Árboles de distribución de reloj - Clock tree

22 de noviembre de 2024

1. Introducción

La señal de reloj dentro de un circuito sincrónico, es la señal que mayor fanout (carga capacitiva) posee ya que alimenta todos los pines CLK de todos DFFs del diseño (esto también sucede con la señal de reset). Es a su vez, es la señal que mayor actividad de conmutación posee.

Solución: un árbol de buffers que distribuya la carga que imponen los flip flops.

Se puede decir entonces que el clock tree es el encargado de suministrar la señal de clock a todos los flip-flops del circuito haciendo cumplir las restricciones de temporización para cada uno de ellos:

$$\begin{array}{rcll} (T_{clk} + ETCC_i) & - & (LTLC_{i-1} + LTAD_i + t_{su_i} + U_{su_i}) \geq 0 & \text{(Setup time analysis)} \\ (ETLC_{i-1} + ETAD_i) & - & (LTCC_i + t_{h_i} + U_{h_i}) \geq 0 & \text{(Hold time analysis)} \end{array}$$

$$\begin{array}{r} i = 1, 2, \ldots, N \\ i, i-1 \text{ son DFFs consecutivos.} \end{array} \tag{1}$$

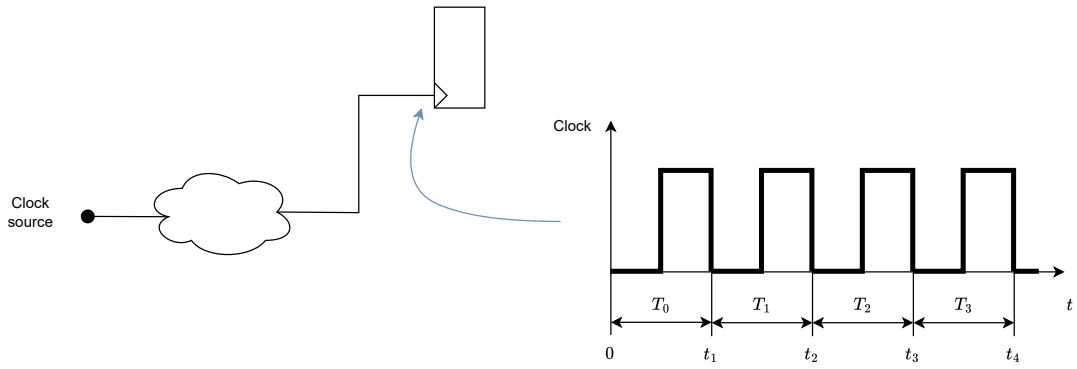
donde

- N: la cantidad de DFFs del diseño.
- $LTLC_i$: late time of launching clock at $(i-1)$-DFF.
- $LTAD_i$: late time of arriving data at i-DFF.
- $ETCC_i$: early time of capturing clock at i-DFF.
- $ETLC_i$: early time of launching clock at $(i-1)$-DFF.
- $ETAD_i$: early time of arriving data at i-DFF.
- $LTCC_i$: late time of capturing clock at i-DFF.
- t_{su_i}: i-DFF setup time.
- t_{h_i}: i-DFF hold time.
- U_{su_i}: incertidumbre (o margen de seguridad) para el análisis de setup time en el i-DFF.
- U_{su_i}: incertidumbre (o margen de seguridad) para el análisis de hold time en el i-DFF.

LLamamos entonces *clock schedule* al conjunto de inecuaciones 1 para todos los DFF del diseño. LLamamos clock tree síntesis al proceso (automático o no) de obtener una implementación del árbol de reloj que cumple con el clock schedule.

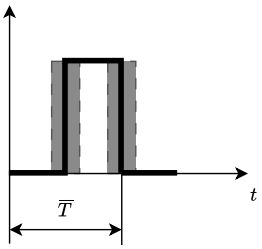
2. Clock jitter

Definición: Jitter es la variación aleatoria del instante de arribo del flanco de reloj con respecto al instante esperado.



En general, $T_i \neq T_j$.

Si se solapan los períodos de reloj se obtiene el siguiente gráfico:



Definición: El período de reloj $Tclk$ se define como el promedio de los T_i.

$$T_{clk} = \overline{T} = E[T_i]$$

Definición: *Phase jitter* es la diferencia entre el instante actual de arribo del flanco del reloj y el instante esperado.

$$\phi_n = t_n - nT_{clk}$$

Definición: *Period jitter* es la diferencia entre el período actual y el período de reloj ideal.

$$\phi'_n = (t_n - t_{n-1}) - T_{clk} = T_n - T_{clk} = \phi_n - \phi_{n-1}$$

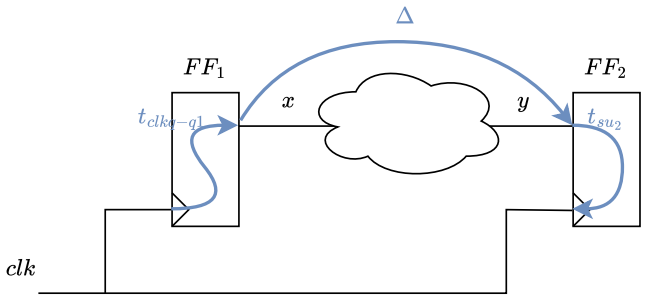
Definición: *Cycle to cycle jitter* es la diferencia del valor del período entre dos ciclos consecutivos del reloj.

