta lógica. Em todos os casos, a altura mínima seria a mesma, determinada pelas quatro faixas de metal.

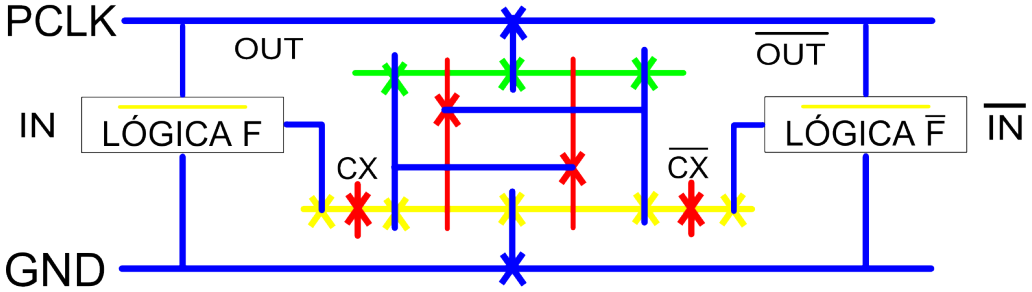

Figura 4.9: Solução de leiaute para CAL.

4.3 TSEL

A família TSEL é uma das que tem um menor consumo, aliada com uma facilidade de implementação por necessitar de somente uma fonte e o cascateamento ser facilitado por intercalar portas lógicas NMOS com portas PMOS. Essa família usa poucos transistores como base e todos do mesmo tipo, facilitando o leiaute.

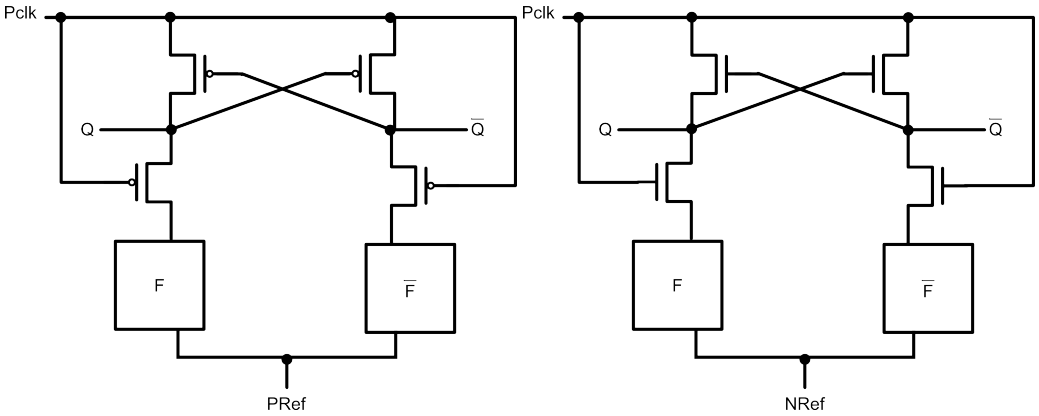

Figura 4.10: Característica da família lógica TSEL.

Como pode ser visto na Figura 4.10, são necessários dois transistores para efetuar a memorização das saídas e dois transistores de controle. Isso, no leiaute, consegue ser feito com uma linha de transistores, NMOS ou PMOS. O problema do bulk continua e é aumentado para a porta NMOS também, visto que a tensão de referência não é nem igual ao terra nem igual a tensão do sistema. Na Figura 4.11 exemplifica-se um leiaute utilizando transistores PMOS, mas essa abordagem serve igualmente para transistores NMOS. A altura mínima também é determinada pelos quatro faixas de metal.

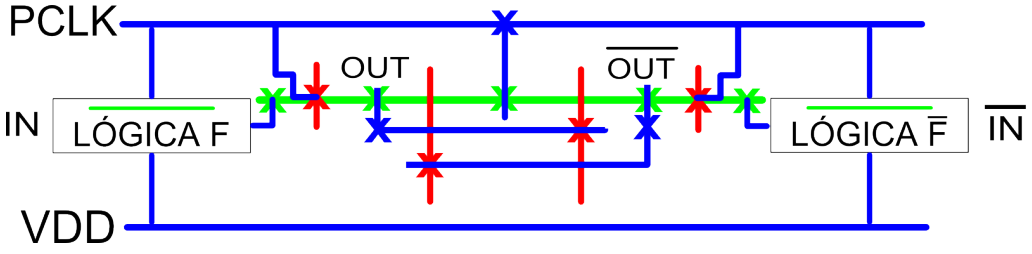

Figura 4.11: Solução de leiaute para TSEL - PMOS.

4.4 SCAL

A família SCAL é muito semelhante à família TSEL e pode ser considerada uma versão melhorada por controlar a tensão de referência com um sinal de entrada. Porém, esse sinal pode complicar o roteamento bem como o leiaute da porta lógica.

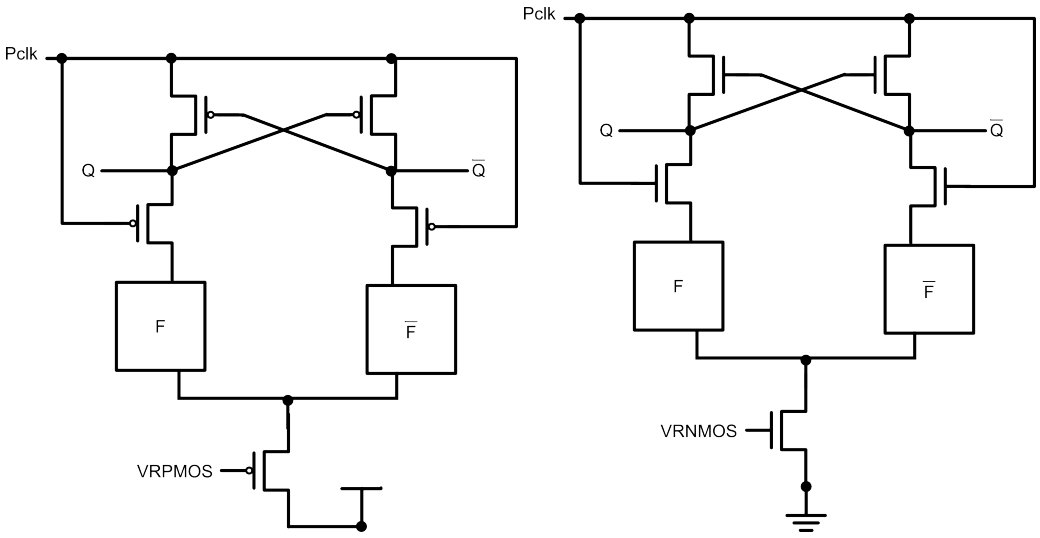

Figura 4.12: Característica da família lógica SCAL.

Usando a mesma abordagem utilizada para TSEL, a altura mínima continua determinada pelas quatro linhas de metal, mas existe um aumento na área e uma dificuldade maior no roteamento pois existem dois sinais e dois transistores a mais na porta lógica.

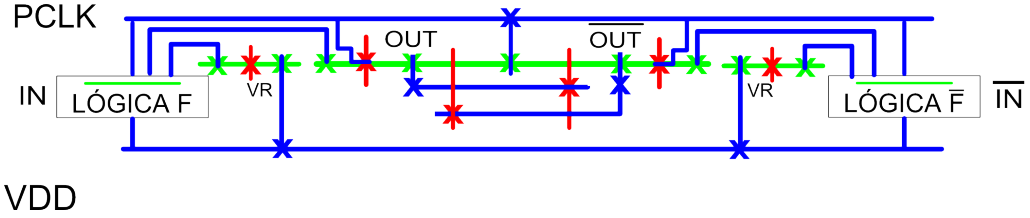

Figura 4.13: Solução de leiaute para SCAL - PMOS.

Uma alternativa que aumentaria o tamanho mínimo da porta lógica em uma faixa de metal, mas diminuiria a área pois diminuiria um transistor por porta lógica, facilitando também o roteamento, seria a da Figura 4.14. Ela usa um transistor extra, que necessariamente precisa estar em outra linha, gerando uma altura mínima de cinco linhas de metal. Assim como para o TSEL, exemplifica-se um leiaute utilizando transistores PMOS, mas essa abordagem serve igualmente para transistores NMOS.

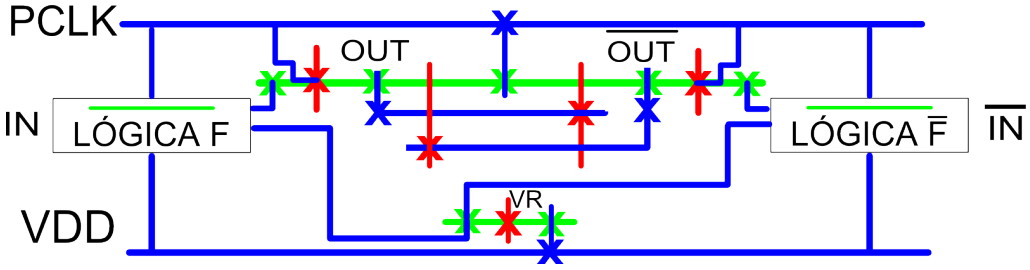

Figura 4.14: Solução alternativa de leiaute para SCAL - PMOS.

