UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Chrystian de Sousa Guth

ANÁLISE DE *TIMING* ESTÁTICA E A AVALIAÇÃO DO IMPACTO DO ATRASO DAS INTERCONEXÕES EM CIRCUITOS DIGITAIS

Florianópolis - Santa Catarina

2013

Chrystian de Sousa Guth

ANÁLISE DE *TIMING* ESTÁTICA E A AVALIAÇÃO DO IMPACTO DO ATRASO DAS INTERCONEXÕES EM CIRCUITOS DIGITAIS

Trabalho de Conclusão de Curso submetido ao Curso de Bacharelado em Ciências da Computação para a obtenção do Grau de Bacharel em Ciências da Computação.
Orientador: M.Sc. Vinícius dos Santos Livramento
Coorientador: Prof. Dr. José Luís Almada Güntzel

Florianópolis - Santa Catarina

2013

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

Guth, Chrystian
Análise de Timing Estática e a Avaliação do Impacto do Atraso das Interconexões em Circuitos Digitais / Chrystian Guth ; orientador, Vinícius dos Santos Livramento ; co-orientador, José Luís Almada Güntzel. - Florianópolis, SC, 2013.
89 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico. Graduação em Ciências da Computação.

Inclui referências

1. Ciências da Computação. 2. Análise de Timing Estática. 3. Standard Cell. 4. Automação de Projeto Eletrônico. 5. Atraso de Interconexões. I. dos Santos Livramento, Vinícius. II. Almada Güntzel, José Luís. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

Chrystian de Sousa Guth

ANÁLISE DE *TIMING* ESTÁTICA E A AVALIAÇÃO DO IMPACTO DO ATRASO DAS INTERCONEXÕES EM CIRCUITOS DIGITAIS

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de "Bacharel em Ciências da Computação", e aprovado em sua forma final pelo Curso de Bacharelado em Ciências da Computação.

Florianópolis - Santa Catarina, 09 de Dezembro 2013.

Prof. Dr. Renato Cislaghi
Coordenador

Banca Examinadora:

M.Sc. Vinícius dos Santos Livramento
Orientador

Prof. Dr. José Luís Almada Güntzel
Coorientador

Dr. Renan Alves Fonseca

À minha família.

AGRADECIMENTOS

À minha mãe, Ieda, pelo amor, apoio e dedicação que nunca faltaram. Também aos meus irmãos, Ralf e Elis Regina, pela força e confiança nesses 4 anos de graduação.

Agradeço à minha namorada Lígia pelo amor, compreensão e paciência, principalmente nos últimos meses em que me dediquei a este trabalho.

Ao meu orientador, Vinícius dos Santos Livramento, pela confiança, dedicação e aprendizado proporcionado desde o início de 2011, em que fui seu assistente, até a conclusão deste trabalho. Agradeço também por sua excelente orientação e rigor exigido, os quais foram fundamentais para o sucesso deste trabalho de conclusão de curso.

Ao meu coorientador, professor José Luís Almada Güntzel, pela grande colaboração, que muito contribuiu para a conclusão deste trabalho.

Ao membro da banca, Renan Alves Fonseca, pelo tempo dedicado para uma revisão rigorosa e pelas sugestões que contribuíram com este trabalho.

Aos colegas da graduação André Camargo e Cláudio Dettoni, Gabriel Gava, Lucas Pereira, Renan Netto e demais colegas do ECL que de alguma forma participaram deste trabalho.

Ao CNPq pelo custeio parcial da execução deste trabalho, com bolsa na modalidade de iniciação tecnológica (Processo número: 182980/2013-8).

RESUMO

Análise de *timing* estática (*STA: Static Timing Analysis*) é a técnica mais utilizada para estimar o atraso de circuitos digitais durante o fluxo de síntese física. Com o advento das tecnologias CMOS (Complementary Metal-Oxide Semiconductor) nanométricas, o atraso das interconexões passou a ser dominante em relação ao atraso das portas lógicas e por este motivo, não pode mais ser desprezado. A técnica de Elmore, baseada no primeiro momento da resposta ao impulso é amplamente utilizada para se calcular os atrasos das interconexões, porém, pode ser imprecisa por desconsiderar o efeito de *resistive shielding*. Algumas técnicas modificam a técnica de Elmore, a fim de contornar o problema do efeito de *resistive shielding*, obtendo resultados mais precisos, mantendo um baixo custo computacional. A consideração do efeito de *resistive shielding* requer a implementação de uma técnica para obtenção da capacitância efetiva em cada segmento da interconexão, impactando também no atraso da porta lógica que a interconexão está ligada na saída (*driver*). A ferramenta de STA implementada neste trabalho realiza o cálculo dos atrasos das interconexões gerando resultados que são, em média, $4,28\%$ mais otimistas do que aqueles gerados pela ferramenta Synopsys PrimeTime, porém com tempo de execução cerca de 8 vezes menor.

Palavras-chave: Automação de Projeto Eletrônico (EDA), Biblioteca *Standard Cell*, Análise de *Timing* Estática *(STA)*, *Complementary Metal-Oxide Semiconductor*

ABSTRACT

Static timing analysis is the most used technique to calculate the critical path in digital circuits during the standard cell design flow. Since feature size of CMOS devices are reducing, the interconnect delay becomes much more significant than before. Elmore delay model, based on the first moment of the impulse response is widely used to compute the interconnect delay, but, the technique doesn't consider the effect called resistive shielding. Some techniques modify the Elmore delay model in order to workaround the problem caused by the resistive shielding effect, getting more acurately results, keeping a low computational cost. The consideration of the resistive shielding effect requires the implementation of a technique to obain the effective capacitance value in each interconnect segment. This effective capacitance value impacts in the driver delay and slew. The STA tool implemented in this work does the interconnect delay calculation, getting results that are, in average 4.28% optimistics than those that are obtained by an industrial tool, with 8 times less runtime.

Keywords: Electronic Design Automation (EDA), Standard Cell Library, Static Timing Analysis, Complementary Metal-Oxide Semiconductor

LISTA DE FIGURAS

Figura 1 Fluxo de projeto *Standard Cell*. Adaptado de (BHASKER; CHADHA, 2009). 28
Figura 2 (a) Uma porta lógica *CMOS* $u1$, de função **NAND**, com duas entradas é *driver* da interconexão *n1*. (b) Algumas características temporais (*delay* e *slew*) da porta lógica $u1$. 33
Figura 3 Uma *lookup table* para atraso de subida (*rise delay*) de um arco de *timing*. As linhas são endereçadas por *load* (capacitância de saída da porta lógica) e as colunas por *input slew* (*slew* aplicado na entrada do *timing arc*). Adaptada de (OZDAL et al., 2013). . . . 35
Figura 4 Modelo RC Distribuído. Obtida de (RABAEY; CHANDRAKASAN; NIKOLIC, 2008). 36
Figura 5 Modelo de Capacitância Concentrada. Adaptada de (RABAEY; CHANDRAKASAN; NIKOLIC, 2008). 37
Figura 6 Uma árvore RC. Obtida de (RABAEY; CHANDRAKASAN; NIKOLIC, 2008). 38
Figura 7 Representações utilizadas para as árvores RC em um contexto de *pre-layout*. Obtida de (BHASKER; CHADHA, 2009). 39
Figura 8 (a) Interconexão RC obtida do circuito *simple* da competição de *sizing* do ISPD. (b) SPEF referente à Figura 8(a). 41
Figura 9 (a) Um circuito composto por três portas lógicas ($u1$, $u2$ e $u3$), uma célula sequencial ($f1$) e uma interconexão em forma de árvore RC, que liga a saída de $u1$ às entradas de $u2$, $u3$ e $f1$; (b) São apresentadas as modelagens para os *timing arcs* da porta lógica $u1$; O modelo da interconexão é abstraído, recebendo um valor de capacitância efetiva. As setas indicam que a interconexão oferece um atraso e uma degradação no *slew*. Cada destino da interconexão é representado como um valor de capacitância de seus pinos de entrada. 43
Figura 10 O grafo correspondente à interconexão da Figura 9(a), com cinco vértices e quatro arestas. 46
Figura 11 Formas de onda na saída de uma porta lógica em função da abordagem utilizada para cálculo da capacitância. Obtida de (BHASKER; CHADHA, 2009). 49
Figura 12 Visão geral da técnica iterativa para o cálculo do atraso da interconexão, capacitância efetiva e degradação do *slew*. Adap-

tada de (PURI; KUNG; DRUMM, 2002). 50
Figura 13 Cálculo da capacitância efetiva utilizando rampa de entrada. Obtida de (PURI; KUNG; DRUMM, 2002). 50
Figura 14 Degradação no *slew* em um segmento de uma árvore RC. Obtida de (PURI; KUNG; DRUMM, 2002). 53
Figura 15 Análise de *timing* estática. Adaptado de (BHASKER; CHADHA, 2009). 58
Figura 16 (a) Circuito *simple* retirado do banco de *benchmarks* da competição de *sizing* do ISPD; (b) Grafo correspondente ao circuito da letra (a). 59
Figura 17 Grafo de *timing* dividido em dois sub-circuitos devido à existência de uma célula sequencial. 59
Figura 18 Grafo de *timing* com célula sequencial atuando como entrada e saída primária do circuito. 60
Figura 19 Grafo de *timing* com representação dos *timing points*, *timing arcs* e interconexões. 61
Figura 20 Na lista ordenada, observando o elemento *u1:o*, os elementos de menor ou de igual nível lógico (*fonte, inp1, inp2, f1:q, u1:a, u1:b, u2:a*) se encontram à esquerda, e os de maior ou igual (*u2:o, f1:d, out, terminal*) se encontram à direita. 64
Figura 21 Fluxo utilizado na validação da ferramenta implementada neste trabalho perante a ferramenta industrial *PrimeTime*. . . 68
Figura 22 Modelagem utilizada no *PrimeTime* para o *driver*, interconexão e destinos. Obtida de (SYNOPSYS, 2013). 71
Figura 23 Distribuição das frequências dos erros percentuais calculados nas saídas primárias dos circuitos: (a) *matrix_mult*; (b) *pci_bridge32*. Na primeira parte os erros foram calculados considerando a configuração de menor consumo de *leakage*. Na segunda parte, considerando a configuração padrão. E na terceira, considerando a configuração de maior consumo de *leakage*. μ e σ representam a média e desvio padrão das amostras, respectivamente. 80
Figura 24 Distribuição das frequências das relações C_{eff}/C_{total} das interconexões do circuito *pci_bridge32*. Na primeira parte, as frequências são das interconexões com menor valor de resistência total, na segunda parte, das com valor de resistência total médio, e na terceira, das com maiores valores de resistência total. 81
Figura 25 Distribuição das frequências das relações C_{eff}/C_{total} das interconexões do circuito *matrix_mult*. Na primeira parte, as frequências são das interconexões com menor valor de resistência

total, na segunda parte, das com valor de resistência total médio, e na terceira, das com maiores valores de resistência total. 82
Figura 26 Erro relativo dos *arrival times* em relação aos resultados obtidos pelo *PrimeTime*, ao decorrer dos níveis lógicos, no *benchmark pci_bridge*32. O *arrival time* utilizado na comparação é o *arrival time* no *timing point* de saída de cada porta lógica. Em azul, cada ponto representa uma porta lógica. Em vermelho, é a curva referente às portas lógicas pertencentes ao caminho crítico. A curva em verde, é referente às portas lógicas pertencentes ao maior caminho, ou seja, ao caminho com maior número de portas. 83
Figura 27 Erro relativo dos *arrival times* em relação aos resultados obtidos pelo *PrimeTime*, ao decorrer dos níveis lógicos, no *benchmark matrix_mult*. O *arrival time* utilizado na comparação é o *arrival time* no *timing point* de saída de cada porta lógica. Em azul, cada ponto representa uma porta lógica. Em vermelho, é a curva referente às portas lógicas pertencentes ao caminho crítico. A curva em verde, é referente às portas lógicas pertencentes ao maior caminho, ou seja, ao caminho com maior número de portas. 84

LISTA DE TABELAS

Tabela 1 Técnicas validadas nos experimentos e as respectivas tabelas que apresentam os resultados obtidos. 67
Tabela 2 Comparação das informações de *timing* calculadas pela ferramenta implementada *versus* informações fornecidas pelo *PrimeTime*, utilizando o modelo de interconexões de capacitância concentrada. 70
Tabela 3 Valores obtidos pelo *PrimeTime* no *benchmark* experimental utilizado neste trabalho. 73
Tabela 4 Valores obtidos pela ferramenta implementada neste trabalho nos circuitos da competição de *sizing* do ISPD. 74
Tabela 5 Experimentos utilizando o modelo capacitância concentrada para carga de saída dos *drivers*, técnica de Elmore para computar os atrasos das interconexões, e degradação do *slew* conforme apresentado no Capítulo 3. 75
Tabela 6 Experimentos utilizando o modelo capacitância efetiva para carga de saída dos *drivers*, técnica de Elmore utilizando as capacitâcias efetivas de cada nodo interno das interconexões, para computar seus atrasos. Neste experimento, a degradação do *slew* não foi considerada. 76
Tabela 7 Relação C_{eff}/C_{total} média por circuito. 77

LISTA DE ABREVIATURAS E SIGLAS

RTL Register Transfer Level ... 27
STA Static Timing Analysis ... 27
VLSI Very-large-scale integration ... 29
CMOS Complementary Metal-Oxide Semiconductor ... 30
EDA Electronic Design Automation ... 30
RC Resistor-Capacitor ... 31
ISPD International Symposium on Physical Design ... 31
NLDM Non-Linear Delay Model ... 35
SPEF Standard Parasitic Exchange Format ... 39
SPF Standard Parasitic Format ... 39
DSPF Detailed Standard Parasitic Format ... 39
RSPF Reduced Standard Parasitic Format ... 39
IEEE Institute of Electrical and Electronics Engineers ... 39
AWE Asymptotic Waveform Evaluation ... 45
PRIMA Passive Reduced-Order Interconnect Macromodeling Algorithm ... 45
HDL Hardware Description Language ... 57
PERT Program Evaluation and Review Technique ... 61
CPM Critical Path Method ... 61
TNS Total Negative Slack ... 68
PO Primary Output ... 68
RAM Random-Access Memory ... 70
GB Gigabyte ... 70
EMPA Erro Médio Percentual Absoluto ... 70
D2M Delay With 2 Moments ... 78
PERI Probability Distribution Function Extension for Ramp Inputs ... 78

LISTA DE SÍMBOLOS

C Capacitor 50

R Resistor 50

C_{eff} Capacitância Efetiva 52

$slew_i$ Slew no nodo i de uma interconexão 53

$Ceff_i$ Capacitância efetiva no nodo i de uma interconexão 54

$Ctotal_i$ Capacitância total *downstream* no nodo i de uma interconexão 54

τ_i Atraso de Elmore no nodo i de uma interconexão 54

K_j Fator de *shielding* correspondente ao efeito causado pelo resistor R_j no nodo j 55

ε Menor número representável em ponto flutuante, utilizado como métrica de precisão 55

$G(V, E)$ Grafo de *timing* 60

V $\{ v_i | v_i$ é um *timing point* (pino de *timing*), que pode ser a entrada ou saída de uma porta lógica, aqui referenciado como pino. Um *timing point* pode também representar uma entrada ou saída primária do circuito. $\}$ 60

I $\{ (v_i, v_j) | v_i, v_j \in V$ e (v_i, v_j) é uma interconexão do circuito, que conecta v_i em v_j. v_i é um pino de saída de uma porta lógica ou uma entrada primária, e v_j pode ser a entrada de uma porta lógica ou uma saída primária. $\}$ 60

A $\{ (v_i, v_j) | v_i, v_j \in V$ e (v_i, v_j) é um *timing arc*. Portanto, v_i e v_j são pinos de entrada e saída (respectivamente) de uma mesma porta lógica. $\}$ 60

E I $\cup$ A 60

$inputs(i)$ Conjunto de *timing points* que se ligam com v_i através de um *timing arc*. Todo $v_j \in input(i)$ é necessariamente um pino de entrada de uma porta lógica, e v_i é um pino de saída 61

a_i *arrival time*, ou tempo de chegada no pino v_i. O *arrival time* é definido pelo atraso do caminho parcial que inicia em uma entrada primária e termina em v_i 61

$slew_i$ O *slew* no pino v_i; 61

$d_{j \to i}$ O *delay* do *timing arc* que vai do pino v_j até o pino v_i 61

$slew_{j \to i}$
O *slew* do *timing arc* que vai do pino v_j até o pino v_i 61

$iD_{i \to k}$ O atraso de propagação na interconexão que liga o pino v_i até o pino v_k. No modelo de capacitância concentrada, $iD_{i \to k} = 0$ 61

$iS_{i \to k}$ Degradação do *slew* através da interconexão que liga v_i em v_k 61

$fanouts(i)$
Conjunto dos pinos que são destino da interconexão para qual v_i é *driver* 61

r_i É o *required time* no *timing point* v_i. O *required time* é o tempo máximo que o valor de a_i pode assumir para que a restrição de desempenho seja respeitada. Se v_i é uma saída primária do circuito, então $r_i = T$, onde $f = \frac{1}{T}$ é a frequência mínima de operação do circuito digital 61

$slack_i$ Folga de tempo no ponto v_i, ou seja, quanto o *arrival time* pode atrasar neste ponto, de modo que o período máximo continue sendo respeitado. Se em um determinado ponto do circuito o *slack* é negativo, então o caminho em questão está violando a restrição de atraso máximo do sistema 61

