UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Vinícius dos Santos Livramento

SIZING DISCRETO BASEADO EM RELAXAÇÃO LAGRANGEANA PARA MINIMIZAÇÃO DE *LEAKAGE* EM CIRCUITOS DIGITAIS

Florianópolis(SC)

2013

Vinícius dos Santos Livramento

SIZING DISCRETO BASEADO EM RELAXAÇÃO LAGRANGEANA PARA MINIMIZAÇÃO DE *LEAKAGE* EM CIRCUITOS DIGITAIS

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do Grau de Mestre em Ciência da Computação.
Orientador: Prof. Dr. José Luís Almada Güntzel
Coorientador: Prof. Dr. Marcelo de Oliveira Johann

Florianópolis(SC)

2013

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Livramento, Vinícius dos Santos
Sizing Discreto Baseado em Relaxação Lagrangeana para Minimização de Leakage em Circuitos Digitais / Vinícius dos Santos Livramento ; orientador, José Luís Almada Güntzel ; co-orientador, Marcelo de Oliveira Johann. - Florianópolis, SC, 2013.
131 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Automação de Projeto Eletrônico (EDA). 3. Sizing Discreto de Portas Lógicas. 4. Relaxação Lagrangeana. I. Güntzel, José Luís Almada. II. Johann, Marcelo de Oliveira. III. Universidade Federal de Santa Catarina. Programa de Pós-Graduação em Ciência da Computação. IV. Título.

Vinícius dos Santos Livramento

SIZING DISCRETO BASEADO EM RELAXAÇÃO LAGRANGEANA PARA MINIMIZAÇÃO DE _LEAKAGE_ EM CIRCUITOS DIGITAIS

Esta Dissertação foi julgada aprovada para a obtenção do Título de "Mestre em Ciência da Computação", e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

Florianópolis(SC), 23 de Fevereiro 2013.

Prof. Dr. Ronaldo dos Santos Mello
Coordenador

Prof. Dr. José Luís Almada Güntzel
Orientador

Banca Examinadora:

Prof. Dr. José Luís Almada Güntzel
Presidente

Prof. Dr. Marcelo de Oliveira Johann
Coorientador

Prof. Dr. Djones Vinicius Lettnin

Prof. Dr. Luiz Cláudio Villar dos Santos

Prof. Dr. Marcelo Soares Lubaszewski

À minha família.

AGRADECIMENTOS

Aos meus pais, Mário e Cristina, pelo amor, carinho e dedicação que nunca faltaram. Também aos meus irmãos, Natália e Vitor, pelo apoio e compreensão.

Agradeço à minha namorada Juliana pela paciência logo início do namoro, que coincidiu justamente com a etapa final da dissertação.

Ao meu orientador, José Luís Güntzel, pela confiança, dedicação e incentivos presentes durante todo o período do mestrado. Agradeço também por sua excelente orientação e rigor exigido, os quais foram fundamentais para o sucesso desta dissertação de mestrado.

Ao meu coorientador, Marcelo de Oliveira Johann, pela grande colaboração com o trabalho. Em especial às horas de reunião via skype, as quais foram fundamentais a este trabalho.

Aos demais membros da banca pelo tempo dedicado para uma revisão rigorosa e pelas sugestões que contribuiram com este trabalho.

Aos colegas do ECL que de alguma forma participaram deste trabalho. Em particular, ao colega Chrystian Guth por todo suporte técnico prestado, essencial para o sucesso deste trabalho.

Ao CNPq, no âmbito do Programa Nacional de Microeletrônica, pelo custeio parcial da execução deste trabalho (Processo número: 551476/2010-9).

"If the automobile industry advanced as rapidly as the semiconductor industry, a Rolls Royce would get half a million miles per gallon, and it would be cheaper to throw it away than to park it." (Gordon Moore, 1998).

RESUMO

A minimização da corrente de *leakage* é um passo essencial do projeto de circuitos digitais, uma vez que nas tecnologias CMOS recentes a potência de *leakage* tornou-se comparável à potência dinâmica. *Gate sizing* é uma técnica amplamente utilizada para minimização da potência de *leakage* devido à sua eficácia e ao baixo impacto que ele causa no fluxo *standard cell*. Em tal fluxo, o problema de *sizing* corresponde a selecionar, para cada porta do circuito, uma combinação de largura de porta e tensão de *threshold* disponível na biblioteca de células, de modo a satisfazer as restrições de projeto. A natureza discreta do problema, a qual o torna NP-difícil, e o grande número de portas nos circuitos contemporâneos têm motivado a busca por heurísticas eficientes, que sejam capazes de resolvê-lo em tempo de execução aceitável. Este trabalho apresenta três contribuições principais ao estado da arte. A primeira é uma formulação aperfeiçoada para o problema de *sizing* discreto baseada em Relaxação Lagrangeana (LR), a qual considera valores máximos de *slew* de entrada e de capacitância de saída das portas, impostas pelas bibliotecas *standard cell*. A segunda é uma heurística topológica gulosa para resolver a formulação LR proposta utilizando informações locais para guiar as decisões do algoritmo. A terceira contribuição reside em uma técnica híbrida de três passos para superar algumas das limitações da heurística topológica gulosa. Tal técnica híbrida inicia resolvendo a formulação LR assumindo um atraso crítico ligeiramente maior do que o atraso crítico-alvo e em seguida, aplica uma heurística rápida de recuperação de atraso para que o atraso crítico-alvo original seja satisfeito. Como terceiro passo, é usada uma heurística de recuperação de potência para reduzir ainda mais a potência de *leakage* explorando o espaço para otimização deixado pelos dois passos anteriores. Os experimentos práticos foram gerados utilizando-se a infraestrutura da Competição de *Sizing* Discreto do ISPD2012, a qual provê uma base comum para comparações justas com os trabalhos correlatos mais recentes. Os resultados experimentais para a formulação LR usando a heurística topológica gulosa foram comparados com os resultados obtidos pelas três equipes melhor classificadas na Competição do ISPD 2012, os quais representavam o estado da arte no momento em que tais experimentos foram realizados. A potência de *leakage* obtida é, em média, 18,9%, 16,7% e 43,8% menor do que aquelas obtidas pelas três melhores equipes da Competição do ISPD2012, respectivamente, ao passo que o tempo de execução total é 38, 31 e 39 vezes menor. Com relação à técnica híbrida, a potência de *leakage* obtida é, em média, 8,15% menor do que aquela relatada pelo trabalho que representa o estado da arte na ocasião em que estes experimentos foram realizados, sendo o tempo total de execução uma ordem de magnitude menor. É

Importante ressaltar que o trabalho estado da arte referido já havia superado as três melhores equipes da Competição do ISPD2012.

Palavras-chave: Automação de projeto eletrônico (EDA). Fluxo de projeto *standard cell*. Minimização da potência de *leakage*. *Sizing* discreto de portas. Relaxação Lagrangeana.

ABSTRACT

Leakage current minimization is an essential step in the design of digital circuits, as leakage power became comparable to dynamic power in recent CMOS technologies. Gate sizing is a widely used technique to minimize leakage due to its effectiveness and its low impact on the standard cell flow. In such flow, the sizing problem corresponds to selecting, for each gate in the circuit, a combination of gate width and threshold voltage available in the cell library in such a way the design constraints are met. The discrete nature of the problem, which makes it NP-hard, and the large number of gates in contemporary circuits motivate the search for efficient heuristics able to solve it within acceptable runtimes. This work presents three main contributions to the state-of-the-art. The first one is an improved Lagrangian Relaxation (LR) formulation for the discrete gate sizing problem that accounts for the maximum gate input slew and maximum gate output capacitance constraints imposed by standard cell libraries. The second one is a topological greedy heuristic for solving the proposed LR formulation relying on local information to guide the algorithm's decisions. The third contribution relies on a three-step hybrid technique to overcome some limitations of the topological greedy heuristic. Such hybrid technique begins by solving the LR formulation by slightly loosening the delay constraint and then applies a fast delay recovery heuristic to meet the original delay constraint. As a third step, a leakage power recovery heuristic is used to further reduce leakage power by exploring the room for optimization left by the two previous steps. The practical experiments relied on the up-to-date ISPD 2012 Discrete Gate Sizing Contest infrastructure, which provided a common basis for fair comparisons with most recent related works. The experimental results for the LR formulation using the topological greedy heuristic were compared to those from the top three teams of the ISPD 2012 Contest, which represented the state-of-the-art at the time such experiments were conducted. The obtained leakage power is, on average, 18.9%, 16.7% and 43.8% smaller than those obtained by the top three teams of the ISPD 2012 Contest, respectively, while the total runtime is 38, 31 and 39 times shorter. Concerning the hybrid technique, the obtained leakage power is, on average, 8.15% smaller than that reported by the state-of-the-art work at that time, being the total execution time one order of magnitude faster. It is important to highlight that the referred state-of-the-art work had already surpassed the top three teams of the ISPD 2012 Contest.

Keywords: Electronic Design Automation (EDA). Standard cell design flow. Leakage power minimization. Discrete gate sizing. Lagrangian Relaxation (LR).

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ALGORITMOS

LISTA DE ACRÔNIMOS

CMOS	*Complementary Metal-Oxide Semiconductor*
DAG	*Directed Acyclic Graph* (Grafo Acíclico Direcionado)
DP	*Dynamic Programming* (Programação Dinâmica)
EDA	*Eletronic Design Automation* (Automação de Projeto Eletrônico)
ERC	*Electrical Rule Checking* (Checagem de Regras Elétricas)
ITA	*Incremental Timing Analysis* (Análise de *Timing* Incremental)
ITRS	*International Technology Roadmap for Semiconductors*
KKT	*Karush-Kuhn-Tucker*
LDP	*Lagrangian Dual Problem* (Problema Lagrangeano Dual)
LF	*Lagrangian Function* (Função Lagrangeana)
LM(s)	*Lagrangian Multiplier(s)* (Multiplicador(es) de Lagrange)
LP	*Linear Programming* (Programação Linear)
LR	*Lagrangian Relaxation* (Relaxação Lagrangeana)
LRS	*Lagrangian Relaxation Subproblem* (Subproblema Lagrangeano Relaxado)
NLDM	*Non-Linear Delay Model*
PMD	*Personal Mobile Device* (Dispositivo Pessoal Portátil)
RC	*Resistor-Capacitor*
PP	*Primal Problem* (Problema Primal)
RTL	*Register Transfer Level* (Nível de Transferência de Registradores)
SDC	*Synopsys Design Constraints*
SF	*Sensitivity Function* (Função de Sensibilidade)
SPEF	*Standard Parasitic Exchange Format*

STA *Static Timing Analysis*
(Análise de *Timing* Estática)
TNS *Total Negative Slack*

LISTA DE SÍMBOLOS

Representação de um circuito digital:
V: Conjunto de vértices em um DAG, onde $V = X \cup PI \cup PO$.
X: Conjunto de portas lógicas em um DAG.
PI: Conjunto de entradas primárias.
PO: Conjunto de saídas primárias.
E: Conjunto de arestas do DAG.
S: Nodo fonte do DAG.
T: Nodo terminal do DAG.
v_i: Um vértice i do DAG.
v_i^k: Um vértice i em sua opção de implementação k.
e_{ji}: Uma aresta entre um nodo v_j e um nodo v_i.

Análise de Timing Estática:
a_i^f: Tempo de chegada de descida na saída de v_i.
a_i^r: Tempo de chegada de subida na saída de v_i.
r_i^f: Tempo requerido de descida na saída de v_i.
r_i^r: Tempo requerido de subida na saída de v_i.
s_i^f: *Slack* de descida na saída de v_i.
s_i^r: *Slack* de subida na saída de v_i.
$d_{j\to i}^f$: Atraso de descida entre uma entrada j e a saída de v_i.
$d_{j\to i}^r$: Atraso de subida entre uma entrada j e a saída de v_i.
$slew_i^f$: *Slew* de descida na saída de v_i.
$slew_i^r$: *Slew* de subida na saída de v_i.
$slew_{j\to i}^f$: *Slew* de descida entre uma entrada j e a saída de v_i.
$slew_{j\to i}^r$: *Slew* de subida entre uma entrada j e a saída de v_i.
cap_i: Capacitância de saída de um nodo v_i.
max_slew: Restrição de máximo *slew* imposta pela biblioteca *standard cell*.
max_cap_i: Restrição de máxima capacitância de v_i imposta pela biblioteca *standard cell*.
A_o: Restrição de atraso crítico-alvo.
$fanin(v_i)$: Conjunto de portas que são *fanin* do nodo v_i.
$fanout(v_i)$: Conjunto de portas que são *fanout* do nodo v_i.

Biblioteca standard cell:
w_i: Largura de v_i.
u_i: Tensão de *threshold* de v_i.

W_i: Conjunto discreto das larguras disponíveis na biblioteca para v_i.
U_i: Conjunto discreto das tensões de *threshold* disponíveis na biblioteca para v_i.
p_i: Consumo de *leakage* de um nodo v_i.
$\Delta d(v_i^k, v_i^l)$: Variação de atraso resultante da troca da opção de implementação de v_i.
$\Delta leakage(v_i^k, v_i^l)$: Variação de *leakage* resultante da troca da opção de implementação de v_i.

Multiplicadores de Lagrange:
λ: Multiplicador de Lagrange relativo às restrições de *timing* do circuito.
$\lambda_{j\to i}^{f}$: Multiplicador de Lagrange associado ao arco $j \to i$ de descida de v_i.
$\lambda_{j\to i}^{r}$: Multiplicador de Lagrange associado ao arco $j \to i$ de subida de v_i.
γ: Multiplicador de Lagrange relativo à restrição de máximo *slew*.
γ_i^f: Multiplicador de Lagrange associado ao *slew* de descida de v_i.
γ_i^r: Multiplicador de Lagrange associado ao *slew* de subida de v_i.
β: Multiplicador de Lagrange relativo às restrições de máxima capacitância de saída.
β_i: Multiplicador de Lagrange associado à capacitância de saída do nodo v_i.

SUMÁRIO

1 INTRODUÇÃO

Este capítulo visa apresentar o problema de *sizing* discreto, objeto desta disssertação, destacando sua relevância para o contexto do projeto de circuitos digitais contemporâneos. Inicialmente, são apresentadas a motivação e uma breve contextualização do tema proposto. Em seguida, o fluxo de projeto de circuitos digitais contemporâneos é apresentado, apontando-se onde se encaixa o problema-alvo dessa dissertação. Após, o problema de *sizing* é formalmente definido e os principais desafios são destacados. Finalmente, são apresentados a justificativa do tema, o escopo da dissertação, a infraestrutura experimental, as principais contribuições científicas e a organização do texto.

1.1 MOTIVAÇÃO

A evolução da tecnologia CMOS (*Complementary Metal-Oxide Semiconductor*), com a consequente redução das dimensões dos componentes, possibilitou a integração de bilhões de transistores em um único *chip*. Tal capacidade de integração foi vital para viabilizar o desenvolvimento dos dispositivos móveis pessoais (PMDs - *Personal Mobile Devices*) contemporâneos, tais como *smartphones*, *tablets*, consoles de jogos etc, os quais oferecem um número impressionante de funcionalidades. Devido ao fato de serem alimentados por bateria e, ao mesmo tempo, terem que executar aplicações computacionalmente intensivas (e.g., codificação e decodificação de fotos e vídeos), os PMDs precisam aliar alto desempenho a baixo consumo de energia e ainda oferecer ao usuário serviços de qualidade. Ademais, o consumo de energia se tornou uma métrica importante em microprocessadores de propósito geral, servidores e supercomputadores, conforme ressaltado por Ozdal, Burns e Hu (2011).

Desde o início da década de 1990 o consumo de energia em circuitos integrados CMOS tem sido objeto de intensa investigação[1]. Contudo, para os nodos tecnológicos anteriores a 90 nm, a componente estática da potência era negligenciável perante a componente dinâmica. Foi a partir das tecnologias nanométricas (90 nm e mais recentes), caracterizadas por baixas tensões de alimentação (V_{dd}) e de *threshold*, e pelo uso de óxido de *gate* extremamente fino, que a componente estática passou a ser importante. Por isso, no projeto de circuitos CMOS com tecnologias nanométricas ambas componentes da potência devem ser minimizadas. No caso específico dos *chips* que equipam

[1] Um dos primeiros trabalhos de destaque foi o de Chandrakasan, Sheng e Brodersen (1992).

os PMDs, tal providência é essencial para prolongar a vida útil da bateria.

A dissipação de potência estática está associada às chamadas correntes de *leakage* (fuga) do transistor MOS. A Figura 1, apresentada por Rabaey (2009), identifica as principais correntes de *leakage* na estrutrura de um transistor MOS em tecnologia nanométrica, quais sejam:

- **Sub-threshold leakage**: corrente de fuga entre *source* e *drain* quando o transistor MOS está operando abaixo de sua tensão de *threshold* (ou seja, o transistor não está completamente desligado). Esta região de operação recebe o nome de inversão fraca (*weak inversion*).
- **Gate leakage**: corrente de fuga que flui do *gate* para o substrato do transistor através do óxido devido aos efeitos de tunelamento.
- **Junction leakage**: corrente de fuga que flui do *source* para o substrato e do *drain* para o substrato devido às junções *pn* reversamente polarizadas.

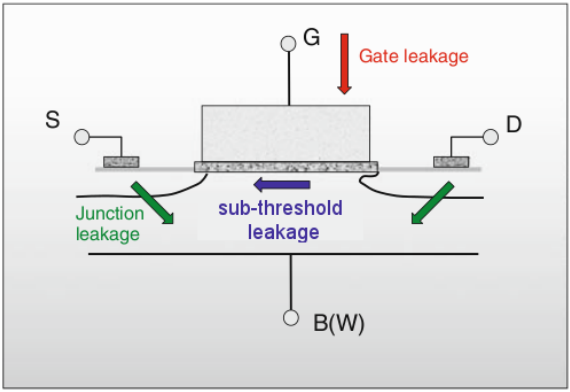


Figura 1: Principais componentes de corrente de *leakage* em um transistor MOS em tecnologia nanométrica. Adaptado de Rabaey (2009).

De acordo com Rabaey (2009), as duas primeiras componentes excedem a terceira (*junction leakage*) em 3 a 5 ordens de magnitude e por isso, são bem mais importantes.

Com relação à contribuição da potência estática ao longo da evolução da tecnologia CMOS, as primeiras projeções eram bastante alarmantes e acabaram induzindo a indústria e a academia a buscar técnicas para reduzir as correntes de *leakage* e, consequentemente, diminuir o consumo estático dos circuitos. A Figura 2 apresenta um gráfico contendo projeções para as

potências dinâmica e estática em um *chip*, publicado no trabalho de Kim et al. (2003). Tais projeções foram baseadas em dados de 2002, normalizados em relação aos dados do ITRS (*International Technology Roadmap for Semiconductors*) de 2001 (conforme apresentado em Assoc. (2002)). Além disso, a curva de potência dinâmica assumiu a Lei de Moore[2] para estimar o aumento do número de transistores por *chip*.

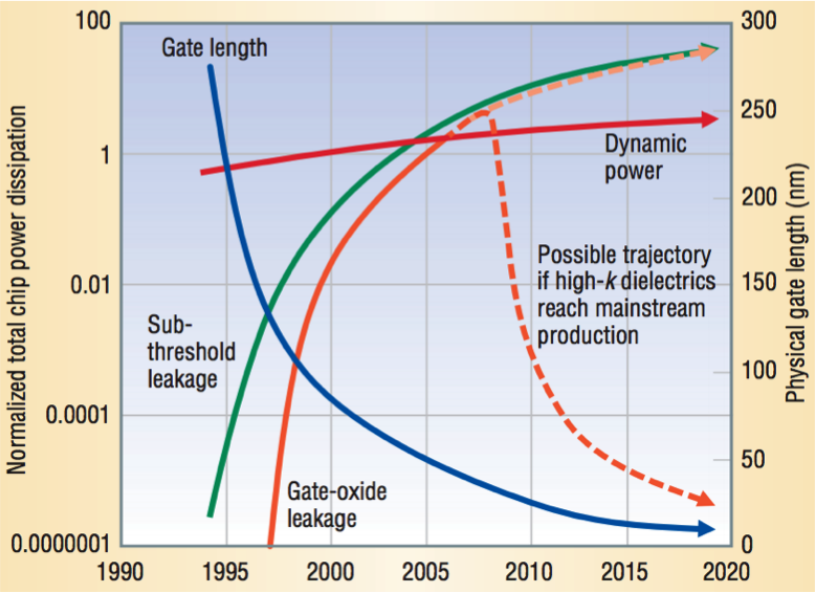


Figura 2: Projeções para as potências dinâmica e estática em um *chip*, baseadas no ITRS (*International Technology Roadmap for Semiconductors*), atualização de 2002. Fonte: Kim et al. (2003).

Também no ano de 2003, as projeções para a potência por porta lógica indicavam um decréscimo da componente dinâmica (em razão do *scaling* das dimensões dos transistores) e um incremento significativo da componente estática (decorrente do agravamento dos mecanismos de *leakage*), conforme ilustrado no gráfico da Figura 3, publicado por Sakurai (2003).

É interessante observar que, não obstante a diminuição da potência dinâmica por porta, a potência dinâmica do *chip* segue aumentando em consequência do acréscimo de transistores por *chip*, ainda que técnicas de projeto de baixa potência venham sendo aplicadas. Por um lado, a componente

[2]Em 1965 Gordon Moore publicou um artigo no qual especulava que o número de transistores em um circuito integrado dobraria a cada dois anos (MOORE, 1965).

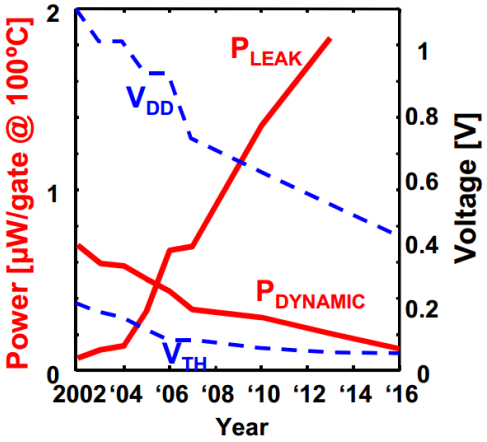


Figura 3: Projeções para as potências dinâmica e estática em uma porta. Nesta figura V_{dd} é a tensão de alimentação e V_{th} é a tensão de *threshold*. Fonte: (SAKURAI, 2003).

estática não aumentou tanto a ponto de ultrapassar a componente dinâmica. Segundo Weste e Harris (2010), em 2010 a potência estática podia corresponder à terça parte da potência total do *chip*, o que não deixa de ser uma parcela significativa. Esta diferença entre a projeção de 2003 e a realidade deveu-se às alterações das técnicas de fabricação e de projeto dos circuitos. Por exemplo, a introdução da técnica de fabricação de transistores com *gate* metálico e isolante com alta constante dielétrica (*high-k*), a partir de 2007, proporcionou uma redução drástica na componente de *gate leakage*[3], conforme previsto no gráfico da Figura 2. Por outro lado, as sucessivas reduções das tensões de alimentação e de *threshold* promovidas para viabilizar as tecnologias nanométricas mais recentes causaram o aumento da componente de *sub-threshold leakage* previsto no gráfico da Figura 2, transformando-a na principal componente da potência estática.

A Equação 1.1, obtida de Keating et al. (2007), apresenta uma boa aproximação da corrente de *sub-threshold* em um transistor MOS. Note que a corrente de *sub-threshold* tem dependência linear com a largura (W) do transistor (e consequentemente da porta) e dependência exponencial em relação à tensão de *threshold* do transistor, representada na Equação 1.1 por V_{th}.

[3]De acordo com Rabaey (2009), fabricantes como a IBM e Intel adotaram o uso de *dióxido de hafnium* (HfO_2) como material dielétrico a partir dos processos CMOS de 45nm (AUTH et al., 2008) e 32nm (NATARAJAN et al., 2008).

$$I_{sub} = \mu C_{ox} V_t^2 \frac{W}{L} . e^{\frac{V_{gs} - V_{th}}{nV_t}} \quad (1.1)$$

Dentre as técnicas de projeto empregadas para reduzir a corrente de *sub-threshold leakage* nos circuitos digitais contemporâneos destacam-se o uso de diferentes tensões de alimentação (*Multi-*V_{dd}), *power gating*, *gate sizing* (dimensionamento de portas) e o uso de diferentes tensões de *threshold* (*Multi-*V_t) (KEATING et al., 2007). Por serem minimamente intrusivas (pois implicam em um menor impacto no fluxo de projeto de circuitos digitais) as duas últimas são amplamente usadas e, por essa razão, são objeto de estudo do presente trabalho. Dimensionamento de portas (ou transistores) implica no uso de portas (ou transistores) com diferentes larguras de canal em um mesmo circuito, resultando em um impacto linear na corrente de *sub-threshold*. *Multi-*V_t, por sua vez, baseia-se na uso de portas (ou transistores) com diferentes tensões de *threshold*, o que causa impacto exponencial na corrente de *leakage* de *sub-threshold*.

Dada a importância de se minimizar a potência estática, principalmente em dispositivos móveis, cuja principal componente está associada à *sub-threshold leakage*, este trabalho tem como foco a minimização de *leakage*[4] em circuitos digitais.

1.2 FLUXO DE PROJETO DE CIRCUITOS DIGITAIS

O grande número de portas lógicas nos circuitos digitais contemporâneos exige a adoção de um fluxo de projeto com ferramentas de automação de projeto eletrônico (*EDA - Electronic Design Automation*). A grande maioria dos projetos digitais usa o estilo *semi-custom*[5], o qual se baseia em biblioteca *standard cell*. Uma biblioteca *standard cell* reúne os leiautes pré-projetados de portas lógicas, *flip-flops*, *latches* e eventualmente, outros elementos mais complexos (e.g., multiplexador 2-1, *full-adder* etc), referidos por células. Uma biblioteca *standard cell* também contém diversas informações referentes a cada célula como informações geométricas (e.g., leiaute, dimensões), de atraso, de potência, bem como outras características elétricas importantes para garantir o sucesso da síntese do circuito. Desta forma, o uso de biblio-

[4] Deste ponto em diante, o termo *leakage* será utilizado para fazer referência ao somatório das três componentes de *leakage*.

[5] O estilo de projeto *full-custom*, por outro lado, é usado principalmente no projeto de microprocessadores e FPGAs, uma vez que o alto custo destes projetos pode ser amortizado com um grande volume de produção (KAHNG et al., 2011).

teca *standard cell* visa reduzir o esforço de projeto e o tempo para o mercado (*time-to-market*)[6] e, consequentemente, o preço final do produto (KAHNG et al., 2011).

O fluxo de projeto de um circuito integrado digital é composto por diversas etapas, partindo da especificação e descrição em alto-nível de abstração até chegar a uma descrição que permita a implementação física (etapas de baixo-nível). As etapas de alto-nível são responsáveis principalmente pela especificação das funções e requisitos do sistema (e.g., potência, desempenho, área, entre outros), definição da arquitetura básica do sistema (e.g., tamanho das memórias, definição dos blocos de propriedade intelectual (IP), etc). Em seguida, os diferentes módulos do sistema são descritos em nível de transferência de registradores (RTL) através do uso de linguagens de descrição de *hardware* (HDL) apropriadas, tais como *Verilog* e *VHDL*.

A Figura 4 apresenta as etapas que caracterizam o fluxo de projeto baseado em *standard cell*, iniciando pela síntese lógica e passando pelas etapas relacionadas à chamada síntese física. A seguir, são descritas as ações realizadas em cada uma das etapas:

- **Síntese Lógica (*Logic Synthesis*)**: Etapa responsável pela conversão da descrição RTL para um conjunto de portas lógicas e elementos sequenciais e mapeamento deste conjunto para as células disponíveis na biblioteca *standard cell*;

- **Planejamento Topológico (*Floorplanning*)**: Determina o formato (leiaute) dos blocos do *chip* a partir das células instanciadas na etapa anterior, baseando-se nas informações da biblioteca *standard cell*. Também são determinadas as dimensões do *chip*, localização das portas de entrada e saída etc;

- **Posicionamento (*Placement*)**: Define a localização espacial dos leiautes das células que foram escolhidos na etapa anterior;

- **Síntese da Árvore de Relógio (*Clock Tree Synthesis*)**: Cria a árvore de distribuição do sinal de relógio para os elementos sequenciais do projeto, visando atender alguns requisitos, como, por exemplo, minimizar o *skew*[7];

- **Roteamento (*Routing*)**: Responsável por criar as conexões entre as células instanciadas no projetos, utilizando diferentes camadas de metal e *vias*. O objetivo é satisfazer as restrições de *timing* do projeto e, ao mesmo tempo, minimizar o comprimento médio das conexões;

[6] Corresponde ao tempo entre a concepção e a comercialização do produto.

[7] Diferença do tempo de chegada do sinal de relógio entre os elementos sequenciais.

- **Sign-off**: Esta etapa tem como principal objetivo a verificação das regras de desenho e da funcionalidade. Ela também confere se as restrições de máximo *slew*, máxima capacitância e máximo *fanout* impostas pela biblioteca *standard cell* estão dentro dos limites especificados[8].

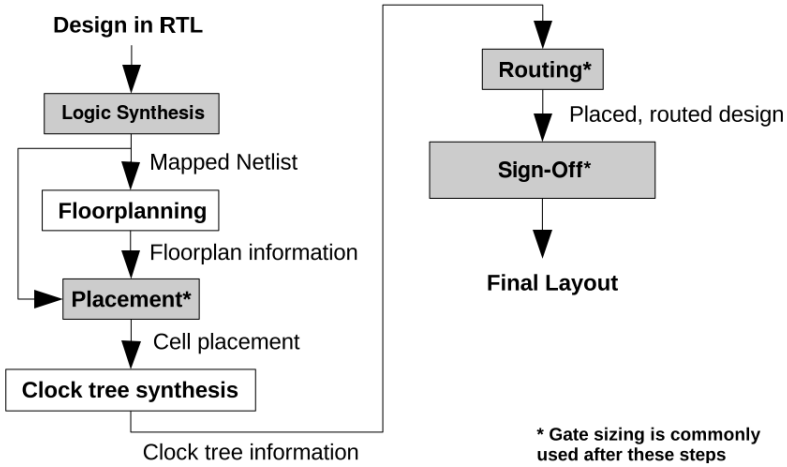


Figura 4: Fluxo simplificado de projeto de circuitos digitais baseado em *standard cell*. Adaptado de: (LEE; GUPTA, 2012).

Diversas técnicas de otimização são utilizadas durante o fluxo de projeto para diferentes objetivos, tais como minimizar *skew* do sinal de relógio, minimizar área, minimizar atraso, minimizar potência, dentre outros. Porém, dentre todas as técnicas, *gate sizing* está entre as mais utilizadas, pois além de ser extremamente efetiva, causa um impacto muito pequeno no fluxo de projeto (LEE; GUPTA, 2012). Assim, *gate sizing* é usada em diversas etapas do fluxo para corrigir problemas de *setup* e *hold* de elementos sequenciais, reduzir atrasos (para satisfazer a restrições de *timing*), dentre outros. Recentemente, *gate sizing* passou a ser amplamente utilizada para buscar um melhor compromisso (*tradeoff*) entre *leakage* e atraso do circuito (ABRISHAMI et al., 2011).

[8]O nome da subetapa responsável por essas conferências é *ERC - Electrical Rule Checking* (KAHNG et al., 2011).

1.3 O PROBLEMA DE *SIZING* DISCRETO

Gate sizing, ou somente *sizing*, é uma técnica amplamente usada na otimização de circuitos digitais, conforme já descrito na seção anterior. O problema (ou técnica) de *sizing* consiste na escolha dos parâmetros das portas lógicas, tais como largura (w) e tensão de *threshold* (u), visando otimizar desempenho, potência ou área do circuito, e ao mesmo tempo respeitar as restrições especificadas no projeto[9] (GUPTA et al., 2010) (KAHNG; KANG, 2012).

O problema de *sizing* de circutos digitais pode ser sub-dividido em *sizing* contínuo e *sizing* discreto. O primeiro caso considera que a largura e a tensão de *threshold* das portas são contínuas e portanto, podem assumir quaisquer valores dentro de um intervalo válido. O segundo, por outro lado, considera que a largura e a tensão de *threshold* das portas só podem assumir valores pertencentes a um conjunto pré-determinado, no caso, uma biblioteca de células[10].