5 COMPARAÇÃO DE IMPLEMENTAÇÃO DE UM SOMADOR COMPLETO

Um somador completo, ou um *full-adder*, é um somador de 1 bit, que soma dois valores mais um carry de entrada (Cin) ou vem-um, ou seja, soma três bits. Esse somador tem duas saídas: o resultado da soma e um carry de saída (Cout), ou vai-um. Uma implementação simples, levando em conta somente as entradas, seria: $soma = (a \oplus b) \oplus cin$ e $Cout = (a \cdot b) + (cin \cdot (a \oplus b))$, de acordo com a Tabela 5.

Entradas			Saídas	
A	B	Cin	Soma	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tabela 5.1: Tabela verdade de um somador completo.

Esse somador é uma das portas lógicas complexas mais utilizadas nos circuitos integrados, além de utilizar funções lógicas básicas: E, OU e Ou-exclusivo. Por isso, será o circuito escolhido para avaliação. Serão avaliados três circuitos: um CMOS padrão, um CMOS operando em *subthreshold* e um somador completo implementado com 2N-2N2P, um dos circuitos adiabáticos com resultado satisfatório no Capítulo 3. Novamente será utilizado o SpiceOpus com o modelo PTM de 45nm para alto desempenho, uma frequência de 250 MHz e uma carga na saída de um inversor com tamanho quatro vezes maior que o tamanho mínimo da tecnologia, utilizado em todos outros transistores.

5.1 CMOS

Para poder ser feita uma comparação simples entre os circuitos, será implementado um somador completo com circuitos de soma e de Cout separados, como pode ser visto na Figura 5.1, o que facilitará depois a comparação com o circuito adiabático.

O funcionamento do circuito pode ser visto na Figura 5.2, de cima para baixo: as três entradas, a soma e o Cout. Foram utilizados atrasos de subida e descida na ordem de 100 vezes menores que o tamanho do período.

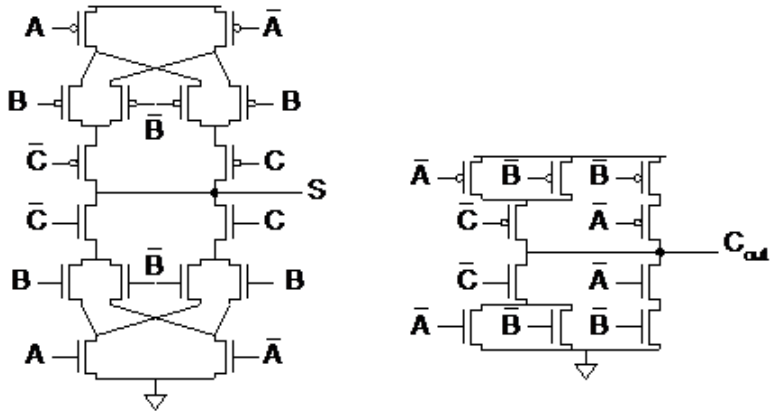

Figura 5.1: Implementação do somador completo.

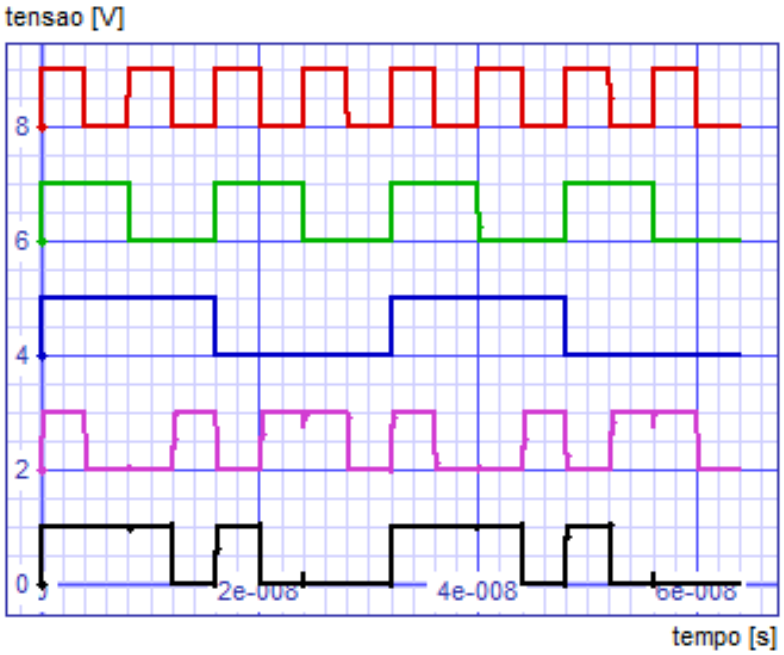

Figura 5.2: Gráfico com o funcionamento do somador completo CMOS.

Enquanto em um inversor o consumo pode ser avaliado simplesmente em um período de uma transição, no caso do somador completo isso não pode ser feito. Isso acontece porque, além de incluir dois circuitos distintos, cada transição tem um gasto de energia diferente devido ao consumo de capacitâncias parasitas dos transistores. Portanto, o consumo foi avaliado entre 64ns e 128ns, ou todas as 16 transições possíveis no circuito.

Como pode ser visto na Figura 5.3, existe parte da corrente que retorna para a fonte ou para o circuito anterior, efeito conhecido como *charge pump* (WU; CHANG, 1998). Como em circuitos normais essa corrente não é aproveitada, foi feita uma análise com essa energia reutilizada e outra com isso sendo desperdiçado, tendo uma diferença desprezível na medida e sendo pego o valor ignorando esse fato: $7.353e^{-15}$W.

Conforme o resultado obtido e a Figura 5.4, observa-se um circuito robusto, com bom desempenho e um consumo baixo. O modelo utilizado, mesmo sendo para maior desempenho, obteve um excelente resultado em questão de consumo, reafirmando que com a diminuição do tamanho do transistor o consumo também diminui.

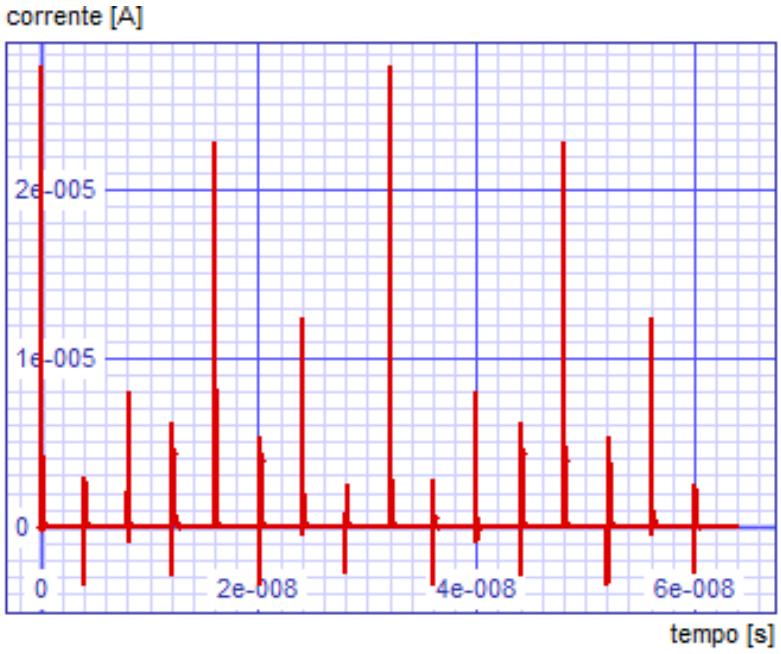

Figura 5.3: Gráfico com a corrente fornecida pela fonte para o somador completo CMOS.

5.2 *Subthreshold Logic*

O circuito utilizado para avaliar o funcionamento em modo *subthreshold* será o mesmo que o utilizado para CMOS padrão, mas com a tensão do sistema abaixo da tensão de *threshold*: 0.4V. No entanto, o circuito não funcionou na frequência de 250 MHz, devido a perda de desempenho que a tensão muito baixa leva, não conseguindo carregar a saída. Somente para comparação, foi feita uma análise do circuito funcionando a 25 MHz. A ordem dos sinais na Figura 5.5, de cima para baixo: as três entradas, a soma e o Cout, onde pode ser verificada a maior dificuldade de carregar a saída

Nota-se na Figura 5.6 que a ordem de grandeza da corrente diminuiu em relação ao circuito CMOS, o que implicará em uma diminuição no consumo, aliado à diminuição na tensão, de acordo com a Equação 3.1.