SUMÁRIO

1 INTRODUÇÃO 27
1.1 FLUXO DE PROJETO *STANDARD CELL* 27
1.2 MOTIVAÇÃO 29
1.3 JUSTIFICATIVA 30
1.4 OBJETIVOS 30
1.4.1 Objetivo Geral 30
1.4.2 Objetivos Específicos 31
1.5 ESCOPO 31
1.6 ORGANIZAÇÃO DESTE TRABALHO 32
2 CONCEITOS FUNDAMENTAIS DE CIRCUITOS DIGITAIS 33
2.1 CARACTERÍSTICAS TEMPORAIS DAS PORTAS LÓGICAS 33
2.1.1 Modelo de atraso adotado em fluxo *standard cell* 35
2.2 MODELOS DE INTERCONEXÃO 36
2.2.1 Modelo RC Distribuído (*Distributed RC Model*) 36
2.2.2 Modelo de Capacitância Concentrada (*Lumped C Model*) 37
2.2.3 Modelo RC Concentrado (*Lumped RC Model*) 37
2.2.4 Extração de Elementos Parasitas no Projeto de Circuitos Digitais 38
2.3 CARACTERÍSTICAS TEMPORAIS DAS INTERCONEXÕES 40
3 CÁLCULO DAS CARACTERÍSTICAS TEMPORAIS DA INTERCONEXÃO 45
3.1 REPRESENTAÇÃO DAS INTERCONEXÕES 45
3.2 CÁLCULO DO ATRASO DAS INTERCONEXÕES 45
3.3 TÉCNICA DE Puri, Kung e Drumm (2002) PARA O CÁLCULO DA CAPACITÂNCIA EFETIVA E DEGRADAÇÃO DO *SLEW* 49
3.3.1 Cálculo da Capacitância Efetiva 50
3.3.2 Degradação do *Slew* Através da Árvore RC 52
3.3.3 O Algoritmo de Puri, Kung e Drumm (2002) 54
4 ANÁLISE DE *TIMING* ESTÁTICA 57
4.1 REPRESENTAÇÃO DE CIRCUITOS DIGITAIS 58
4.2 CÁLCULO DO PIOR ATRASO DO CIRCUITO 61
4.3 IMPLEMENTAÇÃO DA FERRAMENTA DE STA 63
4.3.1 O Modelo de Grafo Adotado 64

4.3.2 Algoritmo de Análise de *Timing* Estática 64
5 EXPERIMENTOS 67
5.1 METODOLOGIA E INFRAESTRUTURA EXPERIMENTAL 67
5.2 VALIDAÇÃO DO MODELO DE CAPACITÂNCIA CONCENTRADA PERANTE FERRAMENTA INDUSTRIAL .. 70
5.3 ANÁLISE DE *TIMING* ESTÁTICA EM FERRAMENTA INDUSTRIAL 71
5.3.1 Modelagem de *Driver* (*Driver model*) 72
5.3.2 Modelagem do Destino (*Receiver model*) 72
5.3.3 Modelagem da Interconexão (*Reduced-order network model*) 72
5.3.4 Resultados Obtidos 73
5.4 VALIDAÇÃO DA TÉCNICA IMPLEMENTADA PERANTE FERRAMENTA INDUSTRIAL 73
5.4.1 Relação entre C_{eff} e C_{total} 75
5.4.2 Erro de *Arrival Times* nas Saídas Primárias 78
5.4.3 Nível Lógico *versus* Erro Relativo 78
6 CONCLUSÃO 85
6.1 TRABALHOS FUTUROS 85
Referências Bibliográficas 87

1 INTRODUÇÃO

Este capítulo tem por objetivo, apresentar uma visão geral sobre o fluxo de projeto *standard cell* e a importância da análise de *timing* estática no desenvolvimento de circuitos digitais. Serão apresentadas também a motivação e a justificativa deste trabalho.

1.1 FLUXO DE PROJETO *STANDARD CELL*

O crescimento da complexidade dos circuitos digitais contemporâneos[1] e a necessidade de um *time-to-market* (tempo de entrega ao mercado) curto faz com que o projeto de tais circuitos adote o fluxo *standard cell* (Figura 1).

No fluxo *standard cell* as portas lógicas são caracterizadas e validadas previamente em uma dada tecnologia, originando as chamadas "células". Essas células [2] são catalogadas com suas diversas características elétricas em uma biblioteca, podendo ser reutilizadas em diversos projetos que usem a mesma tecnologia. O reuso amortiza o custo dos projetos inseridos neste nodo tecnológico e possibilita um *time-to-market* mais curto.

O fluxo *standard cell* pode ser subdividido em etapas, e ao decorrer dessas etapas, a análise de *timing* pode ser requisitada milhares de vezes. De acordo com Bhasker e Chadha (2009), essas são algumas das etapas importantes no fluxo *standard cell*:

- **Síntese:** Responsável por criar uma representação em nível de portas lógicas, a partir de uma descrição no nível de transferência entre registradores (*RTL: Register Transfer Level*). A partir desta etapa, a análise de *timing* estática (*STA: Static Timing Analysis*) é utilizada, para estimar as características *temporais* do circuito;

- **Otimização Lógica:** Responsável por minimizar a lógica do circuito sintetizado. A análise de *timing* estática pode ser realizada antes desta etapa, para verificar os caminhos de maior atraso, também chamados de caminhos críticos. Se a análise de *timing* for realizada depois desta etapa, o objetivo é identificar

[1] Um processador para *desktop* desenvolvido no ano de 2008 tem cerca de 731 milhões de transistores, excluindo a área de memória (INTEL, 2008).

[2] Célula é a instância de *layout* para a implementação física de uma porta lógica.

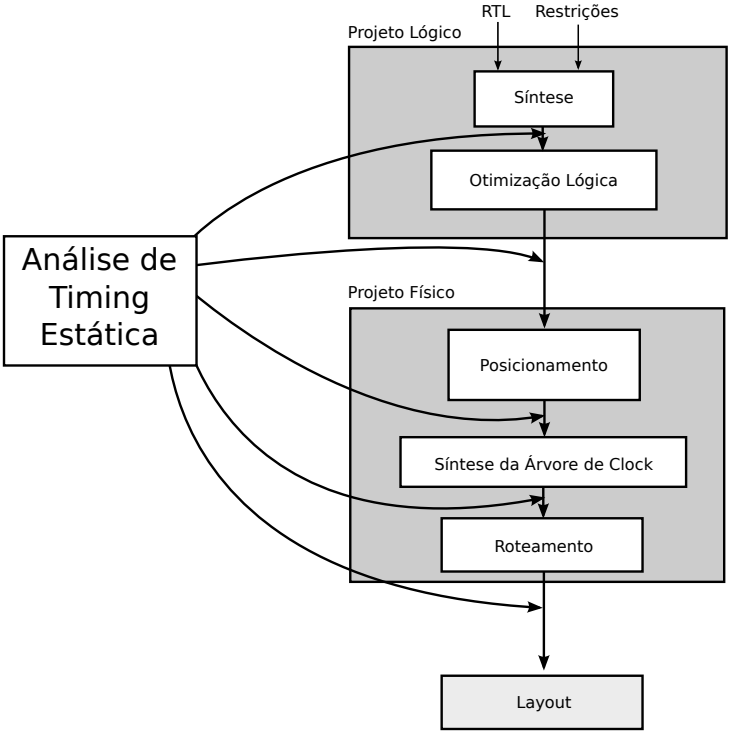


Figura 1 – Fluxo de projeto *Standard Cell*. Adaptado de (BHASKER; CHADHA, 2009).

quais caminhos ainda precisam ser otimizados ou identificar os caminhos críticos;

- **Posicionamento:** Define a localização espacial dos *layouts* das células. Antes dessa etapa, modelos de interconexão ideais são adotados, pois ainda não se possui as informações necessárias de posicionamento. Uma maneira alternativa para se modelar as interconexões é utilizar um modelo de *wireload*, que estima o tamanho das interconexões de acordo com o seu número de destinos, ou *fanouts*;

- **Síntese da Árvore de *Clock*:** No início da síntese física, as árvores dos relógios são consideradas como ideais, ou seja, não possuem atraso de propagação. O objetivo desta etapa é mini-

mizar o *clock skew*, que é a diferença entre os tempos de chegada do sinal de relógio nas entradas dos registradores. A análise de *timing* estática é importante nesta etapa para avaliar essas diferenças nos tempos de chegada.

- **Roteamento:** Responsável por criar as conexões entre as diferentes células incluídas no projeto, utilizando as diferentes camadas de metal. As mudanças nas topologias nesta etapa necessitam diversas avaliações das informações temporais.

Os projetos de circuitos digitais no fluxo *standard cell* são realizados visando, além das funcionalidades requisitadas, a operação em uma frequência especificada. Por isso, diversas otimizações são efetuadas ao longo do fluxo, para que tais funcionalidades consigam ser realizadas na frequência definida. Nas primeiras etapas de um projeto no fluxo *standard cell*, apenas as questões relacionadas à funcionalidade do projeto são verificadas, pois ainda não estão disponíveis informações detalhadas referentes ao comportamento elétrico do circuito. Nas etapas posteriores, as informações temporais precisam ser avaliadas com precisão, para que as etapas de otimização garantam a satisfação dos requisitos do projeto.

1.2 MOTIVAÇÃO

No fluxo *standard cell*, a partir da descrição RTL, uma série otimizações são realizadas ao decorrer de suas etapas. Como consequência dessas otimizações, as topologias das interconexões se alteram, levando à necessidade, ao decorrer do fluxo, de diversas avaliações de suas informações temporais. Kahng et al. (2013) trata do problema de *gate sizing* utilizando diversas modelagens para os atrasos das interconexões, bem como seu impacto na propagação do *slew* do circuito.

Em diversos sistemas projetados atualmente, entre 50% a 70% do ciclo de relógio é "consumido" pelo atraso de propagação de suas interconexões (CONG et al., 1996). Nas tecnologias com alta escala de integração (*VLSI: Very-large-scale integration*) atuais, onde diversas otimizações tem por objetivo reduzir a resistência dos *drivers*, as interconexões passam a ser cada vez mais impactantes no desempenho do circuito digital.

Durante as otimizações nas etapas iniciais do fluxo (*pre-layout*), a análise de *timing* é requisitada milhares de vezes, sendo assim necessário que a ferramenta de análise de *timing* tenha o melhor desem-

penho possível. Como as informações relacionadas ao aspecto físico do circuito, como posicionamento (WANG; YANG; SARRAFZADEH, 2000) e roteamento (RYZHENKO; BURNS, 2012) nas etapas iniciais precisam ser aproximadas, a ferramenta de análise de *timing* fornece estimativas pessimistas sobre o *timing* do circuito.

Já nas etapas finais (*pós-layout*), a análise de *timing* precisa ser a mais precisa possível. Porém, a modelagem dos elementos dos circuitos digitais torna-se mais complexa, diminuindo o desempenho da ferramenta.

Como as informações de *timing* precisam ser avaliadas centenas ou milhares de vezes durante os processos de otimização, ferramentas de análise de *timing* eficientes e escaláveis precisam ser desenvolvidas e aperfeiçoadas para acompanhar a evolução da tecnologia *CMOS* (*Complementary Metal-Oxide Semiconductor*).

1.3 JUSTIFICATIVA

Diversas otimizações são realizadas no decorrer do fluxo de projeto *standard cell* e o uso de ferramentas para a automação de projeto eletrônico (*EDA: Electronic Design Automation*) é indispensável em suas diferentes etapas. A inexistência de ferramentas de análise de *timing* estática precisas de domínio público e a restrição no acesso à ferramentas industriais (devido ao alto custo de suas licenças) resultam em um problema de infraestrutura de pesquisa. Assim, este trabalho tem como resultado uma alternativa de ferramenta de análise de *timing* para projetistas de circuitos digitais, bem como uma infraestrutura realista e precisa para desenvolvedores de ferramentas, que necessitam da análise de *timing* em alguma etapa do fluxo de projeto *standard cell*.

1.4 OBJETIVOS

1.4.1 Objetivo Geral

Este trabalho tem por objetivo o projeto, validação, avaliação e documentação de uma ferramenta de análise de *timing* estática voltada para o fluxo *standard cell*.

1.4.2 Objetivos Específicos

1. Avaliação e análise experimental do modelo de interconexão com capacitância concentrada, desprezando-se o impacto das resistências;

2. Avaliação e análise experimental da técnica de Elmore para cálculo do atraso das interconexões baseando-se em um modelo de interconexão RC concentrado;

3. Avaliação e análise experimental da técnica para cálculo do atraso de interconexões utilizando a abordagem de capacitância efetiva;

4. Construção de uma ferramenta de análise de *timing* estática incluindo as funcionalidades descritas nos objetivos 1, 2 e 3, bem como sua validação empírica perante uma ferramenta de análise de *timing* industrial.

1.5 ESCOPO

Este trabalho aborda o problema da análise de *timing* estática utilizando técnicas para estimação dos atrasos das interconexões. A análise de *timing* é realizada propagando os atrasos de cada porta lógica em ordem topológica, a fim de estimar o desempenho do circuito. Os modelos de atraso (*delay*) e *slew* utilizados neste trabalho são os mesmos utilizados no *fluxo standard cell*[3].

As interconexões serão modeladas de duas formas:

- **Modelo da capacitância concentrada**, impactando apenas nos *atrasos* de seus *drivers*;

- **Modelo RC concentrado**[4], apresentando também, seus próprios atrasos como impacto no atraso do circuito.

Não faz parte do escopo deste trabalho a consideração dos tempos de *setup* e *hold* das células sequenciais, como os registradores. Eles serão modelados pelo *timing arc*[5] da entrada de relógio até a saída.

[3] O cálculo dos atrasos das portas lógicas será melhor apresentado na Seção 2.1.

[4] Este modelo pode ser chamado de modelo RC distribuído em alguns trabalhos científicos, como na competição de *sizing* do *ISPD (International Symposium on Physical Design)* de 2013 (OZDAL et al., 2013).

[5] O conceito de *timing arc* será apresentado na Seção 2.1.

1.6 ORGANIZAÇÃO DESTE TRABALHO

Este trabalho está organizado da seguinte forma:

O Capítulo 2 apresenta os conceitos básicos essenciais para o entendimento do presente trabalho.

No Capítulo 3 é apresentada uma revisão bibliográfica acerca das técnicas utilizadas para cálculo do atraso das interconexões e da capacitância efetiva.

Já o Capítulo 4 trata da análise de *timing*, apresentando seus algoritmos e particularidades na implementação.

O Capítulo 5 apresenta os experimentos realizados utilizando a ferramenta implementada neste trabalho.

Finalmente, as conclusões e algumas perspectivas de trabalhos futuros são apresentadas no Capítulo 6.

2 CONCEITOS FUNDAMENTAIS DE CIRCUITOS DIGITAIS

Este capítulo apresenta os conceitos básicos relacionados à temporização e modelagem de circuitos digitais, essenciais para o entendimento do presente trabalho. A Seção 2.1 apresenta as características temporais das portas, assim como os modelos de atraso adotados no fluxo *standard cell*. Os modelos de interconexões e as suas características temporais serão apresentados nas Seções 2.2 e 2.3, respectivamente.

2.1 CARACTERÍSTICAS TEMPORAIS DAS PORTAS LÓGICAS

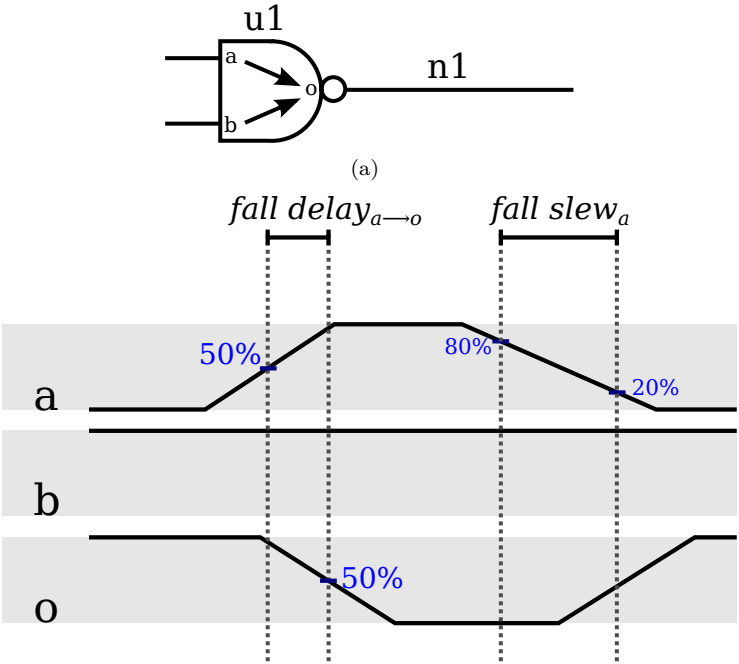


Figura 2 – (a) Uma porta lógica *CMOS* $u1$, de função **NAND**, com duas entradas é *driver* da interconexão *n1*. (b) Algumas características temporais (*delay* e *slew*) da porta lógica $u1$.

As características temporais do circuito são derivadas das características temporais de suas partes, quais sejam, as portas lógicas e as interconexões que o compõem. Para as portas lógicas, as informações a seguir são relevantes (LIVRAMENTO, 2013):

- ***Timing Arc* (Arco de Tempo):** é um conceito utilizado para associar um pino de entrada de uma porta com a saída dessa mesma porta. Uma porta *NAND* de duas entradas, como a apresentada na Figura 2(a) possui dois *timing arcs*: um entre a entrada a e a saída ($a \rightarrow o$) e outro entre a entrada b e a saída ($b \rightarrow o$). Para elementos sequenciais, como os registradores, normalmente consideram-se como *timing arcs* as conexões entre o sinal de relógio e as saídas. O arco é chamado *positive unate* se uma transição de subida (descida) na entrada causa uma transição de subida (descida) na saída. Se uma transição de subida (descida) da entrada causa uma transição de descida (subida) na saída, o arco é chamado *negative unate* (BHASKER; CHADHA, 2009).

- ***Delay* (Atraso de Propagação):** é o tempo que o sinal em um pino de saída o leva para atingir um limiar[1] de sua transição total, devido a uma mudança no sinal em um pino de entrada. Se a transição em o for do nível lógico 0 para 1, o atraso é chamado de atraso de subida (*rise delay*), caso o contrário, é chamado de atraso de descida (*fall delay*) (Figura 2).

- ***Slew* (Tempo de Transição):** é o tempo que um sinal leva para transicionar de uma porcentagem do valor de referência (V_{dd}) à outra (BHASKER; CHADHA, 2009)[2]. Se a transição for de um valor for de uma porcentagem menor para uma maior, ela é chamada de transição de subida (*rise slew*), caso contrário, trata-se de ua transição de descida (*fall slew*) (Figura 2).

- **Propagação do *Slew*:** é a política utilizada para propagação dos *slews* das entradas até as saídas das portas lógicas. A estratégia geralmente adotada é a de propagar para o pino de saída da porta lógica o maior dentre os *slews* associados aos *timing arcs*.

