Análise de *Timing* Estática e a Avaliação do Impacto do Atraso das Interconexões em Circuitos Digitais

Chrystian de Sousa Guth[1]

[1]Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Campus Universitário – Trindade – Florianópolis – SC – Brasil

csguth@inf.ufsc.br

Abstract. *Static timing analysis is the most used technique to estimate the delay in digital circuits during the Standard Cell flow. As the circuits must be represented in two different parts (the logic gates and interconnects), an efficient technique must be implemented to calculate the interconnect delay. The STA tool implemented in this work do the interconnect delay calculation getting the results that are, in average 4.28% optimistics than those that are obtained by a industrial tool, with a 8 times inferior runtime.*

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. Como os circuitos devem ser modelados em duas partes (portas lógicas e interconexões), uma técnica eficiente deve ser implementada para se estimar os atrasos das interconexões. 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.*

1. Introdução

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.

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

[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.

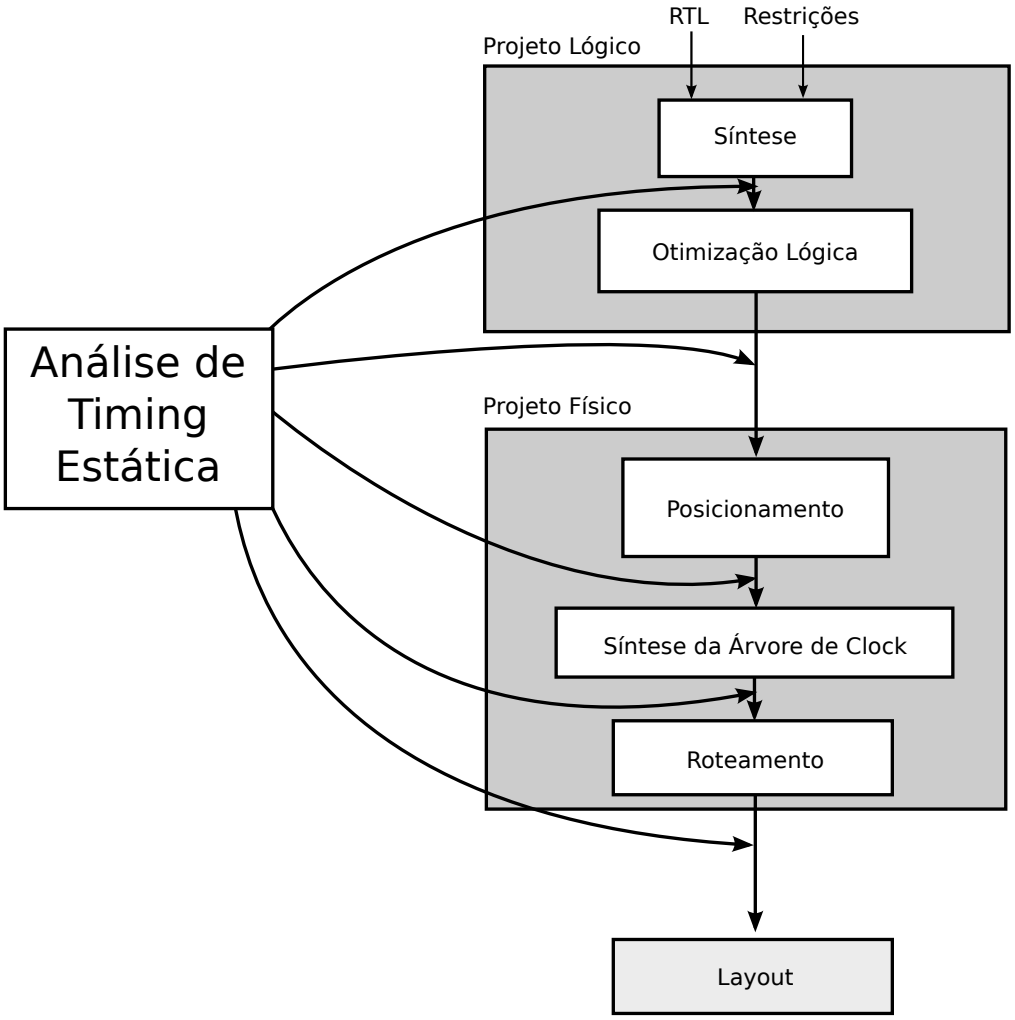


Figura 1. Fluxo de projeto *Standard Cell*. Adaptado de [Bhasker and Chadha 2009].