A grande vantagem desse estilo lógico é o consumo, visto que é bem inferior, chegando a $8.237e^{-16}$W para o período compreendido entre 0 e 640ns, como pode ser visto na Figura 5.7. Claro que isso só é válido onde essa perda de desempenho seja permitida. No entanto, se esse consumo for necessário, não seria alcançado com CMOS convencional, que gasta aproximadamente a mesma energia para uma frequência de operação mais baixa ou mais alta.

5.3 2N-2N2P

Para um correto cascateamento de um circuito 2N-2N2P é necessário o uso de quatro fontes intercaladas. Como o objetivo era avaliar o consumo e na ferramenta isso é feito através de uma fonte, simplificou-se isso utilizando dois circuitos com a mesma fonte, um para calcular a soma e um para calcular o Cout e por isso não se utilizou um somador completo mais eficiente. O formato do circuito pode ser visualizado na Figura 5.8, onde é separado entre soma e Cout.

Em ambos as Figuras 5.9 e 5.10 com o funcionamento do circuito, os sinais são, de

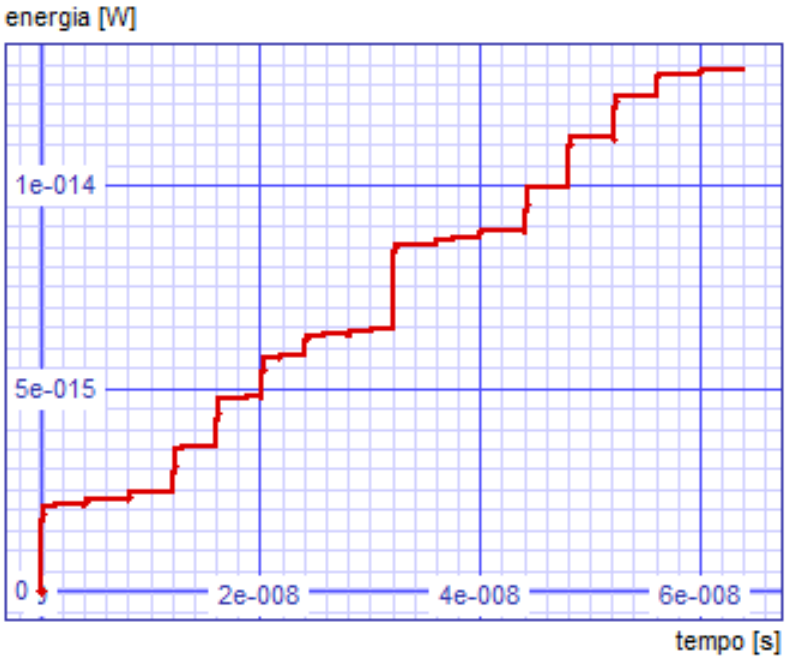

Figura 5.4: Gráfico com a energia fornecida pela fonte para o somador completo CMOS.

cima para baixo: fonte do circuito, as três entradas com seu valor negado, e a saída negada e a saída normal.

O formato da corrente gasta pela fonte é bem parecido com o do inversor, retornando parte da corrente para a fonte, caracterizando um circuito parcialmente adiabático.

O consumo, no entanto, não foi tão baixo quanto o esperado. Fazendo o cálculo da potência média para o período entre 64ns e 128ns, obteve-se o consumo de $7.791e^{-15}$W, aproximadamente o mesmo consumo que o CMOS convencional.

O gráfico do consumo na Figura 5.12 é interessante porque mostra o problema que ocorre para essa tecnologia: a tensão de *threshold* do transistor é muito alto se comparado a tensão do sistema, em torno de 50%. Como a recuperação da energia é feita até a tensão de *threshold*, isso é muito pouco considerando o custo energético de se ter duas redes de transistores com duas saídas e memorização. Em tecnologias como a utilizada para testar o TSEL na seção 3.3.1, onde a tensão do sistema é 3.3V e a tensão de *threshold* 0.4V, a recuperação de energia seria muito mais considerável, valendo a pena utilizar o circuito adiabático.

5.4 Comparação entre famílias lógicas

Na Tabela 5.4 podemos observar o resultado obtido nas simulações das seções anteriores. O CMOS obteve um excelente resultado, tratando-se de uma lógica amplamente utilizada e bastante robusta. O CMOS operando em nível *subthreshold* obteve o resultado esperado, tendo um funcionamento somente em uma frequência mais baixa, mas um consumo com uma ordem de grandeza inferior. O 2N-2N2P teve um consumo baixo, porém mais alto que o CMOS. Isso acontece pela pequena diferença entre a tensão do sistema e a tensão de *threshold*, recuperando pouca energia, além de existir um consumo extra, gerando sempre uma saída negada. Talvez o mais justo seria comparar com uma lógica CMOS *dual-rail*, ideia surgida após o término do trabalho.

Para possibilitar a comparação dos atrasos das saídas do somador completo, rodou-se

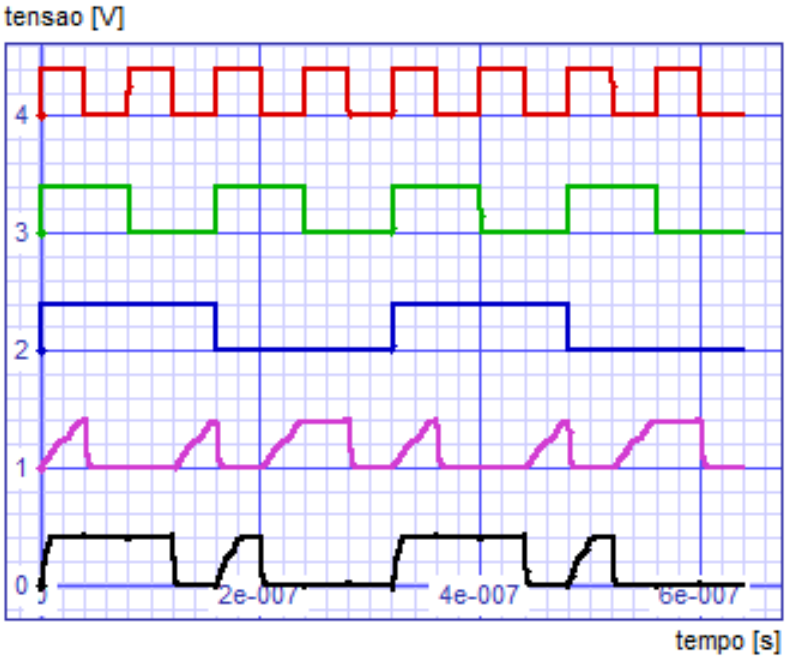

Figura 5.5: Gráfico com o funcionamento do somador completo em modo de operação *subthreshold*.

Lógica	Frequência	Consumo
CMOS	250MHz	$7.353e^{-15}$W
CMOS *Subthreshold*	25MHz	$8.237e^{-16}$W
2N-2N2P	250MHz	$7.791e^{-15}$W

Tabela 5.2: Comparação de consumo de um somador completo.

todos circuitos funcionando a 25MHz e multiplicou-se a saída do circuito funcionando em *subthreshold* para que todas saídas tivessem igualdade de amplitude. Nas Figuras 5.13 e 5.14 pode-se observar o CMOS robusto, funcionando sempre, o adiabático mantendo a saída válida somente por um período e o CMOS operando em *subthreshold* com muita dificuldade de carregar o inversor na saída. Dessa forma, o circuito que opera em nível *subthreshold* deveria ser implementado com uma frequência ainda menor para um correto funcionamento.

Na soma, cujo caminho crítico de transistores é de três transistores, teve um resultado pior em questão de atraso quanto do Cout, que tem um caminho crítico de dois transistores. Pode-se concluir então que quanto maior a rede de transistores do circuito, menor será a frequência máxima de funcionamento do circuito funcionando em nível *subthreshold*.

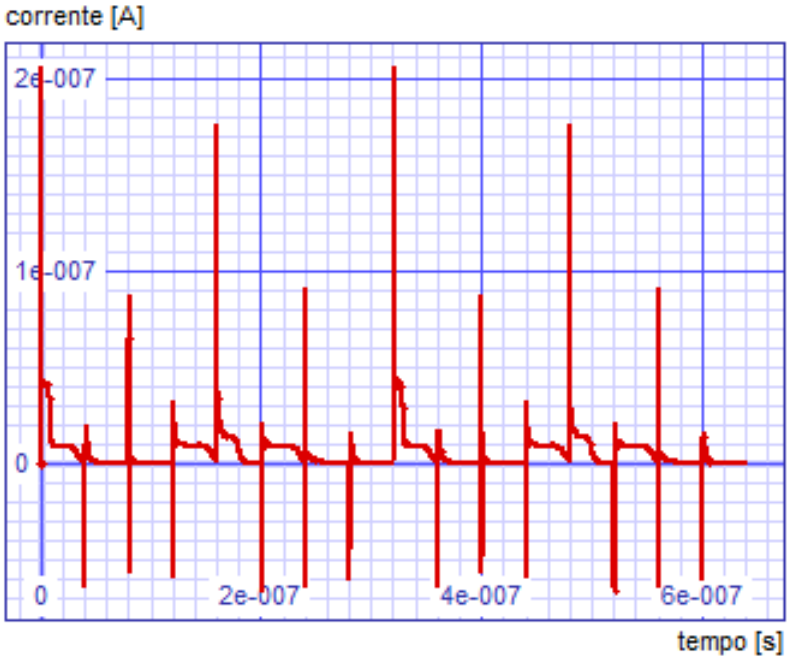

Figura 5.6: Gráfico com a corrente fornecida pela fonte para o somador completo em modo de operação *subthreshold*.