[1] Este limiar geralmente é definido nas bibliotecas de célula como sendo 50% do Vdd.

[2] Nas bibliotecas de células, essas porcentagens geralmente são definidas como 20% e 80% ou 10% e 90%

- ***Driver:*** é a porta lógica (ou o pino de saída de uma porta lógica) que gera o sinal para uma interconexão. Cada interconexão possui apenas um *driver*.

2.1.1 Modelo de atraso adotado em fluxo *standard cell*

Nas bibliotecas *standard cell* atuais, modelos de atrasos não-lineares[3] são fornecidos para os *timing arcs* das células disponíveis. Esses modelos, que geralmente são obtidos através de simulações em nível elétrico, são armazenados na forma de *lookup tables*, como a da Figura 3. Uma *lookup table* descreve o *delay* ou o *slew* de uma célula em função de dois fatores: o *slew* na entrada do *timing arc* (colunas), e a capacitância de saída (*load*) (linhas).

Utilizando a *lookup table* da Figura 3 para estimar o *delay* de um dos *timing arcs* de uma célula *CMOS* e supondo que o *slew* na entrada deste *timing arc* seja de 8.0, e a capacitância vista na saída seja 0.1, obtém-se que $delay = 3.49$, pois 3.49 é o valor endereçado pelos índices da função (*slew* e *load*). Caso os valores de *slew* ou *load* não existam na tabela, uma interpolação linear é realizada. Da mesma forma, o cálculo do *slew* do *timing arc* é realizado com base na *lookup table* específica para o *slew*.

```
rise_delay (delay_table) {
 load (0.0, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2) ;
 input_slew (0.5, 3.0, 5.0, 8.0, 14.0, 20.0, 30.0, 50.0) ;
 values (
   1.17, 1.82, 2.26, 2.76, 3.48, 4.04, 4.82, 6.12,
   1.69, 2.34, 2.86, 3.49, 4.41, 5.11, 6.06, 7.58,
   2.21, 2.86, 3.38, 4.12, 5.22, 6.05, 7.16, 8.90,
   3.25, 3.90, 4.42, 5.20, 6.60, 7.67, 9.08, 11.23,
   5.33, 5.98, 6.50, 7.28, 8.84, 10.30, 12.24, 15.14,
   9.50, 10.15, 10.67, 11.45, 13.01, 14.57, 17.15, 21.33,
   17.83, 18.48, 19.00, 19.78, 21.34, 22.90, 25.50, 30.70
  );
}
```

Figura 3 – Uma *lookup table* para atraso de subida (*rise delay*) de um arco de *timing*. As linhas são endereçadas por *load* (capacitância de saída da porta lógica) e as colunas por *input slew* (*slew* aplicado na entrada do *timing arc*). Adaptada de (OZDAL et al., 2013).

[3] Conhecidos na indústria por *NLDM* (*Non-Linear Delay Model*)

2.2 MODELOS DE INTERCONEXÃO

Modelos de interconexão devem ser adotados de acordo com a etapa que o projeto se encontra no fluxo. Nas etapas iniciais, ou de *pre-layout*, ainda não há informações sobre o posicionamento e sobre o roteamento. Assim, as interconexões recebem modelos simplistas, possibilitando que as otimizações necessárias sejam realizadas, sem degradação no desempenho, para que as informações reais dos parasitas sejam apuradas. Nas etapas mais próximas da síntese física, ou *pos-layout*, as interconexões são modeladas em função de suas capacitâncias e resistências, com o intuito de fornecer uma simulação mais precisa possível.

Esta seção tem por objetivo, apresentar alguns modelos de representação de interconexões, suas vantagens e desvantagens. Também será apresentado o formato de representação de parasitas mais utilizado no projeto de circuitos digitais.

2.2.1 Modelo RC Distribuído (*Distributed RC Model*)

Uma interconexão pode ser representada idealmente como uma linha distribuída (Figura 4): a linha de comprimento L é dividida em segmentos de tamanho ΔL, com $\Delta L \to 0$, e cada segmento é representado por um valor de resistência r e um valor de capacitância c. Assim, a resistência e a capacitância total da linha são $r \times L$ e $c \times L$, respectivamente. O cálculo dos atrasos no modelo RC distribuído implica na resolução de equações diferenciais, as quais possuem soluções complexas. Uma solução numérica seria realista, porém resulta em um custo computacional muito elevado, tornando inviável sua adoção em fluxo *standard cell*. Para tal objetivo, utilizam-se modelos de interconexão simplificados.

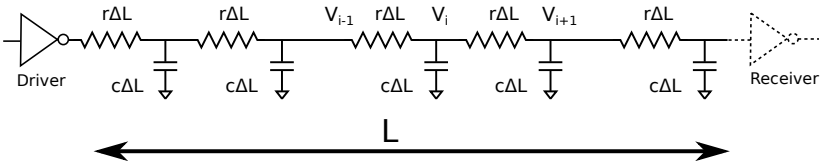


Figura 4 – Modelo RC Distribuído. Obtida de (RABAEY; CHANDRAKASAN; NIKOLIC, 2008).

2.2.2 Modelo de Capacitância Concentrada (*Lumped C Model*)

O Modelo de capacitância concentrada é geralmente utilizado nas etapas iniciais do projeto, pois se trata de um modelo simples com fácil simulação. Quando a resistência da interconexão é desprezível, devido o fato de que a resistência do *driver* é substancialmente maior que a resistência total da interconexão, ou quando as informações parasitas ainda não foram obtidas com detalhe, o fio pode ser representado como um capacitor C, que corresponde à capacitância total da interconexão. Seu atraso de propagação é desconsiderado, já que o fio não possui resistências. Seu único impacto no desempenho é a sua contribuição na capacitância vista pelo *driver* (RABAEY; CHANDRAKASAN; NIKOLIC, 2008).

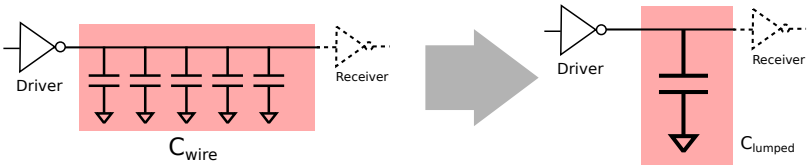


Figura 5 – Modelo de Capacitância Concentrada. Adaptada de (RABAEY; CHANDRAKASAN; NIKOLIC, 2008).

2.2.3 Modelo RC Concentrado (*Lumped RC Model*)

O modelo RC concentrado é amplamente adotado no fluxo *standard cell* para modelagem das interconexões. No modelo RC concentrado, concentra-se toda a resistência de cada segmento da interconexão em um único resistor R e similarmente, combina-se a capacitância total em um único capacitor C. A rede resistor-capacitor é normalmente representada como uma árvore RC (Figura 6). De acordo com Rabaey, Chandrakasan e Nikolic (2008), uma árvore RC possui as seguintes propriedades:

- A rede tem apenas um nodo de entrada, chamado de **fonte** (*source*);
- Todos os capacitores são entre um nodo e o terra;
- A rede não possui *loops* resistivos, por isso é chamada de Árvore.

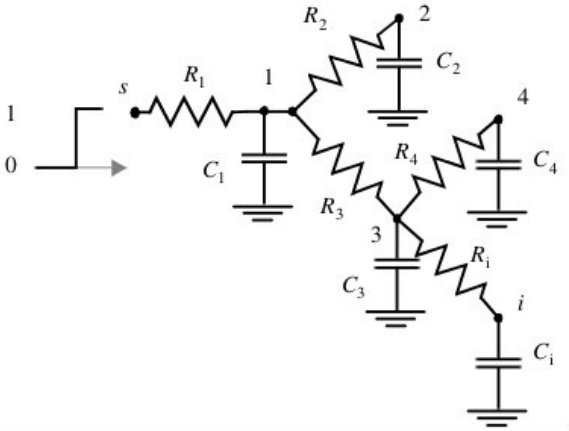


Figura 6 – Uma árvore RC. Obtida de (RABAEY; CHANDRAKASAN; NIKOLIC, 2008).

2.2.4 Extração de Elementos Parasitas no Projeto de Circuitos Digitais

Quando se tem as informações de capacitância e resistência totais de uma interconexão, C_{wire} e R_{wire} respectivamente, em uma fase de *pré-layout*, é necessário criar uma topologia para este fio, uma vez que o atraso da interconexão depende de como ela está estruturada. Existem três topologias (Figura 7) que podem ser utilizadas a fim de representar a interconexão (BHASKER; CHADHA, 2009):

- **Árvore de melhor caso (*Best-case tree*):** (Figura 7-a) Assume-se que cada pino de destino é fisicamente adjacente ao *driver*. Assim, nenhuma resistência estará no caminho entre *driver* e destino, e todos os pinos de destino atuarão como *load* na saída da interconexão.

- **Árvore balanceada (*Balanced tree*):** (Figura 7-b) Na árvore balanceada, todos os pinos de destino se encontram na mesma distância do *driver*, e o caminho para cada destino corresponde a mesma quantidade de capacitância e resistência que os outros caminhos.

- **Árvore de pior caso (*Worst-case tree*):** (Figura 7-c) Neste caso, todos os destinos se encontram no fim da interconexão. Assim, cada pino de destino vê a resistência e a capacitância total da interconexão.

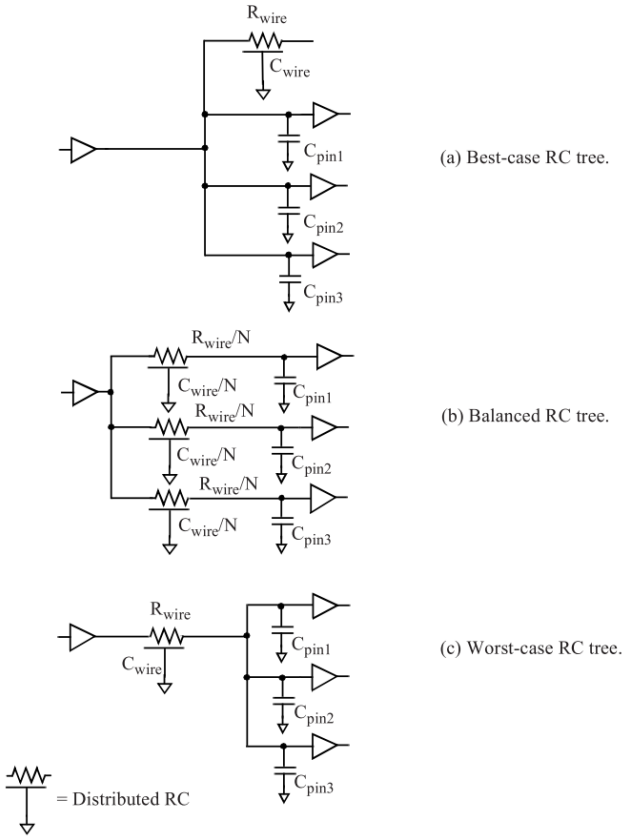


Figura 7 – Representações utilizadas para as árvores RC em um contexto de *pre-layout*. Obtida de (BHASKER; CHADHA, 2009).

Em projetos de circuitos digitais, as capacitâncias parasitas são geralmente descritas no formato SPEF [4] (*Standard Parasitic Exchange*

[4] Existem outros formatos como SPF (*standard parasitic format*), DSPF (*detailed standard parasitic format*), RSPF (*reduced standard parasitic format*) e SBPF (*Synopsys binary parasitic format*).

Format) definido pelo *IEEE* (*Institute of Electrical and Electronics Engineers*). O formato SPEF é um padrão feito para garantir a interoperabilidade entre ferramentas de automação de projeto eletrônico (*EDA: Electronic Design Automation*). Os parasitas podem ser representados em diferentes níveis de sofisticação, desde o simplista modelo de capacitância concentrada, até uma representação mais precisa de Árvores RC.

Um exemplo de interconexão descrita no formato SPEF (IEEE, 1999) pode ser visualizado nas Figuras 8(a) e 8(b). A linha 1 no código SPEF da Figura 8(b) apresenta o nome da interconexão (`inp1`) e o valor de sua capacitância total (5.4). As linhas 2, 3 e 4 indicam que existe uma conexão entre uma entrada primária `inp1`, indicado por `*P inp1 I`, e a entrada de um pino interno *a* da porta `u1`, indicado por `*I u1:a I`. Da linha 6 até a linha 9 são representadas as capacitâncias da árvore RC.

A representação de um capacitor num arquivo SPEF se dá pelo formato:

```
[Número] [Nome] [Capacitância]
```

De maneira semelhante, os resistores, como pode ser visto nas linhas 11 até 13, são descritos no formato:

```
[Número] [Capacitor Fonte] [Capacitor Destino] [Resistência]
```

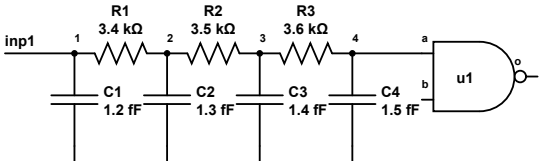
O valor $*END$ (linha 14) é utilizado para determinar o fim da descrição de uma interconexão.

2.3 CARACTERÍSTICAS TEMPORAIS DAS INTERCONEXÕES

A Figura 9 ilustra as três principais contribuições das interconexões, sobre o atraso do circuito:

- **Capacitância Vista Pelo *Driver*:** É necessário modelar a carga capacitiva a ser carregada pelo *driver* da interconexão com o objetivo de se obter a informação de *load*, a qual é utilizada no cálculo do *delay* e *slew* dos *timing arcs* das portas lógicas, como visto anteriormente. Nesta capacitância é incluído também o impacto causado pelos pinos de destino da interconexão [5]. Na

[5] Um pino de destino de uma interconexão é um pino que se liga na interconexão, que não é o pino *driver*. Por exemplo, na Figura 9(a), os pinos de destino da interconexão são o segundo pino de entrada da porta $u2$, o pino de entrada da porta $u3$ e o pino d do *flip-flop* $f1$.



(a)

```
1  *D_NET inp1 5.4
2  *CONN
3  *P inp1 I
4  *I u1:a I
5  *CAP
6  1 inp1 1.2
7  2 inp1:1 1.3
8  3 inp1:2 1.4
9  4 u1:a 1.5
10 *RES
11 1 inp1 inp1:1 3.4
12 2 inp1:1 inp1:2 3.5
13 3 inp1:2 u1:a 3.6
14 *END
```

(b)

Figura 8 – (a) Interconexão RC obtida do circuito *simple* da competição de *sizing* do ISPD. (b) SPEF referente à Figura 8(a).

fase *pré-layout*, essa estimativa é realizada somando a capacitância total da interconexão com a capacitância de cada pino de destino dela. Porém, ao se tratar de interconexões com característica resistiva, o uso da abordagem de capacitância concentrada é impreciso. Para que os modelos de atraso não-lineares, que dependem do valor de capacitância de saída, sejam utilizados para os *drivers* diretamente, é necessário o uso de uma abordagem conhecida como **Capacitância Efetiva (C_{eff})**. Tal abordagem tenta encontrar um valor de capacitância que pode ser utilizado como carga equivalente, em termos de *timing*, para a saída do

driver (BHASKER; CHADHA, 2009). Algumas técnicas serão abordadas no Capítulo 3.

- **Atraso da Interconexão:** Além do impacto local nos *delays* e *slews* de seus *drivers*, as interconexões exercem impacto global no circuito, com seu próprio atraso de propagação (Figura 9(b)), devido a sua característica resistiva. Com a alta frequência de operação dos circuitos digitais atuais e o dimensionamento dos transistores para escalas nanométricas, os atrasos das interconexões, que antes não eram significativos, hoje chegam consumir de 50% a 70% do ciclo do relógio, e esta porcentagem tende a aumentar na medida que os transistores diminuem (CONG et al., 1996). Uma das métricas mais populares para se calcular o atraso em interconexões é o atraso de Elmore (*Elmore Delay*) (ELMORE, 1948), pela simplicidade e razoável correlação com os atrasos reais. Esta técnica será apresentada com mais detalhes na Seção 3.2.

- **Degradação do *Slew*:** O cálculo do *slew* é crucial para determinar a precisão de uma avaliação de *timing* em um circuito digital (ZHOU et al., 2007). Os *delays* dos *timing arcs* dependem do *slew* de entrada e do *slew* de saída. Quando um sinal se propaga por uma interconexão, seu *slew* (i.e., sua declividade) sofre uma degradação devido ao efeito resistivo da mesma (Figura 9(b)). A não-modelagem desta degradação pela interconexão, acarreta em erros de até 50% (SHEEHAN, 2002). A abordagem para degradação do *slew* utilizada neste trabalho será apresentada na Seção 3.3.2.

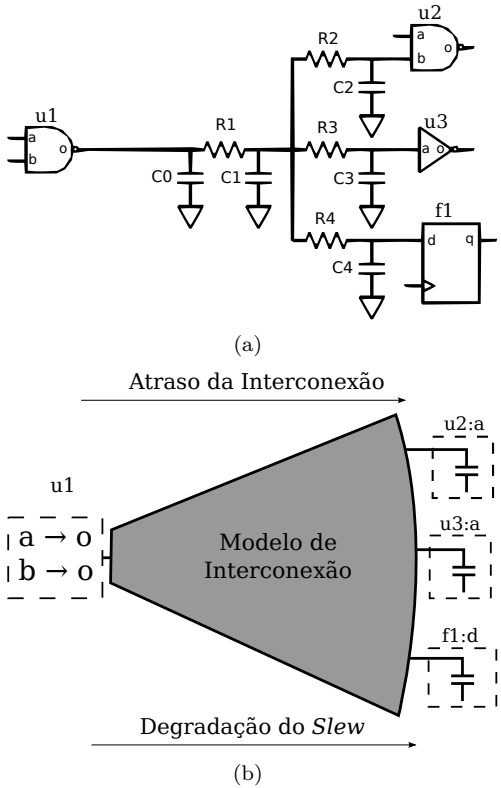


Figura 9 – (a) Um circuito composto por três portas lógicas ($u1$, $u2$ e $u3$), uma célula sequencial ($f1$) e uma interconexão em forma de árvore RC, que liga a saída de $u1$ às entradas de $u2$, $u3$ e $f1$; (b) São apresentadas as modelagens para os *timing arcs* da porta lógica $u1$; O modelo da interconexão é abstraído, recebendo um valor de capacitância efetiva. As setas indicam que a interconexão oferece um atraso e uma degradação no *slew*. Cada destino da interconexão é representado como um valor de capacitância de seus pinos de entrada.