Devido à sua importância como técnica de otimização no fluxo de projeto de circuitos digitais, o problema de *sizing* é um tópico extensivamente pesquisado desde meados da década de 1980 e uma grande variedade de técnicas pode ser encontrada na literatura. A grande maioria destas técnicas utiliza *sizing* contínuo (e.g., Fishburn e Dunlop (1985) Chen, Chu e Wong (1999)), que não se correlaciona adequadamente com o fluxo de projeto baseado em *standard cell*, conforme será detalhado mais adiante.

De acordo com o que foi frisado na Seção 1.1, a minimização de *leakage* é essencial tanto no projeto de PMDs, quanto no projeto de microprocessadores de propósito geral. Para tais projetos, o problema de *sizing* tem dois objetivos conflitantes: minimizar o *leakage* e, ao mesmo tempo, reduzir o atraso para garantir que o desempenho do circuito satisfaça às especificações.

O problema de *sizing contínuo* para ajustar a largura das portas do circuito visando minimizar o consumo de *leakage* pode ser formalmente definido pelo conjunto de Equações 1.2 a 1.4 (MAHESHWARI; SAPATNEKAR, 1998). O objetivo do problema consiste em encontrar a largura das portas do circuito ($\vec{w}$) para minimizar o consumo total de *leakage* (Equação 1.2) e

[9] A escolha dos parâmetros pode ocorrer em diversas granularidades como, por exemplo, no nível de transistor, modificando a largura de cada transistor independentemente dos demais, ou no nível de porta lógica, na qual todos os transistores da porta são modificados simultaneamente. Isto vale também para atribuição da tensão de *threshold*. Em projetos baseados em bibliotecas *standard cell*, a granularidade é definida no nível de porta e somente as opções disponíveis na biblioteca podem ser utilizadas.

[10] Alguns trabalhos encontrados na literatura denominam este problema como seleção da opção de implementação das portas (*gate implementation selection*) (LI et al., 2012) (HUANG; HU; SHI, 2011).

garantir que o atraso máximo do circuito não seja maior que o período de relógio especificado (Equação 1.3). Além disso, a restrição representada pela Equação 1.4 especifica a largura mínima e a largura máxima de uma porta permitida pela tecnologia utilizada. Note que *leakage* e atraso são objetivos conflitantes, ou seja, a redução da largura de uma porta resulta em menor *leakage* mas em contrapartida, acarreta em aumento do atraso da porta, o que pode tornar o circuito mais lento.

$$\textbf{Minimize} \quad : \quad Power(\vec{w}) \tag{1.2}$$

$$\textbf{Sujeito a} \quad : \quad Delay(\vec{w}) \leq A_o \tag{1.3}$$

$$W_{min} \leq w_{gate} \leq W_{max}, \forall gate \in Circuit \tag{1.4}$$

Por um lado, a abordagem contínua do problema de *sizing* permite o uso de técnicas consolidadas, nas quais a solução ótima do problema pode ser encontrada[11]. Por outro lado, o projeto de circuitos digitais modernos é baseado em bibliotecas *standard cell*, as quais são compostas por um número limitado de opções de implementação por porta. Poucos trabalhos de *sizing* encontrados na literatura resolvem o problema diretamente no domínio discreto (e.g., Coudert (1997), Li et al. (2012), Ozdal, Burns e Hu (2012)), o qual é provado ser NP-difícil[12] (LI, 1994) e, desse modo, é necessário o uso de heurísticas eficientes para encontrar soluções de qualidade dentro de um tempo de execução viável. A formulação matemática do problema de *sizing* discreto para minimizar *leakage* é apresentada nas Equações 1.5 a 1.7. Note que o parâmetro de configuração (largura w_{gate}) de cada porta é limitado às opções discretas W_{gate}, onde cada porta possui um conjunto de opções de implementação disponível na biblioteca[13].

[11] Abordagens contínuas permitem o uso de modelos convexos de atraso, os quais tem como vantagem permitir que se averigue a qualidade da solução, já que possuem a propriedade de que todo mínimo local da função é também um mínimo global. Em outras palavras, é possível saber quão longe do mínimo global do problema se está e obter um certificado de qualidade da solução encontrada (BOYD; VANDENBERGHE, 2004).

[12] Refere-se à classe de problemas cuja solução ótima não pode ser encontrada em tempo polinomial em relação ao tamanho da entrada do problema. Ou seja, problemas cuja complexidade de tempo não pode ser definida por $O(n^k)$, onde n é o tamanho da entrada do problema e k uma constante (CORMEN et al., 2009).

[13] No caso da inclusão da possibilidade da escolha da tensão de *threshold* de cada porta, deve-se incluir a restrição $u_{gate} \in U_{gate}$, a qual define que a porta só pode assumir uma das tensões de *threshold* disponíveis na biblioteca.

$$\textbf{Minimize} \quad : \quad Power(\vec{w}) \tag{1.5}$$

$$\textbf{Sujeito a} \quad : \quad Delay(\vec{w}) \leq A_o \tag{1.6}$$

$$w_{gate} \in W_{gate}, \forall gate \in Circuit \tag{1.7}$$

Ademais, o problema de *sizing* discreto apresenta diversos desafios quando aplicado ao projeto de circuitos industriais contemporâneos. Segundo Ozdal et al. (2012), os principais desafios são:

- **Parâmetros discretos**: Como o projeto de circuitos digitais contemporâneos segue a metodologia de projeto *standard cell*, o uso de técnicas de *sizing* que assumem os parâmetros contínuos requer o mapeamento das soluções encontradas para as disponíveis na biblioteca de células, o que pode levar a soluções subótimas;

- **Modelos de atraso das portas**: O grande número de parâmetros de uma porta e uso de técnicas de leiaute específicas, como *transistor folding* (CHINNERY; KEUTZER, 2005), resulta em modelos de atraso complexos. Como consequência, nem mesmo modelos de atraso convexos são suficientemente precisos, visto que o atraso de uma porta não é função convexa[14] de sua largura, tampouco da tensão de *threshold*;

- **Restrições temporais complexas**: Circuitos modernos apresentam vários domínios de relógio, falsos caminhos, etc. Além disso, o atraso das interconexões é cada vez mais significativo em nodos tecnológicos nanométricos e portanto, o uso de modelos simplificados (e.g., *Elmore* (ELMORE, 1948)) não provê a precisão requerida pelos projetos contemporâneos;

- **Efeitos do *slew***: O atraso das portas em uma biblioteca de células é função da capacitância de saída e do *slew* de entrada da porta. Portanto, é essencial considerar o *slew* durante o processo de otimização. Além disso, há restrições de máximo *slew* que devem ser consideradas;

- **Escalabilidade das técnicas de otimização**: Alguns blocos dentro de um circuito digital podem ter centenas de milhares (ou alguns milhões) de portas e portanto, as técnicas de *sizing* devem ser escaláveis para lidar com circuitos da ordem de milhões de portas.

[14] Uma função convexa possui a propriedade de que todo mínimo local da função é também um mínimo global (BOYD; VANDENBERGHE, 2004).

1.4 JUSTIFICATIVA

Conforme mencionado na seção anterior, o problema de *sizing* discreto é NP-difícil (LI, 1994). Por este motivo, as pesquisas neste tema têm se concentrado na busca por heurísticas eficientes. Recentemente, o interesse por *sizing* discreto intensificou-se tanto por parte da academia quanto da indústria, na medida em que ele emergiu como uma opção efetiva para redução de *leakage* em circuitos nanométricos projetados com o fluxo *standard cell*. Prova deste interesse é o grande número de artigos científicos sobre o tema que tem sido publicados recentemente nos anais de conferências importantes na área de EDA, tais como ICCAD[15] (IEEE/ACM International Conference on Computer-Aided Design), DAC (ACM/EDAC/IEEE Design Automation Conference), DATE (Design, Automation & Test in Europe), ISLPED (International Symposium on Low Power Electronics and Design), ISPD (ACM/SIGDA International Symposium on Physical Design), com autoria de membros da academia e da indústria.

Um dos problemas enfrentados pelos pesquisadores reside na ausência de uma infraestrutura comum que permita comparar justamente um conjunto de soluções para um determinado problema. Para o problema em questão, tal infraestrutura deve ser composta por um conjunto de *benchmarks* e uma biblioteca *standard cell* realistas que capturem os principais detalhes do problema. Neste sentido, o trabalho proposto por Gupta et al. (2010) tenta resolver tal problema, ao propor um método para a criação de uma infraestrutura de *benchmarks* que permita encontrar a solução ótima de cada circuito. Infelizmente, este trabalho apresenta duas limitações sérias, quais sejam, o número reduzido de portas dos circuitos (em média 10K), e o modelo de atraso simplificado, o qual não considera *slew*. Por outro lado, a falta de uma solução definitiva para o problema de *sizing* discreto e a ausência de uma infraestrutura de comparação adequada motivou os pesquisadores da Intel a organizarem a Competição de *Sizing* Discreto (OZDAL et al., 2012), realizada na mais recente edição do ISPD, em 2012. Tal infraestrutura permite comparar diferentes técnicas de *sizing* discreto sob uma infraestrutura realista que captura a maior parte dos desafios listados na seção anterior.

Como em outras competições (e.g., Competição de roteamento realizada pelo ISPD nos anos de 2007 e 2008), a disponibilidade de *benchmarks* realistas proporciona a oportunidade de confrontar diferentes técnicas sob uma mesma infraestrutura. Desde a Competição de *Sizing* Discreto do ISPD 2012 (março 2012), a qual proporcionou os resultados das equipes partici-

[15] No ICCAD do ano de 2011, o artigo de *sizing* discreto de pesquisadores da Intel (OZDAL; BURNS; HU, 2011) ganhou o prêmio de melhor artigo da conferência.

pantes, novos trabalhos foram publicados (e.g., Hu et al. (2012), Li et al. (2012), Livramento et al. (2013)), os quais fizeram uso da infraestrutura disponibilizada. Tais trabalhos ampliaram o conhecimento sobre como enfrentar o problema de forma mais eficiente, e por isso obtiveram valores de *leakage* e tempos de execução ainda menores do que aqueles apresentados pelos competidores.

Por outro lado, apesar de tal evolução, ainda não há um consenso sobre qual técnica de *sizing* discreto é a melhor, visto que nenhuma das técnicas apresentadas até o momento mostrou-se predominante. Por este motivo, a pesquisa nesse tema continua intensa. O mesmo grupo de pesquisadores da Intel está organizando uma segunda edição da Competição de *sizing* discreto, associada ao ISPD do corrente ano (2013).

1.5 ESCOPO DESTA DISSERTAÇÃO

Uma vez que o projeto de circuitos digitais segue o fluxo *standard cell*, é essencial que as técnicas de *sizing* trabalhem no domínio discreto, ao invés de considerar larguras e/ou tensões de *threshold* dentro de um intervalo contínuo. Ademais, as técnicas que abordam o problema no domínio contínuo fazem uso de modelos de atraso simplificados, os quais não modelam com precisão o atraso das portas das bibliotecas *standard cell* (OZDAL; BURNS; HU, 2012). O escopo deste trabalho é bastante semelhante ao definido na infraestrutura da Competição de *Sizing* Discreto do ISPD 2012 (OZDAL et al., 2012), visto que os principais desafios de *sizing* no fluxo de projeto *standard cell* são capturados, permitindo a comparação direta e justa de diferentes técnicas.

Este trabalho aborda o problema de *sizing* diretamente no domínio discreto, objetivando minimizar a potência de *leakage*, enquanto considerando restrições de *timing* definidas por um atraso crítico-alvo. Tal minimização é feita escolhendo, para cada porta lógica combinacional do circuito, uma opção de implementação (combinando uma opção de largura e uma opção de tensão de *threshold*) disponível em bibliotecas *standard cell*. Os modelos de atraso e *slew* aqui utilizados estão capturados diretamente pelas tabelas associadas à biblioteca de células[16], os quais são funcão da capacitância de saída e do *slew* de entrada da porta. Neste trabalho são consideradas as restrições de máxima capacitância de saída das portas e máximo *slew* na entrada das portas, ambas comumente definidas nas bibliotecas *standard cell* contemporâneas. As interconexões do circuito são modeladas como capacitâncias

[16] Mais detalhes sobre as tabelas de atraso das bibliotecas de células são apresentados na Seção 2.3.

concentradas (*lumped*) sem resistência, ou seja, sem atraso.

Por questões de infraestrutura, não faz parte do escopo dessa dissertação a escolha da largura e da tensão de *threshold* de elementos sequenciais, como *flip-flops* e registradores, visto que no fluxo de projeto digital esta escolha se restringe aos elementos combinacionais[17] (SAPATNEKAR, 2004). Tampouco se considera tempo de *setup* e *hold* desses elementos.

1.6 INFRAESTRUTURA EXPERIMENTAL PARA ESTE TRABALHO

O presente trabalho faz uso da infraestrutura disponibilizada pela Competição de *Sizing* Discreto do ISPD 2012, a qual é composta por:

- Um conjunto de 7 circuitos, cujos tamanhos variam de 25K a 959K portas, cada circuito sujeito a duas restrições diferentes de atraso crítico-alvo, indentificadas como *slow* e fast;
- Uma biblioteca *standard cell* realista, composta por 11 portas combinacionais de funções lógicas diferentes e 1 flip-flop;
- Ferramenta de análise de *timing* estática Synopsys PrimeTime® (SYNOPSYS, 2012)[18];
- Um conjunto de *scripts* para validar os resultados finais de *timing*, slew e capacitância, os quais invocam a ferramenta Synopsys PrimeTime®. Estes *scripts* calculam o total de violações de máximo *slew* na entrada das portas e de máxima capacitância de saída das portas, além do consumo total de *leakage* do circuito.

Os circuitos são derivados dos *benchmarks* do IWLS 2005 (SYNTHESIS, 2012) e cada circuito inclui uma descrição na linguagem Verilog, um arquivo no formato IEEE SPEF (*Standard Parasitic Exchange Format*) descrevendo as capacitâncias parasitas das interconexões, além das restrições de *timing* no formato SDC (*Synopsys Design Constraints*).

A biblioteca de células é definida conforme o padrão industrial *Liberty* (LIBERTY, 2012), no qual cada porta possui tabelas (*lookup tables*) contendo informações de atraso, tempo de transição do sinal (*slew*) de saída, máxima capacitância de saída e potência de *leakage*. Cada uma das 11 portas combinacionais da biblioteca possui 3 opções de tensão de *threshold* e 10 opções de

[17] Diversas técnicas podem ser utilizadas simultaneamente com a técnica de *sizing* para otimização dos elementos sequenciais, tais como *retiming* (MAHESHWARI; SAPATNEKAR, 1998) e otimização de *skew* (CHUANG; SAPATNEKAR; HAJJ, 1993) (ROY et al., 2008a).

[18] Acesso (restrito), mediante assinatura de termo de compromisso (NDA - *Non-Disclosure Agreement*).

largura, totalizando 30 opções de implementação para cada porta combinacional. Já o *flip-flop* possui apenas uma opção de implementação. Além disso, a biblioteca também inclui um limite global (restrição) de máximo *slew* nas entradas/saídas das portas e do circuito.

Foram assumidas algumas simplificações no problema, como modelo de capacitância concentrada e atraso nulo para as interconexões. Também os tempos de *setup* e *hold* dos registradores foram desconsiderados.

As comparações entre as diferentes técnicas são feitas usando os resultados de *leakage* sem violações. As violações consideradas são:

- Ocorrência de *slack* negativo (i.e., atraso crítico menor ou igual ao atraso crítico-alvo);
- Ocorrência de *slews* acima do limite especificado pela biblioteca (no caso da biblioteca da Competição, o valor é 300ps);
- Ocorrência de uma ou mais portas apresentando capacitância de saída maior do que aquela especificada na biblioteca.

As violações de *timing* na Competição do ISPD 2012 são calculadas através do somatório dos *slacks* negativos nas saídas primárias do circuito, definido como TNS (*Total Negative Slack*). As demais violações são calculadas através do somatório da quantidade que ultrapassa os limites especificados na biblioteca *standard cell*.

Para cálculo das informações de *timing* do circuito durante a execução das técnicas de otimização propostas nesse trabalho, foi implementada uma ferramenta de análise de *timing* estática em conformidade com bibliotecas *standard cell* realistas, a qual foi validada frente à ferramenta industrial Synopsys PrimeTime®(SYNOPSYS, 2012). Esta ferramenta foi utilizada para cálculo das informações de *timing* durante a execução dos algoritmos presentes nesta dissertação, enquanto a ferramenta Synopsys PrimeTime®(SYNOPSYS, 2012) foi utilizada somente com propósito de validação dos resultados, conforme definido na Competição do ISPD 2012. Os resultados experimentais apresentados nessa dissertação, nos Capítulos 5 e 6, foram executados em uma máquina com 2 CPUs Intel®Xeon®E5620 @ 2.4GHz com 12GB RAM.

Com o intuito de facilitar a compreensão dos experimentos realizados nesse trabalho, a Tabela 1 resume a infraestutura dos experimentos apresentados nos Capítulos 5 e 6.

Tabela 1: Resumo da infraestrutura utilizada nos Capítulos 5 e 6.

Resumo da Infraestrutura	Capítulo 5	Capítulo 6
Biblioteca da Competição ISPD 2012	X	X
***Benchmarks* da Competição ISPD 2012**	X	X
Synopsys PrimeTime®	X	X
Scripts da Competição ISPD 2012	X	X

1.7 PRINCIPAIS CONTRIBUIÇÕES

O presente trabalho apresenta como principais contribuições científicas:

- Uma nova formulação do problema de *sizing* discreto utilizando a técnica de Relaxação Lagrangeana (LR - *Lagrangian Relaxation*). As novidades da formulação LR proposta residem na incorporação das restrições de máxima capacitância e máximo *slew* — impostas pelas bibliotecas *standard cell* — na função objetivo (em adição às restrições de *timing* usualmente relaxadas). O trabalho associado a esta contribuição, incluindo os resultados obtidos, foi apresentado oralmente e publicado nos anais da *IEEE International Conference on Electronics, Circuits and Systems* — ICECS 2012 (LIVRAMENTO et al., 2012b);

- Uma heurística gulosa para resolver a formulação LR proposta (conforme item anterior), a qual baseia-se em informações locais para guiar as decisões. Um artigo relatando esta heurística e os resultados alcançados foi apresentado oralmente e publicado nos anais da conferência *Design, Automation & Test in Europe* — DATE 2013 (LIVRAMENTO et al., 2013);

- Uma técnica híbrida para *sizing* discreto que tem por objetivo contornar as limitações resultantes do uso exclusivo de LR. Um artigo para periódico com esta contribuição e com os resultados obtidos está em fase de preparação.

As contribuições específicas referentes a cada um dos artigos são apresentadas nos Capítulos 4, 5 e 6, respectivamente.

1.8 ORGANIZAÇÃO DESSA DISSERTAÇÃO

Esta dissertação está organizada da seguinte forma.

O Capítulo 2 apresenta os conceitos fundamentais necessários para o entendimento desta dissertação, tornando-a auto-contida.

No Capítulo 3 são apresentadas as diferentes técnicas para resolver o problema de *sizing* em circuitos digitais, com especial atenção àquelas mais recentes que abordam o problema diretamente no dominínio discreto (*sizing* discreto).

Já o Capítulo 4 detalha o trabalho Livramento et al. (2012b), apresentando a formulação proposta para o problema-alvo, a qual é baseada em Relaxação Lagrangeana.

O Capítulo 5 descreve o trabalho Livramento et al. (2013), o qual propõe uma heurística gulosa baseada em Relaxação Lagrangeana para resolver o problema-alvo e apresenta resultados experimentais.

No Capítulo 6 é apresentada uma técnica híbrida para resolver o problema-alvo, bem como os resultados experimentais obtidos. Tal técnica híbrida será descrita em artigo para periódico, em preparação.

Finalmente, as conclusões e perspectivas de trabalhos futuros são apresentados no Capítulo 7.

2 CONCEITOS FUNDAMENTAIS

Este capítulo apresenta a terminologia associada à estrutura e às características temporais dos circuitos digitais, além de uma breve introdução à Relaxação Lagrangeana, conceitos necessários para a compreensão das técnicas de minimização de *leakage* descritas nos Capítulos 4, 5 e 6.

2.1 MODELAGEM DA ESTRUTURA DOS CIRCUITOS DIGITAIS

Atualmente, a metodologia de projeto digital predominante baseia-se no modelo de circuitos sequenciais síncronos (RABAEY; CHANDRAKASAN; NIKOLIC, 2003), nos quais o sincronismo é proporcionado por um sinal cíclico e monótono denominado relógio (*clock*). Os circuitos sequenciais síncronos são compostos por portas lógicas, elementos de armazenamento, os quais podem ser *flip-flops* ou *latches*, e conexões (fios). O modelo mais simples utiliza somente um sinal de relógio para sincronizar todos os elementos de armazenamento, ao passo que modelos mais sofisticados utilizam dois ou mais sinais divididos em diferentes regiões do circuito. Neste último caso, é possível dividir o circuito em partes que operem em frequências de relógio diferentes visando reduzir o consumo de energia.

Os elementos sequenciais criam barreiras temporais, uma vez que capturam os sinais que estão em suas entradas de dados no momento em que são ativados pelo sinal de relógio. Assim, em um circuito sequencial é possível identificar blocos puramente combinacionais (i.e., compostos apenas por portas lógicas e conexões, e sem realimentações), mediante a identificação dos elementos de armazenamento de entrada e dos elementos de armazenamento de saída. Em outras palavras, um circuito sequencial síncrono pode ser visto como um conjunto de blocos combinacionais interligados por elementos de armazenamento. Tal estrutura provê uma decomposição natural do circuito, a qual é explorada para viabilizar a manipulação de projetos com grande número de componentes. Além disso, a decomposição em blocos combinacionais permite a aplicação de técnicas de otimização voltadas para circuitos combinacionais. Um exemplo seria a aplicação da técnica de *sizing* em cada bloco combinacional, conforme adotado pela maioria das técnicas encontradas na literatura.

A Figura 5 mostra uma porção de um circuito sequencial, onde é possível identificar um bloco combinacional e os respectivos *flip-flops* de entrada e de saída. Para tornar o exemplo suficientemente genérico, nem todas as entradas deste bloco combinacional provêm de *flip-flops*, da mesma forma

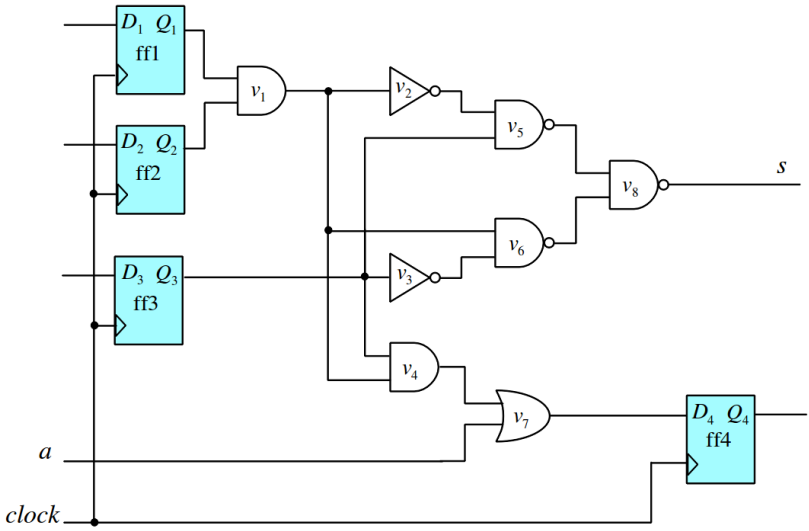


Figura 5: Exemplo de circuito combinacional.

que nem todas suas saídas alimentam *flip-flops*.

No contexto de técnicas de *sizing* e portanto, também no contexto da presente dissertação, os seguintes conceitos associados aos blocos combinacionais são essenciais:

- **Entradas primárias**, representado por PI, é o conjunto dos sinais que alimentam um bloco combinacional. Para o bloco combinacional da Figura 5, $PI = \{Q_1, Q_2, Q_3, a\}$;

- **Saídas primárias**, representado por PO, é o conjunto dos sinais gerados por um bloco combinacional. Para o bloco combinacional da Figura 5, $PO = \{s, D_4\}$;

- ***Fanin*** de uma porta v_i, representado por $fanin(v_i)$, é o conjunto das portas e/ou entrada primárias que estão diretamente conectadas às entradas de v_i. Por exemplo, no bloco combinacional da Figura 5, fanin(v_1) = $\{Q_1, Q_2\}$;

- ***Fanout*** de uma porta v_i, representado por $fanout(v_i)$, é o conjunto das portas e/ou saídas primárias que estão diretamente conectadas à saída de v_i. Por exemplo, no bloco combinacional da Figura 5, fanout(v_1) = $\{v_2, v_4, v_6\}$.

- **Entradas** de uma porta v_i, representado por $input(v_i)$, corresponde ao seu conjunto de pinos de entrada. Por exemplo, no bloco combinacional da Figura 5, a porta v_1 tem 2 pinos de entrada, ao passo que v_2 tem 1 pino de entrada.

- **Cone lógico** de uma porta v_i é o conjunto de portas que podem ser influenciadas pelo sinal gerado na saída de v_i. Isto inclui as portas que são *fanout* de v_i, as portas que são *fanout* destas últimas e assim por diante, até que as saídas primárias sejam atingidas. Além disso, as portas que são *fanin* de v_i também são influenciadas no caso da troca da opção de implementação de v_i por uma opção de largura diferente. Isto ocorre devido à variação da capacitância de saída das portas fanin de v_i.

As ferramentas de EDA costumam modelar cada bloco combinacional como um grafo acíclico direcionado (*Directed Acyclic Graph* - DAG) $G(V,E)$, onde V é o conjunto de nodos e E é o conjunto de arestas. Cada $v_i \in V$ representa ou uma porta ($v_i \in X$), ou uma entrada primária ($v_i \in PI$) ou uma saída primária ($v_i \in PO$), e desta forma, $V = X \cup PI \cup PO$. Adicionalmente, $e_{j,i} \in E$ representa uma conexão (fio) entre v_j e v_i. A Figura 6 mostra o DAG para o bloco combinacional da Figura 5. Observe que no DAG constuma-se incluir um nodo fonte (S - *Source*) e um nodo terminal (T - *Terminal*) para facilitar o processamento. As saídas do S estão conectadas às entradas primárias enquanto as entradas do T estão conectadas às saídas primárias (KAHNG et al., 2011).

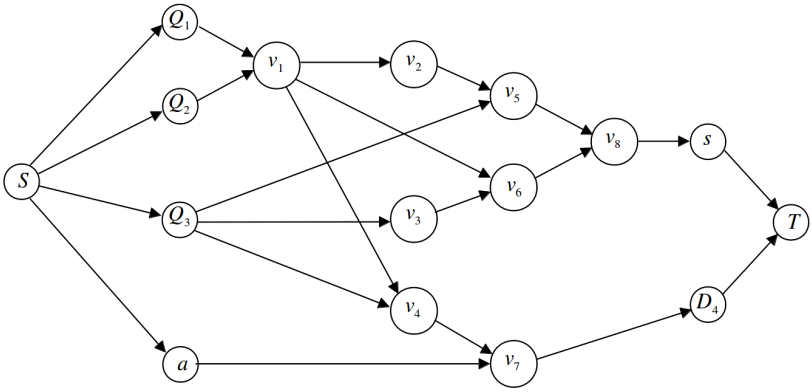


Figura 6: Grafo Acíclico Direcionado (DAG) do circuito combinacional apresentado na Figura 5.

2.2 CARACTERÍSTICAS DAS PORTAS E DOS *FLIP-FLOPS*

A fim de garantir o desempenho requerido para o projeto, tanto a síntese quanto a otimização precisam levar em conta as características temporais do circuito. Estas, por sua vez, são derivadas das características temporais e funcionais dos componentes básicos, tais como portas lógicas, elementos de armazenamento e conexões. Uma vez que o fluxo de síntese industrial está baseado no uso de *standard cells*, algumas das características explicadas a seguir estão relacionadas com informações que constam nas bibliotecas *standard cell*. No que se refere às portas lógicas, as seguintes características são relevantes:

- **Atraso de descida** da porta v_i com relação à sua entrada j, representado por $d^f_{j\to i}$, é o tempo transcorrido entre o sinal na entrada j de v_i atingir 50% de sua excursão total até o sinal na saída de v_i atingir 50% de sua excursão total, sendo que este último realiza uma transição de descida. Uma vez que os sinais internos aos circuitos digitais possuem um comportamento analógico, é necessário adotar-se alguma referência para a medida de atrasos. A referência de 50% da excursão total é um padrão amplamente adotado na indústria (BHASKER; CHADHA, 2009).

- **Atraso de subida** da porta v_i com relação à sua entrada j, representado por $d^r_{j\to i}$, é o tempo transcorrido entre o sinal na entrada j de v_i atingir 50% de sua excursão total até o sinal na saída de v_i atingir 50% de sua excursão, sendo que este último realiza uma transição de subida.

- ***Slew* de descida** da porta v_i em função da entrada j, representado por $slew^f_{j\to i}$, é o tempo transcorrido para que o sinal na saída de v_i realize uma transição de descida, medido entre uma tensão de referência[1] inicial (nível lógico 1) e uma tensão de referência final (nível lógico 0).

- ***Slew* de subida** da porta v_i em função da entrada j, representado por $slew^r_{j\to i}$, é o tempo transcorrido para que o sinal na saída de v_i realize uma transição de subida, medido entre uma tensão de referência inicial (nível lógico 0) e uma tensão de referência final (nível lógico 1).

- **Propagação do *slew*** define o método utilizado para escolher qual dos *slews* de descida $slew^f_{j\to i}$ (ou subida $slew^r_{j\to i}$) será propagado para a saída de v_i. Como uma porta lógica apresenta diferentes valores de

[1] Atualmente, diferentes tensões de referência inicial e final são utilizadas pelas bibliotecas, dependendo da tecnologia utilizada. Alguns exemplos de tensões de referência inicial e final são: $0.9V_{dd}$ e $0.1V_{dd}$, $0.8V_{dd}$ e $0.2V_{dd}$ ou $0.7V_{dd}$ e $0.3V_{dd}$ (BHASKER; CHADHA, 2009).

slew para cada combinação $j \rightarrow i$, diferentes métodos são utilizados. O método mais utilizado por ferramentas industriais de análise de *timing*, por ser pessimista, propaga o pior dos *slews* de entrada de descida (subida) para a saída da porta, sendo também o método adotado nessa dissertação. Aplicado este método, o *slew* de descida (subida) na saída de v_i é representado por $slew_i^f$ ($slew_i^r$).

- **Positive Unateness**: diz-se que uma porta v_i é *positive unate* quando uma transição de subida (descida) em uma de suas entradas resulta em uma transição de subida (descida) em sua saída. As portas AND e OR são exemplos de portas *positive unate*.

- **Negative Unateness**: diz-se que uma porta v_i é *negative unate* quando uma transição de subida (descida) em uma de suas entradas resulta em uma transição de descida (subida) em sua saída. As portas CMOS estáticas complementares são exemplos de portas *negative unate*. É interessante notar que as portas XOR não são *negative unate* tampouco *positive unate*, sendo denominadas *non-unate*.

Visto que a biblioteca *standard cell* adotada nos experimentos desse trabalho possui apenas portas CMOS estáticas complementares, ou seja portas *negative unate* (com execeção do *flip-flop*), por questão de didática, os algoritmos e fórmulas apresentados nessa dissertação assumem apenas portas *negative unate*.

A Figura 7 apresenta as características temporais de um inversor para ilustrar os conceitos apresentados anteriormente. É importante observar que, de uma maneira mais genérica, o conceito de *slew* está associado ao tempo de transição de qualquer sinal do circuito, inclusive dos sinais nas entradas e saídas dos elementos de armazenamento e nas entradas e saídas primárias do bloco combinacional. Também vale a pena observar que quando duas portas estão conectadas (por exemplo, saída de v_j conectada a uma das entradas de v_i), se as interconexões do circuito são modeladas com resistência zero (sem atraso), o *slew* de entrada de v_i corresponde ao *slew* de saída de v_j. Este é o caso assumido na infraestrutura da Competição do ISPD 2012, usada nesse trabalho. Os conceitos referentes às caracterísicas temporais dos elementos sequenciais, como tempo de *setup* e tempo de *hold*, não serão apresentados visto que na infraestrutura utilizada eles são desconsiderados e, portanto, não fazem parte do escopo dessa dissertação.