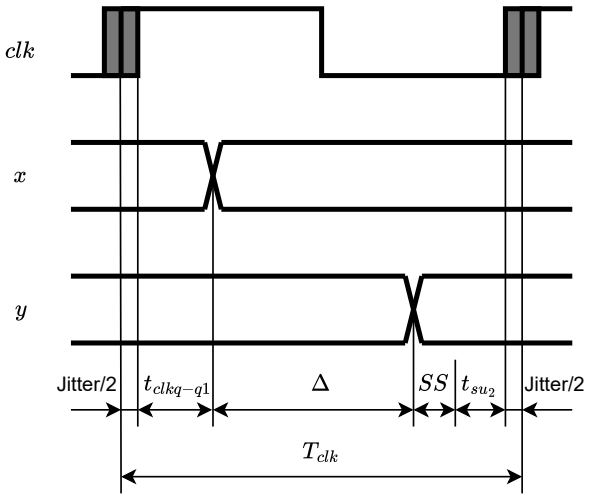
$$\phi''_n = (t_n - t_{n-1}) - (t_(n-1) - t_{n-2}) = T_n - T_{n-1} = \phi'_n - \phi'_{n-1}$$

2.1. Impacto del jitter en el STA

El valor mínimo del period jitter, $\text{mín}_{\forall n \in \mathbb{N}}\{\phi'_n\}$ es el valor que nos interesa para elegir el valor mínimo de período de reloj para que el circuito sincrónico funcione en el cálculo del setup slack.

Por otro lado, el jitter de la fuente no influye en el análisis del hold time ya que el mismo depende sólo del flanco actual.





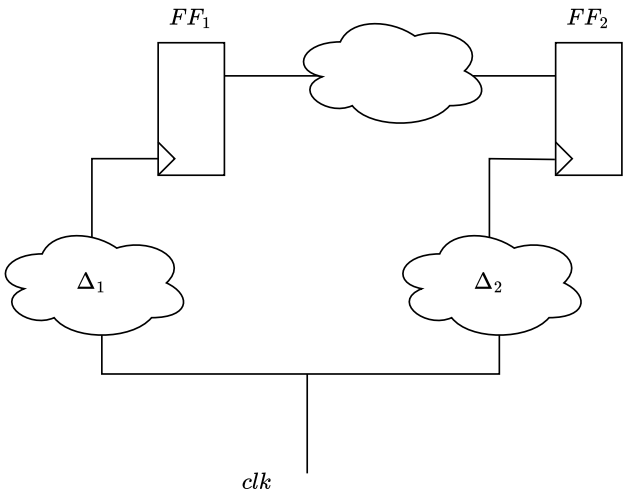
$$Setup\_slack = T_{clk} - (t_{clk-q1} + \Delta + t_{su_2} + Jitter)$$

Además, el jitter afecta el ancho de pulso de reloj, por lo cual, el ancho efectivo de los pulsos de reloj en el peor caso de jitter debe ser más ancho que el ancho mínimo soportado por los DFFs.

3. Clock skew

Definición: *Skew* es la diferencia del instante de arrivo del flanco activo de reloj a dos DFFs distintos.

$$Skew = \Delta_2 - \Delta_1$$



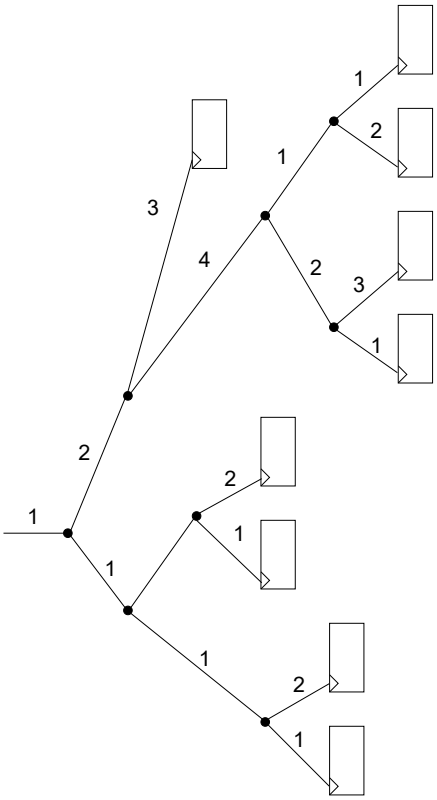
La ec. (1) puede reescribirse en términos del skew:

$$\begin{aligned} Skew_{min} &= ETCC_i - LTLC_{i-1} \\ Skew_{max} &= LTCC_i - ETLC_{i-1} \end{aligned}$$

$$\begin{aligned} (T_{clk} + Skew_{min}) &- (LTAD_i + t_{su_i} + U_{su_i}) \geq 0 \quad \text{(Setup time analysis)} \\ (ETAD_i) &- (Skew_{max} + t_{h_i} + U_{h_i}) \geq 0 \quad \text{(Hold time analysis)} \end{aligned}$$

$$\begin{aligned} i &= 1, 2, \ldots, N \\ i, i-1 &\text{ son DFFs consecutivos.} \end{aligned} \tag{2}$$

Definición: Un timing tree (TT) corresponde a un árbol donde la raiz es la fuente de reloj, las hojas son flip flops y el valor de cada rama es un número positivo que corresonponde al retardo de propagación entre dos nodos del árbol.



Definición: Un deterministic timing tree (DTT) es un (TT) donde el retardo de cada rama es constante en el tiempo.

Definición: El path delay dp_i de un DTT es la sua de todos los branch delays comprendidos entre la raiz y cada una de las hojas (flip-flops del diseño).