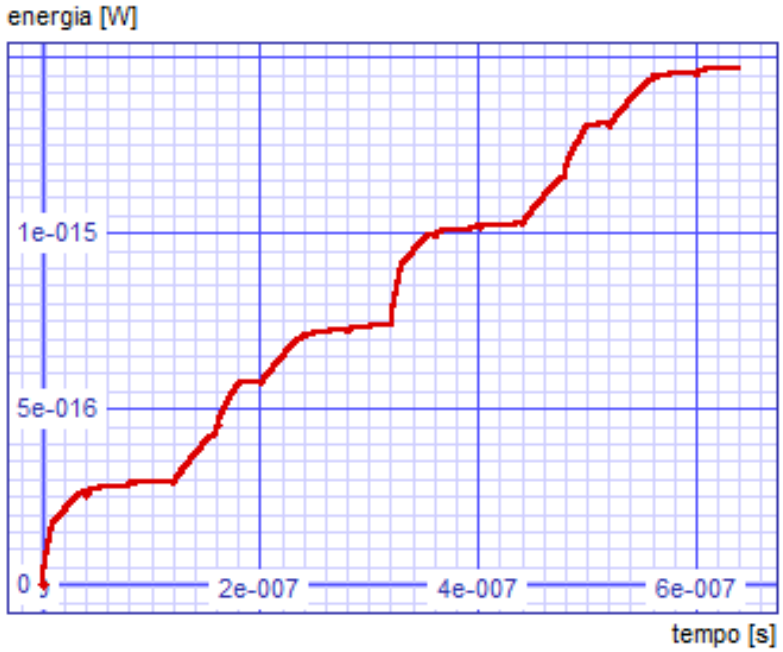

Figura 5.7: Gráfico com a energia fornecida pela fonte para o somador completo em modo de operação *subthreshold*.

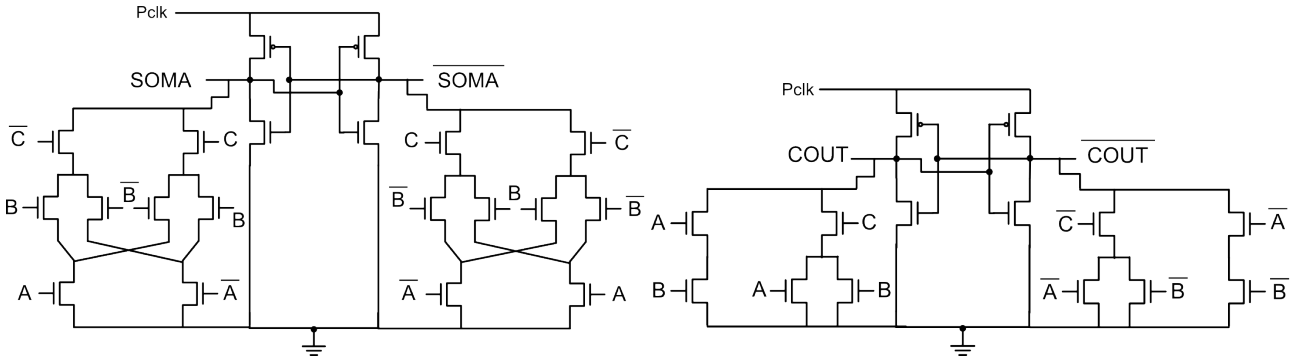

Figura 5.8: Implementação de um somador completo utilizando 2N-2N2P.

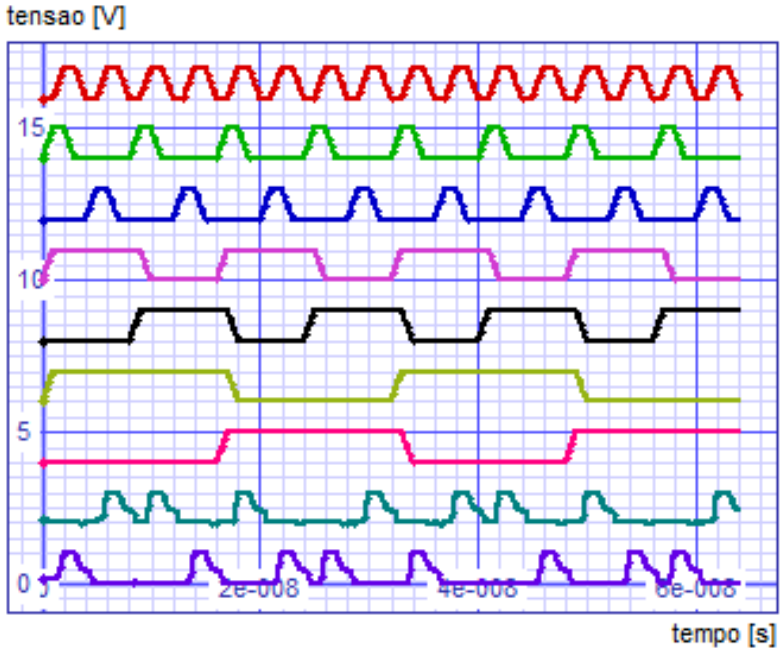

Figura 5.9: Gráfico com o funcionamento do somador completo 2N-2N2P - soma.

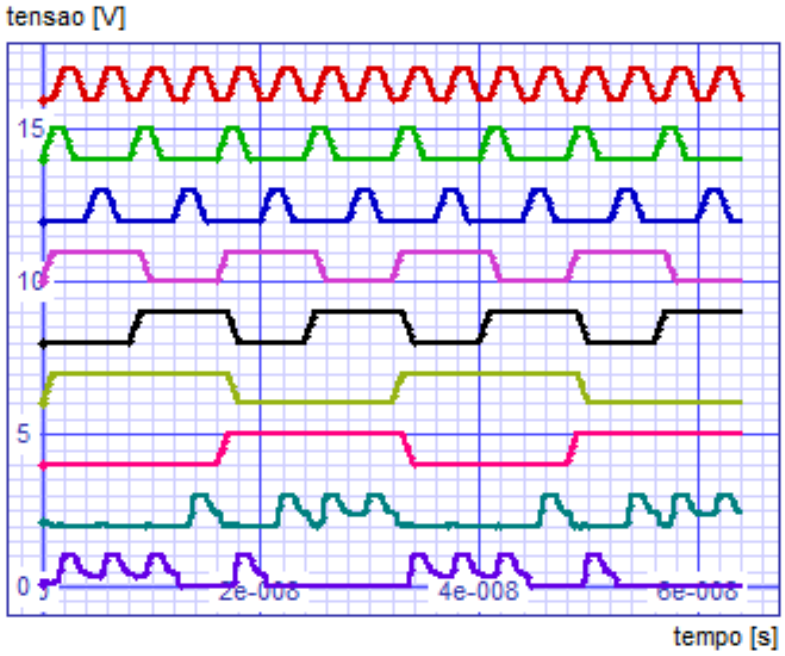

Figura 5.10: Gráfico com o funcionamento do somador completo 2N-2N2P - Cout.

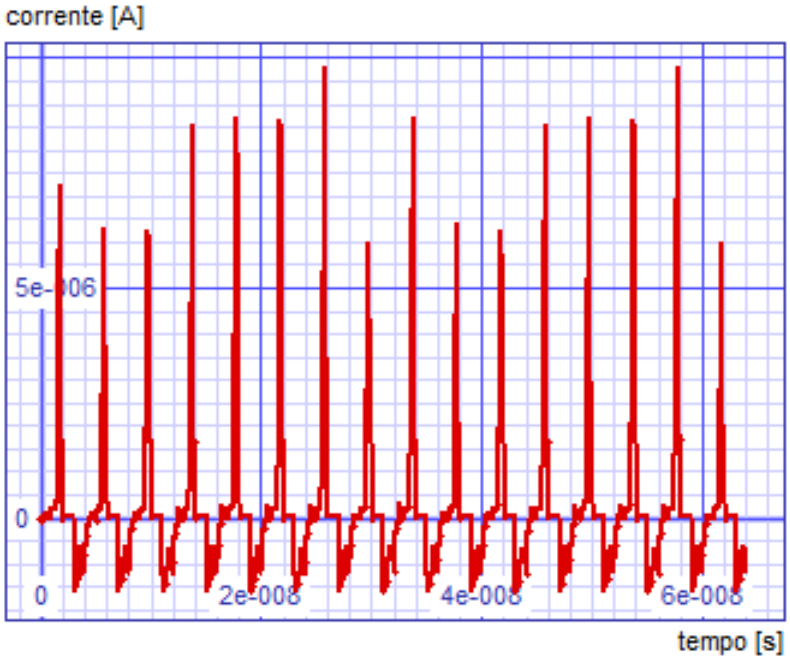

Figura 5.11: Gráfico com a corrente fornecida pela fonte para o somador completo 2N-2N2P.