As ferramentas comerciais de análise de *timing* estática, cujo funcionamento será abordado na Seção 2.4, adotam o conceito de **arcos de tempo**[2]

[2]É interessante notar que o conceito de arcos de tempo, quando aplicado às portas, equivale ao chamado atraso pino-a-pino (*pin-to-pin delay* (GUNTZEL, 2000).

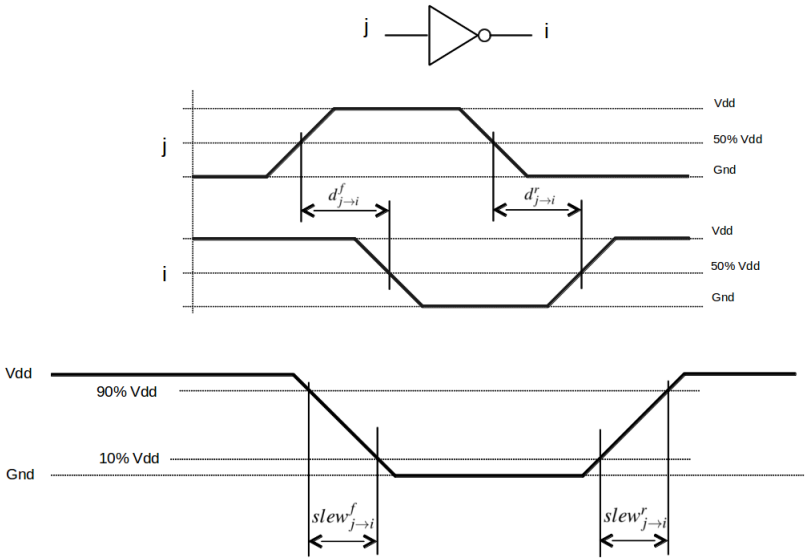
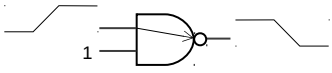


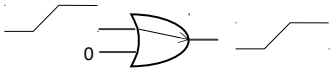
Figura 7: Características temporais de uma porta, ilustradas pelo comportamento de um inversor. A parte superior ilustra a definição de atraso de descida e subida. Já a parte inferior apresenta uma ilustração do *slew* de descida e subida, respectivamente.

(*timing arcs*) (BHASKER; CHADHA, 2009). Um arco de tempo serve para relacionar o atraso de propagação de um sinal entre dois pontos adjacentes do circuito (LEE; GUPTA, 2012). Assim, pode-se associar um arco de tempo para cada rede ou para cada fio do circuito (conexão entre o ponto de origem e um destino), conforme for o modelo de atraso de conexões adotado. Uma porta lógica v_i (e.g., NAND, NOR) possui um par de arcos de tempo (subida e descida) entre cada entrada j e a saída da porta. Já para um *flip-flop*, associa-se um arco entre a sua entrada de dados e a saída. Como consequência prática, o atraso de um caminho qualquer em um circuito digital ou em um de seus blocos combinacionais pode ser calculado como a soma dos arcos de tempo associados aos elementos atravessados pelo referido caminho.

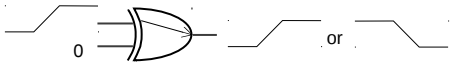
Para as portas lógicas, os arcos de tempo também são usados para associar o atraso da porta com o tipo de transição de entrada e o consequente tipo de transição na saída (segundo a propriedade *unateness* da porta). A Figura 8 ilustra tal associação, sem contudo identificar o valor do atraso.



(a) Negative unate arc



(b) Positive unate arc



(c) Non-unate arc

Figura 8: Associação entre *timing arc* e propriedade *unateness*. Adaptado de Bhasker e Chadha (2009).

2.3 MODELOS DE ATRASO E BIBLIOTECA *STANDARD CELL*

As técnicas de otimização de potência estática propostas neste trabalho estão voltadas ao fluxo de projeto *standard cell*. No fluxo *standard cell*, as informações de atraso, *slew* e potência das células (instância de uma porta lógica) da biblioteca são disponibilizadas em um arquivo no formato Liberty (LIBERTY, 2012). Até o início dos anos 2000 o modelo de atraso de célula utilizado pelo fluxo *standard cell* era o chamado modelo linear, cuja forma geral é mostrada na Equação 2.1 para uma porta *negative unate*. Nesta equação, $slew_j^r$ é o *slew* de subida na entrada j da célula, enquanto cap_i é a carga capacitiva na sua saída. Os demais valores, $D0$, $D1$ e $D2$, são constantes determinadas a partir de simulações no nível elétrico.

$$d_{j\to i}^f = D0 + D1 \times slew_j^r + D2 \times cap_i \tag{2.1}$$

O modelo linear não é suficientemente preciso para os intervalos de valores de *slew* de entrada e de cargas capacitivas de saída que comumente ocorrem em nodos tecnológicos nanométricos (BHASKER; CHADHA, 2009). Por esta razão, as bibliotecas *standard cell* contemporâneas utilizam o modelo de atraso não-linear (NLDM - *Non-Linear Delay Model*), o qual armazena as informações de atraso e *slew* de cada opção de implementação (célula) de cada função lógica em tabelas bi-dimensionais. Para cada entrada de toda

célula e para cada sentido de transição (subida e descida) existe uma tabela que descreve seu atraso e outra que descreve seu *slew* de saída. Cada tabela é indexada por meio de dois parâmetros independentes: capacitância de saída e *slew* de entrada. O número de combinações de capacitância de saída e *slew* de entrada em cada tabela varia de acordo com a biblioteca. No caso da biblioteca da Competição do ISPD 2012, cada tabela possui sete índices de capacitância (os quais indexam as linhas) e oito índices de *slew* de entrada (os quais indexam as colunas). A Figura 9 mostra valores de atraso de descida para uma das entradas de uma célula NAND de duas entradas pertencente à biblioteca da Competição do ISPD 2012. Note que há 56 valores de atraso que correspondem à combinação de 7 valores de capacitância de saída e 8 valores de *slew* de entrada. Por exemplo, para uma capacitância de saída de 0,5fF e um *slew* de entrada de 30ps, o atraso de descida desta porta NAND é de 44,46ps. A tabela de atrasos (completa) para esta porta ainda contém 56 valores de atraso de subida com relação à mesma entrada, bem como 112 valores de atraso para a outra entrada (56 de descida e 56 de subida).

Cell Name	Cell Footprint	Max Capacitance (fF)	Cell Leakage Power (uW)	Cell Area (pm)
na02s01	na02	6.40	2	2

	Input Slew (ps)							
Output Capacitance (fF)	*5.00*	*30.00*	*50.00*	*80.00*	*140.00*	*200.00*	*300.00*	*500.00*
	Cell fall delay (ps)							
0.0	32.58	39.25	44.58	52.58	67.18	78.65	94.22	119.16
0.5	37.79	44.46	49.79	57.79	73.21	85.71	102.59	129.34
1.0	43.00	49.67	55.00	63.00	78.87	92.34	110.48	139.00
2.0	53.42	60.08	65.42	73.42	89.42	104.59	125.11	157.03
4.0	74.25	80.92	86.25	94.25	110.25	126.25	150.97	189.20
8.0	115.92	122.58	127.92	135.92	151.92	167.92	194.58	243.48
16.0	199.25	205.92	211.25	219.25	235.25	251.25	277.92	331.25

Figura 9: Tabela retirada da biblioteca da Competição do ISPD 2012 (ISPD, 2012). Esta tabela contém as informações de atraso de descida de uma das entradas de uma porta NAND como função da capacitância de saída e do *slew* de entrada.

Note que o número de combinações de capacitância de saída e *slew* de entrada para as quais uma célula é caracterizada (por simulação em nível elétrico) é limitado e portanto, interpolação bi-dimensional é utilizada para estimar o valor do atraso (ou *slew*) para uma combinação não disponível (BHASKER; CHADHA, 2009). Suponha que se necessite estimar o atraso de descida $d^f_{j\rightarrow i}$ de uma célula para uma combinação (cap_i, $slew^r_j$) não disponível na tabela. O primeiro passo é identificar os dois índices de capacitância de saída mais próximos de cap_i e os dois índices de *slew* de entrada mais próximos de $slew^r_j$. Suponha que os índices mais próximos de cap_i sejam cap_0 e cap_1, ao passo que os índices mais próximos de $slew^r_j$ sejam $slew_0$

e $slew_1$. Suponha também que as informações de atraso da tabela em relação a cada uma das combinações dos índices sejam d_{00}, d_{01}, d_{10} e d_{11}. Portanto, o valor de $d^f_{j\to i}$ será estimado por (ISPD, 2012):

$$d^f_{j\to i} = (1-Y_0)(1-X_0)d_{00} + Y_0(1-X_0)d_{01} + (1-Y_0)X_0 d_{10} + Y_0 X_0 d_{11} \tag{2.2}$$

onde

$$X_0 = (cap_i - cap_0)/(cap_1 - cap_0) \tag{2.3}$$
$$Y_0 = (slew^r_j - slew_0)/(slew_1 - slew_0) \tag{2.4}$$

Nas bibliotecas *standard cell* contemporâneas, o NLDM também é usado para modelar a parcela de curto-circuito da potência dinâmica de uma célula em função da capacitância de saída e do *slew* de entrada. Semelhantemente ao caso do atraso, para cada entrada de toda célula e para cada sentido de transição (subida e descida) existe uma tabela que descreve o *slew* de saída e uma tabela que descreve a potência dinâmica em função destes dois parâmetros. Já a potência de *leakage* é modelada por uma tabela que contém o consumo para cada uma das combinações de entrada da porta[3], além de um valor padrão que geralmente corresponde ao valor médio de todas as combinações de *leakage*. No caso da biblioteca da Competição do ISPD 2012, não há informações de potência dinâmica porque o objetivo é exclusivamente otimizar a potência de *leakage*. Além disso, considera-se apenas o valor médio de potência de *leakage* por porta (confome ilustrado na Figura 9). É importante ressaltar que para utilizar a potência de *leakage* referente a cada combinação de entrada seria necessário um arquivo contendo a atividade de chaveamento do circuito. Tal atividade de chaveamento somente pode ser obtida por meio de simulação e nem sempre está disponível, conforme observado por Ozdal, Burns e Hu (2012). Por esse motivo, na infraestrutura da Competição do ISPD 2012 utiliza-se somente a potência de *leakage* média.

Além das informações de atraso e potência, o arquivo no formato Liberty também traz, em seu cabeçalho, as informações sobre as unidades utilizadas para medida de atrasos e *slews*, tensão, corrente, potência e capacitância (tipicamente, ps, V, mA, μW, fF) etc, bem como as definições do *threshold* lógico e os limites inferior e superior para medida de *slew*. No cabeçalho

[3]Para uma porta NAND de duas entradas, por exemplo, há um valor de leakage para cada uma das combinações: 00, 01, 10, 11. A potência de *leakage* é modelado dessa forma já que a corrente de *sub-threshold* tem forte dependência nos valores de entrada devido ao *stacking effect* (RABAEY, 2009).

geral da biblioteca também há um limite que define o máximo *slew* permitido na entrada de cada porta (e pino) do circuito (no caso da biblioteca da Competição do ISPD 2012 o valor é $300ps$). As bibliotecas *standard cell* também apresentam no cabeçalho de cada célula a capacitância máxima de saída, conforme observado na Figura 9 (coluna *Max Capacitance*).

2.4 ANÁLISE DE *TIMING* ESTÁTICA

A verificação dos requisitos temporais dos circuitos digitais pode ser levada a cabo por meio de simulação ou por meio de análise de *timing* estática (GUNTZEL, 2000) (SAPATNEKAR, 2004) (BHASKER; CHADHA, 2009).

O primeiro método baseia-se em exercitar um modelo do circuito utilizando estímulos de entrada definidos pelo projetista/verificador. Desta forma, a precisão dos resultados e o tempo de execução são diretamente proporcionais à precisão do modelo e ao número de estímulos utilizados. Como o número de estímulos de entrada cresce exponencialmente com o aumento do número de entradas, este método justifica-se apenas para a verificação final (*sign-off*) de porções pequenas do circuito (GUNTZEL, 2000).

O segundo método, Análise de *Timing* Estática (STA) (*Static Timing Analysis* - STA), calcula os tempos máximos para a propagação dos sinais ao longo do bloco combinacional visitando o seu DAG em ordem topológica[4], enquanto assumindo condições de propagação pessimistas [5] (SAPATNEKAR, 2004) (BHASKER; CHADHA, 2009) (KAHNG et al., 2011). Por ser independente dos estímulos de entrada e por utilizar modelos de propagação de atraso simplificados, este método tende a ser ordens de grandeza mais rápido do que a simulação, sendo por este motivo utilizado por diversos métodos de otimização, incluindo as técnicas de *sizing*. É importante ressaltar que a grande maioria das técnicas de *sizing* encontradas na literatura invocam diversas vezes a rotina de STA para verificar o *timing* do circuito. Portanto, é importante que a análise de *timing* seja rápida e precisa.

Uma vez que o DAG deve ser percorrido em ordem topológica sempre que uma rodada de STA é realizada, é vantajoso criar uma lista com a ordem topológica dos nodos em uma etapa de pré-processamento. Tal processamento está detalhado no Algoritmo 1. Após receber como entrada o DAG de um circuito, este algoritmo faz uma busca em largura[6] a partir do nodo

[4]Corresponde ao ordenamento em que qualquer nodo v_i é processado somente após todos os seus nodos fanins (v_j) terem sido processados (KAHNG et al., 2011).

[5]Tal STA é referida por análise de atraso máximo ou de pior caso, e visa certificar-se que os requisitos de atraso máximo nas saídas primárias não são violados.

[6]Uma busca em largura a partir de um nodo v_i visita todos os seus nodos adjacentes. No caso de STA, um nodo v_k é considerado adjacente a v_i se $v_k \in \text{fanout}(v_i)$ (CORMEN et al., 2009).

Algoritmo 1: TOPOLOGICAL_SORT

```
   Input  : G(V,E)
   Output: topo_sort
 1 fifo ← ∅;
 2 topo_sort ← ∅;
 3 ENQUEUE(fifo, S);
 4 while fifo ≠ ∅ do
 5 |   v_i ← DEQUEUE(fifo);
 6 |   ENQUEUE(topo_sort, v_i);
 7 |   for ∀v_k ∈ fanout(v_i) do
 8 |   |   increment_num_of_visited_inputs(v_k);
 9 |   |   if visited_inputs(v_k) = num_input(v_k) then
10 |   |   |   ENQUEUE(fifo, v_k);
11 |   |   end
12 |   end
13 end
```

fonte (S) utilizando uma estrutura auxiliar do tipo fila (*first-in-first-out - fifo*) para armazenar os nodos a serem visitados. Quando um nodo v_i é retirado da fifo, ele é inserido na lista topo_sort, a qual armazena os nodos em ordem topológica. Em seguida, cada um de seus fanouts (v_k) é visitado. Ao visitar um nodo v_k, o algoritmo testa se todos os nodos que são *fanin* deste também já foram visitados. Em caso afirmativo, v_k é inserindo na fifo. O procedimento é repetido enquanto a fifo não estiver vazia. O resultado final do processamento é a criação da lista topo_sort, a qual contém um ordenamento topológico para o DAG do circuito. Levando-se em conta que as operações de *ENQUEUE* e *DEQUEUE* possuem complexidade constante $O(1)$, o tempo necessário para criar a lista de ordem topológica é $O(|E|)$ (CORMEN et al., 2009).

O Algoritmo 2 (Static Timing Analysis) realiza os cálculos das características temporais de um circuito no contexto de STA. Desprezando-se os atrasos das conexões, as características mais relevantes de um circuito são:

- **Tempo de chegada de descida** (*fall arrival time*) da porta v_i, definido por a_i^f, é o instante de tempo mais tarde no qual o sinal de descida na saída de v_i estabiliza;
- **Tempo de chegada de subida** (*rise arrival time*) da porta v_i, definido por a_i^r, é o instante de tempo mais tarde no qual o sinal de subida na saída de v_i estabiliza;
- **Tempo requerido de descida** (*fall required time*) da porta v_i, definido

por r_i^f, é o instante de tempo máximo no qual o sinal de descida na saída de v_i deve estabilizar, dado o **atraso crítico-alvo** (*target delay*) (A_o) do circuito. O atraso crítico-alvo é determinado a partir da frequência máxima do sinal de relógio (ou menor período de relógio), conforme as especificações do projeto;

- **Tempo requerido de subida** (*rise required time*) da porta v_i, definido por r_i^r, é o instante de tempo máximo no qual o sinal de subida na saída de v_i deve estabilizar, dado o atraso crítico-alvo do circuito;
- **slack de descida** na saída da porta v_i, definido por s_i^f, é a diferença entre o tempo requerido de descida e o tempo de chegada de descida na saída de v_i;
- **slack de subida** na saída da porta v_i, definido por s_i^r, é a diferença entre o tempo requerido de subida e o tempo de chegada de subida na saída de v_i.

Se o *slack* for positivo, ele indica a quantidade de tempo disponível que pode ser explorado para otimização (LEE; GUPTA, 2012). Se ele for negativo, indica que há uma violação de *timing* naquele ponto.

É importante ressaltar que os valores de atraso pino-a-pino (e.g., $d_{j\to i}^f$, $d_{i\to k}^r$) e *slew* pino-a-pino (e.g., $slew_{j\to i}^r$, $slew_{j\to i}^f$) das portas utilizados no Algoritmo 2 são capturados diretamente da biblioteca *standard cell*, na qual as tabelas são mapeadas pelo *slew* de entrada e capacitância de saída das portas, conforme detalhado na Seção 2.3.

Na linha 1 do Algoritmo 2, o Algoritmo 1 é invocado para criar a lista com a ordem topológica para o DAG do circuito. No trecho compreendido entre as linhas 2 e 16, o DAG é percorrido em ordem topológica direta para o cálculo dos tempos de chegada e dos *slews*. É importante observar que um nodo v_i é visitado somente após todos os seus *fanins* serem visitados, ou seja, após o cálculo das informações de tempo chegada e *slew* em cada uma de suas entradas. Os tempos de chegada a_i^f e a_i^r nas entradas primárias são computados como o maior dentre os atrasos $d_{j\to i}^f$ e $d_{j\to i}^r$, respectivamente[7]. Para as demais portas do circuito (ou saídas primárias) o tempo de chegada de descida a_i^f será calculado como: $\max_{j\in input(v_i)}(a_j^r + d_{j\to i}^f)$, onde a_j é o tempo de chegada de um *fanin* de v_i (igual ao tempo de chegada na entrada j de v_i quando os atrasos das conexões são desprezados) e $d_{j\to i}^f$ é o atraso de descida

[7]Essa definição é válida quando os tempos de chegada em cada um dos pinos de entrada das entradas primárias do circuito são 0, como é o caso da infraestrutura da Competição do ISPD 2012. Caso contrário, o cálculo dos tempos de chegada nas entradas primárias é feito da mesma forma que nas demais portas.

Algoritmo 2: STATIC_TIMING_ANALYSIS

Input : $G(V,E)$, L, A_o
Output: Timing information $\forall v_i \in V$

topo_sort $\leftarrow$ TOPOLOGICAL_SORT(DAG);
for $\forall v_i \in topo\_sort$ **do**
  $a_i^r, a_i^f \leftarrow -\infty$;
  $slew_i^r, slew_i^f \leftarrow -\infty$;
  for $\forall j \in input(v_i)$ **do**
    if $v_i \in PI$ **then**
      $a_i^f \leftarrow \max(a_i^f, d_{j\rightarrow i}^f)$;
      $a_i^r \leftarrow \max(a_i^r, d_{j\rightarrow i}^r)$;
    else
      $a_i^f \leftarrow \max(a_i^f, a_j^r + d_{j\rightarrow i}^f)$;
      $a_i^r \leftarrow \max(a_i^r, a_j^f + d_{j\rightarrow i}^r)$;
    end
    $slew_i^f \leftarrow \max(slew_i^f, slew_{j\rightarrow i}^f)$;
    $slew_i^r \leftarrow \max(slew_i^r, slew_{j\rightarrow i}^r)$;
  end
end
for $\forall v_i \in topo\_sort\ in\ reverse\ order$ **do**
  if $v_i \in PO$ **then**
    $r_i^f, r_i^r \leftarrow A_o$;
  else
    $r_i^f, r_i^r \leftarrow \infty$;
    for $\forall v_k \in fanout(v_i)$ **do**
      $r_i^f \leftarrow \min(r_i^f, r_k^f - d_{i\rightarrow k}^f)$;
      $r_i^r \leftarrow \min(r_i^r, r_k^r - d_{i\rightarrow k}^r)$;
    end
  end
  $s_i^f \leftarrow r_i^f - a_i^f$;
  $s_i^r \leftarrow r_i^r - a_i^r$;
end

de v_i em relação à sua entrada j. O cálculo do tempo de chegada de subida a_i^r é realizado de maneira similar, como pode ser observado no Algoritmo 2. O *slew* de descida na saída de v_i (linha 13) é determinado como o máximo dentre os *slews* de descida de v_i, ao passo que o *slew* de subida (linha 14) recebe o máximo dentre os *slews* de subida da porta.

No trecho compreendido entre as linhas 17 e 29, o DAG é percorrido em ordem topológica reversa para o cálculo dos tempos requeridos e dos *slacks*. Para cada nodo visitado, se v_i for uma saída primária, tanto o tempo requerido de descida (r_i^f), quanto o de subida (r_i^r) receberão o atraso crítico-alvo (A_o) especificado. Caso contrário, o algoritmo percorre todos os nodos v_k que são fanouts de v_i para calcular os tempos requeridos de descida e subida. Para o cálculo de r_i^f, o tempo requerido é definido como: $\min_{v_k \in fanout(v_i)} (r_k^f - d_{i\to k}^f)$, onde r_k^r é o tempo requerido de subida de um *fanout* de v_i (igual ao tempo requerido na saída de v_i quando os atrasos das conexões são desprezados) e $d_{i\to k}^f$ é o atraso de descida de v_k em relação à sua entrada i. O tempo requerido de subida (r_i^r) é calculado da mesma forma, invertendo-se as transições de subida e descida. Após o cálculo dos tempos requeridos de v_i (e já tendo sido calculados os tempos de chegada de todos os nodos do DAG pelas linhas 2 a 16), os *slacks* (s_i^f e s_i^r) são calculas nas linha 27 e 28. A análise de *timing* estática em um DAG é computada em tempo $O(|V|+|E|)$. Por apresentar uma complexidade linear, a STA é uma técnica escalável e aplicável em circuitos com centenas de milhões de portas (KAHNG et al., 2011).

Dependendo da técnica de *sizing* utilizada, pode não ser necessário fazer a análise de *timing* completa do circuito entre duas (ou um grupo de) iterações, bastando apenas verificar a porção de circuito que foi influenciada pela troca da opção de implementação de uma porta. Isto pode ser feito invocando-se o algoritmo de STA a partir de um nodo qualquer v_i que corresponde à origem do cone lógico associado ao sub-circuito que foi alterado. Tal procedimento, convencionalmente denominado de *Análise de Timing Incremental* (ITA - *Incremental Timing Analysis*) (JU; SALEH, 1991) (SAPATNEKAR, 2004) resulta em apreciável redução do esforço computacional e por consequência, do tempo de execução. A análise de timing incremental poderia ser realizada pelo Algoritmo 2 se este receber como parâmetro de entrada um nodo v_i, ao invés de partir do nodo fonte S.

2.5 RELAXAÇÃO LAGRANGEANA

Esta seção apresenta os conceitos básicos de Relaxação Lagrangeana, necessários para o entendimento dessa dissertação. Relaxação Lagrangeana

(LR - *Lagrangian Relaxation*) é uma técnica amplamente usada em diversos problemas de otimização com restrições, como é o caso do problema de *sizing* (discreto ou contínuo). LR tem como vantagens a flexibilidade e escalabilidade, o que permite aplicá-la em problemas de grande porte, sendo eles lineares, convexos, não-lineares, combinatórios, etc[8]. A ideia básica de LR reside em transformar um problema com restrições difíceis de manipular em um problema mais simples de ser resolvido. Isto é feito removendo-se as restrições difíceis do problema e incorporando-as na função objetivo do novo problema[9]. Cada uma das restrições incorporadas na função objetivo é acompanhada por um termo de penalidade, chamado Multiplicador de Lagrange (LM) (FISHER, 1985) (BOYD; VANDENBERGHE, 2004).

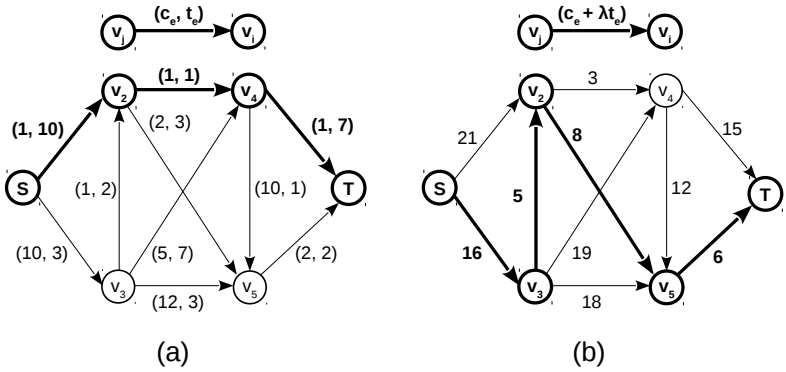


Figura 10: (a) Apresenta uma instância do problema do caminho mínimo com restrição, no qual cada aresta $e \in E$ possui um par de valores associados (c_e, t_e), onde c_e e t_e representam o custo e tempo necessário para percorrer a aresta e, respectivamente. (b) Exemplo do problema do caminho mínimo após a aplicação de Relaxação Lagrangeana assumindo um $\lambda = 2$. Note que o custo de cada aresta não é mais o par (c_e, t_e), mas sim o valor $c_e + \lambda t_e$. Adaptado de (AHUJA; MAGNANTI; ORLIN, 1993).

Os principais conceitos de Relaxação Lagrangeana serão explicados

[8]Para maiores detalhes sobre problemas lineares e não-lineares, consulte Bazaraa, Sherali e Shetty (2006). Para uma bibliografia mais extensa sobre problemas convexos, consulte Boyd e Vandenberghe (2004). Uma boa referência para problemas combinatórios é encontrada em Kreher e Stinson (1999).

[9]Não existe uma regra para escolher quais restrições devem ou não ser removidas e incorporadas na função objetivo e, portanto, ela se dá de maneira empírica (FISHBURN; DUNLOP, 1985). Maiores detalhes sobre quais restrições podem ser removidas são apresentadas no exemplo a seguir.

por meio de sua aplicação ao problema do caminho mínimo com restrição (*constrained shortest path*) devido à sua semelhança com o problema de *sizing*. Vale a pena observar que o problema do caminho mínimo com restrição é um problema NP-difícil (AHUJA; MAGNANTI; ORLIN, 1993). Considere um grafo direcionado $D(V,E)$ com um nodo fonte S e um nodo terminal T. Cada aresta $e \in E$ representa um caminho e possui dois valores associados: o custo para percorrer esta aresta, representado por c_e, e o tempo necessário para percorrê-la, representado por t_e. A Figura 10 (a) ilustra essa definição. O problema de caminho mínimo com restrição consiste em encontrar o caminho entre S e T com menor custo, tal que o tempo necessário para percorrê-lo não seja maior que um limite de tempo b especificado. A formalização do problema como problema linear inteiro é apresentada nas Equações 2.5 a 2.8, assim-chamado problema primal (PP - *Primal Problem*). A Equação 2.5 define como objetivo a minimização do custo do caminho entre S e T, ao passo que a Equação 2.6 formaliza a restrição para garantir que haja um caminho entre S e T. Já a Equação 2.7 define a restrição que o tempo total do caminho entre S e T não deve ser maior que um limite b, enquanto a Equação 2.8 restringe x_e a uma variável binária.

$$\textbf{Minimize} \quad : \quad \sum_{e \in E} c_e x_e \tag{2.5}$$

$$\textbf{Subject to} \quad : \quad \sum_{e \ enters \ v_i} x_e - \sum_{e \ leaves \ v_i} x_e = \begin{cases} -1 & \text{if } v_i = S \\ +1 & \text{if } v_i = T \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

$$\sum_{e \in E} t_e x_e \leq b \tag{2.7}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.8}$$

Suponha que as restrições possam ser divididas em dois conjuntos. A restrição definida na Equação 2.7 é difícil de manipular durante a resolução do problema e sua remoção do PP torna-o mais fácil de resolver, enquanto as demais restrições são facilmente manipuláveis. Observe que ao remover a restrição definida na Equação 2.7, o problema do caminho mínimo com restrição recai no problema clássico do caminho mínimo (*shortest path*). A ideia básica consiste em remover a restrição definida na Equação 2.7 e incorporá-la na nova função objetivo do problema, associando-a ao termo de penalidade Multiplicador de Lagrange (λ), dando origem à Função Lagrangeana (LF - *Lagrangian Function*), definida na Equação 2.9.

$$L_\lambda(x) \quad = \quad \sum_{e \in E} c_e x_e + \lambda \left(\sum_{e \in E} t_e x_e - b \right) \tag{2.9}$$

O problema é então dividido em duas etapas, as quais podem ser resolvidas de forma iterativa:

1. Subproblema Lagrangeano Relaxado (LRS - *Lagrangian Relaxation Subproblem*) formalizado nas Equações 2.10 a 2.12, tem por objetivo minimizar a LF definida anteriormente. Observe que, por recair no problema clássico do caminho mínimo, a solução ótima do LRS pode ser encontrada em tempo polinomial utilizando algoritmos consolidados na literatura como, por exemplo, *Dijkstra* ou *Bellman-Ford* (CORMEN et al., 2009). A Figura 10 (b) ilustra os novos custos das arestas após a incorporação da restrição de tempo máximo na função objetivo. É importante frisar que o termo de penalidade (λ) assume um valor fixo (não negativo) durante a resolução do LRS. Uma propriedade importante do LRS é que a sua solução ótima $z(\lambda)$ para qualquer $\lambda \geq 0$ induz a um limite inferior $z(\lambda) \leq z^*$, onde z^* é o valor da solução ótima do PP (BOYD; VANDENBERGHE, 2004). É trivial verificar tal propriedade, uma vez que qualquer solução factível do PP define um caminho entre S e T, tal que $\sum_{e \in E} t_e x_e \leq b$. Dessa forma, como qualquer solução factível do PP é também uma solução factível do LRS e o LM assume somente valores não negativos ($\lambda \geq 0$), verifica-se que $\sum_{e \in E} c_e x_e + \lambda \left(\sum_{e \in E} t_e x_e - b \right) \leq \sum_{e \in E} c_e x_e$ e, como consequência, $z(\lambda) \leq z^*$ (BOYD; VANDENBERGHE, 2004).

$$\textbf{Minimize} \quad : \quad L_\lambda(x) \tag{2.10}$$

$$\textbf{Subject to} \quad : \quad \sum_{e \text{ enters } v_i} x_e - \sum_{e \text{ leaves } v_i} x_e = \begin{cases} -1 & \text{if } v_i = S \\ +1 & \text{if } v_i = T \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

$$x_e \in \{0,1\}, \forall e \in E \tag{2.12}$$

2. Uma vez que $z(\lambda)$ é um limite inferior para z^*, deseja-se maximizá-lo a fim de se obter um limite inferior mais apertado para z^*, o que origina o assim-chamado Problema Lagrangeano Dual (LDP - *Lagrangian Dual Problem*), formalizado nas Equações 2.13 e 2.14. O objetivo do LDP é maximizar a solução ótima encontrada no LRS ($z(\lambda)$), atualizando os LMs (λ) relativos à restrição removida do PP. Um algoritmo bas-

tante usado para resolver o LDP é o método do subgradiente (FISHER, 1985). A ideia principal desse método consiste em atualizar os Multiplicadores de Lagrange com base no subgradiente (nome dado ao termo $\sum_{e \in E} t_e x_e - b$) que indica a direção na qual os LMs devem ser aumentados ou diminuídos, de acordo com a violação ou não do valor máximo determinado por b. Considere que o algoritmo está em uma iteração k. Então, os LMs referentes à iteração $k+1$ são atualizados da seguinte forma: $\lambda_{k+1} = max\{0, \lambda_k + \rho_k(\sum_{e \in E} t_e x_e - b)\}$, onde ρ_k, definido como tamanho do passo (*step size*), deve ser um valor não negativo (normalmente definido empiricamente)[10].

$$\textbf{Maximize} \quad : \quad z(\lambda) \tag{2.13}$$

$$\lambda \geq 0 \tag{2.14}$$

Sob uma condição específica, conhecida como dualidade forte[11], o valor ótimo do LDP é também o valor ótimo do PP (i.e., $z(\lambda) = z^*$), o que significa que o *gap* de otimalidade é zero. Considerando-se o problema de *sizing*, objeto de pesquisa nesta dissertação, nem o problema de *sizing* contínuo, tampouco o discreto satisfazem tais condições que garantem que o *gap* de otimalidade seja zero. Para a resolução do problema de *sizing* contínuo baseado em LR, a ideia é resolver o LRS $\rightarrow$ LDP iterativamente até que o *gap* seja menor que um valor pré-determinado (i.e., $z^* - z(\lambda) \leq \varepsilon$). Isto é possível pois o LRS em *sizing* contínuo pode ser resolvido de maneira ótima, como a solução proposta por Chen, Chu e Wong (1999). Já no caso de *sizing* discreto, apesar de o LRS ser mais simples de resolver que o PP, o LRS ainda assim é um problema NP-difícil e, portanto, não se pode obter o valor ótimo em tempo polinomial. Como consequência, as técnicas de *sizing* discreto normalmente executam o laço *LRS* $\rightarrow$ *LDP* um número fixo de iterações, como é o caso de Huang, Hu e Shi (2011), Ozdal, Burns e Hu (2012) e Li et al. (2012). Maiores detalhes da aplicação de Relaxação Lagrangeana no problema de *sizing* são apresentados no Capítulo 4.

