UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GRACIELI POSSER

Dimensionamento de Portas Lógicas Usando Programação Geométrica

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Ricardo Augusto da Luz Reis
Orientador

Gustavo Reis Wilke
Co-orientador

Porto Alegre, Janeiro de 2011

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Posser, Gracieli

Dimensionamento de Portas Lógicas Usando Programação Geométrica / Gracieli Posser. – Porto Alegre: PPGC da UFRGS, 2011.

105 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2011. Orientador: Ricardo Augusto da Luz Reis; Co-orientador: Gustavo Reis Wilke.

1. Dimensionamento de portas. 2. Síntese física. 3. Programação Geométrica. 4. Modelo de atraso de Elmore. 5. Microeletrônica. I. Reis, Ricardo Augusto da Luz. II. Wilke, Gustavo Reis. III. Título.

“No fim tudo dá certo, e se não deu certo é porque ainda não chegou ao fim”.
Fernando Sabino

AGRADECIMENTOS

Agradeço primeiramente a DEUS, por me acompanhar sempre e em todo lugar. Aos meu PAIS que sempre me apoiaram e me deram a maior riqueza que alguém pode ter: amor e educação. A toda minha FAMÍLIA que sempre torceu por mim, em especial ao meu noivo EDUARDO, que sempre me ajudou em tudo, compreendendo a necessidade de dedicar um tempo a mais para os estudos.

Agradeço de coração ao GUILHERME FLACH que me ajudou muito em todo o mestrado e aos demais COLEGAS (Cristina, Anelise, Adriel, Tania, Jorge, Daniel, Felipe, Glauco, Walter, Gerson, Tiago, Jimmy, Julia, Luiza, Maurício, Jozeanne, Lucas, Davi ...) que sempre foram companheiros e amigos, não medindo esforços para ajudar ou compartilhar as dúvidas e estudos.

Ao meu orientador RICARDO REIS e co-orientador GUSTAVO WILKE por todo o incentivo e apoio em todas as etapas do mestrado. Aos AMIGOS e FAMILIARES que sempre me deram força e compartilharam esta fase comigo.

Agradeço também aos demais professores que fizeram parte da minha formação, pela educação, instrução e apoio.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

AOI	Porta lógica complexa composta pelas portas AND, OR e inversor
CAD	Projeto Auxiliado por Computador (*Computer-Aided Design*)
CCS	Fonte de Corrente Composta (*Composite Current Source*)
CI	Circuito Integrado
CIF	Formato Intermediário Caltech (*Caltech Intermediate Format*)
CMOS	Semicondutor Metal-Óxido Complementar (*Complementary Metal-Oxide Semiconductor*)
ECSM	Modelo de fonte de corrente efetiva (*Effective Current Source Model*)
ELC	Caracterizador de células da Cadence *Encounter Library Characterizer*
FF	Flip-Flop
FO4	*Fanout* de 4 (*Fanout of 4*)
GDS	Arquivo em formato binário com a descrição das formas geométricas de cada célula (*Graphic Data System*)
GND	Terra ou suprimento de energia negativo (*Ground*)
HDL	linguagem de descrição de hardware (*Hardware Description Language*)
INV	Porta lógica que inverte o sinal que recebe em sua entrada
LEF	Formato para Troca de Bibliotecas (Library Exchange Format)
NAND	Célula lógica que representa a função booleana (A . B)
NLDM	Modelo de atraso não linear (*Non-Linear Delay Models*)
NLPM	Modelo de potência não linear (*Non-Linear Power Models*)
NMOS	Canal N de Semicondutor Metal-Óxido
NOR	Célula lógica que representa a função booleana (A + B)
PG	Programação Geométrica
RTL	Nível de Transferência entre Registradores (*Register Transfer Level*)
SPICE	Programa de simulação com ênfase em Circuitos Integrados (*Simulation Program with Integrated Circuit Emphasis*)
SSTA	Análise de *timing* estática estatística (*Statistical Static Timing Analysis*)

STA	Análise de *timing* estática (*Static Timing Analysis*)
UFRGS	Universidade Federal do Rio Grande do Sul
VDD	Suprimento de energia positivo

LISTA DE SÍMBOLOS

a	Atto
F	Farad
f	Femto
μ	Micra/Micrômetro
m	Mili
n	Nano
Ω	Ohms
p	Pico
$\sum$	Soma

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Neste trabalho é desenvolvida uma ferramenta de dimensionamento de portas lógicas para circuitos integrados, utilizando técnicas de otimização de problemas baseadas em Programação Geométrica (PG).

Para dimensionar as portas lógicas de um circuito, primeiramente elas são modeladas usando o modelo de chaves RC e o atraso é calculado usando o modelo de Elmore, que produz funções posinomiais possibilitando a resolução do problema por programação geométrica. Para cada porta é utilizado um fator de escala que multiplica a largura dos seus transistores, onde as variáveis que representam os fatores de escala são as variáveis de otimização do problema.

O dimensionador de portas desenvolvido neste trabalho é para circuitos CMOS e é parametrizável para diversas tecnologias de fabricação CMOS. Além disso, a otimização pode ser feita de duas maneiras, minimizando o atraso restringindo a área do circuito ou, minimizando a área e restringindo o atraso do circuito.

Para testar o dimensionador de portas foram consideradas duas tecnologias de fabricação diferentes, $45nm$ e $350nm$, onde os resultados foram comparados com o dimensionamento fornecido em uma típica biblioteca de células. Para a tecnologia de $45nm$, o dimensionamento de portas minimizando o atraso, fornecido pelo método proposto neste trabalho, obteve uma redução, em média, de 21% no atraso, mantendo a mesma área e potência do dimensionamento fornecido pela biblioteca de *standard cells*. Após, fez-se uma otimização de área, ainda considerando a tecnologia de $45nm$, onde o atraso é restrito ao valor encontrado na minimização de atraso. Essa otimização secundária resultou em uma redução média de 28,2% em área e 27,3% em potência, comparado aos valores dados pela minimização de atraso. Isso mostra que, ao fazer a minimização de atraso seguida da minimização de área, ou vice-versa, encontra-se o menor atraso e a menor área para o circuito, onde uma otimização não impede a outra.

As mesmas otimizações foram feitas para a tecnologia de $350nm$, onde o dimensionamento de portas considerando a minimização de atraso obteve uma redução, em média, de 4,5% no atraso, mantendo os valores de consumo de potência e área semelhantes aos valores dados pelo dimensionamento fornecido em uma biblioteca comercial de células em $350nm$. A minimização de área, feita em seguida, restringindo o atraso ao valor dado pela minimização de atraso foi capaz de reduzir a área em 29,9%, em média, e a potência em 28,5%, em média.

Palavras-chave: Dimensionamento de portas, Síntese física, Programação Geométrica, Modelo de atraso de Elmore, Microeletrônica.

Gate Sizing Using Geometric Programming

ABSTRACT

In this work a gate sizing tool is developed using problem optimization techniques based on Geometric Programming.

To size the gates in a circuit, first, the logic gates are modeled using the RC switch model and the delay is calculated using Elmore delay model, which produces posynomial functions, enabling the problem solution by geometric programming. For each port a scale factor is set that multiplies the transistors width, where the variables that represent the scale factors are the problem optimization variables.

Gate sizing developed in this work is for CMOS circuits and is configurable to several CMOS manufacturing technologies. Moreover, the optimization can be done in two ways, minimizing delay restricting area or by minimizing area restricting circuit delay.

In this work, gate sizing tests were made considers two different technologies, $45nm$ and $350nm$, where the results were compared with the sizing available in a typical standard-cell library. For $45nm$ technology, the gate sizing proposed in this work considering delay minimization, obtained a reduction, in average, of 21% in delay, keeping the same area and power values of the sizing provided by standard-cells library. After, it was made an area optimization restricting delay to the value found at delay minimization. This optimization allowed an average reduction of 28.2% in area and 27.3% in power consumption, compared to the values obtained by delay minimization. This shows that by making the minimization of delay followed by the minimization of area, the smallest delay and the smallest area for the circuit is found, where an optimization does not prevent the other.

The same optimizations were made for $350nm$ technology, where gate sizing considering delay minimization achieved a reduction, on average, of 4.5% in delay, keeping power consumption and area values similar to the values given using the sizes found in a commercial standard-cell library in $350nm$. The area minimization, restricting delay to the value given by delay minimization, was able to reduce the area in 29.9% and power at 28.5%, on average.

Keywords: Gate sizing, Physical synthesis, Geometric Programming, Elmore Delay model, Microelectronics.

1 INTRODUÇÃO

O trabalho dos projetistas de circuitos integrados tem sido cada vez mais difícil, pois a complexidade dos circuitos é cada vez maior e o mercado exige produtos mais eficientes implementados em um curto prazo de tempo. Isto aumenta a necessidade de desenvolver ferramentas CAD (*Computer Aided Design*) (PHYSICAL DESIGN AUTOMATION, 2006), (DESIGN TOOLS AND METHODS FOR CHIP PHYSICAL DESIGN, 2011) para ampliar a capacidade do projetista, permitindo uma produtividade maior em um tempo menor.

Para alcançar um eficiente balanceamento entre área e performance nos circuitos integrados, tem sido usado pela indústria e academia, desde um longo tempo, o fluxo de síntese baseado em *standard cells*. Entretanto, recentemente, o fluxo de síntese livre de biblioteca vem ganhando maior atenção por possibilitar o projeto de circuitos com um número menor de transistores, conforme é detalhado a seguir.

1.1 Fluxo de Síntese Tradicional

O fluxo de síntese tradicional, ou seja, baseado em *standard cells*, normalmente segue a metodologia mostrada na Figura 1.1, onde a síntese inicia da descrição do circuito em linguagem de hardware (HDL). A descrição é mapeada para um conjunto de funções lógicas disponíves em uma biblioteca de células. Essas células são comumente chamadas de *standard cells* e, normalmente, são projetadas de forma manual. Após o mapeamento do circuito para um conjunto de portas, são realizadas as etapas de síntese física:

- *floorplanning* ou planta baixa do circuito. Esta etapa inicia antes do mapeamento do circuito, onde é feito o planejamento de cada bloco que compõe o circuito e é finalizada após o mesmo, considerando as células escolhidas para compor o circuito;
- posicionamento das células;
- síntese da árvore de *clock* (CTS - *Clock Tree Synthesis*), no caso de circuitos sequenciais, e
- roteamento, obtendo, ao final, o leiaute (descrição física) do circuito.

Esse fluxo de síntese é conhecido por ser muito confiável e previsível, desde que a mesma biblioteca de células possa ser utilizada em vários projetos diferentes.

Entretanto, a qualidade dos circuitos projetados é proporcional ao número de células disponíveis na biblioteca de células, quanto maior o número de células disponíveis, maior será o espaço de soluções a ser explorado e, por consequência, maior a possibilidade de uma boa escolha para a implementação do circuito (GUAN; SECHEN, 1996).

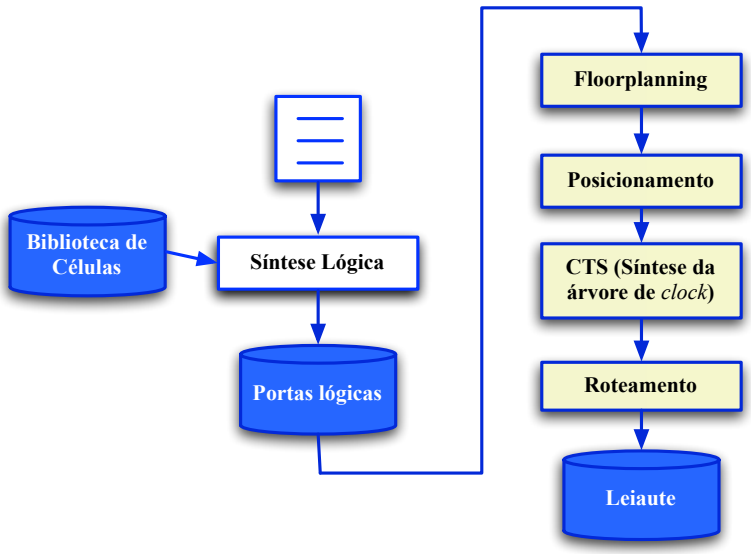


Figura 1.1: Fluxo de síntese tradicional de circuitos.

Isto leva a um projeto melhor ajustado e otimizado com um número menor de transistores (GAVRILOV et al., 1997), (RIERA et al., 1997), (BUTZEN et al., 2007), (VALIDAÇÃO DE BIBLIOTECAS DE CÉLULAS PARA PROJETOS DE CIRCUITOS INTEGRADOS DIGITAIS, 2009). Por outro lado, cada vez que uma nova tecnologia é lançada, as bibliotecas de células devem ser migradas ou reprojetadas para a nova tecnologia e quanto maior a biblioteca maior será o tempo necessário para reprojetá-la ou migrá-la.

A Tabela 1.1 (DETJENS et al., 1987) mostra o número de funções possíveis de realizar, considerando portas CMOS, usando um número máximo de transistores PMOS e de transistores NMOS empilhados. Pode-se observar, por exemplo, que se é escolhido um número máximo de 4 transistores PMOS e 4 transistores NMOS em série é possível fazer 3503 funções lógicas diferentes. Este número de funções lógicas é muito maior que o número de funções encontradas em típicas bibliotecas de *standard cells*, que é em torno de 50 a 150 funções diferentes (NANGATE, 2009), (SYSTEMS, 2009). Implementar bibliotecas de células com todas as funções lógicas possíveis torna-se inviável, pois, normalmente, as células são projetadas de forma manual, o que torna o tempo de projeto e reprojeto a cada nova tecnologia muito alto. Uma solução é usar uma abordagem onde as células são projetadas sob demanda, durante o projeto físico.

Tabela 1.1: Número de funções possíveis usando um número máximo de transistores PMOS e NMOS em série.

		Número de transistores PMOS em série				
		1	2	3	4	5
Número de transistores NMOS em série	1	1	2	3	4	5
	2	2	7	18	42	90
	3	3	18	87	396	1677
	4	4	42	396	3503	28435
	5	5	90	1677	28435	125803

1.2 Fluxo de Síntese Física Livre de Biblioteca

Tentando alcançar um melhor balanceamento entre qualidade e custo do projeto que o fluxo de síntese baseado em *standard cells*, o fluxo de síntese de leiaute livre de biblioteca começou a ganhar mais atenção recentemente, comercialmente (NANGATE, 2010a) e (PROLIFIC, 2009) e academicamente (ANCEAU; REIS, 1982), (REIS et al., 1997), (TOGNI et al., 2002), (CORREIA; REIS, 2002), (RIEPE; SAKALLAH, 2003), (IIZUKA; IKEDA; ASADA, 2005), (ROSA JR. et al., 2007), (MARQUES et al., 2007), (REIS, 2008), (LAZZARI; ZIESEMER; REIS, 2009), principalmente devido à necessidade de redução de potência. Esta metodologia baseia-se no fato de que, se a ferramenta de síntese lógica (ELIS, 2010), (BERKELEY, 2010) tem a liberdade de usar qualquer função lógica e qualquer tamanho possível (respeitando as restrições elétricas e de tecnologia), o mapeamento tecnológico resultante é capaz de diminuir o número de transistores significativamente, reduzindo o atraso, o consumo de potência e a área do circuito (REIS, 1987).

Para que o fluxo de projeto livre de biblioteca possa ser utilizado de forma adequada, as funções lógicas definidas pela ferramenta de síntese lógica devem ter seu leiaute físico construído, que pode ser feito manualmente, onde é gasto um grande tempo, ou pode ser gerado automaticamente utilizando uma ferramenta de geração automática de leiautes.

Uma das etapas que constitui o fluxo de projeto livre de biblioteca é o dimensionamento de portas, onde é permitido tamanhos contínuos de células, desde que as mesmas sejam geradas sob demanda. Isso é uma vantagem em relação às *standard-cells* que são limitadas a alguns poucos tamanhos de transistores.

A geração automática de células causa uma dúvida em relação à qualidade do leiaute gerado automaticamente, pois, normalmente, espera-se que ele possua qualidade inferior às *standard-cells* que são confeccionadas manualmente por projetistas experientes, visando obter as melhores características em área, potência e atraso. Já o leiaute gerado automaticamente não possui toda essa interação com o projetista, pois ele é confeccionado diretamente por uma ferramenta. Essa questão é melhor discutida no Capítulo 4.

1.3 Ferramentas de Síntese Física Segundo a Metodologia TRANCA/UFRGS

O desenvolvimento de ferramentas para a síntese física de circuitos integrados vem sendo realizado há algum tempo em nosso grupo de pesquisa. Um resumo dos trabalhos feitos seguindo a metodologia TRANCA (*TRANsparent-Cell Approach*) (REIS, 1987) foi apresentada por (LAZZARI, 2003) e (ZIESEMER, 2007). Uma linha do tempo com cada ferramenta correspondente, atualizada por (ZIESEMER, 2007), pode ser vista na Figura 1.2.

O gerador TRAMO (*TRANca Module*) (REIS; GOMES; LUBASZEWSKI, 1988), (LUBASZEWSKI, 1990) foi o primeiro sistema desenvolvido baseado na metodologia TRANCA e fazia a geração de leiautes baseados em uma biblioteca de células. Já na ferramenta TRAGO (*TRanca Automatic GeneratOr*) (MORAES, 1990) a síntese era executada de tal forma que os leiautes eram gerados automaticamente sem uma biblioteca de células.

No projeto MARCELA (GÜNTZEL; RIBAS; REIS, 1991), (GÜNTZEL, 1993), (GÜNTZEL et al., 1995), a geração do leiaute constrói uma matriz de uso genérico composta por células básicas dispostas dentro de unidades básicas repetidas ao longo da ma-

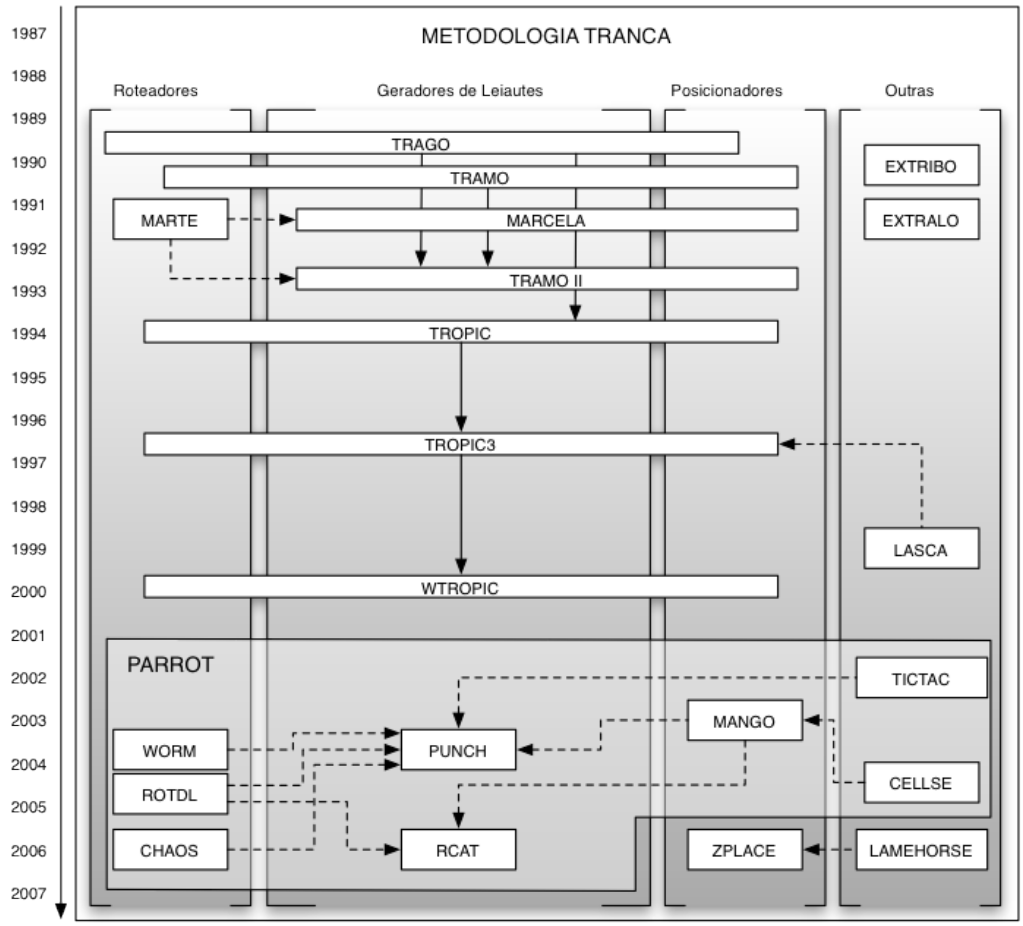


Figura 1.2: Linha do tempo das ferramentas de síntese física segundo a metodologia TRANCA (ZIESEMER, 2007).

triz. Cada unidade básica é delimitada por duas linhas de GND, sendo cruzada por uma linha de VCC ao centro. As células básicas foram projetadas de modo a acomodar trilhas de roteamento tanto na vertical como na horizontal visando o roteamento compartilhado (*over-the-cells routing*).

MARLA (GÜNTZEL et al., 1993) e MARTE (JOHANN; REIS, 1993) são ferramentas para a geração de leiautes baseados na abordagem MARCELA, onde MARLA gera o leiaute sobre uma matriz e MARTE faz o roteamento das células (LAZZARI, 2003).

A ferramenta TRAMOII (REIS; REIS, 1991), (REIS, 1993) fazia a geração de circuitos onde as células eram transparentes às conexões de metal 1 e metal 2, diferente de TRAMO, onde somente uma camada de metal era usada.

TROPIC (*Transparent Reconfigurable Optimized Parametrizable Integrated Circuit generator*) foi apresentada por (MORAES, 1994) e fazia a geração do leiaute baseada em um *netlist* de transistores e nas regras de projeto. Os leiautes eram limitados às tecnologias com duas camadas de metais. Na ferramenta TROPIC3 (MORAES; REIS; LIMA, 1997), 3 camadas de metal são usadas no roteamento.

Uma ferramenta para extração elétrica dos circuitos gerados pelo TROPIC, chamada LASCA, foi apresentada em (FERREIRA; MORAES; REIS, 2000).

Em (FRAGOSO, 2001) é apresentado o WTROPIC, que é uma ferramenta de geração de leiautes para ser usada via Internet.

A ferramenta desenvolvida em (WILKE et al., 2002) e os demais trabalhos relacio-

nados à verificação temporal e identificação do atraso crítico de circuitos combinacionais receberam o nome de TICTAC.

(HENTSCHKE, 2002) apresenta uma ferramenta de posicionamento chamada MANGO PARROT, sendo que a mesma produz um posicionamento relativo das células ao longo das bandas do circuito com o objetivo de reduzir o comprimento total das interconexões.

A ferramenta PARROT PUNCH foi desenvolvida em (LAZZARI, 2003), onde sua principal característica é a geração de leiautes sob demanda, sem a necessidade de uma biblioteca de células, onde o leiaute é gerado como se o circuito inteiro fosse uma célula gigante, sem hierarquia (ZIESEMER, 2007).

Para realizar as interconexões do leiaute do circuito gerado pela ferramenta PUNCH, foi desenvolvido um roteador em malha chamado WORM. Sendo que, o fluxo de geração de leiaute criado com o uso destas ferramentas recebeu o nome de PARROT e diversas ferramentas foram desenvolvidas posteriormente e integradas à este fluxo (ZIESEMER, 2007).

Em (FLACH; HENTSCHKE; REIS, 2004) foi desenvolvido um roteador mais robusto que o WORM, chamado ROTDL. E, em (ZIESEMER, 2006), foi apresentado um gerador de estimativas de tamanhos de células chamado CELLSE, possibilitando estimar as dimensões das células de forma mais automatizada e precisa que o estimador usado pela ferramenta MANGO.

ChAOS (SANTOS; JOHANN; REIS, 2006) é um roteador que surgiu com o objetivo de ser convergente e ágil, pois é implementado com algoritmos de baixa complexidade. A convergência é garantida pela inserção de espaços em branco e seus tempos de execução são muito inferiores ao WORM e ROTDL, podendo haver um pequeno aumento na área final devido aos espaços inseridos (ZIESEMER, 2007).

(MEINHARDT, 2006) desenvolveu a ferramenta RCAT, que é um gerador de leiautes regulares baseado em matrizes de células e foi integrada ao fluxo PARROT.

As ferramentas ZPLACE (HENTSCHKE, 2007) e LAMEHORSE (SAWICKI et al., 2007) foram desenvolvidas para a utilização em circuitos 3D. ZPLACE é um posicionador quadrático para a redução do comprimento de fio com observância do caminho crítico. Ele realiza também o posicionamento de circuitos 2D e suporta circuitos muito maiores que o MANGO PARROT. Já a LAMEHORSE é uma ferramenta de particionamento e posicionamento dos PADs e pinos de entrada e saída em circuitos 3D.

Além dessas ferramentas, em (ZIESEMER, 2007) é apresentada a ferramenta de geração automática de leiautes de células chamada ASTRAN, que é detalhada a seguir.

1.3.1 Ferramenta de Geração Automática de Leiautes ASTRAN

A ferramenta acadêmica *netlist-to-layout* ASTRAN foi desenvolvida por (ZIESEMER, 2007) com o objetivo de automatizar o processo de desenvolvimento do leiaute de células. Ela está disponível para os Sistemas Operacionais Windows, Linux e Mac OS (ASTRAN, 2010).

A ferramenta de geração automática de leiautes gera as células sobre uma matriz linear 1D e é capaz de suportar células com diferentes redes de transistores, obedecendo o dimensionamento dos transistores especificado pelo projetista. Além disso, para manter uma altura uniforme para todas as células, o gerador de células permite fazer *folding* dos transistores, que consiste em quebrar os transistores de maior tamanho em transistores menores, conectados em paralelo.

O ASTRAN usa uma versão do algoritmo Threshold Accepting (HENTSCHKE, 2007)

para o posicionamento dos transistores, sendo que o mesmo objetiva a redução da largura das células, ou seja, maximiza o compartilhamento da difusão e minimiza o comprimento das interconexões. O roteamento interno das células é feito usando um algoritmo baseado em negociação de congestionamento similar ao PathFinder (MCMURCHIE; EBELING, 1995).

O resolvedor de Programação Linear LPSolver (LPSOLVER, 2009) é utilizado para a compactação do leiaute, produzindo o leiaute final conforme os resultados fornecidos pelas etapas de posicionamento e roteamento e as regras de projeto da tecnologia utilizada.

O fluxo utilizado pela ferramenta ASTRAN para fazer a geração das células é mostrado na Figura 1.3, onde a primeira etapa é a leitura dos arquivos de entrada, depois é feito o *folding* dos transistores, caso seja necessário. Após, vem as etapas de posicionamento dos transistores e roteamento interno das células e, por fim, a compactação do leiaute.

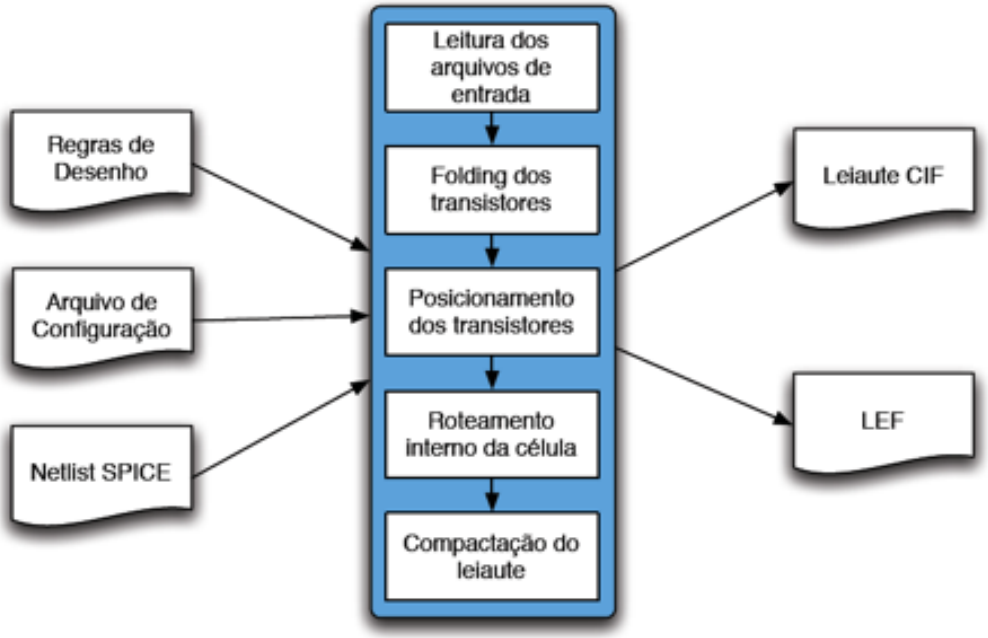


Figura 1.3: Fluxo para a geração de células utilizado pela ferramenta ASTRAN (ZIESEMER, 2007).

Para a geração do leiaute, a ferramenta recebe como entrada três arquivos:

- Um arquivo no formato SPICE com o *netlist* das células;
- Um arquivo de configuração (que define a topologia do leiaute e os parâmetros de controle para o gerador), e
- Um arquivo de tecnologia (que contém a descrição das regras de projeto).

O arquivo no formato SPICE possui para cada célula (porta lógica) a lista de seus transistores, informando o comprimento e a largura dos mesmos, as redes às quais pertencem e seus pinos de entrada e saída. O gerador recebe este arquivo como entrada e gera o leiaute de cada uma das células de acordo com as regras da tecnologia especificada.

O leiaute é construído pela ferramenta seguindo algumas características básicas:

- Formado por 2 linhas horizontais de difusão paralelas, uma P e outra N, com os *gates* dos transistores posicionados na vertical;
- Poço posicionado e dimensionado de forma a permitir o espelhamento de células;
- Roteamento interno da célula feito exclusivamente utilizando polisilício, metal 1 e difusão;
- Pinos de entrada/saída da célula posicionados sobre as trilhas centrais e alinhados à grade de roteamento.

As células geradas automaticamente pela ferramenta seguem um estilo de leiaute regular que permite seu uso em um fluxo de síntese com posicionamento e roteamento automático das células, compatível ao fluxo *standard cells*. Um exemplo de célula gerada automaticamente pelo ASTRAN na tecnologia de $350nm$ pode ser vista na Figura 1.4 (a). A mesma célula, XOR20, da biblioteca de células comercial da AMS para a tecnologia de $350nm$ é mostrada na Figura 1.4 (b).

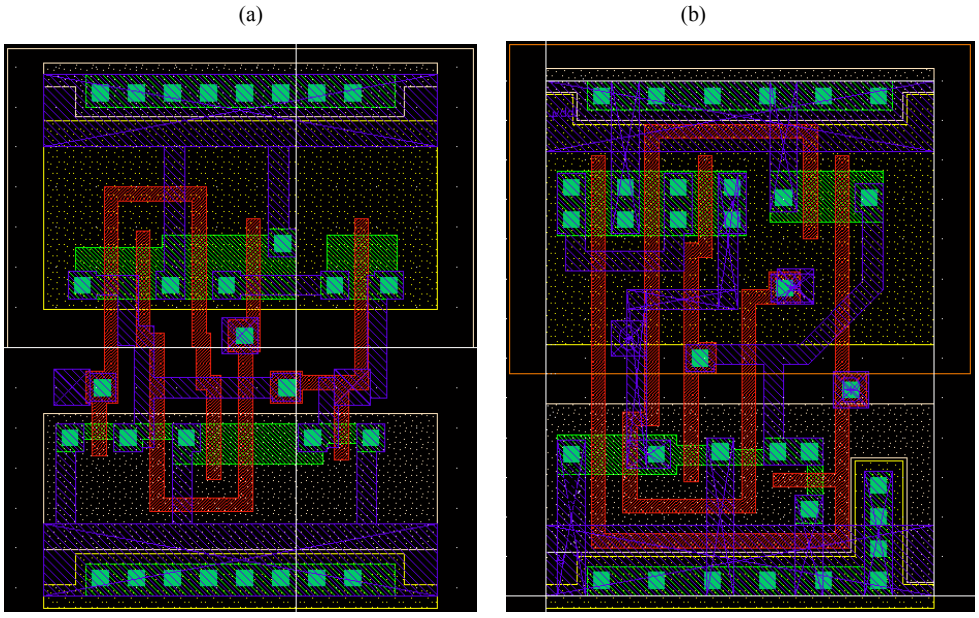


Figura 1.4: Leiaute da célula XOR20 gerada automaticamente pela ferramenta ASTRAN (a) e proveniente de uma biblioteca de células (b).