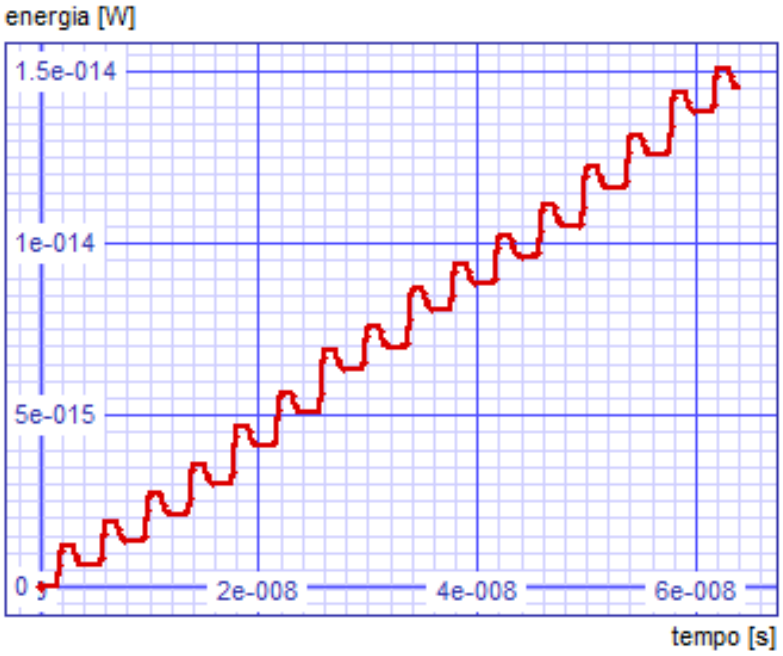

Figura 5.12: Gráfico com a energia fornecida pela fonte para o somador completo 2N-2N2P.

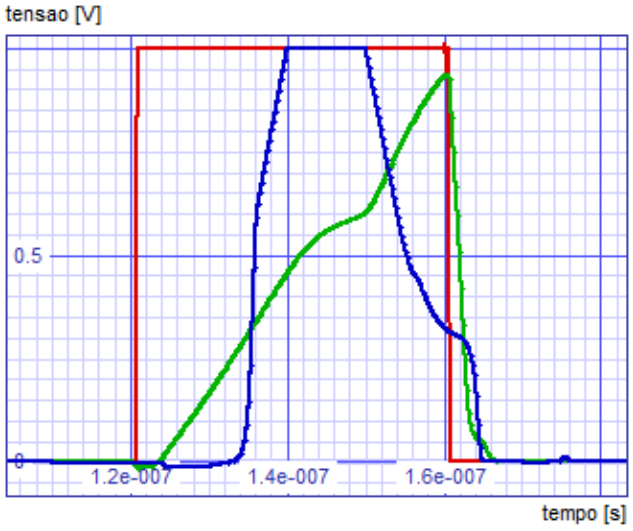

Figura 5.13: Comparação de atrasos entre implementações de soma.

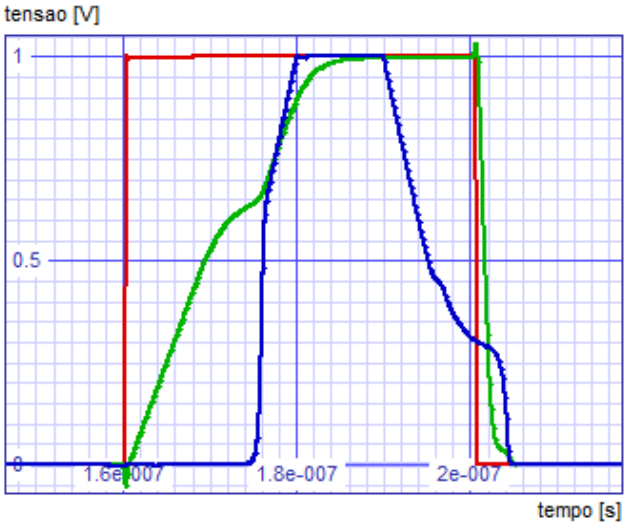

Figura 5.14: Comparação de atrasos entre implementações de Cout.

6 CONCLUSÃO

Com a diminuição do tamanho e da tensão de funcionamento dos transistores, torna-os mais rápidos e com menor consumo. Isso realmente é muito importante e deve continuar acontecendo até que um limite físico impeça esse avanço. Porém, tem-se uma nova fronteira também no desenvolvimento de circuitos que gastem menos energia. Os estudos feitos na década de 90 e 2000, que foram utilizados como base para fazer esse trabalho, já não são totalmente aplicáveis na tecnologia atual e muito menos na que está por vir, embora fossem de grande valia para a tecnologia da época. No entanto, a ideia ainda é totalmente válida, visto que existe uma empresa oferecendo soluções fechadas com redução de consumo de até 50%, reduzindo o consumo no resfriamento do circuito e no encapsulamento do circuito integrado. O que é necessário são mais estudos na área visando as novas tecnologias, novas alternativas para tensões mais baixas e métodos mais simples de cascateamento e controle dos circuitos. Quanto a utilização de células padrão, isso se limitaria a utilização de famílias lógicas com somente uma fonte, visto que com mais fontes dificultaria muito o cascateamento ou geraria um overhead grande, criando-se mais linhas de tensão, sendo que só seria usada uma por cada célula. Como existem trabalhos com somente uma fonte e com bons resultados, é possível pensar em uma biblioteca de células adiabáticas. O trabalho em si foi bastante interessante por aprender sobre diversos aspectos no ramo da microeletrônica, implementar uma lógica alternativa, aprender a utilizar um simulador elétrico diferente e rever conceitos vistos na faculdade. Foi bastante desgastante precisar entender os artigos científicos da área, todos em inglês, utilizando uma linguagem difícil e com poucos detalhes, mas o resultado foi interessante e importante como revisão da literatura.

REFERÊNCIAS

ANUAR, N.; TAKAHASHI, Y.; SEKINE, T. Adiabatic logic versus CMOS for low power applications. **Proc. ITC–CSCC 2009**, [S.l.], p.302–305, 2009.

ASHENDEN, P. **The designer's guide to VHDL**. [S.l.]: Morgan Kaufmann Pub, 2002.

ASU. **Página do projeto PTM da ASU**. Disponível em: <http://ptm.asu.edu/>. Acesso em: maio 2010.

COX, P. M. et al. Acceleration of global warming due to carbon-cycle feedbacks in a coupled climate model. **Nature**, [S.l.], v.408, n.6809, p.184–7, 2000.

DENKER, J. **A review of adiabatic computing**. 1994. 94–97p.

DUNLOP, A.; KERNIGHAN, B. A procedure for placement of standard cell VLSI circuits. **IEEE Transactions on Computer-Aided Design**, [S.l.], v.4, n.1, p.92–98, 1985.

FRANK, M. Common mistakes in adiabatic logic design and how to avoid them. **submitted to Methodologies in Low-Power Design, Las Vegas, NV, June**, [S.l.], 2003.

KIM, S.; PAPAEFTHYMIOU, M. **True single-phase energy-recovering logic for low-power, high-speed VLSI**. 1998. 167–172p.

KIM, S.; PAPAEFTHYMIOU, M. **Single-phase source-coupled adiabatic logic**. 1999. 97–99p.

KRAMER, A. et al. **2nd order adiabatic computation with 2N-2P and 2N-2N2P logic circuits**. 1995. 191–196p.

LJUBLJANA, U. of. **Página do projeto SpiceOpus da Universidade de Ljubljana**. Disponível em: <http://spiceopus.si/>. Acesso em: maio 2010.

LOGIC, A. **Página da empresa Adiabatic Logic do Cambridge Technology Group**. Disponível em: <http://www.adiabaticlogic.com/>. Acesso em: maio 2010.

MAKSIMOVIC, D. et al. Clocked CMOS adiabatic logic with integrated single-phasepower-clock supply. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S.l.], v.8, n.4, p.460–463, 2000.

MOON, Y.; JEONG, D. An efficient charge recovery logic circuit. **IEEE Journal of Solid-State Circuits**, [S.l.], v.31, n.4, p.514–522, 1996.

NAVI, K. et al. Low-Power and High-Performance 1-Bit CMOS Full-Adder Cell. **Journal of computers**, [S.l.], v.3, n.2, p.48, 2008.