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*.

Este artigo se organiza da seguinte maneira: Na Seção 2 serão apresentados alguns conceitos básico para entendimento das técnicas que serão apresentadas posteriormente. A seguir, na Seção 3 será apresentada a técnica utilizada neste trabalho para cálculo das informações temporais dos circuitos digitais. Na Seção 4 será apresentada a análise de *timing* estática, bem como algumas particularidades no seu algoritmo. Finalmente, nas Seções 5 e 6 serão apresentados alguns resultados obtidos nos experimentos realizados e as conclusões finais, respectivamente.

2. Conceitos

As características temporais do circuito são derivadas das características temporais de suas partes, quais sejam, as portas lógicas e as interconexões que o compõem. O atraso das portas lógicas são obtidos das pré-caracterizações presentes nas bibliotecas de células. No que se refere às interconexões, elas geralmente são modeladas como árvores RC (Figura 2). Segundo [Rabaey et al. 2008], uma árvore RC possui três características: não possui *loops* resistivos; cada nodo possui uma capacitância com o *ground*; cada nodo possui um resistor que o conecta com o seu nodo pai.

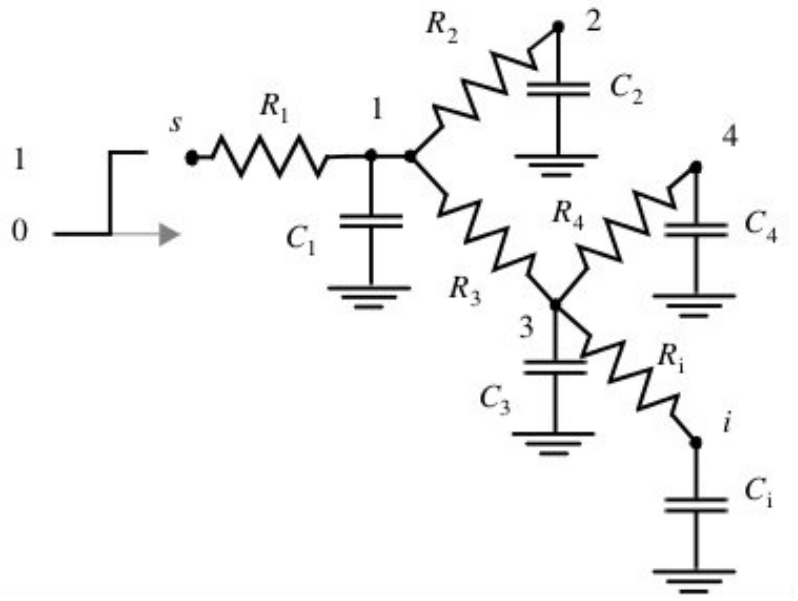


Figura 2. Uma árvore RC. Obtida de [Rabaey et al. 2008].

A Figura 3 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 [3]. Na 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 and Chadha 2009].

[3]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 3(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* $f1$.

- **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 3(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.
- **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 3(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. Cálculo das Características Temporais da Interconexão

Para que o atraso de uma interconexão seja estimado com precisão, um modelo de grafo (Figura 4) 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 $d\}$

A técnica de [Elmore 1948] é uma técnica baseada no primeiro momento da resposta ao impulso amplamente utilizada no cálculo dos atrasos das interconexões. De acordo com [Rabaey et al. 2008], em um nodo c_i da árvore RC, o atraso de Elmore (τ_i) pode ser facilmente calculado como:

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

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*.