3 CÁLCULO DAS CARACTERÍSTICAS TEMPORAIS DA INTERCONEXÃO

Este capítulo tem por objetivo apresentar uma técnica utilizada para o cálculo das características temporais das interconexões, necessário para a estimativa de *timing* global dos circuitos digitais. Na Seção 3.1 será apresentado um modelo computacional para as interconexões. A Seção 3.2 apresentará uma revisão bibliográfica mostrando algumas técnicas para cálculo do atraso das interconexões, bem como a técnica de Elmore e a técnica escolhida para ser implementada no presente trabalho, que será detalhada na Seção 3.3.

3.1 REPRESENTAÇÃO DAS INTERCONEXÕES

Para que o atraso de uma interconexão seja estimado com precisão, um modelo de grafo (Figura 10) pode ser utilizado para representar o fio em termos de capacitâncias e resistências.

No modelo de grafo $I(C, R)$ utilizado, o conjunto dos vértices é composto pelos nodos internos da interconexão, que representam cada capacitor. As arestas do grafo modelam os resistores, e cada resistor conecta um par de capacitores. Sendo assim:

- $\mathbf{C} = \{c | c$ é um capacitor da rede RC$\}$
- $\mathbf{R} = \{(c, d) |$ existe um resistor que conecta os capacitores c e $d\}$

3.2 CÁLCULO DO ATRASO DAS INTERCONEXÕES

Diversas técnicas são empregadas no cálculo do atraso das interconexões. Uma vez que o cálculo real dos atrasos das interconexões possui custo muito elevado para ser realizado para milhares de interconexões em centenas de vezes, modelos aproximados geralmente são utilizados no fluxo *standard cell*. A avaliação assintótica da forma de onda (*AWE: Asymptotic Waveform Evaluation*) (PILLAGE; ROHRER, 1990) é uma técnica amplamente utilizada para geração de modelos de ordem reduzida, realizando uma aproximação na função de transferência via aproximação de Padé. Outras técnicas como PRIMA (Passive Reduced-Order Interconnect Macromodeling Algorithm) (ODABASIOGLU; CELIK; PILEGGI, 1997) possuem uma alta complexidade

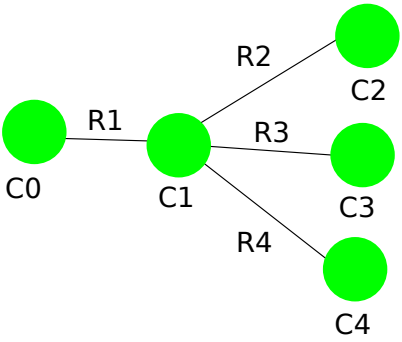


Figura 10 – O grafo correspondente à interconexão da Figura 9(a), com cinco vértices e quatro arestas.

computacional, sendo muito lentas para serem utilizadas no contexto de uma técnica de otimização, como *gate sizing* (KAHNG et al., 2013).

A técnica de Elmore (1948) é uma técnica baseada no primeiro momento da resposta ao impulso amplamente utilizada no cálculo dos atrasos das interconexões. A popularidade da técnica de Elmore deve-se aos fatores que seguem:

- Boa correlação com os atrasos reais nos nodos mais afastados do *driver* (KASHYAP; ALPERT; DEVGAN, 2000);
- Utiliza uma fórmula fechada, que envolve apenas as resistências e capacitâncias do circuito (HOROWITZ, 1983);
- Provê um limite superior provado para o atraso real de qualquer árvore RC (GUPTA et al., 1997);
- É aditiva, ou seja, o atraso do nodo A até o nodo C passando pelo nodo B é a soma dos atrasos entre A e B e entre B e C (KASHYAP; ALPERT; DEVGAN, 2000).

De acordo com Rabaey, Chandrakasan e Nikolic (2008), em um nodo c_i da árvore RC, o atraso de Elmore (τ_i) pode ser facilmente calculado como:

$$\tau_i = \sum_{k=1}^{N} C_k R_{ik} \tag{3.1}$$

Onde N é o número de capacitores da árvore RC, C_k é o valor de capacitância do nodo c_k e R_{ik} é a resistência compartilhada entre os caminhos $s \rightarrow i$ e $s \rightarrow k$ (s é o nodo fonte), ou seja:

$$R_{ik} = \sum R_j \Rightarrow (R_j \in [caminhos(s \rightarrow i) \cap caminhos(s \rightarrow k)]) \quad (3.2)$$

Na topologia da Figura 9(a), $C0$ é o nodo fonte. Assim, o atraso de Elmore para o nodo $C4$ é:

$$\tau_4 = C_1\ R_1 + C_2\ R_1 + C_3\ R_1 + C_4(R_1 + R_4) \quad (3.3)$$

A técnica de Elmore pode ser implementada também em sua forma recursiva. O algoritmo para cálculo do atraso de Elmore recebe como entrada o grafo $I(C, R)$ da interconexão e é executado após a inicialização das capacitâncias totais *downstream* (C_{total_i}) de cada nodo interno.

1. **Inicialização das capacitâncias totais *downstream*:** Os nodos internos são numerados de 1 até n em ordem topológica, sendo n o tamanho do conjunto de vértices. Assim, o passo de inicialização de cada $c_i \in C$ acontece em ordem topológica reversa, seguindo a Equação 3.4;

$$C_{total_i} = C_i + \sum_{j \in filhos(i)} C_{total_j} \quad (3.4)$$

   Sendo que C_i é o valor de capacitância do nodo c_i. O conjunto $filhos(i)$ é o conjunto de capacitores que estão interligados diretamente com o capacitor c_i através de um resistor R, que tenham um nível topológico maior que este[1]. Analogamente, o $pai(i)$ é um capacitor que precede c_i[2] e se conecta com ele, também, através de um resistor.

2. **Cálculo dos atrasos utilizando a técnica de Elmore:** O atraso de Elmore em cada nodo c_i da interconexão é calculado recursivamente, somando o atraso no pai de c_i com o valor da resistência que liga c_i ao seu pai multiplicado pela capacitância total *downstream* de c_i, como mostrado na Equação 3.5.

[1] Caso c_i seja um nodo terminal, seu conjunto $filhos(i)$ é vazio.
[2] Se c_i não for o nodo fonte da árvore.

$$\tau_i = \tau_{pai(i)} + R(pai(i), i) \times C_{total_i} \tag{3.5}$$

A função $R(i, j)$ retorna o valor da resistência que liga dois capacitores c_i e $c_j \in C$. O atributo C_{total_i} é a capacitância total *downstream* de um nodo c_i. Assim, o cálculo do atraso em cada nodo da interconexão compõe o atraso da interconexão partindo do *driver* até cada pino de destino.

A técnica de Elmore para atraso de interconexões fornece boas aproximações quando o efeito conhecido como *resistive shielding* não é tão alto. Este efeito acontece devido ao fato de que as resistências alteram o tempo que as capacitâncias levam para serem carregadas ou descarregadas. O efeito de *resistive shielding* faz com que o atraso do *driver* de uma interconexão seja menor que o atraso dele considerando a capacitância concentrada da interconexão. Similarmente, o efeito faz com que o atraso da interconexão seja menor que o atraso de Elmore utilizando o valor de capacitância total para cada segmento. Considere a interconexão da Figura 9(a), no caso extremo em que $R4 = \infty$, o capacitor $C4$ nunca seria carregado, e portanto, o atraso da interconexão não deveria levar em consideração o valor do capacitor $C4$.

Algumas adaptações na técnica de Elmore foram propostas para que o cálculo do atraso das interconexões capturem também o efeito do *resistive shielding* utilizando a abordagem da capacitância efetiva em cada nodo da interconexão. Na Figura 11, as transições na saída de um *driver* são comparadas ao se utilizar a abordagem de capacitância efetiva (linha pontilhada) e de capacitância concentrada (linha contínua). Pode-se observar que para o valor escolhido de capacitância efetiva, o sinal leva o mesmo tempo para atingir o ponto médio da curva ($V_{dd} = 50\%$) que quando o *driver* está conectado diretamente à carga real da árvore RC (linha tracejada). Note a diferença neste ponto em relação à curva de capacitância concentrada, mostrando a imprecisão de se utilizar este modelo em certos casos.

Como a abordagem de capacitância efetiva está relacionada à consideração do efeito de *resistive shielding*, ao utilizá-la em cada segmento da interconexão, é possível obter-se um atraso na interconexão mais preciso do que o atraso de Elmore, mesmo sem considerar momentos de maior ordem da resposta ao impulso.

Kashyap, Alpert e Devgan (2000) propuseram uma técnica para calcular o atraso da interconexão levando em conta o efeito de *resistive shielding*. Com a mesma complexidade da técnica de Elmore, a

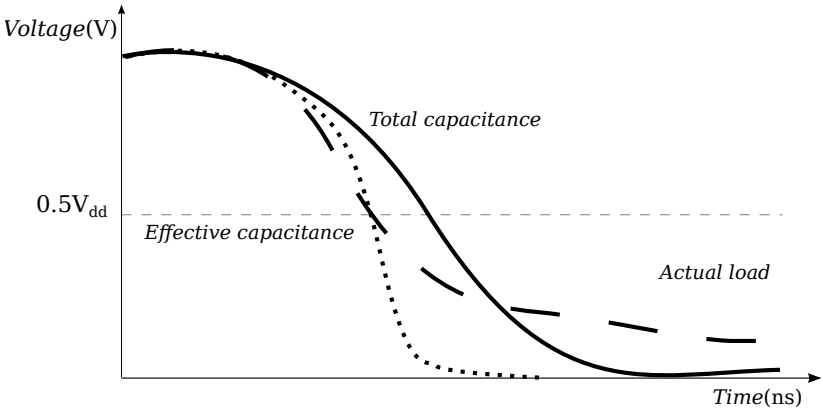


Figura 11 – Formas de onda na saída de uma porta lógica em função da abordagem utilizada para cálculo da capacitância. Obtida de (BHASKER; CHADHA, 2009).

técnica proposta para cálculo de atraso em uma árvore RC calcula também o valor de capacitância efetiva. Porém, Kashyap, Alpert e Devgan (2000) não consideravam o *driver* da interconexão como sendo uma porta lógica *CMOS*. Como consequência, sua aproximação para o *slew* na entrada da árvore RC era imprecisa. Como o cálculo da capacitância efetiva de uma árvore depende do *slew* que incide nesta, e o *slew* depende da capacitância vista pelo *driver*, Puri, Kung e Drumm (2002) propuseram uma técnica que leva em consideração o impacto da capacitância no *slew* do driver, e também, do *slew* no cálculo da capacitância efetiva. Esta técnica será apresentada na Seção 3.3 e foi a técnica implementada neste trabalho.

3.3 TÉCNICA DE Puri, Kung e Drumm (2002) PARA O CÁLCULO DA CAPACITÂNCIA EFETIVA E DEGRADAÇÃO DO *SLEW*

O objetivo desta seção é apresentar a técnica para cálculo das informações referentes às características temporais das interconexões que foi escolhida para ser implementada neste trabalho.

Devido ao fato de que o valor de capacitância efetiva de uma interconexão depende do *slew* incidente nesta, que por sua vez, depende do valor de capacitância efetiva, Puri, Kung e Drumm (2002)

propuseram uma técnica iterativa para simular esta interdependência. A técnica em questão obtém o atraso de Elmore com capacitância efetiva para a interconexão, bem como a degradação do *slew* e o valor de capacitância utilizado no cálculo do *delay* e *slew* do *driver*.

A seguir serão apresentadas as técnicas para cálculo da capacitância efetiva e degradação do *slew*, bem como o algoritmo implementado para realização desses cálculos.

3.3.1 Cálculo da Capacitância Efetiva

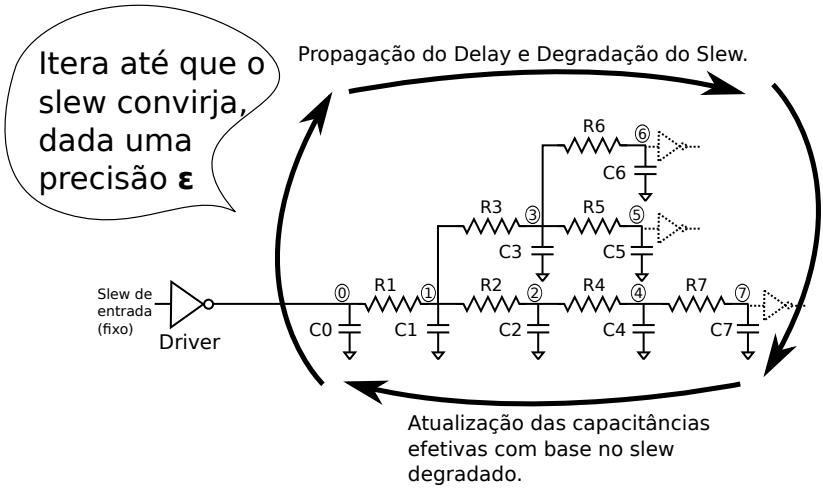


Figura 12 – Visão geral da técnica iterativa para o cálculo do atraso da interconexão, capacitância efetiva e degradação do *slew*. Adaptada de (PURI; KUNG; DRUMM, 2002).

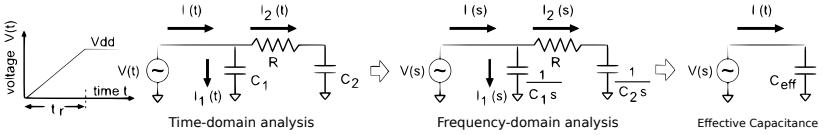


Figura 13 – Cálculo da capacitância efetiva utilizando rampa de entrada. Obtida de (PURI; KUNG; DRUMM, 2002).

Considere uma rede π $C_1 - R - C_2$ alimentada por uma fonte de tensão $V(t)$, como a ilustrada na Figura 13. Seja $I(t)$ a corrente total fornecida pela fonte de tensão $V(t)$, $I_1(t)$ a corrente através de C_1 e $I_2(t)$ a corrente através de $R - C_2$. De acordo com Puri, Kung e Drumm (2002), realizando uma análise no domínio da frequência, obtemos que:

$$I(s) = I_1(s) + I_2(s) \tag{3.6}$$

Se

$$I_1(s) = \frac{V(s)}{1/C_1 s} \tag{3.7}$$

$$I_2(s) = \frac{V(s)}{R + 1/C_2 s} \tag{3.8}$$

Então:

$$I(s) = \frac{V(s)}{1/C_1 s} + \frac{V(s)}{R + 1/C_2 s} \tag{3.9}$$

ou

$$I(s) = V(s)\left(C_1 s + \frac{C_2 s}{1 + RC_2 s}\right) \tag{3.10}$$

Agora, considerando que a fonte de tensão $V(t)$ é uma rampa com tempo de subida t_r, $V(t)$ é dado por:

$$V(t) = \begin{cases} \frac{V_{dd}}{t_r} \times t & \text{se } t < t_r \\ V_{dd} & \text{caso contrário} \end{cases} \tag{3.11}$$

E no domínio da frequência:

$$V(s) = \frac{V_{dd}}{t_r} \times \frac{1}{s^2} \times (1 - e^{-st_r}) \tag{3.12}$$

Substituindo a $V(s)$ da Equação 3.10 pela Equação 3.12 e voltando ao domínio do tempo obtém-se:

$$I(t) = \frac{V_{dd}}{t_r}((C_1 + C_2) - C_2 e^{-\frac{t}{RC_2}}) \quad \text{para } t < t_r \tag{3.13}$$

Em termos de *timing*, na capacitância efetiva, a carga transferida Q é a mesma que da rede π, no ponto médio da curva (tempo que a curva atinge $50\%V_{dd}$). A carga transferida Q é a integral da corrente $I(t)$ com o tempo indo de 0 até $t_r/2$:

$$Q = \int_0^{t_r/2} I(t)\mathrm{d}t = \int_0^{t_r/2} \frac{V_{dd}}{t_r}((C_1+C_2) - C_2 e^{-\frac{t}{RC_2}})\mathrm{d}t \tag{3.14}$$

A carga transferida Q para carregar a capacitância efetiva da rede π (C_{eff}) até 50% de V_{dd} é dada, também, por $\frac{C_{eff}V_{dd}}{2}$. Ao igualar as duas equações de transferência de carga, obtemos:

$$C_{eff} = C_1 + C_2(1 - \frac{2RC_2V_{dd}}{t_r}(1 - e^{-\frac{t_r}{2RC_2}})) \tag{3.15}$$

Assim, $C_{eff} = C_1 + C_2 \times K$, onde K é o fator de *shielding*, definido por:

$$K = 1 - 2x(1 - e^{-\frac{1}{2x}}), \quad \text{onde } x = \frac{RC_2}{t_r} \tag{3.16}$$

3.3.2 Degradação do *Slew* Através da Árvore RC

O *slew* é degradado na árvore RC para obtenção de coeficientes mais precisos para os fatores de *shielding*. Considere um segmento de uma árvore RC, mostrado na Figura 14, que é alimentado por uma fonte de tensão $V(t) = \frac{V_{dd}}{t_r} \times t$, onde t_r é o *slew* da rampa de entrada. Conforme (PURI; KUNG; DRUMM, 2002), a tensão de saída neste caso, quando $t = t_r$, pode ser derivada como:

$$\frac{V_{dd}}{t_r}(t_r - RC_2 + RC_2 e^{-\frac{t_r}{RC_2}}) \tag{3.17}$$

Conforme a Figura 14:

$$\frac{V_1}{t_r} = \frac{V_{dd}}{t'_r} \tag{3.18}$$

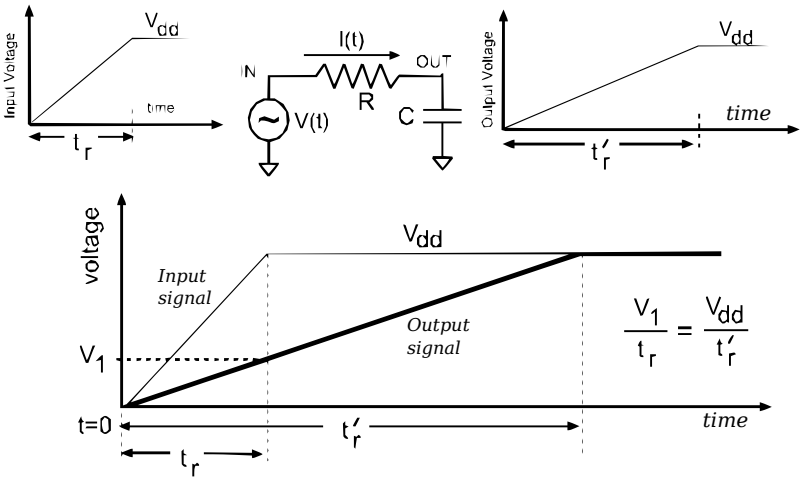


Figura 14 – Degradação no *slew* em um segmento de uma árvore RC. Obtida de (PURI; KUNG; DRUMM, 2002).