1.4 Dimensionamento

O dimensionamento de portas consiste em determinar o melhor tamanho dos transistores de cada porta lógica do circuito com o objetivo de obter o menor atraso limitado a uma certa área. Na literatura são propostas inúmeras maneiras de dimensionar as portas de um circuito, algumas priorizando o atraso, outras priorizando potência e área e, outras ainda que aplicam técnicas sem considerar essas variáveis diretamente, como é o caso do *fanout-of-4 inverter* (HARRIS et al., 1997). Neste trabalho, é desenvolvido um método de

dimensionamento de portas baseado em programação geométrica utilizando duas abordagens, uma onde o objetivo é minimizar o atraso do circuito, restringindo a área e a outra, onde o objetivo é minimizar a área sob uma restrição de atraso.

O dimensionador de portas lógicas desenvolvido neste trabalho faz parte do fluxo de síntese que pode ser livre de bibliotecas, onde a ferramenta ASTRAN é utilizada para fazer a geração automática dos leiautes, conforme mostra a Figura 1.5, sendo que a etapa de dimensionamento é representada pelo componente em vermelho. O fluxo livre de biblioteca é composto das seguintes etapas:

1. A ferramenta de síntese lógica recebe como entrada a descrição comportamental do circuito do qual deseja-se obter o leiaute e o mapeia para portas lógicas. No caso de mapeamentos baseados na metodologia *standard cells*, o circuito é mapeado para instâncias de células de uma biblioteca escolhida pelo projetista, já o mapeamento livre de biblioteca é executado buscando a lógica mais otimizada sem estar restrito a um conjunto de funções. Um exemplo de ferramenta que pode ser utilizada nesta etapa é a ELIS (*Environment for Logic Synthesis*) (ELIS, 2010), que foi desenvolvida na UFRGS, assim como a MVSIS (BERKELEY, 2010).

2. No mapeamento fornecido pela síntese lógica não é definido, ainda, o tamanho dos transistores que compõem o circuito, sendo necessário fazer o dimensionamento das portas lógicas antes da geração do leiaute, utilizando o dimensionador implementado neste trabalho.

3. Após obter o circuito com as portas lógicas dimensionadas, a ferramenta ASTRAN está apta a gerar o leiaute do mesmo e, consequentemente, realizar as etapas de posicionamento e roteamento das células, analisando se os requisitos de *timing* são atendidos, caso contrário, a(s) etapa(s) anterior(es) deve(m) ser refeita(s).

4. Ao final, é obtido o leiaute do circuito.

1.5 Contribuições

O dimensionador de portas, neste trabalho, é desenvolvido para circuitos CMOS, utilizando o Elmore (ELMORE, 1948) como modelo de atraso que produz funções posinomiais, possibilitando a resolução do problema por Programação Geométrica (PG) de forma ótima através de um resolvedor de programação geométrica, neste caso foi usado o GGPLAB (GGPLAB: A MATLAB TOOLBOX FOR GP, 2010) em conjunto com o Matlab. Considerando o desenvolvimento do dimensionador de portas, as contribuições deste trabalho podem ser resumidas como segue:

- Um dimensionador de portas que utiliza Programação geométrica para resolver o problema de dimensionamento, convergindo para um mínimo e pode ser utilizado juntamente com uma ferramenta de geração automática de leiautes, habilitando a concepção de circuitos integrados pelo fluxo de projeto livre de biblioteca. Além da vantagem de poder usar tamanhos contínuos para as portas e transistores, evitando problemas de arredondamento que são comuns nas abordagens tradicionais de dimensionamento de portas.

- Um dimensionador de portas parametrizável para diversas tecnologias de fabricação CMOS, bastando apenas mudar os parâmetros relativos à tecnologia.

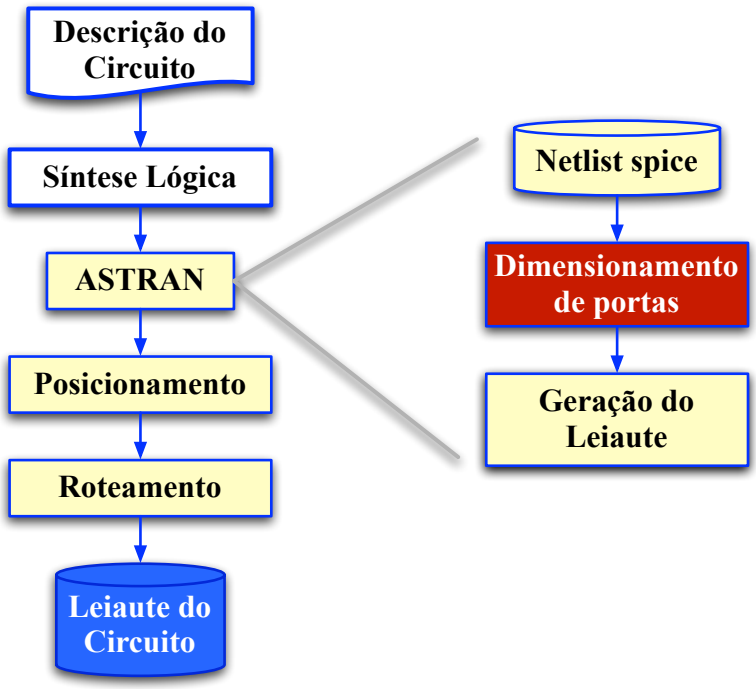


Figura 1.5: Fluxo de síntese onde o trabalho se encaixa.

- Um método de dimensionamento de portas que habilita a utilização de duas abordagens, uma onde o objetivo é minimizar o atraso do circuito, restringindo a área e a outra, onde o objetivo é minimizar a área sob uma restrição de atraso, podendo, desta forma, encontrar o atraso mínimo e a área mínima para o circuito.

- A junção dos trabalhos de dimensionamento (BOYD et al., 2005) e (BOYD et al., 2007), sendo que, em nosso trabalho, os transistores de cada porta lógica do circuito são modelados usando o modelo de chaves RC apresentado por (BOYD et al., 2005), buscando uma modelagem mais precisa do atraso da porta. Enquanto que o dimensionamento das portas é modelado conforme (BOYD et al., 2007), usando um fator de escala para cada porta lógica do circuito. Considerando os trabalhos da literatura, não encontrou-se, até o presente momento, outro trabalho que tenha feito essa junção. Além disso, nosso trabalho proporciona a minimização da área do circuito, conforme foi proposto por (SAPATNEKAR et al., 1993).

- Uma análise da qualidade das células geradas pela ferramenta ASTRAN comparada a *standard-cells* de uma biblioteca comercial de células.

1.6 Organização da Dissertação

O restante do trabalho está organizado como segue. No Capítulo 2, são apresentados conceitos e definições sobre o dimensionamento de portas e outros termos utilizados no desenvolvimento da dissertação, como a caracterização de células e a Programação Geométrica (PG), utilizada para resolver o problema de dimensionamento. O Capítulo 3 apresenta os trabalhos mais relevantes encontrados na literatura relacionados ao dimensionamento de transistores e de portas lógicas, mostrando em mais detalhes os métodos nos quais este trabalho está baseado.

O Capítulo 4 apresenta um estudo da qualidade dos leiautes gerados pela ferramenta ASTRAN comparados às *standard-cells* de uma biblioteca comercial.

No capítulo 5, é apresentado o desenvolvimento e a implementação do dimensionador de portas, a verificação do seu funcionamento e os resultados comparados ao dimensionamento utilizado em uma biblioteca de células. Além disso, para a tecnologia de $45nm$, o teste do dimensionamento de portas é feito primeiramente com o objetivo de minimizar o atraso sob uma restrição de área e após é minimizada a área sob a restrição de atraso encontrada pela otimização do atraso.

Finalmente, o Capítulo 6 apresenta as conclusões e os trabalhos futuros.

2 DIMENSIONAMENTO

Neste capítulo, são apresentados os conceitos sobre dimensionamento de portas, definições temporais em circuitos e os termos programação geométrica e atraso de Elmore, que são utilizados para calcular e resolver o problema de dimensionamento. Os conceitos sobre caracterização de células também são apresentados, sendo que ela é utilizada na análise dos resultados, buscando através de simulação SPICE obter resultados mais precisos de atraso e potência do circuito.

2.1 Conceitos e Definições

Um circuito digital é composto por um conjunto de portas lógicas interconectadas por fios. A saída de cada porta é conectada por um fio às entradas de uma ou mais portas ou a algum circuito externo. A entrada de cada porta é conectada à saída de outra porta ou a algum circuito externo. As portas lógicas e, consequentemente, os transistores que compõem o circuito devem estar dimensionados de forma que o atraso máximo seja o menor possível dentro de uma dada especificação de área.

O dimensionamento de portas consiste em determinar o melhor tamanho para cada porta lógica do circuito com o objetivo de reduzir a área e o consumo de potência sem violar os requisitos de atraso, considerando que quanto maior o tamanho da porta, menor será a sua resistência e, consequentemente, menor será o seu atraso. O objetivo do dimensionamento é minimizar o atraso, a área ou o consumo de potência.

No dimensionamento de portas, as larguras dos transistores que compõem uma porta são previamente definidas e escaladas de acordo com o fator de escala da porta lógica a qual ele pertence. Existe um fator de escala ótimo para cada porta lógica, considerando que ao aumentar o tamanho da porta e, consequentemente, dos transistores, aumenta-se a sua capacidade de alimentar uma carga, reduzindo o tempo necessário para a porta transicionar o seu sinal. No entanto, ao aumentar o tamanho da porta:

- se reduz a sua resistência de saída, e
- aumenta-se sua capacitância de entrada, dando ao seu "condutor" maior carga capacitiva.

Por isso, deve-se encontrar o tamanho ótimo, onde a porta é capaz de alimentar a carga ligada a ela sem produzir uma alta carga para a porta que a está alimentando.

A Figura 2.1 demostra como é feito o dimensionamento de um transistor com largura base W pelo fator de escala X.

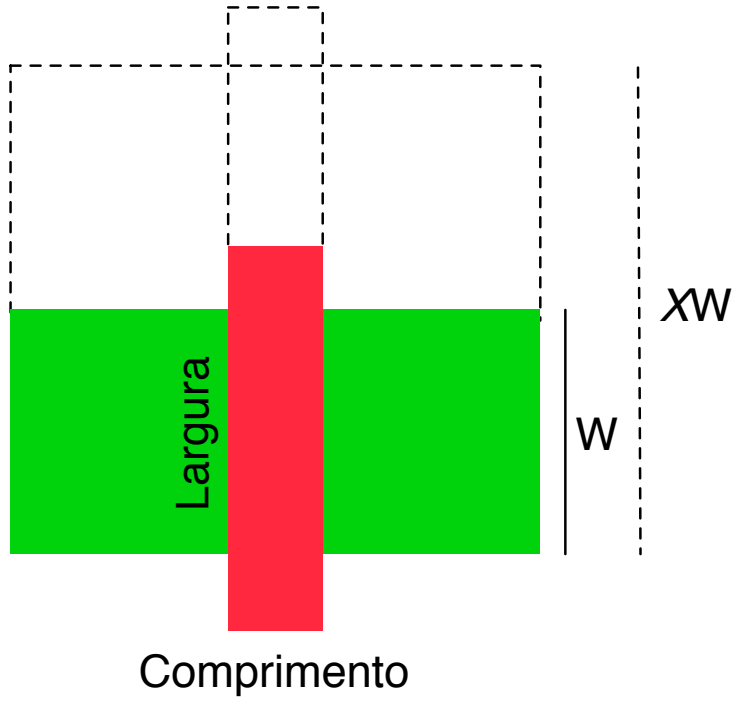


Figura 2.1: Dimensionamento de um transistor com largura base W pelo fator de escala X, sendo que o comprimento do transistor é constante conforme a tecnologia em questão.

2.1.1 Porta, Célula e Instância

Neste trabalho, os termos "porta" e "célula" são usados de forma intercambiável para falar das portas lógicas de um circuito, sendo que uma célula é um modelo (ou gabarito) que descreve a topologia e os tamanhos base dos transistores de uma porta lógica.

Já uma "instância" descreve o uso de uma célula dentro do circuito, onde os transistores têm sua largura base aumentada/diminuída por um fator de escala. Dessa forma, apesar de se usar o termo "dimensionamento de portas", por conveniência, são as instâncias do circuito que têm seus tamanhos alterados de acordo com seu fator de escala.

2.1.2 Fator de escala

Cada porta lógica k do circuito, possui um fator de escala X_i associado para dimensionar os transistores que compõem a porta. As variáveis que representam o fator de escala de cada instância são as variáveis de otimização do problema de dimensionamento.

A Figura 2.2 exemplifica o funcionamento do fator de escala, onde a instância Y de um inversor possui fator de escala 1, ou seja, seus transistores estão com seu tamanho base. Já a instância Z, possui fator de escala 3, ou seja, a largura dos seus transistores é três vezes maior que a largura dos transistores do inversor Y.

2.1.3 Definições Temporais

Dentre os vários dados temporais utilizados no contexto de circuitos integrados, seguem as definições dos mais utilizados neste trabalho (VALIDAÇÃO DE BIBLIOTECAS DE CÉLULAS PARA PROJETOS DE CIRCUITOS INTEGRADOS DIGITAIS, 2009):

- Tempo de Subida (t_r): tempo necessário para que a saída da célula, na transição de

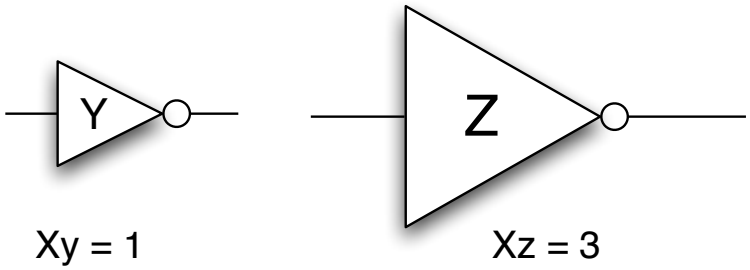


Figura 2.2: Exemplo de um inversor não escalado, ou seja, com fator de escala igual a 1 (esquerda) e um inversor com fator de escala 3 (direita).

subida, varie seu sinal de 10% até 90% do seu valor estável. Os valores percentuais dessas medidas podem variar para cada fabricante, sendo que os valores usuais compreendem as faixas de 10%-90%, 20%-80% e 30%-70%.

- Tempo de Descida (t_f): tempo necessário para que a saída da célula, na transição de descida, varie seu sinal de 90% até 10% do seu valor estável. Os valores percentuais dessas medidas também podem variar para cada fabricante, sendo que os valores usuais compreendem as faixas de 90%-10%, 80%-20% e 70%-30%.

- Tempo de proparação do atraso (t_{pd}): define quão rapidamente a saída de uma porta responde a uma variação do sinal de sua entrada. É obtido pela diferença temporal dos pontos de transição de 50% do sinal de saída em relação a 50% do sinal de entrada, sendo avaliados em dois momentos distintos:

  Tempo de propagação do atraso alto-baixo (*high-low*) (tdhl): define a resposta da célula na transição *high-low* (ou positiva) do sinal de saída.

  Tempo de propagação do atraso baixo-alto (*low-high*) (tdlh): define a resposta da célula na transição *low-high* (ou negativa) do sinal de saída.

- Tempo de transição ou inclinação da onda de entrada (*slew*): é a taxa de transição de um sinal, tipicamente em $volts/ns$. O tempo de transição é o tempo que leva para o sinal passar de duas tensões de limiar específicas. Os pontos de limiar são usualmente definidos como uma certa porcentagem da oscilação da tensão.

As definições temporais podem ser melhor visualizados a partir do gráfico contido na Figura 2.3 (RABAEY; CHANDRAKASAN; NIKOLIC, 2002).

2.2 Programação Geométrica

O termo Programação Geométrica (GP) foi introduzido em 1967 por Duffin, Petterson, e Zener (DUFFIN; PETERSON; ZENER, 1967) e é utilizado para definir um tipo de problema de otimização matemática onde a função objetivo é uma função posinomial. Há uma diferença no significado dos termos programação geométrica e otimização geométrica (BOYD et al., 2007):

- programação geométrica se refere à otimização de problemas, e

- otimização geométrica refere-se a problemas de otimização envolvendo geometria.

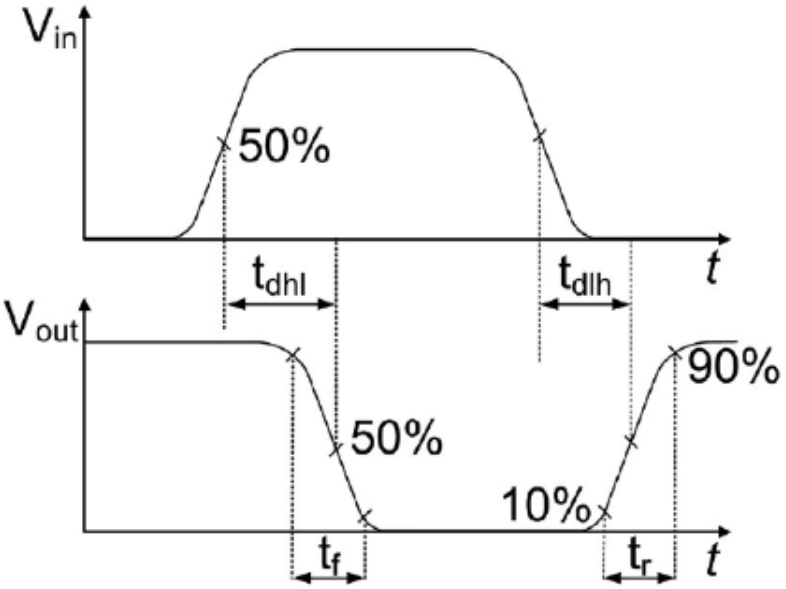


Figura 2.3: Definição de tempo de subida e tempo de descida, tempo de propagação do atraso *low-high* e *high-low* (RABAEY; CHANDRAKASAN; NIKOLIC, 2002).

Com o desenvolvimento de métodos que podem resolver problemas de programação geométrica de forma eficiente, tem-se encontrado um número de problemas práticos, particularmente em projeto de circuitos integrados que são equivalentes ou bem aproximados à Programação Geométrica (BOYD et al., 2007), como é o caso do dimensionamento de portas.

Para entender o que é uma função posinomial, primeiro é definido monômio, conforme mostra a expressão 2.1, onde o coeficiente c pode ser qualquer número positivo, e os expoentes podem ser quaisquer números reais.

$$f(x)^{\gamma} = (cx_1^{a_1}x_2^{a_2}...x_n^{a_n})^{\gamma} = c^{\gamma}x_1^{\gamma a_1}x_2^{\gamma a_2}...x_n^{\gamma a_n} \tag{2.1}$$

A soma de um ou mais monômios, isto é, a função na forma:

$$f(x) = \sum_{k=1}^{K} c_k x_1^{a_{1k}} x_2^{a_{2k}} ... x_n^{a_{nk}} \tag{2.2}$$

onde $c_k > 0$, é chamada de função posinomial ou um posinômio (com *K* termos, e variáveis de x_1, ..., x_n).

Um monômio é também um posinômio. Dividindo-se um posinômio por um monômio, o resultado será outro posinômio. E, multiplicando vários posinômios ou o elevando a um expoente positivo, também será obtido um posinômio.

Um programa geométrico é um problema de otimização na forma:

$$\begin{aligned} &\text{minimizar} && f_0(x) \\ &\text{sujeito a} && f_i(x) \leq 1, \quad i = 1, ..., m, \\ &&& g_i(x) = 1, \quad i = 1, ..., p \end{aligned} \tag{2.3}$$

onde f_i são funções posinomiais, g_i são funções monomiais e x_i são as variáveis a serem otimizadas, que devem ser sempre positivas.

Um problema de programação geométrica na forma padrão tem sua função objetivo na forma posinomial, as restrições de igualdade podem somente ter a forma de um monômio igual a um, e as restrições de desigualdade podem somente ter a forma de um posinômio menor ou igual a um. Mas, várias extensões são facilmente manuseadas para que possam ser utilizados outros valores além do 1. Se f_i é um posinômio e g_i é um monômio, então a restrição $f_i(x) \leq g_i(x)$ pode ser manuseada expressando como $f_i(x)/g_i(x) \leq 1$ (desde que f_i/g_i seja um posinômio). Isto inclui como um caso especial uma restrição da forma $f_i(x) \leq a$, onde f_i é um posinômio e $a > 0$. De forma similar, se h_1 e h_2 são ambas funções monomiais, então pode-se manusear a restrição de igualdade $h_1(x) = h_2(x)$ expressando como $h_1(x)/h_2(x) = 1$ (desde que h_1/h_2 seja monomial) (HINDI, 2004).

Os problemas de otimização de programação geométrica podem ser transformados em problemas de otimização convexa, através da mudança de variáveis e uma transformação da função objetivo e das restrições (HINDI, 2004), conforme o trabalho (SAPATNEKAR et al., 1993) mostra.

2.2.1 Posinômios Generalizados

Posinômios generalizados, dentro do contexto de programação geométrica, são expressões que não podem ser consideradas posinômios, porém podem ser transformadas recursivamente em posinômios. Considerando o exemplo do máximo entre dois posinômios, já que essa expressão é bastante utilizada neste trabalho, dada a expressão $max\{f_i(x), g_i(x)\}$, as duas expressões podem ser limitadas da seguinte forma, considerando que a variável u é o limite superior da função:

$$\mathrm{f}_i(x) \leq u \qquad g_i(x) \leq u \tag{2.4}$$

2.3 Modelo de Atraso de Elmore

O modelo de atraso de Elmore (ELMORE, 1948) é uma aproximação para o valor de atraso através de uma rede RC em um circuito integrado. É amplamente usado pois é simples para computar (principalmente em redes estruturadas em árvore, que são a grande maioria das redes dentro de circuitos integrados) e é razoavelmente preciso.

Neste trabalho, o atraso de Elmore é calculado para as redes estruturadas em forma de árvore e é considerado para o cálculo a seguinte definição:

- Um nodo j é um descendente do nodo i na árvore T se o caminho do nodo raiz de T até j contém o nodo i. O nodo i é chamado de predecessor do nodo j. O $T_{d,i}$, delay do nodo i, de uma árvore RC é dada pela fórmula 2.5, considerando que P_i é o único caminho da raiz da árvore RC para o nodo i, e $desc(j)$ representa todos os nodos que são descendentes do nodo j na árvore (SAPATNEKAR, 1996).

$$T_{d,i} = \sum_{j \in P_i} R_j \sum_{K \in desc(j)} C_k \tag{2.5}$$

2.4 Caracterização de Células

Neste trabalho, a caracterização elétrica de células através de simulação SPICE é utilizada para a geração do arquivo *Liberty* (.lib) que possui os valores de estimativa de atraso, potência e ruído que cada célula caracterizada terá após sua fabricação.

Existem várias ferramentas comerciais com o objetivo de caracterizar células:

- ELC (Encounter Library Characterizer) da empresa Cadence (CADENCE, 2009a), que caracteriza as células utilizando um dos seguintes modelos: ECSM (Effective Current Source Model) para atraso, ruído, potência e modelagem estática ou CCS (Composite Current Source Model) para atraso, ruído e potência e pode utilizar como simulador elétrico o SPECTRE (CADENCE, 2010), da própria Cadence, ou o HSPICE (SYNOPSYS, 2010a), da empresa Synopsys.

- Nangate Library Characterizer da empresa Nangate (NANGATE, 2010b) que também suporta os modelos ECSM e CCS.

- Liberty NCX da empresa Synopsys (SYNOPSYS, 2010b) caracteriza as células utilizando o modelo CCS e para gerar um arquivo utilizando algum dos outros modelos, ECSM, Non-Linear Delay Models (NLDM), Non-Linear Power Models (NLPM), ela escala os valores do arquivo inicial em CCS.

Para este trabalho, as células foram caracterizadas eletricamente através da ferramenta ELC (Encounter Library Characterizer) da Cadence, utilizando como simulador elétrico o HSPICE.

Conforme citado acima, a ferramenta ELC utiliza os modelos baseados em fontes de corrente (CSM - Current Source Models) (CCS TIMING, 2006), ECSM que foi proposto pela empresa Cadence e CCS, proposto pela empresa Synopsys. Nesses modelos, a informação extraída para a estimativa de STA (Static Timing Analysis) é a forma de onda da corrente (CCS) ou da tensão (ECSM) de saída da porta lógica durante a sua transição (VALIDAÇÃO DE BIBLIOTECAS DE CÉLULAS PARA PROJETOS DE CIRCUITOS INTEGRADOS DIGITAIS, 2009). Para este trabalho as células foram caracterizadas utilizando o modelo ECSM, que é o modelo efetivamente utilizado pela empresa Cadence.

2.4.1 Caracterização de Células no ELC

Para fazer a caracterização da célula, o ELC necessita de três arquivos de entrada, conforme mostra a Figura 2.4, o arquivo de configuração, o arquivo com os subcircuitos no formato SPICE e o arquivo com o modelo do transistor. Utilizando estes três arquivos, é executada uma sequência de comandos para que o ELC faça a caracterização. Como saída do ELC tem-se o arquivo liberty (.lib).

Os arquivos necessários para fazer a caracterização devem ser calibrados no momento de sua construção para que a caracterização produza valores o mais próximo possível dos valores reais de funcionamento das células.

O arquivo que contém os subcircutos a serem caracterizados é um arquivo comum no formato SPICE, conforme é mostrado no Apêndice A.1. Os arquivos de modelo elétrico de transistor utilizados para fazer a caracterização são mostrados nos Apêndices A.3 e A.4, para $350nm$ e $45nm$, respectivamente. Como o arquivo de configuração possui características mais específicas, ele é detalhado a seguir.

2.4.2 Arquivo de Configuração

No arquivo de configuração, utilizado para caracterizar as células, são informados os valores nos quais a caracterização se baseia para fazer as simulações SPICE e gerar os resultados. Um exemplo de arquivo de configuração que foi usado para caracterizar as células na tecnologia $45nm$ é mostrado no Apêndice A.1.

O arquivo de conifguração inicia com a definição dos valores de “corner” para a simulação, que podem ser:

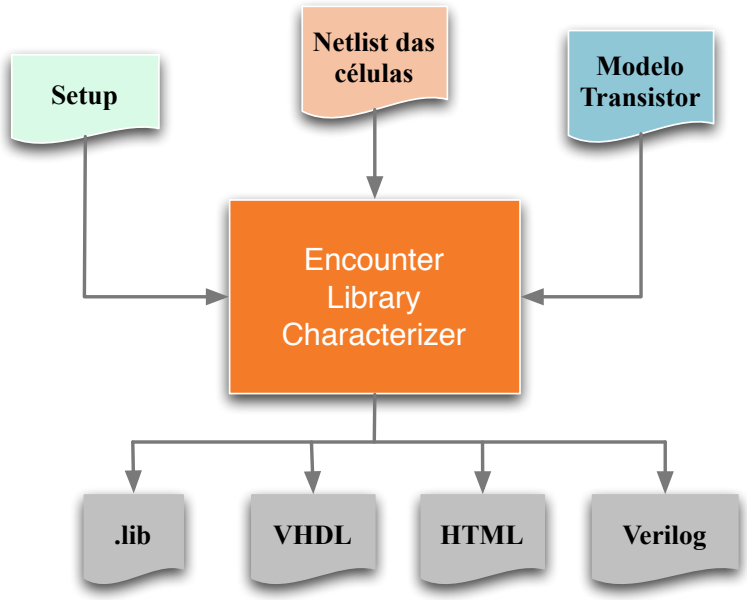


Figura 2.4: Fluxo do ELC para a caracterização de células.

- Típico (*Typical*): considera que o circuito trabalha em condições normais do ambiente, com temperatura de 25^0C e tensão padrão da tecnologia (3,3V para 350nm e 1,1V para 45nm).
- Pior (*Worst*): quando o circuito trabalha nas piores condições, com o valor de tensão 10% abaixo do normal (2,97V para 350nm e 0,99V para 45nm) e temperatura de 125^0C.
- Melhor (*Best*): quando o circuito trabalha em condições melhores, com temperatura a 0^0C e tensão 10% acima do normal (3,63V para 350nm e 1,21V para 45nm).

Para caracterizar uma célula, é necessário ter um conjunto de valores de inclinação da onda de entrada da porta e um conjunto de valores de capacitância de carga para os quais a porta é simulada, produzindo curvas dos valores de atraso, potência e ruído gerados. Os valores de inclinação da onda de entrada e de capacitância de carga das células, são baseados nos valores encontrados em arquivos *liberty* de bibliotecas de células comerciais para a tecnologia, tensão de alimentação e temperatura correspondentes ao que é necessário. Os valores são diferenciados conforme o tamanho dos transistores da célula, ou seja, para cada tamanho de célula há um conjunto de valores diferentes para os quais a célula é caracterizada. Além disso, há algumas células que possuem valores diferenciados para a sua caracterização, que é caso de alguns repetidores (*buffers*), multiplexadores (MUX), flip-flops (FF).

3 TRABALHOS RELACIONADOS

Neste capítulo, inicialmente são citados os trabalhos referentes ao dimensionamento de transistores, onde é feito um detalhamento dos dois métodos de dimensionamento mais relevantes para o desenvolvimento deste trabalho, (BOYD et al., 2005) e (SAPATNEKAR et al., 1993). Após, os trabalhos relacionados ao dimensionamento de portas são expostos e é apresentada uma explicação mais detalhada sobre os métodos de esforço lógico, *fanout* de quatro (FO4) e o dimensionamento usando programação geométrica mostrado em (BOYD et al., 2007), que é utilizado neste trabalho.

3.1 Dimensionamento de Transistores

O dimensionamento de transistores é um problema clássico que demanda a utilização de automação, EDA (*Electronic Design Automation*) e tem recebido muita atenção na literatura onde são encontrados inúmeros trabalhos.

TILOS (FISHBURN; DUNLOP, 1985) foi o primeiro algoritmo a tentar dimensionar transistores usando o modelo de atraso de Elmore, onde:

- Identifica um caminho crítico no atraso e utiliza um método heurístico para reduzir o atraso ao longo deste caminho.
- O processo é iterativo e para quando o caminho crítico encontra a restrição de atraso.
- Determina o tamanho mínimo para todos os transistores e só aumenta o tamanho dos transistores que fazem parte do caminho crítico.
- Tenta aumentar o tamanho do menor número possível de transistores.

Após este dimensionador, outros trabalhos surgiram, como é o caso do dimensionador de transistores que modela o atraso como funções posinomiais do tamanho dos transistores (SAPATNEKAR et al., 1993), formando um problema geométrico que é transformado em convexo, buscando, dessa forma, uma solução exata. Esse trabalho é mostrado na Seção 3.1.2.

(BORAH; OWENS; IRWIN, 1996) buscaram, em seu trabalho, dimensionar os transistores de circuitos CMOS para baixo consumo de potência sobre uma restrição de atraso, onde:

- Mostra que o consumo de potência estática é uma função convexa da área ativa do circuito.

- Utiliza uma formulação analítica para calcular a dissipação de potência considerando o tamanho do transistor, incluindo ambas as dissipações de potências, a capacitiva e a de curto circuito.

- Faz simulações SPICE para confirmar os resultados do modelo analítico.

- Apresenta a ferramenta Aesop que faz o dimensionamento dos transistores buscando minimizar o consumo de potência enquanto a restrição de atraso é atendida, sem inserir mudanças na estrutura do circuito e no número de portas.