[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 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 et al. 2002], ocorre em cinco passos:

1. **Inicialização:**
   (a) A capacitância efetiva C_{eff_i} de cada nodo c_i da Árvore RC é inicializada 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 1), 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 2.

$$slew_i = \frac{slew_j}{1 - \frac{R_i\ C_{eff_i}}{slew_j}(1 - e^{-\frac{slew_j}{R_i\ C_{eff_i}}})} \tag{2}$$

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

$$C_{eff_i} = C_i + \sum_{j \in filhos(i)} K_j \times C_{tot_j} \tag{3}$$

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

$$K_j = 1 - \frac{2R_jC_{eff_j}}{slew_i}(1 - e^{-\frac{slew_i}{2R_jC_{eff_j}}}) \tag{4}$$

   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 et al. 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 desta seção 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 and 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 and 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. 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.

Na STA, o circuito pode ser representado como um grafo direcionado acíclico (*DAG: Directed Acyclic Graph*), onde o conjunto dos vértices representa os pinos de

saída e entrada das portas lógicas e o conjunto das arestas representa as interconexões e os *timing arcs*. Para melhorar a eficiência da ferramenta de STA, os vértices são armazenados em listas ordenadas topologicamente.

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 5(b). 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.

5. Resultados

Essa seção tem por objetivo descrever os experimentos realizados neste trabalho e apresentar os resultados obtidos.

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 e 6) foram adotados como métricas para estimar a qualidade das informações de *timing* reportadas pela ferramenta implementada neste trabalho.

$$EP_t = \frac{(A_t - P_t)}{A_t} \times 100 \tag{5}$$

$$EMPA = \frac{\sum_{t=1}^{n} |EP_t|}{n} \tag{6}$$

O erro percentual é calculado para cada uma das informações comparadas com o *PrimeTime* utilizando a equação 5, 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_bridge*32: 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*).

5.1. 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 1.

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 $(6,48)$.

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

5.2. 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 (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. Os resultados deste experimento podem ser vistos na Tabela 2.

A penúltima linha apresenta a média dos valores de cada coluna e a última linha apresenta o $EMPA$ para cada uma das informações em relação ao *PrimeTime*. Com esse experimento, conclui-se que a ferramenta desenvolvida neste trabalho fornece informações de *timing* próximas às reportadas pelo *PrimeTime*[5], 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

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

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

os menores erros, nos menores circuitos.

No experimento mostrado na Tabela 3, 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 na Seção 3. Como esperado, os modelos utilizados neste experimento refletem em uma aproximação pessimista para o atraso do circuito[6], já que a técnica de Elmore pura[7] 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 4. 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 na Seção 3 é aplicada, sem considerar a degradação do *slew* nos destinos das interconexões.

6. Conclusões

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

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

[7]Sem utilizar a abordagem de capacitância efetiva.

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 3. 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 na Seção 3.

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 4. 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.

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 et al. 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 et al. 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

Bhasker, J. and Chadha, R. (2009). *Static Timing Analysis for Nanometer Designs: A Pratical Approach*. Springer, 1 edition.

Cong, J., He, L., Koh, C.-K., and Madden, P. H. (1996). Performance optimization of vlsi interconnect layout. *Integr. VLSI J.*, 21(1-2):1–94.

Elmore, W. C. (1948). The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19(1):55–63.

Guntzel, J. L. A. (2000). *Functional Timing Analysis of VLSI Circuits Containing Complex Gates*. PhD thesis, Universidade Federal do Rio Grande do Sul, RS-Brazil.

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

Intel (2008). Intel(r) core(tm) i7-920 processor specifications.

Puri, R., Kung, D. S., and Drumm, A. D. (2002). Fast and accurate wire delay estimation for physical synthesis of large asics. In *Proceedings of the 12th ACM Great Lakes symposium on VLSI*, GLSVLSI '02, pages 30–36, New York, NY, USA. ACM.

Rabaey, J. M., Chandrakasan, A., and Nikolic, B. (2008). *Digital Integrated Circuits*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition.