Onde t'_r é o *slew* na saída. Substituindo V_1 da Equação 3.18 pela Equação 3.17, obtemos o valor de *slew* na saída, em função do *slew* aplicado na entrada:

$$t'_r = \frac{t_r}{1 - x(1 - e^{-\frac{1}{x}})} \quad \text{onde } x = \frac{RC_2}{t_r} \tag{3.19}$$

Generalizando para uma árvore RC qualquer [3] e utilizando o valor de capacitância efetiva, o $slew_i$ no nodo c_i é definido pela equação:

$$slew_i = \frac{slew_j}{1 - \frac{R_i\ C_{eff_i}}{slew_j}(1 - e^{-\frac{slew_j}{R_i\ C_{eff_i}}})} \tag{3.20}$$

Onde $slew_j$ é o *slew* na entrada de v_i, vindo pelo nodo pai v_j e R_i é o valor do resistor que conecta c_i com c_j

[3] Representada conforme apresentado na Seção 3.1.

3.3.3 O Algoritmo de Puri, Kung e Drumm (2002)

Dado o grafo de uma árvore RC[4], o algoritmo apresentado nessa seção calcula os valores de capacitância efetiva e *slew* em cada nodo interno da interconexão. Para o atraso da interconexão, o método implementa a técnica de Elmore utilizando os valores de capacitância efetiva, ao invés dos valores de capacitância total *downstream*, simulando o efeito de *resistive shielding*.

A capacitância efetiva é denotada em cada nodo c_i por C_{eff_i} e o *slew* por $slew_i$. O valor do *slew* aplicado no nodo fonte da árvore RC, denotado por $slew_1$, é exatamente o valor do *slew* na saída do *driver* desta interconexão, o qual é função do *slew* na entrada da porta lógica *driver* e da capacitância efetiva vista na saída. Este valor de *slew* será refinado iterativamente para se estimar o valor de capacitância efetiva da interconexão.

O algoritmo para cálculo iterativo da capacitância efetiva de uma interconexão, bem como seu atraso e a degradação no *slew*, conforme (PURI; KUNG; DRUMM, 2002), ocorre em cinco passos:

1. **Inicialização:**
   (a) A capacitância efetiva C_{eff_i} de cada nodo c_i da Árvore RC é incializada com o valor de capacitância total *downstream* de c_i, ou seja $C_{eff_i} = C_{total_i}$;
   (b) O *slew* no nodo fonte da árvore RC $slew_1$ é calculado utilizando o modelo de atraso da porta lógica *driver*, considerando a capacitância concentrada da árvore RC (i.e., $\sum_{i=1}^{N} C_i$): $slew_1 = f(C_{total_1})$.

2. **Atualização dos *slews* em ordem topológica:**
   (a) Atraso τ_i do nodo fonte c_1 para cada nodo c_i da árvore é calculado utilizando a técnica de Elmore (Equação 3.5), substituindo C_{eff_i} por C_{total_i}, para simular o efeito de *resistive shielding*;
   (b) A degradação do *slew* em cada nodo c_i é calculada utilizando a Equação 3.20.

3. **Atualização das capacitâncias efetivas em ordem topológica reversa:**

[4] Com os nodos numerados de 1 a n em ordem topológica, onde n é o tamanho do conjunto de vértices e o nodo c_1 é o nodo fonte da árvore RC.

(a) A capacitância efetiva (C_{eff_i}) de cada nodo c_i é calculada como a soma da capacitância do nodo c_i e todas as capacitâncias dos nodos filhos:

$$C_{eff_i} = C_i + \sum_{j \in filhos(i)} K_j \times C_{tot_j} \tag{3.21}$$

Onde K_j é o fator de *shielding*, definido por:

$$K_j = 1 - \frac{2R_j C_{eff_j}}{slew_i}(1 - e^{-\frac{slew_i}{2R_j C_{eff_j}}}) \tag{3.22}$$

Onde R_j é o valor da resistência que conecta o nodo c_j ao seu pai, no caso, c_i.

4. **Atualização do *Slew* do *Driver*:** O *slew* no nodo fonte $slew_1$ é calculado diretamente, utilizando o C_{eff_1} atual ;

5. **Iteração:** Os passos de 2 até 4 são repetidos até que $slew_1$ convirja, dada uma precisão ε .

Na implementação apresentada neste trabalho, o ε foi definido como sendo 1% e na maioria dos casos observados, cerca de 5 iterações são necessárias para realizar o cálculo da capacitância efetiva (PURI; KUNG; DRUMM, 2002). Como cada iteração do algoritmo percorre a lista em ordem topológica (direta e reversa), a complexidade assintótica de pior caso de cada iteração do algoritmo é de $O(n)$ onde n é o número de nodos da árvore, ao passo que a complexidade do algoritmo é $O(c.n)$, onde c é o número de iterações. Entretanto, como o número de iterações é na grande maioria dos casos menor que 5 (E portanto c é muito menor que n), assume-se que o crescimento no tempo de execução tem comportamento linear.

4 ANÁLISE DE *TIMING* ESTÁTICA

O objetivo deste capítulo é apresentar a análise de *timing* estática (*STA: Static Timing Analysis*), bem como os conceitos importantes referentes à esta técnica, juntamente com o algoritmo de STA.

Análise de *timing* estática, ou *static timing analysis* (GUNTZEL, 2000) (BHASKER; CHADHA, 2009), é uma das técnicas utilizadas para se estimar o atraso crítico de circuitos digitais. A análise de *timing* é chamada de estática quando ela não realiza simulação e portanto, independe de estímulos de entrada, considerando apenas a topologia do circuito. É um processo completo e exaustivo (BHASKER; CHADHA, 2009) que verifica as mais diversas informações de *timing* em um circuito, como os *delays*, *slews*, *slacks* (folgas), *required times* (tempos requeridos) e diversas violações de restrições de projeto.

Dada a descrição do projeto usando alguma linguagem de descrição de hardware (*HDL: Hardware Description Language*), restrições de projeto e uma biblioteca de células, o objetivo da análise de *timing* é apresentar informações temporais em todos os pontos do circuito e apontar as possíveis violações (Figura 15). Essas informações são utilizadas para avaliar se o projeto sob verificação pode operar na velocidade estipulada, ou seja, se o circuito final poderá funcionar com segurança na frequência de relógio escolhida, sem que existam violações nas restrições de projeto.

O fluxo básico de uma ferramenta de análise de *timing* é:

1. **Leitura dos arquivos de entrada:** Nesta etapa, os arquivos referentes às bibliotecas de célula, descrição do circuito juntamente com as restrições do projeto são lidos e suas informações são armazenadas em estruturas de dados, que serão consultadas na geração do modelo de grafo e na atualização das informações temporais;

2. **Geração do grafo de *timing*:** Responsável por implementar o modelo de grafo de *timing*. As estruturas de dados utilizadas na implementação do modelo de grafo têm impacto direto no desempenho da ferramenta de *timing*.

3. **Atualização de informações temporais:** Etapa onde a propagação dos atrasos através dos *timing arcs*, bem como a avaliação do cumprimento ou não das restrições de desempenho são realizados.

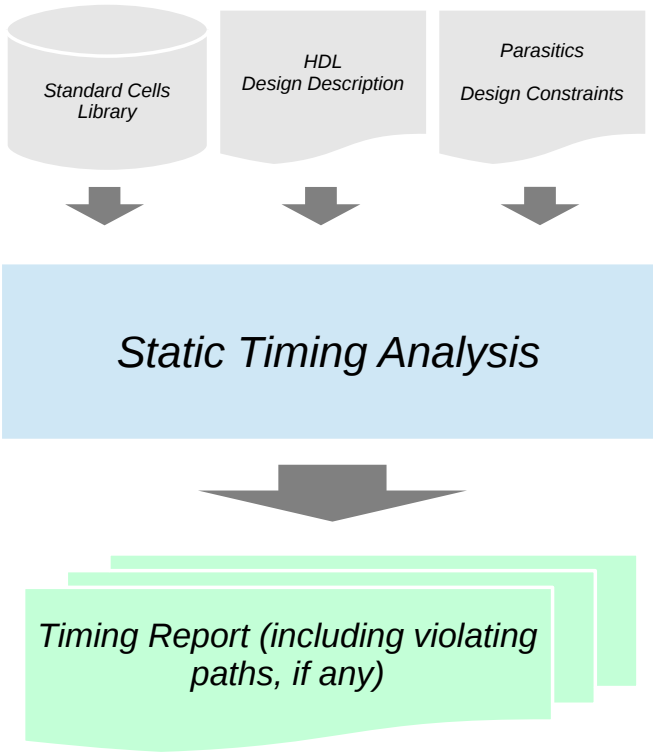


Figura 15 – Análise de *timing* estática. Adaptado de (BHASKER; CHADHA, 2009).

Na Seção 4.1 será apresentado o modelo de grafo utilizado para modelar os circuitos digitais na análise de *timing*. A Seção 4.2 mostrará a nomenclatura utilizada para as diversas informações temporais relevantes na análise de *timing* estática. Finalmente, as informações particulares sobre a implementação da ferramenta construída neste trabalho serão apresentadas na Seção 4.3.

4.1 REPRESENTAÇÃO DE CIRCUITOS DIGITAIS

Um circuito combinacional pode ser representado por um grafo de *timing*. Neste grafo, as portas lógicas e os pinos de entrada e saí-

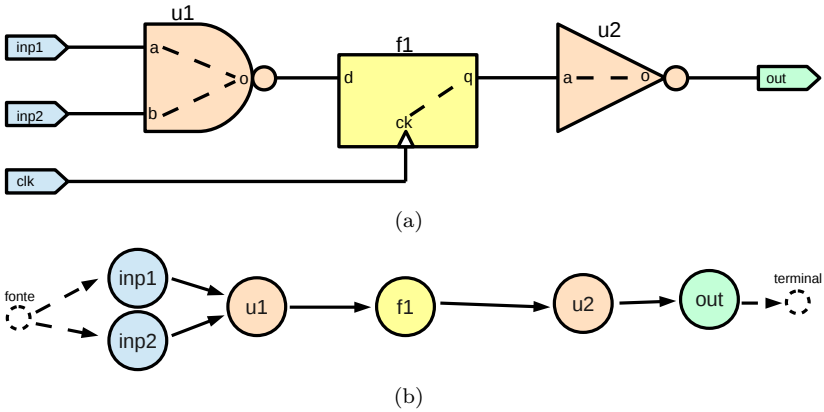


Figura 16 – (a) Circuito *simple* retirado do banco de *benchmarks* da competição de *sizing* do ISPD; (b) Grafo correspondente ao circuito da letra (a).

das primárias são os vértices e as interconexões são as arestas, como mostrado na Figura 16 (BHASKER; CHADHA, 2009).

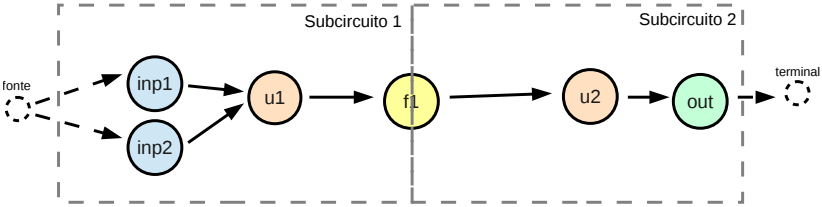


Figura 17 – Grafo de *timing* dividido em dois sub-circuitos devido à existência de uma célula sequencial.

Um circuito que consiste de células combinacionais e sequenciais (*flip-flops* e *latches*) pode ser representado como um conjunto de blocos combinacionais, divididos pelos *latches* (Figura 17). Assim, a entrada de uma célula sequencial pode ser tratada como uma saída primária do circuito, e a saída dessa pode ser tratada como uma entrada primária de outro circuito (Figura 18).

Em um contexto de projeto com o fluxo *standard cell*, é interessante que o grafo modele também os *timing arcs* das portas lógicas.

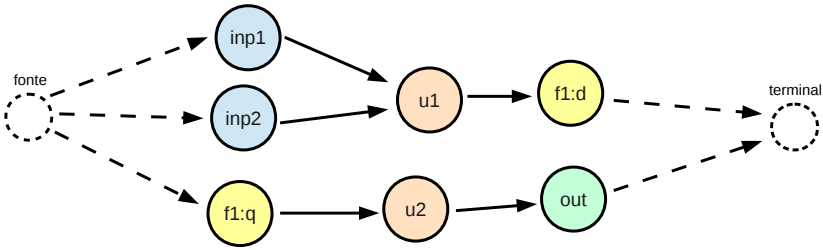


Figura 18 – Grafo de *timing* com célula sequencial atuando como entrada e saída primária do circuito.

Assim, alternativamente, os vértices do grafo de *timing* representam os pinos de entrada e saída das portas lógicas, entradas e saídas primárias e as arestas representam os *timing arcs* e as interconexões. Um grafo representando o modelo escolhido pode ser visualizado na Figura 19.

A nomenclatura usada no grafo direcionado $G(V, E)$ deste segundo modelo, adotado no presente trabalho, é a seguinte:

- V = { $v_i | v_i$ é um *timing point* (pino de *timing*), que pode ser a entrada ou saída de uma porta lógica, aqui referenciado como pino. Um *timing point* pode também representar uma entrada ou saída primária do circuito. }

- I = { $(v_i, v_j) | v_i, v_j \in V$ e (v_i, v_j) é uma interconexão do circuito, que conecta v_i em v_j. v_i é um pino de saída de uma porta lógica ou uma entrada primária, e v_j pode ser a entrada de uma porta lógica ou uma saída primária. }

- A = { $(v_i, v_j) | v_i, v_j \in V$ e (v_i, v_j) é um *timing arc*. Portanto, v_i e v_j são pinos de entrada e saída (respectivamente) de uma mesma porta lógica. }

- Por fim, o conjunto das arestas E = I $\cup$ A.

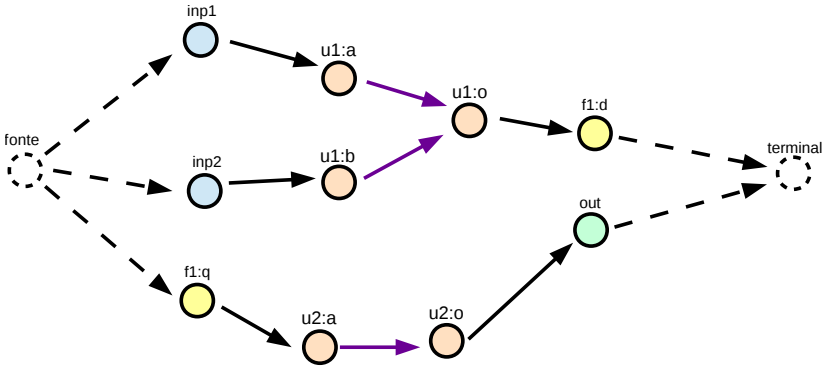


Figura 19 – Grafo de *timing* com representação dos *timing points*, *timing arcs* e interconexões.

Nos vértices, ou *timing points*, são armazenadas as informações temporais para os pinos do circuito, tais como os *arrival times*, *slews* e *slacks* que serão apresentadas na Seção 4.2.

4.2 CÁLCULO DO PIOR ATRASO DO CIRCUITO

O cálculo do pior atraso do circuito é realizado propagando os *arrival times* (tempos de chegada) das portas lógicas em ordem topológica através de um método conhecido como *PERT/CPM (Program Evaluation and Review Technique / Critical Path Method)*. Para o entendimento das políticas de propagação dos atrasos na avaliação do desempenho de um circuito, os termos a seguir são importantes:

- **Caminho:** uma sequência de vértices (*timing points*) tal que, para cada um de seus vértices há uma aresta (*timing arc* ou interconexão) para o próximo vértice da sequência. O primeiro *timing point* da sequência é uma entrada primária e o último é uma saída primária;
- $\boldsymbol{inputs(i)}$: conjunto de *timing points* que se ligam com v_i através de um *timing arc*. Todo $v_j \in input(i)$ é necessariamente um pino de entrada de uma porta lógica, e v_i é um pino de saída;
- $\boldsymbol{a_i}$: *arrival time*, ou tempo de chegada no pino v_i. O *arrival time* é definido pelo atraso do caminho parcial que inicia em uma entrada primária e termina em v_i;

- $\boldsymbol{slew_i}$: o *slew* no pino v_i;
- $\boldsymbol{d_{j \to i}}$: o *delay* do *timing arc* que vai do pino v_j até o pino v_i;
- $\boldsymbol{slew_{j \to i}}$: o *slew* do *timing arc* que vai do pino v_j até o pino v_i;
- $\boldsymbol{iD_{i \to k}}$: o atraso de propagação na interconexão que liga o pino v_i até o pino v_k. No modelo de capacitância concentrada, $iD_{i \to k} = 0$;
- $\boldsymbol{iS_{i \to k}}$: degradação do *slew* através da interconexão que liga v_i em v_k;
- $\boldsymbol{fanouts(i)}$: conjunto dos pinos que são destino da interconexão para qual v_i é *driver*;
- $\boldsymbol{r_i}$: é o *required time* no *timing point* v_i. O *required time* é o tempo máximo que o valor de a_i pode assumir para que a restrição de desempenho seja respeitada. Se v_i é uma saída primária do circuito, então $r_i = T$, onde $f = \frac{1}{T}$ é a frequência mínima de operação do circuito digital;
- $\boldsymbol{slack_i}$: folga de tempo no ponto v_i, ou seja, quanto o *arrival time* pode atrasar neste ponto, de modo que o período máximo continue sendo respeitado. Se em um determinado ponto do circuito o *slack* é negativo, então o caminho em questão está violando a restrição de atraso máximo do sistema.