Definición:

- El retardo mínimo de un DTT corresponde al valor mínimo de los dp_i,

$$d_{min} = \min_{i=1,2,\ldots,N}\{dp_i\}$$

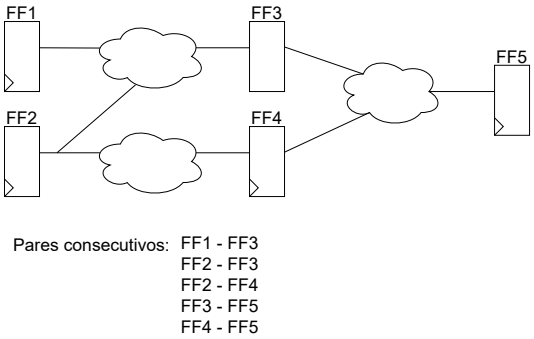
- El retardo máximo de un DTT corresponde al valor máximo de los dp_i,

$$d_{min} = \max_{i=1,2,\ldots,N}\{dp_i\}$$

- El retardo de inserción del cock tree corresponde a

$$\Delta_{DTT} = \frac{d_{min} + d_{max}}{2}$$

Definición: Se dice que dos flip-flops son consecutivos cuando existe un camino lógico entre ambos.



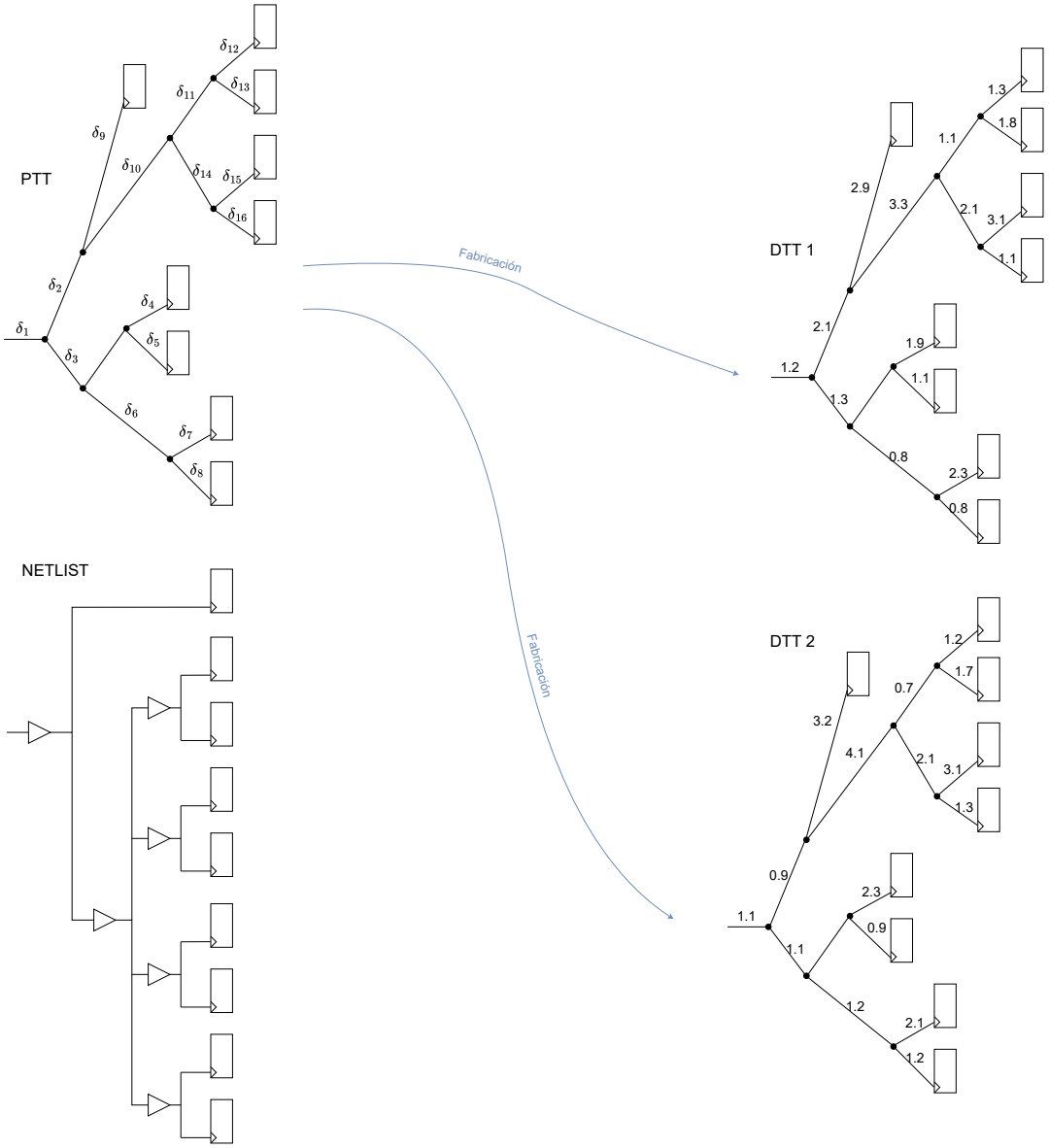
Definición: *Global skew* (S_G) es la diferencia máxima entre dp_{min} y dp_{max}.

$$S_G = d_{max} - d_{min}$$

Definición: *Local skew* (S_L) es la diferencia máxima entre dp_i y dp_j cuando los flip-flops i, j son consecutivos.

$$S_L = \max_{i,j \text{consecutivos}} |dp_i - dp_j|$$

Variaciones en el proceso de fabricación pueden causar incertidumbre en los retardos de cada una de las ramas de un clock tree. Dichos retardos pueden ser considerados constantes (la variación aleatoria ciclo a ciclo se asigna a jitter) para cada circuito fabricado. Por lo cual, podemos decir que el circuito fabricado corresponde a una realización o muestra de un probabilistic timing tree (PTT).