- Utiliza algoritmos lineares, que também são convexos, para computar o dimensionamento.

O trabalho (BEECE et al., 2010) é mais recente e utiliza uma abordagem mais incrementada que os trabalhos anteriores, dimensionando transistores de circuitos digitais customizados e de alto desempenho. Além disso, utiliza considerações de rendimento paramétrico, que é a sensibilidade do circuito às variações no processo de fabricação, temperatura e tensão de alimentação (EISNER, 2003). Dessa forma, (BEECE et al., 2010) endereçam o problema de dimensionamento estatisticamente, onde os tamanhos dos transistores são automaticamente ajustados para maximizar o rendimento paramétrico para um dado desempenho, ou maximizar o desempenho para um requerido rendimento paramétrico. Resolve um problema de otimização não-linear, onde a função objetivo é diretamente dependente das variações de processo estatísticas.

Um dos primeiros trabalhos considerando dimensionamento de transistores em nosso grupo de pesquisa foi o trabalho de Cristiano Santos (SANTOS, 2005), o qual tratava do dimensionamento de transistores considerando somente os caminhos críticos, resultando em várias publicações, entre elas (SANTOS et al., 2003), (SANTOS et al., 2004), (SANTOS et al., 2005/a), (SANTOS et al., 2005/b). O trabalho (SANTOS, 2005) faz o dimensionamento de transistores para uma ferramenta de geração automática de leiautes (LAZZARI et al., 2003) e não utiliza métodos de otimização baseados em programação matemática.

Para o desenvolvimento do presente trabalho, duas abordagens da literatura foram estudadas mais profundamente, (SAPATNEKAR et al., 1993) e (BOYD et al., 2005), e uma explicação mais detalhada das mesmas é mostrada nas seções seguintes, considerando que as duas calculam o atraso pelo modelo de Elmore.

3.1.1 Método de Dimensionamento de Transistores proposto por (BOYD et al., 2005)

Inicialmente, é feita uma explicação dos componentes de uma porta lógica e seu funcionamento, para depois iniciar a descrição de como o problema é modelado e tratado por este método.

Uma porta CMOS estática é uma combinação de uma rede *pull-down* de transistores NMOS e uma rede *pull-up* de transistores PMOS, como mostrado na Figura 3.1. A rede *pull-up* fornece uma conexão entre a saída e a alimentação do circuito quando a saída da porta é logicamente alta (Vdd), e a rede *pull-down* fornece uma conexão entre a saída da porta e *ground* quando a saída da porta é logicamente baixa (Gnd).

A idéia do trabalho (BOYD et al., 2005) é escolher a largura $w_1, ..., w_p$ dos transistores nas redes *pull-down* e *pull-up* em uma porta lógica, sujeito a restrições da área da porta, atraso, capacitância de entrada, potência, etc. Para uma simples porta NAND, por

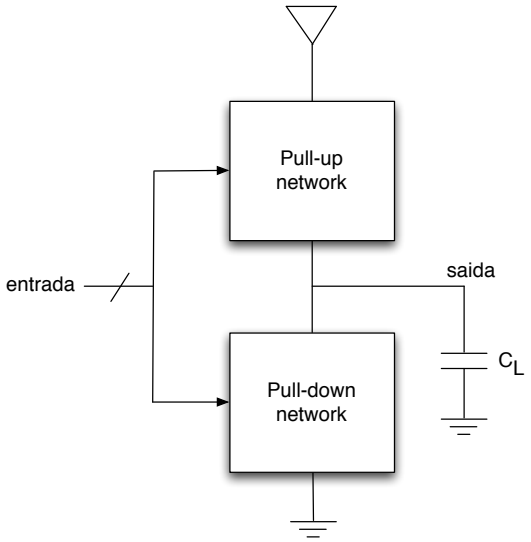


Figura 3.1: Redes *pull-up* e *pull-down* de uma porta lógica CMOS.

exemplo, há quatro larguras para escolher, uma para cada transistor que compõe a porta. As larguras dos transistores afetam a área da porta, a capacitância de entrada, o atraso e as potências dinâmica e estática da porta.

O trabalho (BOYD et al., 2005) segue uma abordagem básica, utilizando restrições que são facilmente formuladas. Restrições de manufatura especificam os valores mínimo e máximo permitidos para a largura dos transistores. A área e a capacitância de entrada dos transistores podem ser representadas aproximadamente como uma função linear ou afim das larguras dos dispositivos, com coeficientes positivos, sendo assim, é posinomial.

Para o projeto de portas que não possuam muitas entradas, é comum modelar o comportamento da porta para cada transição de entrada, modelando a perda de energia e o atraso para a saída.

O problema de projeto de uma porta básica pode ter a seguinte forma:

$$\begin{array}{ll} \text{minimizar} & D = max\{D_1, ..., D_k\} \\ \text{sujeito a} & w_i^{\min} \leq w_i \leq w_i^{\max}, i = 1, ..., p, \\ & A \leq A^{\max}, \\ & (1/K)\sum_{K=1}^{K} E_K \leq E^{\max}, \\ & C_i^{in} \leq C^{in,max}, i = 1, ..., m, \end{array} \tag{3.1}$$

onde A denota a área da porta, D_K é o atraso para a transição K, E_K é a perda de energia durante a transição K, e C_i^{in} denota a capacitância de entrada para a entrada da porta i. Sendo que, K é o número total de transições e m é o número de entradas para a porta. Portanto, procura-se as larguras dos transistores que fazem parte da porta onde o máximo atraso seja minimizado, sujeito a uma área limite $A_{\max}$, a uma perda de energia média que não excede $E_{\max}$ e a um valor máximo de capacitância de entrada $C^{in,max}$, onde todas as expressões são posinômios generalizados.

3.1.1.1 Modelo de Chaves RC para uma Porta Lógica

A primeira etapa é modelar a porta lógica CMOS usando o modelo de chaves, onde as chaves são abertas ou fechadas, dependendo do valor das entradas. A porta é vista como um conjunto de árvores RC, uma para cada possível vetor de entrada e o atraso da porta é

o máximo atraso gerado pelas árvores que a compõe.

Alguns vetores de entrada podem gerar ciclos/laços, estruturas que não são árvores, que não podem ser modeladas em uma forma geométrica. Felizmente, árvores RC (resistência e capacitância) são sempre limites superiores de atraso para qualquer rede RC gerada. Portanto, estruturas que não são árvores são simplesmente ignoradas sem qualquer perda na estimativa do atraso da porta usando o modelo de Elmore.

A perda de energia é modelada como a perda total de energia no circuito RC e o atraso da porta é modelado como o atraso de Elmore para o nodo de saída da porta. Sendo que, ambas as medidas são funções posinomiais da largura dos dispositivos.

A Figura 3.2 mostra um modelo simples de circuito RC a nível de chave para um transistor. Pode-se ver que o circuito RC consiste de uma resistência R de fonte para dreno, que modela a resistência do canal quando o dispositivo está ligado e cinco capacitâncias parasitas entre os quatro terminais: substrato (*bulk*)(B), *gate*(G), fonte(S) e dreno(D). Apesar dos transistores PMOS e NMOS serem diferentes, eles têm o mesmo modelo de chaves, somente com parâmetros (capacitâncias e resistência) diferentes. Na abordagem apresentada por (BOYD et al., 2005), a capacitância de *gate* para dreno foi ignorada para simplificar o modelo, mas ela pode ser manuseada de forma compatível com a Programação Geométrica. Assume-se que cada um desses parâmetros é uma função posinomial da largura dos transistores. Em uma modelagem simples, por exemplo, a resistência é inversamente proporcional à largura dos transistores e as capacitâncias são funções lineares (ou afins da largura).

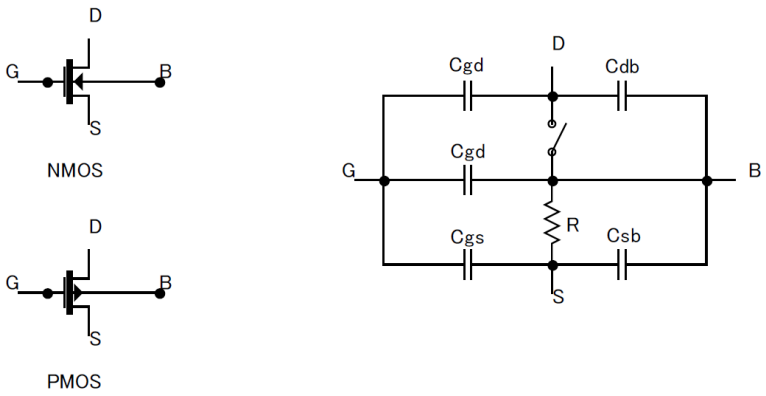


Figura 3.2: Transistores MOS (esquerda) e modelo RC a nível de chaves (direita).

Substituindo o modelo RC a nível de chaves para cada transistor em uma porta, obtém-se um modelo de chaves de um circuito RC para uma porta e, para cada transição de entrada, obtém-se um circuito RC, com condições iniciais conhecidas.

Para ilustrar a abordagem descrita, é considerada como exemplo uma porta NAND de duas entradas carregando uma capacitância C_L, conforme mostra a Figura 3.3. Usando o modelo de dispositivo RC a nível de chaves, a porta NAND pode ser modelada como o circuito RC mostrado na Figura 3.4(a) que é eletricamente equivalente ao circuito RC da Figura 3.4(b), onde é feita a seguinte simplificação:

$C_1 = C_{db1} + C_{db2} + C_{db3} + C_L$,

$C_2 = C_{sb3} + C_{db4}$.

As capacitâncias de entrada para os pinos A e B são:

$C_A^{in} = C_{gb2} + C_{gs2} + C_{gb3} + C_{gs3}$,

$C_B^{in} = C_{gb1} + C_{gs1} + C_{gb4} + C_{gs4}$.

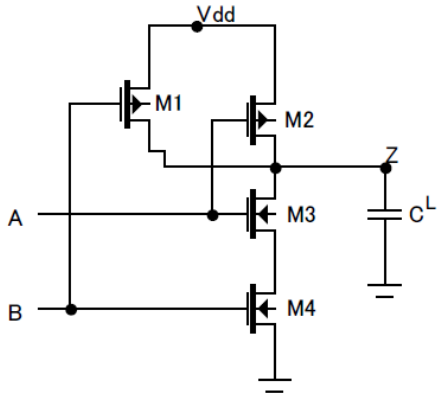


Figura 3.3: Porta NAND de duas entradas

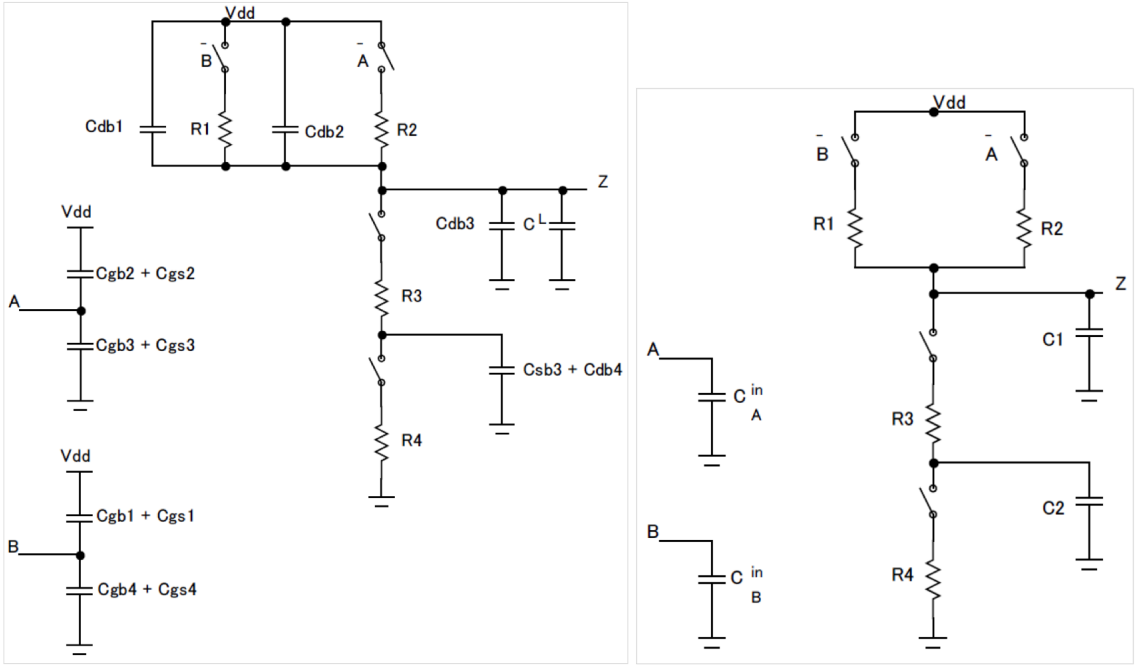


Figura 3.4: Modelo de circuito RC da porta NAND de duas entradas (a) e o seu modelo eletricamente equivalente (b)

As resistências e capacitâncias no circuito RC chaveado são posinômios das larguras dos transistores. Com isso, pode-se analisar cada transição separadamente. Há quatro estados de entrada e, portanto, 12 possíveis transições. Destas 12 transições, em três a saída Z desce de V_{dd} para Gnd e em três a saída Z sobe de Gnd para V_{dd}.

Segue um exemplo que ilustra o modelo de atraso e de energia para uma transição subindo, onde: $B : 1- > 0$ e $A : 1- > 1$. Antes da transição, as chaves A e B estão fechadas, e as chaves $\overline{A}$ e $\overline{B}$ estão abertas. A saída Z está em Gnd, e as capacitâncias C_1 e C_2 possuem carga inicial igual a Gnd. Quando a entrada B da porta desce para Gnd,

a chave B abre, e a chave $\overline{B}$ fecha. A representação deste circuito RC pode ser vista no lado esquerdo da Figura 3.5. O atraso de Elmore para o nodo saída é
$D = R_1 * (C_1 + C_2)$,
e a dissipação de energia é
$E = (C_1 + C_2) * V_{dd}^2/2$. Estes são posinômios dos tamanhos dos transistores.

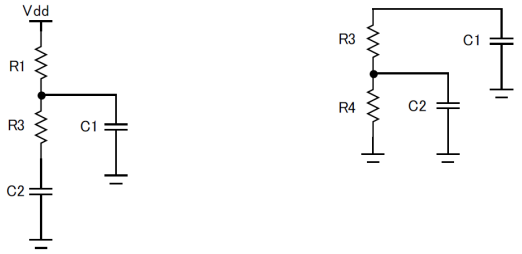


Figura 3.5: Modelo RC de uma porta NAND de duas entradas para duas transições na entrada: A: 1->1 e B 1->0 (esquerda); A e B: 0->1 (direita)

Considerando outro exemplo, onde a transição é em ambas as entradas da porta, sendo que A e B sobem de Gnd para V_{dd}, fazendo a saída Z descer de V_{dd} para Gnd. Nesta transição as chaves A e B fecham, e as chaves $\overline{A}$ e $\overline{B}$ abrem. Isto resulta na árvore RC mostrada no lado direito da Figura 3.5. Para esta transição, a tensão inicial v_2^{init} em C_2 não é bem definida, ela pode ser algum valor entre Gnd e V_{dd}, neste caso é considerado o pior caso que seria V_{dd}. Dessa forma, o atraso de Elmore para o nodo saída Z é $D = R_3 * C_1 + R_4 * (C_1 + C_2)$ e a dissipação de energia durante esta transição é $E = (C_1 + C_2) * V_{dd}^2/2$, que são posinômios da largura dos transistores.

As expressões de atraso e energia para as seis transições de entrada que resultam em uma transição na saída são dadas na Tabela 3.1. Para as duas últimas transições, a expressão exata depende de v_2^{init}, sendo usado o pior caso limite, V_{dd}. Na terceira transição, quando ambas as entradas descem para Gnd, o circuito RC resultante não é uma árvore, pois as duas resistências (R_1 e R_2) formam um ciclo. Como resultado, a expressão de atraso não é posinomial de R_i. Neste caso, o valor de atraso é sempre menor que o valor dado quando somente uma entrada desce de V_{dd} para Gnd, então o atraso para esta transição pode ser ignorado. O atraso máximo sobre todas as transições é $D = \max\{R_3C_1 + R_4(C_1 + C_2), R_1(C_1 + C_2), R_2C_1\}$ que é um posinômio generalizado. Os modelos de energia mostrados na Tabela 3.1 são também posinômios generalizados.

Tabela 3.1: Expressões de atraso para a porta NAND de duas entradas para as seis transições de entrada que rendem uma transição na saída.

A	**B**	**Z**	**Atraso**	**Energia Dissipada**
$1->1$	$1->0$	$0->1$	$R_1(C_1 + C_2)$	$(C_1 + C_2)V_{dd}^2/2$
$1->0$	$1->1$	$0->1$	R_2C_1	$C_1V_{dd}^2/2$
$1->0$	$1->0$	$0->1$	$C_1R_1R_2/(R_1 + R_2)$	$C_1V_{dd}^2/2$
$1->1$	$0->1$	$1->0$	$R_3C_1 + R_4(C_1 + C_2)$	$(C_1 + C_2)V_{dd}^2/2$
$0->1$	$1->1$	$1->0$	$R_3C_1 + R_4C_1$	$(C_1 + C_2)V_{dd}^2/2$
$0->1$	$0->1$	$1->0$	$R_3C_1 + R_4(C_1 + C_2)$	$(C_1 + C_2)V_{dd}^2/2$

3.1.2 Método de Dimensionamento de Transistores Proposto por (SAPATNEKAR et al., 1993)

O objetivo do trabalho "Uma solução exata para o problema de dimensionamento de transistor para circuitos CMOS usando otimização convexa" (*An exact solution to the transistor sizing problem for CMOS circuits using convex optimization*, título original em inglês) (SAPATNEKAR et al., 1993) é minimizar a área de um estágio combinacional restringindo seu atraso para ser menor que uma dada especificação. Considerando que o atraso de um circuito pode ser controlado variando o tamanho da largura dos transistores, provocando um gasto adicional na área do circuito. O atraso é computado permitindo formas de onda com tempos de subida e descida diferentes de zero e permite que os atrasos de subida e descida sejam calculados separadamente. Os requisitos de tempo de *hold* não são considerados. A forma de onda de transição da saída é modelada como uma função que varia linearmente com o tempo.

Considerando um circuito CMOS combinacional com um conjunto de nós de entrada(s) primária(s) e saída(s) primária(s), o circuito é dividido em componentes (conjunto de transistores conectados pelos nós de dreno e fonte) e cada componente é representado como um grafo não direcionado. O objetivo é encontrar o pior caso de atraso de Elmore para um nó de saída do componente sobre todas as possíveis combinações de entrada, considerando que o caminho de pior caso é o caminho mais resistivo.

A área é medida como a soma dos tamanhos dos transistores que compõem o circuito e é dada por uma função afim dos mesmos (FISHBURN; DUNLOP, 1985). Considerando que n denota o número de transistores de um circuito combinacional e $X = [X_1, X_2, ..., X_n]$ é um vetor n-dimensional dos tamanhos dos transistores, a área total do circuito é posinomial em X e é dada por:

$$Area(x) = \sum_{i=1}^{n} X_i. \tag{3.2}$$

O atraso ao longo de um caminho do circuito pode ser representado por funções posinomiais dos tamanhos dos transistores e o atraso do circuito é definido pelo máximo dos atrasos de todos os caminhos. A equação para o atraso global $Delay(x)$ usando o modelo de atraso proposto por (SAPATNEKAR et al., 1993), é um posinômio na forma

$$Delay(x) = \sum_{j} \gamma_j \prod_{i=1}^{n} x_i^{\alpha_{ij}} = \sum_{j} \gamma_j x_1^{\alpha_{1j}} x_2^{\alpha_{2j}} ... x_n^{\alpha_{nj}} \tag{3.3}$$

onde, $\gamma_j \geq 0$, $\alpha_{ij} \in \{-1, 0, 1\}\ \forall i = 1, 2, ..., n$. α_{ij} pode ser "-1" somente para transistores críticos, ou seja, os transistores que estão no caminho mais resistivo de um componente e no cálculo do atraso RC contribuem com expoente "-1", atuando como uma resistência. Os demais transistores podem ainda contribuir com expoente "1", quando eles atuam como uma capacitância, ou podem não contribuir para o produto RC, ou seja, o expoente é igual a "0".

O problema de dimensionamento é formulado como um problema de programação geométrica, permitindo encontrar a solução ótima.

$$\begin{aligned} &\text{minimizar} \quad Area(x) = \textstyle\sum_{i=1}^{n} X_i \\ &\text{sujeito a} \quad\ \ Delay(x) \leq T_{spec}, \end{aligned} \tag{3.4}$$

Uma função posinomial pode ser mapeada dentro de uma função convexa através de uma transformação de variável $(x_i) = (e^{z_i})$. Fazendo esta transformação, o problema de programação geométrica original de dimensionamento de transistores é tratado como um problema de programação convexa de minimização de uma função convexa sobre um conjunto de restrições convexas:

$$\begin{aligned} &\text{minimizar} \quad Area(z) = \textstyle\sum_{i=1}^{n} e^{z_i} \\ &\text{sujeito a} \quad Delay(z) \leq T_{spec}, \end{aligned} \tag{3.5}$$

O problema é resolvido utilizando um algoritmo de otimização convexa, considerando sua propriedade unimodal, onde algum mínimo local é também um mínimo global, garantindo encontrar a solução exata.

3.2 Dimensionamento de Portas

Há muitos trabalhos na literatura em dimensionamento de portas. O método mais amplamente conhecido é esforço lógico (SUTHERLAND; SPROULL; HARRIS, 1999), (SUTHERLAND; SPROULL, 1991), que fornece heurísticas rápidas ou orientações de projeto para resolver o problema de dimensionamento de portas aproximadamente. Em (BERKELAAR; JESS, 1990) e (BHATTACHARYA; RANGANATHAN, 2008) o dimensionamento é resolvido através de Programação Linear, já em (CIRIT, 1987), (SAPATNEKAR; CHUANG, 2000) e (MAHALINGAM; RANGANATHAN, 2005) é utilizada Programação Não Linear. Modelos analíticos de potência, atraso e área são utilizados em (HOPPE et al., 1990) e (BORAH; OWENS; IRWIN, 1995). As metodologias tradicionais de dimensionamento de portas (BOYD et al., 2005), (BOYD et al., 2007), (SAPATNEKAR et al., 1993), (SINGH et al., 2005) usam o modelo de atraso de Elmore (ELMORE, 1948), sendo que muitas delas formulam o dimensionamento como um problema de Programação Geométrica (PG) baseado em restrições posinomiais e utilizam um resolvedor de PG para obter uma solução exata, como é o caso deste trabalho.

O dimensionamento de portas pode ser tratado junto com outras técnicas de otimização de circuitos, como é o caso do dimensionamento de fios e a inserção de *buffers*. O trabalho (CHEN; CHU; WONG, 1998) faz o dimensionamento de portas lógicas e de fios simultaneamente considerando o modelo de atraso de Elmore, onde as otimizações são realizadas considerando o projeto inteiro. Apresenta um algoritmo que minimiza a área total sujeito ao limite de um máximo atraso, onde ele é iterativo e com garantia de convergência em soluções ótimas globais. Os segmentos de fio são modelados usando o modelo π (pi).

O artigo (JIANG; SHI, 2008) apresenta um algoritmo onde a inserção de *buffers* e o dimensionamento de portas são manuseados de forma similar, onde há um balanceamento entre o custo e a modelagem linear de um atraso não linear. Foi utilizada uma técnica de partição para dividir os circuitos em sub-circuitos, aplicando o esquema de dividir e conquistar. As interconexões são modeladas usando o modelo RC π, utilizando o modelo de atraso de Elmore. Considerando que a solução de dimensionamento de portas e a solução de inserção de buffers afetam uma a outra e, pode-se ter soluções subótimas se estas técnicas forem aplicadas sequencialmente em vez de simultaneamente, o artigo de (ALPERT et al., 2002) extende o algoritmo de inserção de buffers de van Ginneken (GINNEKEN, 1990) para incorporar simultaneamente o dimensionamento de portas.

Diversos trabalhos sobre dimensionamento de portas buscam obter estimativas de

atraso mais precisas, para que o resultado seja o mais próximo possível do real. No trabalho (TENNAKOON; SECHEN, 2005) é incluído um esquema de propagação de atraso que combina tempos de chegada e a inclinação da onda de entrada. Utiliza uma variável controlando a largura dos transistores NMOS e outra controlando a largura dos transistores PMOS. A formulação usada no trabalho (TENNAKOON; SECHEN, 2005), forma o modelo de atraso caracterizando um biblioteca de portas, com limites máximo e mínimo nos tamanhos de transistores. O modelo de atraso é dependente da inclinação da onda de entrada, tamanhos dos transistores e carga de saída. Usa STA (*Static Timing Analysis*) que utiliza um método de propagação de atraso realístico com uma combinação dos tempos de chegada e suas inclinações na onda de entrada para determinar a propagação de atraso, sendo que o atraso é a restrição do problema e o objetivo é minimizar a área.

(GUTHAUS et al., 2005) modelam os atrasos como distribuições estatísticas durante a análise e otimização. Esse é o primeiro artigo que utiliza análise de *timing* estática estatística (*Statistical Static Timing Analysis* - SSTA) para guiar o dimensionamento de portas. Além disso, parâmetros de processos são considerados usando um modelo de atraso linear. O problema é formulado, onde o objetivo é computar os tamanhos das portas lógicas tal que uma restrição de atraso seja encontrada enquanto consome a menor área e potência.

Há ainda trabalhos que buscam dimensionar portas lógicas para constituir uma biblioteca. O artigo (HU; KETKAR; HU, 2007) trata do problema de dimensionamento discreto de portas e propõe um algoritmo que é uma abordagem de programação dinâmica guiada por uma solução contínua, com o objetivo de minimizar a área sob uma restrição de atraso. Dessa forma, integra a qualidade da solução da programação dinâmica com o baixo tempo de execução da solução contínua de arredondamento. (BEEFTINK et al., 1998) também propõem um algoritmo que busca selecionar um bom conjunto de tamanhos de portas para as portas primitivas de uma biblioteca de *standard-cells*. Para isso, utiliza uma função de erro que quantifica a discrepância dada quando um tamanho de porta requerido é trocado por um tamanho disponível na biblioteca. O problema de seleção do tamanho da porta é encontrar o conjunto de tamanhos de porta que minimize essa função de erro.

(JOSHI; BOYD, 2008) preocupam-se com a escalabilidade do circuito, ou seja, o tempo de execução necessário para dimensionar circuitos grandes, com 100.000 portas, 1 milhão de portas. Para isso, utiliza um número fixo de iterações para computar a solução, tornando a complexidade linear no número de portas. Otimiza o circuito utilizando o modelo de atraso RC para cada porta, que não é tão preciso, e depois faz otimizações locais.

Nas seções seguintes, são detalhados os métodos de dimensionamento por esforço lógico e *fanout* de 4, que são bastante utilizados e o método de dimensionamento proposto por (BOYD et al., 2007) onde é utilizado programação geométrica para resolver o problema de dimensionamento e no qual este trabalho se baseia.

3.2.1 Esforço Lógico

O método de esforço lógico é um método simples que otimiza redes de portas para obterem maior velocidade. Esta seção pretende dar uma ideia de como é seu funcionamento baseada nos trabalhos (SUTHERLAND; SPROULL, 1991), (WESTE; HARRIS, 2005) e (SUTHERLAND; SPROULL; HARRIS, 1999).

O esforço lógico de uma função lógica depende principalmente da topologia do seu circuito e ligeiramente das propriedades elétricas do processo de fabricação usado para construí-lo. A porta lógica considerada mais simples é a porta inversora que é muito utili-

zada como amplificador para alimentar grandes capacitâncias. Portas lógicas que computam outras funções em CMOS, utilizam mais transistores que podem estar conectados em paralelo ou em série. As conexões em série prejudicam a capacidade da porta de passar corrente, pois aumentam a resistência da mesma. O método de esforço lógico quantifica esses efeitos para simplificar a análise do atraso, assinalando um esforço lógico para cada função lógica baseada em um inversor, o qual tem seu esforço lógico definido como um. Dessa forma, o esforço lógico para qualquer outra função lógica descreve quanto pior ela é do que um inversor para fornecer a corrente de saída, dando um valor equivalente da capacitância de entrada. Em CMOS, o esforço lógico de cada entrada de uma função lógica comum de duas entradas fica em cerca de 4/3 para uma NAND e 4 para uma XOR. O esforço lógico de funções com mais de duas entradas é geralmente maior (SUTHERLAND; SPROULL, 1991).

O esforço lógico para estágios individuais da lógica podem ser combinados para encontrar o esforço lógico de redes. Quando a corrente circula entre diversos estágios, como uma cadeia, o esforço global é o produto dos esforços individuais. Quando vários dispositivos lógicos são alimentados por uma mesma fonte, o esforço global é a soma dos esforços individuais. Circuitos compostos com menor esforço lógico global podem ser feitos para executar mais rápido que circuitos logicamente equivalentes com maior esforço lógico (SUTHERLAND; SPROULL; HARRIS, 1999).

3.2.1.1 Atraso em uma Porta Lógica

O método de esforço lógico reformula um modelo de atraso *RC* convencional e simples em uma porta lógica CMOS. Antes de ser iniciada a explicação de como é calculado o atraso, visando isolar os efeitos de um processo particular de fabricação, define-se uma unidade de atraso para expressar todos os atrasos em função dela. Assim, o atraso absoluto (d_{abs}) é o produto do atraso, sem unidade, d, pela unidade básica de atraso, τ:

$$d_{abs} = d\tau \tag{3.6}$$

O atraso em uma porta lógica pode ser expressado como a soma de dois componentes, uma parte fixa, chamada atraso parasita, p, e uma parte proporcional a carga na saída da porta, chamada de atraso do esforço, f:

$$d = f + p \tag{3.7}$$

O atraso de esforço depende da carga e das propriedades da porta lógica alimentando a carga. Devem ser definidos dois termos para estes efeitos: o esforço lógico, g, e o esforço elétrico, h. O atraso de esforço da porta lógica é o produto destes dois fatores:

$$f = gh \tag{3.8}$$

O esforço lógico captura o efeito da topologia da porta lógica na sua habilidade de produzir corrente para a saída. Ele é independente dos tamanhos dos transistores no circuito. O esforço elétrico descreve como o comportamento elétrico da porta afeta a performance e como o tamanho dos transistores determinam a capacidade da porta de alimentar uma carga. O esforço elétrico também é conhecido como $fanout$ e é definido por:

$$h = C_{out}/C_{in} \tag{3.9}$$

onde C_{out} é a capacitância que a porta lógica necessita carregar e C_{in} é a capacitância apresentada pela porta lógica para um de seus terminais de entrada.

Combinando as equações 3.7 e 3.8, obtém-se a equação básica que modela o atraso através de uma única porta lógica, em unidades de τ:

$$d = gh + p \tag{3.10}$$

Esta equação mostra claramente que, ambos, o esforço lógico e o esforço elétrico contribuem para o atraso na mesma proporção. Esta formulação separa τ, g, h e p, as quatro contribuições para o atraso. Pode-se notar que p e g são independentes do tamanho dos transistores na porta lógica, enquanto h relata diretamente os tamanhos dos transistores.

Valores do esforço lógico para alguns circuitos são mostrados na Tabela 3.2. O esforço lógico é definido tal que um inversor tem um esforço lógico de um, conforme já citado anteriormente. Por isso, τ é o atraso de um inversor ideal sem atrasos parasitas que alimentam outros inversores idênticos.

Na Tabela 3.2 pode-se notar que as células mais complexas possuem esforço lógico maior. Além disso, o esforço lógico da maioria das portas lógicas cresce com o número de entradas da porta. Portas lógicas maiores ou mais complexas exibirão maior atraso. Estas propriedades fazem com que o esforço lógico seja útil para contrastar diferentes escolhas de estruturas lógicas. Projetos que minimizam o número de estágios da lógica requerem mais entradas para cada porta lógica e por isso têm maior esforço lógico. Projetos com poucas entradas, e por isso menos esforço lógico por estágio podem requerer mais estágios de lógica. O método de esforço lógico permite escolher entre tais alternativas. Na Tabela 3.2, n é uma variável que representa o número de entradas de uma porta lógica e os valores separados por hífen ($-$), como por exemplo 6-12, significam que o esforço lógico pode ir do valor 6 até o valor 12.