Na análise de *timing* estática, os piores atrasos de cada porta lógica são propagados visitando-se o grafo direcionado em ordem topológica. Para cada $v_i \in V$ que são pinos de saída de portas lógicas, os *arrival times*, bem como os *slews* são determinados de modo a respeitar as seguintes restrições:

$$a_i = \max_{\forall v_j \in inputs(i)} (a_j + d_{j \to i}) \tag{4.1}$$

$$slew_i = \max_{\forall v_j \in inputs(i)} (slew_{j \to i}) \tag{4.2}$$

Se $v_i \in V$ é um pino de entrada de uma porta lógica e $v_j \in V$ é o *driver* da interconexão que conecta v_j em v_i, o *arrival time* e o *slew* em v_i são definidos por:

$$a_i = a_j + iD_{j \to i} \tag{4.3}$$

$$slew_i = slew_j + iS_{j \to i} \tag{4.4}$$

Após a propagação dos *arrival times* em todos os pinos, é necessário realizar a propagação dos tempos requeridos, o que é feito percorrendo-se o grafo em ordem topológica reversa, a fim de obterem-se os valores dos *slacks*. Em um pino de saída v_i de uma porta lógica, o tempo requerido pode ser obtido facilmente, observando o menor dos tempos requeridos dentre os seus *fanouts* e suas interconexões, ou seja:

$$r_i = \min_{\forall v_j \in fanouts(i)} (r_j - iD_{i \to j}) \tag{4.5}$$

Para se propagar o *required time* do pino v_j de saída de uma porta lógica para uma entrada v_i, utiliza-se o valor de *delay* do arco que liga v_i em v_j, o qual já foi calculado previamente:

$$r_i = r_j - d_{i \to j} \tag{4.6}$$

A partir dos *required times* e *arrival times*, podemos determinar os *slacks* nos diversos pontos do circuito, através da equação:

$$slack_i = r_i - a_i \tag{4.7}$$

Se em algum ponto v_i, $slack_i = 0$, então v_i se encontra em um caminho crítico. Se $slack_i < 0$, então v_i se encontra em um caminho que viola a restrição de desempenho.

O **pior slack** é definido como o menor valor de *slack* entre as saídas primárias. O valor **total de slack negativo** é o somatório dos módulos dos *slacks* negativos das saídas primárias.

4.3 IMPLEMENTAÇÃO DA FERRAMENTA DE STA

Esta seção apresentará as estratégias utilizadas para o desenvolvimento da ferramenta de análise de *timing*. Serão ilustradas as principais estruturas de dados, modelos de grafo, e serão apresentados também os algoritmos implementados na ferramenta desenvolvida.

4.3.1 O Modelo de Grafo Adotado

As estruturas de dados utilizadas para armazenar os elementos do grafo são essencialmente listas ordenadas topologicamente. Em uma lista ordenada topologicamente, dado um elemento i, à esquerda necessariamente se encontram os elementos de mesmo ou menor nível lógico, e à direita, de nível igual ou maior, como mostrado na Figura 20. Da mesma maneira, os *timing arcs* e as interconexões também são ordenados topologicamente, em suas respectivas listas. Com essa escolha, o algoritmo de análise de *timing* estática passa a ser apenas de uma varredura em ordem, na lista de *timing points*, atualizando a informação de *timing* acumulada para cada vértice do grafo.

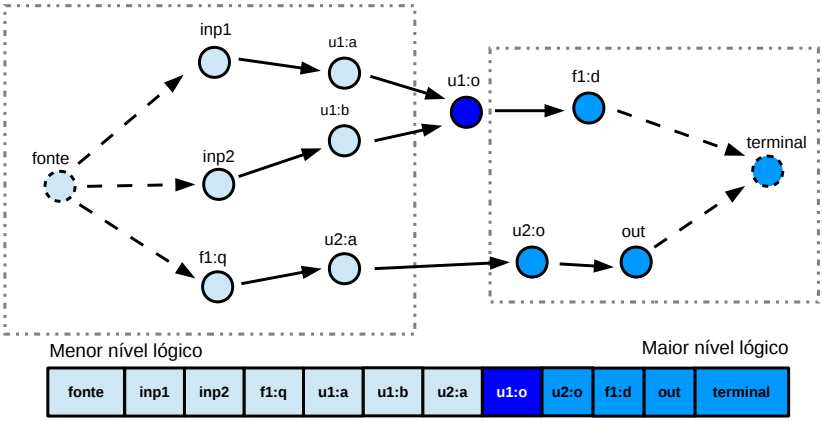


Figura 20 – Na lista ordenada, observando o elemento *u1:o*, os elementos de menor ou de igual nível lógico (*fonte, inp1, inp2, f1:q, u1:a, u1:b, u2:a*) se encontram à esquerda, e os de maior ou igual (*u2:o, f1:d, out, terminal*) se encontram à direita.

4.3.2 Algoritmo de Análise de *Timing* Estática

Com o modelo de grafo definido e implementado em suas devidas estruturas de dados, a análise de *timing* estática é realizada atualizando as informações de *timing* de cada nodo do grafo de *timing*, em ordem topológica, como mostrado no algoritmo 1.

No algoritmo 1 é apresentada a rotina de análise de *timing*. Os

Algoritmo 1: Análise de *timing* estática.

Entrada: Grafo de *timing* $G(V, E)$
Saída: Informações de *timing* para os elementos do grafo (*timing points*, interconexões e *timing arcs*)

```
1  para todo v_i ∈ V em ordem topológica hacer
2      se v_i é um pino de entrada então
3          C_eff ← calcular_C_eff();
4          d_{i→o} ← delay_biblioteca(s_i, C_eff);
5          slew_{i→o} ← slew_biblioteca(slew_i, C_eff);
6          propagar_atrasos();
7      senão se v_i é um pino de saída ou é uma entrada
       primária então
8          propagar_para_fanouts();
9      fim
10 fin
11 para todo cada v_i ∈ V em ordem topológica reversa
   hacer
12     atualiza_folgas(v_i);
13 fin
```

vértices são processados topologicamente, propagando os atrasos calculados de cada *timing arc* das portas lógicas para suas saídas.

Para os pinos de entrada das portas lógicas, o procedimento realizado compreende da Linha 3 até a Linha 6. Na Linha 3 é realizado o cálculo da capacitância efetiva, que posteriormente é utilizada para se obter os valores de *delay* e *slew* nos arcos que partem de v_i (Linhas 4 e 5). A seguir, na Linha 6, representada pela rotina *propagar_atrasos*(), os *arrival times* e *slews* são propagados para o pino de saída utilizando as Equações 4.1 e 4.2, respectivamente.

Já para os pinos de saída, a rotina *propagar_para_fanouts*() é executada, e corresponde à propagação dos *arrival times* e *slews* através das interconexões, utilizando as Equações 4.3 e 4.4.

Após todos os *arrival times* serem calculados, as folgas são obtidas propagando-se os *required times* em ordem topológica reversa, como foi apresentado na Seção 4.2 pelas Equações 4.5, 4.6 e 4.7, correspondendo ao procedimento *atualiza_folgas*().

5 EXPERIMENTOS

Esse capítulo tem por objetivo descrever os experimentos realizados neste trabalho e apresentar os resultados obtidos.

Na Seção 5.1 será apresentada a metodologia e infraestrutura utilizadas para a realização dos experimentos.

Na Seção 5.2 é apresentado o primeiro experimento, que trata da validação da ferramenta perante o *PrimeTime*, utilizando o modelo de interconexões de capacitância concentrada.

Na Seção 5.3 será apresentada a maneira como o *PrimeTime* modela o circuito, e como as informações de *timing* são calculadas.

E por fim, a Seção 5.4 tem por objetivo mostrar os resultados obtidos com a técnica de cálculo de capacitância efetiva e degradação do *slew* implementadas neste trabalho. A organização dos experimentos realizados a fim de validar a ferramenta pode ser observada na Tabela 1.

Driver	Interconexão		Tabela
	Atraso	Degradação do *Slew*	
C_{total}	**SEM**	**SEM**	2
C_{eff}	Elmore (C_{eff})	(PURI; KUNG; DRUMM, 2002)	4
C_{eff}	Elmore (C_{eff})	**SEM**	6
C_{total}	Elmore (C_{total})	(PURI; KUNG; DRUMM, 2002)	5

Tabela 1 – Técnicas validadas nos experimentos e as respectivas tabelas que apresentam os resultados obtidos.

5.1 METODOLOGIA E INFRAESTRUTURA EXPERIMENTAL

Para realizar a avaliação das técnicas abordadas neste trabalho, uma ferramenta para análise de *timing* na linguagem de programação C++ foi implementada. A ferramenta realiza a análise de *timing* e considera dois possíveis modelos de interconexão: o modelo da capacitância concentrada e o modelo RC concentrado.

Como parte dos experimentos é realizada comparando as informações calculadas pela ferramenta implementada com as informações reportadas pelo *PrimeTime*, o erro percentual (EP_t) e o erro médio percentual absoluto ($EMPA$) (Equações 5.1 e 5.2) foram adotados como métricas para estimar a qualidade das informações de *timing* reporta-

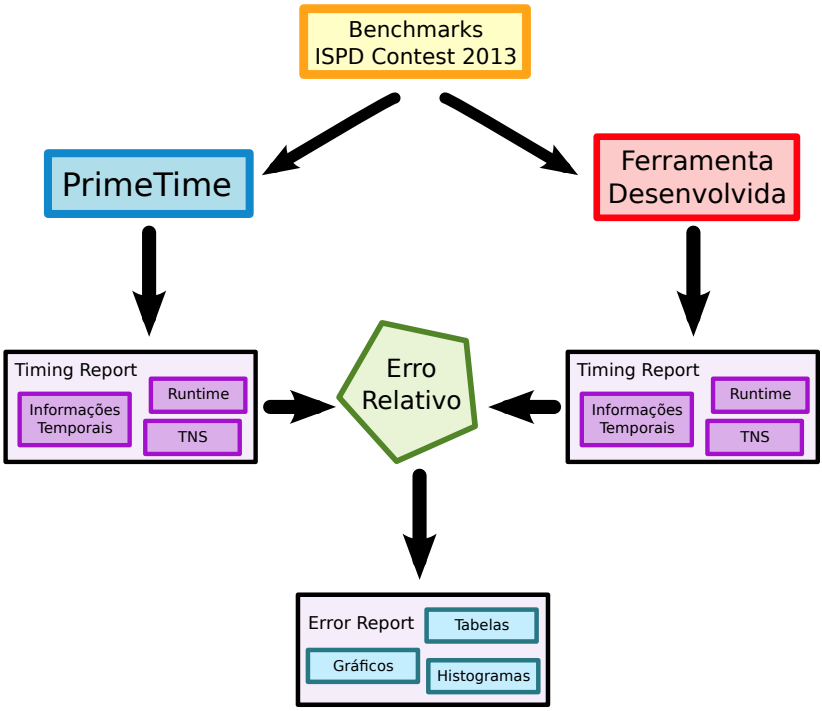


Figura 21 – Fluxo utilizado na validação da ferramenta implementada neste trabalho perante a ferramenta industrial *PrimeTime*.

das pela ferramenta implementada neste trabalho (Figura 21).

$$EP_t = \frac{(A_t - P_t)}{A_t} \times 100 \tag{5.1}$$

$$EMPA = \frac{\sum_{t=1}^{n} |EP_t|}{n} \tag{5.2}$$

O erro percentual é calculado para cada uma das informações comparadas com o *PrimeTime* utilizando a equação 5.1, sendo que A_t é a informação obtida pelo *PrimeTime* e P_t é a informação calculada pela ferramenta implementada. Tais informações usadas para fim de validação da ferramenta foram:

- ***TNS (Total Negative Slack)*:** O somatório de *slack* negativo nas saídas primárias.

- ***Violating POs:*** Número de saídas primárias violando a restrição de desempenho mínimo.

- ***Runtime (s):*** Tempo de execução, em segundos, para realizar uma análise de *timing* em um circuito, desconsiderando o tempo constante de inicialização da ferramenta.

- ***Critical Path:*** Valor do caminho crítico do circuito.

Os resultados dos experimentos serão apresentados posteriormente por meio de gráficos, tabelas e histogramas.

Este trabalho utilizou como base a infraestrutura disponibilizada pela competição de *gate sizing* discreto do ISPD de 2013, a qual fornece:

- Um conjunto de 8 circuitos da competição do ISPD de 2013:

  1. ***usb_phy***: com 511 células combinacionais, 98 células sequenciais, 15 entradas e 19 saídas primárias;
  2. ***pci_bridge32***: com 27316 células combinacionais, 3359 células sequenciais, 160 entradas e 201 saídas primárias;
  3. ***fft***: com 30297 células combinacionais, 1984 células sequenciais, 1026 entradas e 1026 saídas primárias;
  4. ***cordic***: com 40371 células combinacionais, 1230 células sequenciais, 34 entradas e 64 saídas primárias;
  5. ***des_perf***: com 103842 células combinacionais, 8802 células sequenciais, 234 entradas e 201 saídas primárias;
  6. ***edit_dist***: com 125000 células combinacionais, 5661 células sequenciais, 2562 entradas e 12 saídas primárias;
  7. ***matrix_mult***: com 30297 células combinacionais, 1984 células sequenciais, 3202 entradas e 1600 saídas primárias;
  8. ***netcard***: com 884427 células combinacionais, 97831 células sequenciais, 1836 entradas e 10 saídas primárias.

- Uma biblioteca *standard cell* realista, composta por onze células combinacionais de diversas funções lógicas e um célula sequencial;

- Uma ferramenta de análise de *timing* estática PrimeTime ® da empresa Synopsys (2012) para comparação de resultados;

Os circuitos são compostos por descrições no formato Verilog, capacitâncias parasitas e resistências descritas no formato IEEE SPEF (*Standard Parasitic Exchange Format*) (IEEE, 1999), e restrições de *timing* descritas no formato SDC (*Synopsys Design Constraints*).

5.2 VALIDAÇÃO DO MODELO DE CAPACITÂNCIA CONCENTRADA PERANTE FERRAMENTA INDUSTRIAL

Este experimento tem por objetivo validar a ferramenta de *STA* desenvolvida perante a ferramenta industrial *PrimeTime*, utilizando a abordagem de capacitância concentrada para modelar as interconexões. Para uma comparação justa, a ferramenta industrial foi também configurada para utilizar este modelo. Para tanto, utilizou-se um computador *desktop* com processador *Intel Core i7*, de 4 núcleos, e 4GB de *RAM*, e os resultados deste experimento são apresentados na Tabela 2.

Lumped Capacitance Interconnect Model

BENCHMARK	TNS (ps)	Viol. POs	Critical Path (ps)	Runtime (s)
usb_phy	0,00E+00	0	3,40E+02	0,00
pci_bridge32	2,08E+03	46	1,05E+03	0,02
fft	0,00E+00	0	1,51E+03	0,03
cordic	2,98E+04	185	3,16E+03	0,03
des_perf	0,00E+00	0	9,24E+02	0,09
edit_dist	0,00E+00	0	2,92E+03	0,11
matrix_mult	0,00E+00	0	2,04E+03	0,14
netcard	2,60E+06	11925	3,11E+03	24,83
Média	**3,29E+05**	**1519,5**	**1,88E+03**	**3,16**
EMPA	**0,00**	**0,00**	**0,00**	**-**

Tabela 2 – Comparação das informações de *timing* calculadas pela ferramenta implementada *versus* informações fornecidas pelo *PrimeTime*, utilizando o modelo de interconexões de capacitância concentrada.

As células marcadas correspondem aos valores que são menores que os obtidos na ferramenta comercial. A penúltima linha apresenta a média dos valores calculados para cada coluna da tabela. A última linha mostra o $EMPA$ para cada uma das informações mostradas nas colunas. Os valores de $EMPA$ valendo $0{,}00\%$ indicam que a ferramenta calcula os mesmos valores que a ferramenta industrial para as

informações comparadas. A média de *runtime* obtida é 6,92 vezes menor que a média da ferramenta industrial, sendo 50,05 vezes menor para os 7 primeiros circuitos (excluindo o *netcard*). No circuito *usb_phy*, a diferença de *runtime* é de 20 vezes e nos outros circuitos (exceto o *netcard*) a diferença tem valor médio de 52,71 vezes com baixo desvio padrão (6,48).

5.3 ANÁLISE DE *TIMING* ESTÁTICA EM FERRAMENTA INDUSTRIAL

Esta seção tem por objetivo apresentar o modelo (Figura 22) utilizado para cálculo do desempenho na ferramenta industrial *PrimeTime* e apresentar algumas informações de *timing* relevantes, obtidas para cada circuito da competição de *sizing* do ISPD.

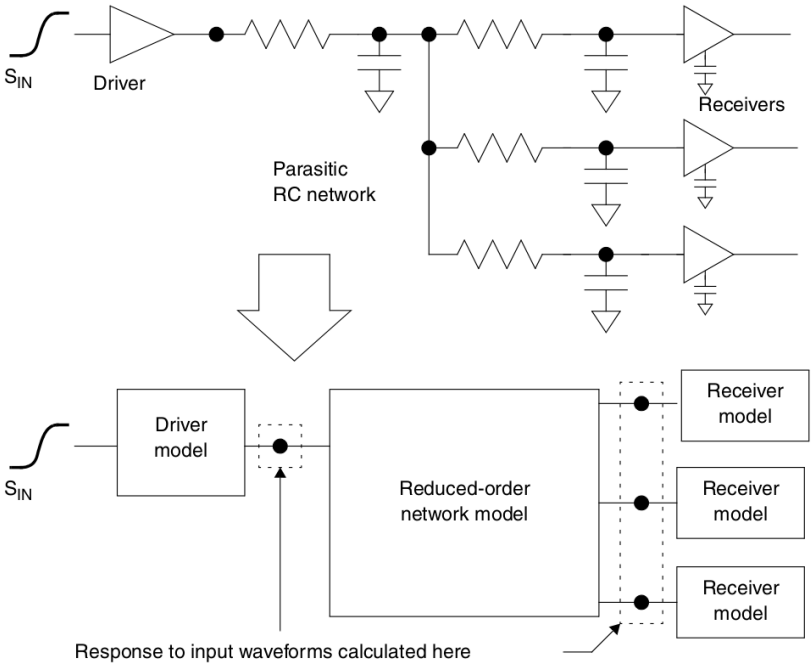


Figura 22 – Modelagem utilizada no *PrimeTime* para o *driver*, interconexão e destinos. Obtida de (SYNOPSYS, 2013).

