FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

Departamento de Electrónica

(66.33) - Laboratorio de sistemas digitales

Trabajo Práctico Final

Verificación del modelo Logical Effort usando métodos heurísticos para una cadena de inversores

Autores

Fernando Daniel Kreps - 83157
Andrew Parlane

Introducción

Este trabajo ataca el problema de determinar el área óptima que resuelva el tiempo de propagación mínimo dado un conjunto de transistores que conforman un path simple.
Cuando se caracteriza el tamaño de un transistor, se lo suele hacer sobre el ancho, ya que en lógica digital por lo general se usa transistores con el longitud mínima permitido por la tecnología.
Este tipo de optimización, suele ser compleja ya que involucra un *trade-off* entre el ancho de los transistores de una etapa y el efecto capacitivo que ejerce sobre etapas colindantes, por ejemplo, aumentando el ancho se disminuye la resistencia, por ende la etapa tendrá menor tiempo de propagación a costa de incrementar la capacitancia de carga de la etapa anterior y por ende, aumentando el tiempo de propagación de la última.
Para este tipo de trabajo, hay métodos heurísticos como TILOS (TImed LOgic Synthesizer), optimización convexa iCONTRAST, multiplicadores de Lagrange entre otros.
Es importante destacar que este problema es de optimización convexo, tal cual está demostrado en [S04], lo cual se tiene las siguientes propiedades[SEC13]:

- Cada mínimo local es un mínimo absoluto
- Si la función a analizar es estrictamente convexa, por lo menos, hay un único punto de optimización
- El conjunto de optimizadores es convexo

Objetivo del Trabajo

Se busca conocer el grado de exactitud que posee el método logical effort utilizando como referencia simulaciones sobre los modelos nMOS/pMOS spice de TSMC180.
Las simulaciones se realizarán usando el método heurístico con el objetivo de obtener la combinación de anchos de inversores que optimice el tiempo de propagación del path, el cual es el mismo objetivo que persigue Logical Effort.
En este trabajo se ejecutará varios experimentos de simulación de un sistema de inversores CMOS mediante la biblioteca PySPICE de python.

Logical Effort

Logical effort expresa el tiempo de propagación de un camino lógico normalizando el tiempo de propagación de cada etapa en base al tiempo de propagación de un inversor mínimo diseñado con la misma tecnología.
Una vez normalizado el tiempo de propagación de una compuerta, este valor se expresa como

$$d = gh + p$$

$$d_{absoluto} = d \cdot \tau$$

Donde

- ***d*** : delay normalizado
- ***g***: Relación de capacitancia de gate entre la compuerta en análisis y la de un inversor de la misma tecnología capaz de entregar la misma corriente de salida, por ende está relacionado con el manejo de corriente en la topología de dicha compuerta
- ***h*** : está definido como C_{out}/C_{in} , donde C_{out} representa la capacidad de carga de la compuerta y C_{in} es la capacidad presente en el gate de la compuerta.
- ***p*** : es el delay intrínseco a la compuerta, el cual se mide cuando la compuerta no posee alguna carga.
- τ : Es el parámetro de normalización de velocidad, correspondiente a la velocidad de un inversor mínimo

De lo anterior se desprende que

- **h**: Relaciona la carga del transistor C_{out} **,**con el tamaño de los transistores en la entrada de la compuerta C_{in}
- **g** : indica que tan “peor” es el manejo de corriente de una compuerta comparada con un inversor de la misma tecnología, dicho valor solo depende de la topología del circuito, por ende es un valor normalizado con respecto al inversor.

Los parámetros ***g*** y ***h*** definen el *effort delay* o *stage effort como* $f = gh$, el cual depende de solo de la carga de la compuerta.

Analisis de logical effort multietapa

Está técnica es aplicada en 2 formas para optimizar velocidad en lógica de varias etapas

- Revela el número óptimo de etapas para usar en una red
- Revela el delay total mínimo balanceando los delay de cada etapa de una red

Se denota el logical effort de varias etapas como el producto de logical effort de etapas simples

$$G = \prod g_i$$

Para el caso del electrical effort, es simplemente el cociente entre la carga de los 2 extremos del camino.

$$H = C_{out} / C_{in}$$

Este último parámetro se puede expresar en función de los anchos de la compuerta

$$H = C_{out} / C_{in} = (W_{out}/W_{in})$$

Se introduce un nueva medición de effort llamado ***branching effort***, que relaciona la cargas capacitivas del path analizado o activado *(on)* con respecto a los restantes *(off)*