Tabela 3.2: Esforço lógico para entradas de portas CMOS, assumindo que o transistor PMOS possui a metade da condutância do transistor NMOS com geometria idêntica (SUTHERLAND; SPROULL; HARRIS, 1999).

Tipo de porta	**Número de entradas**					
	1	**2**	**3**	**4**	**5**	**6**
Inversor	1					
NAND		4/3	5/3	6/3	7/3	(n + 2)/3
NOR		5/3	7/3	9/3	11/3	(2n + 1)/3
Multiplexador		2	2	2	2	2
XOR (paridade)		4	6-12	16-32		

O esforço elétrico é usualmente expressado como uma razão das larguras dos transistores em vez de capacitâncias reais. Se for assumido que todos os transistores possuem o mesmo comprimento mínimo, a capacitância de *gate* de um transistor é proporcional a sua largura. Como muitas portas lógicas alimentam outras portas lógicas, ambos C_{in} (capacitância de entrada) e C_{out} (capacitância de saída) podem ser expressados em termos da largura dos transistores (SUTHERLAND; SPROULL, 1991).

3.2.2 Dimensionamento via Regra de *Fanout*

Um circuito com dimensionamento de portas mais preciso terá um funcionamento melhor, ou seja, será mais rápido, terá uma área e um consumo potência menor e, devido à importância desta etapa, encontram-se na literatura inúmeras formas de dimensionar um circuito. Uma maneira simples e eficiente de fazer o dimensionamento das portas é chamada de *fanout-of-4 inverter* (FO4) (SUTHERLAND; SPROULL; HARRIS, 1999) (WESTE; HARRIS, 2005) (RABAEY; CHANDRAKASAN; NIKOLIC, 2002). É um processo independente das métricas de atraso, área e potência e consiste em dimensionar as portas de um circuito baseada no atraso de um inversor, ou seja, todas as portas do circuito devem ser modeladas com base no atraso de um inversor. Considera-se que o menor atraso para atacar uma carga X é obtido por uma cadeia de n inversores onde cada inversor é quatro vezes maior que o anterior. O valor de *fanout* é dado pelo seguinte cálculo:

$$Fanout = \frac{C_{load}}{C_{in}} \tag{3.11}$$

onde, C_{load} é a capacitância de carga total da porta em questão e C_{in} é a capacitância de entrada dessa porta. Para a regra de *fanout* de 4, o cálculo torna-se:

$$C_{in} = \frac{C_{load}}{4} \tag{3.12}$$

A regra de *fanout* de 4 baseia-se na ideia que se um inversor tiver uma carga 4 vezes maior que a sua capacitância de entrada, ou seja, o inversor de tamanho 1 está ligado a um inversor de tamanho 4 (ou 4 inversores de tamanho 1) e esse inversor de tamanho 4 está ligado a um inversor de tamanho 16 (ou a uma quantidade de inversores cuja a soma de cargas é igual a 16), e assim por diante, conforme a Figura 3.6, o atraso desses inversores é similar ao atraso que o teriam se estivessem ligados a uma carga menor, não desperdiçando potência, neste caso. Já, se a carga for maior que 4 vezes, o tempo para alimentar a carga é maior, mas economiza-se energia. A proporção deriva de uma minimização, considerando o número e o tamanho dos buffers.

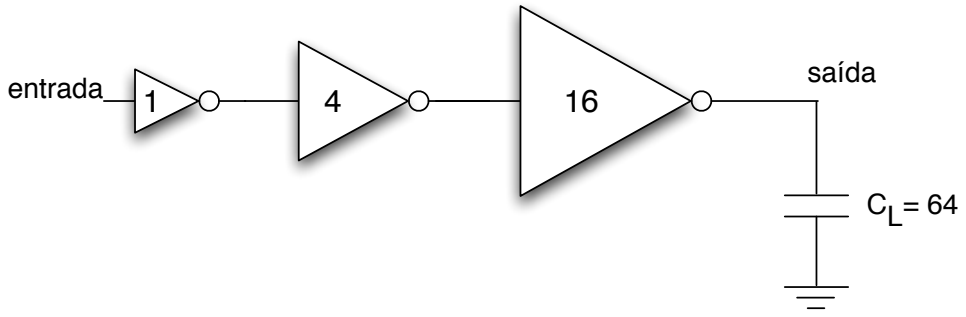


Figura 3.6: Dimensionamento pela metodologia de *Fanout* de 4.

3.2.3 Método de Dimensionamento de Portas Proposto por (BOYD et al., 2007)

Conforme citado anteriormente, o dimensionador de portas desenvolvido na dissertação é baseado neste trabalho (BOYD et al., 2007). A seguir é mostrado como (BOYD

et al., 2007) trata o dimensionamento e como são calculados os valores necessários para executar a otimização.

Primeiramente, é associada uma variável X_i para cada porta lógica que compõe o circuito. Essa variável representa o fator de escala para o tamanho dos transistores. Os fatores de escala das portas, que são as variáveis de otimização, afetam a área total do circuito, a potência consumida e a velocidade do mesmo. A área de uma porta dimensionada é proporcional ao fator de escala X_i, e a área total do circuito é dada pela equação 3.13, considerando que a_i é a área da porta i de tamanho unitário.

$$A = \sum_{i=1}^{n} a_i X_i \tag{3.13}$$

Da mesma forma, a perda de energia quando uma porta transiciona é também aproximadamente proporcional ao fator de escala. Então, o consumo total de potência do circuito é dado pela equação 3.17, onde f_i é a frequência de transição da porta, e e_i é a perda de energia quando a porta de tamanho unitário transiciona.

$$p = \sum_{i=1}^{n} f_i e_i X_i \tag{3.14}$$

A potência e a área total do circuito são ambas funções posinomiais do fator de escala. O valor de perda de energia é previamente calculado usando simulação SPICE, sendo escalado conforme o fator de escala da porta em questão.

Cada porta tem uma capacitância de entrada C_i que é uma função afim de seu fator de escala, e uma resistência de *driving* R_i, que é aproximadamente inversamente proporcional ao fator de escala.

O atraso D_i de uma porta é o produto de sua resistência de *driving*, e a soma das capacitâncias de entrada das portas onde sua saída é conectada (se não for uma porta de saída) ou sua capacitância de carga (se for uma porta de saída):

$$D_i = \begin{cases} R_i \sum_{j \in F(i)} C_j \\ R_i C_i^{out} \end{cases} \tag{3.15}$$

Na equação 3.15, $F(i)$ é o conjunto de portas cuja entrada está conectada a saída da porta i e C_i^{out} representa a capacitância de carga de uma porta de saída. Combinando estas três fórmulas, pode-se ver que o atraso da porta D_i é uma função posinomial dos fatores de escala.

A velocidade do circuito é medida usando seu maior ou pior caso de atraso, D, que é o atraso total máximo ao longo de algum caminho do circuito. Desde que o atraso de cada porta seja posinomial e não haja ciclos, o atraso total ao longo de algum caminho é também posinomial, sendo que ele é a soma dos atrasos das portas.

Pode-se dizer que o problema de dimensionamento de portas é o problema onde escolhe-se o fator de escala para dar o atraso mínimo sujeito a limites na área e potência, conforme apresentado na expressão 3.16, onde P^{max} e A^{max} são limites dados para área e potência totais, sendo que D é um posinômio generalizado.

$$\begin{aligned} &\text{minimizar} && D \\ &\text{sujeito a} && P \leq P^{max}, \quad A \leq A^{max} \\ & && x^i \geq 1, \quad i = 1, ..., n. \end{aligned} \tag{3.16}$$

Considerando o circuito de exemplo mostrado na Figura 3.7 que possui 7 portas lógicas e somente 7 caminhos, o pior caso de atraso é dado por:

$$\begin{aligned} \text{D} = \max\{&D_1 + D_4 + D_6, D_1 + D_4 + D_7, D_2 + D_4 + D_6, \\ &D_2 + D_4 + D_7, D_2 + D_5 + D_7, D_3 + D_5 + D_6, D_3 + D_7\}. \end{aligned} \tag{3.17}$$

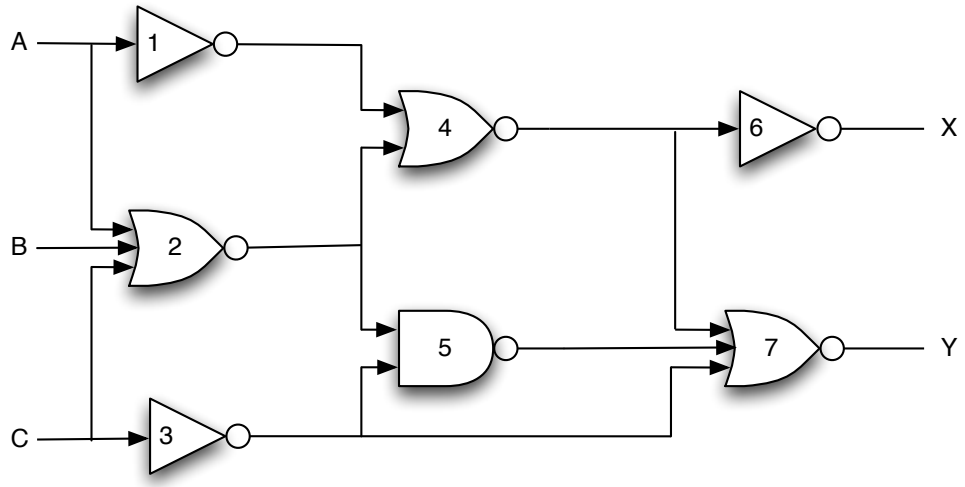


Figura 3.7: Circuito digital com 7 portas lógicas.

4 ESTUDO DE CASO DE COMPARAÇÃO DE CÉLULAS GERADAS AUTOMATICAMENTE PELO ASTRAN E *STANDARD CELLS* DE UMA BIBLIOTECA COMERCIAL

O intuito deste capítulo é avaliar qual é o atual sobrecusto em relação à área, atraso e potência introduzidas através da substituição de uma biblioteca de *standard cells* comercial, neste caso foi utilizada a biblioteca da AMS $350nm$ CMOS, por uma gerada automaticamente pela ferramenta ASTRAN para a tecnologia de $350nm$. A meta é avaliar somente a qualidade do leiaute, não todos os benefícios da metodologia de síntese livre de biblioteca (BUTZEN et al., 2007), (VALIDAÇÃO DE BIBLIOTECAS DE CÉLULAS PARA PROJETOS DE CIRCUITOS INTEGRADOS DIGITAIS, 2009), (REIS, 2008). Por esta razão, ambas, a biblioteca de células geradas automaticamente e a biblioteca *standard cell* comercial, possuem exatamente o mesmo conjunto de 14 células: ADD21, BUF2, CLKBU2, DF1, INV1, JK1, NAND20, NOR20, DFC1, XOR20, INV6, MUX21, AOI211 e OAI210, embora a biblioteca comercial seja constituída por um conjunto maior de células.

4.1 Metodologia de Comparação

A metodologia utilizada para fazer a comparação é mostrada na Figura 4.1. Inicia com a extração do leiaute das 14 células da biblioteca comercial de *standard cells* para um *netlist* SPICE. Este arquivo foi então usado como entrada para a ferramenta ASTRAN para que a mesma gerasse o leiaute equivalente, com exatamente o mesmo tamanho de transistores das *standard cells* para cada uma das células selecionadas. Depois disto, todas as células são importadas para o ambiente da Cadence para serem caracterizadas. A ferramenta Virtuoso foi usada para determinar a posição dos pinos, verificar as regras de projeto e extrair o leiaute com as capacitâncias parasitas (POSSER et al., 2010a), (POSSER et al., 2011).

A caracterização de células é feita usando a ferramenta Encounter Library Characterizer (CADENCE, 2009a) produzindo como saída um arquivo *Liberty* (.lib). O arquivo *Liberty* contém os valores de potência, atraso e ruído das células necessários para os processos de síntese lógica e física. Estes valores são calculados baseados nos resultados obtidos de simulações e por isso as células extraídas do subconjunto de *standard cells* foram re-caracterizadas usando a mesma metodologia utilizada para caracterizar as células geradas automaticamente, pois comparar células caracterizadas em diferentes pontos (inclinação da onda de entrada, capacitância de saída), não produziria uma comparação justa. Seguindo este fluxo, foi possível obter um arquivo *Liberty* para cada um dos métodos .

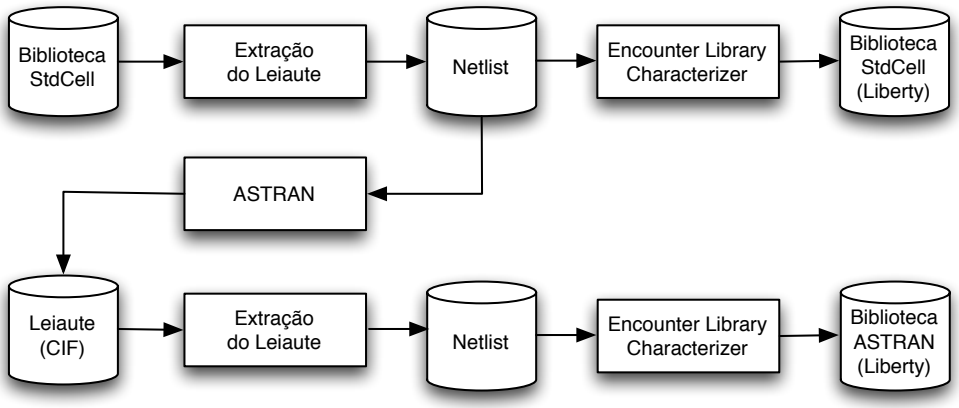


Figura 4.1: Metodologia utilizada para comparar a qualidade das células geradas automaticamente com as *standard cells* (POSSER et al., 2011).

4.2 Comparação Célula a Célula

Para ser capaz de avaliar a qualidade do leiaute gerado automaticamente, a potência, o *timing*, a área e a capacitância de entrada das células geradas automaticamente são comparados célula-a-célula com as células da biblioteca de *standard cell* utilizando os dados obtidos dos arquivos *Liberty* das duas metodologias. Considerando que as mesmas bibliotecas foram caracterizadas usando exatamente o mesmo arquivo de configuração e o mesmo *corner*, típico, onde considera-se a temperatura em 25^0C e a tensão de 3,3V, pode-se comparar diretamente do arquivo *Liberty* os valores de atraso e potência. A Tabela 4.1 mostra os resultados da comparação célula a célula entre *standard cells* e células geradas automaticamente. Os valores estão representados em porcentagem, onde valores positivos significam que as células geradas automaticamente possuem um resultado melhor que as *standard cells*. As diferenças de *timing* e potência não são significativas, já as diferenças nos valores de área e capacitância de entrada são significantes, sendo que as células geradas automaticamente possuem uma capacitância de entrada 13,44% menor, em média, e a área é 16,37% maior, em média.

A Tabela 4.1 indica que células maiores possuem uma capacitância de entrada menor, que é o caso das células geradas automaticamente. Isto deve-se ao fato que a biblioteca comercial de células foi projetada minimizando a área das suas células, reduzindo o espaço disponível para interconexões, consequentemente há mais linhas de metal 1 passando sobre as linhas de polisilício usado para criar o *gate* dos transistores, aumentando a capacitância de entrada do *gate*. Além disso, as *standard cells* fazem mais roteamento em polisilício, o que também gera maior capacitância de acoplamento. A Figura 4.2 ilustra algumas capacitâncias de acoplamento encontradas em um leiaute de *standard cell* comparada ao leiaute gerado automaticamente (POSSER et al., 2010b).

4.3 Efeitos no Mapeamento

Após ser observada a menor capacitância de entrada das células geradas automaticamente, decidiu-se descobrir quanto esse ganho impacta no mapeamento do circuito. Para isso, as duas bibliotecas de células, a *standard cells* (SC) e a gerada automaticamente (AG) foram mapeadas para oito circuitos *benchmarks* do ISCAS'89 (ISCAS'89, 2009)

Tabela 4.1: Comparação célula a célula utilizando os valores da caracterização.

	Timing	**Potência**	**Fuga**	**Transição**	**Área**	**Cap. Entrada**
	Ganho (%)	**Ganho (%)**	**Ganho (%)**	**Ganho (%)**	**Ganho (%)**	**Ganho (%)**
ADD21	2,86	5,88	0,02	0,93	-20	18,07
AOI211	1,59	7,08	0,43	1,34	-20	12,95
BUF2	0,35	1,11	-0,02	0,26	-25	-1,98
CLKBU2	2,48	3,27	-0,04	-2,23	-25	12,72
DFC1	4,22	1,01	-0,16	0,61	-10,52	15,11
DF1	9,76	-1,33	0,10	1,20	-11,76	54,85
INV6	-2,33	2,60	-42,68	-2,31	-40	-6,57
INV1	2,05	8,93	-0,42	1,10	0	42,39
JK1	1,94	-6,59	-0,04	0,54	-20,83	44,59
MUX21	1,60	3,54	-0,03	0,50	-14,28	19,34
NAND20	1,27	6,06	0,04	1,63	0	8,32
NOR20	-0,52	2,47	-0,19	0,77	-25	9,68
OAI210	-0,70	-1,26	0,29	-0,16	-20	15,67
XOR20	-1,85	-3,81	-0,19	-0,80	0	25,68
Média	**2,26**	**0,30**	**-2,20**	**0,23**	**-16,37**	**13,44**

utilizando a ferramenta de síntese lógica RTL Compiler (CADENCE, 2009b). A síntese lógica foi feita focando a melhoria no desempenho, sem incluir restrições de potência e área, utilizando exatamente o mesmo *script* para ambas as bibliotecas. A Tabela 4.2 apresenta os valores de potência (mW), *timing* (ns) e área(μm^2) para o conjunto de *standard cells* (SC) e de células geradas automaticamente (AG) dados pelo mapeamento dos circuitos, sem considerar as etapas de posicionamento e roteamento dos mesmos. Os valores de ganho (G) são mostrados em porcentagem, onde os valores negativos indicam que a biblioteca comercial de células apresenta um resultado melhor.

A biblioteca de células geradas automaticamente apresenta melhores valores de potência, economizando em média 24,44% e em *timing*, onde o projeto é em média 11,5% mais rápido. Este ganho é muito maior que o ganho observado na comparação célula-a-célula apresentado na Tabela 4.1. Isto deve-se ao fato que células geradas automaticamente possuem menor capacitância de entrada, permitindo que a célula possa suportar mais portas ligadas em sua saída, e, consequentemente, a ferramenta de síntese lógica reduz o número total de portas no circuito e no caminho crítico, melhorando dessa forma o *timing* e a potência, enquanto mantém os tempos de transição sob controle (POSSER et al., 2010b).

A única desvantagem dessa metodologia é o aumento em área, mas como a ideia não é utilizar a ferramenta de geração automática de células para fazer uma biblioteca e sim para inseri-la ao fluxo de projeto livre de biblioteca, o número de transistores será muito menor e, consequentemente, a área também será menor. Além disso, a velocidade do circuito será maior e, principalmente, a corrente de fuga será menor, uma questão importante nas tecnologias recentes. Todas essas vantagens são devido ao número menor de transistores (REIS, 2008).

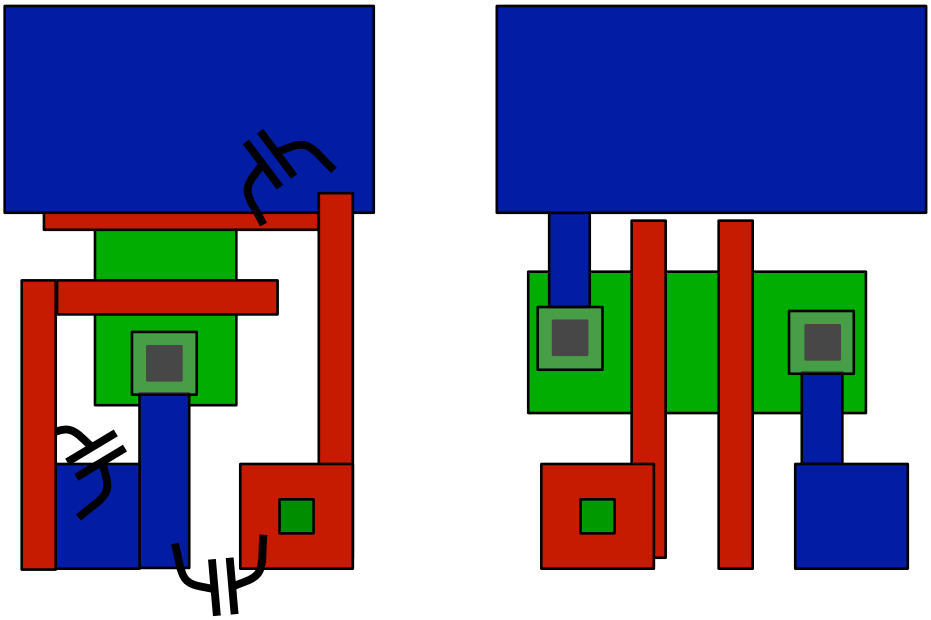


Figura 4.2: Exemplo mostrando maior capacitância de entrada na *standard cell* devido ao leiaute da célula ser mais denso (POSSER et al., 2010b).

4.4 Conclusão

A comparação célula a célula entre os leiautes gerados automaticamente e *standard-cells* mostrou que os resultados de *timing* e potência são comparáveis, com uma diferença menor de 2,3%, já os valores de capacitância de entrada e área das células resultou em uma diferença maior, onde as células geradas automaticamente possuem 13,44% menor capacitância de entrada, em média, enquanto as *standard-cells* possuem área 16,37% menor, em média (POSSER et al., 2010b), (POSSER et al., 2011).

Através do mapeamento dos dois conjuntos de células, geradas automaticamente e *standard-cells*, para os circuitos de *benchmark* pode-se observar que a capacitância de entrada menor apresentada pelas células geradas automaticamente produziu circuitos 11,5% mais rápidos, em média, e com um consumo de potência 24,4% menor, em média. O sobrecusto total de área introduzido também foi menor que a comparação célula a célula, pois a menor capacitância de entrada permitiu que a ferramenta de síntese lógica usasse um número total de células menor (POSSER et al., 2010b).

A principal contribuição desse estudo foi mostrar que as células geradas automaticamente pelo ASTRAN são capazes de apresentar a mesma qualidade global que as células projetadas manualmente, ou ainda melhor. A única desvantagem é o aumento na área da célula, mas, conforme foi mostrado, esse aumento na área da célula é o responsável por permitir que as células apresentem uma capacitância de entrada menor. Também deve ser notado que, desde que a área da célula é maior, é provável que menos espaços em branco sejam necessárias para resolver problemas de roteamento em áreas congestionadas (POSSER et al., 2010b).

Tabela 4.2: Comparação do mapeamento entre *standard cells* (SC) e células geradas automaticamente(AG) (POSSER et al., 2010b).

	Potência (mW)			**Timing (ns)**			**Área (μm^2)**		
	SC	**AG**	**G (%)**	**SC**	**AG**	**G (%)**	**SC**	**AG**	**G (%)**
s1196	29,4	23,3	26,0	2,144	2,146	-0,1	38929	40385	-3,6
s1238	29,2	23,5	24,1	2,31	2,012	14,8	37765	44226	-14,6
s15850	190,2	153,5	24,0	2,779	2,368	17,4	62517	72982	-14,3
s9234	208,9	167,7	24,6	3,538	2,971	19,1	91655	107543	-14,8
s35932	2527,7	2040	23,9	2,52	2,328	8,3	957627	1097041	-12,7
s38417	2322,2	1872	24,0	4,432	4,069	8,9	950604	1074728	-11,6
s38584	1707,6	1377	24,0	2,976	2,68	11,0	738101	853434	-13,5
s13207	466,4	373,2	25,0	2,434	2,16	12,7	161761	184821	-12,5
Média	**935,2**	**753,78**	**24,4**	**2,89**	**2,59**	**11,5**	**379870**	**434395**	**-12,2**

5 DESENVOLVIMENTO DO DIMENSIONADOR DE PORTAS USANDO PROGRAMAÇAO GEOMÉTRICA

Este capítulo apresenta o desenvolvimento do dimensionador de portas, onde o problema de dimensionamento é formulado como um programa geométrico e utiliza um resolvedor de programação geométrica para encontrar a solução ótima do problema.

5.1 Formulação

O dimensionador de portas foi desenvolvido da seguinte forma:

1. Cada instância que compõe o circuito possui um fator de escala associado a ela e tem seus transistores modelados como chaves. Dessa forma, a porta é vista como um conjunto de árvores RC, uma para cada possível vetor de entrada e o atraso da porta é o máximo atraso gerado pelas árvores que a compõe, conforme apresentado em (BOYD et al., 2005).

2. O atraso é calculado pelo modelo de Elmore, Seção 2.3, formando expressões posinomiais que são resolvidas por Programação Geométrica.

3. O atraso do circuito é a soma dos atrasos de cada porta que faz parte do caminho crítico.

4. A área é a soma da largura de cada transistor do circuito.

5. As variáveis de otimização do problema de programação geométrica gerado são os fatores de escala de cada instância do circuito, e elas afetam a área total, a potência consumida e a velocidade do circuito.

Os valores de capacitância e resistência para os cálculos de atraso, pelo modelo de Elmore, e de potência devem ser previamente conhecidos. Para isso, fez-se simulações SPICE para os transistores PMOS e NMOS, conforme é apresentado na Seção 5.2.

A potência é calculada considerando somente o chaveamento do circuito através da expressão:

$$P = (C_{load} + \sum_{i=1}^{n} Cin_i) Vdd^2 \alpha f \tag{5.1}$$

onde, C_{load} é a capacitância de carga do circuito, C_{in} é a capacitância de entrada de cada porta i do circuito, Vdd é o valor da tensão de alimentação, α é a probabilidade de chaveamento, que foi considerada, neste trabalho, como sendo 20% do tempo e f é a frequência do *clock* e, neste trabalho, foi utilizado 500MHz.

5.2 Valores Utilizados no Desenvolvimento do Dimensionador

Para o desenvolvimento do dimensionador de portas, alguns valores precisaram ser calculados e definidos para que as estimativas de atraso sejam as mais próximas possível da realidade.

Os cálculos foram feitos para as duas tecnologias utilizadas na análise dos resultados, $350nm$ e $45nm$. Sendo que a tecnologia de $350nm$ foi escolhida por ser a tecnologia utilizada na versão atual da ferramenta ASTRAN para fazer a geração automática do leiaute das células. E a tecnologia de $45nm$ é utilizada para verificar os resultados do dimensionamento aplicado a uma tecnologia submicrônica. Considerando que o dimensionador pode ser utilizado para outras tecnologias, somente alterando os parâmetros referentes a tecnologia.

Os seguintes valores foram calculados para cada tecnologia utilizando simulações SPICE e seus resultados são mostrados na Tabela 5.1:

- $CgateP$ e $CgateN$ = capacitância de entrada (*gate*) para os transistores PMOS e NMOS, respectivamente;
- $CsbP$ e $CsbN$ = valor da capacitância de dreno/fonte para substrato para os transistores PMOS e NMOS, respectivamente;
- $ReqP$ e $ReqN$ = resistência equivalente do transistor PMOS e NMOS, respectivamente.

Os valores mostrados na Tabela 5.1 são utilizados para resolver o problema de dimensionamento, e foram calculados considerando transistores com comprimento de $350nm$ para a tecnologia de $350nm$ e $50nm$ para a tecnologia de $45nm$ e com $1\mu m$ de largura para ambas as tecnologias, tanto para transistores NMOS quanto PMOS.

Tabela 5.1: Tabela dos valores calculados para resolver o problema de dimensionamento nas tecnologias $350nm$ e $45nm$ considerando um transistor com $1\mu m$ de largura.

Valores Calculados		
	350nm	**45**nm
CgateP (fF)	0,52219	0,7288
CgateN (fF)	1,3767	0,988656
CsbP (fF)	2,2087	0,717122
CsbN (fF)	2,4025	0,794589
ReqP (Ω)	20717	4948,83
ReqN (Ω)	8121	1402,74

Os métodos utilizado para calcular esses valores são descritos a seguir.

5.2.1 Cálculo de Capacitância de Entrada

Para calcular a capacitância de entrada construiu-se um pequeno circuito, conforme mostra a Figura 5.1, onde há uma fonte de alimentação ligada a uma resistência de 10KΩ. Essa resistência está ligada ao *gate* de um transistor NMOS, quando se quer encontrar a capacitância de entrada de um transistor NMOS (no caso da Figura 5.1) ou a um transistor PMOS, para calcular a capacitância do transistor PMOS. A fonte desse transistor está

ligada em GND (*ground*) e o dreno está ligado ao *gate* de outro transistor de tamanho quatro vezes maior que o anterior. Esse segundo transistor possui sua fonte ligada em GND e seu dreno fica em alta-impedância, ou seja, não está ligado a nada.

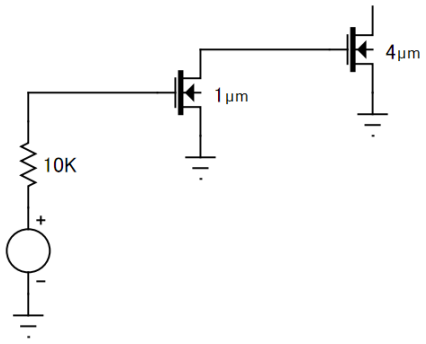


Figura 5.1: Circuito simulado em SPICE para calcular a capacitância de entrada de um transistor NMOS na tecnologia de 350nm.

No Apêndice A.5, é mostrado o arquivo SPICE de simulação usado para calcular o valor de capacitância de entrada para um transistor PMOS na tecnologia de 350nm. O cálculo para o transistor NMOS seguiu o mesmo formato, mudando apenas o tipo de transistor instanciado, que é NMOS ao invés de PMOS. O arquivo do modelo elétrico do transistor usado na simulação para 350nm é mostrado no Apêndice A.3.

5.2.1.1 Valores para 45nm

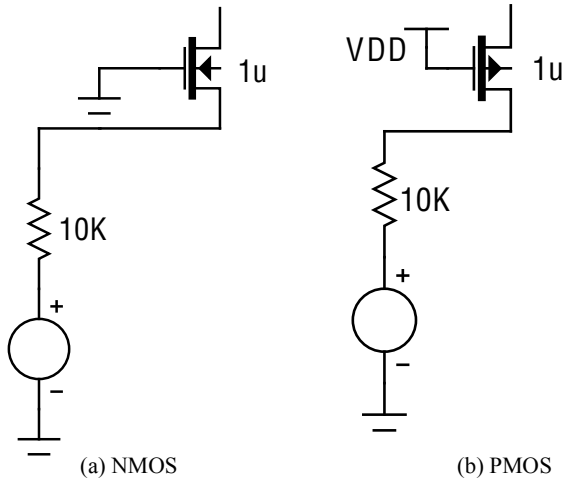
Para calcular os valores de capacitância de entrada para os transistores PMOS e NMOS para a tecnologia de 45nm, foi utilizada a mesma metodologia do cálculo para 350nm, mudando somente o arquivo de tecnologia, que é mostrado no Apêndice A.4, a resistência que era de 10KΩ passou para 1KΩ e o comprimento dos transistores que mudou de 350nm para 50nm.

O arquivo usado para a simulação do transistor NMOS para 45nm é mostrado no Apêndice A.6.