[10] O método do subgradiente sempre converge para o valor ótimo caso a sequência de passos ρ_k satisfaça as seguintes condições: $\lim_{k \to \infty} \rho_k = 0$ e $\sum_{k=1}^{\infty} \rho_k = \infty$ (FISHER, 1985).

[11] Dualidade forte é normalmente verificada em problemas cujo objetivo e restrições são funções convexas. Entretanto, existe uma série de condições as quais garantem que a dualidade forte é verificada (i.e., $z(\lambda) = z^*$), conhecidas como condições de *Slater* (BOYD; VANDENBERGHE, 2004).

3 REVISÃO DOS TRABALHOS CORRELATOS

Este capítulo apresenta uma visão geral dos principais trabalhos de *sizing* encontrados na literatura, os quais são classificados em duas diferentes abordagens: contínuas e discretas. As técnicas pertencentes às abordagens contínuas serão apresentadas genericamente, por questão de contextualização, uma vez que o escopo desse trabalho se restringe ao fluxo *standard cell*. Já as abordagens discretas serão apresentadas detalhadamente, dando ênfase aos trabalhos publicados mais recentemente. Para cada uma das técnicas descritas é feita uma análise das suas principais limitações. Por fim, este capítulo apresenta uma visão geral do estado da arte em *sizing* discreto. É importante ressaltar que as técnicas aqui apresentadas incluem seleção da largura e da tensão de *threshold* de portas (ou transistores), aplicadas de forma conjunta ou independente. Técnicas que incluem seleção de tensão de alimentação não pertencem ao escopo deste trabalho e, portanto, não serão consideradas correlatas.

3.1 ABORDAGENS CONTÍNUAS

As técnicas pertencentes a essa classe de abordagens consideram que as portas lógicas (ou transistores) podem assumir qualquer largura ou tensão de *threshold* dentro de um intervalo contínuo restrito ($w_{min} \leq w \leq w_{max}$; $u_{min} \leq u \leq u_{max}$). A principal vantagem dessa abordagem é a possibilidade do uso de modelos convexos de atraso, o que permite obter garantias de otimalidade, ou seja, determinar a diferença entre a solução encontrada e o mínimo global do problema (BOYD; VANDENBERGHE, 2004). Tal abordagem requer o uso do fluxo de projetos *full custom*, ou de geração automática de leiaute (LAZZARI, 2003), uma vez que as portas do circuito devem ser geradas partindo dos valores determinados na etapa de otimização. Já para aplicar as abordagens contínuas ao fluxo *semi-custom*, é necessário o mapeamento das soluções obtidas para uma das opções (células) disponíveis na biblioteca *standard cell*.

Dentre os trabalhos encontrados na literatura de *sizing* contínuo, TILOS (*TImed LOgic Synthesizer*) (FISHBURN; DUNLOP, 1985) é o mais popular. De acordo com Chinnery e Keutzer (2005), as ferramentas industriais ainda usam técnicas de *sizing* fortemente baseadas no TILOS. A principal contribuição de Fishburn e Dunlop (1985) foi observar que o atraso do circuito modelado como rede RC (*Resistor-Capacitor*) é um posinômio[1] e, por-

[1] Um posinômio é uma função na qual os coeficientes são números reais positivos e os ex-

tanto, pode ser transformado em uma função convexa por meio de mudança de variáveis, a qual tem a propriedade de que qualquer mínimo da função é um mínimo global (BOYD; VANDENBERGHE, 2004). Tal modelo convexo permitiria o uso de resolvedores geométricos (MOSEK, 2013), que por sua vez conseguem encontrar a solução ótima do problema. Entretanto, a baixa escalabilidade de tais resolvedores não permitiria a resolução de instâncias de problemas da ordem de centenas de milhares de portas, conforme observado por Chinnery e Keutzer (2005). Devido às limitações de escalabilidade dos resolvedores geométricos, os autores de TILOS propuseram uma heurística gulosa baseada na função de sensibilidade (SF - *Sensitivity Function*) apresentada na Equação 3.1[2]. Esta heurística gulosa visita as portas (ou transistores) do circuito e as classifica por meio da SF. A partir de tal classificação é possível definir qual porta (ou transistor) será trocada em cada iteração. Note que a SF apresentada expressa a variação de atraso pela variação de potência resultantes da troca da opção de implementação v_i^k por uma nova opção v_i^l. Diversas técnicas encontradas na literatura baseiam-se no uso de funções de sensibilidade para guiar a otimização, como, por exemplo, Dharchoudhury et al. (1997), Sirichotiyakul et al. (1999), Nguyen et al. (2003) e Srivastava, Sylvester e Blaauw (2004).

$$SF(v_i^k, v_i^l) = \frac{\Delta d(v_i^k; v_i^l)}{\Delta p(v_i^k; v_i^l)} \tag{3.1}$$

Além das técnicas baseadas em funções de sensibilidade, uma grande variedade de técnicas pode ser encontrada na literatura de *sizing* contínuo. Diversas técnicas utilizaram programação linear (BERKELAAR; JESS, 1990) (SRIVASTAVA, 2003) (NGUYEN et al., 2003) (JEONG; KAHNG; YAO, 2009). A principal limitação de tais técnicas reside na linearização dos modelos de atraso. A aplicação de Relaxação Lagrangeana (LR) no problema de *sizing* contínuo foi proposta por Chen, Chu e Wong (1999), cuja principal contribuição foi a simplificação do Subproblema Lagrangeano Relaxado com o objetivo de remover da função objetivo os tempos de chegada (para maiores detalhes consulte a Seção 4.1). Posteriormente, diversos outros trabalhos utilizaram LR (CHEN; CHU; WONG, 1999) (TENNAKOON; SECHEN, 2002) (HSINWEI; WANG; CHEN, 2005) (TENNAKOON; SE-

---

poentes são números reais. Exemplo: $p(x) = \sum_{j=1}^{k} c_j \prod_{i=1}^{n} x_i^{\alpha_{ij}}$, onde $c_j \in \mathbb{R}^+$ e $\alpha_{ij} \in \mathbb{R}$ (BOYD; VANDENBERGHE, 2004).

[2] A função de sensibilidade proposta em TILOS era atraso vs. área. Entretanto, como esta dissertação tem como objetivo minimização de potência, a função de sensibilidade aqui apresentada é atraso vs. potência.

CHEN, 2008) (WANG; DAS; ZHOU, 2009) assumindo a simplificação proposta por Chen, Chu e Wong (1999). O uso de técnicas de otimização convexas foi adotado por diversos trabalhos, os quais propuseram modelos convexos de atraso, como é o caso de Kasamsetty, Ketkar e Sapatnekar (2000), Roy et al. (2007) e Joshi e Boyd (2008).

Também incluem-se nas abordagens contínuas: 1) As técnicas que assumem a otimização no domínio contínuo e, a partir do resultado contínuo, fazem uso de heurísticas ou de métodos de arredondamento para obter a respectiva solução discreta disponível na biblioteca de células (CHUANG; SAPATNEKAR; HAJJ, 1995) (SIRICHOTIYAKUL et al., 1999) (S.SHAH et al., 2005) (REN; DUTT, 2008) (ROY et al., 2008b). 2) As técnicas que utilizam métodos contínuos para reduzir o espaço de busca (obter solução inicial) e, em seguida, limitar o espaço de busca das soluções discretas às opções que estão mais próximas das soluções contínuas encontradas (HU; KETKARY; HU, 2009) (RAHMAN; TENNAKOON; SECHEN, 2011).

A principal limitação das abordagens contínuas é a necessidade de mapeamento das soluções encontradas (e.g. larguras das portas) para as opções disponíveis na biblioteca *standard cell*, resultando na degradação da solução obtida (OZDAL; BURNS; HU, 2012). Além disso, a maioria das técnicas utiliza modelos simplificados de atraso e, de acordo com Chinnery e Keutzer (2005), nem mesmo modelos de atraso convexos ou posinomiais possuem precisão suficiente para modelar o atraso das portas contidos nas bibliotecas *standard cell* contemporâneas. Em relação às técnicas que propõem o uso de heurísticas para mapeamento das soluções, a principal desvantagem reside no fato de que as opções de implementação (e.g., diferentes larguras) das portas em uma biblioteca *standard cell* não estão uniformemente distribuídas — ao contrário, estão geometricamente distribuídas[3] (BEEFTINK et al., 1998). Lee e Gupta (2012) ressaltam que mesmo as técnicas que são guiadas pelas soluções contínuas sofrem dos mesmos problemas que as abordagens discretas, visto que o espaço de soluções ainda assim é muito grande e nem sempre a solução ótima está dentro do espaço reduzido, devido à natureza combinatória do problema de *sizing* discreto.

3.2 ABORDAGENS DISCRETAS

Nesta seção são apresentados os principais detalhes e contribuições das técnicas que abordam o problema de *sizing* diretamente no domínio dis-

[3] O uso de opções geometricamente distribuidas é necessário, visto que uma distribuição uniforme resultaria em um número muito grande de portas em uma biblioteca, o que não é desejável (BEEFTINK et al., 1998).

creto, considerando somente as opções de implementação disponíveis na biblioteca de células (e.g., largura e tensão de *threshold*). O problema de *sizing* discreto é um problema de otimização combinatória o qual foi provado ser NP-difícil (LI, 1994). Em função da limitação de espaço, serão detalhados somente os trabalhos mais recentes encontrados na literatura, além dos trabalhos que tem importância histórica devido ao grande número de citações.

3.2.1 Coudert (1997)

Coudert (1997) propôs uma técnica de *sizing* visando contornar a principal limitação das heurísticas gulosas aplicadas ao problema de *sizing* discreto: a grande possibilidade de ficarem presas em mínimos locais. Além disso, o autor ressaltou a limitação dos trabalhos que utilizam modelos de atraso lineares ou posinomiais, argumentando que estes podem apresentar erros de até 20% quando comparados com os modelos de atraso usados em bibliotecas *standard cell*. O autor afirmou que o esforço computacional para se reavaliar o *timing* do circuito em técnicas que trocam uma porta por iteração é alto, já que é necessário atualizar as portas pertencentes ao cone lógico da porta trocada através de uma análise de *timing* incremental. Para contornar esse problema, foram propostas duas heurísticas visando reduzir o *overhead* computacional da análise de *timing* incremental . A primeira avalia somente o subcircuito em volta da porta trocada, composto por um (ou dois) níveis de *fanin* e *fanout*, em vez de avaliar todas as portas afetadas pela troca. Já a segunda heurística, atualiza o nodo somente após um de seus vizinhos ter sido redimensionado também, conseguindo-se assim um bom compromisso entre precisão de *timing* e tempo de execução.

Para redução de potência sujeita a restrições de atraso, Coudert (1997) apresentou uma heurística gulosa baseada em três artifícios para evitar mínimos locais. 1) Perturbação da solução por meio de variáveis aletórias. 2) Relaxação das restrições do problema. 3) Múltiplas trocas simultâneas. Inicialmente a heurística minimiza o atraso crítico do circuito (maximiza o *slack*). Em seguida, ela minimiza a potência do circuito observando as restrições de atraso. O autor afirma que minimizando o atraso crítico e ficando "longe" da região infactível (i.e., atraso crítico muito menor que o atraso crítico-alvo) diminuem-se as chances da etapa de redução de potência ficar presa em um mínimo local. A Equação 3.2 apresenta a função custo usada na etapa de redução de potência para calcular o impacto da troca da opção de uma porta na potência total do circuito. Note que as trocas que resultam em *slack* nega-

tivo são penalizadas com custo infinito.

$$P_{cir} = \begin{cases} +\infty & \text{if slack} < 0 \\ P & \text{otherwise} \end{cases} \tag{3.2}$$

Já as Equações 3.3 e 3.4 apresentam a função de relaxação usada para avaliar a relação de redução de potência por aumento de atraso resultante da troca da opção de uma determinada porta v_i. Nessas equações, ε e α são valores pré-computados de acordo com as características do circuito, ao passo que s_i corresponde ao *slack* da porta i antes da troca, e Δs_i corresponde à variação de *slack* resultante da troca da opção da porta i. A função Φ, por sua vez, define uma relação custo/benefício da variação de potência e do *slack* do circuito resultantes da troca da opção de uma porta.

Como resultado da aplicação das funções de relaxação apresentadas anteriormente e da perturbação do espaço de soluções, a heurística proposta por Coudert (1997) tem menores chances de ficar presa em mínimos locais. De qualquer forma, não há garantias de otimalidade, visto que o problema de *sizing* discreto é NP-difícil.

$$Relax_i = \begin{cases} \varepsilon & \text{if } \Delta p_i > 0 \text{ or } (s_i + \Delta s_i < 0) \\ (\varepsilon - \alpha \Delta p_i) \times \Phi(\frac{\Delta s_i}{\varepsilon + s_i}) & \text{otherwise} \end{cases} \tag{3.3}$$

$$\Phi(x) = \begin{cases} 1 + x & \text{if } x \geq 0 \\ \frac{1}{1-x} & \text{otherwise} \end{cases} \tag{3.4}$$

A principal limitação da técnica proposta por Coudert (1997) é o alto esforço computacional, o que torna proibitiva a sua aplicação em circuitos da ordem de centenas de milhares de portas.

3.2.2 Chinnery e Keutzer (2005)

Chinnery e Keutzer (2005) propuseram uma extensão à técnica de Nguyen et al. (2003), apresentando melhorias no modelo de atraso, as quais consideram múltiplos arcos de tempo em uma porta, transições de subida/descida e *slew*. Os valores de atraso e potência foram capturados diretamente de uma biblioteca *standard cell*. Os autores propuseram uma formulação de *sizing* discreto baseada em Programação Linear (LP - *Linear Programming*), cuja essência é capturada entre as Equações 3.5 e 3.11. Primeiramente, para cada porta do circuito, a formulação LP encontra a opção de implementação com o menor valor para a função de sensibilidade da Equação 3.12, com res-

peito à sua opção atual. Em seguida, ela determina, por meio da variável γ_i, se a porta representada por v_i deve ou não ser trocada por sua opção de menor sensibilidade. Note que a Equação 3.10 permite que a variável γ assuma qualquer valor real entre 0 e 1, visto que o uso de variáveis binárias tornaria o tempo de execução proibitivo. Com isso, os autores estabeleceram um limiar de 0,99 que define se uma porta deve ou não ser trocada por sua opção alternativa. As Equações 3.6 e 3.7 definem as restrições de atraso crítico-alvo do circuito[4]. Os termos $\Delta d^f_{j\to i}$ e $\Delta d^r_{j\to i}$ das Equações 3.8 e 3.9 representam a variação de atraso resultante da troca da porta v_i por sua opção alternativa.

$$\textbf{Minimize} \quad : \quad \sum_{v_i \in X} \gamma_i \Delta p_i \tag{3.5}$$

$$\textbf{Sujeito a} \quad : \quad a^f_i \leq A_o, \forall v_i \in PO \tag{3.6}$$

$$: \quad a^r_i \leq A_o, \forall v_i \in PO \tag{3.7}$$

$$a^r_j + d^f_{j\to i} + \gamma_i \Delta d^f_{j\to i} \leq a^f_i, \forall j \in input(v_i), \forall v_i \in (X \cup PI) \tag{3.8}$$

$$a^f_j + d^r_{j\to i} + \gamma_i \Delta d^r_{j\to i} \leq a^r_i, \forall j \in input(v_i), \forall v_i \in (X \cup PI) \tag{3.9}$$

$$0 \leq \gamma_i \leq 1, \forall v_i \in X \tag{3.10}$$

$$\gamma_i = 0, \forall v_i \in (PI \cup PO) \tag{3.11}$$

$$SF(v^k_i, v^l_i) = \frac{\Delta p(v^k_i; v^l_i)}{\Delta d(v^k_i; v^l_i)} \tag{3.12}$$

O fluxo de minimização de potência proposto Chinnery e Keutzer (2005) pode ser dividido em cinco etapas, as quais são executadas iterativamente. 1) Minimiza-se o atraso do circuito a fim de se maximizar o *slack*. 2) Encontra-se a opção de implementação de cada porta que apresenta a menor sensibilidade em relação à opção atual. 3) Formula-se o LP com as opções definidas na etapa anterior para escolher quais portas serão trocadas através das variáveis γ. 4) Nessa etapa, as portas cujo γ seja maior que 0.99 são trocadas por sua opção candidata. 5) Verifica-se o *timing* do circuito através de uma STA. Caso não haja violação, volta-se novamente para o passo 2. Caso contrário, os autores utilizam uma formulação LP alternativa para reestabelecer o atraso crítico-alvo do circuito e, então, voltar ao passo 2 novamente. O

[4]Tal formulação das restrições de atraso crítico-alvo será explicada em detalhes na Seção 4.1.

laço entre os passos 2 e 5 é repetido até que a redução de potência entre duas iterações consecutivas seja menor que um valor pré-determinado.

A principal limitação dessa técnica é a utilização de uma variável contínua (γ) para definir se uma porta é trocada ou não, a qual pode resultar em soluções subótimas ou infactíveis (OZDAL; BURNS; HU, 2011). Outra limitação reside no fato de que o modelo LP não considera a influência do atraso nas portas vizinhas.

3.2.3 Liu e Hu (2010)

Liu e Hu (2010) formularam o problema de *sizing* utilizando Relaxação Lagrangeana para minimizar a potência sujeito a restrições de atraso. Para a resolução do Subproblema Lagrangeano Relaxado (LRS), os autores propuseram uma heurística baseada em Programação Dinâmica (DP - *Dynamic Programming*) [5] ao passo que o Problema Lagrangeano Dual foi resolvido através do tradicional método do subgradiente (o mesmo utilizado por (CHEN; CHU; WONG, 1999)).

$$L_\lambda : \sum_{v_i \in X} p_i + \sum_{v_i \in (X \cup PI)} \Big(\sum_{v_j \in input(v_i)} \lambda_{j \to i} d_{j \to i} \Big) \tag{3.13}$$

A modelagem LR utilizada pelos autores assumiu a mesma simplificação proposta por Chen, Chu e Wong (1999), cuja função objetivo consiste em minimizar o somatório das potências das portas do circuito e o somatório dos atrasos das portas do circuito, conforme a Equação 3.13. Estes últimos são acompanhados do termo de penalidade, Multiplicador de Lagrange (λ). A aplicação de DP ao problema de *sizing* tem por objetivo resolver o problema em etapas, onde cada etapa consiste na escolha da opção de uma porta do circuito. A Equação 3.14 apresenta a função utilizada para o cálculo do custo, ordem topológica reversa, de cada opção m de cada porta v_i.

$$f(v_i^m) = \sum_{v_k \in fanout(v_i)} \Big(\min_{h \in options(v_k)} (f(v_k^h) + \lambda_{i \to k} d_{i \to k}) + p_i^m \Big) \tag{3.14}$$

Uma vez calculados os custos das portas, o grafo do circuito é percor-

[5] Programação Dinâmica é uma técnica de otimização normalmente aplicada a problemas cuja solução ótima pode ser construída através da resolução de problemas menores, normalmente de maneira recursiva. Uma propriedade importante de DP é que ela permite construir a solução ótima do problema original a partir da resolução ótima dos problemas menores (CORMEN et al., 2009).

rido em ordem topológica direta para a escolha da opção de implementação de cada porta que resulte no menor custo. A Equação 3.15 define a função utilizada para a escolha da opção de porta do circuito.

$$solution(v_i) = \min_{m \in options(v_i)} (f(v_i^m) + \sum_{v_j \in input(v_i)} (\lambda_{j \to i} d_{j \to i} + p_j)) \quad (3.15)$$

Uma das limitações dessa técnica advém da utilização de modelos de atraso simplificados, os quais não são precisos o suficiente para representar o atraso em bibliotecas *standard cell* contemporâneas. Outra limitação reside no cálulo impreciso dos atrasos das portas cujo *fanout* é maior que um, resultando assim em soluções subótimas.

3.2.4 Huang, Hu e Shi (2011)

Huang, Hu e Shi (2011) concentraram-se na resolução do Problema Lagrangeano Dual (LDP). Os autores afirmam que a natureza discreta do problema de *sizing* torna o Problema Dual não-convexo, ao contrário das abordagens contínuas, onde o LDP é convexo e permite o uso eficiente do método do subgradiente. Para contornar tal limitação, os autores propuseram um método heurísitico chamado "projection-based descent" para atualizar os Multiplicadores de Lagrange relativos às restrições de *timing* do circuito. A Equação 3.16 define a variação dos LMs de saída de cada porta do circuito (μ_i) a cada iteração. Tal atualização leva em conta a redução de tempo de chegada (a_i) necessária para que a porta atinja o tempo requerido (r_i)[6]. No caso de *slack* positivo ($r_i > a_i$), a ideia é aumentar o tempo de chegada para possibilitar reduções de potência. A função *eta* linha (η') retorna a direção (*slope*) na qual o Multiplicador de Lagrange deve ser atualizado para aumentar ou diminuir o tempo de chegada. Em outras palavras, essa função modela a interação entre o LM e tempo de chegada de uma porta baseado no seu histórico recente (últimas iterações).

$$\Delta\mu_i = \frac{r_i - a_i}{\eta_i'} \quad (3.16)$$

A principal limitação dessa técnica reside na utilização de um modelo

[6]Como os autores não consideram transições de subida e descida, a_i refere-se ao maior tempo de chegada entre subida e descida, enquanto r_i representa o maior tempo requerido entre subida e descida.

de atraso simplificado que não considera múltiplos arcos de tempo por porta, tampouco transições de subida e descida. Outro ponto a se questionar é a falta de uma avaliação experimental mais extensa que permita concluir se a técnica é realmente efetiva em circuitos realistas.

3.2.5 Ozdal, Burns e Hu (2012)

Ozdal, Burns e Hu (2012) (extensão do artigo apresentado no ICCAD 2011 (OZDAL; BURNS; HU, 2011)) apresentaram uma formulação baseada em Relaxação Lagrangeana cujo objetivo foi obter o melhor compromisso entre atraso crítico do circuito e potência de *leakage*. Uma das contribuições apresentadas pelos autores foi uma formulação LR que separa a análise de *timing* estática da rotina de otimização. Segundo os autores, a formulação LR usada por outros trabalhos (e.g., Chen, Chu e Wong (1999) Liu e Hu (2010) Huang, Hu e Shi (2011)), a qual incorpora a STA diretamente na formulação do problema, não é apropriada para circuitos industriais modernos pois estes apresentam restrições temporais complexas tais como vários domínios de relógio, falsos caminhos, dentre outras. Segundo os autores, para tal tarefa o uso de ferramentas industriais de STA seria mais eficiente. Os autores observaram que a atualização dos Multiplicadores de Lagrange de cada uma das portas do circuito é feita com base na diferença entre os caminhos da porta (no caso de uma transição de descida, $a_j^r + d_{j\to i}^f - a_i^f$). Entretanto, os valores de tempo de chegada das portas normalmente não são reportados pelas ferramenta industriais de STA e tal atualização não é possível. Por outro lado, essas ferramentas reportam os *slacks* das portas. Dessa forma, cada um dos tempos de chegada pode ser substituído pela diferença entre o *slack* na saída da porta e o *slack* na entrada (no caso de uma transição de descida, $-s_j^r + s_i^f$). O conjunto de Equações 3.17 a 3.20 apresenta essa definição passo a passo. Como resultado, é possível atualizar os Multiplicadores de Lagrange utilizando os *slacks* (conforme Equação 3.20) calculados por uma ferramenta industrial de STA, visto que eles são mais precisos do que os calculados por ferramentas de STA não-industriais.

Como segunda contribuição, Ozdal, Burns e Hu (2012) propuseram um modelo de grafo para capturar precisamente a formulação LR. Tal modelo leva em conta que o atraso de uma porta em uma biblioteca *standard cell* contemporânea não é função linear[7] em relação à sua capacitância de saída, ao contrário de outros trabalhos (LIU; HU, 2010).

[7] Uma função é dita linear se f(x + y) = f(x) + f(y) (BOYD; VANDENBERGHE, 2004). O atraso de uma porta não é função linear de sua capacitância de saída, i.e., $d_{ji}(x+y) \neq d_{ji}(x) + d_{ji}(y)$.

A terceira contribuição apresentada pelos autores é uma heurística baseada em Programação dinâmica (DP) para resolver a formulação LR capturada no modelo de grafo mencionado acima. Para resolver o problema de reconvergências em um DAG durante o processo de DP, os autores propuseram uma heurística de cortes de árvores no grafo. A ideia básica consiste em percorrer o circuito em ordem topológica direta e, quando visitar um nodo com *fanout* maior que um, escolhe-se o *fanout* mais crítico para continuar a árvore crítica. Tal escolha é feita comparando-se os LMs. Após dividir o circuito em árvores, a heurística DP é aplicada ao grafo do circuito para uma opção de implementação para cada porta do circuito.

Definição de slack :

$$s_j^f = r_j^f - a_j^f \; ; \; s_j^r = r_j^r - a_j^r$$
$$s_i^f = r_i^f - a_i^f \; ; \; s_i^r = r_i^r - a_i^r \tag{3.17}$$

Diferença entre slack na saída e slack na entrada :

$$-s_j^f + s_i^r = -(r_j^f - a_j^f) + (r_i^r - a_i^r)$$
$$-s_j^r + s_i^f = -(r_j^r - a_j^r) + (r_i^f - a_i^f) \tag{3.18}$$

Tempo requerido na entrada reescrito :

$$-s_j^f + s_i^r = -(r_i^r - d_{j\to i}^r - a_j^f) + (r_i^r - a_i^r)$$
$$-s_j^r + s_i^f = -(r_i^f - d_{j\to i}^f - a_j^r) + (r_i^f - a_i^f) \tag{3.19}$$

Representação do tempo de chegada como diferença de slacks :

$$-s_j^f + s_i^r = a_j^f + d_{j\to i}^r - a_i^r$$
$$-s_j^r + s_i^f = a_j^r + d_{j\to i}^f - a_i^f \tag{3.20}$$

A primeira contribuição do trabalho, que tem por objetivo a utilização dos *slacks* para atualizar os LMS é puramente teórica, visto que as ferramentas industriais de STA reportam os tempos de chegada das portas — ao contrário do que foi afirmado pelos autores. A heurística baseada em DP tem duas principais limitações: a primeira é o longo tempo de execução, uma vez que todas as opções de implementação de cada porta do circuito e de suas vizinhas são avaliadas a cada iteração, o que pode tornar o tempo de execução proibitivo para circuitos muito grandes. A segunda limitação é consequência da heurística usada para o corte de árvores no grafo, a qual é fonte de subotimalidade. Ademais, as restrições de máximo *slew* na entrada de uma porta e máxima capacitância de saída de uma porta definidas nas bibliotecas *standard*

cell contemporâneas não são consideradas durante o processo de otimização.

3.2.6 Rahman, Tennakoon e Sechen (2012)

Rahman, Tennakoon e Sechen (2012) propuseram uma heurística selecionar a tensão de *threshold* de cada porta buscando minimizar a potência de *leakage* do circuito. Para tal, os autores assumem uma solução inicial com *slacks* positivos. A heurística é baseada na função de sensibilidade apresentada na Equação 3.21. Tal função tem por objetivo capturar a variação do *slack* positivo do circuito dividida pela redução de leakage, ambas resultantes da troca de uma opção de implementação v_i^k por uma nova opção v_i^l — no caso, a troca da tensão de *threshold* da porta pela opção imediatamente superior, caso exista. A partir de tal função as sensibilidades de todas as portas do circuito são calculadas. Em seguida, as portas são avaliadas uma a uma, em ordem não-decrescente de sensibilidade, sendo que somente as trocas que não resultam em *slack* negativo no circuito são realizadas.

$$SF(v_i^k, v_i^l) = \frac{\Delta TS(v_i^k, v_i^l)}{\Delta leakage(v_i^k, v_i^l)} \tag{3.21}$$

Com o objetivo de investigar os menores valores de *leakage* possíveis para um determinado circuito, os autores utilizaram um algoritmo iterativo no qual a heurística mencionada no parágrafo anterior assume um atraso crítico-alvo ligeiramente incrementado a cada iteração (Δ unidades de tempo). Com isso, esperava-se uma maior redução de *leakage* por parte da heurística proposta pelos autores visto que resultaria em uma maior quantidade de *slacks* positivos no circuito, os quais seriam utilizados para reduzir *leakage*. Como passo subsequente deste algoritmo, a técnica de Rahman, Tennakoon e Sechen (2011) foi utilizada para reestabelecer o atraso crítico-alvo original do circuito. Obviamente, à medida que o incremento do atraso crítico-alvo do circuito é muito grande, o custo de *leakage* para recuperação do atraso crítico-alvo original é maior do que o ganho proveniente da heurística de redução de potência.

A principal limitação dessa técnica reside na necessidade de iniciar o processo de otimização apenas com *slacks* positivos, necessitando assim de uma solução inicial obtida a partir de outra técnica. Além disso, as restrições de máximo *slew* e máxima capacitância não são consideradas pelos autores durante as trocas de tensão de *threshold*.

3.2.7 Hu et al. (2012)

Hu et al. (2012) propuseram o uso de meta-heurísticas para resolver o problema de seleção da largura e da tensão de *threshold* das portas do circuito visando minimização de *leakage*. A técnica proposta foi validada experimentalmente usando a infraestrutura da Competição do ISPD de 2012 (OZDAL et al., 2012). A ideia consiste basicamente em invocar diversas heurísticas gulosas de forma paralela, cada uma com um parâmetro aleatório diferente, guardando um pequeno conjunto com as melhores soluções encontradas (método *go-with-the-winners* GWTW (ALDOUS; VAZIRANI, 1994)). O método proposto é dividido em duas partes:

1. A primeira etapa[8] tem por objetivo reduzir o atraso crítico do circuito de maneira iterativa até que o atraso crítico-alvo do circuito seja satisfeito e, ao mesmo tempo, respeitar as restrições de máxima capacitância e máximo *slew*. Nesta etapa, a sensibilidade de cada opção para cada porta do circuito é calculada usando a Equação 3.22 (nesta equação, α define a importância da potência de *leakage*) para então gerar uma lista de sensibilidade. Obviamente, para estimar a variação da quantidade total de *slack* negativo (TNS - *Total Negative Slack*) no circuito é necessária uma análise de *timing* incremental, o que pode resultar em esforço computacional proibitivo. Para contornar esse problema, os autores propuseram estimar a variação de TNS (resultante da troca da opção v_i^k pela opção v_i^l) de acordo com a Equação 3.23. Nesta equação, N_i define um conjunto contendo v_i e as portas que tem um *fanin* em comum com v_i, ao passo que Δd_j^k define a variação de atraso em uma porta v_j decorrente da nova opção de v_i^l, $NPaths_j$ define o número de caminhos com *slack* negativo que passam pela porta v_j. As sensibilidades são ordenadas em ordem não-crescente e um fator γ é usado para determinar a quantidade máxima de trocas antes de uma nova STA completa, já que o erro acumulado depois de um certo número de trocas pode ser muito grande. É importante ressaltar que as variáveis γ e α foram obtidas de maneira empírica e diferentes valores foram usados para cada circuito. Ao final de cada iteração, uma heurística é usada para consertar as violações de máxima capacitância remanescentes. Os autores observaram que ao resolver as violações de capacitância, os *slews* de saída das portas ficam dentro do limite especificado pela biblioteca.