Se supondrá que el retardo asociado a cada rama corresponde a una variable aleatoria con cierta distribución asociada. Esta variación en el retardo se debe a variaciones de los transistores de los buffers del clock tree y variaciones de los parásitos y resistencias de conexionado. Para obtener una estimación a priori del skew se puede suponer que cada branch delay (δ_i) tiene una distribución gaussiana $N(\mu_i, \sigma_i)$ (lo cual no es cierto ya que el retardo no puede ser negativo, pero puede ser una buena aproximación inicial al problema) descorrelacionadas entre sí.

Para este caso, el path delay será

$$dp_i \sim N\left(\sum_k \mu_k, \sqrt{\sum_k \sigma_k^2}\right)$$

Mientras que para el global skew se tendrá una distribución

$$dp_i - dp_j \sim N\left(\sum_{k_i} \mu_{k_i} - \sum_{k_j} \mu_{k_j}, \sqrt{\sum_{k_i} \sigma_{k_i}^2 + \sum_{k_j} \sigma_{k_j}^2}\right)$$

Finalmente, para estimar el valor máximo $max|dp_i - dp_j|$ con un cierto nivel de confianza se puede recurrir a una simulación montecarlo.

Este método, sin embargo es muy pesimista al suponer que que todos los retardos están descorrelacionados. En la práctica, los δ_i sí están fuertemente correlacionados (es casi imposible que si un buffer del clock tree está en las mejores condiciones PVT, otro esté en las peores).

3.1. Correlación espacial

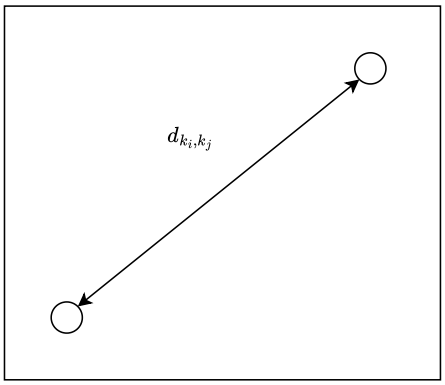
Distintos modelos de correlación se pueden utilizar para determinar el valor de $dp_i - dp_j$.

Obviamente, hay dos modelos extremos de correlación: si la correlación es 1 para todos k_i, k_j, siendo i, j consecutivos, entonces nos econtramos en presencia del método BC-WC. Por el contrario, si la correlación es -1 para todos k_i, k_j, siendo i, j consecutivos, entonces nos encontramos en presenciadel método OCV.

Si se quieren utilizar modelos intermedios entre BC-WC y OCV, un modelo de correlación espacial puede utilizarse:

$$\rho_{k_i,k_j} = \rho(d(k_i, k_j))$$

Obviamente, $\rho(0) = 1$ ya que si $d = 0$ entonces $i = j$ (mismo nodo en el PTT).



Algunos modelos de correlación comúnmente usados son:

- Constante,

$$\rho(d) = \begin{cases} 1 & d = 0 \\ \alpha & d > 0 \end{cases} \tag{3}$$

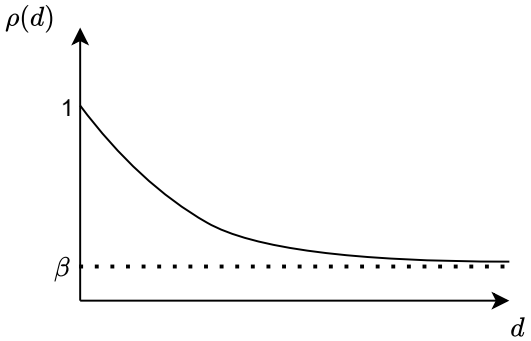
  donde $0 \leq \alpha < 1$

- Lineal,

$$\rho(d) = 1 - \alpha d \tag{4}$$

- Exponencial, puede haber dos modelos con uno o dos grados de libertad,

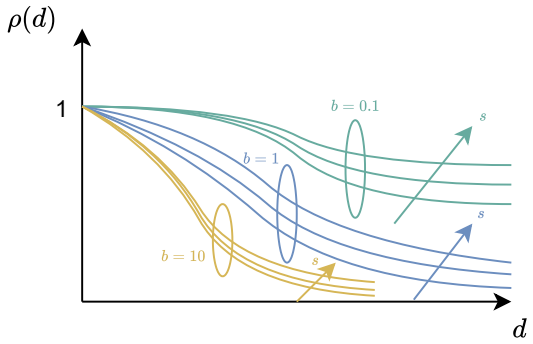
$$\rho(d) = e^{-\alpha d}, \quad \rho(d) = (1 - \beta)e^{-\alpha d} + \beta \tag{5}$$



- Bessel,

$$\rho(d) = 2\left(\frac{bd}{2}\right)^{s-1} K_{s-1}(bd)\Gamma^{-1}(d-1) \tag{6}$$