5.2.2 Cálculo da Capacitância de Fonte/Dreno para Substrato

O valor de capacitância de dreno para substrato e fonte para substrato é o mesmo, já que o transistor é um dispositivo simétrico, portanto é feita somente uma simulação para obter esses valores. Novamente é utilizada simulação SPICE, onde o circuito simulado é constituído de uma fonte de alimentação ligada em uma resistência de 10KΩ. Essa resistência está ligada ao dreno ou à fonte de um transistor PMOS, quando o transistor PMOS é simulado, ou NMOS, para simular um transistor NMOS. O *gate* deste transistor está ligado em GND no caso de um transistor NMOS, conforme mostra o circuito da Figura 5.2 (a), ou em VDD no caso de um transistor PMOS, Figura 5.2(b). A fonte ou dreno (dependendo de como foi chamado o anterior) desse transistor está em alta-impedância, ou seja, não está ligado em nada.

O arquivo usado para a simulação do transistor NMOS é mostrado no Apêndice A.7.



(a) NMOS (b) PMOS

Figura 5.2: Circuito simulado no SPICE para o cálculo da capacitância de dreno/fonte para substrato do transistor NMOS (a) e PMOS (b) na tecnologia de $350nm$.

5.2.2.1 Valores para 45nm

Os cálculos para a tecnologia de $45nm$ foram feitos usando o mesmo arquivo de simulação usado para $350nm$, com as mesmas mudanças de valores feitas para calcular a capacitância de entrada para $45nm$: modelo de transistor para $45nm$, a resistência ligada à fonte de alimentação é de 1KΩ e o comprimento dos transistores, conforme já mostrado. O arquivo SPICE usado para simular o transistor PMOS em $45nm$ pode ser visto no Apêndice A.8.

5.2.3 Cálculo da Resistência Equivalente

Para calcular a resistência equivalente de um transistor PMOS e de um transistor NMOS, foi utilizado como referência uma porta inversora, sendo que, quando o inversor está com o valor da saída alto, ou seja, igual a V_{dd}, obtém-se o valor da resistência equivalente do transistor PMOS e quando a saída do inversor está com o nível lógico baixo, ou seja, GND, obtém-se o valor da resistência equivalente do transistor NMOS.

Este cálculo, assim como os demais, foi feito utilizando simulação SPICE. O circuito utilizado na simulação possui uma fonte de alimentação ligada na entrada do inversor. A carga ligada à saída do inversor é equivalente a uma capacitância de $10fF$. Foi utilizada a mesma largura de transistor tanto para o NMOS como para o PMOS, como nos cálculos anteriores, ou seja, $1\mu m$. O comprimento do transistor também foi o mesmo, $350nm$ para a tecnologia de $350nm$. O arquivo de simulação para o transistor NMOS é mostrado no Apêndice A.9.

5.2.3.1 Valores para 45nm

A resistência equivalente calculada para $45nm$ seguiu a mesma formulação do cálculo para 350nm, alterando o modelo de transistor para $45nm$ e o comprimento dos transistores para $50nm$, a largura continua a mesma, 1 μm, tanto para o transistor PMOS quanto para o NMOS. O arquivo de simulação utilizado para calcular a resistência equivalente do transistor PMOS é mostrado no Apêndice A.10.

5.3 Desenvolvimento do Dimensionador de Portas

O dimensionador de portas lógicas foi desenvolvido baseado nos trabalhos (BOYD et al., 2005), (BOYD et al., 2007), (SAPATNEKAR et al., 1993), sendo que a novidade do mesmo está na junção dos trabalhos (BOYD et al., 2005) e (BOYD et al., 2007), onde os transistores de cada porta lógica do circuito são modelados usando o modelo de chaves RC apresentado por (BOYD et al., 2005), Seção 3.1.1.1, buscando uma modelagem mais precisa do atraso da porta. Enquanto que o dimensionamento das portas é modelado conforme (BOYD et al., 2007), Seção 3.2.3, usando um fator de escala para cada porta lógica do circuito, sendo que a largura dos transistores da porta é escalada de acordo com o fator de escala da porta que o mesmo pertence. Além disso, nosso trabalho faz a minimização da área do circuito sob uma restrição de atraso, conforme foi proposto por (SAPATNEKAR et al., 1993).

Os valores calculados na Seção 5.2 são utilizados para calcular atraso, área e potência na formulação do problema de programação geométrica que procura escolher de forma ótima o fator de escala de cada instância do circuito.

O dimensionador de portas, aqui implementado, gera um arquivo com a descrição do problema de programação geométrica, um exemplo desse arquivo é mostrado no Apêndice B. Esse arquivo deve conter todas as informações referentes ao problema de dimensionamento para que ele seja corretamente resolvido. A seguir, são detalhadas as etapas para a criação desse arquivo e para entender melhor como é o funcionamento da ferramenta, o circuito mostrado na Figura 5.3 será usado como exemplo nesta seção.

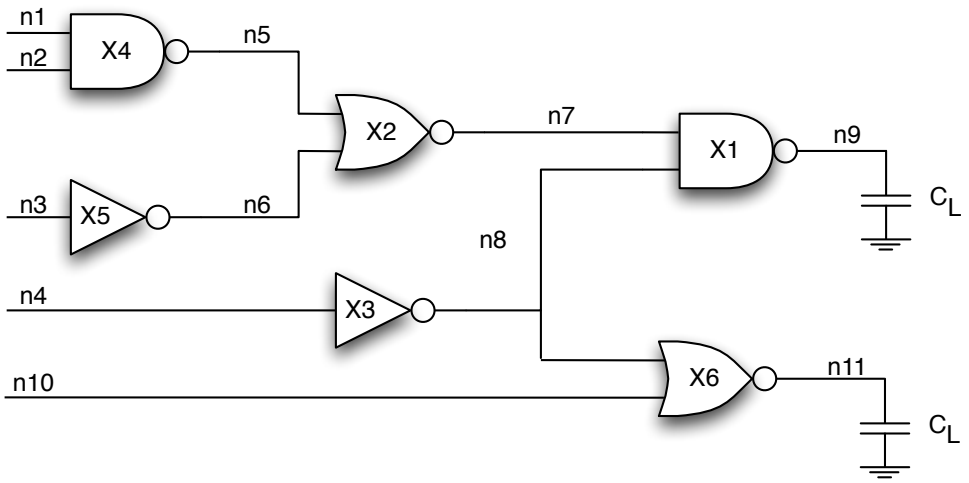


Figura 5.3: Circuito utilizado como exemplo para explicar o desenvolvimento do dimensionador de portas.

5.3.1 Definição das Constantes do Problema de Dimensionamento

Antes de definir as constantes que fazem parte do cálculo de dimensionamento de portas, são definidas as variáveis do problema, neste caso, são os fatores de escala de cada instância.

Segue uma lista de constantes que devem ser definidas, além dos valores mostrados na Tabela 5.1, para o cálculo do dimensionamento das portas. Sendo que, alguns valores são definidos pelo usuário (restrições do problema) e os demais valores são previamente

definidos pela ferramenta:

- $Cload$ = capacitância de carga ou de saída do circuito;
- $constrArea$ = restrição de área (quando está otimizando o atraso);
- $constrCin$ = restrição da capacitância de entrada do circuito, evitando que o circuito que estiver alimentado o circuito em questão tenha que alimentar uma carga muito elevada;
- $maxDelay$ = restrição de atraso, ou seja, atraso máximo que o circuito pode ter (quando está minimizando a área do circuito);
- $Xmax$ = fator de escala máximo que a porta pode ter;
- $Xmin$ = fator de escala mínimo da porta, neste caso é 1;
- Xp e Xn = tamanhos mínimos permitidos para os transistores PMOS e NMOS, respectivamente, conforme o limite de manufatura da tecnologia utilizada;
- Vdd = tensão de alimentação.

A Tabela 5.2 mostra os valores das constantes definidas para resolver o problema de dimensionamento que não são restrições do problema. Além dessas constantes, devem ser considerados os valores já definidos na Tabela 5.1.

Tabela 5.2: Tabela dos valores constantes utilizados para resolver o problema de dimensionamento nas tecnologias 350nm e 45nm.

Valores constantes		
	350nm	**45**nm
Xmax	32	32
Xmin	1	1
Xn (μm)	1	0,09
Xp (μm)	1,6	0,135
Vdd (V)	3,3	1,1

O tamanho de cada transistor dentro das células pode ser inicialmente dimensionado para os tamanhos mínimos Xp e Xn ou podem ser mantidos os valores definidos na descrição do circuito. Isso vai depender do mapeamento do circuito, pois, por exemplo, se o mapeamento foi feito livre de biblioteca utilizando a ferramenta ELIS, os transistores devem ser dimensionados para o mínimo inicialmente, pois o ELIS não se preocupa em dar valores coerentes para a largura de cada transistor, informando o mesmo valor para todos. Após os transistores terem seus valores definidos, as instâncias podem ser dimensionadas variando esses valores conforme seu fator de escala.

5.3.2 Cálculo da Capacitância de Entrada de cada Pino

O próximo passo é calcular, para cada instância que compõe o circuito, o valor de capacitância de seus pinos de entrada. Então, para cada pino de entrada, percorre os transistores ligados a este pino e soma-se, separadamente, as larguras dos transistor PMOS e NMOS. Após passar por todos os transistores, a soma da largura dos transistores PMOS é multiplicada pelo valor de capacitância de entrada, calculado na Seção 5.2, para o transistor PMOS. O mesmo é feito para a soma das larguras dos transistores NMOS. A capacitância de entrada do pino em questão é dada pela soma dessas duas multiplicações. Por exemplo, para um pino A que alimenta dois transistores, um transistor PMOS e outro NMOS, sendo que a largura do transistor PMOS é 1,6μm e do NMOS é 1μm a capacitância do pino é dada por $Cin(A) = (1,6 * CgateP) + (1 * CgateN)$.

Neste trabalho, o valor de capacitância calculado para a porta não escalada, conforme mostrado acima, é chamado de capacitância base e quando a capacitância base está multiplicando o fator de escala da porta, ela é chamada de capacitância final. Como o valor de capacitância é proporcional ao tamanho da porta, para calcular o valor de capacitância para uma instância escalada é necessário somente multiplicar esse valor base pelo fator de escala da instância.

5.3.3 Cálculo da Capacitância de Carga para cada Instância

O valor de capacitância de carga deve ser definido para cada instância que compõe o circuito. Esse cálculo é feito executando as seguintes etapas:

- Passa-se por cada instância do circuito e dentro da estrutura de cada instância, passa-se por cada pino de saída da mesma;

- Testa se o pino é uma saída primária, se for, a capacitância de carga do mesmo é igual ao valor de capacitância de carga definida para o circuito. No caso do circuito de exemplo, Figura 5.3, a capacitância de carga das células $X1$ e $X6$ é o valor definido como capacitância de carga do circuito.

- Se o pino não for uma saída primária, é feita a soma das capacitâncias de cada pino conectado a saída da instância e o valor do somatório é a capacitância de carga da porta. Por exemplo, a capacitância de carga para a porta $X3$ ($Cload(X3)$), Figura 5.3, é a soma da capacitância de entrada do pino B da instância $X1$ ($Cin(X1)(B)$) e a capacitância de entrada do pino A da instância $X6$ ($Cin(X6)(A)$), ou seja, $Cload(X3) = Cin(X1)(B) + Cin(X6)(A)$. Os valores de capacitância não são constantes, eles dependem do fator de escala da porta ao qual pertencem.

5.3.4 Modelagem RC para cada Instância

Após definir todos os valores necessários para o cálculo do dimensionamento, são calculados os valores de atraso, área e potência para cada instância que compõe o circuito criando seu modelo RC.

Inicialmente são definidos os seguintes valores, que são dependentes do fator de escala de cada instância:

- Largura base de cada transistor dentro de uma porta, a qual é multiplicada pelo fator de escala respectivo a cada instância de tal porta.

- A resistência equivalente de um transistor dimensionado pelo fator de escala k, dado por $R = (Runit * Wbase)/k$, onde $Runit$ é a resistência do transistor de tamanho unitário obtida por simulação SPICE e $Wbase$ é o tamanho base do transistor definido na topologia da porta.

- A capacitância de dreno e fonte do transistor dada por $C = Cunit * Wbase * k$, onde $Cunit$ é a capacitância de dreno/fonte do transistor de tamanho unitário obtida por simulação SPICE, $Wbase$ é o tamanho base do transistor definido na topologia da porta e k é o fator de escala.

- Área base da porta que é dada pela soma da largura de todos os transistores que compõem a porta em seu tamanho base, ou seja, não escalado. Por exemplo, a área base da instância $X1$ ($Abase(X1)$), é calculada da seguinte forma: $Abase(X1) = \sum_{i=1}^{n} W_i$, onde n é o número de transistores e W é a largura de cada transistor.

- Área final, que é o valor da área base multiplicado pelo fator de escala da porta lógica em questão. Por exemplo, a área final da porta $X1$ ($Afinal(X1)$), conforme a Figura 5.3, é dada por $Afinal(X1) = Abase(X1) * X1$, onde é utilizado como variável para o fator de escala o mesmo nome dado para a intância da porta, neste caso é $X1$.

A etapa seguinte é a modelagem da porta como um circuito RC, onde para cada transição de entrada obtém-se um circuito RC, conforme apresentado na Seção 3.1.1.1. Para cada circuito RC:

- Calcula-se a capacitância de cada nó que compõe a porta, que é dada pela soma das capacitâncias ligadas ao mesmo, ou seja, a capacitância dos transistores conectados ao nó por meio da fonte ou dreno e, se for o caso, a capacitância de carga da célula.

- É calculada a capacitância de *downstream*, que consiste na soma das capacitâncias do nó em questão e dos nós que estiverem no caminho para chegar à saída da porta. Por exemplo, se no caminho do nó Y até a saída da porta conter um nó Z, a capacitância de *downstream* do nó Y ($C(Y)$) será dada pela soma de sua capacitância, conforme calculado no item acima, com a capacitância do nó Z ($C(Z)$), resultando em $C(Y) = C(Y) + C(Z)$.

- É definida a expressão de atraso para a transição que está sendo analisada, seguindo o modelo de atraso de Elmore. A expressão é dada pelo produto RC do transistor que está conduzindo corrente, considerando a capacitância de *downstream* do nó que está em seu caminho para a saída. Por exemplo, o atraso da transição 0 para a instância $X1$ ($D(0)(X1)$), Figura 5.3, é dado pela multiplicação da resistência do transistor que está conduzindo ($Rtrans3(X1)$) e a capacitância de *downstream* do nó que está no caminho, neste caso é o nó 4 ($Cdown(4)(X1)$), resultando no seguinte cálculo $D(0)(X1) = Rtrans3(X1) * Cdown(4)(X1)$.

O atraso da instância é dado pelo máximo atraso entre os atrasos de cada transição. Por exemplo, se a instância $X1$, Figura 5.3, tiver três transições, 1, 2 e 3, ($D(0)(X1)$, $D(1)(X1)$, $D(2)(X1)$), o atraso da mesma é dado por $delay(X1) = max\{D(0)(X1)$, $D(1)(X1)$, $D(2)(X1)\}$.

5.3.5 Cálculo do Atraso do Circuito

Para calcular o atraso do circuito, ou seja, o maior atraso entre todos os caminhos possíveis, é percorrido cada instância do circuito. Inicialmente, define-se para as instâncias ligadas às entradas primárias que o atraso delas, por exemplo $D(X3)$ (atraso da instância $X3$), é o seu atraso somente ($delay(X3)$), ou seja, considerando o circuito de exemplo, Figura 5.3, ficaria:

- $D(X3) = delay(X3)$;
- $D(X5) = delay(X5)$;
- $D(X4) = delay(X4)$;

Depois disso, calcula-se o atraso das instâncias ligadas às instâncias primárias (portas ligadas às entradas primárias do circuito), que é dado pelo atraso dela mesma somado com o máximo atraso entre as instâncias que a alimentam, considerando o exemplo da Figura 5.3, o atraso da instância $X2$ é dado por:

- $D(X2) = delay(X2) + max(D(X5), D(X4))$;

O passo acima é repetido para todas as demais instâncias que não são primárias.

Concluindo o cálculo do atraso das instâncias do circuito de exemplo, faltou calcular o atraso das células ligadas a saída:

- $D(X1) = delay(X1) + max(D(X3), D(X2))$;
- $D(X6) = delay(X6) + D(X3)$;

O atraso do circuito ($Delay$) é o maior atraso entre os atrasos das instâncias que alimentam as saídas, ou seja, $Delay = max(D(X1), D(X6))$, para o exemplo da Figura 5.3.

5.3.6 Cálculo da Área e Potência do Circuito

O cálculo da área total ou final do circuito é dada pela soma da área final de cada instância, ou seja, a área base multiplicada pelo fator de escala da instância. Considerando que n denota o número de instâncias do circuito e $A_{final}(X_i)$ é a área final da instância i, o cálculo da área final do circuito é dado por:

$$A_{final} = \sum_{i=1}^{n} A_{final}(X_i) \tag{5.2}$$

O cálculo da área base do circuito é a soma da área base de cada instância i do circuito ($A_{base}(X_i)$), ou seja, é a soma das larguras dos transistores quando todos estão com seu tamanho base.

$$A_{base} = \sum_{i=1}^{n} A_{base}(X_i) \tag{5.3}$$

O cálculo de consumo de potência do circuito é efetuado seguindo a expressão 5.1 apresentada anteriormente.

5.3.7 Definição das Restrições do Problema de Dimensionamento

Após ter todos os valores necessários para resolver o problema de dimensionamento definidos em função dos fatores de escala do tamanho de cada instância, deve-se informar as restrições do problema, que são as seguintes:

- Área total do circuito, que deve ser menor ou igual que o valor definido como restrição, A_{max}. Essa restrição é usada quando a função objetivo é minimizar o atraso.
- O atraso máximo do circuito, que deve ser menor ou igual à restrição imposta, D_{max}. Sendo que esta restrição é usada para a minimização de área.
- O fator de escala de cada instância, sendo menor ou igual a um tamanho máximo, U_i, e maior ou igual a um;
- Capacitância de entrada do circuito, C_{in}, que deve ser menor ou igual a um determinado valor, C_{in}^{max}, evitando que a carga produzida pelo circuito em questão seja muito alta para o circuito que o está alimentado.

Quando o objetivo é minimizar o atraso, o problema de otimização usado neste trabalho é descrito da seguinte forma:

$$\begin{array}{ll} \text{minimizar} & Delay \\ \text{sujeito a} & 1 \leq X_i \leq U_i \\ & C_{in} \leq C_{in}^{max} \quad A \leq A^{max} \end{array} \tag{5.4}$$

Se a intenção for minimizar a área do circuito, muda apenas a função objetivo que passa a ser a área e o atraso passa a ser uma restrição, escrevendo o problema da seguinte forma:

$$\begin{array}{ll} \text{minimizar} & Area \\ \text{sujeito a} & 1 \leq X_i \leq U_i \\ & C_{in} \leq C_{in}^{max} \quad Delay \leq D^{max} \end{array} \tag{5.5}$$

5.4 Verificação do Funcionamento da Ferramenta

O modelo de Elmore é um limite superior para o atraso real (GUPTA et al., 1995). Desta forma, minimizar o Elmore não garante necessariamente redução do atraso computado por um simulador elétrico, como é o caso do HSPICE (SYNOPSYS, 2010a) e do SPECTRE (CADENCE, 2010). Nessa seção, são analisados os resultados de dimensionamento usando o simulador HSPICE, considerando a tecnologia de $350nm$, verificando o quão efetivo é o dimensinamento baseado em Elmore para a redução do atraso real de um circuito. Foi considerado o dimensionamento onde o objetivo é minimizar o atraso sob uma restrição de área.

Para a verificação, foi feito o dimensionamento dos transistores de algumas portas lógicas, mudando a formulação do dimensionador de portas, onde, ao invés de possuir uma variável para o fator de escala de cada porta lógica, foi considerada uma variável para o tamanho de cada transistor que compõe a porta. A porta é vista como um conjunto de árvores RC, uma para cada possível vetor de entrada e o atraso da porta é o máximo atraso gerado pelas árvores que a compõem. O valor de área é calculado pela soma da largura de cada transistor (POSSER et al., 2011)..

As portas lógicas dimensionadas para a verificação foram as seguintes:

- NAND de 2 entradas;
- NOR de 2 entradas;
- AOI (composta pelas portas lógicas AND, OR e inversor);
- INV (inversor);
- Porta complexa com 12 transistores.

Essas portas foram comparadas para outras quatro diferentes formas de dimensionamento:

- Utilizando o mesmo tamanho para todos os transistores que compõem a porta, onde é feita a divisão da área definida para a célula pelo número de transistores da mesma;
- Igualando o *drive strength* para a rede *pull-down* e *pull-up* baseado no *drive strength* de um inversor, conforme pode ser visto em (SUTHERLAND; SPROULL, 1991).
- Utilizando a *standard cell* correspondente de uma biblioteca de células comercial.
- Utilizando a razão P/N que iguala os atrasos de descida e subida. Para a tecnologia de $350nm$, tecnologia que foi utilizada na verificação, esse valor é 2,2486, ou seja, o transistor PMOS deve ser 2,2486 vezes maior que o transistor NMOS para que os atrasos de subida e descida do sinal sejam iguais.

Foi utilizada a mesma área para todos os métodos, considerando como área base a área da *standard cell* correspondente. Dessa forma, a área para o dimensionamento de transistores usando GP foi restringida para ser menor ou igual a área da *standard cell* equivalente. Após dimensionar os transistores de cada porta utilizando os métodos citados acima, as portas lógicas foram caracterizadas eletricamente através da ferramenta Encounter Library Characterizer (ELC) da Cadence usando o HSPICE como simulador elétrico para obter os resultados de atraso, potência, ruído.

Para a caracterização, as células foram descritas em SPICE, sem incluir os parasitas, e submetidas às simulações do caracterizador, considerando como valor de inclinação da onda de entrada, o menor valor apresentado na caracterização de uma biblioteca comercial de células para a tecnologia de $350nm$, neste caso foi $30ps$, pois o dimensionador implementado neste trabalho não considera os valores de inclinação da onda de entrada em sua formulação. O valor de carga de saída para o qual a célula foi caracterizada, é equivalente a quatro vezes a capacitância de entrada mínima ($4 * C_{in}$) da porta em questão. Por exemplo, a entrada A da porta em questão possui dois transistores ligados a ela, um transistor PMOS, onde a largura do mesmo é 1,6μm, e um transistor NMOS de 2μm de largura. Esses valores são multiplicados pelos seus respectivos valores de capacitância de entrada calculados por simulação SPICE, considerando o transistor PMOS ($CgateP$) e o NMOS ($CgateN$). O cálculo resultante é o seguinte $C_{load} = (1,6 * CgateP + 2 * CgateN) * 4$. Outro fator importante a ser definido para a caracterização das células é o *corner* de processo ao qual ela é submetida, neste caso foi escolhido o típico (*typical*) onde a temperatura das simulações é de 25^0C e a tensão de alimentação é de 3,3V.

Utilizando os valores de atraso e de potência do arquivo caracterizado, as portas dimensionadas pelos diferentes métodos foram comparadas considerando o pior caso de atraso, ou seja, a transição que possui o maior atraso. As comparações de cada porta são mostradas a seguir.

5.4.1 Comparação para uma Porta NAND

A Tabela 5.3 apresenta os valores de comparação para uma porta NAND de duas entradas, com área dos transistores equivalente a área da porta de tamanho 1 encontrada na biblioteca de células comercial e é utilizado esse mesmo valor de área para todos os métodos. Pode-se observar que o dimensionamento utilizando Programação Geométrica obteve menor atraso e menor consumo de potência em todos os casos, sendo que a menor redução foi dada pela comparação do mesmo com o dimensionamento utilizando a razão P/N. Neste caso, a redução em atraso foi de 0,6% e a redução em potência foi de 0,3%. Os resultados foram parecidos, neste caso, pois os tamanhos dos transistores informados pelos dois métodos de dimensionamento ficaram com valores muito próximos e como os parasitas não são considerados na análise, os valores de atraso e potência ficaram muito parecidos (POSSER et al., 2011).

Tabela 5.3: Valores de atraso e potência para a porta NAND de duas entradas comparada com outros métodos de dimensionamento para a tecnologia de $350nm$ (POSSER et al., 2011).

Porta NAND2				
	Atraso (ns)	**Redução (%)**	**Potência (pW)**	**Redução (%)**
Programação Geométrica	0,19927	-	0,15756	-
Transistores com mesmo tam.	0,21264	6,3	0,16503	4,5
Drive strength	0,21264	6,3	0,16503	4,5
Standard cell	0,24936	20,1	0,17429	9,6
Razão P/N	0,20047	0,6	0,15797	0,3

5.4.2 Comparação para uma Porta NOR

Na Tabela 5.4, podem ser vistos os resultados da comparação para a porta NOR de duas entradas dimensionada pelos diferentes métodos. Para esta comparação, o dimensionamento de transistores utilizando PG encontrou o menor atraso, assim como, o método *drive strength*, que obteve praticamente o mesmo valor de atraso. Além disso, o dimensionamento usando PG obteve um consumo de potência 0,3% menor que o dimensionamento considerando o *drive strength*. Pode-se notar que os resultados dos dimensionamentos usando PG e *drive strength* ficaram praticamente iguais, isso aconteceu porque o dimensionamento dos transistores fornecido pelos dois métodos ficou bastante semelhante. Para os demais métodos, o dimensionamento usando PG obteve reduções ainda maiores, tanto no atraso quanto no consumo de potência (POSSER et al., 2011).

5.4.3 Comparação para uma Porta AOI

Na Tabela 5.5, são mostrados os valores das comparações das diferentes formas de dimensionamento. Considerando que o objetivo da minimização é encontrar o menor atraso, o dimensionamento usando PG obteve o menor atraso comparado aos demais métodos, ou seja, o ponto com o menor atraso calculado por Elmore, resultou, também, no menor atraso calculado pela caracterização elétrica da porta lógica em questão. Em contrapartida, o dimensionamento usando PG resultou em maior consumo de potência comparado

Tabela 5.4: Valores de atraso e potência para a porta NOR de duas entradas comparada com outros métodos de dimensionamento para a tecnologia de $350nm$ (POSSER et al., 2011).

Porta NOR2				
	Atraso (ns)	**Redução (%)**	**Potência (pW)**	**Redução (%)**
Programação Geométrica	0,15297	-	0,10693	-
Transistores com mesmo tam.	0,27647	44,7	0,13507	20,8
Drive strength	0,15304	0,0	0,10729	0,3
Standard cell	0,16428	6,9	0,11096	3,6
Razão P/N	0,18622	17,9	0,11769	9,1

com os métodos *drive strength* e razão P/N, pois o objetivo dele era encontrar o menor atraso obedecendo a restrição de área, não considerando diretamente o consumo de potência. A menor redução em atraso foi de 4,8% comparado com a metodologia *drive strength*, sendo que o consumo de potência aumentou na mesma proporção (POSSER et al., 2011).

Tabela 5.5: Valores de atraso e potência para a porta AOI dimensionada usando PG e comparada com outros métodos de dimensionamento para a tecnologia de $350nm$ (POSSER et al., 2011).

Porta AOI				
	Atraso (ns)	**Redução (%)**	**Potência (pW)**	**Redução (%)**
Programação Geométrica	0,18623	-	0,21413	-
Transistores com mesmo tam.	0,33085	43,7	0,26306	18,6
Drive strength	0,19557	4,8	0,20438	-4,8
Standard cell	0,21936	15,1	0,11096	2,6
Razão P/N	0,20048	7,1	0,11769	-4,2

5.4.4 Comparação para um Inversor

Para a porta inversora, os valores de potência ficaram mais parecidos entre os vários métodos, conforme a Tabela 5.6 mostra. O dimensionamento utilizando PG obteve os mesmos tamanhos de transistores que o dimensionamento considerando a razão P/N, ou seja, os valores de atraso e potência foram iguais e, para os demais métodos, a redução em atraso foi desde 4,4% até 26,3%.

5.4.5 Comparação para uma Porta Complexa

A porta complexa de doze transistores usada para a comparação é mostrada na Figura 5.4, como esta não é uma porta de uma biblioteca de células, o seu dimensionamento não foi comparado considerando uma *standard cell*. Os valores de comparação obtidos entre os demais métodos são mostrado na Tabela 5.7. Pode ser observado que para todos

Tabela 5.6: Valores de atraso e potência para a porta inversora comparada com outros métodos de dimensionamento para a tecnologia de $350nm$.

Porta INV				
	Atraso (ns)	**Redução (%)**	**Potência (pW)**	**Redução (%)**
Programação Geométrica	0,06665	-	0,05546	-
Transistores com mesmo tam.	0,09040	26,3	0,05662	2,1
Drive strength	0,06972	4,4	0,05566	0,4
Standard cell	0,07577	12,0	0,05566	0,4
Razão P/N	0,06665	0,0	0,05546	0,0

os casos, o dimensionamento usando PG obteve um atraso menor, sendo, no mínimo, de 6,1%, atingindo o objetivo do dimensionador proposto, que é reduzir o atraso. Quanto ao consumo de potência, ele obteve um consumo maior que os outros métodos, representados pelos valores negativos da Tabela 5.4.

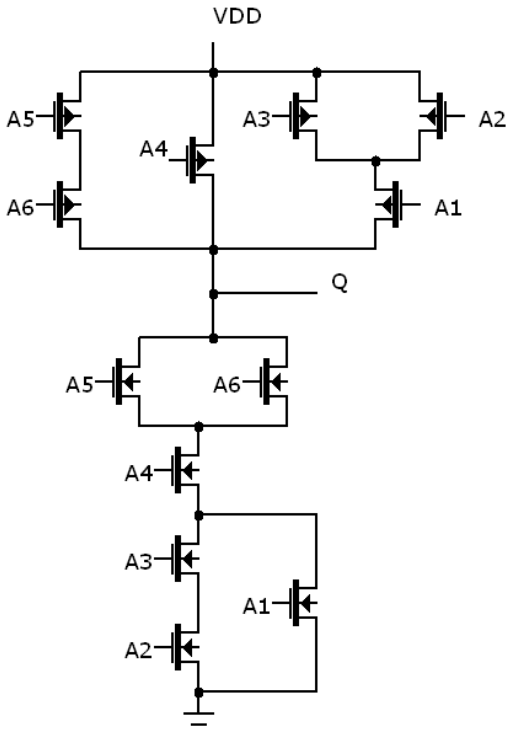


Figura 5.4: Esquemático elétrico da porta complexa utilizada para fazer o dimensionamento de transistores.

5.4.6 Conclusões

A verificação do funcionamento da ferramenta é feita dimensionando transistores ao invés de portas, mudando somente as variáveis de otimização que passam a ser uma variável para cada transistor do circuito, ao invés de uma variável para cada porta. Isto justifica a escolha de dimensionar portas em vez de transistores neste trabalho, pois o número de

Tabela 5.7: Valores de atraso e potência da porta complexa comparada com outros métodos de dimensionamento considerando a tecnologia de $350nm$.

Porta Complexa com 12 transistores				
	Atraso (ns)	**Redução (%)**	**Potência (pW)**	**Redução (%)**
Programação Geométrica	1,67653	-	0,93878	-
Transistores com mesmo tam.	1,92395	12,9	0,88714	-5,8
Drive strength	1,78605	6,1	0,92135	-1,9
Razão P/N	1,86939	10,3	0,78295	-19,9

variáveis torna-se significativamente menor, habilitando o dimensionamento de circuitos maiores em um tempo de execução menor.

O dimensionamento dos transistores das portas lógicas de exemplo utilizando programação geométrica, produziu resultados onde o atraso foi até 20% menor, Tabela 5.3, que o dimensionamento fornecido em uma *standard cell*, no caso da porta NAND. Além disso, comparado aos outros métodos de dimensionamento, a ferramenta desenvolvida neste trabalho obteve sempre o menor atraso ou o mesmo atraso, para os casos onde o dimensionamento obtido por programação geométrica era o mesmo dado por outro método. Com isso, a verificação mostrou que o dimensionamento implementado neste trabalho, usando o modelo de atraso de Elmore, é efetivo para a redução no atraso real do circuito.