[8] Esta etapa parte de uma solução inicial com todas as portas do circuito na menor configuração de potência, i.e., menor largura e maior tensão de *threshold*.

$$SF(v_i^k, v_i^l) = \frac{\Delta TNS(v_i^k, v_i^l)}{\Delta leakage^{\alpha}(v_i^k, v_i^l)} \tag{3.22}$$

$$\Delta TNS(v_i^k, v_i^l) \approx \sum_{v_j \in N_i} -\Delta d_j^k \sqrt{NPaths_j} \tag{3.23}$$

2. A partir da solução anterior, a segunda etapa tem por objetivo reduzir a potência do circuito, aceitando apenas trocas que não violem nem as restrições de atraso, tampouco as restrições de máxima capacitância e máximo *slew*. Esta etapa faz o uso de uma heurística baseada em cinco funções de sensibilidade diferentes. Tais sensibilidades são utilizadas para criar uma lista de opções a serem trocadas, de maneira similar a Rahman, Tennakoon e Sechen (2012). As Equações 3.24 a 3.28 apresentam as funções de sensibilidade onde Δd e s_i definem, respectivamente, a variação de atraso de uma porta e a variação de slack na saída dessa porta, ambas resultantes da troca de v_i^k por v_i^l, ao passo que $\#paths$ define a quantidade de caminhos que passam pela porta. Os autores ressaltaram que nenhuma das cinco funções de sensibilidade obteve superioridade absoluta nos circuitos da Competição do ISPD 2012. Por isso, a heurística proposta foi invocada diversas vezes (de maneira paralela) para então escolher o menor resultado de *leakage* obtido.

$$SF_1(v_i^k, v_i^l) = \frac{-\Delta leakage(v_i^k, v_i^l)}{\Delta d(v_i^k, v_i^l)} \tag{3.24}$$

$$SF_2(v_i^k, v_i^l) = -\Delta leakage(v_i^k, v_i^l) \times s_i \tag{3.25}$$

$$SF_3(v_i^k, v_i^l) = \frac{-\Delta leakage(v_i^k, v_i^l)}{\Delta d(v_i^k, v_i^l) \times \#paths} \tag{3.26}$$

$$SF_4(v_i^k, v_i^l) = -\Delta leakage(v_i^k, v_i^l) \times \frac{s_i}{\#paths} \tag{3.27}$$

$$SF_5(v_i^k, v_i^l) = -\Delta leakage(v_i^k, v_i^l) \times \frac{s_i}{\Delta d(v_i^k, v_i^l) \times \#paths} \tag{3.28}$$

$$\tag{3.29}$$

Uma das limitações reside no cálculo aproximado do total de *slacks* negativos (TNS) utilizado na primeira etapa. Tal cálculo considera o impacto de uma troca somente nas portas *fanin* da porta trocada, enquanto as *fanouts* são desconsideradas. Outra limitação é o alto esforço computacional necessário para otimizar circuitos da ordem de centenas de milhares de por-

tas. Isso é consequência do uso único e exclusivo de técnicas baseadas em funções de sensibilidade, as quais requerem o cálculo das sensibilidades de todas as opções de implementação de todas as portas do circuito, mesmo que o impacto do atraso e *slew* seja limitado aos *fanins* da porta trocada.

3.2.8 Li et al. (2012)

Li et al. (2012) propuseram o uso de uma técnica híbrida para o problema de *sizing* discreto que, assim como a técnica de Hu et al. (2012), também foi validada usando a infraestrutura da Competição do ISPD de 2012. Tal técnica é dividida em três etapas:

1. Os autores afirmaram que o uso de LR para otimizar simultaneamente atraso e potência não é eficiente para *sizing* discreto. Por esse motivo, eles propuseram uma heurística baseada em formulação LR para minimizar o atraso crítico do circuito sem considerar *leakage*.

2. Esta etapa baseia-se em uma modelagem do problema de *sizing* utilizando fluxo em redes para redução de potência, sujeita às restrições de atraso do circuito, partindo da configuração de mínimo atraso. A abordagem de fluxo em redes utilizada se difere das demais pois adiciona nodos e arestas extras para representar os múltiplos arcos de tempo das portas. Para cada porta, a sua opção de implementação candidata é definida utilizando uma função de sensibilidade *leakage* versus atraso (similar à Equação 3.24). Assim como no trabalho de Chinnery e Keutzer (2005), a opção candidata é a que apresentar a menor sensibilidade.

3. Esta parte tem por objetivo utilizar os *slacks* residuais resultantes da etapa anterior para redução de potência de *leakage* sem violar o atraso crítico-alvo. Para isso utiliza uma heurística gulosa baseada na função de sensibilidade *leakage* versus atraso da etapa anterior. Visto que para calcular a sensibilidade de cada opção de implementação de cada porta do circuito seria necessária uma análise de *timing* incremental, o que pode tornar o tempo proibitivo, os autores propuseram uma abordagem conservativa e rápida para o cálculo das sensibilidades. Esse cálculo baseia-se apenas em informações locais da porta (e de seus *fanins* imediatos) para avaliar se a troca de uma porta por sua opção candidata viola ou não as restrições de *timing*.

A principal limitação reside na necessidade de arredondamento das soluções encontradas na etapa de fluxo em redes. Assim como no trabalho de Chinnery e Keutzer (2005), arredondamento pode degradar a qualidade das

soluções, ou até mesmo gerar soluções infactíveis que violam as restrições de *timing*.

3.2.9 O Estado da arte em *Sizing* Discreto

Além das técnicas de *sizing* discreto apresentadas anteriormente, é importante destacar algumas das técnicas utilizadas pelas equipes melhores colocadas na Competição do ISPD 2012[9]. Há uma grande variedade de técnicas, incluindo meta-heurísticas, métodos híbridos e Relaxação Lagrangeana. Mais especificamente, sabe-se que a técnica utilizada pela primeira colocada (NTUgs) foi baseada na formulação e nas heurística propostas por Ozdal, Burns e Hu (2011), enquanto a segunda colocada (UFRGS-Brazil) utilizou meta-heurísticas, mais especificamente *Simulated Annealing*.

Analisando os principais trabalhos de *sizing* encontrados na literatura, é possível observar que Relaxação Lagrangeana é uma técnica bastante usada tanto nas abordagens contínuas, quanto nas discretas. Sua principal vantagem é permitir otimizar, simultaneamente, diversos objetivos conflitantes e, ao mesmo tempo, ser escalável para circuitos com centenas de milhares de portas, como é o caso de Li et al. (2012) e Ozdal, Burns e Hu (2012).

Vale a pena observar também a tendência do uso de técnicas híbridas no problema de *sizing* discreto, em especial nos trabalhos publicados mais recentemente (HU et al., 2012) (LI et al., 2012). A principal vantagem de utilizar diferentes técnicas é que cada uma tem uma característica específica e pode contornar as limitações das outras.

Desde a Competição do ISPD 2012, a qual proporcionou os primeiros resultados considerando a infraestrutura disponibilizada na Competição, novos trabalhos foram publicados utilizando essa infraestrutura e obtiveram melhores resultados de potência de *leakage* e tempo de execução que os obtidos pelas equipes melhores colocadas. Entretanto, não se observou um predomínio total de nenhum dos trabalhos mais recentes em todos os circuitos da Competição. Além disso, Hu et al. (2012) apresentaram um estudo mostrando que os resultados de *leakage* por eles obtidos, apesar de serem os melhores até então, ainda assim poderiam ser melhorados, em especial nos circuitos com restrição de atraso crítico-alvo mais apertada (*fast*). A Tabela 2 apresenta um resumo das principais características dos trabalhos de *sizing* discreto descritos na Seção 3.2, das ferramentas de *sizing* das três equipes melhores colocadas na Competição do ISPD 2012 e das duas técnicas propostas nessa dissertação, as quais são detalhadas nos Capítulos 5 e 6, respectivamente.

[9] As técnicas utilizadas pelas equipes participantes não foram divulgadas oficialmente. Entretanto, algumas são informalmente conhecidas.

Tabela 2: Resumo dos principais e mais recentes trabalhos encontrados na literatura de *sizing* discreto.

Trabalho Correlato	Método de Resolução	Modelo de Atraso	Considera Restrições de Slew e Capacitância?	Escalabilidade	Infraestrutura Experimental
Coudert (1997)	Heurística Gulosa Baseada em Randomização	*Lookup*	Não	Ruim	ISCAS'85
Chinnery e Keutzer (2005)	Programação Linear	*Linear*	Não	Média	ISCAS'85
Liu e Hu (2010)	Relaxação Lagrangeana e Programação Dinâmica	*Elmore*	Não	Média	IWLS'05
Huang, Hu e Shi (2011)	Relaxação Lagrangeana e Heurística Gulosa	*Elmore*	Não	Boa	ISCAS'85
Ozdal, Burns e Hu (2012)	Relaxação Lagrangeana e Programação Dinâmica	*Lookup*	Não	Média	Circuitos Industriais
Rahman, Tennakoon e Sechen (2012)	Heurística Baseada em Função de Sensibilidade	*Lookup*	Não	Boa	ITC'99 e Circuitos Industriais
Hu et al. (2012)	Heurísticas Baseadas em Funções de Sensibilidade e *Go-With-the-Winners*	*Lookup*	Sim	Média	Infraestrutura da Competição do ISPD 2012
Li et al. (2012)	Relaxação Lagrangeana e Fluxo em Redes	*Lookup*	Sim	Boa	Infraestrutura da Competição do ISPD 2012
NTUgs – 1^{ro} Colocado na Competição de Sizing Discreto do ISPD 2012	Relaxação Lagrangeana e Programação Dinâmica	*Lookup*	Sim	Boa	Infraestrutura da Competição do ISPD 2012
UFRGS – 2^{do} Colocado na Competição de Sizing Discreto do ISPD 2012	*Simulated Annealing*	*Lookup*	Sim	Boa	Infraestrutura da Competição do ISPD 2012
PowerValve – 3^{ro} Colocado na Competição de Sizing Discreto do ISPD 2012	Não Disponível	*Não* Disponível	Sim	Boa	Infraestrutura da Competição do ISPD 2012
Proposta 1: Heurística Gulosa Topológica	Relaxação Lagrangeana e Heurística Gulosa	*Lookup*	Sim	Boa	Infraestrutura da Competição do ISPD 2012
Proposta 2: Técnica Híbrida	Relaxação Lagrangeana e Heurística Gulosa e Recuperação de Atraso	*Lookup*	Sim	Boa	Infraestrutura da Competição do ISPD 2012

Apesar dos diversos trabalhos publicados recentemente no escopo de *sizing* discreto para minimização de *leakage*, até a Competição do ISPD 2012 não havia uma infraestrutura que permitisse uma comparação justa e direta entre as diferentes técnicas. Por este motivo, normalmente comparava-se apenas com as ferramentas industriais, as quais estão muito aquém das técnicas mais recentemente publicadas. Os experimentos reportados nos Capítulos 5 e 6 utilizaram a infraestrutura completa da Competição, incluindo todos os circuitos. Os resultados obtidos no Capítulo 5 foram comparados com os obtidos pelas três equipes melhores colocadas na Competição, visto que elas representavam o estado da arte naquele momento. No Capítulo 6, além dos melhores resultados da Competição, incluiram-se na comparação os resultados de Li et al. (2012) e Hu et al. (2012), os quais apresentavam grandes melhorias, tanto em termos de *leakage* quanto em termos de tempo de execução, em relação aos melhores resultados da Competição, e por isso representavam até então um novo estado da arte. Em relação aos dois últimos trabalhos, Hu et al. (2012) apresentaram os melhores resultados de *leakage* ao passo que Li et al. (2012) proporcionaram os menores tempos de execução. Após os resultados obtidos pelos dois últimos trabalhos de *sizing* discreto, surgem duas questões:

- Ainda é possível obter reduções de *leakage* significativas?
- É possível construir uma técnica que obtenha os baixos valores de potência de *leakage* obtidos por Hu et al. (2012) e o desempenho apresentado por Li et al. (2012), ou em outras palavras, um melhor compromisso entre redução de *leakage* e tempo de execução?

Este trabalho apresenta novas técnicas e métodos que ajudam a compreender e contribuir em ambos os aspectos mencionados acima.

4 *SIZING* DISCRETO BASEADO EM RELAXAÇÃO LAGRANGEANA

Chen, Chu e Wong (1999) foram pioneiros na utilização de Relaxação Lagrangeana (LR) para modelar o problema de *sizing*. Sua principal contribuição residiu na simplificação do problema aplicando as condições Karush-Kuhn-Tucker (KKT), conforme será explicado na Seção 4.2. Posteriormente, diversos outros trabalhos de *sizing* utilizaram tal simplificação (LIU; HU, 2010) (HUANG; HU; SHI, 2011) (OZDAL; BURNS; HU, 2012) (LI et al., 2012). Este capítulo apresenta uma formulação LR para o problema de *sizing* discreto, a qual também utiliza as condições KKT para simplificar o Subproblema Lagrangeano Relaxado (LRS), conforme proposto por Chen, Chu e Wong (1999). A principal contribuição científica ao estado da arte é uma nova formulação LR, a qual incorpora na função objetivo as restrições de máxima capacitância e máximo *slew* impostas pelas biblioteca *standard cell*, em adição às restrições de *timing*. Tal contribuição está relatada em artigo publicado nos anais da ICECS 2012 (LIVRAMENTO et al., 2012b).

4.1 FORMULAÇÃO DO PROBLEMA DE *SIZING* DISCRETO

O problema de *sizing* discreto pode ser definido da seguinte forma: para um dado circuito, $\forall v_i \in X$ encontre uma opção de implementação disponível na biblioteca (no caso deste trabalho, uma combinação de largura w e tensão de *threshold* u) de tal forma que a potência de *leakage* total do circuito seja minimizada, enquanto as restrições de *timing* são satisfeitas. O conjunto de Equações 4.1 a 4.7 formalizam matematicamente o problema-alvo. A função objetivo do problema, definida na Equação 4.1, corresponde ao somatório do *leakage* (p_i) das portas do circuito, cada porta acompanhada de um fator de importância α. Este fator tem por objetivo definir o peso da potência perante às restrições de atraso do problema (Equações 4.2 a 4.5). Vale ressaltar que todas as portas do circuito possuem o mesmo valor de importância de potência, ou seja, α é um valor global. As Equações 4.6 e 4.7 restringem a largura w_i e tensão de *threshold* u_i às opções disponíveis na biblioteca *standard cell*, respectivamente. Note que os conjuntos W_i e U_i são particulares de cada porta lógica.

As restrições de atraso crítico-alvo do circuito (A_0) são expressas entre as Equações 4.2 e 4.5. Particularmente, as Equações 4.2 e 4.3 definem que os tempos de chegada (de descida e de subida) nas saídas primárias não podem ser maiores do que o atraso crítico-alvo do circuito (A_0). Já as Equações 4.4 e

4.5 particionam os tempos de chegada ao longo do circuito, ou seja, relaciona o tempo de chegada na saída de cada porta lógica v_i com os tempos de chegada em suas entradas e com os seus atrasos. Em outras palavras, o tempo de chegada de descida na saída de v_i (a_i^f) recebe o máximo dentre os tempos de chegada de descida para cada arco da porta ($\max_{j \in input(v_i)} (a_j^r + d_{j \to i}^f)$), ao passo que o tempo de chegada de subida de v_i (a_i^r) recebe o máximo dentre os tempos de chegada de subida para cada arco da porta ($\max_{j \in input(v_i)} (a_j^f + d_{j \to i}^r)$)[1]. O particionamento da restrição de atraso crítico-alvo entre as saídas primárias e as portas do circuito garante que todo e qualquer caminho do circuito também irá satisfazer a tal restrição, sem que para isso seja necessário definir explicitamente tal restrição para cada caminho do circuito. Assim, enquanto uma enumeração de todos os caminhos do circuito geraria um número exponencial de restrições temporais, tal particionamento gera um número de restrições temporais que é polinomial com relação ao número de portas do circuito (CHEN; CHU; WONG, 1999) (CHINNERY; KEUTZER, 2005) (LI et al., 2012), sendo por este motivo utilizado na maioria dos trabalhos de sizing encontrados na literatura.

$$\textbf{Minimize} \quad : \quad \sum_{v_i \in X} \alpha p_i \tag{4.1}$$

$$\textbf{Sujeito a} \quad : \quad a_i^f \leq A_o, \forall v_i \in PO \tag{4.2}$$

$$: \quad a_i^r \leq A_o, \forall v_i \in PO \tag{4.3}$$

$$a_j^r + d_{j \to i}^f \leq a_i^f, \ \forall j \in input(v_i), \ \forall v_i \in (X \cup PI) \tag{4.4}$$

$$a_j^f + d_{j \to i}^r \leq a_i^r, \ \forall j \in input(v_i), \ \forall v_i \in (X \cup PI) \tag{4.5}$$

$$w_i \in W_i, \ \forall v_i \in X \tag{4.6}$$

$$u_i \in U_i, \ \forall v_i \in X \tag{4.7}$$

Conforme mencionado no Capítulo 1, as bibliotecas *standard cell* contemporâneas impõem restrições elétricas que devem ser satisfeitas para que o circuito fabricado tenha chances aceitáveis de sucesso. As restrições elétricas são avaliadas durante a etapa de ERC e incluem: corretude dos sinais de alimentação do circuito, *fanout* máximo de uma porta, máximo *slew* na entrada/saída das portas e máxima capacitância de saída de uma porta (KAHNG et al., 2011). Dentre tais restrições, as duas últimas devem ser consideradas durante a etapa de *sizing* para que sejam mantidas sob controle. Observando a

[1] As equações apresentadas neste Capítulo consideram que as portas são *negative unate*, conforme observado no Capítulo 2.

importância de tais restrições, a infraestrutura da Competição de Sizing Discreto do ISPD 2012 (OZDAL et al., 2012) também as considera. As restrições de máximo *slew* e máxima capacitância definidas pela biblioteca de células são formalizadas nas Equações 4.8, 4.9 e 4.10. As duas primeiras definem um limite único de *slew* de descida e de subida na saída de cada porta do circuito, incluindo as entradas e saídas primárias. Já a terceira restringe a máxima capacitância que conectada à saída de cada porta do circuito, ou às entradas primárias. Esta restrição é particular da opção de implementação de cada porta (no caso, a largura w_i e a tensão de threshold u_i).

$$slew_i^f \leq max\_slew, \forall v_i \in (X \cup PI) \tag{4.8}$$

$$slew_i^r \leq max\_slew, \forall v_i \in (X \cup PI) \tag{4.9}$$

$$cap_i \leq max\_cap_i, \forall v_i \in (X \cup PI) \tag{4.10}$$

O problema de *sizing* discreto para seleção simultânea da largura e da tensão de *threshold* das portas para minimizar *leakage* satisfazendo a restrição de atraso crítico-alvo, *slew* e capacitância (Equações 4.1 a 4.10), conforme definido na Competição do ISPD 2012, tem como entradas:

- *Nestlist* em nível de portas lógicas e seu respectivo grafo;
- Descrição da biblioteca *standard cell* (em formato *Liberty*) contendo as opções de implementação de cada porta, bem como tabelas com informações de atraso, *slew*, *leakage* e restrições de *slew* e capacitância;
- Arquivo contendo as restrições de *timing* do circuito no formato SDC (*Synopsys Design Constraints*);
- Arquivo contendo as informações de capacitância das interconexões do circuito no formato SPEF (*Standard Parasitic Exchange Format*).

A solução será constituída pela definição da largura da tensão de *threshold* de cada porta, objetivando:

- Minimizar o consumo de *leakage* do circuito;
- Garantir que as restrições de atraso crítico, *slew* e capacitância sejam satisfeitas.

Note que o problema de *sizing* (discreto ou contínuo) para minimização de *leakage* sujeito a restrições de atraso é um problema com dois objetivos distintos e conflitantes. Por um lado, a potência deve ser minimizada, ou seja,

as portas dos circuito devem ser trocadas por portas de maior tensão de *threshold* (ou de menor largura) o que as torna mais lentas. Por outro lado, o atraso crítico do circuito não deve ser maior do que o atraso crítico-alvo especificado no projeto e, como consequência, uma parte das portas do circuito (basicamente as pertencentes aos caminhos críticos) deve ter menor atraso (o que resulta em maior potência). Tal problema pode ser resolvido de diversas formas, das quais vale a pena destacar duas formas amplamente referenciadas na literatura:

1. Primeiramente, minimiza-se o atraso do circuito (normalmente, sem considerar a potência) com o objetivo de maximizar o *slack* positivo do circuito e, como consequência, satisfazer o atraso crítico-alvo. Em seguida, utiliza-se uma técnica para minimizar a potência do circuito, utilizando os *slacks* positivos da etapa anterior, sem violar as restrições de *timing*. A principal limitação de se resolver o problema dessa forma é a grande possibilidade da etapa de redução de potência cair rapidamente em um mínimo local e não utilizar de maneira eficiente os *slacks do circuito*. Alguns trabalhos que utilizam esta estratégia são: Coudert (1997), Rahman, Tennakoon e Sechen (2012), Li et al. (2012) e Hu et al. (2012).

2. Aplica-se Relaxação Lagrangeana, removendo as restrições de *timing* (Equações 4.2 a 4.4) e incorporando-as na função objetivo. Com isso, se resolve um novo problema cuja função objetivo captura simultaneamente a potência (*leakage*) e o atraso das portas do circuito[2]. Visto que LR é uma técnica escalável e flexível, conforme explicado na Seção 2.5, este trabalho propõe uma formulação alternativa LR para o problema de *sizing* discreto, conforme descrito na próxima seção.

4.2 *SIZING* DISCRETO BASEADO EM RELAXAÇÃO LAGRANGEANA

Relaxação Lagrangeana (LR) é uma técnica bastante utilizada no contexto de *sizing*. A maioria dos trabalhos encontrados na literatura utilizam LR para *sizing* contínuo baseado em modelos de atraso posinomiais, e.g., Chen, Chu e Wong (1999), Tennakoon e Sechen (2002), Wang, Das e Zhou (2009). Tais modelos, sob uma transformação de variáveis, tornam o problema convexo (BOYD; VANDENBERGHE, 2004). Por outro lado, o problema de *sizing* no fluxo *standard cell* é inerentemente discreto e provado ser NP-Difícil

[2]Vale a pena ressaltar a similaridade com o problema do caminho mínimo apresentado na Seção 2.5.

(LI, 1994). Alguns trabalhos publicados recentemente aplicam LR diretamente no domínio discreto, e.g., Liu e Hu (2010), Huang, Hu e Shi (2011), Ozdal, Burns e Hu (2012), Li et al. (2012).

Chen, Chu e Wong (1999) foram os primeiros a aplicar LR no problema de *sizing* removendo as restrições de *timing* do circuito e adicionando-as à função objetivo. Seguindo a mesma ideia, a Função Lagrangeana (LF) associada ao PP formalizado entre as Equações 4.1 a 4.5 é definida conforme a Equação 4.11. Note que há um Multiplicador de Lagrange (λ) associado a cada restrição de tempo de chegada, seja ele de descida ou subida. Observe também que as restrições que limitam w_i e u_i das portas somente às opções disponíveis na biblioteca não são incorporadas na nova função objetivo, visto que elas podem ser manipuladas mais facilmente restringindo-se o domínio do problema aos valores discretos[3]. Por questão de clareza, a LF é definida em função de vetores que representam o conjunto das larguras $\vec{w}$, tensões de *threshold* $\vec{u}$ e tempos de chegada (descida e subida) $\vec{a}$ de cada uma das portas do circuito, conforme Equação 4.11.

$$
\begin{aligned}
L_\lambda(\vec{w},\vec{u},\vec{a}) \quad : \quad & \sum_{v_i \in X} \alpha p_i + \sum_{v_i \in PO} \lambda^f_{po}(a^f_i - A_o) + \sum_{v_i \in PO} \lambda^r_{po}(a^r_i - A_o) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^f_{j \to i}(a^r_j + d^f_{j \to i} - a^f_i) \Big) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^r_{j \to i}(a^f_j + d^r_{j \to i} - a^r_i) \Big) \qquad (4.11)
\end{aligned}
$$

Como uma das contribuições deste trabalho, incorporam-se as restrições de máxima capacitância e máximo *slew* impostas pelas bibliotecas *standard cell* (Equações 4.8 a 4.10) na função objetivo, originando a formulação da Equação 4.12. Observe que cada uma das restrições de *slew* de descida e subida está multiplicada por um LM γ^f_i e γ^r_i, respectivamente, enquanto cada uma das restrições de capacitância está multiplicada pelo LM β_i. Apesar do trabalho proposto por Jiang, Chang e Jou (2000) ter relaxado restrições adicionais às de *timing*, como por exemplo *crosstalk*, este é o primeiro trabalho a propor a relaxação das restrições de máxima capacitância e máximo *slew* no contexto de *sizing* baseado em Relaxação Lagrangeana.

[3]No caso de *sizing* contínuo, o domínio pode se restringir aos intervalos $w_{min} \leq w_i \leq w_{max}$ e $u_{min} \leq u_i \leq u_{max}$ (CHEN; CHU; WONG, 1999).

$$
\begin{aligned}
L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a}) \quad : \quad & \sum_{v_i \in X} \alpha p_i + \sum_{v_i \in PO} \lambda^f_{po}(a^f_i - A_o) + \sum_{v_i \in PO} \lambda^r_{po}(a^r_i - A_o) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{v_j \in input(v_i)} \lambda^f_{j \to i}(a^r_j + d^f_{j \to i} - a^f_i) \Big) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{v_j \in input(v_i)} \lambda^r_{j \to i}(a^f_j + d^r_{j \to i} - a^r_i) \Big) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^f_i (slew^f_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^r_i (slew^r_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \beta_i (cap_i - max\_cap_i) \qquad (4.12)
\end{aligned}
$$

A partir da relaxação das restrições de *timing*, *slew* e capacitância, e incorporação na Função Lagrangeana ($L_{\lambda,\gamma,\beta}$), originam-se dois problemas:

1. O Subproblema Lagrangeano Relaxado (LRS), formalizado na Equação 4.13 e rotulado por *LRS*1, o qual deve atribuir uma largura e uma tensão de *threshold* a cada porta do circuito objetivando minimizar a LF da Equação 4.12. Também faz parte do *LRS*1 a atribuição dos tempos de chegada ao longo do circuito de forma a distribuir os *slacks* da melhor maneira possível. É importante ressaltar que os valores dos Multiplicadores de Lagrange estão fixos durante a resolução do *LRS*1. Para resolver o LRS diretamente no domínio discreto, o qual é um problema NP-difícil, diversas técnicas foram utilizadas, tais como Programação Dinâmica (LIU; HU, 2010), (OZDAL; BURNS; HU, 2012), heurísticas gulosas (HUANG; HU; SHI, 2011), dentre outras.

$$
\begin{aligned}
\textbf{LRS1} - \textit{Minimize} \quad & : \quad L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a}) \\
\textit{Sujeito a} \quad & : \quad w_i \in W_i, \forall v_i \in X \\
& \qquad u_i \in U_i, \forall v_i \in X \qquad (4.13)
\end{aligned}
$$

2. O Problema Lagrangeano Dual (LDP), formalizado na Equação 4.14, tem por objetivo a atualização dos LMs λ, γ e β, visando maximizar a solução encontrada pelo *LRS*1, rotulada por $Q(\vec{\lambda},\vec{\gamma},\vec{\beta})$. Observe que no caso do LDP, tanto a largura quanto a tensão de *threshold* das portas são fixas. Para resolver o LDP no contexto de *sizing*, o método

do subgradiente é normalmente utilizado para atualizar os LMs referentes às restrições de *timing* (CHEN; CHU; WONG, 1999). Neste método o subgradiente da transição de descida nas saídas primárias é $a_i^f - A_o$, ao passo que o subgradiente de descida das demais portas é $a_j^r + d_{j\to i}^f - a_i^f$. Para se obter os subgradientes de subida, deve-se inverter as transições[4].

$$\begin{aligned} \textbf{LDP} - \textit{Maximize} \quad &: \quad Q(\vec{\lambda},\vec{\gamma},\vec{\beta}) \\ \textit{Sujeito a} \quad &: \quad \lambda \geq 0,\ \gamma \geq 0,\ \beta \geq 0 \end{aligned} \tag{4.14}$$

Para remover os tempos de chegada do *LRS*1 (Equação 4.13), e, por consequência simplificá-lo, no presente trabalho aplicam-se as condições de otimalidade de *Karush-Kuhn-Tucker (KKT)* nas restrições de *timing*, conforme proposto por Chen, Chu e Wong (1999). As condições de otimalidade KKT foram aplicadas pela primeira vez no problema de *sizing* por Chen, Chu e Wong (1999). Posteriormente, uma série de outros trabalhos também utilizaram tal simplificação, tanto no domínio contínuo (TENNAKOON; SECHEN, 2002) (TENNAKOON; SECHEN, 2008), quanto no domínio discreto (LIU; HU, 2010) (OZDAL; BURNS; HU, 2012) (LI et al., 2012). As condições KKT implicam que a solução ótima $(\vec{w},\vec{u},\vec{a})^*$ do problema primal deve satisfazer: $\frac{\partial L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})^*}{\partial a_i^r} = 0$ e $\frac{\partial L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})^*}{\partial a_i^f} = 0, \forall v_i \in (X \cup PI)$. A aplicação das condições de otimalidade implica que os Multiplicadores de Lagrange (λ) associados a cada uma das restrições de tempo de chegada devem satisfazer as condições de fluxo formalizadas nas Equações 4.15 e 4.16, a qual considera apenas portas *negative unate*. As condições de fluxo definem que, para cada porta do circuito, o somatório de seus LMs de descida (subida) deve ser igual ao somatório dos LMs de subida (descida) referentes às suas portas *fanout*.

$$\sum_{j \in input(v_i)} \lambda_{j\to i}^f = \sum_{v_k \in fanout(v_i)} \lambda_{i\to k}^r,\ \forall v_i \in (X \cup PI) \tag{4.15}$$

$$\sum_{j \in input(v_i)} \lambda_{j\to i}^r = \sum_{v_k \in fanout(v_i)} \lambda_{i\to k}^f,\ \forall v_i \in (X \cup PI) \tag{4.16}$$

[4]Observando as limitações do método do subgradiente quando aplicado ao problema de *sizing*, alguns trabalhos propuseram o uso de métodos alternativos para atualizar os LMs no contexto contínuo (TENNAKOON; SECHEN, 2002) e no contexto discreto (HUANG; HU; SHI, 2011).