OKLOBDZIJA, V.; MAKSIMOVIC, D.; LIN, F. Pass-transistor adiabatic logic using single power-clock supply. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, [S.l.], v.44, n.10, p.842–846, 1997.

PENGJUN, W.; KUNPENG, L.; FENGNA, M. Design of a DTCTGAL circuit and its application. **Journal of Semiconductors**, [S.l.], v.30, p.115006, 2009.

RABAEY, J.; CHANDRAKASAN, A.; NIKOLIĆ, B. **Digital integrated circuits**: a design perspective. [S.l.]: Prentice hall New Jersey, 1996.

SOMASEKHAR, D.; YE, Y.; ROY, K. An energy recovery static RAM memory core. **Memory**, [S.l.], v.5, p.T6, 1995.

SU, L. et al. **An Investigation for Leakage Reduction of Dual Transmission Gate Adiabatic Logic Circuits with Power-Gating Schemes in Scaled CMOS Processes**. 2010. 290–293p.

SYNOPSYS, I. **Liberty User Guide**. [S.l.]: Version, 1999.

THOMAS, D.; MOORBY, P. **The Verilog hardware description language**. [S.l.]: Springer Verlag, 2008.

TUMA, T.; BURMEN, A. **Circuit Simulation with SPICE OPUS**: theory and practice. [S.l.]: Birkhauser, 2009.

UEHARA, T.; VANCLEEMPUT, W. Optimal layout of CMOS functional arrays. **IEEE Transactions on Computers**, [S.l.], v.100, n.30, p.305–312, 1981.

VETULI, A.; PASCOLI, S.; REYNERI, L. Positive feedback in adiabatic logic. **Electronics Letters**, [S.l.], v.32, n.20, p.1867–1869, 1996.

WU, J.; CHANG, K. MOS charge pumps for low-voltage operation. **IEEE Journal of solid-state circuits**, [S.l.], v.33, n.4, p.592–597, 1998.

YOUNIS, S. **Asymptotically zero energy computing using split-level charge recovery logic**. 1994.

ZIESLER, C.; KIM, S.; PAPAEFTHYMIOU, M. **A resonant clock generator for single-phase adiabatic systems**. 2001. 159–164p.

ZIMMERMANN, R.; FICHTNER, W. Low-Power Logic Styles: cmos versus pass-transistor logic. **IEEE JOURNAL OF SOLID-STATE CIRCUITS**, [S.l.], v.32, n.7, p.1, 1997.

APÊNDICE A NETLISTS DOS CIRCUITOS SIMULADOS

A.1 Inversores TSEL

```
INVERSOR TSEL
* Power supply
vref    (vdd 0) dc=3.3
vdd1   (vdd1 0) sin=(1.65 1.65 100meg 10n)
vddp   (vp 0)     dc=1.65
vddn   (vn 0)     dc=1.65
vin   (in 0)   dc=0   pulse=(0.0 3.3 0n 0.8n 0.8n 80n 160n)
vinb  (inb 0)  dc=0   pulse=(3.3 0.0 0n 0.8n 0.8n 80n 160n)
xinversorp (in inb outp outpb vdd1 vp vdd) inverter_pmos
xinversorn (in inb outn outnb vdd1 vn 0) inverter_nmos
coutn  (othern 0) 1f
coutnb (othernb 0) 1f
coutp  (otherp 0) 1f
coutpb (otherpb 0) 1f
xoutn  (outn othern vdd 0) cmos_inverter
xoutnb (outnb othernb vdd 0) cmos_inverter
xoutp  (outp otherp vdd 0) cmos_inverter
xoutpb (outpb otherpb vdd 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=720n l=180n
mn1 (out in g g)   nmos w=720n l=180n
.ends
.subckt inverter_pmos in inb out outb v vp vdd
xbase (out outb node nodeb v vdd) tsel_pbase
mp1 (node in vp vdd)  pmos w=180n l=180n
mp2 (nodeb inb vp vdd) pmos w=180n l=180n
.ends
.subckt inverter_nmos in inb out outb v vn g
xbase (out outb node nodeb v g) tsel_nbase
mn1 (node in vn g)  nmos w=180n l=180n
mn2 (nodeb inb vn g) nmos w=180n l=180n
.ends
.subckt tsel_pbase out outb node nodeb v vdd
mp1 (out outb v vdd)   pmos w=180n l=180n
mp2 (outb out v vdd)   pmos w=180n l=180n
```

```
mp3 (out v node vdd) pmos w=180n l=180n
mp4 (outb v nodeb vdd) pmos w=180n l=180n
.ends
.subckt tsel_nbase out outb node nodeb v g
mn1 (out outb v g)   nmos w=180n l=180n
mn2 (outb out v g)   nmos w=180n l=180n
mn3 (out v node g) nmos w=180n l=180n
mn4 (outb v nodeb g)   nmos w=180n l=180n
.ends
.include tsmc18b.txt
.control
destroy all
tran 0.01n 640n 320n
plot v(vdd1)+16 v(in)+12 v(inb)+8 v(outp)+4 v(outpb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor PMOS'
plot v(vdd1)+16 v(in)+12 v(inb)+8 v(outn)+4 v(outnb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor NMOS'
.endc
.end
```

A.2 Inversor PAL

```
INVERSOR
* Power supply
vref  (vdd 0) dc=1.0
vsrc1  (vdd1 0) sin=(0.5 0.5 25meg 10n)
vsrc2  (vdd2 0) sin=(0.5 0.5 25meg 0.5n)
vin   (in 0) pulse=(0.0 1.0 0n 10n 10n 20n 80n)
vinb  (inb 0) pulse=(1.0 0.0 0n 10n 10n 20n 80n)
cout  (other 0) 0.1f
coutn (otherb 0) 0.1f
xout  (out other vdd 0) cmos_inverter
xoutn (outb otherb vdd 0) cmos_inverter
xinversor (in inb out outb vdd1 vdd 0) inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt inverter in inb outb out v vdd g
xbase (out outb v vdd)  pal_base
mn1 (outb inb v g)  nmos w=45n l=45n
mn2 (out in v g)  nmos w=45n l=45n
.ends
.subckt pal_base out outb v vdd
```

```
mp1 (out outb v vdd) pmos w=45n l=45n
mp2 (outb out v vdd) pmos w=45n l=45n
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 560n 400n
plot v(vdd1)+8 v(in)+6 v(inb)+4 v(out)+2 v(outb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor'
plot -i(vsrc1)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
tran 0.01n 480n 440n
print mean(integrate(-i(vsrc1)))
.endc
.end
```

A.3 Inversor CAL

```
INVERSOR CAL
* Power supply
vref  (vdd 0) dc=1.0
vsrc1  (vdd1 0)  pulse=(0.0 1.0 1n 1n 1n 1n 4n)
vclk  (ck 0)    pulse=(0.0 1.0 0.5n 1p 1p 4n 8n)
vin   (in 0) pulse=(0.0 1.0 0n 1n 1n 1n 8n)
vinb  (inb 0) pulse=(0.0 1.0 4n 1n 1n 1n 8n)
xinversor (in inb out outb vdd vdd1 vdd 0) inverter
cout  (other 0) 0.1f
coutn (otherb 0) 0.1f
xout  (out other vdd 0) cmos_inverter
xoutn (outb otherb vdd 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt inverter in inb out outb ck v vdd g
xbase (out outb node nodeb ck v vdd g) cal_base
mn1 (node in g g)  nmos w=45n l=45n
mn2 (nodeb inb g g) nmos w=45n l=45n
.ends
.subckt cal_base out outb node nodeb ck v vdd g
mp1 (out outb v vdd)   pmos w=45n l=45n
mp2 (outb out v vdd)   pmos w=45n l=45n
mn1 (out outb g g) nmos w=45n l=45n
mn2 (outb out g g)  nmos w=45n l=45n
```

```
mn3 (out ck node g)  nmos w=45n l=45n
mn4 (outb ck nodeb g)  nmos w=45n l=45n
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 32n 16n
plot v(vdd)+10 v(vdd1)+8 v(in)+6 v(inb)+4 v(out)+2 v(outb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor'
plot -i(vsrc1)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
tran 0.01n 24n 20n
print mean(integrate(-i(vsrc1)*v(vdd1)))
.endc
.end
```