Pode-se observar que o dimensionamento desenvolvido neste trabalho obteve menor atraso e consumo de potência comparado ao dimensionamento apresentado pelas *standard-cells* para todos os casos de teste. Isso acontece porque as *standard-cells* são células robustas, confeccionadas para a inserção em qualquer tipo de projeto, sendo, normalmente, superdimensionadas. E, o dimensionamento proposto neste trabalho é mais dedicado, pois fornece os tamanhos de transistores conforme as células precisam, considerando o contexto onde elas estão inseridas. Essa metodologia produz resultados melhores, mas é mais complexa e requer o uso de mais ferramentas para projetar um circuito.

5.5 Experimentos e Resultados do Dimensionamento de Portas

Para analisar o funcionamento da ferramenta de dimensionamento de portas, foram feitos testes usando circuitos de benchmark do ISCAS'85 (ISCAS'85, 2010).

Os circuitos foram dimensionados para as tecnologias $45nm$ e $350nm$, buscando minimizar o atraso do mesmo sob uma restrição de área. Após, os circuitos foram dimensionados com o objetivo de minimizar a área sob uma restrição de atraso. O valor da restrição de atraso é o mesmo dado pela minimização de atraso.

5.5.1 Resultados para 45nm Minimizando o Atraso

Os experimentos para $45nm$ para a minimização de atraso, iniciaram com o mapeamento dos circuitos usando a ferramenta RTL Compiler (CADENCE, 2009b) da Cadence para a biblioteca de células da NANGATE, considerando somente as células descritas em CMOS estático. Os circuitos mapeados foram inseridos na ferramenta de dimensionamento de portas, onde os valores de área, atraso e potência foram calculados. O valor de área obtido nesta etapa é usado como restrição para dimensionar o circuito pelo método

aqui proposto. Usando esta descrição e todas as instâncias em seu tamanho base, as instâncias que compõem o circuito foram dimensionadas usando a formulação da equação 5.4, ou seja, foram dimensionadas com o objetivo de minimizar o atraso, restrito a um valor de área.

Os valores obtidos pelas duas metodologias de dimensionamento, considerando os tamanhos encontrados em uma biblioteca de células (SC) e o dimensionamento usando programação geométrica (PG) desenvolvido neste trabalho, são mostrados na Tabela 5.8. O valor de capacitância de saída foi restrito para ser seis vezes maior que o valor de capacitância de entrada de uma porta não escalada e a capacitância de entrada foi restrita para ser um valor quatro vezes maior que a capacitância de entrada de uma porta não escalda, isto é, as instâncias conectadas às entradas primárias do circuito devem ser no máximo quatro vezes maior que a porta base com tamanho 1. Os valores de redução (R) na Tabela 5.8, mostram qual foi a redução em potência, atraso e área obtida pelo dimensionamento usando programação geométrica, sendo que os valores negativos significam que o dimensionamento disponível na biblioteca de células da NANGATE obteve um resultado melhor que o dimensionamento obtido pelo dimensionador implementado neste trabalho.

Tabela 5.8: Resultados da comparação entre o dimensionamento baseado nos tamanhos encontrados em uma biblioteca *standard cell* (SC) e o dimensionamento de portas usando Programação Geométrica (PG) proposto neste trabalho para os circuitos de benchmark do ISCAS'85

		Potência (μW)			**Atraso** (ps)			**Área** (μm^2)		
	# Portas	**SC**	**PG**	**R (%)**	**SC**	**PG**	**R (%)**	**SC**	**PG**	**R (%)**
C432	184	22,2	22,4	-0,9	718	666	7,3	210,4	210,4	0,0
C499	403	58,3	58,4	-0,2	750	651	13,1	536,4	536,4	0,0
C1908	259	33,6	33,7	-0,3	472	425	10,0	304,3	304,3	0,0
C880	232	31,4	31,1	1,1	451	330	26,8	281,0	277,4	1,3
apex1	1728	239,8	239,5	0,1	673	504	25,2	2304	2296	0,4
apex2	4110	527,1	523,6	0,7	863	650	24,7	5180	5145	0,7
apex3	1939	254,3	251,9	0,9	687	507	26,3	2441	2413	1,2
apex5	1942	264,6	258,3	2,4	662	431	34,9	2512	2446	2,6
Média	**1350**	**178,9**	**177,3**	**0,5**	**660**	**521**	**21,0**	**1721**	**1704**	**0,8**

Os circuitos dimensionados usando o método proposto por este trabalho apresentam resultados melhores que os circuitos mapeados para os tamanhos de células disponíveis em uma biblioteca de células. A redução no atraso foi em média de 21%, mantendo a mesma área e potência do dimensionamento baseado nos tamanhos encontrados em uma biblioteca *standard cell*.

Analisando a Tabela 5.8, pode ser observado que, a redução em atraso dada pelo dimensionamento de portas usando PG é maior para os circuitos com um número maior de instâncias. Isso acontece, porque um circuito com mais instâncias, provavelmente, possui um número maior de caminhos, o que dá maior liberdade ao método, permitindo que ele subdimensione os caminhos não críticos e concentre o seu esforço na redução do atraso dos caminhos mais críticos.

5.5.2 Resultados para 45nm Minimizando a Área

Os testes até então feitos foram com o objetivo de minimizar o atraso do circuito restringindo sua área. Nesta seção, serão apresentados alguns testes onde o objetivo é minimizar a área restringindo o atraso, considerando a tecnologia de 45nm. Para isso, deve ser conhecido um valor de atraso para ser determinado como restrição. Neste caso, foi utilizado o valor dado pela minimização de atraso mostrada na Tabela 5.8.

Na minimização de atraso, o objetivo do dimensionador é encontrar o ponto com menor atraso, obedecendo a restrição de área, contudo sem se preocupar com a minimização de área. Como existem vários pontos ótimos que minimizam o atraso, dada uma restrição de área, é provável que a solução do resolvedor não tenha atingido a área mínima. Sendo assim, é possível minimizar a área gerando um novo problema, onde se visa minimizar a área com restrição do atraso obtido na minimização de atraso.

Considerando essas observações, chega-se a conclusão que um bom método de fazer o dimensionando para obter mínimo atraso e área é minimizar primeiramente o atraso sob uma restrição de área e após minimizar a área sob a restrição de atraso encontrada. Isso foi observado também no trabalho (TENNAKOON; SECHEN, 2005), onde é mencionado que, uma vez que o limite de atraso mínimo é obtido a área pode ser minimizada para um dado atraso alvo.

A Tabela 5.9 mostra os valores obtidos para o dimensionamento de portas minimizando a área e restringindo o atraso ao valor mínimo encontrado pela minimização de atraso, mostrado na Tabela 5.8. Por isso, a Tabela 5.9 não mostra os valores de atraso, mostra somente os valores de área e potência dados pela minimização de atraso (min. Atraso) e pela minimização de área (Min. área) e a redução em potência e área obtida através da otimização de área.

Tabela 5.9: Resultados para o dimensionamento minimizando área restringindo o atraso para os circuitos de benchmark do ISCAS'85.

	Potência (μW)			**Área** (μm^2)		
	Min. Atraso	**Min. Área**	**Redução (%)**	**Min. Atraso**	**Min. Área**	**Redução (%)**
C432	22,4	22,4	0,00	210,4	210,4	0,00
C499	58,4	58,4	0,00	536,4	536,4	0,00
C1908	33,7	33,7	0,00	304,3	304,3	0,00
C880	31,1	20,2	34,9	277,4	171	38,4
apex1	239,5	137,3	42,7	2295,6	1293,5	43,7
apex2	523,6	270,3	48,4	5144,8	2647,3	48,5
apex3	251,9	135,8	46,1	2413	1274,1	47,2
apex5	258,3	138,5	46,4	2446,4	1269	48,1
Média	**177,3**	**102,1**	**27,3**	**1704**	**963,3**	**28,2**

Conforme pode ser visto na Tabela 5.9, a minimização de área possibilitou uma redução, em média, de 28,2% na área dos circuitos e em 27,3% na potência consumida, considerando o mesmo valor de atraso, comparado aos valores atingidos pela minimização de atraso, Tabela 5.8. Essa melhora é possível quando existem vários pontos ótimos que minimizam o atraso, dada uma restrição de área, possibilitando que a minimização de área encontre o ponto ótimo onde há a menor área considerando o atraso mínimo do

circuito. Pode ser observado também, que houve redução em área e potência em somente 5 dos oito circuitos considerados. Esses cinco, são os circuitos que a minimização de atraso obteve alguma redução em área, ou seja, o menor atraso foi encontrado para uma área menor que a restrição dada, conforme pode ser observado na Tabela 5.8,.

Na otimização de área, assim como na de atraso, pode ser observado que, há uma redução maior na área e potência para os circuitos com um número maior de instâncias, devido a maior liberdade proporcionada ao método quando o circuito possui mais caminhos, o que permite que subdimensione os caminhos não críticos e concentre maior esforço nos caminhos mais críticos.

5.5.3 Resultados para 350nm Minimizando o Atraso

O dimensionador de portas também foi testado para a tecnologia de 350nm considerando oito circuitos de benchmark do ISCAS'85, sendo os mesmos circuitos utilizados para o teste em 45nm. Os testes foram executados com o objetivo de minimizar o atraso dos circuitos sob a restrição de uma área máxima.

Inicialmente, os circuitos foram mapeados para uma biblioteca de células comercial na tecnologia de 350nm, onde foram consideradas somente as células CMOS da biblioteca. O mapeamento foi executado utilizando a ferramenta Design Compiler (SYNOPSYS, 2010c) da empresa Synopsys. Os circuitos mapeados foram inseridos na ferramenta de dimensionamento de células, onde os valores de área, atraso e potência foram calculados. O valor de área obtido nesta etapa é usado como restrição para dimensionar o circuito minimizando o atraso. Após, todas as instâncias do circuito foram descritas em seu tamanho base, ou seja, com fator de escala 1. Usando esta descrição, as instâncias que compõem o circuito foram dimensionadas, considerando que as instâncias conectadas às saídas do circuito possuem como capacitância de carga um valor seis vezes maior que o valor de capacitância de entrada de uma porta não escalada, neste caso, esse valor foi 13,27fF, equivalente a $Cload = (1,6 * CgateP + 1 * CgateN) * 6$, os valores de $CgateP$ e $CgateN$ para a tecnologia 350nm são mostrados na Tabela 5.1 e os valores 1,6 e 1 correspondem ao tamanho mínimo dos transistores PMOS e NMOS, respectivamente, considerados neste trabalho. A capacitância das instâncias conectadas às entradas do circuito foi restrita, neste caso, para ser quinze vezes maior que a capacitância de entrada de uma porta em seu tamanho base, isto é, as portas conectadas às entradas primárias do circuito devem ser no máximo quinze vezes maiores que a porta base com tamanho 1. A restrição de capacitância de entrada foi maior nesta análise do que na análise para 45nm, pois observou-se que o circuito mapeado para a biblioteca de células, possuía instâncias com tamanho até quinze vezes maior que o tamanho base ligadas às entradas primárias dos circuitos, então, para não gerar discrepância na comparação, a capacitância de entrada do circuito foi restrita ao esse valor.

A Tabela 5.10 mostra os valores de atraso, potência e área e seus respectivos valores de redução (R) obtidos pelos dois métodos de dimensionamento, o dimensionamento apresentado em uma biblioteca de células comercial (SC) e o dimensionamento usando programação geométrica (PG) desenvolvido neste trabalho. Considerando que os valores negativos apresentados para a redução (R) significam que o circuito dimensionado utilizando programação geométrica obteve um resultado pior.

Considerando a tecnologia de 350nm, os circuitos dimensionados usando o dimensionador de portas proposto neste trabalho apresentam uma redução, em média, de 4,5% no atraso, e os valores de área e potência são similares aos valores apresentados pelo circuito mapeado para a biblioteca de *standard cells*, conforme mostra a Tabela 5.10.

Tabela 5.10: Resultados da comparação entre o dimensionamento baseado nos tamanhos encontrados em uma biblioteca *standard cell* (SC) e o dimensionamento de portas usando Programação Geométrica (PG) proposto neste trabalho para os circuitos de benchmark do ISCAS'85 para $350nm$.

		Potência (mW)			**Atraso** (ns)			**Área** (μm^2)		
	# Portas	**SC**	**PG**	**R (%)**	**SC**	**PG**	**R (%)**	**SC**	**PG**	**R (%)**
C432	15	0,56	0,56	0,0	1,58	1,52	4,2	444	444	0,00
C499	388	7,99	8,07	-1,0	7,29	6,68	8,4	8015	8015	0,00
C1908	79	3,00	3,05	-1,5	3,26	3,09	5,2	2407,4	2407,4	0,00
C880	177	6,03	5,59	7,3	3,8	3,68	3,2	5609	5303	5,5
apex1	1455	27,9	27,9	0,0	7,28	7,11	2,4	27803	27783	0,1
apex2	778	15,2	15,2	0,1	6,59	6,25	5,2	15786	15765	0,1
apex3	1715	34,5	34,3	0,4	6,03	5,89	2,3	34965	34940	0,1
apex5	958	19,6	19,6	0,0	4,73	4,49	5,0	18699	18685	0,1
Média	**659,6**	**14,4**	**14,3**	**0,6**	**5,07**	**4,74**	**4,5**	**14216**	**14168**	**0,7**

Pode ser observado que a redução no atraso para a tecnologia de $350nm$ foi bem menor que a redução alcançada na tecnologia de $45nm$. Uma explicação para isso pode ser a biblioteca de células utilizada nas duas comparações, sendo que a biblioteca em $45nm$ é de código aberto, já a biblioteca em $350nm$ é comercial, além disso, a biblioteca em $45nm$ provavelmente deve considerar o atraso das interconexões, superdimensionando as células, já que é uma tecnologia submicrônica, onde os fios dominam o atraso do circuito (CONG, 1997). Outra diferença entre as duas comparações é a ferramenta utilizada para o mapeamento, sendo que, para $45nm$ foi utilizada a ferramenta RTL Compiler da Cadence e em $350nm$ foi utilizado o Design Compiler da Synopsys, que reduziu, em média, para mais da metade o número de instâncias utilizadas. Um outro motivo para dar essa diferença, que pode ser considerado o motivo mais provável, é que a tecnologia de $350nm$ não é submicrônica e o seu atraso depende mais das células do que das conexões, resultando em células com tamanhos mais "encorpados" para poderem carregar uma carga maior e suprir os tamanhos intermediários de células que não fazem parte da biblioteca, fazendo com que as otimizações de atraso sejam menores.

5.5.4 Resultados para 350nm Minimizando a Área

Da mesma forma que foi feita a otimização de área em $45nm$, ela também foi feita para $350nm$, onde o valor de atraso é a restrição do problema. Neste caso, o valor de atraso utilizado é o mesmo dado pela minimização de atraso mostrado na Tabela 5.10.

O dimensionamento minimizando área foi executado considerando como valor de capacitância de entrada o mesmo valor utilizado na minimização de atraso, quinze vezes a capacitância de entrada de uma porta com tamanho base, e o valor de capacitância de carga também foi considerado o mesmo valor utilizado na minimização de atraso, $13{,}27fF$.

Os valores obtidos para o dimensionamento de portas minimizando a área e restringindo o atraso ao valor mínimo encontrado pela minimização de atraso são mostrados na Tabela 5.11, sendo que o valor de atraso não é mostrado pois é o mesmo apresentado na Tabela 5.10. Sendo assim, a Tabela 5.11 apresenta os valores de área e potência dados

pela minimização de atraso (Min. atraso) e pela minimização de área (Min. área) e os valores de redução, em percentual, que puderam ser obtidos através da otimização de área. Pode ser observado, que a área e a potência puderam ser reduzidas somente para os cinco últimos circuitos apresentados na Tabela 5.11, que foram os circuitos que tiveram o menor atraso sem ocupar toda a área dada como restrição na minimização de atraso, como pode ser visto na Tabela 5.10.

Tabela 5.11: Resultados para o dimensionamento minimizando área restrindo o atraso para os circuitos de benchmark do ISCAS'85 considerando a tecnologia de $350nm$.

	Potência (mW)			**Área** (μm^2)		
	Min. Atraso	**Min. Área**	**Redução (%)**	**Min. Atraso**	**Min. Área**	**Redução (%)**
C432	0,56	0,56	0,00	444,0	444,0	0,00
C499	8,07	8,07	0,00	8015	8015	0,00
C1908	3,05	3,05	0,00	2407,4	2407,4	0,00
C880	5,59	3,11	44,4	5303,2	2710,2	48,9
apex1	27,89	14,79	47,0	27782,9	14282,4	48,6
apex2	15,22	8,208	46,1	15764,8	8419,4	46,6
apex3	34,35	16,89	50,8	34939,7	16659,3	52,3
apex5	19,56	11,82	39,6	18684,7	10748,3	42,5
Média	**14,3**	**8,3**	**28,5**	**14168**	**7960,7**	**29,9**

A Tabela 5.11 mostra que a minimização de área possibilitou uma redução, em média, de 29,9% na área dos circuitos e em 28,5% na potência consumida, considerando o mesmo valor de atraso dado pela minimização de atraso. Conforme já citado na minimização para $45nm$, essa melhora é possível quando há vários pontos ótimos que minimizam o atraso sob a restrição de área, possibilitando que a minimização de área encontre o ponto ótimo onde há a menor área para o atraso mínimo do circuito.

A ferramenta desenvolvida neste trabalho possibilita a realização das duas otimizações, de área e de atraso. Sendo possível, dessa forma, encontrar o ponto onde o atraso e a área convergem para um mínimo, obtendo um circuito dimensionado para o menor atraso e a menor área, conforme os exemplos mostrados na Tabela 5.11.

5.5.5 Considerações sobre o Dimensionador de Portas

Quanto à complexidade do problema de dimensionamento e do resolvedor de PG utilizado, não foram feitas análises a respeito, mas pode-se dizer que é polinomial, pois existe um algoritmo exato que resolve o problema em tempo praticável.

Quanto ao tempo de execução, ele pode crescer exponencialmente com o número de variáveis de otimizacão, assim como, com o número de entradas das portas ao calcular o atraso pelo modelo de chaves RC, mas, também, não tem-se dados exatos sobre isso.

Na versão atual da ferramenta, um circuito com 1715 portas lógicas levou cerca de cinquenta minutos para ser dimensionado, e um circuito de 1455 células levou trinta e quatro minutos. O número de equações e definições geradas para resolver o problema de dimensionamento foi de 66095 para o circuito com 1455 células e 80043 para o circuito com 1715 portas. Essas equações podem ser simplificadas, reduzindo significativamente esse número, tornando a execução mais rápida. Com a simplificação das equações, o cir-

cuito de 1455 células ficou com 12300 equações e o circuito de 1715 células gerou 14780 equações, reduzindo, em média, em 5,4 vezes o número de equações e consequentemente o tempo de execução.

Como limitações do dimensionador de portas desenvolvido neste trabalho tem-se:

- ele é somente para circuitos CMOS;
- o arquivo de entrada deve ser uma descrição em formato SPICE do circuito;
- usa um modelo simplificado para calcular o atraso;
- não considera as interconexões para calcular o atraso e potência do circuito.

6 CONCLUSÕES

Neste trabalho, foi feita uma revisão dos métodos mais relevantes que tratam do dimensionamento de portas e transistores. Entre os métodos apresentados, foi escolhido resolver o problema de dimensionamento de portas através da programação geométrica, pois conduz à solução ótima para o modelo dado. O resultado ótimo é dado por um balanceamento entre a área/potência e o atraso do circuito, onde, para cada porta lógica do circuito, deve ser encontrado seu melhor tamanho, ou seja, o tamanho que produz o menor atraso para alimentar a carga que está ligada a ela sem degradar o funcionamento da porta anterior que está alimentando-a.

O problema de programação geométrica utilizado neste trabalho para o dimensionamento de portas pode ser formulado de duas formas:

1. Com o objetivo de minimizar o atraso do circuito restrito a um valor máximo de área, ou

2. Minimizando a área sob uma restrição de atraso.

O dimensionador de portas desenvolvido é parametrizável para diferentes tecnologias de fabricação CMOS, bastando apenas mudar os parâmetros relativos à tecnologia. Além disso, ele pode ser manipulado para fazer o dimensionamento de transistores, onde as variáveis de otimização passam a ser o tamanho de cada transistor do circuito ao invés de ser o fator de escala de cada porta lógica, aumentando o número de variáveis a serem otimizadas e consequentemente o tempo de execução.

O dimensionador de portas teve seu funcionamento verificado através do dimensionamento de transistores de cinco portas lógicas para a tecnologia de $350nm$. Esse dimensionamento foi comparado com outros quatro métodos com o objetivo de minimizar o atraso sob uma restrição de área, sendo que a restrição de área é igual para todos os métodos. A verificação mostrou que o atraso obtido pelo dimensionamento desenvolvido neste trabalho sempre convergiu para o mínimo, ou seja, obteve um atraso menor ou igual (quando o dimensionamento informado era similar ao dimensionamento dado por outro método) aos outros métodos, atingindo o objetivo esperado, que era produzir o menor atraso.

Após o método ter sido validado, o dimensionador de portas foi testado para oito circuitos de *benchmark* do ISCAS' 85 considerando a tecnologia de $45nm$. Os testes mostraram que o dimensionamento otimizando o atraso através da PG reduziu o atraso dos circuitos em 21%, em média, comparado ao dimensionamento disponível em uma biblioteca de células, mantendo a mesma área e o mesmo consumo de potência. Após, os circuitos foram dimensionados buscando minimizar a área, onde o atraso foi restrito ao valor obtido na minimização de atraso. Através desta segunda otimização, pode-se obter uma redução em área de 28,2% e 27,3% em potência comparado aos valores dados pela

minimização de atraso. Dessa forma, utilizando as duas abordagens de otimização, uma após a outra, é possível obter o atraso mínimo e a área mínima para o circuito.

Após os testes em $45nm$, foram feitas as mesmas análises para a tecnologia de $350nm$, considerando os mesmos oito circuitos de *benchmark* utilizados para $45nm$. A área é restrita para ser a mesma utilizada pelo circuito mapeado para uma biblioteca de células comercial. Nesta tecnologia, o atraso pode ser reduzido em 4,5%, em média, comparado ao dimensionamento de portas fornecido por uma biblioteca de células, mantendo a área e o consumo de potência. A minimização de área, feita em seguida, restringindo o atraso ao valor dado pela minimização de atraso foi capaz de reduzir a área em 29,9%, em média, e a potência em 28,5%, em média. Pode-se observar que a minimização de atraso nesta tecnologia não foi tão significativa como em $45nm$, uma explicação para isso pode estar na tecnologia das bibliotecas de células utilizadas nas duas comparações, sendo que a tecnologia de $350nm$ não é submicrônica e o seu atraso depende mais das células do que das conexões, resultando em células com tamanhos mais "encorpados" para poderem carregar uma carga maior e suprir os tamanhos intermediários de células que não fazem parte da biblioteca, fazendo com que as otimizações de atraso sejam menores. Outra diferença entre as duas comparações é a ferramenta utilizada para o mapeamento, sendo que, para $45nm$ foi utilizada a ferramenta RTL Compiler da Cadence e em $350nm$ foi utilizado o Design Compiler da Synopsys, que reduziu, em média, mais da metade do número de instâncias utilizadas.

Os testes mostraram que o dimensionamento utilizando programação geométrica obteve resultados melhores que o dimensionamento apresentado em uma biblioteca de células. Além disso, o dimensionador de portas pode ser acoplado à ferramenta de geração automática de leiautes ASTRAN proporcionando algumas vantagens no projeto de circuitos:

- Possibilita realizar o fluxo de projeto livre de biblioteca, onde as células podem ser geradas com o tamanho desejado, de acordo com o dimensionamento fornecido pelo dimensionador de portas aqui proposto, onde é encontrado o tamanho ótimo para o modelo fornecido. Após a geração do leiaute, as células devem ser caracterizadas, possibilitando sua utilização em ambientes de síntese comerciais, permitindo fazer estimativas de atraso e potência considerando o leiaute do circuito.

- O circuito pode ser implementado utilizando portas complexas, reduzindo o número de transistores necessários para o mapeamento.

- Pode ser utilizado para verificar possíveis células que podem ser adicionadas em uma biblioteca e gerar automaticamente o leiaute dessas células, aumentando o desempenho do circuito.

- Gerar o leiaute do conjunto de células necessárias para projetar um determinado circuito, considerando o tamanho otimizado encontrado pela ferramenta de dimensionamento.

6.1 Trabalhos Futuros

Primeiramente deve-se investigar porque a redução em atraso foi muito maior para a tecnologia de $45nm$ do que para a tecnologia de $350nm$ no dimensionamento minimizando o atraso. Após obter essa resposta, pretende-se inserir os tempos de subida e descida dos sinais de entrada nas estimativas de atraso calculadas pelo modelo de Elmore.

Além disso, utilizar uma modelagem para o cálculo de potência mais precisa, considerando o consumo estático, permitindo a formulação do problema de dimensionamento minimizando a potência e restringindo o atraso e a área.

Outra abordagem pretendida é fazer o dimensionamento de portas após a etapa de posicionamento das células, onde o atraso das interconexões pode ser considerado, buscando uma forma de dimensionamento mais precisa. Em novas tecnologias o atraso depende muito das interconexões, por isso é importante que elas sejam consideradas no cálculo do dimensionamento, pois a carga que uma porta deve alimentar depende também da capacitância do fio que está conectado a ela.

Uma próxima etapa deste trabalho consiste na modelagem das variações de processo e dos problemas causados pelo envelhecimento do transistor junto ao problema de dimensionamento de portas, usando programação geométrica. Essa modelagem deve ser manuseada junto com as variáveis de otimização e de restrições do problema.

Por fim, outro trabalho futuro considerando o dimensionamento de portas consiste em concluir o fluxo de projeto livre de biblioteca e testar seu funcionamento, analisando o desempenho de circuitos produzidos por ele e pelo fluxo tradicional *standard-cell*, utilizando a ferramenta de dimensionamento de portas implementada neste trabalho para fazer o dimensionamento das portas. A ferramenta ASTRAN será utilizada para gerar o leiaute das células, que depois devem ser caracterizadas, possibilitando estimar atraso, potência e área do circuito após as etapas de posicionamento e roteamento.

REFERÊNCIAS

ALPERT, C. et al. Simultaneous driver sizing and buffer insertion using a delay penalty estimation technique Full. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, 2002., San Diego, California, USA. **Anais...** [S.l.: s.n.], 2002. p.104–109.

ANCEAU, F.; REIS, R. Complex Integrated Circuit Design Strategy. **IEEE Journal of Solid State Circuits**, New York, v.17, n.3, p.459–64, June 1982.

ASTRAN. Disponível em: <`http://www.inf.ufrgs.br/~amziesemerj/icpd/`>. Acesso em: jul, 2009.

BEECE, D. K. et al. Transistor Sizing of Custom High-Performance Digital Circuits with Parametric Yield Considerations. In: ACM IEEE DESIGN AUTOMATION CONFERENCE, DAC'2010, 47., Anaheim, California, USA. **Anais...** [S.l.: s.n.], 2010. p.781–786.

BEEFTINK, F. et al. Gate-Size Selection for Standard Cell Libraries. In: IEEE/ACM INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD 1998, 1998., San Jose, California, USA. **Anais...** [S.l.: s.n.], 1998. p.545–550.

BERKELAAR, M. R. C. M.; JESS, J. A. G. Gate Sizing in MOS Digital Circuits with Linear Programming. In: EDAC'90: CONFERENCE ON EUROPEAN DESIGN AUTOMATION, Glasgow, Scotland. **Anais...** [S.l.: s.n.], 1990. p.pp. 217–221.

BERKELEY, U. de. **MVSIS**: logic synthesis and verification. Disponível em: <http://embedded.eecs.berkeley.edu/Respep/Research/mvsis/>. Acesso em: ago, 2010.

BHATTACHARYA, K.; RANGANATHAN, N. A Linear Programming Formulation for Security-aware Gate Sizing. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, 18., Orlando, Florida - USA. **Anais...** [S.l.: s.n.], 2008. p.pp. 273–278.

BORAH, M.; OWENS, R.; IRWIN, M. Transistor Sizing for Low Power CMOS Circuits. **IEEE Transactions on Computer-Aided Design of Integrated circuits and Systems**, [S.l.], v.15, n.6, p.665–671, 1996.

BORAH, M.; OWENS, R. M.; IRWIN, M. J. Transistor Sizing for Minimizing Power Consumption of CMOS Circuits under Delay Constraint. In: INTERNATIONAL SYMPOSIUM ON LOW POWER DESIGN, 1995., Dana Point, California - USA. **Anais...** [S.l.: s.n.], 1995. p.pp. 167–172.

BOYD, S. et al. Digital Circuit Optimization via Geometric Programming. **Operations Research**, [S.l.], v.53, n.6, p.899–932, Nov.-Dec. 2005.

BOYD, S. et al. A Tutorial on Geometric Programming. **Springer Science+Business Media, LLC 2007**, USA, p.67–127, 2007.

BUTZEN, P. F. et al. Modeling and Estimating Leakage Current in Series-Parallel CMOS Networks. In: GLSVLSI '07: 17TH ACM GREAT LAKES SYMPOSIUM ON VLSI, Stresa-Lago Maggiore, Italy. **Anais...** [S.l.: s.n.], 2007. p.pp. 269–274.

CADENCE. **Encounter Library Characterizer - ELC**. Disponível em: <http://www.cadence.com>. Acesso em: ago, 2009.

CADENCE. **RTL Compiler**. Disponível em: <http://www.cadence.com>. Acesso em: ago, 2009.

CADENCE. **Virtuoso Spectre Circuit Simulator**. Disponível em: <http://www.cadence.com>. Acesso em: nov, 2010.

CCS Timing. Disponível em: <http://www.opensourceliberty.org/ccspaper/ccs_timing_wp.pdf>. Acesso em: abr, 2010, Technical White paper, Synopsys, Inc.

CHEN, C. P.; CHU, C. C. N.; WONG, D. F. Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, San Jose, Califónia, USA. **Anais...** [S.l.: s.n.], 1998. p.617–624.

CIRIT, M. A. Transistor Sizing in CMOS Circuits. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, 24., Miami Beach, Florida - USA. **Anais...** [S.l.: s.n.], 1987. p.pp. 121–124.

CONG, J. Challenges and Opportunities for Design Innovations in Nanometer Technologies. In: **Frontiers in Semiconductor Research**: a collection of src (semiconductor research corp.) working papers. [S.l.: s.n.], 1997.

CORREIA, V. P.; REIS, A. I. Advanced Technology Mapping for Standard-cell Generators. In: SBCCI 2002: 15TH SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, Porto de Galinhas Beach, Ipojuca, PE - Brazil. **Anais...** [S.l.: s.n.], 2002. p.pp. 254–259.

HüBNER, M.; BECKER, J. (Ed.). **Design Tools and Methods for Chip Physical Design**. [S.l.]: Springer Science, 2011. p.155–166.

DETJENS, E. et al. Technology Mapping in MIS. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD 1987, Los Alamitos, California. **Anais...** [S.l.: s.n.], 1987. p.116–119.

DUFFIN, R. J.; PETERSON, E. L.; ZENER, C. Geometric Programming-theory and Application. **John Wiley & Sons**, New York, USA, 1967.

EISNER, J. **Increasing Parametric Yield**. [S.l.]: Cadence Design Systems, Inc., White paper, 2003.

ELIS. **Environment for Logic Synthesis**. Disponível em: <http://www.inf.ufrgs.br/logics/>. Acesso em: mar, 2010.

ELMORE, W. The Transient Analysis of Damped Linear Networks with Particular Regard to Wideband Amplifiers. **J. Applied Physics**, [S.l.], v.19, 1948.

FERREIRA, F.; MORAES, F.; REIS, R. LASCA - Interconnect Parasitic Extraction Tool for Deep-Submicron IC Design. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN. **Anais...** [S.l.: s.n.], 2000. p.327–332.

FISHBURN, J. P.; DUNLOP, A. E. TILOS: a posynomial programming approach to transistor sizing. In: INT. CONFERENCE ON COMPUTER AIDED DESIGN, Las Vegas, Nevada - USA. **Anais...** [S.l.: s.n.], 1985. p.pp. 326–328.