Sheehan, B. N. (2002). Osculating thevenin model for predicting delay and slew of capacitively characterized cells. In *Proceedings of the 39th annual Design Automation Conference*, pages 866–869. ACM.

Synopsys (2012). Synopsys primetime user's manual.

Zhou, Y., Li, Z., Kanj, R., Papa, D., Nassif, S., and Shi, W. (2007). A more effective ceff for slew estimation. In *Integrated Circuit Design and Technology, 2007. ICICDT'07. IEEE International Conference on*, pages 1–4. IEEE.

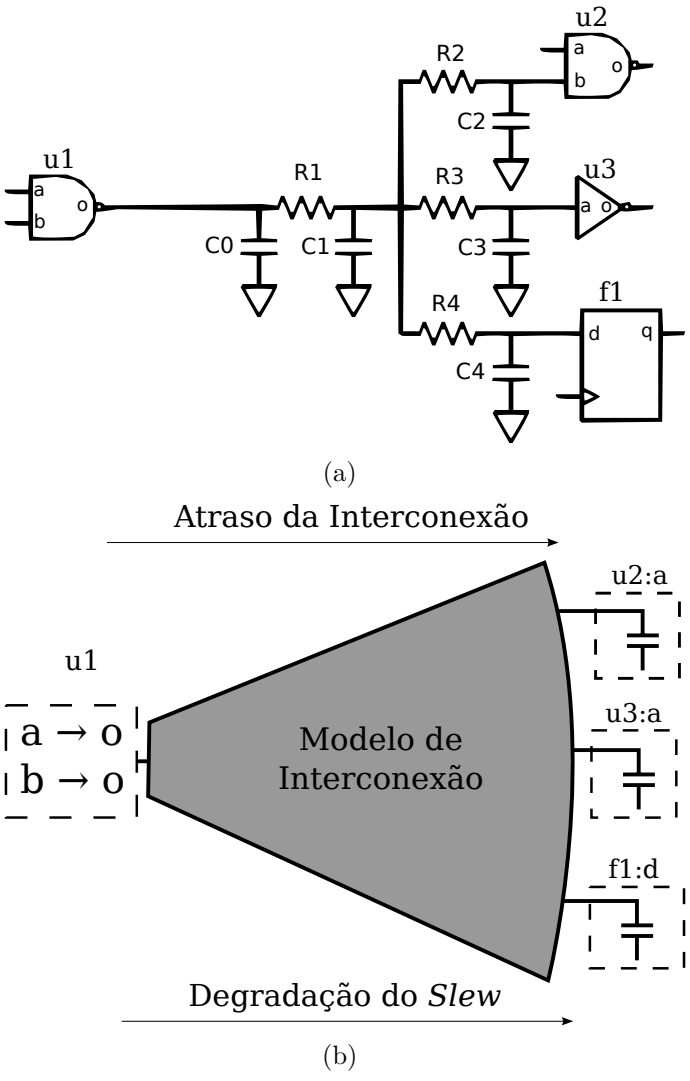


Figura 3. (a) Um circuito composto por três portas lógicas ($u1$, $u2$ e $u3$), uma célula sequencial ($f1$) e uma interconexão em forma de árvore RC, que liga a saída de $u1$ às entradas de $u2$, $u3$ e $f1$; (b) São apresentadas as modelagens para os *timing arcs* da porta lógica $u1$; O modelo da 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.