A.4 Inversor ECRL

```
INVERSOR ECRL
* Power supply
vref  (vdd 0) dc=1.0
vsrc1  (vdd1 0)   pulse=(0.0 1.0 1n 1n 1n 1n 4n)
vsrc2  (vdd2 0)   pulse=(0.0 1.0 2n 1n 1n 1n 4n)
vsrc1b (vdd1b 0)   pulse=(0.0 1.0 3n 1n 1n 1n 4n)
vsrc2b (vdd2b 0)   pulse=(0.0 1.0 4n 1n 1n 1n 4n)
vin   (in 0) pulse=(0.0 1.0 0n 1n 1n 1n 8n)
vinb  (inb 0) pulse=(0.0 1.0 4n 1n 1n 1n 8n)
xinversor (in inb out outb vdd1 vdd 0) inverter
cout  (other 0) 0.1f
coutn (otherb 0) 0.1f
xout  (out other vdd 0) cmos_inverter
xoutn (outb otherb vdd 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt inverter in inb out outb v vdd g
xbase (out outb v vdd)  ecrl_base
mn1 (out in g g)  nmos w=45n l=45n
mn2 (outb inb g g)  nmos w=45n l=45n
.ends
.subckt ecrl_base out outb v vdd
mp1 (outb out v vdd) pmos w=45n l=45n
mp2 (out outb v vdd) pmos w=45n l=45n
```

```
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 32n 16n
plot v(vdd1)+8 v(in)+6 v(inb)+4 v(out)+2 v(outb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor'
plot -i(vsrc1)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
tran 0.01n 24n 20n
print mean(integrate(-i(vsrc1)*v(vdd1)))
.endc
.end
```

A.5 Inversor 2N-2N2P

```
INVERSOR 2N-2N2P
* Power supply
vref  (vdd 0) dc=1.0
vsrc1  (vdd1 0)   pulse=(0.0 1.0 1n 1n 1n 1n 4n)
vsrc2  (vdd2 0)   pulse=(0.0 1.0 2n 1n 1n 1n 4n)
vsrc1b (vdd1b 0)   pulse=(0.0 1.0 3n 1n 1n 1n 4n)
vsrc2b (vdd2b 0)   pulse=(0.0 1.0 4n 1n 1n 1n 4n)
vin   (in 0) pulse=(0.0 1.0 0n 1n 1n 1n 8n)
vinb  (inb 0) pulse=(0.0 1.0 4n 1n 1n 1n 8n)
xinversor (in inb out outb vdd1 vdd 0) inverter
cout  (other 0) 0.1f
coutn (otherb 0) 0.1f
xout  (out other vdd 0) cmos_inverter
xoutn (outb otherb vdd 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt inverter in inb out outb v vdd g
xbase (out outb v vdd g)  2n2n2p_base
mn1 (out in g g)  nmos w=45n l=45n
mn2 (outb inb g g)  nmos w=45n l=45n
.ends
.subckt 2n2n2p_base out outb v vdd g
mp1 (outb out v vdd)  pmos w=45n l=45n
mp2 (out outb v vdd)  pmos w=45n l=45n
mn4 (outb out g g)   nmos w=45n l=45n
mn3 (out outb g g)    nmos w=45n l=45n
```

```
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 32n 16n
plot v(vdd1)+8 v(in)+6 v(inb)+4 v(out)+2 v(outb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor'
plot -i(vsrc1)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
tran 0.01n 24n 20n
print mean(integrate(-i(vsrc1)*v(vdd1)))
.endc
.end
```

A.6 Inversor PFAL

```
INVERSOR PFAL
* Power supply
vref  (vdd 0) dc=1.0
vsrc1  (vdd1 0)   pulse=(0.0 1.0 1n 1n 1n 1n 4n)
vsrc2  (vdd2 0)   pulse=(0.0 1.0 2n 1n 1n 1n 4n)
vsrc1b (vdd1b 0)   pulse=(0.0 1.0 3n 1n 1n 1n 4n)
vsrc2b (vdd2b 0)   pulse=(0.0 1.0 4n 1n 1n 1n 4n)
vin   (in 0) pulse=(0.0 1.0 0n 1n 1n 1n 8n)
vinb  (inb 0) pulse=(0.0 1.0 4n 1n 1n 1n 8n)
xinversor (in inb out outb vdd1 vdd 0) inverter
cout  (other 0) 0.1f
coutn (otherb 0) 0.1f
xout  (out other vdd 0) cmos_inverter
xoutn (outb otherb vdd 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt inverter in inb out outb v vdd g
xbase (out outb v vdd g)  pfal_base
mn1  (v in outb g) nmos w=45n l=45n
mn2  (v inb out g)  nmos w=45n l=45n
.ends
.subckt pfal_base out outb v vdd g
mp1 (outb out v vdd)  pmos w=45n l=45n
mp2 (out outb v vdd)  pmos w=45n l=45n
mn3 (outb out g g) nmos w=45n l=45n
mn4 (out outb g g)  nmos w=45n l=45n
```

```
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 32n 16n
plot v(vdd1)+8 v(in)+6 v(inb)+4 v(out)+2 v(outb)
+ xlabel 'tempo [s]'
+ ylabel 'tensao [V]'
+ title  'Funcionamento do inversor'
plot -i(vsrc1)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
tran 0.01n 24n 20n
print mean(integrate(-i(vsrc1)*v(vdd1)))
.endc
.end
```

A.7 Somador Completo CMOS

```
FULL-ADDER
* Power supply
vref  (vdd 0) dc=1.0
vref2  (vdd2 0) dc=1.0
va    (a 0)   pulse=(0.0 1.0 0n 0.04n 0.04n 4n 8n)
vab   (ab 0)   pulse=(1.0 0.0 0n 0.04n 0.04n 4n 8n)
vb    (b 0)   pulse=(0.0 1.0 0n 0.04n 0.04n 8n 16n)
vbb   (bb 0)   pulse=(1.0 0.0 0n 0.04n 0.04n 8n 16n)
vcin  (cin 0) pulse=(0.0 1.0 0n 0.04n 0.04n 16n 32n)
vcinb (cinb 0) pulse=(1.0 0.0 0n 0.04n 0.04n 16n 32n)
xsoma (a ab b bb cin cinb sum vdd 0) soma
xcout (ab bb cinb cout vdd 0) carryout
cl1   (other 0)     0.1f
cl2   (otherb 0) 0.1f
xout  (sum other vdd2 0) cmos_inverter
xoutn (cout otherb vdd2 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt carryout a b cin out vdd g
mn1 (out a nm4 g)       nmos w=45n l=45n
mn2 (nm4 b g g)          nmos w=45n l=45n
mn3 (out cin nm5 g)      nmos w=45n l=45n
mn4 (nm5 a g g)       nmos w=45n l=45n
mn5 (nm5 b g g)        nmos w=45n l=45n
mp1 (out a nm1 vdd)   pmos w=45n l=45n
mp2 (nm1 b vdd vdd)   pmos w=45n l=45n
```

```
mp3 (out cin nm2 vdd)   pmos w=45n l=45n
mp4 (nm2 a vdd vdd)   pmos w=45n l=45n
mp5 (nm2 b vdd vdd)      pmos w=45n l=45n
.ends
.subckt soma a ab b bb cin cinb out vdd g
mn1 (out cin nm1 g)  nmos w=45n l=45n
mn2 (out cinb nm2 g)    nmos w=45n l=45n
mn3 (nm1 b  nm3 g)      nmos w=45n l=45n
mn4 (nm1 bb nm4 g)      nmos w=45n l=45n
mn5 (nm2 b nm4 g) nmos w=45n l=45n
mn6 (nm2 bb nm3 g)      nmos w=45n l=45n
mn7 (nm3 ab g g)        nmos w=45n l=45n
mn8 (nm4 a g g)      nmos w=45n l=45n
mp1 (out cin pm1 vdd)  pmos w=45n l=45n
mp2 (out cinb pm2 vdd)  pmos w=45n l=45n
mp3 (pm1 b  pm3 vdd)  pmos w=45n l=45n
mp4 (pm1 bb pm4 vdd)    pmos w=45n l=45n
mp5 (pm2 b pm4 vdd) pmos w=45n l=45n
mp6 (pm2 bb pm3 vdd)  pmos w=45n l=45n
mp7 (pm3 ab vdd vdd)    pmos w=45n l=45n
mp8 (pm4 a vdd vdd)  pmos w=45n l=45n
.ends
.subckt inverter in out vdd g
mp1 (out in vdd vdd) pmos w=45n l=45n
mn1 (out in g g)  nmos w=45n l=45n
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 64n
plot v(a)+8 v(b)+6 v(cin)+4 v(sum)+2 v(cout)
+ xlabel 'tempo [s]'
+ ylabel 'tensão [V]'
+title  ' SUM and Cout'
plot -i(vref)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
plot integrate(-i(vref)*v(vdd))
+ xlabel 'tempo [s]'
+ ylabel 'energia [W]'
+ title  'Carga utilizada pela fonte'
tran 0.01n 128n 64n
print mean(integrate(-i(vref)*v(vdd)))
.endc
.end
```