FLACH, G.; HENTSCHKE, R.; REIS, R. Algorithms for improvement of RotDL router. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2004, Ijuí: Unijuí. **Anais...** [S.l.: s.n.], 2004.

FRAGOSO, J. **WTROPIC - um Gerador Automático de Macro Células CMOS Acessível via WWW**. 2001. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

GAVRILOV, S. et al. Library-less Synthesis for Static CMOS Combinational Logic Circuits. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD 1997, San Jose, California. **Anais...** [S.l.: s.n.], 1997. p.658–662.

GGPLAB: a matlab toolbox for gp. Disponível em: <`http://www.stanford.edu/boyd/ggplab/`>. Acesso em: mai, 2010.

GINNEKEN, L. P. van. Buffer Placement in Distributed RC-Tree Networks for Minimal Elmore Delay. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, New Orleans, LA , USA. **Anais...** [S.l.: s.n.], 1990. p.865 – 868.

GUAN, B.; SECHEN, C. Large Standard Cell Libraries and their Impact on Layout Area and Circuit Performance. In: IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN: VLSI IN COMPUTERS AND PROCESSORS, ICCD 1996, Austin, TX , USA. **Anais...** [S.l.: s.n.], 1996. p.378 – 383.

GUPTA, R. et al. The Elmore Delay as a Bound for RC Trees with Generalized Input Signals. In: ACM/IEEE DESIGN AUTOMATION CONFERENCE, DAC 1995, 32., San Francisco, California, USA. **Anais...** [S.l.: s.n.], 1995.

GUTHAUS, M. R. et al. Sizing Using Incremental Parameterized Statistical Timing Analysis. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, San Jose, California, USA. **Anais...** [S.l.: s.n.], 2005. p.1029–1036.

GÜNTZEL, J. L. **Geração de Circuitos Utilizando Matrizes de Células Prédifundidas**. 1993. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

GÜNTZEL, J. L. et al. Analysis of a Sea-of-Cells Assignment Too. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRôNICA, Campinas, São Paulo. **Anais...** [S.l.: s.n.], 1993. p.XII.13 – XII.15.

GÜNTZEL, J. L. et al. A Novel Approach for ASIC Layout Generation. In: MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, Rio de Janeiro. **Anais...** [S.l.: s.n.], 1995.

GÜNTZEL, J. L.; RIBAS, R.; REIS, R. MARCELA - Uma Nova Abordagem para Pré-difundidos. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRôNICA, Belo Horizonte. **Anais...** [S.l.: s.n.], 1991. p.534–543.

HARRIS, D. et al. **The Fanout-of-4 Inverter Delay Metric**. Unpublished manuscript. Disponível em: http://odin.ac.hmc.edu/ harris/research/FO4.pdf.

HENTSCHKE, R. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. 2002. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

HENTSCHKE, R. **Algorithms for Wire Length Improvement of VLSI Circuits with Concern to Critical Paths**. 2007. Tese (Doutorado em Ciência da Computação) — Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, RS - Brasil.

HINDI, H. A Tutorial on Convex Optimization. In: AMERICAN CONTROL CONFERENCE, Anaheim, California - USA. **Anais...** [S.l.: s.n.], 2004. v.4, p.pp. 3252–3265.

HOPPE, B. et al. Optimization of High-Speed CMOS Logic Circuits with Analytical Models for Signal Delay, Chip area and Dynamic Power Dissipation. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, USA, v.9, n.3, p.236–246, 1990.

HU, S.; KETKAR, M.; HU, J. Gate Sizing for Cell Library-Based Designs. In: ACM IEEE DESIGN AUTOMATION CONFERENCE, DAC 2007, 44., San Diego, California, USA. **Anais...** [S.l.: s.n.], 2007. p.847–852.

IIZUKA, T.; IKEDA, M.; ASADA, K. Exact Minimum-Width Transistor Placement without Dual Constraint for CMOS Cells. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, GLSVSLI, 15., Chicago, Illinois, USA. **Anais...** New York: ACM Press, 2005. p.74–77.

ISCAS'85. **Circuitos de Benchmark ISCAS'85**. Disponível em: <`http://courses.ece.illinois.edu/ece543/iscas85.html`>. Acesso em: abr, 2010.

ISCAS'89. **ISCAS'89 Benchmark Circuits**. Disponível em: <`http://courses.ece.illinois.edu/ece543/iscas89.html`>. Acesso em: jul, 2009.

JIANG, Z.; SHI, W. Circuit-wise Buffer Insertion and Gate Sizing Algorithm with Scalability. In: DESIGN AUTOMATION, 45., Anaheim, California, USA. **Anais...** [S.l.: s.n.], 2008. p.708–713.

JOHANN, M.; REIS, R. MARTE - MAze RouTer Environment. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE MICROELETRôNICA, Campinas, São Paulo. **Anais...** [S.l.: s.n.], 1993. p.XII.37 – XII.39.

JOSHI, S.; BOYD, S. An Efficient Method for Large-Scale Gate Sizing. **IEEE Transactions on Circuits and Systems**, [S.l.], v.55, n.9, p.2760 – 2773, October 2008.

LAZZARI, C. **Automatic Layout Generation of Static CMOS Circuits Targeting Delay and Power Reduction**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

LAZZARI, C. et al. A New-Macrocell Generation Strategy for Three Metal Layers CMOS Technologies. In: VLSI-SOC 20043, Darmstadt, Germany. **Anais...** [S.l.: s.n.], 2003. p.143–147.

LAZZARI, C.; ZIESEMER, A.; REIS, R. An Automated Design Methodology for Layout Generation Targeting Power Leakage Minimization. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, ICECS 2009, Tunisia. **Anais...** [S.l.: s.n.], 2009. p.81–84.

LPSOLVER. **Mixed Integer Linear Programming (MILP) Solver**. Disponível em: <http://lpsolve.sourceforge.net/5.5/>. Acesso em: fev, 2010.

LUBASZEWSKI, M. **Geração Automática de Lógica Aleatória Utilizando a Metodologia TRANCA**. 1990. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

MAHALINGAM, V.; RANGANATHAN, N. A Nonlinear Programming Based Power Optimization Methodology for Gate Sizing and Voltage Selection. In: ISVLSI 2005: IEEE COMPUTER SOCIETY ANNUAL SYMPOSIUM ON VLSI, Tampa, Florida - USA. **Anais...** [S.l.: s.n.], 2005. p.pp. 180–185.

MARQUES, F. S. et al. DAG Based Library-free Technology Mapping. In: ACM GREAT LAKES SYMPOSIUM ON VLSI, 17., Stresa-Lago Maggiore, Italy. **Anais...** [S.l.: s.n.], 2007. p.pp. 293–298.

MCMURCHIE, L.; EBELING, C. PathFinder: a negotiation-based performance-driven router for fpgas. In: ACM INTERNATIONAL SYMPOSIUM ON FIELD-PROGRAMMABLE GATE ARRAYS, FPGA, 3., Monterey, California, United States. **Anais...** New York: ACM Press, 1995. p.111–117.

MEINHARDT, C. **Geração de Leiautes Regulares Baseados em Matrizes de Células**. 2006. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

MORAES, F. **TRAGO - Síntese Automática de Leiaute para circuitos em Lógica Aleatória**. 1990. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

MORAES, F. **Synthèse Topologique de Macro-Cellules en Technologie CMOS**. 1994. Doutor em Ciência da Computação — Université Montpellier II, França.

MORAES, F.; REIS, R.; LIMA, F. An Efficient Layout Style for Three-Metal CMOS Macro-Cells. In: VLSI'97, Gramado. **Anais...** [S.l.: s.n.], 1997. p.415–426.

NANGATE. **Nangate Open Cell Library v1.0, FreePDK v1.3 Package**. Available: <http://www.nangate.com>. Acesso em: set, 2010.

NANGATE. **Nangate Library Creator**. Disponível em: <http://www.nangate.com>. Acesso em: mar, 2010.

NANGATE. **Nangate Library Characterizer**. Disponível em: <http://www.nangate.com>. Acesso em: set, 2010.

REIS, R.; GÜNTZEL, J.; JOHANN, M. **Physical Design Automation**. [S.l.]: Springer, 2006. p.83– 108.

POSSER, G. et al. Automatic Cell Layouts Generation Using the ASTRAN Tool. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2010, Porto Alegre, RS. **Anais...** [S.l.: s.n.], 2010. p.27–30.

POSSER, G. et al. A Study on Layout Quality of Automatic Generated Cells. In: IEEE INTERNATIONAL CONFERENCE ON ELECTRONICS, CIRCUITS AND SYSTEMS, ICECS 2010, Atenas, Grécia. **Anais...** [S.l.: s.n.], 2010.

POSSER, G. et al. Estudo da Qualidade do Leiaute de Células Geradas Automaticamente. In: IBERCHIP 2011, Bogotá, Colômbia. **Anais...** [S.l.: s.n.], 2011.

POSSER, G. et al. Gate Sizing using Geometric Programming. In: LASCAS 2011, Bogotá, Colômbia. **Anais...** [S.l.: s.n.], 2011.

PROLIFIC. **ProGenesys**. Disponível em: <http://www.prolificinc.com/products/progenesis.html>. Acesso em: mar, 2010.

RABAEY, J. M.; CHANDRAKASAN, A. P.; NIKOLIC, B. **Digital Integrated Circuits**. [S.l.]: Prentice-Hall, 2002.

REIS, A. et al. Library Free Technology Mapping. **VLSI: Integrated Systems on Silicon, Chapman-Hall**, [S.l.], p.303–314, 1997.

REIS, A. I. **Geração de Células Transparentes**. 1993. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

REIS, A. I.; REIS, R. TRAMOII - Proposta para um Gerador de Leiaute Baseado em Células para Circuitos CMOS Digitais com Dois Níveis de Metal. In: SIMPóSIO BRASILEIRO DE CONCEPçãO DE CIRCUITOS INTEGRADOS, Jaguariúna. **Anais...** [S.l.: s.n.], 1991. p.90–99.

REIS, R. A New Standard Cell CAD Methodology. In: IEEE CUSTOM INTEGRATED CIRCUITS CONFERENCE, Portland, Oregon. **Anais...** [S.l.: s.n.], 1987. p.385–388.

REIS, R. Physical Design Automation at Transistor Level. In: IEEE NORCHIP 2008 (INVITED TALK), Tallin. **Anais...** [S.l.: s.n.], 2008. p.5 p.

REIS, R.; GOMES, R.; LUBASZEWSKI, M. An Efficient Design Methodology for Standard Cell Circuits. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, Espoo. **Anais...** [S.l.: s.n.], 1988. p.1213–1216.

RIEPE, M. A.; SAKALLAH, K. A. Transistor Placement for Noncomplementary Digital VLSI Cell Synthesis. **ACM Trans. Des. Autom. Electron. Syst.**, New York, NY, USA, v.8, n.1, p.81–107, 2003.

RIERA, J. et al. Switch-Level Technology Mapping and Modeling. In: PATMOS 1997, Louvain-la-Neuve, Belgium. **Anais...** [S.l.: s.n.], 1997.

ROSA JR., L. S. et al. A Comparative Study of CMOS Gates with Minimum Transistor Stacks. In: SBCCI 2007: 20TH ANNUAL CONFERENCE ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, Rio de Janeiro, Brazil. **Anais...** [S.l.: s.n.], 2007. p.pp. 93–98.

SANTOS, C. et al. A Transistor Sizing Method Applied to an Automatic Layout Generation Tool. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI 2003, 16., São Paulo. **Anais...** [S.l.: s.n.], 2003. p.303–307.

SANTOS, C. et al. Delay/Area Optimization with Transistor Sizing. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2004, Ijuí: Unijuí. **Anais...** [S.l.: s.n.], 2004. p.80–84.

SANTOS, C. et al. Effects of Using a Pin-to-Pin Delay Model on a Library-Free Transistor/Gate Sizing Scheme. In: IEEE INTERNATIONAL MIDWEST SYMPOSIUM ON CIRCUITS AND SYSTEMS, MSCAS 2005, Covington, KY. **Anais...** [S.l.: s.n.], 2005/a. v.1, p.315–318.

SANTOS, C. et al. Incremental Timing Optimization for Automatic Layout Generation. In: IEEE INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS 2005, Kobe, Japan. **Anais...** [S.l.: s.n.], 2005/b. v.4, p.3567–3570.

SANTOS, C. L. dos. **Verificação e Otimização de Atraso durante a Síntese Física de Circuitos Integrados CMOS**. 2005. Dissertação (Mestrado em Ciência da Computação) — PPGC da UFRGS, Porto Alegre, RS.

SANTOS, G.; JOHANN, M.; REIS, R. Channel Based Routing in Channel-less Circuits. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS 2006. **Anais...** [S.l.: s.n.], 2006.

SAPATNEKAR, S. S. Wire Sizing as a Convex Optimization Problem: exploring the area-delay tradeoff. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.15, n.8, p.1001–1011, 1996.

SAPATNEKAR, S. S.; CHUANG, W. Power-delay Optimization in Gate Sizing. **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, New York, USA, v.5, n.1, p.98–114, 2000.

SAPATNEKAR, S. S. et al. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits Using Convex Optimization. **IEEE Transactions on Computer Aided Design of Integrated circuits and Systems**, [S.l.], v.12, n.11, p.1621–1634, 1993.

SAWICKI, S. et al. Studying the influence of I/O pads placement on wirelength and 3D-Vias of VLSI 3D integrated circuits. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2007, Porto Alegre. **Anais...** [S.l.: s.n.], 2007. p.89–92.

SINGH, J. et al. Robust Gate Sizing by Geometric Programming. In: IEEE/ACM DESIGN AUTOMATION CONFERENCE (DAC), 42., Anaheim, California - USA. **Anais...** [S.l.: s.n.], 2005. p.pp. 315–320.

SUTHERLAND, I. E.; SPROULL, R. F. Logical Effort: designing for speed on the back of an envelope. In: UNIVERSITY OF CALIFORNIA/SANTA CRUZ CONFERENCE

ON ADVANCED RESEARCH IN VLSI, 1991., Santa Cruz, California, USA. **Anais...** C. Sequin Ed. The MIT Press, 1991.

SUTHERLAND, I.; SPROULL, B.; HARRIS, D. **Logical Effort**: designing fast cmos circuits. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

SYNOPSYS. **HSPICE**. Disponível em: <http://www.synopsys.com/Tools/>. Acesso em: abr, 2010.

SYNOPSYS. **Liberty NCX**. Disponível em: <http://www.synopsys.com/Tools/>. Acesso em: set, 2010.

SYNOPSYS. **Design Compiler (DC)**. Disponível em: <http://www.synopsys.com/Tools/>. Acesso em: dez, 2010.

SYSTEMS, A. A. M. **c35 Cells**. Disponível em: <http://asic.austriamicrosystems.com>. Acesso em: ago, 2009.

TENNAKOON, H.; SECHEN, C. Efficient and Accurate Gate Sizing with Piecewise Convex Delay Models. In: ACM IEEE DESIGN AUTOMATION CONFERENCE, 42., Anaheim, California, USA. **Anais...** [S.l.: s.n.], 2005. p.807–812.

TOGNI, J. D. et al. Automatic Generation of Digital Cell Libraries. In: SBCCI 2004: 17TH SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, Porto Alegre, RS - Brazil. **Anais...** [S.l.: s.n.], 2002. p.pp. 265.

MAURICIO L. PILLA, J. C. B. de Mattos e Leomar S. daRosa Jr. e (Ed.). **VALIDAÇÃO DE BIBLIOTECAS DE CÉLULAS PARA PROJETOS DE CIRCUITOS INTEGRADOS DIGITAIS**. Pelotas,RS: Ed. Da Universidade Federal de Pelotas, 2009. p.141–155.

WESTE, N. H. E.; HARRIS, D. **CMOS VLSI Design**: a circuits and systems perspective. Boston, USA: Addison-Wesley Publishing Company, 2005.

WILKE, G. et al. Finding the Critical Delay of Combinational Blocks by Floating Vector Simulation and Path Tracing. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEM DESIGN, SBCCI, 15, Porto Alegre, RS. **Anais...** [S.l.: s.n.], 2002. p.277–282.

ZIESEMER, A. e. a. Cell Size Estimative in an Automatic Layout Generation Flow. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM 2006, Porto Alegre. **Anais...** [S.l.: s.n.], 2006. p.257–260.

ZIESEMER, A. M. **Geração Automática de Partes Operativas de Circuitos VLSI**. 2007. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre, RS.

APÊNDICE A ARQUIVOS SPICE

A.1 Arquivo de subcircuitos descritos em SPICE na tecnologia de 350nm para serem caracterizados eletricamente pela ferramenta ELC

```
.SUBCKT NAND2X1 A B Q VDD GND
M1 Q A VDD VDD  MODP  L=350E-9 W=1.9160E-6
M2 Q B VDD VDD  MODP  L=350E-9 W=1.9033E-6
M3 GND A 1 GND  MODN  L=350E-9 W=1.6992E-6
M4 Q   B 1 GND  MODN  L=350E-9 W=1.6815E-6
.ENDS NAND2X1

.SUBCKT NANDDS2X1 A B Q VDD GND
M1 Q A VDD VDD  MODP  L=350E-9 W=1.8E-6
M2 Q B VDD VDD  MODP  L=350E-9 W=1.8E-6
M3 GND A 1 GND  MODN  L=350E-9 W=1.8E-6
M4 Q   B 1 GND  MODN  L=350E-9 W=1.8E-6
.ENDS NANDDS2X1

.SUBCKT NANDMIN2X1 A B Q VDD GND
M1 Q A VDD VDD  MODP  L=350E-9 W=1.8E-6
M2 Q B VDD VDD  MODP  L=350E-9 W=1.8E-6
M3 GND A 1 GND  MODN  L=350E-9 W=1.8E-6
M4 Q   B 1 GND  MODN  L=350E-9 W=1.8E-6
.ENDS NANDMIN2X1

.SUBCKT NANDSC2X1 A B Q VDD GND
M1 Q A VDD VDD  MODP  L=350E-9 W=1.6E-6
M2 Q B VDD VDD  MODP  L=350E-9 W=1.6E-6
M3 GND A 1 GND  MODN  L=350E-9 W=2E-6
M4 Q   B 1 GND  MODN  L=350E-9 W=2E-6
.ENDS NANDSC2X1

.SUBCKT NANDRAT2X1 A B Q VDD GND
M1 Q A VDD VDD  MODP  L=350E-9 W=1.905E-6
M2 Q B VDD VDD  MODP  L=350E-9 W=1.905E-6
M3 GND A 1 GND  MODN  L=350E-9 W=1.694E-6
M4 Q   B 1 GND  MODN  L=350E-9 W=1.694E-6
.ENDS NANDRAT2X1
```

A.2 Arquivo de *setup* utilizado na caracterização de células para a tecnologia de 45nm utilizando a ferramenta ELC

```
//Setup file for simulation in 45$nm$

Process typical  {
        voltage = 1.1; // as voltage
        temp = 25 ; // as temperature
        Vtn = 0.15 ; // nmos Vt
        Vtp = 0.15 ; // pmos Vt
};

Signal std _cell  {
        unit = REL ; // relative value
        Vh = 1.0 1.0 ; // 100% rise/fall
        Vl = 0.0 0.0 ;
        Vth = 0.5 0.5 ; // 50% rise/fall
        Vsh = 0.8 0.8 ;
        Vsl = 0.2 0.2 ;
        tsmax = 2.0n ; // maximum output slew rate
};

Simulation std _cell  {
        transient = 0.2n 80n 10p ;
        bisec = 4.0n 4.0n 10ps ; // binary search
        resistance = 10K;
};
```

```
Index NAND2 _X1 _XG3  {
        slew = 0.007500N 0.018750N 0.037500N 0.075000N 0.150000N 0.30000N 0.6000N ;
        load = 0.000400P 0.000800P 0.001600P 0.003200P 0.006400P 0.01280P 0.0256P ;
} ;

Index INV _X1 _XG32  {
        slew = 0.007500N 0.018750N 0.037500N 0.075000N 0.150000N 0.30000N 0.6000N ;
        load = 0.000400P 0.000800P 0.001600P 0.003200P 0.006400P 0.01280P 0.0256P ;
} ;

Index NOR2 _X1 _XG45  {
        slew = 0.007500N 0.018750N 0.037500N 0.075000N 0.150000N 0.30000N 0.6000N ;
        load = 0.000400P 0.000800P 0.001600P 0.003200P 0.006400P 0.01280P 0.0256P ;
} ;

Index NAND4 _X1 _XG48  {
        slew = 0.007500N 0.018750N 0.037500N 0.075000N 0.150000N 0.30000N 0.6000N ;
        load = 0.000400P 0.000800P 0.001600P 0.003200P 0.006400P 0.01280P 0.0256P ;
} ;

Index NAND2 _X1 _XG49  {
        slew = 0.007500N 0.018750N 0.037500N 0.075000N 0.150000N 0.30000N 0.6000N ;
        load = 0.000400P 0.000800P 0.001600P 0.003200P 0.006400P 0.01280P 0.0256P ;
} ;

Group INV _X1  {
        CELL = *INV _X1 ;
} ;

Group NAND2 _X1 _XG3  {
        CELL = *NAND2 _X1 _XG3 ;
} ;

Group INV _X1 _XG32  {
        CELL = *INV _X1 _XG32 ;
} ;

Group NOR2 _X1 _XG45  {
        CELL = *NOR2 _X1 _XG45 ;
} ;

Group NAND4 _X1 _XG48  {
        CELL = *NAND4 _X1 _XG48 ;
} ;

Group NAND2 _X1 _XG49  {
        CELL = *NAND2 _X1 _XG49 ;
} ;

Margin m0  {
        setup = 1.0 0.0 ;
        hold = 1.0 0.0 ;
        release = 1.0 0.0 ;
        removal = 1.0 0.0 ;
        recovery = 1.0 0.0 ;
        width = 1.0 0.0 ;
        delay = 1.0 0.0 ;
        power = 1.0 0.0 ;
        cap = 1.0 0.0 ;
} ;

Nominal n0  {
        cap     = 0.0:0.5:1.0 0.0:0.5:1.0 ;  // rise / fall
        check   = 1.0:1.0:1.0 1.0:1.0:1.0 ;  // rise / fall
        current = 0.0:0.5:1.0 0.0:0.5:1.0 ;  // rise / fall
        power   = 0.0:0.5:1.0 0.0:0.5:1.0 ;  // rise / fall
        slew    = 0.0:0.5:1.0 0.0:0.5:1.0 ;  // rise / fall
        delay   = 0.0:0.5:1.0 0.0:0.5:1.0 0.0:0.5:1.0;  // rise / fall / Z
        delay = 0.5 0.5 ; // as rise fall
        power = 0.5 0.5 ;
        cap = 0.5 0.5 ;
} ;

set process (best,typical,worst)  {
        simulation = std _cell  ;
        index = X1 ;
        signal = std _cell ;
        margin = m0 ;
        nominal = n0 ;
} ;

set index (best,typical,worst)  {
Group (INV _X1) = INV _X1 ;
Group (NAND2 _X1 _XG3) = NAND2 _X1 _XG3 ;
Group (INV _X1 _XG32) = INV _X1 _XG32 ;
Group (NOR2 _X1 _XG45) = NOR2 _X1 _XG45 ;
Group (NAND4 _X1 _XG48) = NAND4 _X1 _XG48 ;
Group (NAND2 _X1 _XG49) = NAND2 _X1 _XG49 ;
} ;
```

A.3 Arquivo do modelo elétrico de transistor para a tecnologia 350nm utilizado nas simulações SPICE

```
*================= Modelo de transistor - 350nm ==================
.MODEL MODN NMOS LEVEL=49
```

```
+MOBMOD =1.000e+00 CAPMOD =2.000e+00
+NOIMOD =3.000e+00
+VERSION=3.11
+K1     =5.0296e-01
+K2     =3.3985e-02 K3     =-1.136e+00 K3B    =-4.399e-01
+NCH    =2.611e+17 VTH0   =4.979e-01
+VOFF   =-8.925e-02 DVT0   =5.000e+01 DVT1   =1.039e+00
+DVT2   =-8.375e-03 KETA   =2.032e-02
+PSCBE1 =3.518e+08 PSCBE2 =7.491e-05
+DVT0W  =1.089e-01 DVT1W  =6.671e+04 DVT2W  =-1.352e-02
+UA     =4.705e-12 UB     =2.137e-18 UC     =1.000e-20
+U0     =4.758e+02
+DSUB   =5.000e-01 ETA0   =1.415e-02 ETAB   =-1.221e-01
+NFACTOR=4.136e-01
+EM     =4.100e+07 PCLM   =6.948e-01
+PDIBLC1=3.571e-01 PDIBLC2=2.065e-03 DROUT  =5.000e-01
+A0     =2.541e+00 A1     =0.000e+00 A2     =1.000e+00
+PVAG   =0.000e+00 VSAT   =1.338e+05 AGS    =2.408e-01
+B0     =4.301e-09 B1     =0.000e+00 DELTA  =1.442e-02
+PDIBLCB=3.222e-01
+W0     =2.673e-07 DLC    =3.0000e-08
+DWC    =9.403e-08 DWB    =0.000e+00 DWG    =0.000e+00
+LL     =0.000e+00 LW     =0.000e+00 LWL    =0.000e+00
+LLN    =1.000e+00 LWN    =1.000e+00 WL     =0.000e+00
+WW     =-1.297e-14 WWL    =-9.411e-21 WLN    =1.000e+00
+WWN    =1.000e+00
+TNOM   =27.0 AT     =3.300e+04 UTE    =-1.800e+00
+KT1    =-3.302e-01 KT2    =2.200e-02 KT1L   =0.000e+00
+UA1    =0.000e+00 UB1    =0.000e+00 UC1    =0.000e+00
+PRT    =0.000e+00
+CGDO   =1.300e-10 CGSO   =1.200e-10 CGBO   =1.100e-10
+CGDL   =1.310e-10 CGSL   =1.310e-10 CKAPPA =6.000e-01
+CF     =0.000e+00 ELM    =5.000e+00
+XPART  =1.000e+00 CLC    =1.000e-15 CLE    =6.000e-01
+RDSW   =3.449e+02
+CDSC   =0.000e+00 CDSCB  =1.500e-03 CDSCD  =1.000e-03
+PRWB   =-2.416e-01 PRWG   =0.000e+00 CIT    =4.441e-04
+TOX    =7.575e-09 NGATE  =0.000e+00
+NLX    =1.888e-07
+XL     =0.000e+00 XW     =0.000e+00
+ALPHA0 =0.000e+00 BETA0  =3.000e+01
+AF     =1.3600e+00 KF     =5.1e-27 EF     =1.000e+00
+NOIA   =1.73e+19 NOIB   =7.000e+04 NOIC   =-5.64e-13
+ACM    =2
+RD     =0.000e+00 RS     =0.000e+00 RSH    =7.000e+01
+RDC    =0.000e+00 RSC    =0.000e+00
+LINT   =-5.005e-08  WINT   =9.403e-08
+LDIF   =0.000e+00 HDIF   =8.000e-07 WMLT   =1.000e+00
+LMLT   =1.000e+00 XJ     =3.000e-07
+JS     =1.000e-05 JSW    =0.000e+00 IS     =0.000e+00
+N      =1.000e+00 NDS    =1000.
+VNDS   =-1.000e+00 CBD    =0.000e+00 CBS    =0.000e+00 CJ     =9.400e-04 CJSW   =2.500e-10
+FC     =0.000e+00 MJ     =3.400e-01 MJSW   =2.300e-01 TT     =0.000e+00
+PB     =6.900e-01 PHP    =6.900e-01

.MODEL NWDINSUB D LEVEL=1
+IS     =6.0000e-05 JSW    =2.7000e-10 N      =1.000e+00
+CJ     =8.0000e-05 M      =3.9000e-01 VJ     =5.3000e-01 TT     =0.000e+00
+CJSW   =5.1000e-10 MJSW   =2.7000e-01 FC     =0.500e+00
+EG     =1.110e+00 XTI    =3.000e+00 AF     =1.000e+00 KF     =0.000e+00

.MODEL MODP PMOS LEVEL=49
+MOBMOD =1.000e+00 CAPMOD =2.000e+00
+NOIMOD =3.000e+00
+VERSION=3.11
+K1     =5.9959e-01
+K2     =-6.038e-02 K3     =1.103e+01 K3B    =-7.580e-01
+NCH    =9.240e+16 VTH0   =-6.915e-01
+VOFF   =-1.170e-01 DVT0   =1.650e+00 DVT1   =3.868e-01
+DVT2   =1.659e-02 KETA   =-1.440e-02
+PSCBE1 =5.000e+09 PSCBE2 =1.000e-04
+DVT0W  =1.879e-01 DVT1W  =7.335e+04 DVT2W  =-6.312e-03
+UA     =5.394e-10 UB     =1.053e-18 UC     =1.000e-20
+U0     =1.482e+02
+DSUB   =5.000e-01 ETA0   =2.480e-01 ETAB   =-3.917e-03
+NFACTOR=1.214e+00
+EM     =4.100e+07 PCLM   =3.184e+00
+PDIBLC1=1.000e-04 PDIBLC2=1.000e-20 DROUT  =5.000e-01
+A0     =5.850e-01 A1     =0.000e+00 A2     =1.000e+00
+PVAG   =0.000e+00 VSAT   =1.158e+05 AGS    =2.468e-01
+B0     =8.832e-08 B1     =0.000e+00 DELTA  =1.000e-02
+PDIBLCB=1.000e+00
+W0     =1.000e-10 DLC    =2.4500e-08
+DWC    =3.449e-08 DWB    =0.000e+00 DWG    =0.000e+00
+LL     =0.000e+00 LW     =0.000e+00 LWL    =0.000e+00
+LLN    =1.000e+00 LWN    =1.000e+00 WL     =0.000e+00
+WW     =1.894e-16 WWL    =-1.981e-21 WLN    =1.000e+00
+WWN    =1.040e+00
+TNOM   =27.0 AT     =3.300e+04 UTE    =-1.300e+00
+KT1    =-5.403e-01 KT2    =2.200e-02 KT1L   =0.000e+00
+UA1    =0.000e+00 UB1    =0.000e+00 UC1    =0.000e+00
+PRT    =0.000e+00
+CGDO   =8.600e-11 CGSO   =8.600e-11 CGBO   =1.100e-10
+CGDL   =1.080e-10 CGSL   =1.080e-10 CKAPPA =6.000e-01
+CF     =0.000e+00 ELM    =5.000e+00
+XPART  =1.000e+00 CLC    =1.000e-15 CLE    =6.000e-01
+RDSW   =1.033e+03
+CDSC   =2.589e-03 CDSCB  =2.943e-04 CDSCD  =4.370e-04
+PRWB   =-9.731e-02 PRWG   =1.477e-01 CIT    =0.000e+00
```

```
+TOX    =7.754e-09 NGATE  =0.000e+00
+NLX    =1.770e-07
+XL     =0.000e+00 XW     =0.000e+00
+ALPHA0 =0.000e+00 BETA0  =3.000e+01
+AF     =1.48e+00 KF     =8.5e-27 EF     =1.000e+00
+NOIA   =1.52e+18 NOIB   =7.75e+03 NOIC   =5.0e-13\
+ACM    =2
+RD     =0.000e+00 RS     =0.000e+00 RSH    =1.290e+02
+RDC    =0.000e+00 RSC    =0.000e+00
+LINT   =-7.130e-08  WINT   =3.449e-08
+LDIF   =0.000e+00 HDIF   =8.000e-07 WMLT   =1.000e+00
+LMLT   =1.000e+00 XJ     =3.000e-07
+JS     =9.000e-05 JSW    =0.000e+00 IS     =0.000e+00
+N      =1.000e+00 NDS    =1000.
+VNDS   =-1.000e+00 CBD    =0.000e+00 CBS    =0.000e+00 CJ     =1.360e-03 CJSW   =3.200e-10
+FC     =0.000e+00 MJ     =5.600e-01 MJSW   =4.300e-01 TT     =0.000e+00
+PB     =1.020e+00 PHP    =1.020e+00

.MODEL NWDINSUB D LEVEL=1
+IS     =6.0000e-05 JSW    =2.7000e-10 N      =1.000e+00
+CJ     =8.0000e-05 M      =3.9000e-01 VJ     =5.3000e-01 TT     =0.000e+00
+CJSW   =5.1000e-10 MJSW   =2.7000e-01 FC     =0.500e+00
+EG     =1.110e+00 XTI    =3.000e+00 AF      =1.000e+00 KF      =0.000e+00
```