$b = \frac{C_{on-path} - C_{off-path}}{C_{on-path}}$ notar que si no hay branches $\Rightarrow b = 1$

$B = \prod b_i$: Se tiene en cuenta a todos los branches

Podemos definir el análogo del effort delay f para todos los paths del circuito como

$$F = GBH$$

Path Delay

Path delay es la suma de todos los delays de cada etapa del camino lógico, este delay se puede descomponer en

- Effort delay : D_F
- Parasitic delay : P

$$D[unidades\ de\ retardo] = \sum d_i = D_F + P$$

$$D_F = \sum g_i h_i \; ; \; P = \sum p_i$$

El objetivo es encontrar el path que menor delay tenga, y este se dará cuando cada etapa tenga el mismo delay

$$\hat{f} \; = \; g_i h_i \; = \; F^{\,(1/N)} \quad \forall \; i$$

Donde **N** es el número de etapas en la ruta, y $\hat{f}$ effort delay optimo.

Esto deja como resultado el delay mínimo que puede tener una etapa de la red

$$\hat{D} \; = N \; \cdot F^{1/N} + P$$

La forma de alcanzar dicho effort para cada stage es dimensionando los gates de cada etapa, tal que se cumpla

$$h_i \; = \; F^{1/N} / g_i$$

De aquí se puede obtener la capacitancia para el gate de cada etapa de forma recursiva, comenzando por la carga de la ruta.

$$C_{in-i} = \frac{C_{out-i} \cdot g_i}{\hat{f}}$$

Esto determina la capacitancia de entrada de cada puerta, que luego puede distribuirse adecuadamente entre los transistores conectados a la entrada.

Método de Heurístico basado en método de Monte Carlo

El método de Monte Carlo es esencialmente un análisis estadístico que ayudará a calcular la respuesta de un circuito modelado con parámetros aleatorios, los cuales varían entre límites de tolerancia acotados. Con el fin de una convergencia más rápida, en este punto se decide modificar dinámicamente los límites de dichos anchos, en base a los nuevos “mejores anchos” encontrados, o sea, los que dan en nuevo mejor tiempo de propagación, esta estrategia aprovecha el hecho de que no hay mínimos locales, sino un único mínimo absoluto.
Los valores aleatorios se generan mediante una estadística definida, que en el caso de este trabajo, será una distribución uniforme, debido a que no tenemos información que los haga estadísticamente dependientes.

El método genera nuevos conjuntos de valores de forma repetida, dando una distribución en la salida, la cual corresponde al comportamiento “más probable” del sistema, por ende mientras más corridas se generen, más información obtendremos con respecto al comportamiento.
No es raro ejecutar cientos o miles de repeticiones hasta llegar a una descripción satisfactoria del sistema.
El método de montecarlo suele ser utilizando para verificar la robustez en el comportamiento de un sistema ante cambios aleatorios en sus parámetros, o hallar valores óptimos (sean mínimos o máximos absolutos) para lo cual no hay una forma suficientemente simple o una solución “cerrada”

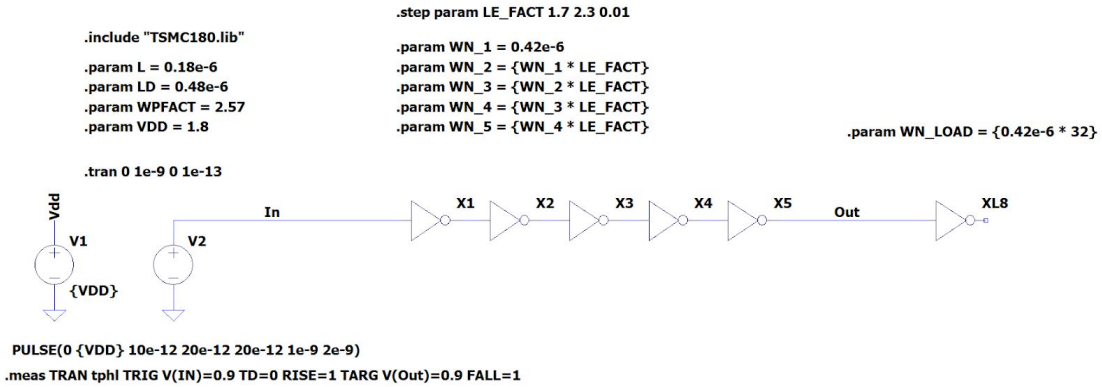
Este trabajo persigue la optimización de tiempo de propagación utilizando como parámetros de ajuste los anchos de gate en una cadena de inversores, se espera que esos valores sean aproximados a los predichos por el método de logical effort.