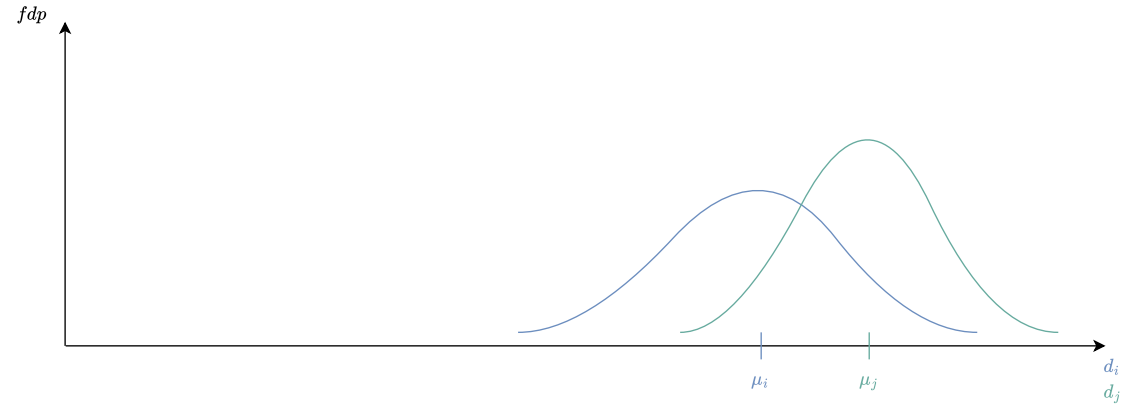
donde K es la función de Bessel modificada de segundo tipo, Γ es la función Gamma.

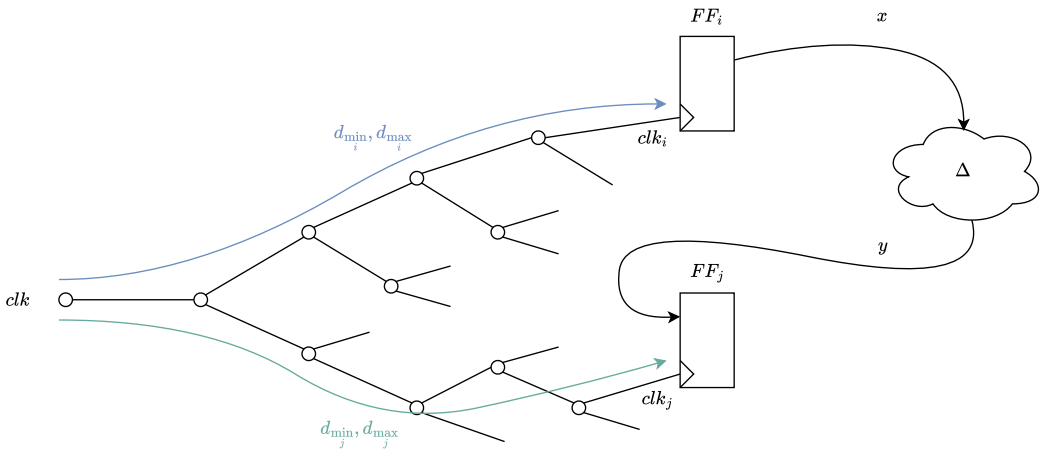


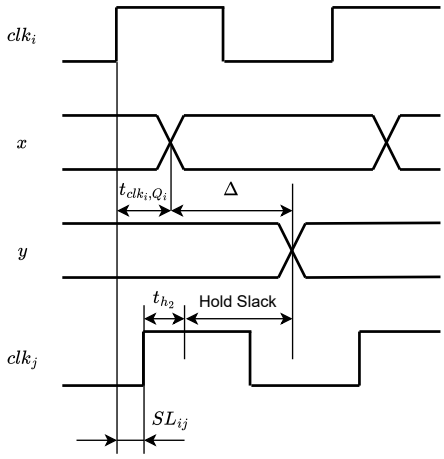
4. Estimación del peor caso de skew

Se puede suponer que dos path consecutivos tienen d_{max} y d_min.

$$SL_{ij} = \text{máx}\left\{|d_{max_i} - d_{min_j}|, |d_{max_j} - d_{min_i}|\right\}$$







$$\text{Hold Slack} = (t_{clk_i,Qi} + \Delta) - (t_{h_2} + SL_{ij}) \geq 0$$

Si el timing path i, j es robusto a un skew $SL_{i,j}$ máximo, entonces este valor limita la profundidad del clock tree que puede rutearse:

$$SL_{ij}max = t_{clk_i,Q_i} + \Delta_{ij} - t_{H_2}$$

Observar que en un diseño de alta performance, siempre además se buscará que Δ_{ij} sea lo menor posible, lo que limita aún más el valor máximo que puede tomar SL_{ij}.

Por lo tanto, si la incertidumbre es mayor al SL_{ij} max aceptable, entonces el clock tree no puede asegurar el correcto funcionamiento del circuito.

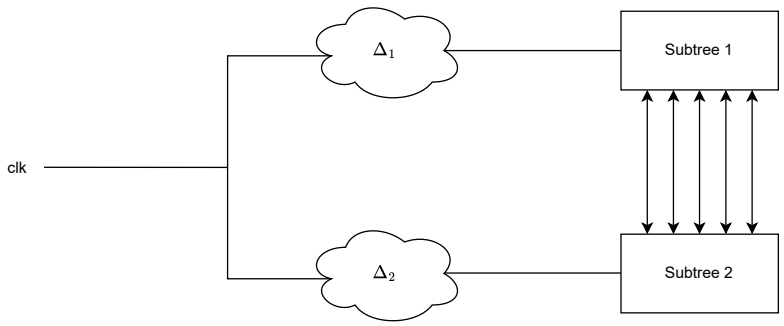
Observar que con un layout delicado, se puede cómo mucho lograr que $fdp_i = fdp_j$, lo cual puede en promedio reducir el skew, y por lo tanto reducir si incertidumbre pero si se sigue aumentando el tamaño del clock tree en algún momento por más delicado que se logre hacer el layout, se encontrará de todas formas el límite que impone la incertidumbre del skew.