A.4 Arquivo de modelo de transistor para a tecnologia 45nm utilizado nas simulações SPICE

```
*Customized PTM 45 PMOS PMOS _VTG

.model  PMOS _VTG  pmos  level = 54

+version = 4.0    binunit = 1    paramchk= 1    mobmod  = 0
+capmod  = 2      igcmod  = 1    igbmod  = 1    geomod  = 1
+diomod  = 1      rdsmod  = 0    rbodymod= 1    rgatemod= 1
+permod  = 1      acnqsmod= 0    trnqsmod= 0

* parameters related to the technology node
+tnom = 27    epsrox = 3.9
+eta0 = 0.0049    nfactor = 2.1    wint = 5e-09
+cgso = 1.1e-10    cgdo = 1.1e-10    xl = -2e-08

* parameters customized by the user
+toxe = 2.65e-09    toxp = 1.9e-09    toxm = 2.65e-09    toxref = 2.65e-09
+dtox = 7.5e-10    lint = 3.75e-09
+vth0 = -0.622    k1 = 0.694    u0 = 0.00444    vsat = 70000
+rdsw = 155    ndep = 2.47e+18    xj = 1.4e-08

*secondary parameters
+ll      = 0           wl      = 0           lln     = 1           wln     = 1
+lw      = 0           ww      = 0           lwn     = 1           wwn     = 1
+lwl     = 0           wwl     = 0           xpart   = 0
+k2      = -0.01       k3      = 0
+k3b     = 0           w0      = 2.5e-006    dvt0    = 1           dvt1    = 2  \
+dvt2    = -0.032      dvt0w   = 0           dvt1w   = 0           dvt2w   = 0
+dsub    = 0.1         minv    = 0.05        voffl   = 0           dvtp0   = 1e-009
+dvtp1   = 0.05        lpe0    = 0           lpeb    = 0
+ngate   = 2e+020      nsd     = 2e+020      phin    = 0
+cdsc    = 0.000       cdscb   = 0           cdscd   = 0           cit     = 0
+voff    = -0.126      etab    = 0
+vfb     = 0.55        ua      = 2.0e-009    ub      = 0.5e-018
+uc      = 0           a0      = 1.0         ags     = 1e-020
+a1      = 0           a2      = 1           b0      = -1e-020     b1      = 0
+keta    = -0.047      dwg     = 0           dwb     = 0           pclm    = 0.12
+pdiblc1 = 0.001       pdiblc2 = 0.001       pdiblcb = 3.4e-008    drout   = 0.56
+pvag    = 1e-020      delta   = 0.01        pscbe1  = 8.14e+008   pscbe2  = 9.58e-007
+fprout  = 0.2         pdits   = 0.08        pditsd  = 0.23        pditsl  = 2.3e+006
+rsh     = 5           rsw     = 85          rdw     = 85
+rdswmin = 0           rdwmin  = 0           rswmin  = 0           prwg    = 3.22e-008
+prwb    = 6.8e-011    wr      = 1           alpha0  = 0.074       alpha1  = 0.005
+beta0   = 30          agidl   = 0.0002      bgidl   = 2.1e+009    cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012       bigbacc = 0.0028      cigbacc = 0.002
+nigbacc = 1           aigbinv = 0.014       bigbinv = 0.004       cigbinv = 0.004
+eigbinv = 1.1         nigbinv = 3           aigc    = 0.69        bigc    = 0.0012
+cigc    = 0.0008      aigsd   = 0.0087      bigsd   = 0.0012      cigsd   = 0.0008
+nigc    = 1           poxedge = 1           pigcd   = 1           ntox    = 1

+xrcrg1  = 12          xrcrg2  = 5
+cgbo    = 2.56e-011   cgdl    = 2.653e-10
+cgsl    = 2.653e-10   ckappas = 0.03        ckappad = 0.03        acde    = 1
+moin    = 15          noff    = 0.9         voffcv  = 0.02

+kt1     = -0.11       kt11    = 0           kt2     = 0.022       ute     = -1.5
+ua1     = 4.31e-009   ub1     = 7.61e-018   uc1     = -5.6e-011   prt     = 0
+at      = 33000

+fnoimod = 1           tnoimod = 0

+jss     = 0.0001      jsws    = 1e-011      jswgs   = 1e-010       njs     = 1
+ijthsfwd= 0.01        ijthsrev= 0.001       bvs     = 10          xjbvs   = 1
+jsd     = 0.0001      jswd    = 1e-011      jswgd   = 1e-010       njd     = 1
+ijthdfwd= 0.01        ijthdrev= 0.001       bvd     = 10          xjbvd   = 1
+pbs     = 1           cjs     = 0.0005      mjs     = 0.5         pbsws   = 1
+cjsws   = 5e-010      mjsws   = 0.33        pbswgs  = 1           cjswgs  = 3e-010
```

```
+mjswgs  = 0.33         pbd     = 1           cjd     = 0.0005      mjd     = 0.5
+pbswd   = 1            cjswd   = 5e-010      mjswd   = 0.33        pbswgd  = 1
+cjswgd  = 5e-010       mjswgd  = 0.33        tpb     = 0.005       tcj     = 0.001
+tpbsw   = 0.005        tcjsw   = 0.001       tpbswg  = 0.005       tcjswg  = 0.001
+xtis    = 3            xtid    = 3
+dmcg    = 0e-006       dmci    = 0e-006      dmdg    = 0e-006      dmcgt   = 0e-007
+dwj     = 0.0e-008     xgw     = 0e-007      xgl     = 0e-008

+rshg    = 0.4          gbmin   = 1e-010      rbpb    = 5           rbpd    = 15
+rbps    = 15           rbdb    = 15          rbsb    = 15          ngcon   = 1

* Customized PTM 45 NMOS NMOS _VTG

.model  NMOS _VTG  nmos  level = 54

+version = 4.0    binunit = 1    paramchk= 1    mobmod  = 0
+capmod  = 2      igcmod  = 1    igbmod  = 1    geomod  = 1
+diomod  = 1      rdsmod  = 0    rbodymod= 1    rgatemod= 1
+permod  = 1      acnqsmod= 0    trnqsmod= 0

* parameters related to the technology node
+tnom = 27    epsrox = 3.9
+eta0 = 0.0049    nfactor = 2.1    wint = 5e-09
+cgso = 1.1e-10    cgdo = 1.1e-10    xl = -2e-08

* parameters customized by the user
+toxe = 2.55e-09    toxp = 1.9e-09    toxm = 2.55e-09    toxref = 2.55e-09
+dtox = 6.5e-10    lint = 3.75e-09
+vth0 = 0.677    k1 = 0.74    u0 = 0.04539    vsat = 147390
+rdsw = 155    ndep = 3.03e+18    xj = 1.4e-08

* secondary parameters
+ll      = 0            wl      = 0           lln     = 1           wln     = 1    \
+lw      = 0            ww      = 0           lwn     = 1           wwn     = 1
+lwl     = 0            wwl     = 0           xpart   = 0           \
+k2      = 0.01         k3      = 0
+k3b     = 0            w0      = 2.5e-006    dvt0    = 1           dvt1    = 2   \
+dvt2    = -0.032       dvt0w   = 0           dvt1w   = 0           dvt2w   = 0
+dsub    = 0.1          minv    = 0.05        voffl   = 0           dvtp0   = 1.0e-009   \
+dvtp1   = 0.1          lpe0    = 0           lpeb    = 0
+ngate   = 2e+020       nsd     = 2e+020      phin    = 0
+cdsc    = 0.000        cdscb   = 0           cdscd   = 0           cit     = 0
+voff    = -0.13        etab    = 0
+vfb     = -0.55        ua      = 6e-010      ub      = 1.2e-018
+uc      = 0            a0      = 1.0         ags     = 1e-020
+a1      = 0            a2      = 1.0         b0      = 0           b1      = 0
+keta    = 0.04         dwg     = 0           dwb     = 0           pclm    = 0.04
+pdiblc1 = 0.001        pdiblc2 = 0.001       pdiblcb = -0.005      drout   = 0.5
+pvag    = 1e-020       delta   = 0.01        pscbe1  = 8.14e+008   pscbe2  = 1e-007
+fprout  = 0.2          pdits   = 0.08        pditsd  = 0.23        pditsl  = 2.3e+006
+rsh     = 5            rsw     = 85          rdw     = 85
+rdswmin = 0            rdwmin  = 0           rswmin  = 0           prwg    = 0
+prwb    = 6.8e-011     wr      = 1           alpha0  = 0.074       alpha1  = 0.005
+beta0   = 30           agidl   = 0.0002      bgidl   = 2.1e+009    cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012        bigbacc = 0.0028      cigbacc = 0.002
+nigbacc = 1            aigbinv = 0.014       bigbinv = 0.004       cigbinv = 0.004 \
+eigbinv = 1.1          nigbinv = 3           aigc    = 0.012       bigc    = 0.0028
+cigc    = 0.002        aigsd   = 0.012       bigsd   = 0.0028      cigsd   = 0.002
+nigc    = 1            poxedge = 1           pigcd   = 1           ntox    = 1

+xrcrg1  = 12           xrcrg2  = 5
+cgbo    = 2.56e-011    cgdl    = 2.653e-10
+cgsl    = 2.653e-10    ckappas = 0.03        ckappad = 0.03        acde    = 1       \
+moin    = 15           noff    = 0.9         voffcv  = 0.02

+kt1     = -0.11        kt1l    = 0           kt2     = 0.022       ute     = -1.5
+ua1     = 4.31e-009    ub1     = 7.61e-018   uc1     = -5.6e-011   prt     = 0
+at      = 33000

+fnoimod = 1            tnoimod = 0

+jss     = 0.0001       jsws    = 1e-011      jswgs   = 1e-010       njs     = 1
+ijthsfwd= 0.01         ijthsrev= 0.001       bvs     = 10          xjbvs   = 1
+jsd     = 0.0001       jswd    = 1e-011      jswgd   = 1e-010       njd     = 1
+ijthdfwd= 0.01         ijthdrev= 0.001       bvd     = 10          xjbvd   = 1
+pbs     = 1            cjs     = 0.0005      mjs     = 0.5         pbsws   = 1
+cjsws   = 5e-010       mjsws   = 0.33        pbswgs  = 1           cjswgs  = 3e-010
+mjswgs  = 0.33         pbd     = 1           cjd     = 0.0005      mjd     = 0.5
+pbswd   = 1            cjswd   = 5e-010      mjswd   = 0.33        pbswgd  = 1
+cjswgd  = 5e-010       mjswgd  = 0.33        tpb     = 0.005       tcj     = 0.001
+tpbsw   = 0.005        tcjsw   = 0.001       tpbswg  = 0.005       tcjswg  = 0.001
+xtis    = 3            xtid    = 3

+dmcg    = 0e-006       dmci    = 0e-006      dmdg    = 0e-006      dmcgt   = 0e-007
+dwj     = 0.0e-008     xgw     = 0e-007      xgl     = 0e-008

+rshg    = 0.4          gbmin   = 1e-010      rbpb    = 5           rbpd    = 15
+rbps    = 15           rbdb    = 15          rbsb    = 15          ngcon   = 1
```

A.5 Arquivo SPICE usado na simulação para calcular o valor de capacitância de entrada para um transistor PMOS na tecnologia de 350nm

```
@================== Cin 350nm PMOS ==================
.include mos _350nm.pm

.param vvdd = 3.3
.param vgnd = 0.0
.param res = 10k
.global vdd
.global gnd
Vsrc vdd 0 3.3

ri gin n1 'res'
M3  n2 n1  gnd vdd modp l=350e-009 w=1e-006
M4  n3 n2  gnd vdd modp l=350e-009 w=4e-006

.measure tran tpdr  trig v(gin) val='vvdd/2' rise=1 targ v(n1) val='vvdd/2' rise=1
.measure cin param='tpdr/(res*0.693)'

Vdt gin 0 pulse(vgnd vvdd 0 0p 0p 10n 20n)

.option post

.tran 1e-13 10n
.end
```

A.6 Arquivo SPICE usado na simulação para calcular o valor de capacitância de entrada para um transistor NMOS na tecnologia de 45nm

```
@================== Cin 45nm NMOS ==================
.include mos _45.sp
.param wmin = 90nm
.param vvdd = 1.1
.param vgnd = 0.0

.param res = 1k

.global vdd
.global gnd

Vsrc vdd 0 1.1

ri gin n1 'res'
M3  n2 n1  gnd gnd NMOS _VTG l=50e-009 w=90e-009
M4  n3 n2  gnd gnd NMOS _VTG l=50e-009 w=360e-009

.measure tran tpdr  trig v(gin) val='vvdd/2' rise=1 targ v(n1) val='vvdd/2' rise=1

.measure cin param='tpdr/(res*0.693)'

Vdt gin 0 pulse(vgnd vvdd 0 0p 0p 5n 10n)

.option post
.tran 1e-13 10n
.end
```

A.7 Arquivo SPICE usado na simulação para calcular o valor de capacitância de dreno/fonte para o substrato de um transistor NMOS na tecnologia de 350nm

```
@================== Cdfb 350nm NMOS ==================
.include mos _350nm.pm
.param vvdd = 3.3
.param vgnd = 0.0
.param res = 10k
.global vdd
.global gnd

Vsrc vdd 0 3.3

ri gin n1 'res'
M3  n1 gnd  n2 gnd modn l=350e-009 w=1e-006

.measure tran tpdr  trig v(gin) val='vvdd/2' rise=1 targ v(n1) val='vvdd/2' rise=1
.measure csb param='tpdr/(res*0.693)'

Vdt gin 0 pulse(vgnd vvdd 0 0p 0p 10n 20n)

.option post

.tran 1e-13 10n
.end
```

A.8 Arquivo SPICE usado na simulação para calcular o valor de capacitância de dreno/fonte para o substrato de um transistor PMOS na tecnologia de 45nm

```
@================== Cdfb 45nm PMOS ==================
.include mos _45.sp
.param vvdd = 1.1
.param vgnd = 0.0
.param res = 1k
.global vdd
.global gnd

Vsrc vdd 0 1.1
ri gin n1 'res'
M3  n1 vdd  n2 vdd PMOS _VTG l=50e-009 w=90e-009

.measure tran tpdr  trig v(gin) val='vvdd/2' rise=1 targ v(n1) val='vvdd/2' rise=1
.measure csb param='tpdr/(res*0.693)'

Vdt gin 0 pulse(vgnd vvdd 0 0p 0p 10n 20n)

.option post

.tran 1e-13 10n
.end
```

A.9 Arquivo SPICE usado na simulação para calcular o valor da resistência equivalente de um transistor NMOS na tecnologia de 350nm

```
@================== Req 350nm NMOS ==================
.include mos _350nm.pm
.param vvdd = 3.3
.param vgnd = 0.0
.param cap = 10f

.global vdd
.global gnd

Vsrc vdd 0 3.3

x1 in out inv
cLoad out 0 'cap'

.measure tran tpdr  trig v(in) val='vvdd/2' rise=1 targ v(out) val='vvdd/2' fall=1
.measure Req param='tpdr/(cap*0.693)'

Vdt in 0 pulse(vgnd vvdd 100p 0p 0p 10n 20n)

.option post

.tran 1e-13 20n

.subckt inv in out
M1  out in  vdd vdd   MODP l=350n w=1u
M2  out in  0   0     MODN l=350n w=1u
.ends inv
.end
```

A.10 Arquivo SPICE usado na simulação para calcular o valor da resistência equivalente de um transistor PMOS na tecnologia de 45nm

```
@================== Req 45nm PMOS ==================
.include mos _45.sp
.param vvdd = 1.1
.param vgnd = 0.0

.param cap = 10f
.global vdd
.global gnd

Vsrc vdd 0 1.1

x1 in out inv
cLoad out 0 'cap'

.measure tran tpdr  trig v(in) val='vvdd/2' fall=1 targ v(out) val='vvdd/2' rise=1
.measure Req param='tpdr/(cap*0.693)'

Vdt in 0 pulse(vvdd vgnd 100p 0p 0p 10n 20n)

.option post
.tran 1e-13 20n
```

```
22
23  .subckt inv in out
24  M1  out in  vdd vdd   PMOS _VTG l=50n w=90n
25  M2  out in  0   0     NMOS _VTG l=50n w=90n
26  .ends inv
27  .end
```

APÊNDICE B ARQUIVO QUE DESCREVE O PROBLEMA DE OTIMIZAÇÃO DE UM CIRCUITO DE EXEMPLO PARA SER RESOLVIDO POR PROGRAMAÇÃO GEOMÉTRICA

```
gpvar X1 X2 X3 X4 X5 X6;
Cload = 8.60927e-16;
constrArea = 2000;
constrCin = 4;
maxDelay = 4.29113e-10;
Xmax = 32;
Xmin = 1;
CsbP = 7.17122e-16;
CsbN = 7.94589e-16;
CgateP = 7.288e-16;
CgateN = 9.88656e-16;
ReqP = 4948.83;
ReqN = 1402.74;
Xn = 0.09;
Xp = 0.135;
Vdd = 1.1;
Vdd2 = 1.21;
Cin _Base _X1 _A = CgateN * 2 + CgateP * 2;
Cin _X1 _A = (Cin _Base _X1 _A) * (X1);
Cin _Base _X1 _B = CgateN * 2 + CgateP * 2;
Cin _X1 _B = (Cin _Base _X1 _B) * (X1);
Cin _Base _X2 _A = CgateN * 2 + CgateP * 2;
Cin _X2 _A = (Cin _Base _X2 _A) * (X2);
Cin _Base _X2 _B = CgateN * 2 + CgateP * 2;
Cin _X2 _B = (Cin _Base _X2 _B) * (X2);
Cin _Base _X3 _IN = CgateN * 4 + CgateP * 4;
Cin _X3 _IN = (Cin _Base _X3 _IN) * (X3);
Cin _Base _X4 _A = CgateN * 2 + CgateP * 2;
Cin _X4 _A = (Cin _Base _X4 _A) * (X4);
Cin _Base _X4 _B = CgateN * 2 + CgateP * 2;
Cin _X4 _B = (Cin _Base _X4 _B) * (X4);
Cin _Base _X5 _IN = CgateN * 4 + CgateP * 4;
Cin _X5 _IN = (Cin _Base _X5 _IN) * (X5);
Cin _Base _X6 _A = CgateN * 2 + CgateP * 2;
Cin _X6 _A = (Cin _Base _X6 _A) * (X6);
Cin _Base _X6 _B = CgateN * 2 + CgateP * 2;
Cin _X6 _B = (Cin _Base _X6 _B) * (X6);
Cload _X1 = (Cload);
Cload _X2 = (Cin _X1 _B);
Cload _X3 = (Cin _X1 _A) + (Cin _X6 _A);
Cload _X4 = (Cin _X2 _B);
Cload _X5 = (Cin _X2 _A);
Cload _X6 = (Cload);
Xn _X1 = Xn*X1;
Xp _X1 = Xp*X1;
Rtrans0 _X1 = 701.37*X1\^-1;
Rtrans1 _X1 = 2474.41*X1\^-1;
Rtrans2 _X1 = 701.37*X1\^-1;
Rtrans3 _X1 = 2474.41*X1\^-1;
Ctrans0 _X1 = 1.58918e-15*X1;
Ctrans1 _X1 = 1.43424e-15*X1;
Ctrans2 _X1 = 1.58918e-15*X1;
Ctrans3 _X1 = 1.43424e-15*X1;
Abase _X1 = 4 + 4;
Afinal _X1 = (Abase _X1) * (X1);
C _0 _4 _X1 = (Ctrans0 _X1) + (Ctrans1 _X1) + (Ctrans3 _X1) + (Cload _X1);
C _0 _5 _X1 = (Ctrans0 _X1) + (Ctrans2 _X1);
Cdown _0 _4 _X1 = (C _0 _4 _X1) + (C _0 _5 _X1);
Cdown _0 _5 _X1 = (C _0 _5 _X1);
D _0 _4 _X1 = (Rtrans3 _X1) * (Cdown _0 _4 _X1);
C _1 _4 _X1 = (Ctrans0 _X1) + (Ctrans1 _X1) + (Ctrans3 _X1) + (Cload _X1);
Cdown _1 _4 _X1 = (C _1 _4 _X1);
D _1 _4 _X1 = (Rtrans1 _X1) * (Cdown _1 _4 _X1);
C _2 _4 _X1 = (Ctrans0 _X1) + (Ctrans1 _X1) + (Ctrans3 _X1) + (Cload _X1);
C _2 _5 _X1 = (Ctrans0 _X1) + (Ctrans2 _X1);
Cdown _2 _4 _X1 = (C _2 _4 _X1);
Cdown _2 _5 _X1 = (C _2 _4 _X1) + (C _2 _5 _X1);
D _2 _5 _X1 = (Rtrans2 _X1) * (Cdown _2 _5 _X1);
D _2 _4 _X1 = (D _2 _5 _X1) + ((Rtrans0 _X1) * (Cdown _2 _4 _X1));
delayX1 = max( D _0 _4 _X1, D _1 _4 _X1, D _2 _4 _X1 );
```

```
Xn _X2 = Xn*X2;
Xp _X2 = Xp*X2;
Rtrans0 _X2 = 701.37*X2\^-1;
Rtrans1 _X2 = 2474.41*X2\^-1;
Rtrans2 _X2 = 701.37*X2\^-1;
Rtrans3 _X2 = 2474.41*X2\^-1;
Ctrans0 _X2 = 1.58918e-15*X2;
Ctrans1 _X2 = 1.43424e-15*X2;
Ctrans2 _X2 = 1.58918e-15*X2;
Ctrans3 _X2 = 1.43424e-15*X2;
Abase _X2 = 4 + 4;
Afinal _X2 = (Abase _X2) * (X2);
C _0 _4 _X2 = (Ctrans0 _X2) + (Ctrans1 _X2) + (Ctrans2 _X2) + (Cload _X2);
C _0 _5 _X2 = (Ctrans1 _X2) + (Ctrans3 _X2);
Cdown _0 _4 _X2 = (C _0 _4 _X2);
Cdown _0 _5 _X2 = (C _0 _4 _X2) + (C _0 _5 _X2);
D _0 _5 _X2 = (Rtrans3 _X2) * (Cdown _0 _5 _X2);
D _0 _4 _X2 = (D _0 _5 _X2) + ((Rtrans1 _X2) * (Cdown _0 _4 _X2));
C _1 _4 _X2 = (Ctrans0 _X2) + (Ctrans1 _X2) + (Ctrans2 _X2) + (Cload _X2);
Cdown _1 _4 _X2 = (C _1 _4 _X2);
D _1 _4 _X2 = (Rtrans0 _X2) * (Cdown _1 _4 _X2);
C _2 _4 _X2 = (Ctrans0 _X2) + (Ctrans1 _X2) + (Ctrans2 _X2) + (Cload _X2);
C _2 _5 _X2 = (Ctrans1 _X2) + (Ctrans3 _X2);
Cdown _2 _4 _X2 = (C _2 _4 _X2) + (C _2 _5 _X2);
Cdown _2 _5 _X2 = (C _2 _5 _X2);
D _2 _4 _X2 = (Rtrans2 _X2) * (Cdown _2 _4 _X2);
delayX2 = max( D _0 _4 _X2, D _1 _4 _X2, D _2 _4 _X2 );
Xn _X3 = Xn*X3;
Xp _X3 = Xp*X3;
Rtrans0 _X3 = 350.685*X3\^-1;
Rtrans1 _X3 = 1237.21*X3\^-1;
Ctrans0 _X3 = 3.17836e-15*X3;
Ctrans1 _X3 = 2.86849e-15*X3;
Abase _X3 = 4 + 4;
Afinal _X3 = (Abase _X3) * (X3);
C _0 _3 _X3 = (Ctrans0 _X3) + (Ctrans1 _X3) + (Cload _X3);
Cdown _0 _3 _X3 = (C _0 _3 _X3);
D _0 _3 _X3 = (Rtrans1 _X3) * (Cdown _0 _3 _X3);
C _1 _3 _X3 = (Ctrans0 _X3) + (Ctrans1 _X3) + (Cload _X3);
Cdown _1 _3 _X3 = (C _1 _3 _X3);
D _1 _3 _X3 = (Rtrans0 _X3) * (Cdown _1 _3 _X3);
delayX3 = max( D _0 _3 _X3, D _1 _3 _X3 );
Xn _X4 = Xn*X4;
Xp _X4 = Xp*X4;
Rtrans0 _X4 = 701.37*X4\^-1;
Rtrans1 _X4 = 2474.41*X4\^-1;
Rtrans2 _X4 = 701.37*X4\^-1;
Rtrans3 _X4 = 2474.41*X4\^-1;
Ctrans0 _X4 = 1.58918e-15*X4;
Ctrans1 _X4 = 1.43424e-15*X4;
Ctrans2 _X4 = 1.58918e-15*X4;
Ctrans3 _X4 = 1.43424e-15*X4;
Abase _X4 = 4 + 4;
Afinal _X4 = (Abase _X4) * (X4);
C _0 _4 _X4 = (Ctrans0 _X4) + (Ctrans1 _X4) + (Ctrans3 _X4) + (Cload _X4);
C _0 _5 _X4 = (Ctrans0 _X4) + (Ctrans2 _X4);
Cdown _0 _4 _X4 = (C _0 _4 _X4) + (C _0 _5 _X4);
Cdown _0 _5 _X4 = (C _0 _5 _X4);
D _0 _4 _X4 = (Rtrans3 _X4) * (Cdown _0 _4 _X4);
C _1 _4 _X4 = (Ctrans0 _X4) + (Ctrans1 _X4) + (Ctrans3 _X4) + (Cload _X4);
Cdown _1 _4 _X4 = (C _1 _4 _X4);
D _1 _4 _X4 = (Rtrans1 _X4) * (Cdown _1 _4 _X4);
C _2 _4 _X4 = (Ctrans0 _X4) + (Ctrans1 _X4) + (Ctrans3 _X4) + (Cload _X4);
C _2 _5 _X4 = (Ctrans0 _X4) + (Ctrans2 _X4);
Cdown _2 _4 _X4 = (C _2 _4 _X4);
Cdown _2 _5 _X4 = (C _2 _4 _X4) + (C _2 _5 _X4);
D _2 _5 _X4 = (Rtrans2 _X4) * (Cdown _2 _5 _X4);
D _2 _4 _X4 = (D _2 _5 _X4) + ((Rtrans0 _X4) * (Cdown _2 _4 _X4));
delayX4 = max( D _0 _4 _X4, D _1 _4 _X4, D _2 _4 _X4 );
Xn _X5 = Xn*X5;
Xp _X5 = Xp*X5;
Rtrans0 _X5 = 350.685*X5\^-1;
Rtrans1 _X5 = 1237.21*X5\^-1;
Ctrans0 _X5 = 3.17836e-15*X5;
Ctrans1 _X5 = 2.86849e-15*X5;
Abase _X5 = 4 + 4;
Afinal _X5 = (Abase _X5) * (X5);
C _0 _3 _X5 = (Ctrans0 _X5) + (Ctrans1 _X5) + (Cload _X5);
Cdown _0 _3 _X5 = (C _0 _3 _X5);
D _0 _3 _X5 = (Rtrans1 _X5) * (Cdown _0 _3 _X5);
C _1 _3 _X5 = (Ctrans0 _X5) + (Ctrans1 _X5) + (Cload _X5);
Cdown _1 _3 _X5 = (C _1 _3 _X5);
D _1 _3 _X5 = (Rtrans0 _X5) * (Cdown _1 _3 _X5);
delayX5 = max( D _0 _3 _X5, D _1 _3 _X5 );
Xn _X6 = Xn*X6;
Xp _X6 = Xp*X6;
Rtrans0 _X6 = 701.37*X6\^-1;
Rtrans1 _X6 = 2474.41*X6\^-1;
Rtrans2 _X6 = 701.37*X6\^-1;
Rtrans3 _X6 = 2474.41*X6\^-1;
Ctrans0 _X6 = 1.58918e-15*X6;
Ctrans1 _X6 = 1.43424e-15*X6;
Ctrans2 _X6 = 1.58918e-15*X6;
Ctrans3 _X6 = 1.43424e-15*X6;
Abase _X6 = 4 + 4;
Afinal _X6 = (Abase _X6) * (X6);
C _0 _4 _X6 = (Ctrans0 _X6) + (Ctrans1 _X6) + (Ctrans2 _X6) + (Cload _X6);
C _0 _5 _X6 = (Ctrans1 _X6) + (Ctrans3 _X6);
Cdown _0 _4 _X6 = (C _0 _4 _X6);
```

```
Cdown _0 _5 _X6 = (C _0 _4 _X6) + (C _0 _5 _X6);
D _0 _5 _X6 = (Rtrans3 _X6) * (Cdown _0 _5 _X6);
D _0 _4 _X6 = (D _0 _5 _X6) + ((Rtrans1 _X6) * (Cdown _0 _4 _X6));
C _1 _4 _X6 = (Ctrans0 _X6) + (Ctrans1 _X6) + (Ctrans2 _X6) + (Cload _X6);
Cdown _1 _4 _X6 = (C _1 _4 _X6);
D _1 _4 _X6 = (Rtrans0 _X6) * (Cdown _1 _4 _X6);
C _2 _4 _X6 = (Ctrans0 _X6) + (Ctrans1 _X6) + (Ctrans2 _X6) + (Cload _X6);
C _2 _5 _X6 = (Ctrans1 _X6) + (Ctrans3 _X6);
Cdown _2 _4 _X6 = (C _2 _4 _X6) + (C _2 _5 _X6);
Cdown _2 _5 _X6 = (C _2 _5 _X6);
D _2 _4 _X6 = (Rtrans2 _X6) * (Cdown _2 _4 _X6);
delayX6 = max( D _0 _4 _X6, D _1 _4 _X6, D _2 _4 _X6 );
D _X3 = (delayX3);
D _X6 = (delayX6) + (D _X3);
D _X5 = (delayX5);
D _X4 = (delayX4);
D _X2 = (delayX2) + (max( D _X5, D _X4 ));
D _X1 = (delayX1) + (max( D _X3, D _X2 ));
delay = max( D _X1, D _X2, D _X3, D _X4, D _X5, D _X6 );
Afinal = (Afinal _X1) + (Afinal _X2) + (Afinal _X3) + (Afinal _X4) + (Afinal _X5) + (Afinal _X6);
Abase = Abase _X1 + Abase _X2 + Abase _X3 + Abase _X4 + Abase _X5 + Abase _X6;
Power = ((Cin _X1 _A) + (Cin _X1 _B) + (Cin _X2 _A) + (Cin _X2 _B) + (Cin _X3 _IN) + (Cin _X4 _A) + (Cin _X4 _B) + (Cin _X5 _IN) + (Cin _X6 _A

constr = [
        Afinal <= constrArea * 1;
        Xmin <= X1;
        X1 <= Xmax;
        Xmin <= X2;
        X2 <= Xmax;
        Xmin <= X3;
        X3 <= Xmax;
        Xmin <= X4;
        X4 <= Xmax;
        Xmin <= X5;
        X5 <= Xmax;
        Xmin <= X6;
        X6 <= Xmax;
        Cin _X3 _IN <= constrCin * Cin _Base _X3 _IN;
        Cin _X4 _A <= constrCin * Cin _Base _X4 _A;
        Cin _X4 _B <= constrCin * Cin _Base _X4 _B;
        Cin _X5 _IN <= constrCin * Cin _Base _X5 _IN;
        Cin _X6 _B <= constrCin * Cin _Base _X6 _B;
];

[ result, solution, status ] = gpsolve( delay, constr, 'min' );

assign(solution);
```