Sistema a analizar

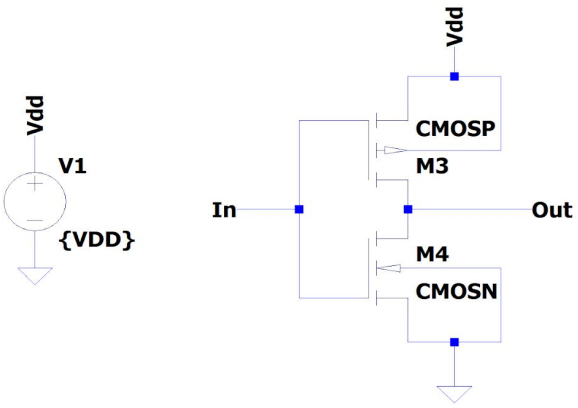
El trabajo está enfocado a hallar los anchos de compuertas óptimos para menor delay de propagación, utilizando solamente inversores. Esto permitirá lograr simulaciones en tiempos razonables, dada las unidades de cómputo utilizadas.

Esquema LTSpice

En la figura de abajo se muestra la topología analizada y la biblioteca de SPICE que especifica la tecnología utilizada en las simulaciones, tanto de LTSpice como en PySPICE .



Las compuertas están especificadas de la siguiente manera




```
.include "TSMC180.lib"
.func ADS(w, ld) = {w * ld}
.func PDS(w, ld) {w + 2*ld}
.param WP = {WN * WPFACT}
```

Análisis de simetría del inversor

Para este trabajo se quiere un modelo de un inversor simétrico, así que el tiempo de propagación por un flanco ascendente es igual al lo de un flanco descendente. Se puede cumplir esto, eligiendo el tamaño del transistor pMOS en comparación con el transistor nMOS:

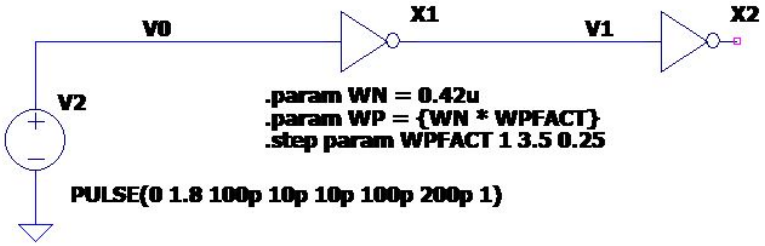
$$WP = W_{PFACTOR} \cdot WN$$

Para encontrar el $W_{PFACTOR}$ que da $T_{PHL} = T_{PLH} = T_P$, se arma un esquemático de LTSpice con un inversor cargado por otro inversor igual [WPFACT], se usa el directivo SPICE “.step” para ejecutar una simulación para cada valor de $W_{PFACTOR}$ en un intervalo. También se usa el directivo SPICE “.meas” para medir T_{PHL} y T_{PLH}, y calcular el T_P.