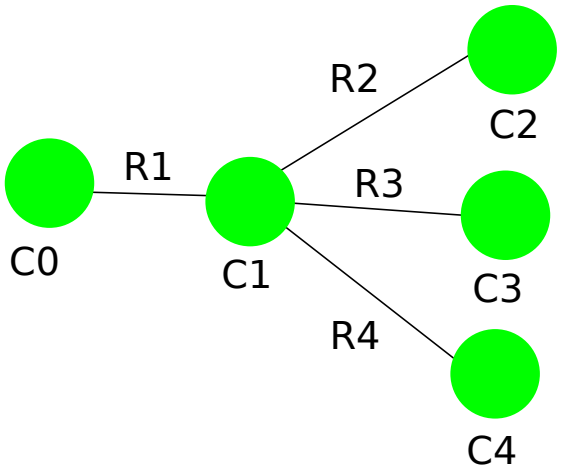
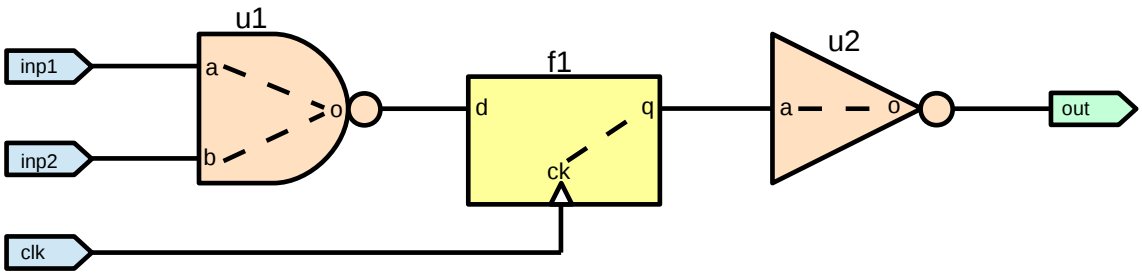
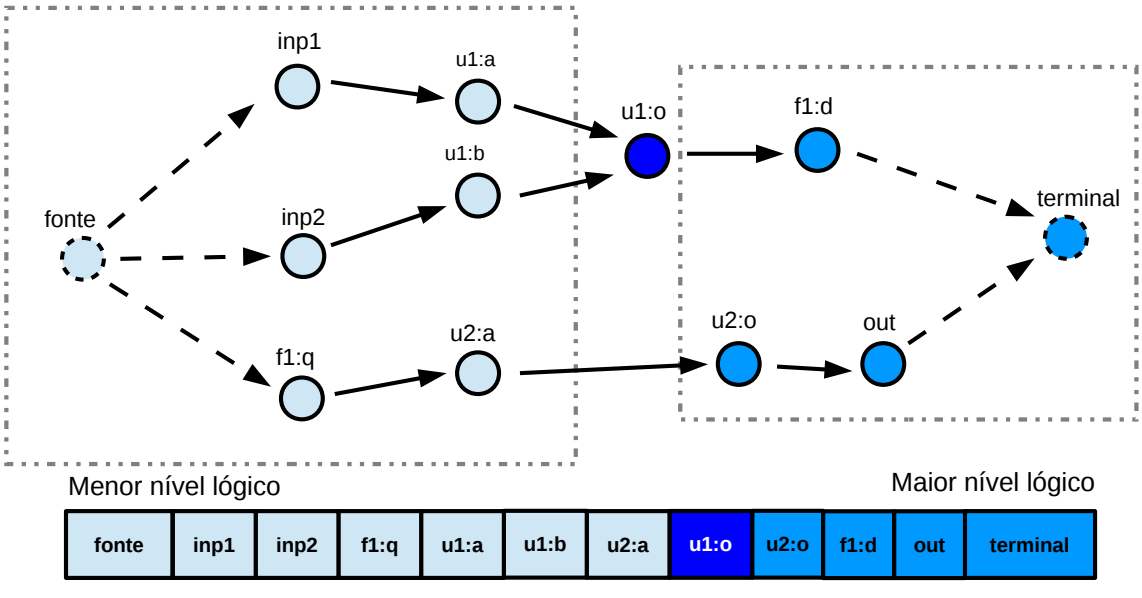


Figura 4. O grafo correspondente à interconexão da Figura 3(a), com cinco vértices e quatro arestas.



(a)



(b)

Figura 5. (a) Circuito *simple* retirado do banco de *benchmarks* da competição de *sizing* do ISPD; (b) Grafo correspondente ao circuito da letra (a).