5.3.1 Modelagem de *Driver* (*Driver model*)

No *PrimeTime*, o *driver* é modelado utilizando uma rampa de tensão em série com um resistor (um modelo Thèvenin). O resistor ajuda a suavizar a rampa de tensão, para que a forma de onda resultante seja similar à forma de onda do *driver* real, que está conectado na interconexão.

De acordo com (SYNOPSYS, 2013), o modelo de *driver* tem três parâmetros:

- A resistência do *driver* R_d;
- O tempo de início da rampa t_Z;
- A duração da rampa Δt.

O *PrimeTime* escolhe os parâmetros de modo que a forma de onda de saída seja o mais próximo possível da simulação. O modelo de *driver* simplificado é construído para cada *timing arc* de cada porta lógica do circuito.

5.3.2 Modelagem do Destino (*Receiver model*)

Cada destino da interconexão é representado como um valor de capacitância, que corresponde à capacitância do pino em que a interconexão está ligada.

5.3.3 Modelagem da Interconexão (*Reduced-order network model*)

Um modelo reduzido de interconexão é uma representação simplificada de uma interconexão, com as mesmas características de resposta da interconexão original, a qual pode ter centenas de capacitâncias e resistências. O *PrimeTime* utiliza a redução de Arnoldi (ODABASIOGLU; CELIK; PILEGGI, 1997) (SILVEIRA et al., 1999) para criar um modelo reduzido. Através do modelo reduzido, a ferramenta escolhe valores para a C_{eff}, de modo que o atraso do *driver* seja igual ao atraso do *driver* ligado à interconexão original. Por falta de fontes na bibliografia, não foi possível implementar este método de ajuste da C_{eff}, impossibilitando uma comparação mais justa. Desta forma,

não foi possível identificar o algoritmo para o cálculo do atraso das interconexões no *PrimeTime*.

5.3.4 Resultados Obtidos

PrimeTime

BENCHMARK	*TNS (ps)*	*Viol. POs*	*Critical Path (ps)*	*Runtime (s)*
usb_phy	4,85E+04	57	1,18E+03	0,24
pci_bridge32	9,97E+06	3034	6,25E+03	2,71
fft	2,46E+07	1983	1,33E+04	3,65
cordic	1,45E+07	1206	1,62E+04	4,60
des_perf	1,26E+07	1648	1,15E+04	11,26
edit_dist	3,20E+07	3416	1,13E+04	13,27
matrix_mult	1,62E+07	2852	1,24E+04	18,98
netcard	2,60E+06	11925	3,11E+03	271,97
Média	1,41E+07	3265,1	9,40E+03	40,84

Tabela 3 – Valores obtidos pelo *PrimeTime* no *benchmark* experimental utilizado neste trabalho.

A Tabela 3 mostra os valores obtidos pelo *PrimeTime* aplicando a STA na infraestrutura experimental. A última linha da tabela mostra as médias para cada informação obtida.

5.4 VALIDAÇÃO DA TÉCNICA IMPLEMENTADA PERANTE FERRAMENTA INDUSTRIAL

Esta seção tem por objetivo comparar a qualidade das informações de *timing* obtidas pela ferramenta de análise de *timing* implementada neste trabalho com as informações reportadas pelo *PrimeTime*. Utilizando as técnicas apresentadas na Seção 3.3 (i. e., capacitância efetiva, atraso de interconexões e degradação de *slew*), a análise de *timing* estática foi aplicada nos circuitos de teste e suas soluções foram comparadas com as fornecidas pela ferramenta industrial. Tal comparação foi realizada com base nas métricas apresentada na Seção 5.1. Os resultados deste experimento podem ser vistos na Tabela 4.

A penúltima linha apresenta a média dos valores de cada coluna

C_{eff} + Elmore (C_{eff}) + Slew Degradation

BENCHMARK	TNS (ps)	Viol. POs	Critical Path (ps)	Runtime (s)
usb_phy	4,91E+04	59	1,19E+03	0,01
pci_bridge32	9,80E+06	3002	6,29E+03	0,31
fft	2,34E+07	1983	1,22E+04	0,45
cordic	1,27E+07	1206	1,55E+04	0,58
des_perf	1,12E+07	1648	1,08E+04	1,08
edit_dist	2,95E+07	3311	1,11E+04	1,79
matrix_mult	1,38E+07	2831	1,11E+04	2,15
netcard	2,17E+06	8944	3,00E+03	12,82
Média	**1,28E+07**	**2873,00**	**8,89E+03**	**2,40**
EMPA	**8,81**	**4,17**	**4,48**	**-**

Tabela 4 – Valores obtidos pela ferramenta implementada neste trabalho nos circuitos da competição de *sizing* do ISPD.

e a última linha apresenta o $EMPA$ para cada uma das informações em relação ao *PrimeTime*, que foram apresentados na Tabela 3. Com esse experimento, conclui-se que a ferramenta desenvolvida neste trabalho fornece informações de *timing* próximas às reportadas pelo *PrimeTime*[1], com um tempo de execução $17,02$ vezes menor.

As células marcadas apresentam os valores que são otimistas em relação do *PrimeTime* (i. e., que são menores que os obtidos pelo *PrimeTime*), correspondendo a 29 dos 36 valores obtidos.

O erro de menor valor absoluto para o *TNS* é de $1,34\%$ e o de maior é $16,7\%$ nos circuitos *usb_phy* e *netcard*, respectivamente, sendo que no segundo, o erro reflete em uma aproximação otimista, e no primeiro, pessimista.

Na média, a análise de *timing* na ferramenta desenvolvida é otimista em relação ao *PrimeTime*, de acordo com o grande número de células marcadas. É possível observar também que os maiores erros são obtidos nos maiores circuitos e os menores erros, nos menores circuitos.

No experimento mostrado na Tabela 5, o modelo de capacitância concentrada foi utilizado para modelar a carga vista pelo *driver*. Para o atraso das interconexões, a técnica de Elmore com capacitância concentrada foi utilizada. Já para a degradação do *slew*, foi utilizada a técnica descrita no Capítulo 3. Como esperado, os modelos utiliza-

[1] $EMPA = 8,21$ e $4,48$ para *TNS* e *critical path*, respectivamente.

C_{total} + Elmore (C_{total}) + Slew Degradation				
BENCHMARK	**TNS (ps)**	**Viol. POs**	**Critical Path (ps)**	**Runtime (s)**
usb_phy	6,25E+04	61	1,34E+03	0,00
pci_bridge32	1,16E+07	3070	6,57E+03	0,09
fft	2,79E+07	1983	1,38E+04	0,12
cordic	1,54E+07	1207	1,72E+04	0,14
des_perf	1,25E+07	1648	1,13E+04	0,35
edit_dist	3,50E+07	3508	1,24E+04	0,44
matrix_mult	1,62E+07	2851	1,19E+04	0,56
netcard	1,07E+07	36934	3,37E+03	6,55
Média	**1,62E+07**	**6407,75**	**9,72E+03**	**1,03**
EMPA	**48,32**	**27,59**	**6,48**	**-**

Tabela 5 – Experimentos utilizando o modelo capacitância concentrada para carga de saída dos *drivers*, técnica de Elmore para computar os atrasos das interconexões, e degradação do *slew* conforme apresentado no Capítulo 3.

dos neste experimento refletem em uma aproximação pessimista para o atraso do circuito[2], já que a técnica de Elmore pura[3] foi aplicada no cálculo dos atrasos das interconexões. A técnica obteve $0,13\%$ e $311,79\%$ de erro para TNS nos circuitos *matrix_mult* e *netcard*, respectivamente. Já para *critical path*, os erros obtidos vão de $1,94\%$ até $13,85\%$, nos circuitos *des_perf* e *usb_phy*, respectivamente.

A importância do cálculo da degradação do *slew* pode ser visualizado na Tabela 6. Os erros obtidos neste experimento ($40,80\%$ para TNS e $21,21\%$ para critical path) mostram resultados muito otimistas em relação ao *PrimeTime*, quando a técnica apresentada no Capítulo 3 é aplicada, sem considerar a degradação do *slew* nos destinos das interconexões.

5.4.1 Relação entre C_{eff} e C_{total}

Este experimento tem por objetivo justificar os baixos erros obtidos pelo experimento apresentado na Tabela 5. Foram obtidas as

[2]Informação obtida do baixo número de células marcadas (total de 4, com exceção das células de *runtime*).

[3]Sem utilizar a abordagem de capacitância efetiva.

C_{eff} + Elmore (C_{eff}) + No Slew Degradation

BENCHMARK	TNS (ps)	Viol. POs	Critical Path (ps)	Runtime (s)
usb_phy	3,14E+04	57	9,51E+02	0,00
pci_bridge32	7,46E+06	2847	5,48E+03	0,29
fft	1,88E+07	1983	1,03E+04	0,40
cordic	9,32E+06	1204	1,27E+01	0,48
des_perf	8,67E+06	1648	9,22E+03	0,96
edit_dist	2,02E+07	3139	9,11E+03	1,54
matrix_mult	9,07E+06	2639	9,28E+03	1,89
netcard	1,41E+05	1033	2,63E+03	12,81
Média	**9,21E+06**	**1818,75**	**5,87E+03**	**0,79**
EMPA	**40,80**	**14,16**	**29,21**	**-**

Tabela 6 – Experimentos utilizando o modelo capacitância efetiva para carga de saída dos *drivers*, técnica de Elmore utilizando as capacitâcias efetivas de cada nodo interno das interconexões, para computar seus atrasos. Neste experimento, a degradação do *slew* não foi considerada.

relações C_{eff}/C_{total} médias para todos os circuitos, os considerando em três configurações diferentes:

- **Min.:** configuração de menor consumo de *leakage*[4];
- **Normal:** configuração padrão, conforme descrita em *verilog*;
- **Max.:** configuração de maior consumo de *leakage*;

Na Tabela 7, observa-se que nos circuitos testados, o valor de C_{eff} médio fica muito próximo do valor de C_{total} na configuração de maior consumo de *leakage*. Nessa configuração, as portas lógicas possuem maiores *slews* em relação às configurações de menor *leakage* e *leakage* padrão. Por exemplo, no circuito *pci_bridge*, a mudança da configuração *Min.* para *Max.* aumenta em 43% o *slew* médio do circuito, e no *matrix_mult*, 93%. Como $C_{eff} = C_1 + C_2 \times K$, sendo K definido na Equação 3.16, o crescimento no valor do *slew* implica no crescimento do valor K, fazendo com que C_{eff} fique mais próxima de $C_1 + C_2$, que por sua vez, é o valor C_{total}. Este experimento explica o

[4] *Leakage* se trata de uma parcela da potência estática que é dissipada pelos transistores *CMOS* mesmo quando estão desligados.

C_{eff} / C_{total}

BENCHMARK	Size		
	Min.	Normal	Max.
usb_phy	0,934	0,934	0,996
pci_bridge32	0,949	0,949	0,999
fft	0,945	0,946	0,995
cordic	0,940	0,940	0,999
des_perf	0,966	0,966	0,999
edit_dist	0,925	0,927	0,996
matrix_mult	0,938	0,939	0,996
Média	**0,942**	**0,943**	**0,997**

Tabela 7 – Relação C_{eff}/C_{total} média por circuito.

baixo erro obtido ao utilizar-se o modelo de capacitância concentrada para estimar o valor da capacitância vista pelo *driver* (Tabela 5).

Foram obtidas também as relações C_{eff}/C_{total} para todas as interconexões dos circuitos *pci_bridge32* e *matrix_mult* e os histogramas com as distribuições de frequências para estas relações são apresentados nas Figuras 24 e 25. No experimento apresentado na Figura 24 são mostradas as relações C_{eff}/C_{total} nas interconexões do circuito *pci_bridge32*, divididas em três partes. Na primeira em verde, se encontram as frequências das relações C_{eff}/C_{total} nas interconexões com menor valor de resistência; a segunda parte, em vermelho, mostra a distribuição das frequências nas interconexões com valor médio de resistência; e a última parte, em azul, mostra as frequências para a última parte das interconexões, ou seja, as com maior valor de resistência total. Note que na Figura 24(a) todos os valores de frequência são mostrados, e na Figura 24(b), apenas os valores de frequência menores que 200 são mostrados. Este mesmo experimento foi também realizado para o circuito *matrix_mult* e seu resultado pode ser visualizado na Figura 25.

Analisando os histogramas apresentados nas Figuras 24 e 25, observa-se o impacto do efeito de *resistive shielding*. Nas interconexões com menor valor resistivo, o valor de C_{eff} é mais próximo de C_{total} na maioria dos casos. Quanto maior for o valor de resistências das interconexões, maior o efeito de *resistive shielding*, fazendo com que as relações C_{eff}/C_{total} assumam mais valores menores que 1.

5.4.2 Erro de *Arrival Times* nas Saídas Primárias

Este experimento avalia os erros nos *arrival times* obtidos pela ferramenta implementada neste trabalho em relação aos calculados pelo *PrimeTime*, nos circuitos *pci_bridge32* e *matrix_mult*. Este experimento complementa o experimento apresentado na Seção 5.4.1 na escolha do modelo a ser utilizado para as interconexões.

Como observado na Tabela 7, quando o circuito está na configuração de maior *leakage*, o valor de C_{eff} se aproxima de C_{total} ($C_{eff}/C_{total} = 0.99$), possibilitando o uso da abordagem de capacitância concentrada para representar as interconexões, refletindo em um resultado semelhante ao da abordagem de capacitância efetiva, porém com um *runtime* cerca de 3.5 vezes menor. A distribuição das frequências dos erros percentuais obtidos neste experimento podem ser visualizados na Figura 23.

Kahng et al. (2013) realizaram um experimento semelhante comparando quatro técnicas para o cálculo do atraso das interconexões e duas para a degradação do *slew*. No contexto de uma técnica de *gate sizing*, a ferramenta de análise de *timing* utilizada tem grande impacto no *runtime* da otimização. Tal experimento influenciou na escolha das técnicas para atraso da interconexão e degradação do *slew* (D2M (ALPERT; DEVGAN; KASHYAP, 2000) e PERI (KASHYAP et al., 2002), respectivamente).

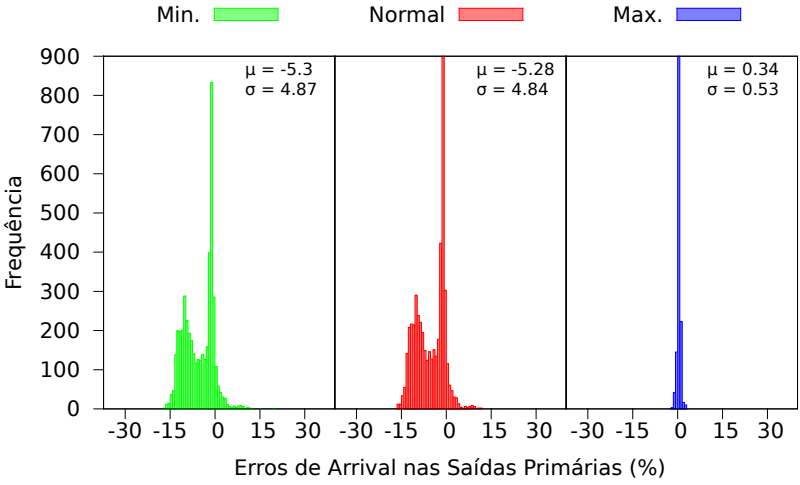
5.4.3 Nível Lógico *versus* Erro Relativo

Neste experimento os erros relativos percentuais dos *arrival times* de saída de cada porta lógica dos circuitos *pci_bridge32* e *matrix_mult* em relação aos *arrival times* reportados pelo *PrimeTime* foram analisados ao decorrer dos níveis lógicos dos circuitos.

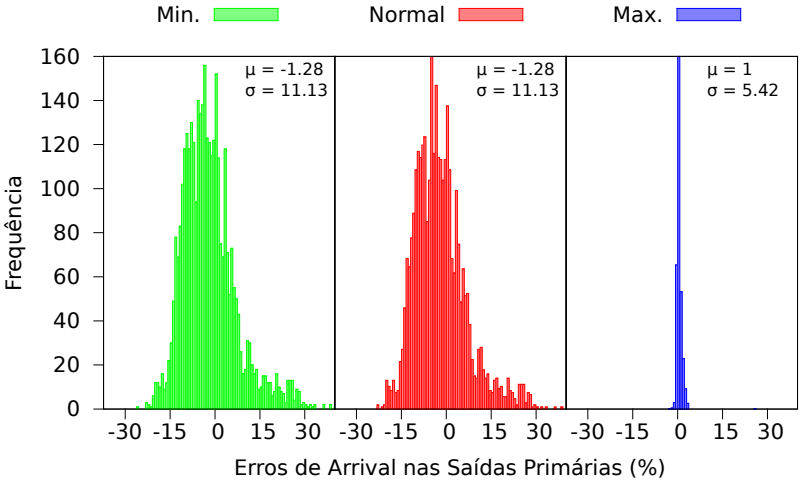
As Figuras 26 e 27 mostram os erros distribuídos pelos níveis lógicos nos circuitos *pci_bridge32* e *matrix_mult*, respectivamente. Em azul, estão apresentados os pontos de cada *arrival time* de saída. Na curva em vermelho, são mostrados os erros para os *arrival times* de saída das portas pertencentes ao caminho crítico. Na curva em verde, são mostrados os erros referentes às portas pertencentes ao caminho com o maior número de portas.