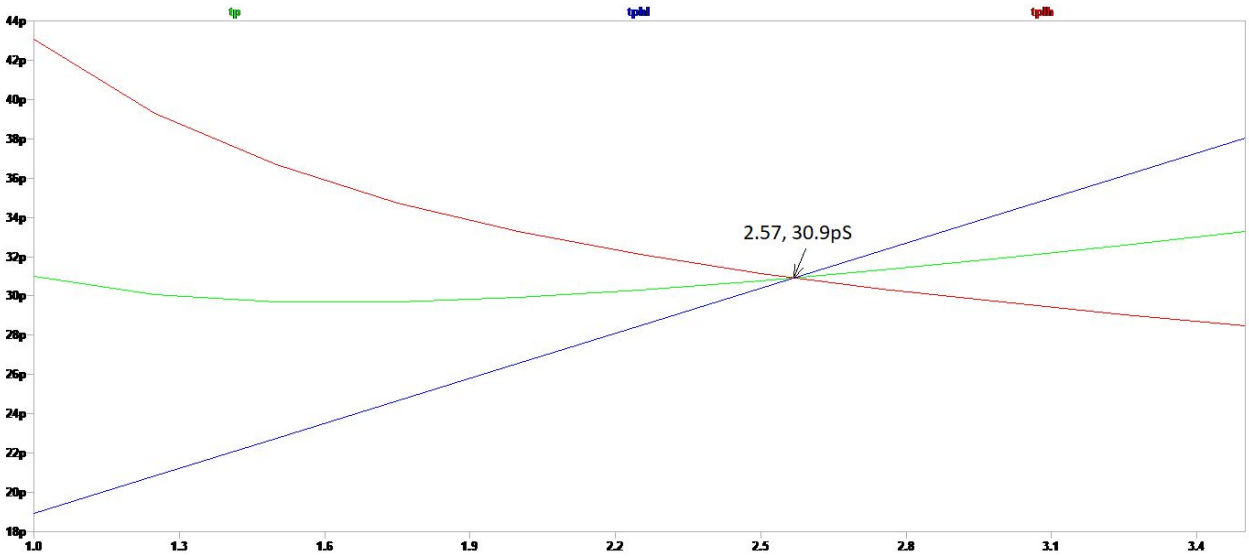
```
.include "TSMC180.lib"
.func ADS(w, ld) = {w * ld}
.func PDS(w, ld) {w + 2*ld}
.param L = 0.18u
.param LD = 0.48u
.param VDD = 1.8

.tran 0 300p 0p 0.1p
```

```
.meas TRAN tphl TRIG V(V0)=0.9 TD=0 RISE=1 TARG V(V1)=0.9 FALL=1
.meas TRAN tplh TRIG V(V0)=0.9 TD=0 FALL=1 TARG V(V1)=0.9 RISE=1
.meas TRAN tp PARAM (tphl + tplh)/2
```



Las simulaciones muestran que por $W_{PFACTOR}$ = 2.57, $T_{PHL} = T_{PLH} = T_P$.



Parámetros de Logical Effort

Logical Effort tiene unos parámetros los que dependen en la tecnología usado. $\boldsymbol{\tau}$ es el delay básico, tiene unidades de segundos, y es usado para convertir un delay abstracto: d, hasta un tiempo $d_{abs} = d\boldsymbol{\tau}$. También hay un conjunto de parámetros p, el delay parásito de cada compuerta. En este trabajo solo se usa inversores, así solo es necesario conocer p_{inv}. Para encontrar estos parámetros se usa la ecuación [calib]:

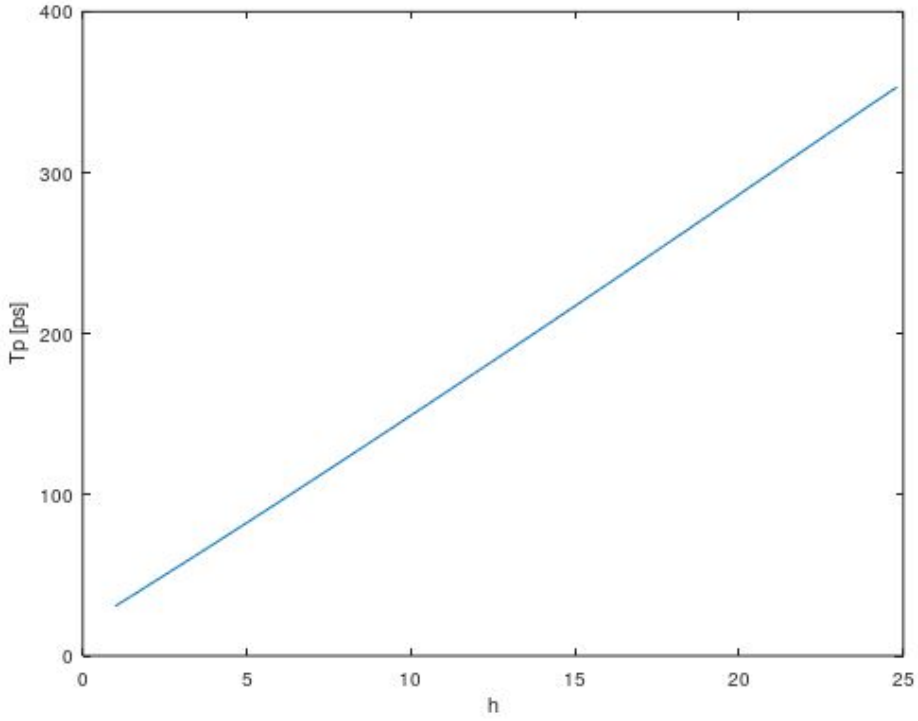
$d_{abs} = T_P = (f + p)\boldsymbol{\tau} = (gh + p)\boldsymbol{\tau}$.