Para melhor entender a aplicação das condições KKT ao *LRS*1, primeiramente será apresentada a Equação 4.17 na qual as restrições de tempo de chegada definidas na Equação 4.12 foram reorganizadas. Aplicando-se as condições de otimalidade KKT no *LRS*1, conforme proposto em Chen, Chu e Wong (1999), os tempos de chegada são eliminados da Função Lagrangeana $L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a})$ da Equação 4.17. Além disso, é possível eliminar os termos $\sum_{v_i \in PO} \lambda^f_{po} A_o$ e $\sum_{v_i \in PO} \lambda^r_{po} A_o$ da LF reescrita na Equação 4.17 uma vez que eles são fixos, e por este motivo, não são alterados durante o processo de resolução do LRS. Como resultado, a nova LF $L_{\lambda,\gamma,\beta}(\vec{w},\vec{u})$ não depende mais dos tempos de chegada, conforme formalizado na Equação 4.18. Agora, considere que $\Omega_\lambda = \{\vec{\lambda}$ tal que $\lambda \geq 0$ e $\vec{\lambda}$ satisfaça as condições de otimalidade KKT$\}$. Então, para qualquer $\vec{\lambda} \in \Omega_\lambda$, resolver o *LRS*1 (Equação 4.13) é equivalente a resolver o *LRS*2, formalizado na Equação 4.19. Note que o *LRS*2 tem por objetivo selecionar a largura w e a tensão de *threshold* das portas do circuito de forma a minimizar a Função Lagrangeana simplificada.

$$
\begin{aligned}
L_{\lambda,\gamma,\beta}(\vec{w},\vec{u},\vec{a}) \quad : \quad & \sum_{v_i \in X} \alpha p_i \; + \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{k \in fanout(v_i)} \lambda^r_{i \to k} - \sum_{j \in input(v_i)} \lambda^f_{j \to i} \Big) a^f_i \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{k \in fanout(v_i)} \lambda^f_{i \to k} - \sum_{j \in input(v_i)} \lambda^r_{j \to i} \Big) a^r_i \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^f_{j \to i} d^f_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^r_{j \to i} d^r_{j \to i} \Big) \\
& - \sum_{v_i \in PO} \lambda^f_{po} A_o - \sum_{v_i \in PO} \lambda^r_{po} A_o \\
& + \sum_{v_i \in (X \cup PI)} \gamma^f_i (slew^f_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^r_i (slew^r_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \beta_i (cap_i - max\_cap_i) \qquad (4.17)
\end{aligned}
$$

$$
\begin{aligned}
L_{\lambda,\gamma,\beta}(\vec{w},\vec{u}) \quad : \quad & \sum_{v_i \in X} \alpha p_i \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^f_{j \to i} d^f_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^r_{j \to i} d^r_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^f_i (slew^f_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^r_i (slew^r_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \beta_i (cap_i - max\_cap_i) \qquad (4.18)
\end{aligned}
$$

$$
\begin{aligned}
\textbf{LRS2} - \; Minimize \quad & : \quad L_{\lambda,\gamma,\beta}(\vec{w},\vec{u}) \\
Sujeito\ a \quad & : \quad w_i \in W_i \; and \; u_i \in U_i, \forall v_i \in V \qquad (4.19)
\end{aligned}
$$

O Algoritmo 3 apresenta a sequência de passos utilizados em diversos trabalhos na literatura para resolver o problema de *sizing* baseado em Relaxação Lagrangeana (CHEN; CHU; WONG, 1999) (TENNAKOON; SECHEN, 2002) (HUANG; HU; SHI, 2011) (OZDAL; BURNS; HU, 2012) (LI et al., 2012) (LIVRAMENTO et al., 2012b) (LIVRAMENTO et al., 2013). Observe que o Algoritmo 3 recebe como entradas um DAG do circuito ($G(V,E)$), uma biblioteca *standard cell* (L) e um atraso crítico-alvo (A_o) e entrega como saída este mesmo grafo onde a opção de implementação de cada porta está definida. Na linha 2 os LMs relacionados às restrições de *timing* são inicializados com um vetor que satisfaça às condições KKT[5]. Após a inicialização, o *LRS*2 (linha 4) e o LDP (linha 6) são resolvidos iterativamente até que um número máximo de iterações pré-definido seja atingido. Além disso, há um passo de atualização das informações de *timing* por meio de uma ferramenta de STA (linha 5) e um passo para distribuir os LMs (linha 7) após o LDP.

Por fim, as opções de implementação do circuito com menor potência de *leakage* e sem violações sobre todas as iterações são assumidas como finais. O próximo capítulo propõe uma nova heurística para resolver o *LRS*2. Além disso, são apresentados os algoritmos para resolver o *LDP* e para distribuir os LMs relacionados às restrições de *timing*.

[5]No caso do presente trabalho, os Multiplicadores de Lagrange relacionados às restrições de capacitância e *slew* também devem ser inicializados.

Algoritmo 3: DISCRETE_SIZING_BASED_ON_LR

Input : $G(V,E)$, L, A_o
Output: *w and u* option $\forall v_i \in X$

```
iteration ← 0;
λ⃗ ← initial positive vector satisfying KKT conditions (Equations 4.15 and 4.16);
while iteration < it_max do
    solve LRS2 to find w⃗ e u⃗;
    update circuit timing information;
    solve LDP to update Lagrange Multipliers;
    Distribute Lagrange Multipliers to satisfy KKT conditions;
    iteration++;
end
```

5 HEURÍSTICA GULOSA BASEADA EM RELAXAÇÃO LAGRANGEANA PARA O PROBLEMA DE *SIZING* DISCRETO

Este capítulo apresenta uma nova técnica para resolver o problema de *sizing* discreto baseado em Relaxação Lagrangeana. As principais contribuições científicas ao estado da arte, relatadas em artigo publicado nos anais da DATE 2013 (LIVRAMENTO et al., 2013), são:

1. Uma heurística gulosa rápida para resolver a formulação *LRS*2, proposta na Seção 4.2, a qual se baseia apenas em informações locais para guiar as decisões do algoritmo. Tais decisões locais são possíveis visto que o atraso de uma porta é função de sua opção de implementação e da opção de suas portas vizinhas (*slew* de entrada e capacitância de saída) e as informações globais de atraso do circuito, representadas pelos tempos de chegada, foram eliminadas do *LRS*2 após a aplicação das condições de otimalidade KKT. Também se propõe uma heurística complementar baseada em Relaxação Lagrangeana para remover as violações de máxima capacitância e de máximo *slew* remanescentes da heurística gulosa;

2. Uma técnica de *sizing* discreto que combina o uso do método do subgradiente modificado proposto por Tennakoon e Sechen (2002), com o escalonamento do fator de importância da potência proposto por Tennakoon e Sechen (2008);

3. Validação experimental rigorosa utilizando a infraestrutura da Competição de *Sizing* Discreto do ISPD 2012, a qual demonstrou que a técnica proposta nesse capítulo proporciona valores de *leakage* significativamente inferiores aos obtidos pelos três primeiros colocados da referida Competição de *Sizing*, com tempos de execução que são pelo menos 30 vezes inferiores.

5.1 A HEURÍSTICA PROPOSTA

Como pode ser observado na formulação do *LRS*2 (Equação 4.19), a informação de atraso global do circuito (tempos de chegada) foi eliminada do problema como resultado da aplicação das condições de otimalidade KKT. Portanto, a função objetivo, reproduzida novamente na Equação 5.1, deve minimizar o somatório dos atrasos, o somatório das potências de *leakage* e

somatório das violações sobre todas as portas do circuito. A potência de *leakage* de uma porta depende somente de sua opção de implementação. Por outro lado, os atrasos das portas do circuito não são independentes entre si, visto que o atraso de uma porta é afetado pela opção de implementação de suas vizinhas devido à influência no *slew* de entrada e na capacitância de saída. As violações de máxima capacitância e máximo slew dependem da opção de implementação da própria porta, sendo que a primeira é afetada também pelas opções de suas *fanouts* imediatas, ao passo que a segunda é influenciada também pelas *fanins* e *fanouts* imediatas.

$$
\begin{aligned}
L_{\lambda,\gamma,\beta}(\vec{w},\vec{u}) \quad : \quad & \sum_{v_i \in X} \alpha p_i \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^f_{j \to i} d^f_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^r_{j \to i} d^r_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^f_i (slew^f_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^r_i (slew^r_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \beta_i (cap_i - max\_cap_i) \qquad (5.1)
\end{aligned}
$$

Alguns trabalhos recentes de *sizing* (OZDAL; BURNS; HU, 2012) (LIU; HU, 2010) utilizaram programação dinâmica com o objetivo de considerar essa dependência entre os atrasos das portas. Entretanto, estes mesmos trabalhos gastam um alto esforço computacional tentando levar essas dependências globais em conta, esquecendo-se do fato de que a parte mais significativa dessas dependências é local, uma vez que uma porta afeta principalmente o atraso de seus *fanins* e *fanouts* imediatos (OZDAL; BURNS; HU, 2012) (LI et al., 2012) (LIVRAMENTO et al., 2012a). O fato de um circuito digital possuir diversas reconvergências conduz à necessidade de adaptações na heurística de programação dinâmica, — como a decomposição do circuito em árvore, proposta por Ozdal, Burns e Hu (2012) — o que resulta em soluções subótimas. Os experimentos realizados no presente trabalho mostram que uma seleção local, a qual leva em conta somente o impacto nas portas *fanins* e *fanouts* imediatas, é rápida e eficiente para resolver a formulação *LRS*2 — ao contrário da heurística baseada em programação dinâmica que é muito custosa. Obviamente, a escolha da opção de implementação de porta com maior ou menor atraso é guiada principalmente pela criticalidade do ca-

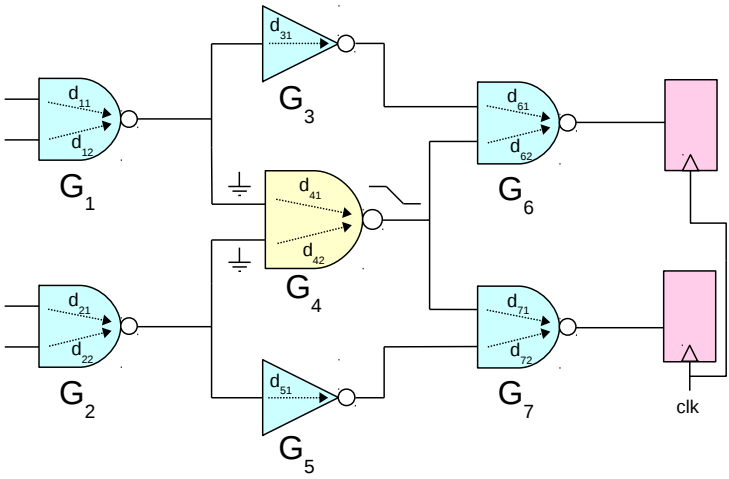


Figura 11: Exemplo de circuito para facilitar o entendimento da heurística proposta.

minho ao qual a porta pertence, cuja informação está representada pelos Multiplicadores de Lagrange (λ).

Partindo de uma solução na qual a cada porta do circuito já está assinalada uma opção de implementação dentre as disponíveis na biblioteca *standard cell* utilizada (referenciada por *opção atual*), a ideia básica da heurística aqui proposta reside em percorrer o circuito em ordem topológica e, para cada porta, escolher, dentre as opções de implementação alternativas, aquela opção que resultar em menor custo. O custo de uma opção alternativa, por sua vez, é avaliado com base no impacto da nova carga capacitiva nos *fanins* imediatos da porta e no impacto da variação do *slew* de saída em seus *fanouts* imediatos. Apesar de Chen, Chu e Wong (1999) já terem proposto uma heurística gulosa para o problema de *sizing* baseado em Relaxação Lagrangeana, ela foi desenvolvida para *sizing* contínuo onde a largura de cada porta era escolhida considerando todas as outras portas fixas, ou seja, mesma largura da iteração anterior. Diferentemente, a heurística gulosa proposta no presente trabalho opera em ordem topológica, escolhendo a opção de implementação (w_i, u_i) de uma porta somente quando a opção de cada uma de suas predecessoras já tiver sido escolhida. Além disso, a heurística proposta também considera o impacto da troca da opção de uma porta nos atrasos das portas que são *fa-*

nin e *fanout* imediatos à porta trocada. Esta última característica é extrema importância, já que a opção de implementação da porta pode degradar consideravelmente o atraso de suas portas vizinhas. A estratégia de considerar apenas os *fanins* e *fanouts* imediatos da porta trocada, adotada na heurística proposta, advém do fato de que o maior impacto da troca de opção de uma porta ocorre justamente em seus *fanins* e *fanouts* imediatos, ao passo que o impacto nas demais portas do cone lógico relacionado (como G_3 e G_5, que são *fanouts* dos *fanins*, e possíveis *fanouts* dos *fanouts*) é muito menor, como já observado nos trabalhos de Coudert (1997) Ozdal, Burns e Hu (2012), Li et al. (2012) e Livramento et al. (2012a).

Para ilustrar o ideia da heurística proposta, suponha que se deseja avaliar o custo decorrente de se trocar a opção atual da porta G_4 da Figura 11 por alguma das opções de implementação alternativas disponíveis. Para isso, as avaliações descritas a seguir serão feitas para cada uma das opções alternativas à opção atual de G_4. Primeiramente, são estimados os impactos da variação da capacitância de entrada de G_4 nos atrasos e nos *slews* de G_1 e de G_2 caso a opção atual de G_4 fosse trocada pela opção alternativa em questão. Também são avaliadas as possíveis violações de *slew* e de capacitância em G_1 e em G_2 que ocorreriam. Em seguida, são estimados os novos atrasos e potências de leakage de G_4 e são avaliadas possíveis violações em G_4. Por último, são avaliados os impactos da variação do slew de saída de G_4 nos atrasos d_{62} de G_6 e d_{71} de G_7. Finalmente, a nova opção de implementação para G_4 será aquela que obtiver o menor custo dentre as opções avaliadas, podendo inclusive ser a própria opção atual (ou seja, a troca não ocorre). Vale lembrar que cada um dos atrasos é multiplicado por seu respectivo Multiplicador de Lagrange, conforme Equação 5.1.

A heurística gulosa topológica proposta é detalhada no Algoritmo 4 e recebe como entrada o DAG do circuito, a biblioteca *standard cell*, o vetor de LMs de *timing*, o vetor de LMs de capacitância e a importância da potência. Observe que o impacto no atraso e nas vioações das portas *fanins* à porta trocada é avaliado entre as linhas 3 e 8. Já o impacto na potência, atraso e possíveis violações da porta em questão é estimado nas linhas 9 e 10, ao passo que o impacto nos *fanouts* é estimado entre as linha 11 e 13. Uma característica importante do Algoritmo 4 reside no fato de que eventuais violações de máxima capacitância na porta trocada ou em alguma de suas portas *fanin* são consideradas (nas linhas 10 e 7, respectivamente). Nos experimentos realizados neste trabalho (Seção 5.3) observou-se que não é necessário incluir-se o custo das violações de *slew* no custo total da opção de implementação de uma porta. Ao contrário, considerando-se no custo apenas as violações de máxima capacitância é suficiente para controlar as violações de *slew* e capacitância, conforme já havia sido constatado por Hu et al. (2012).

Após a escolha da largura e tensão de *threshold* de todas as portas do circuito, as violações remanescentes de capacitância (ou *slew*) são reduzidas (ou eliminadas) pela heurística complementar proposta, invocada na linha 22.

Algoritmo 4: SOLVE_LRS

Input : $G(V,E)$, L, $\vec{\lambda}$, $\vec{\beta}$, α
Output: *w and u* option $\forall v_i \in X$

1 **for** $\forall v_i \in X$ *in topological order* **do**
2 $bestCost \leftarrow \infty$;
3 **for** $\forall w_i \in W_i$ *and* $u_i \in U_i$ **do**
4 **for** $\forall v_j \in fanin(v_i)$ **do**
5 update cap_j w.r.t. current w_i, u_i;
6 $cost \leftarrow \sum_{g \in input(v_j)} (d^f_{g \to j} * \lambda^f_{g \to j}) + \sum_{g \in input(v_j)} (d^r_{g \to j} * \lambda^r_{g \to j})$;
7 $cost \leftarrow cost + \beta_j * (cap_j - max\_cap_j)$;
8 **end**
9 $cost \leftarrow cost + \alpha p_i + \sum_{j \in input(v_i)} (d^f_{j \to i} * \lambda^f_{j \to i}) + \sum_{j \in input(v_i)} (d^r_{j \to i} * \lambda^r_{j \to i})$;
10 $cost \leftarrow cost + \beta_i * (cap_i - max\_cap_i)$;
11 **for** $\forall v_k \in fanout(v_i)$ **do**
12 $cost \leftarrow cost + (d^f_{i \to k} * \lambda^f_{i \to k}) + (d^r_{i \to k} * \lambda^r_{i \to k})$;
13 **end**
14 **if** $cost < bestCost$ **then**
15 $bestCost \leftarrow cost$;
16 $best_w \leftarrow w_i$;
17 $best_u \leftarrow u_i$;
18 **end**
19 **end**
20 implement v_i by $best_w$ and $best_u$;
21 **end**
22 **Call** FIX_VIOLATIONS(V, L, α);

O Algoritmo 5 apresenta a heurística complementar baseada em Relaxação Lagrangeana para remover as violações de capacitância (e consequentemente, de *slew*). Ele percorre o grafo do circuito em ordem topológica reversa, buscando aumentar a largura das portas que violam as restrições de máxima capacitância. A prioridade é dada às larguras que resolvam as

Algoritmo 5: FIX_VIOLATIONS

Input : $G(V,E)$, L, $\vec{\lambda}$, $\vec{\beta}$, α
Output: w option $\forall v_i \in X$

1 **for** $\forall v_i \in X$ *in reverse topological order* **do**
2 **if** $cap_i > max\_cap_i$ **then**
3 $bestCost \leftarrow \infty$;
4 **for** $\forall w_i \in W_i$ **do**
5 $cost \leftarrow$ $\alpha p_i + \sum_{j \in input(v_i)} (d^f_{j \to i} * \lambda^f_{j \to i}) + \sum_{j \in input(v_i)} (d^r_{j \to i} * \lambda^r_{j \to i})$;
6 $cost \leftarrow cost + \beta_i * (cap_i - max\_cap_i)$;
7 **if** $cap_i \leq max\_cap_i$ *w.r.t current* w_i **then**
8 **if** $cost < bestCost$ **then**
9 $bestCost = cost$;
10 $best_w = w_i$;
11 **end**
12 **end**
13 **end**
14 implement v_i by $best_w$;
15 **end**
16 **end**

violações com o menor custo. Ao considerar no custo de uma porta os seus atrasos e sua potência de *leakage*, é possível remover as violações e, ao mesmo tempo, não comprometer a qualidade da solução obtida pelo Algoritmo 4. Isto ocorre porque a escolha da opção de implementação leva em consideração as informações de *timing*, representadas pelos LMs λ, e o fator de importância da potência α. A inclusão de possíveis violações de máxima capacitância no custo (linha 6) tem por objetivo considerar casos em que a violação não possa ser completamente removida na iteração atual.

5.2 TÉCNICA DE *SIZING* DISCRETO UTILIZANDO A HEURÍSTICA PROPOSTA

O procedimento de alto-nível da técnica proposta é apresentado no Algoritmo 6 (similar ao Algoritmo 3), cujo laço principal, entre as linhas 5 e 12, é executado um número fixo de iterações (it_{max}). Partindo de uma solução inicial com todas as portas na menor configuração de *leakage* (i.e., menor

Algoritmo 6: DISCRETE_GATE_SIZING

Input : $G(V,E)$, L, α, A_o
Output: Optimized $G(V,E)$

1 $iteration \leftarrow 0$;
2 assign all $v_i \in X$ to w and u with minimum leakage power;
3 $\vec{\lambda} \leftarrow$ initial positive vector satisfying KKT conditions;
4 $\vec{\beta} \leftarrow$ initial positive vector;
5 **while** $iteration < it_{max}$ **do**
6 STATIC_TIMING_ANALYSIS(G, L, A_o);
7 $\alpha \leftarrow \alpha * \frac{A_o}{max(a_i)}, \forall v_i \in PO$;
8 SOLVE_LRS(G, L, $\vec{\lambda}$, $\vec{\beta}$, α);
9 SOLVE_LDP(G, L, $\vec{\lambda}$, $\vec{\beta}$);
10 DISTRIBUTE_TIMING_LMs(G, $\vec{\lambda}$);
11 $iteration++$;
12 **end**

w e maior u), o laço principal invoca a ferramenta de STA para atualizar os tempos de chegada no circuito. Em seguida, atualiza o fator de importância da potência de acordo com o caminho crítico do circuito e resolve o *LRS*2 utilizando o Algoritmo 4, a fim de definir a largura e a tensão de *threshold* de cada porta. Em seguida, os Multiplicadores de Lagrange são atualizados buscando de refletir o impacto das novas opções de implementação nas restrições do problema, conforme detalhado no Algoritmo 7. Ao contrário do método do subgradiente tradicional[1] para atualizar os LMs de *timing*, o qual se baseia em um valor único de passo ρ_k para todas as portas do circuito, o método proposto por Tennakoon e Sechen (2002) é sensível às informações locais de atraso do circuito, conforme pode ser observado na Equação 5.2. Para atualizar os LMs relacionados às restrições de máxima capacitância (Algoritmo 7, linha 8), utilizou-se um passo θ_k em função da máxima capacitância definida pela opção de implementação de v_i, conforme a Equação 5.3. Note que os LMs relacionados às restrições de máximo *slew* (γ) não são atualizados no LDP, visto que não são incorporados no custo da troca da opção da porta, conforme já mencionado na seção anterior.

[1] Utilizado por Chen, Chu e Wong (1999), Ozdal, Burns e Hu (2012), dentre outros.

Algoritmo 7: SOLVE_LDP

Input : $G(V,E), L, \vec{\lambda}, \vec{\beta}$
Output: updated $\vec{\lambda}$, updated $\vec{\beta}$

for $\forall v_i \in V$ **do**
  if $v_i \in PO$ **then**
    $\lambda^f_{po} \leftarrow \lambda^f_{po} * (\frac{a^f_i}{A_o})$;
    $\lambda^r_{po} \leftarrow \lambda^r_{po} * (\frac{a^r_i}{A_o})$;
  else
    $\lambda^f_{j\to i} \leftarrow \lambda^f_{j\to i} * (\frac{a^r_j + d^f_{j\to i}}{a^f_i})$;
    $\lambda^r_{j\to i} \leftarrow \lambda^r_{j\to i} * (\frac{a^f_j + d^r_{j\to i}}{a^r_i})$;
    $\beta_i \leftarrow \beta_i * (\frac{cap_i}{max\_cap_i})$;
  end
end

$$\rho_k = \frac{\lambda_{j\to i}}{a_i},\ \forall j \in input(v_i), \forall v_i \in (X \cup PI) \tag{5.2}$$

$$\theta_k = \frac{\beta_i}{max\_cap_i}, \forall v_i \in (X \cup PI) \tag{5.3}$$

A distribuição dos LMs (λ) para satisfazer as condições de KKT (Algoritmo 8) é feita de forma similar ao trabalho de Tennakoon e Sechen (2002). Tal distribuição percorre o grafo do circuito em ordem topológica reversa, visitando cada v_i para distribuir a soma dos LMs de saída proporcionalmente para os LMs das entradas. Com isso, os λs das saídas primárias correlatam com a restrição de atraso crítico-alvo (A_o), ao passo que os λs nas demais portas representam suas criticalidades relativas, sendo por isso usados para definir a distribuição dos LMs das saídas primárias ao longo dos caminhos do circuito, buscando satisfazer as condições KKT.

Algoritmo 8: DISTRIBUTE_TIMING_LMs

Input : $G(V,E), \vec{\lambda}$
Output: updated $\vec{\lambda}$
1 **for** $\forall v_i \in V$ **do**
2 $\quad \mu^r_{j\to i} \leftarrow \sum_{v_j \in fanin(v_i)} \lambda^r_{j\to i}$;
3 $\quad \mu^f_{j\to i} \leftarrow \sum_{v_j \in fanin(v_i)} \lambda^f_{j\to i}$;
4 $\quad \mu^r_{i\to k} \leftarrow \sum_{v_k \in fanout(v_i)} \lambda^r_{i\to k}$;
5 $\quad \mu^f_{i\to k} \leftarrow \sum_{v_k \in fanout(v_i)} \lambda^f_{i\to k}$;
6 $\quad$ **for** $\forall j \in input(v_i)$ **do**
7 $\quad\quad \lambda^f_{j\to i} \leftarrow \frac{\lambda^f_{j\to i}}{\mu^f_{j\to i}} * \mu^r_{i\to k}$;
8 $\quad\quad \lambda^r_{j\to i} \leftarrow \frac{\lambda^r_{j\to i}}{\mu^r_{j\to i}} * \mu^f_{i\to k}$;
9 $\quad$ **end**
10 **end**

5.3 RESULTADOS EXPERIMENTAIS

Os Algoritmos 6, 4, 5, 7 e 8 apresentados no presente capítulo foram implementados na linguagem C++ e compilados usando a versão 4.4.5 do gcc. Para computar as informações de *timing* do circuito (Algoritmo 6, linha 6), foi utilizada a ferramenta de STA dedicada mencionada na Seção 1.6 (Infraestrutura Experimental), a qual está em conformidade com bibliotecas *standard cell* realistas. Para a geração dos resultados, foi utilizada a infraestrutura experimental, também descrita na Seção 1.6.

A Tabela 3 apresenta o número de portas e o atraso crítico-alvo A_o (em ps) de cada um dos circuitos da Competição de *Sizing* Discreto do ISPD2012. Ela também apresenta, para cada um dos circuitos, o valor de *leakage* (em W), o *slack* do caminho crítico (em *ps* e em porcentagem de A_o) e o tempo de execução (em horas) resultantes da aplicação da técnica proposta após 60 iterações[2] do Algoritmo 6. Note que o *slacks* remanescentes após a aplicação da técnica proposta em cada circuito indicam que ainda há espaço de otimiza-

[2]Partindo do número de iterações proposto por Ozdal, Burns e Hu (2012) (30), neste trabalho testaram-se diversas possibilidades, sendo 60 iterações aquela que levou ao melhor compromisso entre tempo de execução e redução de potência.

Tabela 3: Características dos circuitos da Competição do ISPD 2012 e resultados da técnica proposta.

	Caract. dos Circuitos		Resultados da Técnica Proposta		
Circuitos	**Número de Portas**	**Atraso Crítico-Alvo A_o [ps]**	**Potência de Leakage [W]**	***Slack* do Caminho Crít. [ps] (% of A_o)**	**Tempo de Execução [h]**
DMA_slow	25K	900	0.165	7.91 (0.88)	0.02
DMA_fast	25K	770	0.380	11.23 (1.46)	0.02
pci_bridge32_slow	33K	720	0.124	3.03 (0.42)	0.03
pci_bridge32_fast	33K	660	0.225	2.5 (0.38)	0.03
des_perf_slow	111K	900	0.807	6.96 (0.77)	0.10
des_perf_fast	111K	735	1.970	1.24 (0.17)	0.10
vga_lcd_slow	165K	700	0.359	14.9 (2.13)	0.13
vga_lcd_fast	165K	610	0.534	1.85 (0.30)	0.13
b19_slow	219K	2500	0.607	16.25 (0.65)	0.26
b19_fast	219K	2100	0.973	5.95 (0.28)	0.26
leon3mp_slow	649K	1800	1.363	8.91 (0.50)	0.58
leon3mp_fast	649K	1500	1.757	1.67 (0.11)	0.6
netcard_slow	959K	1900	1.763	14.69 (0.77)	0.85
netcard_fast	959K	1200	1.909	11.73 (0.98)	0.85

ção. É importante ressaltar que todos os resultados foram obtidos mediante a execução dos *scripts* de validação fornecidos pela Competição do ISPD 2012.

A Tabela 4, por sua vez, apresenta a comparação dos resultados de *leakage* da técnica proposta com aqueles obtidos pelas três equipes melhores classificadas na Competição do ISPD 2012 (1^o NTUgs, 2^o UFRGS-Brazil e 3^o PowerValve)[3]. Conforme mencionado na Seção 3.2.9, estas equipes utilizam técnicas de *sizing* do estado da arte, distintas entre si. A mesma tabela também mostra o menor valor de *leakage* obtido para cada circuito (rotulado por "Menor"), considerando todas as equipes participantes da Competição do ISPD 2012.

Considerando os resultados individuais de *leakage*, a técnica proposta supera as três melhores equipes da Competição do ISPD 2012 para todos os circuitos com mais de 100K portas, com exceção do circuito des_perf_-slow. Ao contrário das equipes NTUgs e UFRGS-Brazil, a técnica proposta não violou nenhuma restrição (nem de *timing*, tampouco de máximo *slew* e máxima capacitância). A média dos resultados de *leakage* obtidos por cada técnica é apresentada na última linha da Tabela 4. Quando comparada com os resultados obtidos pelas três melhores equipes da Competição do ISPD 2012, a técnica proposta obteve valores de *leakage* que são, em média,18.9%, 16.7% e 43.8% menores, respectivamente.

[3]É importante observar que a configuração da máquina utilizada na Competição do ISPD2012 (2 CPUs Intel®Xeon®E5675 @ 3.06GHz com 96GB RAM (OZDAL et al., 2012)) é ligeiramente superior àquela utilizada nos experimentos aqui reportados, a qual está descrita na Seção 1.6.

Tabela 4: Comparação de potência de *Leakage* com as três melhores equipes da Competição do ISPD 2012 (resultados das equipes disponibilizados em domínio público por ISPD (2012)). "X" corresponde aos resultados com violações. [a] Menor valor de *leakage* obtido em cada circuito, considerando-se todas as equipes competidoras. [b] Média calculada ignorando-se resultados com violações.

	Resultados de Potência de *Leakage* ([W])				
Circuitos	**NTUgs**	**UFRGS-Brazil**	**PowerValve**	**Menor[a]**	**Técnica Proposta**
DMA_slow	0.205	0.158	0.147	0.147	0.165
DMA_fast	0.511	0.323	0.312	0.312	0.380
pci_bridge32_slow	0.203	0.115	0.116	0.115	0.124
pci_bridge32_fast	0.512	0.168	0.226	0.168	0.225
des_perf_slow	0.674	0.884	0.697	0.674	0.807
des_perf_fast	2.390	3.520	2.320	2.320	1.970
vga_lcd_slow	0.415	0.378	0.391	0.378	0.359
vga_lcd_fast	0.758	0.580	0.773	0.580	0.534
b19_slow	0.627	0.614	0.736	0.614	0.607
b19_fast	2.710	X	4.490	1.040	0.973
leon3mp_slow	1.420	1.790	2.960	1.420	1.363
leon3mp_fast	X	X	4.940	2.020	1.757
netcard_slow	1.770	1.970	1.940	1.770	1.763
netcard_fast	2.010	2.300	2.970	2.010	1.909
Média de *leakage*[b]	1.140	1.109	1.644	0.969	0.924

As Figuras 12 e 13 mostram a evolução do processo de otimização do circuito leon3mp_fast com a técnica proposta em dois cenários distintos, respectivamente: com e sem o escalonamento do fator de importância da potência α (Algoritmo 6, linha 7). Em ambas figuras, TNS (*Total Negative Slack*) corresponde ao somatório dos *slacks* negativos nas saídas primárias do circuito e serve como medida objetiva das violações de *timing*. Comparando-se as Figuras 12 e 13 é possível verificar o impacto do escalonamento de α para o circuito em questão (leon3mp_fast), cuja otimização resultou na violação de *timing* para as duas melhores equipes da Competição do ISPD 2012. Por um lado, quando α é escalonado, a importância da potência é rapidamente reduzida nas iterações iniciais, permitindo assim que as violações de *timing* sejam resolvidas. Nas demais iterações, as violações de *timing* estão sob controle e a importância da potência é aumentada lentamente para diminuir a potência que foi "trocada" por atraso para resolver as violações de *timing* nas iterações iniciais.

Por outro lado, quando α não é escalonado, alguns caminhos do circuito continuam a violar as restrições de *timing*, apesar do aumento de seus LMs (λ), como pode ser observado na Figura 13, entre as iterações 13 e 21. Uma vez que o passo de atualização dos LMs (Equação 5.2) é reduzido a cada iteração, é difícil remover as violações de *timing* nas iterações finais. Ade-

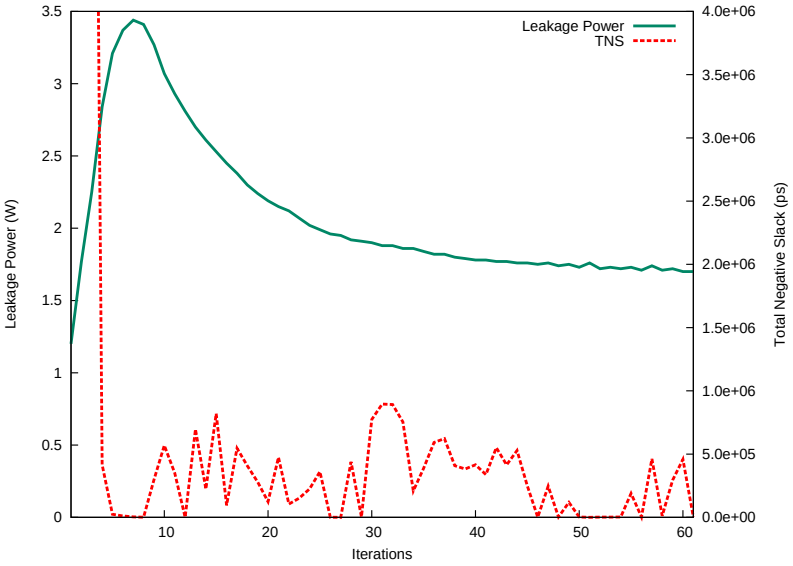


Figura 12: Evolução da otimização do circuito leon3mp_fast (649K portas) utilizando o escalonamento do fator de importância da potência α.

mais, mesmo que os LMs (λ) das portas com pequenas violações de *timing* sejam aumentados, o alto valor de α não permite que tais portas sejam trocadas por outras mais rápidas. A análise realizada neste parágrafo e no anterior mostra que o escalonamento de α é essencial para que a técnica proposta consiga eliminar as violações de *timing*. Semelhante tendência pode ser esperada para outras técnicas de *sizing* discreto baseadas em Relaxação Lagrangiana (e.g., Tennakoon e Sechen (2008) e Rahman, Tennakoon e Sechen (2011)).