Ao analisar os gráficos das Figuras 26 e 27, observa-se que tanto os erros grandes, quanto os erros pequenos dos primeiros níveis lógicos, ao serem propagados, são estabilizados, convergindo para um número

próximo de -7%, o que se reflete também nos erros dos caminhos críticos e maiores caminhos.

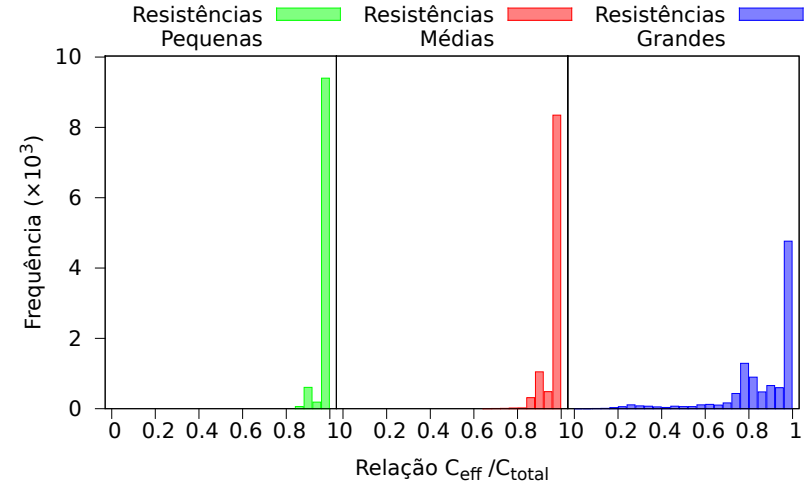


(a)

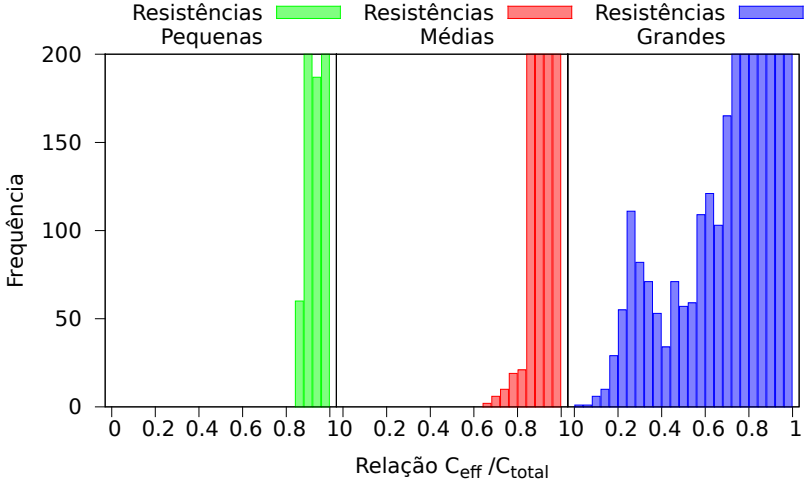


(b)

Figura 23 – Distribuição das frequências dos erros percentuais calculados nas saídas primárias dos circuitos: (a) *matrix_mult*; (b) *pci_bridge*32. Na primeira parte os erros foram calculados considerando a configuração de menor consumo de *leakage*. Na segunda parte, considerando a configuração padrão. E na terceira, considerando a configuração de maior consumo de *leakage*. μ e σ representam a média e desvio padrão das amostras, respectivamente.

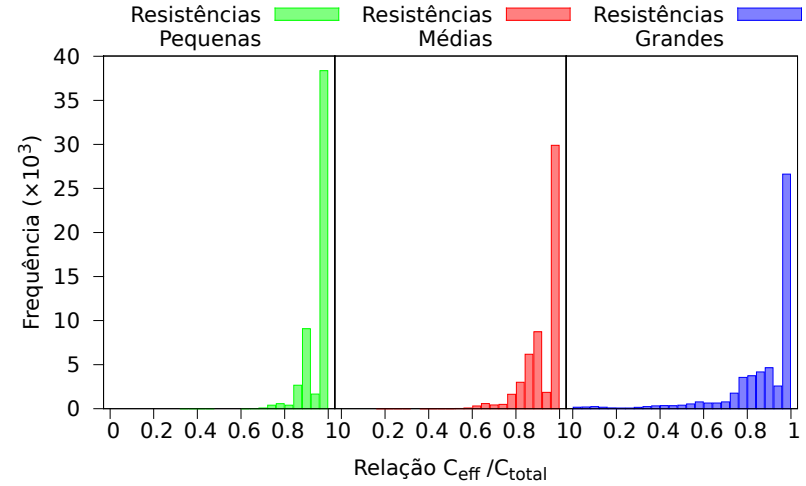


(a)

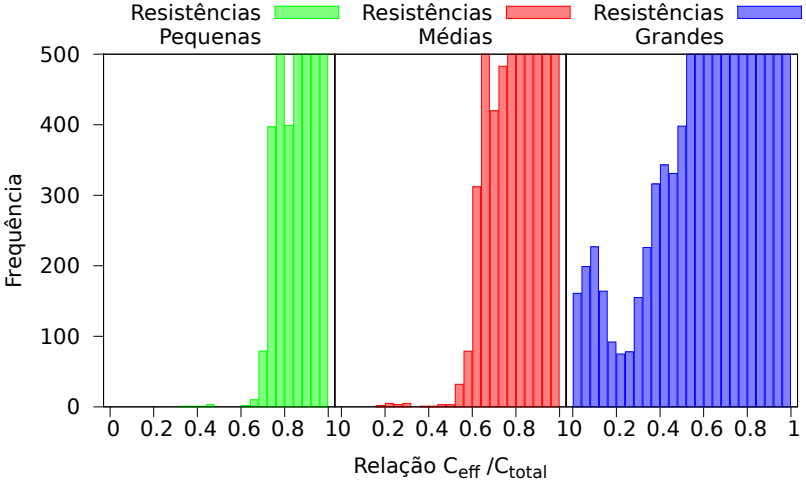


(b)

Figura 24 – Distribuição das frequências das relações C_{eff}/C_{total} das interconexões do circuito *pci_bridge32*. Na primeira parte, as frequências são das interconexões com menor valor de resistência total, na segunda parte, das com valor de resistência total médio, e na terceira, das com maiores valores de resistência total.



(a)



(b)

Figura 25 – Distribuição das frequências das relações C_{eff}/C_{total} das interconexões do circuito *matrix_mult*. Na primeira parte, as frequências são das interconexões com menor valor de resistência total, na segunda parte, das com valor de resistência total médio, e na terceira, das com maiores valores de resistência total.

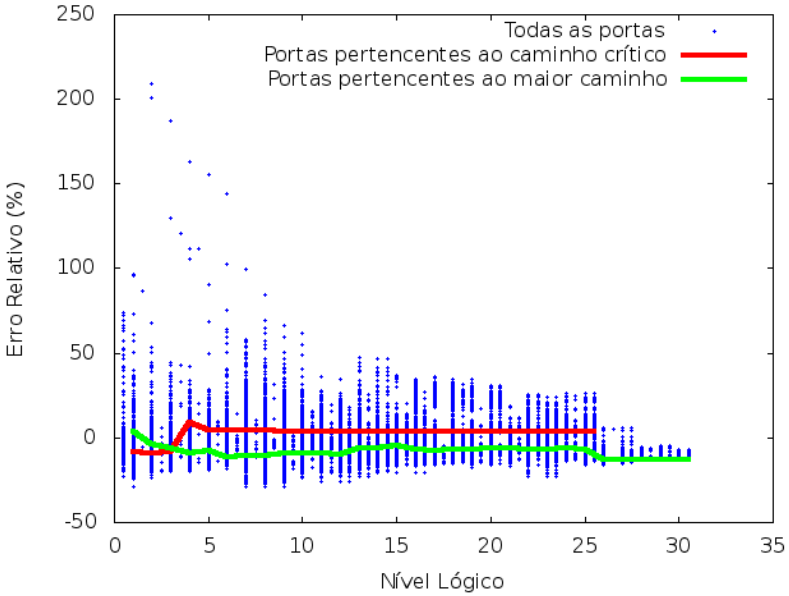


Figura 26 – Erro relativo dos *arrival times* em relação aos resultados obtidos pelo *PrimeTime*, ao decorrer dos níveis lógicos, no *benchmark pci_bridge*32. O *arrival time* utilizado na comparação é o *arrival time* no *timing point* de saída de cada porta lógica. Em azul, cada ponto representa uma porta lógica. Em vermelho, é a curva referente às portas lógicas pertencentes ao caminho crítico. A curva em verde, é referente às portas lógicas pertencentes ao maior caminho, ou seja, ao caminho com maior número de portas.

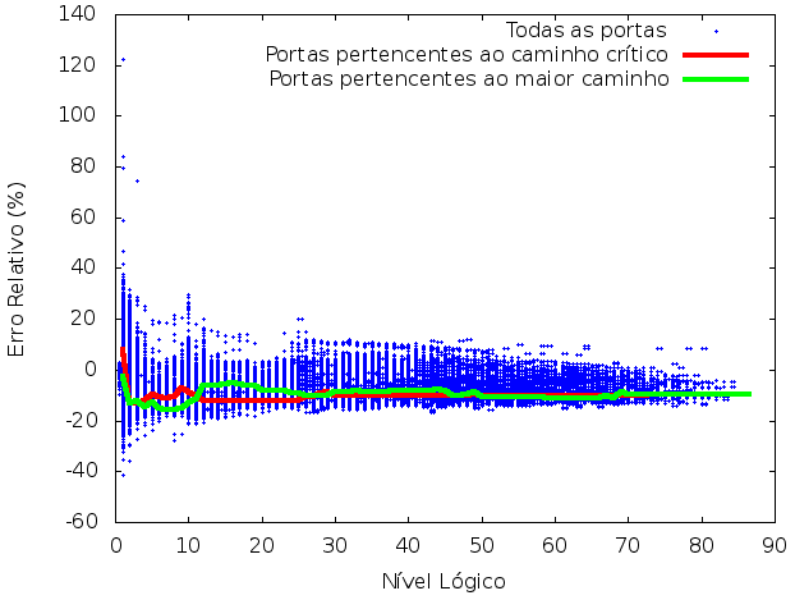


Figura 27 – Erro relativo dos *arrival times* em relação aos resultados obtidos pelo *PrimeTime*, ao decorrer dos níveis lógicos, no *benchmark matrix_mult*. O *arrival time* utilizado na comparação é o *arrival time* no *timing point* de saída de cada porta lógica. Em azul, cada ponto representa uma porta lógica. Em vermelho, é a curva referente às portas lógicas pertencentes ao caminho crítico. A curva em verde, é referente às portas lógicas pertencentes ao maior caminho, ou seja, ao caminho com maior número de portas.

6 CONCLUSÃO

Foram avaliados os impactos das interconexões no contexto da análise de *timing* estática utilizando uma infraestrutura experimental realista.

A consideração do atraso das interconexões no fluxo *standard cell* é de muita importância e a avaliação desses atrasos deve ser eficiente e precisa. Neste trabalho foi possível observar a importância da avaliação dos atrasos das interconexões e também, que a desconsideração da degradação do *slew* através das interconexões pode obter atrasos muito otimistas para o circuito, acarretando erros de cerca de 20% no valor do caminho crítico para os circuitos da competição de *sizing* do ISPD.

A abordagem da capacitância efetiva para interconexões implica na consideração do efeito de *resistive shielding*, o qual impacta na qualidade do cálculo do atraso do circuito. A ferramenta de análise de *timing* desenvolvida neste trabalho implementa a técnica de Puri, Kung e Drumm (2002) para o cálculo da capacitância efetiva, atraso das interconexões e degradação do *slew*. Tal ferramenta apresentou ser cerca de **17,02 vezes mais rápida** que o *PrimeTime*, obtendo resultados para *TNS* e *critical path* que subestimam em cerca de $8,85\%$ e $4,48\%$ respectivamente, os reportados pela ferramenta industrial.

A relação C_{eff}/C_{total} nos circuitos da competição de *sizing* do ISPD de 2013 mostrou-se na média, próxima de 1. A partir dessa informação, o modelo de capacitância concentrada para calcular o atraso dos *drivers*, juntamente com a técnica de Elmore com C_{total} e a técnica de Puri, Kung e Drumm (2002) para degradação do *slew* foi avaliada, apresentando estimativas pessimistas em $10,76\%$ para *TNS* nos circuitos testados (exceto o *netcard*) e $6,48\%$ para *critical path*, sendo que o tempo de execução é cerca de **3 vezes menor** que o da técnica considerando a C_{eff}.

6.1 TRABALHOS FUTUROS

Diversos trabalhos futuros podem ser realizados a fim de complementar a ferramenta avaliada neste trabalho, tais como:

- Investigação detalhada dos erros obtidos pela técnica implementada neste trabalho, quando comparada à ferramenta industrial;
- Avaliação da eficiência da técnica implementada no contexto de

uma técnica de otimização de fluxo *standard cell*, como por exemplo, *gate sizing*;

- Avaliação da técnica implementada utilizando bibliotecas *standard cell* e circuitos comerciais.

REFERÊNCIAS BIBLIOGRÁFICAS

ALPERT, C. J.; DEVGAN, A.; KASHYAP, C. A two moment rc delay metric for performance optimization. In: ACM. Proceedings of the 2000 international symposium on Physical design. [S.l.], 2000. p. 69–74.

BHASKER, J.; CHADHA, R. Static Timing Analysis for Nanometer Designs: A Pratical Approach. 1. ed. [S.l.]: Springer, 2009.

CONG, J. et al. Performance optimization of vlsi interconnect layout. Integr. VLSI J., Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, n. 1-2, p. 1–94, nov. 1996. ISSN 0167-9260. Disponível em: <http://dx.doi.org/10.1016/S0167-9260(96)00008-9>.

ELMORE, W. C. The transient response of damped linear networks with particular regard to wideband amplifiers. Journal of Applied Physics, AIP, v. 19, n. 1, p. 55–63, 1948. Disponível em: <http://link.aip.org/link/?JAP/19/55/1>.

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, R. et al. The Elmore Delay as a Bound for RC Trees with Generalized Input Signals. v. 16, n. 1, p. 95–104, 1997.

HOROWITZ, J. R. P. P. M. Signal delay in rc tree networks. IEEE transactions on computer-aided design, v. 2, n. 3, p. 202–211, 1983.

IEEE. Ieee standard for integrated circuit (ic) delay and power calculation system. IEEE Std 1481-1999, p. i–390, 1999.

INTEL. Intel(R) Core(TM) i7-920 Processor Specifications. 2008. Disponível em: <http://ark.intel.com/products/37147/Intel-Core-i7-920-Processor-8M-Cache-2_66-GHz-4_80-GTs-Intel-QPI>. Acesso em: 26/10/2013.

KAHNG, A. B. et al. High-performance gate sizing with a signoff timer. In: ACM. International Conference on Computer-Aided Design (ICCAD). [S.l.]: ACM, 2013.

KASHYAP, C. V.; ALPERT, C. J.; DEVGAN, A. An effective capacitance based delay metric for rc interconnect. In: IEEE PRESS. `Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design`. [S.l.], 2000. p. 229–235.

KASHYAP, C. V. et al. Peri: a technique for extending delay and slew metrics to ramp inputs. In: ACM. `Proceedings of the 8th ACM/IEEE international workshop on Timing issues in the specification and synthesis of digital systems`. [S.l.], 2002. p. 57–62.

LIVRAMENTO, V. dos S. `Sizing Discreto Baseado em Relaxação Lagrangeana para Minimização de Leakage em Circuitos Digitais`. Dissertação (Mestrado), 2013.

ODABASIOGLU, A.; CELIK, M.; PILEGGI, L. T. Prima: passive reduced-order interconnect macromodeling algorithm. In: IEEE COMPUTER SOCIETY. `Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design`. [S.l.], 1997. p. 58–65.

OZDAL, M. M. et al. An improved benchmark suite for the ispd-2013 discrete cell sizing contest. In: `Proceedings of ACM International Symposium on Physical Design`. [S.l.: s.n.], 2013. p. 168–170.

PILLAGE, L. T.; ROHRER, R. A. Asymptotic waveform evaluation for timing analysis. `Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on`, IEEE, v. 9, n. 4, p. 352–366, 1990.

PURI, R.; KUNG, D. S.; DRUMM, A. D. Fast and accurate wire delay estimation for physical synthesis of large asics. In: `Proceedings of the 12th ACM Great Lakes symposium on VLSI`. New York, NY, USA: ACM, 2002. (GLSVLSI '02), p. 30–36. ISBN 1-58113-462-2. Disponível em: <http://doi.acm.org/10.1145/505306.505314>.

RABAEY, J. M.; CHANDRAKASAN, A.; NIKOLIC, B. `Digital Integrated Circuits`. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2008. ISBN 0132219107, 9780132219105.

RYZHENKO, N.; BURNS, S. Standard cell routing via boolean satisfiability. In: `Proceedings of the 49th Annual Design Automation Conference`. New York, NY, USA: ACM, 2012. (DAC '12), p. 603–612. ISBN 978-1-4503-1199-1. Disponível em: <http://doi.acm.org/10.1145/2228360.2228470>.

SHEEHAN, B. N. Osculating thevenin model for predicting delay and slew of capacitively characterized cells. In: ACM. Proceedings of the 39th annual Design Automation Conference. [S.l.], 2002. p. 866–869.

SILVEIRA, L. M. et al. A coordinate-transformed arnoldi algorithm for generating guaranteed stable reduced-order models of rlc circuits. Computer Methods in Applied Mechanics and Engineering, Elsevier, v. 169, n. 3, p. 377–389, 1999.

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

SYNOPSYS. Synopsys PrimeTime SI Version H-2013.06 User Guide. 2013. Disponível em: <http://www.synopsys.com>. Acesso em: 31/10/2013.

WANG, M.; YANG, X.; SARRAFZADEH, M. Dragon2000: standard-cell placement tool for large industry circuits. In: IEEE PRESS. Proceedings of the 2000 IEEE/ACM international conference on Computer-aided design. [S.l.], 2000. p. 260–263.

ZHOU, Y. et al. A more effective ceff for slew estimation. In: IEEE. Integrated Circuit Design and Technology, 2007. ICICDT'07. IEEE International Conference on. [S.l.], 2007. p. 1–4.