En el caso de un inversor, g = 1 por definición, y $p = p_{inv}$. Así:

$d_{abs} = T_P = h\boldsymbol{\tau} + p_{inv}\boldsymbol{\tau}$.

Se realiza unas simulaciones en LTSpice para encontrar T_P contra $h = C_{out}/C_{in} = W_{load}/W$. Usando un inversor con WN = 0.42 µm, y una carga de un otro inversor con su WN determinado por el directivo SPICE ".step" se puede obtener el T_P por un rango de valores de $h = WN_{LOAD} / 0.42$ µm. El resultado es una línea recta con pendiente 13.57 ps, y intercepto con el eje-Y a 15.2 ps. Así:

$\boldsymbol{\tau}$ = 13.57 ps
p_{inv} = 1.1



Delay asociado al modelo

El delay asociado al modelo de logical effort es interpretado como al carga y descarga de un circuito RC, si no tenemos en cuenta las capacitancias parásitas, ya que en el modelo analizado no lo contemplamos

$$d_{abs} = kR_iC_{out}$$

Donde

C_{out} es la capacidad vista desde la salida de la compuerta

R_i es la resistencia vista desde la entrada de la compuerta, cuando la compuerta se encuentra con un estado lógico definido

k es una constante que depende del proceso de fabricación

Para poder obtener el ancho de los transistores que optimice el delay de propagación se modela la compuerta como escala por un término α con respecto a una compuerta *template*

El factor de α , básicamente escala el ancho de todo los transistores, dejando la longitud de los transistores sin cambios.

Dada la compuerta de *template* se redefinen los parámetros antes mencionados

$$C_{in} = \alpha \cdot C_t \quad R_i = R_t/\alpha$$

Tanto los parámetros C_t y R_t quedan definidos por el área de los transistores y la tecnología de fabricación

$$C_t = k_1 W_n L_n + k_1 W_p L_p$$

$\frac{1}{R_t} = k_1 \mu_n W_n L_n = k_1 \mu_p W_p L_p$, que es la admitancia de pull up y pull down respectivamente para el modelo de inversor usado en la librería TSMC180

Esto nos permite escalar la ecuación de delay de propagación para dejarla en función de las capacitancias de entrada y de salida

$$d_{abs} = kR_i C_{out} = k(R_t/\alpha)(C_{in} C_{out}/C_{in}) = k(R_t/\alpha)(C_t \cdot \alpha)(C_{out}/C_{in})$$

Dado que se compara el modelo de compuerta con un inversor de la misma tecnología, se agregan las constantes de delay del inversor

$$d_{abs} = k(R_t/\alpha)(C_t \cdot \alpha)(C_{out}/C_{in}) = (kR_{inv} C_{inv})(R_t C_t/R_{inv} C_{inv})(C_{out}/C_{in})$$

En base a la ecuación de logical effort

$$d_{abs} = \tau \cdot g \cdot h$$

se puede distinguir las relaciones antes mencionadas

Delay de una compuerta inversora : $\tau = C_{inv} R_{inv} k$

Logical effort : $g = (R_t C_t)/(C_{inv} R_{inv})$

Como se mencionó antes, este parámetro es igual a 1 para el caso de un inversor

Electrical effort : $h = (C_{out}/C_{in})$ la relación de capacitancia de entrada y salida de la compuerta

Este último parámetro se puede expresar en función de los anchos de la compuerta, el cual se ajustará para minimizar el tiempo de propagación.

$$h = (W_{out}/W_{in})$$

Simulaciones de Delay

En esta sección se analiza el comportamiento del tiempo de propagación en un inversor con respecto al ancho del gate pero manteniendo fija la carga, siendo esta última otro inversor con ancho de gate $W_{load} = 32 \cdot W_{min} = 32 \cdot 0,42\ \mu m$, donde W_{min} es el ancho mínimo de un transistor en la tecnología TSMC180, y como se verá más adelante, todos los anchos son definidos como productos de W_{min}.
Se espera que el modelo demuestre una relación inversamente proporcional entre el tiempo de propagación y el ancho de la compuerta del inversor driver.