Solución: Dividir el árbol de reloj en subárboles de menor tamaño de forma de reducir la incertidumbre en cada uno de ellos y balancear los mismos por algún método (post-Si delay tunning).

Pregunta para pensar: Por qué no implementar sincronización entre los subárboles como si fuesen distintos dominios de reloj?

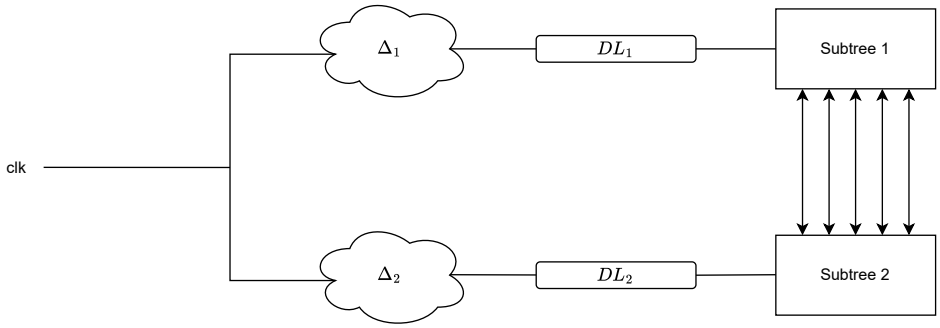
5. Post Silicon Delay Tunning

Sea el siguiente escenario:



Todos los DFFs dentro de cada subtree sufren de un skew acotado que no limita el funcionamiento del circuito ya que la división en subtrees se hizo con el objetivo de limitar la incertidumbre del skew dentro del subtree de forma tal que el subtree opere correctamente.

Por otro lada, cada retardo Δ_1, Δ_2 puede hacerse con un layout delicado que maximice la posibilidad de que las fdp_1 y fdp_2 sean similares. Obviamente, los paths más críticos frente a skew son aquellos que se comunican entre ambos subtrees. Para reducir el skew en dichos paths, una solución es modificar el esquema anterior de la siguiente forma:

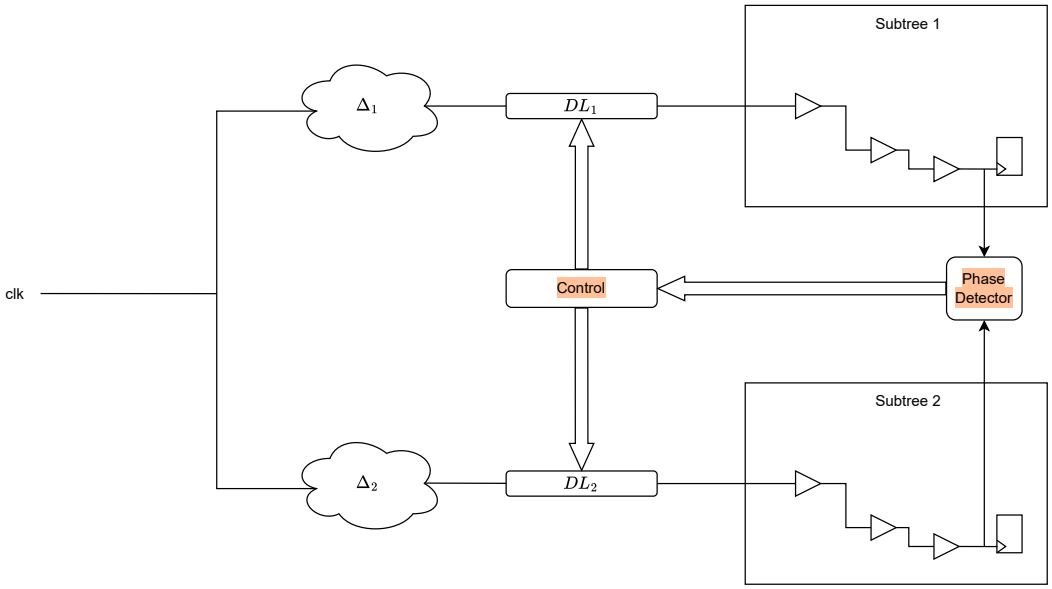


donde DL significa *Delay Line*.

El nuevo enfoque opera de la siguiente forma:

- Si $\Delta_1 > \Delta_2$ entonces la DL_2 aumenta su valor de forma de lograr que $\Delta_2 + DL_2 = \Delta_1$ mientras que $DL_1 = 0$
- Si $\Delta_1 < \Delta_2$ entonces la DL_1 aumenta su valor de forma de lograr que $\Delta_1 + DL_1 = \Delta_2$ mientras que $DL_2 = 0$

El esquema se completa sumando un detector de fase y lógica de control:

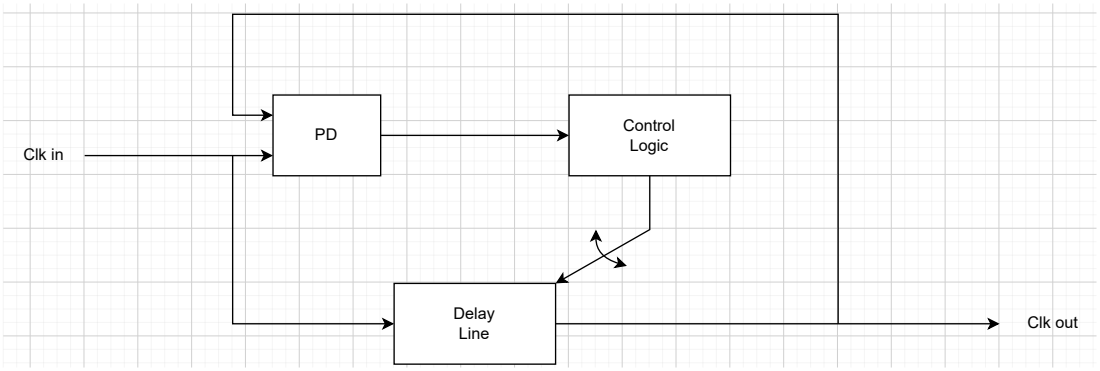


Nota: Pueden hacerse enfoques alternativos para superar el límite que impone la incertidumbre del skew en los paths entre subtrees. Algunos enfoques son del estilo de por ejemplo trabajar esos paths como multicycle, de forma tal que la señal quede congelada durante los períodos de reloj necesarios para luego en el capturing subtree samplear esos datos. Obviamente, esto toma varios períodos de reloj y la performance en cuanto a la velocidad de procesamiento de datos se ve reducida, pero no es necesario implementar un enfoque de deskwing como el mencionado.

Nota: Otro enfoque de deskwing diferente al de utilizar un detector de phase (PD) es el de tener trimming fijo, es decir que cada delay line se trimmea de forma fija (puede ser por medio de algunos bits de EEPROM por ejemplo). Para ello hay que utilizar algún mecanismo de medición de skew on chip que pueda leerse externamente a fin de programar dicha memoria de trimming.

6. Delay Locked Loop

Clock out se engancha a la misma phase que tiene clock in.

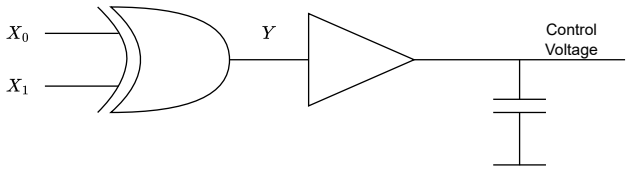


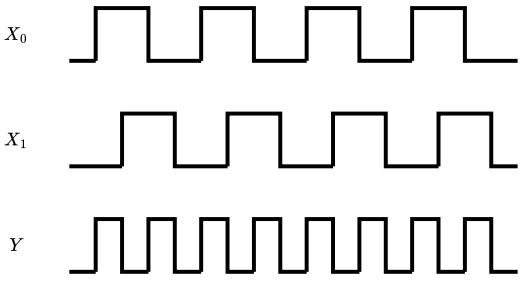
6.1. Phase detector (PD)

6.1.1. XOR phase detector

Desventaja: 180° de incertidumbre, lo cual duplica la DL.

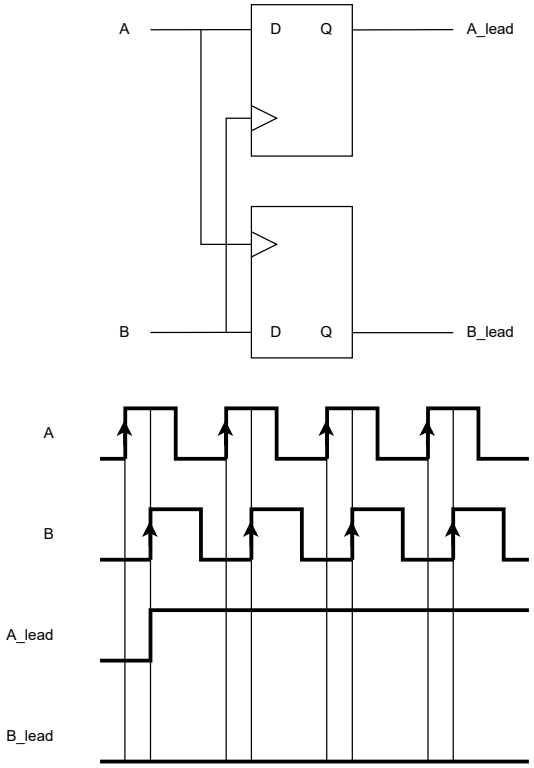
X_0	X_1	Y
0	0	0
0	1	1
1	0	1
1	1	0



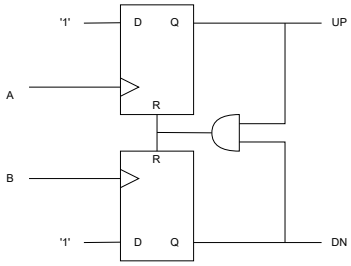


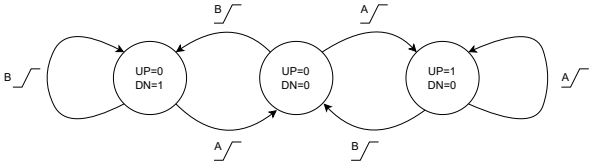
6.1.2. Bang-bang detector

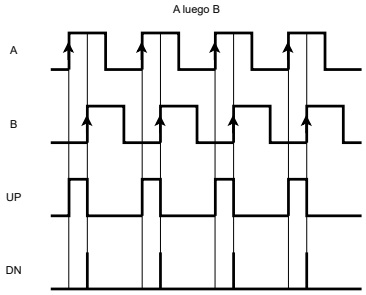
Desventaja: Metaestabilidad, lo cual genera incertidumbre en la detección de la diferencia de fase (y por lo tanto de corrección skew) de $t_{su} + t_h$.