A.8 Somador Completo Subthreshold

```
FULL-ADDER
* Power supply
vref  (vdd 0) dc=0.4
vref2  (vdd2 0) dc=0.4
va    (a 0)   pulse=(0.0 0.4 0n 0.4n 0.4n 40n 80n)
vab   (ab 0)   pulse=(0.4 0.0 0n 0.4n 0.4n 40n 80n)
vb    (b 0)   pulse=(0.0 0.4 0n 0.4n 0.4n 80n 160n)
vbb   (bb 0)   pulse=(0.4 0.0 0n 0.4n 0.4n 80n 160n)
vcin  (cin 0) pulse=(0.0 0.4 0n 0.4n 0.4n 160n 320n)
vcinb (cinb 0) pulse=(0.4 0.0 0n 0.4n 0.4n 160n 320n)
xsoma (a ab b bb cin cinb sum vdd 0) soma
xcout (ab bb cinb cout vdd 0) carryout
cl1   (other 0)     0.1f
cl2   (otherb 0) 0.1f
xout  (sum other vdd2 0) cmos_inverter
xoutn (cout otherb vdd2 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt carryout a b cin out vdd g
mn1 (out a nm4 g)       nmos w=45n l=45n
mn2 (nm4 b g g)          nmos w=45n l=45n
mn3 (out cin nm5 g)      nmos w=45n l=45n
mn4 (nm5 a g g)        nmos w=45n l=45n
mn5 (nm5 b g g)         nmos w=45n l=45n
mp1 (out a nm1 vdd)   pmos w=45n l=45n
mp2 (nm1 b vdd vdd)   pmos w=45n l=45n
mp3 (out cin nm2 vdd)   pmos w=45n l=45n
mp4 (nm2 a vdd vdd)   pmos w=45n l=45n
mp5 (nm2 b vdd vdd)      pmos w=45n l=45n
.ends
.subckt soma a ab b bb cin cinb out vdd g
mn1 (out cin nm1 g)  nmos w=45n l=45n
mn2 (out cinb nm2 g)    nmos w=45n l=45n
mn3 (nm1 b  nm3 g)      nmos w=45n l=45n
mn4 (nm1 bb nm4 g)      nmos w=45n l=45n
mn5 (nm2 b nm4 g) nmos w=45n l=45n
mn6 (nm2 bb nm3 g)      nmos w=45n l=45n
mn7 (nm3 ab g g)        nmos w=45n l=45n
mn8 (nm4 a g g)      nmos w=45n l=45n
mp1 (out cin pm1 vdd)  pmos w=45n l=45n
mp2 (out cinb pm2 vdd)  pmos w=45n l=45n
mp3 (pm1 b  pm3 vdd)  pmos w=45n l=45n
mp4 (pm1 bb pm4 vdd)    pmos w=45n l=45n
mp5 (pm2 b pm4 vdd) pmos w=45n l=45n
mp6 (pm2 bb pm3 vdd)  pmos w=45n l=45n
```

```
mp7 (pm3 ab vdd vdd)    pmos w=45n l=45n
mp8 (pm4 a vdd vdd)  pmos w=45n l=45n
.ends
.subckt inverter in out vdd g
mp1 (out in vdd vdd) pmos w=45n l=45n
mn1 (out in g g)  nmos w=45n l=45n
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 640n
plot v(a)+4 v(b)+3 v(cin)+2 v(sum)+1 v(cout)
+ xlabel 'tempo [s]'
+ ylabel 'tensão [V]'
+title  ' SUM and Cout'
plot -i(vref)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
plot integrate(-i(vref)*v(vdd))
+ xlabel 'tempo [s]'
+ ylabel 'energia [W]'
+ title  'Carga utilizada pela fonte'
tran 0.01n 1280n 640n
print mean(integrate(-i(vref)*v(vdd)))
.endc
.end
```

A.9 Somador Completo 2N-2N2P

```
FULL-ADDER
* Power supply
vref  (vdd 0) dc=1.0
vsrc1  (vdd1 0)   pulse=(0.0 1.0 1n 1n 1n 1n 4n)
vsrc2  (vdd2 0)   pulse=(0.0 1.0 2n 1n 1n 1n 4n)
vsrc1b (vdd1b 0)   pulse=(0.0 1.0 3n 1n 1n 1n 4n)
vsrc2b (vdd2b 0)   pulse=(0.0 1.0 4n 1n 1n 1n 4n)
va   (a 0)   pulse=(0.0 1.0 0n 1n 1n 1n 8n)
vab  (ab 0)  pulse=(0.0 1.0 4n 1n 1n 1n 8n)
vb   (b 0)   pulse=(0.0 1.0 0n 1n 1n 8n 16n)
vbb  (bb 0)  pulse=(0.0 1.0 8n 1n 1n 8n 16n)
vcin (cin 0) pulse=(0.0 1.0 0n 1n 1n 16n 32n)
vcinb (cinb 0)  pulse=(0.0 1.0 16n 1n 1n 16n 32n)
xsum (a ab b bb cin cinb sum sumb vdd1 vdd 0) soma
xcout (a ab b bb cin cinb cout coutb vdd1 vdd 0) carryout
cl1   (other1 0)     0.1f
cl2   (other2 0) 0.1f
cl3   (other3 0) 0.1f
```

```
cl4   (other4 0) 0.1f
xout1  (sum other1 vdd 0) cmos_inverter
xout2  (sumb other2 vdd 0) cmos_inverter
xout3 (cout other3 vdd 0) cmos_inverter
xout4 (coutb other4 vdd 0) cmos_inverter
.subckt cmos_inverter in out vdd g
mp1 (out in vdd vdd) pmos w=180n l=45n
mn1 (out in g g)  nmos w=180n l=45n
.ends
.subckt carryout a ab b bb cin cinb out outb v vdd g
xbase (out outb v vdd g) 2n2n2p_base
mn1  (out a nm1 g)         nmos w=45n l=45n
mn2  (nm1 b g g)           nmos w=45n l=45n
mn3  (out cin nm2 g)       nmos w=45n l=45n
mn4  (nm2 a g g)         nmos w=45n l=45n
mn5  (nm2 b g g)      nmos w=45n l=45n
mn6  (outb ab nm3 g)     nmos w=45n l=45n
mn7  (nm3 bb g g)          nmos w=45n l=45n
mn8  (outb cinb nm4 g)     nmos w=45n l=45n
mn9  (nm4 ab g g)         nmos w=45n l=45n
mn10 (nm4 bb g g)      nmos w=45n l=45n
.ends
.subckt soma a ab b bb cin cinb out outb v vdd g
xbase (out outb v vdd g) 2n2n2p_base
mn1  (out cin nm1 g)  nmos w=45n l=45n
mn2  (out cinb nm2 g)   nmos w=45n l=45n
mn3  (nm1 b  nm3 g)  nmos w=45n l=45n
mn4  (nm1 bb nm4 g)      nmos w=45n l=45n
mn5  (nm2 b nm4 g) nmos w=45n l=45n
mn6  (nm2 bb nm3 g)  nmos w=45n l=45n
mn7  (nm3 ab g g)        nmos w=45n l=45n
mn8  (nm4 a g g)       nmos w=45n l=45n
mn9  (outb cinb nm5 g)  nmos w=45n l=45n
mn10 (outb cin nm6 g)   nmos w=45n l=45n
mn11 (nm5 bb  nm7 g)  nmos w=45n l=45n
mn12 (nm5 b nm8 g)       nmos w=45n l=45n
mn13 (nm6 bb nm8 g) nmos w=45n l=45n
mn14 (nm6 b nm7 g)       nmos w=45n l=45n
mn15 (nm7 a g g)         nmos w=45n l=45n
mn16 (nm8 ab g g)       nmos w=45n l=45n
.ends
.subckt 2n2n2p_base out outb v vdd g
mp1 (outb out v vdd)  pmos w=45n l=45n
mp2 (out outb v vdd)  pmos w=45n l=45n
mn4 (outb out g g) nmos w=45n l=45n
mn3 (out outb g g)  nmos w=45n l=45n
.ends
.subckt inverter in out vdd g
```

```
mp1 (out in vdd vdd) pmos w=45n l=45n
mn1 (out in g g)  nmos w=45n l=45n
.ends
.include 45nm_HP.pm
.control
destroy all
tran 0.01n 64n
plot v(vdd1)+16 v(a)+14 v(ab)+12 v(b)+10 v(bb)+8 v(cin)+6 v(cinb)+4 v(s
+ xlabel 'tempo [s]'
+ ylabel 'tensão [V]'
+title  ' SUM and Cout'
plot v(vdd1)+16 v(a)+14 v(ab)+12 v(b)+10 v(bb)+8 v(cin)+6 v(cinb)+4 v(c
+ xlabel 'tempo [s]'
+ ylabel 'tensão [V]'
+title  ' SUM and Cout'
plot -i(vsrc1)
+ xlabel 'tempo [s]'
+ ylabel 'corrente [A]'
+ title  'Corrente utilizada pela fonte'
plot integrate(-i(vsrc1)*v(vdd1))
+ xlabel 'tempo [s]'
+ ylabel 'energia [W]'
+ title  'Carga utilizada pela fonte'
tran 0.01n 128n 64n
print mean(integrate(-i(vsrc1)*v(vdd1)))
.endc
.end
```