Se han utilizado dos criterios con respecto al tiempo de propagación, el primero y el que será usado para las simulaciones debido a la simetría entre rise y fall time, se define

$$tp = t_{output-50\%VDD} - tp = t_{input-50\%VDD}$$

El segundo criterio se define

$$tp = t_{output-80\%VDD} - tp = t_{input-20\%VDD}$$

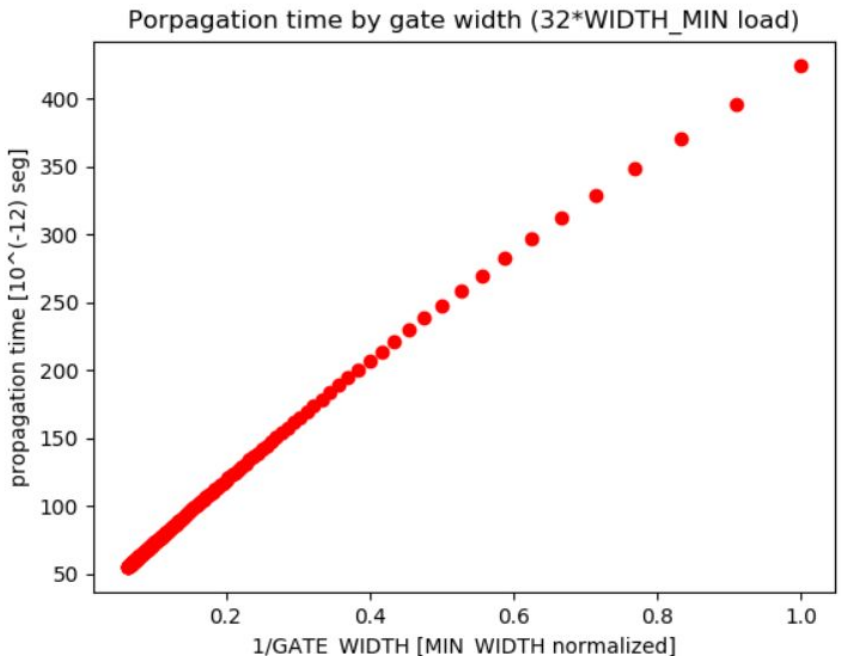
Siempre teniendo en cuenta que el modelo está siendo excitado por un flanco ascendente definido por:

- *Tensión inicial = 0V*
- *Tensión Final = Vdd*
- *Rise Time = 20 pSeg*
- *Fall Time = 20 pSeg*

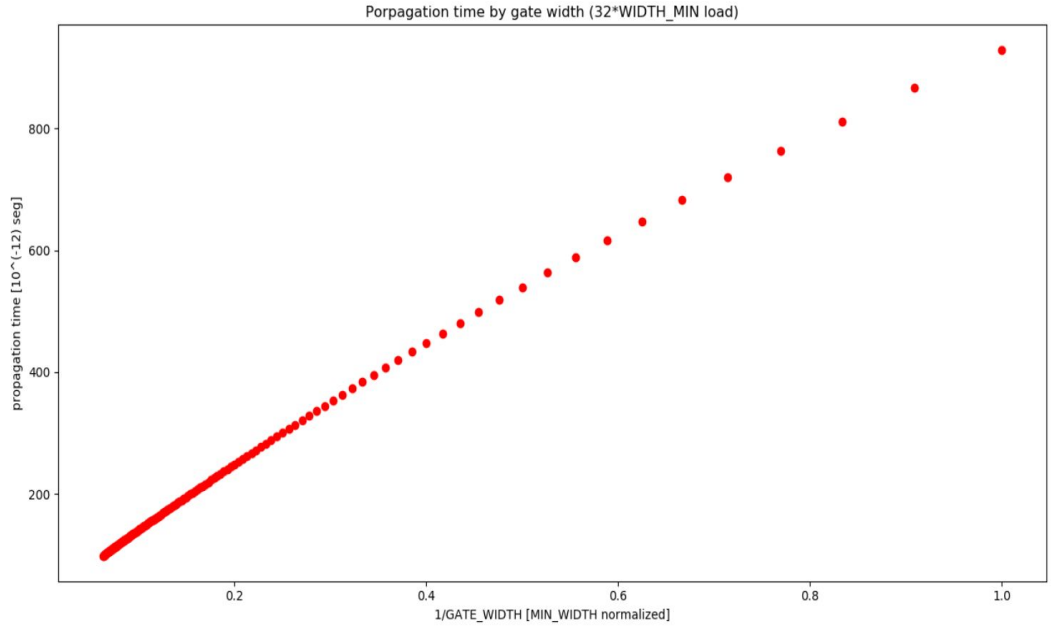
Se han tomado 160 valores de anchos, equidistantes, entre $0,42\ \mu m$ y $0.42\ \mu m \cdot 16 = 6,72\ \mu m$