Outra característica importante da técnica proposta é a incorporação das restrições de máxima capacitância na função objetivo. A Figura 14 apresenta a soma de violações de capacitância e *slew*[4] no circuito leon3mp_fast em três cenários diferentes: 1) as restrições de máxima capacitância não são incorporadas na função objetivo e portanto, não são consideradas (Algoritmo 4, linhas 7 e 10). 2) as restrições de máxima capacitância são incorporadas na função objetivo, mas a heurística complementar para eliminação das violações resultantes não é aplicada (i.e., o Algoritmo 5 não é invocado). 3) as restrições de máxima capacitância são incorporadas na função objetivo e as violações são eliminadas pela heurística complementar, o que

[4]Embora as grandezas envolvidas sejam distintas, o uso de um único valor é pragmático porque permite uma comparação direta.

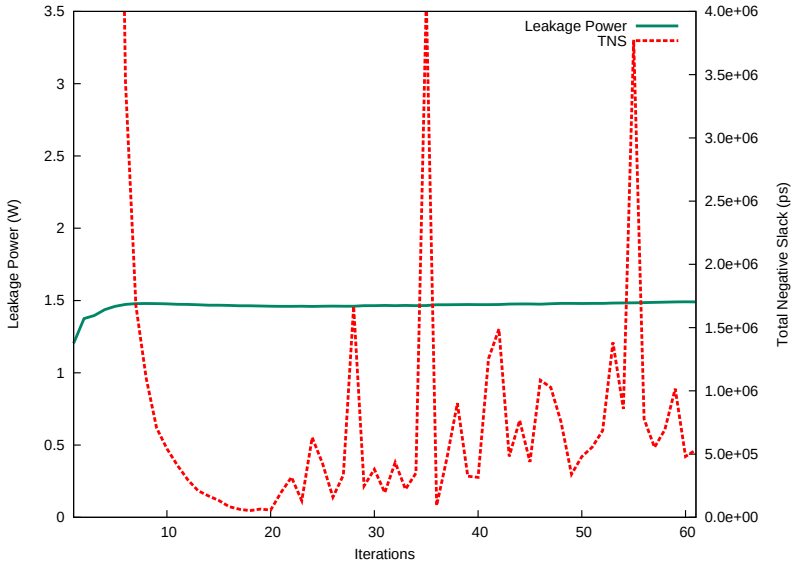


Figura 13: Evolução da otimização do circuito leon3mp_fast (649K portas) sem utilizar o escalonamento do fator de importância da potência α.

corresponde à heurística proposta. O comportamento exibido mostra que tanto a incorporação das restrições de máxima capacitância quanto o uso da heurística complementar são essenciais para a convergência do processo. Além disso, foi constatada, através de experimentos, a importância de se respeitar as restrições de máxima capacitância e máximo *slew* impostas pela biblioteca *standard cell* utilizada, pois assim evita-se que algumas portas do circuito fiquem com sobrecarga de capacitância de saída, o que poderia aumentar significativamente o atraso de alguns caminhos do circuito, dificultando a convergência da técnica de otimização.

O pequeno tempo de execução apresentado pela técnica proposta permite-lhe otimizar circuitos com até 959K portas em apenas 51 minutos[5]. A Tabela 5 apresenta a comparação no quesito tempo de execução da técnica proposta com as três melhores equipes da Competição do ISPD 2012. A coluna "Menor" apresenta o tempo de execução referente à equipe que obteve o menor valor de *leakage* para um dado circuito. Comparando-se o tempo de execução com as três melhores equipes, a técnica proposta é, em média, 38, 31 e 39 vezes mais rápida, respectivamente. A complexidade de pior caso de uma iteração da heurística gulosa topológica proposta (Algoritmo 4) é linear

[5]Note que o tempo de execução pode variar com a máquina utilizada.

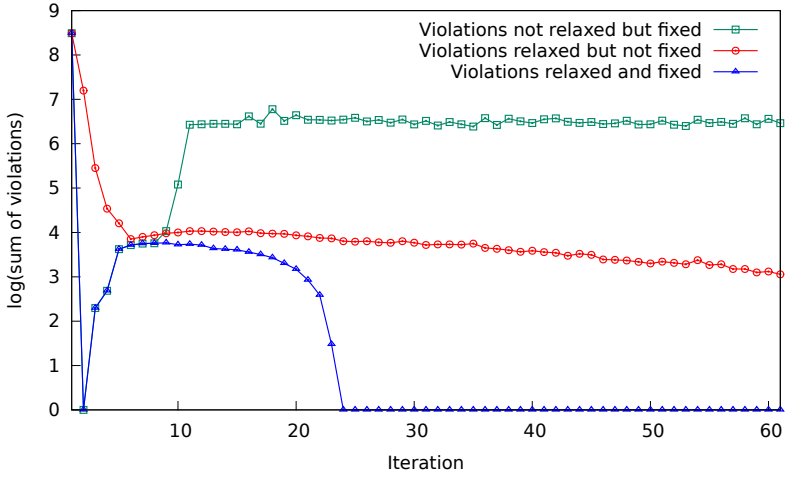


Figura 14: Efeito do controle das violações de máxima capacitância e máximo *slew* na otimização do circuito leon3mp_fast (649K portas).

em relação ao número de portas do circuito ($O(n)$). Por outro lado, a taxa de crescimento do tempo de execução da técnica de *sizing* discreto completa (Algoritmo 6) pode variar com o número de iterações utilizadas. A Figura 15 apresenta o comportamento do tempo de execução da técnica proposta considerando um número fixo de 60 iterações para cada um dos circuitos da Competição do ISPD 2012. A regressão linear dos dados obtidos revela uma taxa de crescimento do tempo de execução da ordem $O(n^{1.0254})$, a qual se aproxima bastante da complexidade linear teórica da heuristica proposta, permitindo vislumbrar a otimização de circuitos com até 10M de portas em menos de 10 horas.

5.4 CONCLUSÕES

Baseado na observação de que a formulação do problema de *sizing* usando relaxação Lagrangeana elimina as restrições de tempo de chegada do problema mediante a aplicação das condições de otimalidade KKT, este capítulo propôs uma heurística gulosa topológica para resolver o Subproblema Lagrangeano Relaxado. A técnica proposta se baseia em informações locais para guiar as decisões do algoritmo, valendo-se do fato de que a cri-

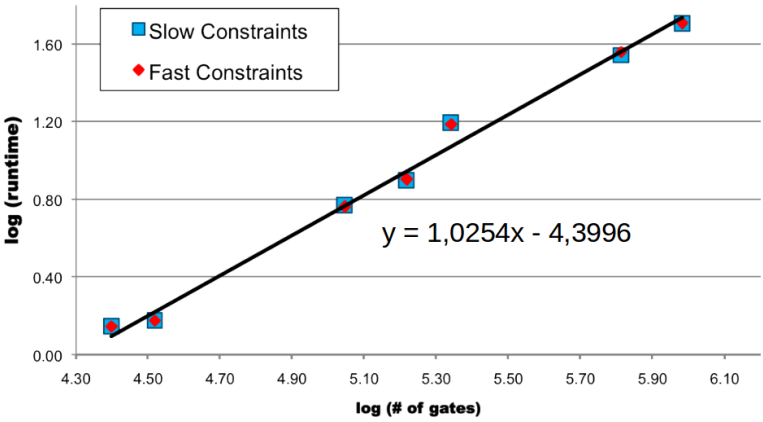


Figura 15: Comportamento do tempo de execução da técnica proposta, calculado empiricamente (considerando 60 iterações).

ticalidade dos caminhos do circuito já está embutida nos Multiplicadores de Lagrange. Esta abordagem é oposta às abordagens de programação dinâmica, cujos tempos de execução podem ser proibitivos em circuitos muito grandes. Observou-se, por meio de experimentos, que o controle das violações de máxima capacitância além de ajudar a diminuir as violações de *slew*, também contribui para reduzir as violações de *timing* do circuito. Foi mostrado também que o escalonamento do fator de importância da potência durante o processo de otimização é essencial para a eliminação das violações de *timing* e, portanto, para a convergência da técnica. Quando comparada com técnicas estado da arte, a técnica proposta obteve reduções médias de *leakage* de 18.9%, 16.7% e 43.8%, com tempos de execução 38, 31 e 39 vezes menores, respectivamente. Os resultados de tempos de execução mostram que a técnica proposta é capaz de otimizar circuitos com centenas de milhares de portas em apenas uma hora.

Tabela 5: Comparação do tempo de execução com as três melhores equipes da Competição de *sizing* do ISPD 2012 (resultados das equipes disponibilizados em domínio público por ISPD (2012)). "X" corresponde aos resultados com violações. Menor[a] corresponde ao menor valor de *leakage* obtido em cada circuito considerando-se todos os competidores. [b] Média calculada ignorando-se resultados com violações.

	Tempo de execução [horas]				
Circuitos	**NTUgs**	**UFRGS-Brazil**	**PowerValve**	**Menor[a]**	**Técnica Proposta**
DMA_slow	2.16	0.33	1.2	1.2	0.02
DMA_fast	1.72	0.54	1.52	1.52	0.02
pci_bridge32_slow	2.26	0.27	0.35	0.27	0.03
pci_bridge32_fast	1.79	0.35	0.91	0.35	0.03
des_perf_slow	7.00	7.25	722	7.00	0.1
des_perf_fast	7.00	7.53	7.22	7.22	0.1
vga_lcd_slow	9.00	3.7	8.12	3.7	0.13
vga_lcd_fast	9.00	5.36	8.12	5.36	0.13
b19_slow	11.00	8.29	9.93	8.29	0.26
b19_fast	11.00	X	9.93	0.17	0.26
leon3mp_slow	20.00	22.54	20.76	20.00	0.58
leon3mp_fast	X	X	20.76	1.30	0.60
netcard_slow	29.00	18.89	28.89	29.00	0.85
netcard_fast	29.00	31.36	28.89	29.00	0.85
Média do tempo de execução[b]	10.764	8.868	11.027	8.170	0.283

6 TÉCNICA HÍBRIDA PARA O PROBLEMA DE *SIZING* DISCRETO

Este Capítulo apresenta uma técnica híbrida para resolver o problema de *sizing* discreto, a qual é composta por 3 etapas. A primeira etapa corresponde à aplicação da técnica de *sizing* discreto baseada em Relaxação Lagrangeana apresentada no Capítulo 5, porém, assumindo um valor de atraso crítico-alvo ligeiramente maior (entre 4% e 5%) do que o atraso crítico-alvo original. A segunda etapa consiste na aplicação de uma heurística de recuperação de atraso, a qual reduz o atraso crítico do circuito até que este satisfaça a restrição de atraso crítico-alvo original. Por fim, na terceira etapa é utilizada uma heurística de recuperação de potência de *leakage*, a qual busca tirar proveito dos *slacks* remanescentes no circuito. Assim sendo, este capítulo traz três contribuições científicas ao estado da arte, as quais serão relatadas em artigo científico a ser submetido em periódico internacional. Estas contribuições são:

1. Uma heurística de recuperação de atraso, usada na segunda etapa da técnica híbrida apresentada deste capítulo, a qual tem como objetivo fazer com que a restrição de atraso crítico-alvo seja satisfeita. Para tanto, esta heurística é capaz de resolver as pequenas violações de atraso resultantes da primeira etapa do método, compensando assim as subotimatidades do método LR. A grande eficiência desta heurística advém do fato de que o atraso crítico de um circuito é determinado por algumas portas, as quais fazem parte dos caminhos críticos. Desta forma, são poucas as portas que precisam ser trocadas, o que resulta em pequeno tempo de execução e baixo impacto na potência de *leakage*;

2. Uma heurística rápida de recuperação de potência, usada na terceira etapa da técnica híbrida objeto deste capítulo. Esta heurística seleciona portas que apresentam *slack* (de saída) positivo, e tenta trocar a opção de implementação destas portas por opções de menor *leakage*, sem afetar o atraso crítico do circuito. Tal procedimento é rápido e bastante eficaz;

3. Uma análise compreensível dos resultados das três etapas da técnica híbrida proposta, baseada em gráficos de atraso versus potência de *leakage*, com o atraso crítico-alvo variando em um intervalo que inclui ambas opções *slow* e *fast* definidas na infraestrutura da Competição do ISPD 2012. Tal análise permitiu observar que os efeitos são previsíveis e não particulares a um dado ponto do espaço de soluções,

assegurando sua generalidade em termos de aplicabilidade em diferentes atrasos críticos-alvo. Foi baseado nesta análise que se chegou aos elementos-chave da técnica híbrida proposta neste capítulo, quais sejam, uma etapa de otimização global baseada em Relaxação Lagrangeana, uma segunda etapa de redução de atraso do circuito e uma terceira de redução de potência. Quando comparada com a técnica estado da arte[1] usando a infrastrutura da Competição do ISPD 2012, a técnica híbrida proposta obteve reduções médias de *leakage* de 8.15% com tempos de execução 11 vezes menores.

6.1 AVALIANDO O ESPAÇO DE OTIMIZAÇÃO DA TÉCNICA BASEADA EM RELAXAÇÃO LAGRANGEANA

O problema de *sizing* discreto para minimizar *leakage* pode ser formulado eficientemente usando Relaxação Lagrangeana (LR), conforme visto no Capítulo 4. A Função Lagrangeana com as restrições de máximo slew e máxima capacitância relaxadas após a aplicação das condições KKT está reproduzida novamente como Equação 6.1.

$$
\begin{aligned}
L_{\lambda,\gamma,\beta}(\vec{w},\vec{u}) \quad : \quad & \sum_{v_i \in X} \alpha p_i \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^f_{j \to i} d^f_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \Big(\sum_{j \in input(v_i)} \lambda^r_{j \to i} d^r_{j \to i} \Big) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^f_i (slew^f_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \gamma^r_i (slew^r_i - max\_slew) \\
& + \sum_{v_i \in (X \cup PI)} \beta_i (cap_i - max\_cap_i)
\end{aligned} \tag{6.1}
$$

Com o objetivo de avaliar a capacidade de se utilizar uma técnica baseada em LR com diferentes valores de atraso crítico-alvo e, ao mesmo tempo, verificar sua capacidade de explorar o compromisso entre atraso e potência de *leakage* para diversos circuitos, executou-se a técnica de *sizing* proposta no

[1] No momento da realização dos experimentos, a técnica de Hu et al. (2012) apresentava os melhores resultados de redução de potência de *leakage*.

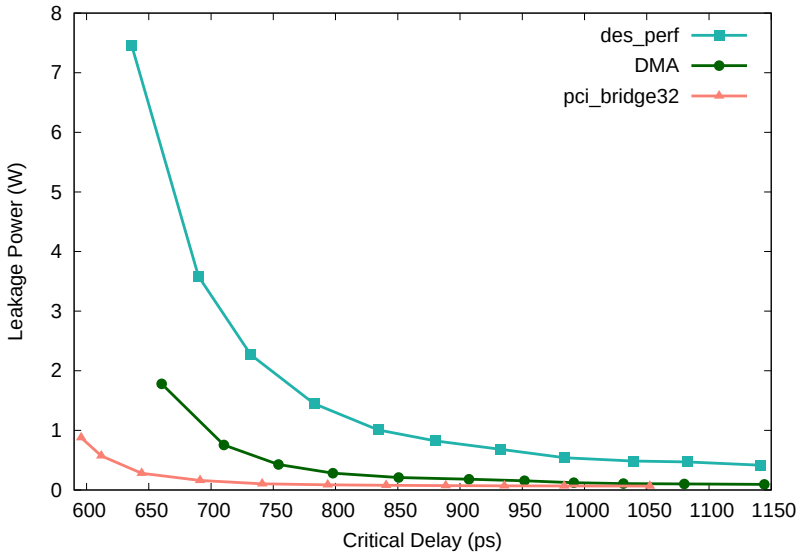


Figura 16: Explorando o espaço de otimização da heurística gulosa baseada em LR em três circuitos da Competição do ISPD 2012.

Capítulo 5 com o atraso crítico-alvo variando em um intervalo significativo. A Figura 16 apresenta as curvas obtidas para três circuitos pertencentes ao conjunto de *benchmarks* da Competição do ISPD 2012. Observe que, apesar das curvas apresentadas na Figura 16 não conterem todos os pontos possíveis de atraso-potência, elas apresentam uma parte significativa deste espaço (incluindo as restrições *slow* e *fast* definidas na Competição do ISPD 2012) e desta forma, caracterizam os resultados do método em particular para estes circuitos. É possivel observar que um método baseado em LR troca suavemente atraso por *leakage* à medida que o valor do atraso crítico-alvo é aumentado. Ou seja, nas regiões mais à direita das curvas as reduções de *leakage* são pequenas pois já estão bem próximas do mínimo valor de *leakage* para cada circuito. Já nas regiões mais à esquerda, uma pequena redução de atraso resulta em um grande aumento de *leakage*.

Duas perguntas sucedem as caracterizações do espaço de otimização dos três circuitos apresentadas na Figura 16. São elas:

- É possível obter valores de *leakage* abaixo das curvas plotadas?
- Caso a resposta à pergunta anterior seja afirmativa, como seria possível atingir-se tais pontos?

As Seções 6.2 e 6.3 apresentam análises que ajudam a responder tais questões, conduzindo à concepção da técnica híbrida citada no início deste capítulo.

6.2 HEURÍSTICAS RÁPIDAS PARA RECUPERAÇÃO DE ATRASO E POTÊNCIA

Ao longo da otimização, uma formulação relaxada (como é o caso da formulação baseada em LR do Capítulo 4) precisa "apertar" progressivamente as restrições por meio dos Multiplicadores de Lagrange, os quais são atualizados pelo Problema Lagrangeano Dual. Este processo é reconhecidamente difícil de ajustar (HUANG; HU; SHI, 2011). Caso seja dada muita liberdade para reduzir potência, as chances de ocorrer violações de *timing* são maiores. Por outro lado, caso seja dada maior ênfase para satisfazer os atrasos, a potência resultante será, inevitavelmente, maior. Esta seção apresenta o algoritmo de recuperação de atraso e o algoritmo de recuperação de potência, mencionados na introdução deste capítulo.

6.2.1 Heurística para Recuperação de Atraso

Para compensar os compromissos da otimização global, propõe-se inicialmente uma heurística de recuperação de atraso capaz de resolver pequenas violações de atraso, preservando a redução de potência de *leakage* previamente obtida.

Algoritmo 9: DELAY_RECOVERY

```
   Input  : G(V,E), L, A_o
   Output: G(V,E) Optimized for timing
 1 while (iterations < 1% number of gates) &&
 2 (circuit has timing violations) do
 3     v_best ← COMPUTE_SENSITIVITY();
 4     if v_best ≠ 0 then
 5         swap v_best to its next lower V_t;
 6         INCREMENTAL_TIMING_ANALYSIS(G, L, A_o, v_best);
 7     else
 8         return 0 ; // no swappable gates ∈ critical
                      path
 9     end
10 end
```

Algoritmo 10: COMPUTE_SENSITIVITY

```
   Input  : G(V,E), L, A_o
   Output: v_best
 1 sensitivity ← ∞;
 2 for ∀v_i ∈ critical path do
 3     if v_i is not lowest V_t then
 4         swap v_i to its next lower V_t;
 5         INCREMENTAL_TIMING_ANALYSIS(G, L, A_o, v_i);
 6         if !(slew or capacitance violations) then
 7             if −ΔTNS / Δleakage > sensitivity then
 8                 sensitivity ← −ΔTNS / Δleakage;
 9                 v_best ← v_i;
10             end
11         end
12         restore v_i to its previous V_t;
13     end
14 end
15 if sensitivity = ∞ then
16     return 0 ;
17 else
18     return gate v_best with highest sensitivity;
19 end
```

Os Algoritmos 9 e 10 descrevem a heurística de recuperação de atraso proposta, a qual baseia-se em uma função de sensibilidade similar a de Hu et al. (2012). Assim como na técnica TILOS (FISHBURN; DUNLOP, 1985), somente as portas pertencentes ao caminho crítico do circuito são visitadas, o que contribui para um pequeno tempo de execução. Quando uma porta do caminho crítico atual é visitada, ela é trocada pela opção de tensão de *threshold* (V_t) imediatamente inferior e sua sensibilidade é calculada, conforme detalhado no Algoritmo 10. A sensibilidade é calculada como a redução da quantidade total de *slack* negativo nas saídas primárias do circuito, dividida pelo incremento de potência de *leakage*, ambos resultantes da mudança de V_t. As informações de *timing* do circuito são atualizadas por um passo de Análise de *Timing* Incremental (ITA), no qual somente os tempos de chegada pertencentes ao cone lógico da porta trocada e os *slacks* das saídas primárias são calculados, ao passo que os tempos requeridos não são atualizados, o que implica em uma redução significativa do tempo de execução. A partir dos resultados experimentais foi possível concluir que limitar o total de trocas a

1% do número de portas do circuito consiste em uma boa estratégia, pois limita o aumento de *leakage* no circuito, além de manter o tempo de execução pequeno. Mas obviamente, o algoritmo pode parar antes, tão cedo o atraso crítico-alvo seja atingido. Por outro lado, se o número de trocas exceder o limite permitido ou se todas as portas do caminho crítico atual já corresponderem às respectivas opções de menor V_t, o algoritmo fracassa em recuperar o atraso crítico-alvo.

O gráfico da Figura 17 mostra a curva de atraso versus potência de *leakage* resultante da otimização do circuito DMA somente com a técnica baseada em LR do Capítulo 5 (curva mais acima). Ele também mostra os resultados de se aplicar a heurística de recuperação de atraso (Algoritmo 9) para diversos valores de atraso crítico-alvo, o que resulta na curva mais abaixo na Figura 17. Para a interpretação destes resultados, considere que o atraso crítico-alvo escolhido para o DMA seja aproximadamente 905 ps. Há duas maneiras para atingir-se tal atraso crítico. A primeira é executando somente a técnica baseada em formulação LR apresentada no Capítulo 5 para um atraso crítico-alvo de 905 ps, o que resulta em um ponto sobre a curva mais acima cuja abscissa é 905 ps. A segunda opção consiste em executar a técnica do Capítulo 5 para um atraso crítico-alvo um pouco maior (digamos, aproximadamente 951 ps), e depois recuperar o atraso crítico-alvo original executanto o Algoritmo 9 para 905 ps, resultando em um ponto na curva inferior cuja abscissa é 905 ps. Observe que a segunda opção é mais vantajosa, uma vez que a potência de *leakage* resultante é mais baixa do que na primeira opção. Apesar desta redução parecer pequena na Figura 17, a potência de *leakage* obtida pela segunda maneira é aproximadamente 24% menor quando comparada à minimização resultante da primeira (usando somente LR).

6.2.2 Heurística para Recuperação de Potência

Conforme já mencionado no início deste capítulo, após a execução da técnica baseada em formulação LR existem *slacks* remanescentes nos caminhos não-críticos do circuito. Tais *slacks* podem ser explorados para obter-se reduções adicionais de potência de *leakage*, sem comprometer o atraso crítico-alvo. É difícil eliminar tais *slacks* em um único passo de otimização (somente LR, por exemplo), uma vez que isso aumentaria a probabilidade de ocorrer violações de *timing*. Porém, existe aí uma oportunidade para se realizar um passo de otimização local, tal como ocorre em diversas etapas do fluxo de EDA. Para tal otimização local, desenvolveu-se uma heurística específica, a qual é expicada a seguir.

O Algoritmo 11 detalha a heurística de recuperação de potência ex-

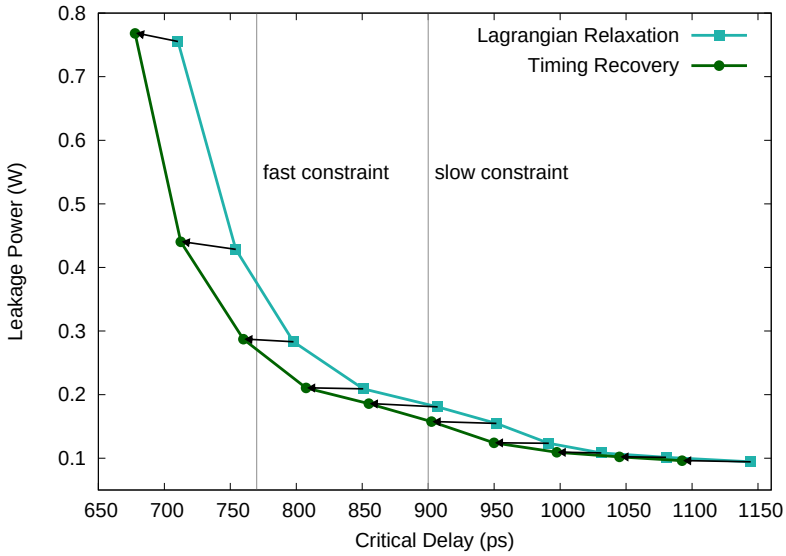


Figura 17: Avaliando o impacto da heurística de recuperação de atraso após a otimização baseada em LR para o circuito DMA da Competição do ISPD 2012.

plorando os *slacks* deixados pelas duas etapas anteriores. Reduzir potência usando-se listas de sensibilidade resultaria em tempos de execução proibitivamente longos para circutos com muitas portas, visto que para redução de potência normalmente se calcula a sensibilidade de todas as portas do circuito (HU et al., 2012). Isto ocorre porque cada cálculo de sensibilidade requer atualização de informações de *timing*, conforme mencionado por Hu et al. (2012). Na abordagem proposta, as portas são visitadas em ordem topológica, e cada porta é trocada pela sua opção com valor de V_t imediatamente superior, caso tal troca não resulte em violações. Conforme observado por Rahman, Tennakoon e Sechen (2012), trocar para a opção com valor de V_t imediatamente superior, ao invés de trocar para a opção de maior V_t (no caso de haver mais de duas opções de V_t), ajuda a distribuir os *slacks* de maneira mais homogênea ao longo do circuito, possivelmente resultando em maior redução de potência de *leakage* no final do processo de otimização. Caso uma porta já esteja em seu maior V_t, a heurística troca para uma opção com largura imediatamente menor se esta troca não resultar em violações. Foi observado nos experimentos que um número pequeno de iterações (3, por exemplo) é suficiente para obter-se uma redução significativa de potência de

Algoritmo 11: POWER_RECOVERY

```
   Input  : G(V,E), L, A_o
   Output: G(V,E) optimized for leakage power
 1 while iterations < it_max do
 2     for ∀v_i ∈ V in topological order do
 3         if v_i is not lowest leakage option then
 4             if v_i is not highest V_t then
 5                 swap v_i to its next higher V_t;
 6             else
 7                 swap v_i to its next lower size w;
 8             end
 9             INCREMENTAL_TIMING_ANALYSIS(G, L, A_o, v_i);
10             if has violations then
11                 restore v_i to its previous option;
12             end
13         end
14     end
15 end
```

leakage.

6.3 A TÉCNICA HÍBRIDA PROPOSTA

A fim de entender como a técnica baseada em LR do Capítulo 5 e as heurísticas apresentadas na Seção 6.2 podem ser usadas conjuntamente para criar-se uma técnica híbrida para *sizing* discreto, traçou-se um novo gráfico com três curvas de atraso versus potência de *leakage*: as duas curvas do gráfico da Figura 17 mais a curva resultante da otimização do circuito DMA com a heurística de recuperação de potência. Para gerar esta última curva, a heurística de recuperação de potência recebeu como entrada descrições do circuito DMA já otimizado pelas duas etapas anteriores (heurística LR e a heurística de recuperação de atraso), cada descrição assumindo um valor distinto para atraso crítico-alvo. A Figura 18 mostra a região central das três curvas mencionadas. Comportamentos similares ao visto nestas curvas também foram observados para outros circuitos da Competição do ISPD 2012. Nesta figura, cada par de setas (uma horizontal e uma vertical) mostra como a aplicação da heurística de recuperação de atraso seguida pela aplicação da heurística de recuperação de potência permite atingir-se um ponto de atraso-

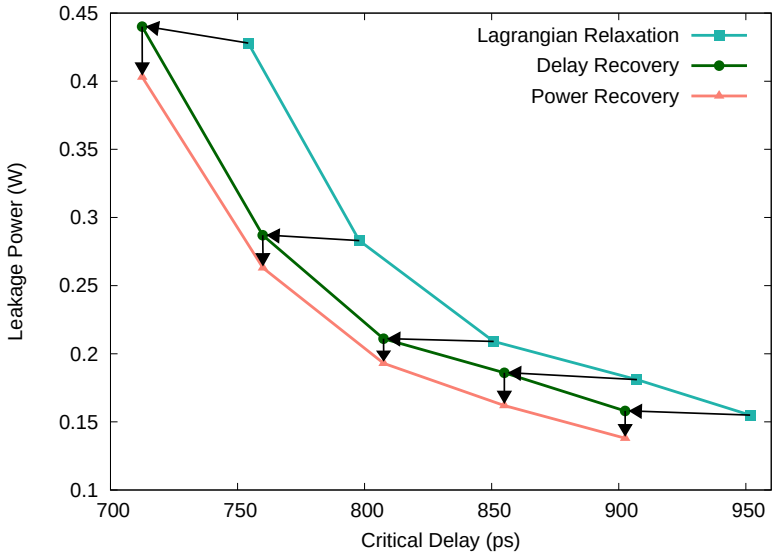


Figura 18: Avaliando o impacto das heurísticas de recuperação de atraso e potência após a otimização baseada em LR para o circuito DMA da Competição do ISPD 2012.

potência que é inferior àquele que pode ser atingido aplicando-se somente uma técnica baseada LR. Desta forma, caso se deseje atingir um determinado atraso crítico-alvo (digamos, 900 ps para o circuito em questão) com valor de potência de *leakage* menor do que aquele proporcionado pela formulação LR sozinha, pode-se executar um passo inicial com a formulação LR assumindo um atraso crítico-alvo ”afrouxado”(por exemplo, 951 ps, conforme visto na Figura 18). Em seguida, aplica-se a heurítica de recuperação de atraso para atingir o atraso crítico-alvo original (905 ps) e finalmente executa-se a heurítisca de recuperação de potência. Após realizar um número significativo de experimentos, concluiu-se que a adição de aproximadamente 5% ao atraso crítico-alvo original cria um maior espaço de otimização para a heurística gulosa baseada em LR, permitindo assim obter-se um valor de *leakage* menor. Esse pequeno incremento do atraso crítico-alvo permite que a heurística de recuperação de atraso reestabeleça o atraso crítico-alvo original mediante a troca de uma pequena fração das portas do circuito, com um aumento de *leakage* muito pequeno. Considerando os experimentos e a análise mencionados nesta seção e na anterior, projetou-se uma nova técnica híbrida para resolver o problema de *Sizing* discreto baseada na seguinte sequência de

etapas:

1. Otimização com a heurística topológica gulosa baseada em LR (apresentada no Capítulo 5), assumindo atraso crítico-alvo aumentado em 5% em relação ao original;

2. Recuperação de atraso;

3. Recuperação de potência de *leakage*.

6.4 RESULTADOS EXPERIMENTAIS

Os Algoritmos 9, 10 e 11 foram implementados na linguagem C++ e compilados usando a versão 4.4.5 do gcc, a fim de compor um protótipo para executar a técnica híbrida proposta neste capítulo. Para computar as informações de *timing* do circuito necessárias (linha 6 do Algoritmo 9, linha 5 do Algoritmo 10 e linha 9 do Algoritmo 11), foi utilizada a ferramenta de STA dedicada mencionada na Seção 1.6 (Infraestrutura Experimental), devidamente configurada para realizar análise de *timing* incremental (ITA). Para a geração dos resultados foi utilizada a infraestrutura experimental, também descrita na seção 1.6.