Arrojando los siguientes resultados para el criterio

$$tp = t_{output-50\%VDD} - tp = t_{input-50\%VDD}$$



Y para el criterio $tp = t_{output-80\%VDD} - tp = t_{input-20\%VDD}$



Se verifica que el tiempo de propagación tiene una relación inversamente proporcional con el ancho de la compuerta

Análisis teórico del esquema

Análisis de los parámetros involucrados

Observando el esquema de 5 inversores en una cadena con carga de un inversor de ancho $32 \cdot W_{min}$, y el primer inversor con ancho fijo W_{min} se puede definir
De la definición de logical effort, tenemos que

$$G = \prod g_i$$

$g_{i_{INV}} = 1$ por lo que el logical effort del camino será $G = \prod g_i = 1$

El branching effort es $B = 1$ por que no hay ramas, mientras que el electrical effort es $H = (W_{out}/W_{in}) = 32 \cdot W_{min} / W_{min} = 32$.

Por ende el path effort sera : $F = GBH = 32$
El delay relativo sera: $\hat{D} = N \cdot F^{1/N} + P_{INV}$

Ahora podemos derivar las capacitancias que debería tener cada etapa dado que

$$\hat{f} = 32^{(1/5)} = 2$$

$$C_{in-i} = \frac{C_{out-i} \cdot g_i}{\hat{f}}$$

O lo que es lo mismo :

$$C_{in-i} \cdot \hat{f} = C_{out-i} \cdot g_i = C_{out-i}$$

Por lo antes mencionado, la capacitancia y el ancho del los transistores son proporcionales

$$W_{in-i} \cdot \hat{f} = W_{out-i} \cdot g_i = W_{out-i}$$

Para nuestro modelo se cumple que se parte de un ancho definido, llamado $W_{min} = 0,42\ \mu m$, entonce el método logical effort predice que los anchos óptimos serán:

$W_1 = 0,42\ \mu m$
$W_2 = 0.42\ \mu m \cdot 2 = 0,84\ \mu m$
$W_3 = 0.42\ \mu m \cdot 4 = 1,64\ \mu m$
$W_4 = 0.42\ \mu m \cdot 8 = 3,36\ \mu m$
$W_4 = 0.42\ \mu m \cdot 16 = 6,72\ \mu m$

El tiempo de propagación de una celda básica de inversor, como el promedio de rise time y fall time siendo en nuestro caso $\tau_{pd} = 17,30\ ps$, ya hallado mediante simulación LTSpice.

Utilizando la ecuación de delay absoluto de logical effort
$\hat{D} = (N \cdot F^{1/N} + P) \cdot \tau = (N \cdot F^{1/N} + N \cdot p_{inv}) \cdot \tau = (32^{1/5} + 1.1) \cdot 5 \cdot 13.57\ ps$ = 210 ps
El cual sería el tiempo de propagación óptimo estimado.

Algoritmo propuesto

Como se indicó antes el algoritmo tiene como objetivo generar muestras aleatorias de anchos de gates entre valores coherentes al modelo. Estos valores aleatorios se irán concentrado alrededor de los valores que generen mínimos.

```
/* Partimos de un "mejor resultado" ridículamente grande: */
gate_widths = UNIFORM(width_min, width_min*40)
current_best_tp = 100 seg
sim_time = 100 nseg
for i from 1 to cant_simulations {
      new_tp = do_simulation(sim_time, gate_widths)
      If new_tp < current_best_tp {
            best_widths = gate_widths
            current_best_tp = new_tp
            sim_time = new_tp +  50 pseg
      }
      for i in range of best_widths {
            min_aux_w = MAX(width_min, best_widths[i]/2)
            max_aux_w = MIN(width_min*40, best_widths[i]*2)
            gate_widths[i] = UNIFORM(min_aux_w, max_aux_w)
      }
}
```

Como se ve, los anchos para una nueva simulación son propuestos de forma aleatoria, entre un rango que se va actualizando a medida que se encuentra un nuevo mínimo, la convergencia a un resultado válido queda garantizado por el hecho de que *cada mínimo local es un mínimo global.*
La constante utilizada ***width_min*** está caracterizada en el modelo bajo análisis siendo 0.42µm

do_simulation() obtiene la diferencia de tiempos entrada/salida, para los cruces al 50% de Vdd (1,8 Volts), este criterio se optó dada la simetría entre rise time y fall time

Simulaciones realizadas

Se ejecutaron 25000 simulaciones de montecarlo, con el fin de obtener una cantidad de adecuada de datos en un tiempo de procesamiento razonable.

Desde el modelo simulado por el PySpice, para el caso de los anchos óptimos estimados por logical effort se obtienen los siguientes valores:

tp(pSeg)	width-1(μm)	width-2(μm)	width-3(μm)	width-4(μm)	width-5(μm)
308.7	0.42	0.84	1.68	3.36	6.73

Ahora viendo las simulaciones de Monte Carlo y tomando los 3 resultados arrojados con menor tiempo:

tp(pSeg)	width-1(μm)	width-2(μm)	width-3(μm)	width-4(μm)	width-5(μm)
306.2	0.42	0.84	1.36	2.34	4.83
306.3	0.42	0.74	1.22	2.14	4.46
306.4	0.42	0.73	1.44	2.54	4.91

Se observa que la estimación de anchos óptimos difiere más a medida que el ancho adecuado es más grande.

El experimento estadístico se ha repetido numerosas veces y se ha llegado a resultados muy próximos. Esto es lo esperado, debido a que se trata de un problema de optimización convexa, y el punto de convergencia es absoluto [S04] .

Conclusiones

Este trabajo detalla el uso de un algoritmo heurístico para hallar los óptimos anchos de las compuertas, resaltando el alto grado de exactitud obtenido, menor al 1% para las compuertas pequeñas y una diferencia considerable de tamaño óptimo para las celdas de mayor ancho.
A lo largo de este trabajo se han visto los beneficios de poder contar con una biblioteca de simulación spice para python, lo cual permite el uso de una inmensa cantidad de herramientas de automatización como de análisis y presentación de datos, siempre y cuando la exigencia sobre exactitud y/o tiempo de procesamiento lo permita.
Si bien el modelo analizado es simple, la misma técnica puede utilizarse para optimizar modelos más complejos, ya que este tipo de problemas tiene convergencia hacia resultados correctos, como se detalló antes.
Aunque es posible encontrar resultados con tiempo de propagación menor a lo que logical effort ofrece, sigue siendo una técnica poderosa. Da resultados aceptables considerando que es un cálculo simple y directo. Este trabajo ofrece una técnica que obtiene resultados más precisos a costa de tiempo de cómputo (en el orden de de las decenas de minutos para 25000 ciclos de simulaciones por una ruta sencillo).
Una de las métricas más importantes de un ASIC es el área del diseño, así la técnica desarrollada en este trabajo tiene la ventaja que produce paths con un área más reducido de lo que da logical effort. El mejor path encontrado con esta técnica tiene un área aproximadamente 75% del área dada por logical effort. Y si se puede aceptar un tiempo de propagación 1% mayor, es posible encontrar un path que optimice a 50% el área.

Sugerencias Para Trabajos Futuros

El Algoritmo heurístico utilizado es paralelizable dada la naturaleza independiente de cada simulación, lo cual permite ganar speed-up.
Queda considerar este modelo también es fácilmente adaptable a rutas más complejas, siempre y cuando sea demostrable que la optimización converge a mínimos globales.

Referencias

[S04]Sachin Sapatnekar. Timing, Capítulo 8.4, 2004 Kluwer Academic Publishers, ISBN 1-4020-8022-0

[SEC13] Stanislaw H. Zak, Edwin K. P. Chong Publisher: An Introduction to Optimization, 4th Edition. John Wiley & Sons Release Date: January 2013 ISBN: 9781118279014, cap 22.3

[WPFACT] Jan M. Rabaey. 1996. *Digital Integrated Circuits: A Design Perspective*. Prentice-Hall, Inc, cap 5.4.3

[CALIB] Ivan Sutherland, Bob Sproull, and David Harris. 1999. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers Inc, cap 5.1

https://github.com/FabriceSalvaire/PySpice/tree/master

http://ngspice.sourceforge.net/

Código del proyecto : https://github.com/andrewparlane/fiuba_6633lsd_tpfinal.git