Os resultados gerados para a técnica híbrida proposta foram comparados com aqueles reportados pelos trabalhos de Hu et al. (2012) e de Li et al. (2012), os quais também utilizaram a infraestrutura da Competição do ISPD 2012[2]. Os resultados de potência de *leakage* obtidos pelo primeiro destes dois trabalhos o credenciavam como estado da arte naquele momento (em termos de redução de *leakage*), ao passo que o trunfo do segundo trabalho residia nos excelentes tempos de execução alcançados.

Os gráficos das Figuras 19 e 20 mostram as reduções de potência de *leakage*, normalizadas com relação aos melhores resultados da Competição do ISPD 2012, obtidas pela técnica híbrida proposta e pelos dois trabalhos correlatos (Hu et al. (2012) e Li et al. (2012)) para os circuitos *fast* e *slow*, respectivamente. Não obstante as técnicas usadas nos dois trabalhos correlatos tenham proporcionado boas reduções de potência de *leakage*, nenhuma delas exibe superioridade absoluta. Por outro lado, a técnica híbrida proposta no presente capítulo proporcionou redução de potência de *leakage* para todos os circuitos (e ambas restrições de atraso crítico-alvo, *fast* e *slow*), sendo tal redução maior do que aquelas alcançadas por ambos trabalhos correlatos.

[2]É importante observar que a configuração da máquina utilizada nos experimentos relatados neste capítulo é ligeiramente inferior às configurações das máquinas usadas pelos trabalhos de Hu et al. (2012) e de Li et al. (2012).

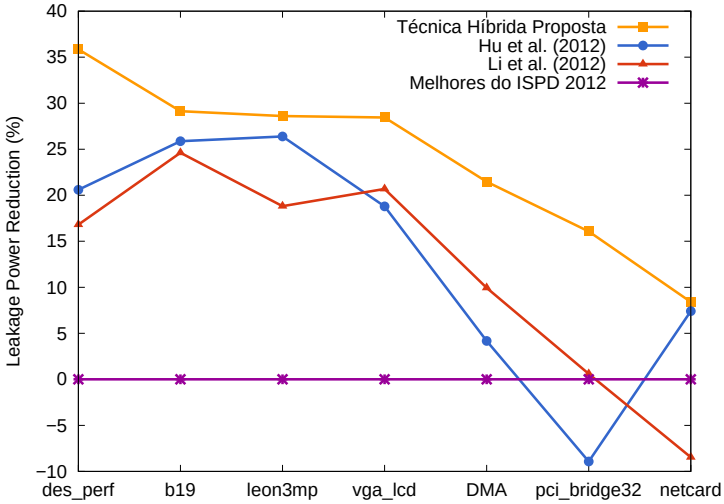


Figura 19: Comparações das reduções de *leakage* normalizadas com relação aos melhores resultados obtidos na Competição do ISPD 2012 para cada um dos circuitos *fast*.

Quando comparadas com os melhores resultados da Competição do ISPD 2012, as reduções médias de potência de *leakage* obtidas pela técnica híbrida proposta, pela técnica de Hu et al. (2012) e pela técnica de Li et al. (2012) foram 16,53%, 8,99% e 6,15%, respectivamente.

Hu et al. (2012) apresentaram uma análise sobre o espaço de otimização disponível nos circuitos da Competição do ISPD 2012 após a aplicação da técnica de *sizing* discreto por eles desenvolvida. Tal análise concluiu que os resultados de potência de *leakage* obtidos por sua técnica são entre 1,002 vezes e 2,29 vezes maiores do que os valores mínimos estimados para os circuitos *slow*. Para os circuitos *fast*, os resultados são entre 1,06 vezes e 6,88 vezes maiores do que os valores mínimos estimados. Isto justifica o fato das reduções de potência de *leakage* obtidas pela técnica híbrida apresentada no presente trabalho serem maiores para os circuitos *fast* do que para os circuitos *slow*. Assim, as reduções médias de potência de *leakage* obtidas pela técnica híbrida, por Hu et al. (2012) e por Li et al. (2012) para os circuitos *fast* são de 24,0%, 13,47% e 11,86%, respectivamente, quando comparados com os melhores resultados da Competição do ISPD 2012. As reduções obtidas pela técncia híbrida proposta variam entre 35,86% e 8,41%, sendo maiores do que

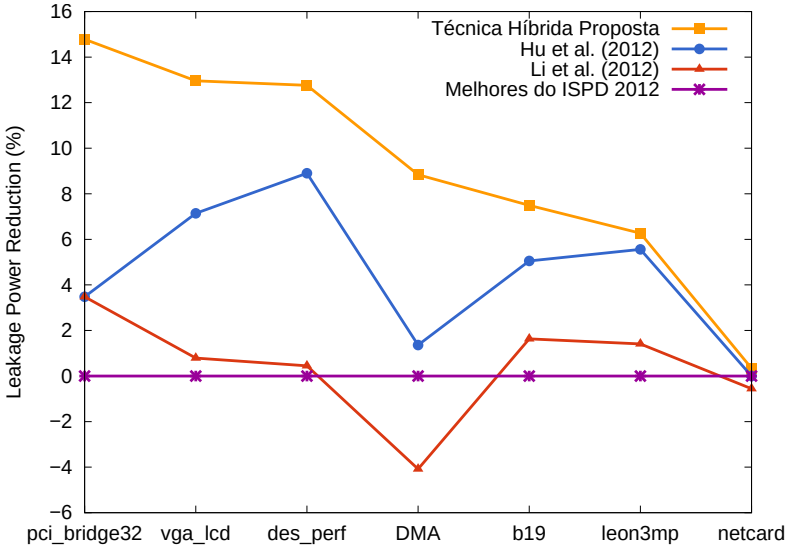


Figura 20: Comparações das reduções de *leakage* normalizadas com relação aos melhores resultados obtidos na Competição do ISPD 2012 para cada um dos circuitos *slow*.

aquelas obtidas pelos dois trabalhos correlatos para os circuitos *fast*. É importante observar que o resultado de *leakage* obtido pela técnica de Hu et al. (2012) para o circuito pci_bridge32_fast é 8,93% maior do que o melhor resultado da Competição do ISPD 2012 para este circuito. Também vale observar que o resultado de *leakage* de Li et al. (2012) para o circuito netcard_fast é 8,46% maior do que o melhor resultado da Competição do ISPD 2012 para este circuito. Considerando os circuitos *slow*, as reduções médias de *leakage* obtidas pela técnica híbrida, por Hu et al. (2012) e por Li et al. (2012) são 9.6%, 4.5% and 0.44%, respectivamente, com relação aos melhores resultados da Competição do ISPD 2012. Novamente, a técnica híbrida proposta possibilitou reduções de *leakage* para todos os circuitos *slow*, ao passo que a técnica de Li et al. (2012) resultou em aumentos de *leakage* de 4,08% e 0,56% para os circuitos DMA_slow e netcard_slow, respectivamente.

A Tabela 6 apresenta os resultados de potência de *leakage* obtidos pela técnica híbrida proposta, bem como uma comparação, circuito a circuito, entre esta e as técnicas dos dois trabalhos correlatos. A partir desta tabela observa-se que a técnica proposta obteve reduções médias de *leakage* de 8,15% e 11.09%, quando comparada com Hu et al. (2012) e com Li et

Tabela 6: Comparação de potência de *Leakage* da técnica híbrida com as técnicas estado da arte no momento que os experimentos foram realizados. (*) Resultados do circuito des_perf_fast foram obtidos por meio do "afrouxamento" de 4% do atraso crítico-alvo.

	Potência de *Leakage* (*W*)			**Redução (%)**	
Benchmarks	**Hu (HU et al., 2012)**	**Li (LI et al., 2012)**	**Técnica Proposta**	**Técnica Proposta vs. (HU et al., 2012)**	**Técnica Proposta vs. (LI et al., 2012)**
DMA_fast	0.299	0.281	0.245	18.06	12.81
pci_bridge32_fast	0.183	0.167	0.140	22.95	15.57
des_perf_fast*	1.842	1.930	1.487	19.22	22.90
vga_lcd_fast	0.471	0.460	0.415	11.89	9.78
b19_fast	0.771	0.784	0.737	4.41	5.99
leon3mp_fast	1.487	1.640	1.441	3.03	12.07
netcard_fast	1.861	2.180	1.841	1.07	15.55
DMA_slow	0.145	0.153	0.134	7.59	12.42
pci_bridge32_slow	0.111	0.111	0.097	11.71	11.71
des_perf_slow	0.614	0.671	0.588	4.23	12.37
vga_lcd_slow	0.351	0.375	0.329	6.27	12.27
b19_slow	0.583	0.604	0.567	2.57	5.96
leon3mp_slow	1.341	1.400	1.330	0.75	4.93
netcard_slow	1.770	1.780	1.763	0.34	0.90
average (slow)	0.70	0.73	0.69	4.78	8.65
average (fast)	0.99	1.06	0.90	11.52	13.53
average	0.84	0.90	0.79	8.15	11.09

al. (2012), respectivamente. Considerando-se os circuitos *fast*, as reduções médias foram de 11,52% e 13,53%, respectivamente, ao passo que as reduções médias para os circuitos *slow* foram de 4,78% and 8,65%, respectivamente.

A Tabela 7 mostra que além de ser mais eficiente em termos de redução de *leakage* quando comparada com Hu et al. (2012), a técnica híbrida proposta ainda apresenta tempo de execução total 11x menor. Tal desempenho pode ser explicado pelo fato de que a técnica proposta realiza múltiplas trocas de opção de implementação (de portas) por iteração (na primeira etapa, quando realiza otimização baseada em LR), enquanto a técnica de Hu et al. (2012) realiza somente uma troca por iteração. A Tabela 7 mostra também que a técnica proposta é 1,47x mais lenta do que a técnica de Li et al. (2012). Sendo esta uma diferença de tempo de execução pequena, é possível especular-se que ela poderia ser reduzida (ou mesmo eliminada) caso seja utilizada uma máquina com configuração superior para executar a técnica proposta.

6.5 CONCLUSÕES

Este capítulo apresentou uma análise detalhada a respeito da maneira como a técnica de *sizing* discreto baseada em LR explora o espaço de otimização, considerando circuitos com características distintas. Tal análise levou à conclusão que o "afrouxamento" da restrição de atraso crítico-alvo

Tabela 7: Comparação do tempo de execução da técnica híbrida com as técnicas estado da arte no momento que os experimentos foram realizados.

		Tempo de Execução (*minutos*)		
Benchmarks	**# of Gates**	**Hu (HU et al., 2012)**	**Li (LI et al., 2012)**	**Técnica Proposta**
DMA_fast	25K	13.9	0.6	2.13
pci_bridge32_fast	33K	13.0	1.2	1.87
des_perf_fast	111K	82.7	6.6	10.02
vga_lcd_fast	165K	45.6	10.2	16.4
b19_fast	219K	206.5	12.0	22.15
leon3mp_fast	649K	1274.0	43.8	56.68
netcard_fast	959K	1096.9	88.8	94.9
DMA_slow	25K	9.9	0.6	1.97
pci_bridge32_slow	33K	10.2	1.2	1.88
des_perf_slow	111K	70.1	6.0	9.55
vga_lcd_slow	165K	87.5	7.8	11.2
b19_slow	219K	213.9	10.2	20.93
leon3mp_slow	649K	1274.0	43.8	56.68
netcard_slow	959K	299.9	48.0	75.9
Total (hours)		79.1	4.86	7.14

permite um maior aproveitamento do espaço de otimização pela técnica baseada em LR. Para restabelecer o atraso crítico-alvo original, foi proposta uma heurística de recuperação de atraso a qual troca apenas algumas portas dos caminhos críticos do circuito. Os *slacks* do circuito remanescentes das duas etapas anteriores foram trocados por redução de potência de *leakage* com a técnica de recuperação de potência. Considerando apenas a redução de *leakage* (e, portanto, ignorando o tempo de execução) e considerando a infraestrutura da Competição do ISPD 2012, técnica híbrida proposta obteve superioridade absoluta sobre todos os métodos de *sizing* discreto disponíveis até então. Em relação aos circuitos que ainda apresentavam um grande espaço de otimização (circuitos *fast*), a técnica proposta possibilitou reduções de *leakage* médias de 11,52% em relação à então técnica do estado da arte (HU et al., 2012), sendo ainda uma ordem de magnitude mais rápida.

7 CONCLUSÕES E TRABALHOS FUTUROS

7.1 CONCLUSÕES

Devido à sua eficácia na otimização de diferentes objetivos em circuitos projetados com o fluxo *standard cell*, tais como atraso, potência e área, *gate sizing* é uma técnica que continua sendo intensamente investigada pela academia e pela indústria de EDA, conforme demonstram os inúmeros trabalhos publicados recentemente. A natureza discreta do problema, o grande número de componentes dos circuitos digitais contemporâneos e a complexidade do modelo de atraso usado nas bibliotecas *standard cell* tornaram o problema de *sizing* extremamente desafiador, sendo que até então não havia uma técnica com superioridade absoluta em termos de redução de *leakage*. Prova disso foi a realização das Competições de *sizing* discreto do ISPD 2012 e do ISPD 2013, organizadas por pesquisadores ligados à indústria.

Relaxação Lagrangeana (LR) tem sido adotada por diversos trabalhos de *sizing* contínuo e de *sizing* discreto devido à sua capacidade de lidar simultaneamente com objetivos conflitantes entre si, tais como atraso e potência de *leakage*, ainda apresentando boa capacidade de escalabilidade. Apesar de diversas técnicas de *sizing* baseadas em LR já terem sido propostas, poucas assumem o problema diretamente no domínio discreto e nenhuma considera em sua formulação as restrições de máxima capacitância de saída e máximo *slew* de entrada das portas conforme definido nas bibliotecas *standard cell* contemporâneas. Com isso, o presente trabalho propôs uma formulação do problema de *sizing* discreto baseado em LR, a qual relaxa tais restrições impostas pelas bibliotecas, incorporando-as na função objetivo. A formulação foi então simplificada através da aplicação das condições de otimalidade KKT propostas em Chen, Chu e Wong (1999) visando remover os tempos de chegada.

As técnicas baseadas em programação dinâmica para resolver o Subproblema Lagrangeano Relaxado tem como principal limitação o alto esforço computacional, o que pode resultar em tempos de execução proibitivos para circuitos da ordem de milhões de portas. Para contornar tal limitação, uma heurística gulosa topológica foi proposta neste trabalho. Esta heurística baseia-se na formulação LR proposta no Capítulo 4 do presente trabalho e utiliza informações locais para guiar a otimização. Isso é possível porque as informações globais de *timing* do circuito já estão representadas pelos Multiplicadores de Lagrange. Para validar a heurística proposta, foram realizados experimentos utilizando a infraestrutura da Competição de *Sizing* Discreto do ISPD 2012. Os resultados experimentais mostraram reduções médias de

leakage de 18,9%, 16,7% e 43,8% em relação às três melhores equipes da Competição, as quais representavam o estado da arte naquele momento. Além disso, a heurística proposta obteve tempos de execução 38, 31 e 39 vezes menores, respectivamente.

Conforme constatado por diversos trabalhos, a principal dificuldade da aplicação de LR no problema de *sizing* discreto para otimizar potência sob restrições de atraso reside no ajuste dos diversos parâmetros, tais como os Multiplicadores de Lagrange e o fator de importância da potência, os quais podem apresentar comportamentos distintos em função das características particulares de cada circuito. Entretanto, um ajuste fino para cada circuito seria inviável pois exigiria um alto esforço computacional. Ao invés disso, a técnica híbrida desenvolvida neste trabalho compensa tal dificuldade no ajuste dos parâmetros dando um maior espaço de otimização para a técnica LR, o que foi obtido "afrouxando-se" o atraso crítico-alvo. A aplicação da heurítica de recuperação de atraso e, em seguida, da heurística de recuperação de potência permitiu satisfazer o atraso crítico-alvo original e ainda atingir um valor de potência de *leakage* inferior àquele que seria possível obter-se somente com a aplicação da técnica LR. Considerando os dois trabalhos então estado da arte que também utilizaram a infraestrutura da Competição do ISPD 2012, Hu et al. (2012) haviam obtido os menores valores de potência de *leakage*, ao passo que Li et al. (2012) haviam atingido os menores tempo de execução. A técnica híbrida proposta, quando comparada com Hu et al. (2012), apresentou menores valores de potência de *leakage* para todos os circuitos considerados, com redução média de 8.15%. No que se refere ao tempo de execução, a técnica proposta foi uma ordem de magnitude mais rápida que a técnica de Hu et al. (2012). Comparada com o técnica de Li et al. (2012), cuja principal característica é o baixo tempo de execução, a técnica proposta foi 1.47 vezes mais lenta, o que foi compensado pela significativa redução de potência de *leakage*. As reduções de *leakage* proporcionadas pela técnica proposta estabelecem um novo estado da arte, além de aumentarem os indícios de que a combinação de diferentes técnicas é essencial para resolver o problema de *sizing* discreto de maneira eficiente, visto que uma técnica pode compensar a limitação da outra.

7.2 TRABALHOS FUTUROS

Uma das possibilidades de trabalho futuro reside na investigação detalhada dos parâmetros usados na heurística gulosa baseada em LR, tais como os Multiplicadores de Lagrange e o fator de importância de potência, a fim de entender por que a técnica baseada em LR aplicada de maneira independente

não consegue chegar à mesma solução encontrada pela técnica híbrida proposta. De fato, esse ajuste dos parâmetros não é uma tarefa trivial, conforme ressaltado em diversos trabalhos como Tennakoon e Sechen (2008), Huang, Hu e Shi (2011), Ozdal, Burns e Hu (2012) e Li et al. (2012). Outra possibilidade de investigação futura seria a resolução do Problema Lagrangeano Dual visto que o método do subgradiente apresenta diversas desvantagens quando aplicado ao *sizing* discreto. Apesar de Huang, Hu e Shi (2011) terem proposto um método alternativo para resolver o LDP aplicado ao problema de *sizing* discreto, diversos detalhes práticos foram desconsiderados, tais como múltiplos arcos de tempo por porta e diferentes transições de subida e descida.

REFERÊNCIAS BIBLIOGRÁFICAS

ABRISHAMI, H. et al. Post sign-off leakage power optimization. In: *Proceedings of Design Automation Conference*. [S.l.: s.n.], 2011.

AHUJA, R. K.; MAGNANTI, T. L.; ORLIN, J. B. *Network Flows: Theory, Algorithms, and Applications*. [S.l.]: Prentice Hall, 1993. 846 p.

ALDOUS, D.; VAZIRANI, U. Go with the winners' algorithms. In: *Proc. FOCS*. [S.l.: s.n.], 1994. p. 492–501.

ASSOC., S. I. *International Technology Roadmap for Semiconductors*. 2002. <http://www.itrs.net/Links/2002Update/2002Update.pdf>. Acessado em 10/01/2013.

AUTH, C. et al. 45nm high-k+metal gate strain-enhanced transistors. *Intel Journal of Technology*, v. 12, p. 77–85, Junho 2008.

BAZARAA, M. S.; SHERALI, H. D.; SHETTY, C. M. *Nonlinear Programming: Theory and Algorithms*. [S.l.]: Wiley-Interscience, 2006. 872 p.

BEEFTINK, F. et al. Gate-size selection for standard cell libraries. In: *Proceedings of International Conference on Computer-Aided Design*. [S.l.: s.n.], 1998. p. 545–550.

BERKELAAR, M. R.; JESS, J. A. Gate sizing in MOS digital circuits with linear programming. In: *Proceedings of Conference on European Design Automation*. [S.l.: s.n.], 1990. p. 217–221.

BHASKER, J.; CHADHA, R. *Static Timing Analysis for Nanometer Designs*. [S.l.]: Springer, 2009. 572 p.

BOYD, S.; VANDENBERGHE, L. *Convex Optimization*. [S.l.]: Cambridge University Press, 2004. 730 p.

CHANDRAKASAN, A.; SHENG, S.; BRODERSEN, R. Low-power digital cmos design. *IEEE Journal of Solid-State Circuits*, v. 27, p. 473–484, Abr 1992.

CHEN, C.-P.; CHU, C. C. N.; WONG, D. F. Fast and exact simultaneous gate and wire sizing by lagrangian relaxation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 18, p. 1014–1025, July 1999.

CHINNERY, D. G.; KEUTZER, K. Linear programming for sizing, vth and vdd assignment. In: *Proceedings of International Symposium on Low Power Electronics and Design*. [S.l.: s.n.], 2005. p. 149–154.

CHUANG, W.; SAPATNEKAR, S. S.; HAJJ, I. N. A unified algorithm for gate sizing and clock skew optimization to minimize sequential circuit area. In: *Proceedings of the International Conference on Computer-Aided Design*. [S.l.: s.n.], 1993. p. 220–223.

CHUANG, W.; SAPATNEKAR, S. S.; HAJJ, I. N. Timing and area optimization for standard-cell vlsi circuit design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 14, p. 308–320, Mar 1995.

CORMEN, T. H. et al. *Introduction to Algorithms*. [S.l.]: MIT Press, 2009. 1312 p.

COUDERT, O. Gate sizing for constrained delay/power/area optimization. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, v. 5, p. 465–472, Dec 1997.

DHARCHOUDHURY, A. et al. Transistor-level sizing and timing verification of domino circuits in the power pc microprocessor. In: *Proceedings of International Conference on Computer Design*. [S.l.: s.n.], 1997. p. 143–148.

ELMORE, W. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, v. 19, p. 55–63, 1948.

FISHBURN, J. P.; DUNLOP, A. E. Tilos: A posynomial programming approach to transistor sizing. In: *Proceedings of International Conference on Computer-Aided Design*. [S.l.: s.n.], 1985. p. 326–328.

FISHER, M. L. An applications oriented guide to lagrangian relaxation. *Interfaces*, v. 15, n. 2, p. 10–21, 1985.

GUNTZEL, J. L. A. *Functional Timing Analysis of VLSI Circuits Containing Complex Gates*. Tese (Doutorado) — Universidade Federal do Rio Grande do Sul, RS-Brazil, 2000.

GUPTA, P. et al. Eyecharts: Constructive benchmarking of gate sizing heuristics. In: *Proceedings of the Design Automation Conference*. [S.l.: s.n.], 2010. p. 597–602.

HSINWEI, C.; WANG, Y.; CHEN, C. Fast and effective gate-sizing with multiple-vt assignment using generalized lagrangian relaxation. In: *Proceedings of the Asia and South Pacific Design Automation Conference*. [S.l.: s.n.], 2005. p. 381–386.

HU, J. et al. Sensitivity-guided metaheuristics for accurate discrete gate sizing. In: *Proceedings of International Conference on Computer-Aided Design*. [S.l.: s.n.], 2012. p. 107–111.

HU, S.; KETKARY, M.; HU, J. Gate sizing for cell-library-based designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 28, p. 818–825, June 2009.

HUANG, Y.-L.; HU, J.; SHI, W. Lagrangian relaxation for gate implementation selection. In: *Proceedings of International Symposium on Physical Design*. [S.l.: s.n.], 2011. p. 167–174.

ISPD. *ISPD 2012 Discrete Gate Sizing Contest and Benchmark Suite*. 2012. <http://archive.sigda.org/ispd/contests/12/ispd2012_contest.html>. Acessado em 01/12/2012.

JEONG, K.; KAHNG, A. B.; YAO, H. Revisiting the linear programming framework for leakage power vs. performance optimization. In: *Proceedings of International Symposium on Quality of Electronic Design*. [S.l.: s.n.], 2009. p. 127–134.

JIANG, I. H.-R.; CHANG, Y.-W.; JOU, J.-Y. Crosstalk-driven interconnect optimization by simultaneous gate andwire sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 19, p. 999–1010, Sept 2000.

JOSHI, S.; BOYD, S. An efficient method for large-scale gate sizing. *IEEE Transactions on Circuits and Systems*, v. 55, p. 2760–2773, Oct 2008.

JU, Y.-C.; SALEH, R. Incremental techniques for the identification of statically sensitizable critical paths. In: *Proceedings of Design Automation Conference*. [S.l.: s.n.], 1991. p. 541–546.

KAHNG, A. B.; KANG, S. Construction of realistic gate sizing benchmarks with known optimal solutions. In: *Proceedings of International Symposium on Physical Design*. [S.l.: s.n.], 2012. p. 153–160.

KAHNG, A. B. et al. *VLSI Physical Design: From Graph Partitioning to Timing Closure*. [S.l.]: Springer, 2011. 310 p.

KASAMSETTY, K.; KETKAR, M.; SAPATNEKAR, S. A new class of convex functions for delay modeling and their application to the transistor sizing problem. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 19, p. 779–788, July 2000.

KEATING, M. et al. *Low Power Methodology Manual for System-on-Chip Design*. [S.l.]: Springer, 2007. 300 p.

KIM, N. S. et al. Leakage current: Moore's law meets static power. *Computer*, v. 36, p. 68–75, December 2003.

KREHER, D. L.; STINSON, D. R. *Combinatorial Algorithms: Generation, Enumeration and Search*. [S.l.]: CRC Press, 1999. 329 p.

LAZZARI, C. *Automatic Layout Generation of Static CMOS Circuits Targeting Delay and Power Reduction*. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Sul, RS-Brazil, 2003.

LEE, J.; GUPTA, P. Discrete circuit optimization: Library based gate sizing and threshold voltage assignment. *Foundations and Trends in Electronics Design Automation*, v. 6, p. 1–120, Jan 2012.

LI, L. et al. An efficient algorithm for library-based cell-type selection in high-performance low-power designs. In: *Proceedings of International Conference on Computer-Aided Design*. [S.l.: s.n.], 2012. p. 226–232.

LI, W. N. Strongly NP-hard discrete gate-sizing problems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 13, p. 1045–1051, Aug 1994.

LIBERTY. *Open Source Liberty*. 2012. <www.opensourceliberty.org>. Acessado em 01/12/2012.

LIU, Y.; HU, J. A new algorithm for simultaneous gate sizing and threshold voltage assignment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 2, p. 223–234, Feb 2010.

LIVRAMENTO, V. dos S. et al. Evaluating the impact of slew on delay and power of neighboring gates in discrete gate sizing. In: *Proceedings of Latin American Symposium On Circuits and Systems*. [S.l.: s.n.], 2012. p. 1–4.

LIVRAMENTO, V. dos S. et al. Lagrangian relaxation-based discrete gate sizing for leakage power minimization. In: *Proceedings of International Conference on Electronics Circuits and Systems*. [S.l.: s.n.], 2012. p. 468–471.

LIVRAMENTO, V. dos S. et al. Fast and efficient lagrangian relaxation-based discrete gate sizing. In: *Proceedings of Design, Automation and Test in Europe*. [S.l.: s.n.], 2013.

MAHESHWARI, N.; SAPATNEKAR, S. *Timing Analysis and Optimization of Sequential Circuits*. [S.l.]: Springer, 1998. 208 p.

MOORE, G. E. Cramming more components onto integrated circuits. *Electronics Magazine*, p. 1–4, 1965.

MOSEK. *Mosek*. 2013. <www.mosek.com/>. Acessado em 15/01/2013.

NATARAJAN, S. et al. A 32nm logic tech. featuring 2nd-generation high-k + metalgate transistors, enhanced channel strain and 0.171m2 sram cell size in a 291mb array. In: *Proceedings of the International Electron Devices Meeting*. [S.l.: s.n.], 2008. p. 1–3.

NGUYEN, D. et al. Minimization of dynamic and static power through joint assignment of threshold voltages and sizing optimization. In: *Proceedings of International Symposium on Low Power Electronics and Design*. [S.l.: s.n.], 2003. p. 158–163.

OZDAL, M. M. et al. The ispd-2012 discrete cell sizing contest and benchmark suite. In: *Proceedings of International Symposium on Physical Design*. [S.l.: s.n.], 2012. p. 161–164.

OZDAL, M. M.; BURNS, S.; HU, J. Gate sizing and device technology selection algorithms for high-performance industrial designs. In: *Proceedings of International Conference on Computer-Aided Design*. [S.l.: s.n.], 2011. p. 724–731.

OZDAL, M. M.; BURNS, S.; HU, J. Algorithms for gate sizing and device parameter selection for high-performance designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 31, p. 1558–1571, Oct 2012.

RABAEY, J. *Low Power Design Essentials*. [S.l.]: Springer, 2009. 300 p.

RABAEY, J.; CHANDRAKASAN, A.; NIKOLIC, B. *Digital Integrated Circuits: a design perspective*. [S.l.]: Pearson Education, 2003. 761 p.

RAHMAN, M.; TENNAKOON, H.; SECHEN, C. Power reduction via near-optimal library-based cell-size selection. In: *Proceedings of the Design, Automation and Test in Europe*. [S.l.: s.n.], 2011. p. 867–870.

RAHMAN, M.; TENNAKOON, H.; SECHEN, C. Post-synthesis leakage power minimization. In: *Proceeding of the Design, Automation and Test in Europe*. [S.l.: s.n.], 2012. p. 99–104.

REN, H.; DUTT, S. A network-flow based cell sizing algorithm. In: *Proceedings of the International Workshop on Logic and Synthesis*. [S.l.: s.n.], 2008. p. 7–14.

ROY, S. et al. Numerically convex forms and their application in gate sizing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 26, p. 1637–1647, Sept 2007.

ROY, S. et al. An optimal algorithm for sizing sequential circuits for industrial library based design. In: *Proceeding of the Asia and South Pacific Design Automation Conference*. [S.l.: s.n.], 2008. p. 148–151.

ROY, S. et al. An optimal algorithm for sizing sequential circuits for industrial library based designs. In: *Proceedings of the Asia and South Pacific Design Automation Conference*. [S.l.: s.n.], 2008. p. 148–151.

SAKURAI, T. Perspectives on power-aware electronics. In: *Proc. of the International Solid-State Circuits Conference*. [S.l.: s.n.], 2003. p. 26–29.

SAPATNEKAR, S. *Timing*. [S.l.]: Springer, 2004. 306 p.

SIRICHOTIYAKUL, S. et al. Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing. In: *Proceedings of Design Automation Conference*. [S.l.: s.n.], 1999. p. 436–441.

SRIVASTAVA, A. Simultaneous vt selection and assignment for leakage optimization. In: *Proc. of the International Symposium on Low Power Electronics and Design (ISLPED'03)*. [S.l.: s.n.], 2003. p. 146–151.

SRIVASTAVA, A.; SYLVESTER, D.; BLAAUW, D. Power minimization using simultaneous gate sizing dual-vdd and dual-vth assignment. In: *Proceedings of the Design Automation Conference*. [S.l.: s.n.], 2004. p. 783–787.

S.SHAH et al. Discrete vt assignment and gate sizing using a self- snapping continuous formulation. In: *Proceeding of International Conference on Computer-Aided Design*. [S.l.: s.n.], 2005. p. 705–711.

SYNOPSYS. *Synopsys PrimeTime User's Manual*. 2012. <http://www.synopsys.com>. Acessado em 01/12/2012.

SYNTHESIS, I. W. on L. *IWLS 2005 Benchmarks*. 2012. <http://iwls.org/iwls2005/benchmarks.html>. Acessado em 01/12/2012.

TENNAKOON, H.; SECHEN, C. Gate sizing using lagrangian relaxation combined with a fast gradient-based pre-processing step. In: *Proceedings of International Conference on Computer-Aided Design*. [S.l.: s.n.], 2002. p. 395–402.

TENNAKOON, H.; SECHEN, C. Nonconvex gate delay modeling and delay optimization. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 27, p. 1583–1594, Sept 2008.

WANG, J.; DAS, D.; ZHOU, H. Gate sizing by lagrangian relaxation revisited. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, v. 28, p. 1071–1084, July 2009.

WESTE, N.; HARRIS, D. *CMOS VLSI Design: a system and circuit perspective*. [S.l.]: Addison-Wesley, 2010. 864 p.