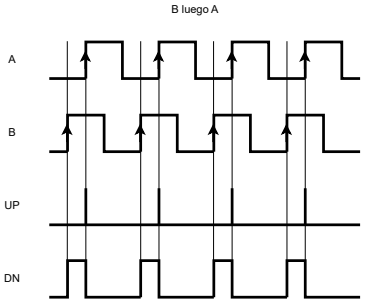


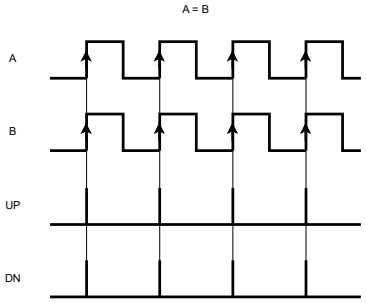
6.1.3. Phase-frequency detector





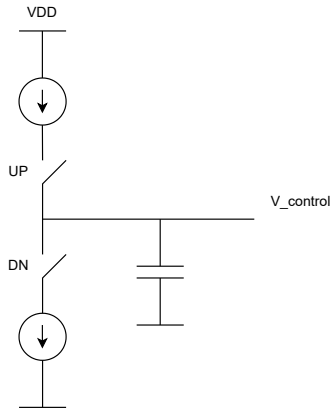




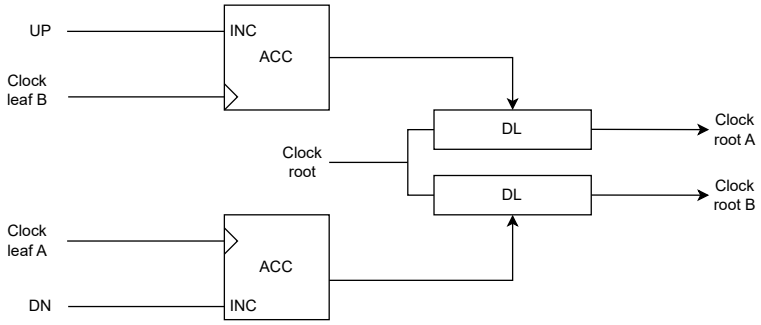


6.2. Control

6.2.1. Analógico

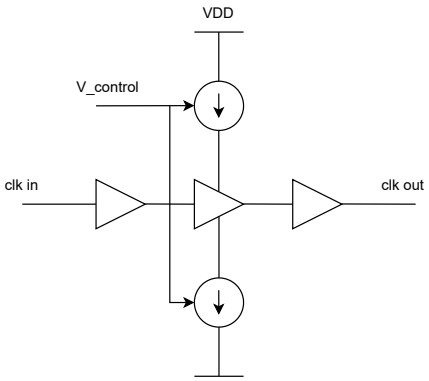


6.2.2. Digital

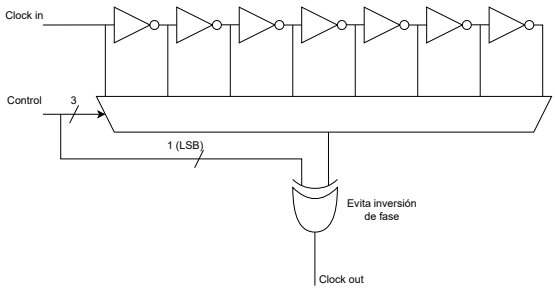


6.3. Delay Line

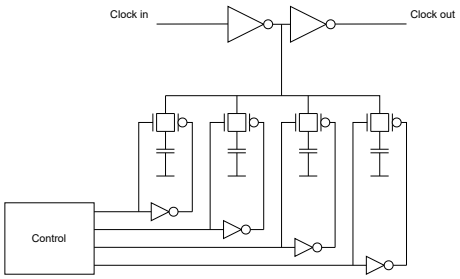
6.3.1. Analógico



6.3.2. Digital



DL con inversores



DL con capacitores

7. Clock tree synthesis

Clock tree synthesis es el proceso de obtener (una vez que el diseño lógico está físicamente dispuesto) un árbol de distribución de reloj tal que cumpla las siguientes restricciones:

- Mantener acotado el fanout (capacidad de salida) en cada hoja (buffer) del árbol.

$$C_{L_i} \leq C_{max}$$

- Mantener acotado el slew en cada hoja (buffer) del árbol.

$$t_{rise} \leq t_{rise_{max}}$$

$$t_{fall} \leq t_{fall_{max}}$$

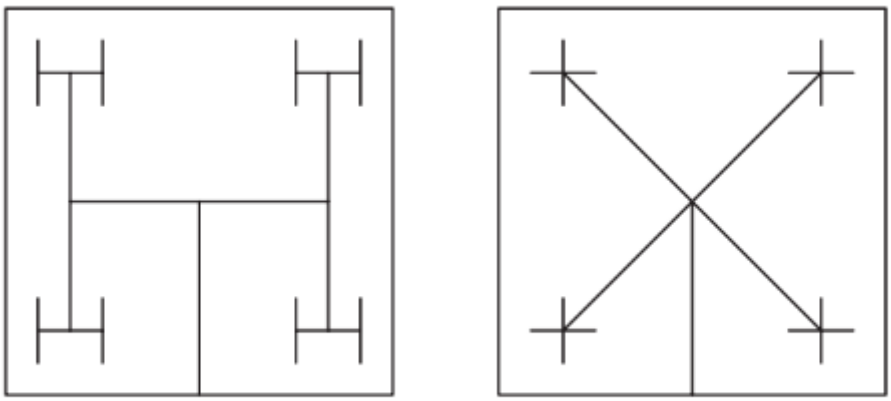
- Cumplir con las restricciones temporales en cada path R2R (eq. 1, eq. 2).

Para lograr esto se puede recurrir a dos enfoques: clock tree estructurado o clock tree no estructurado.

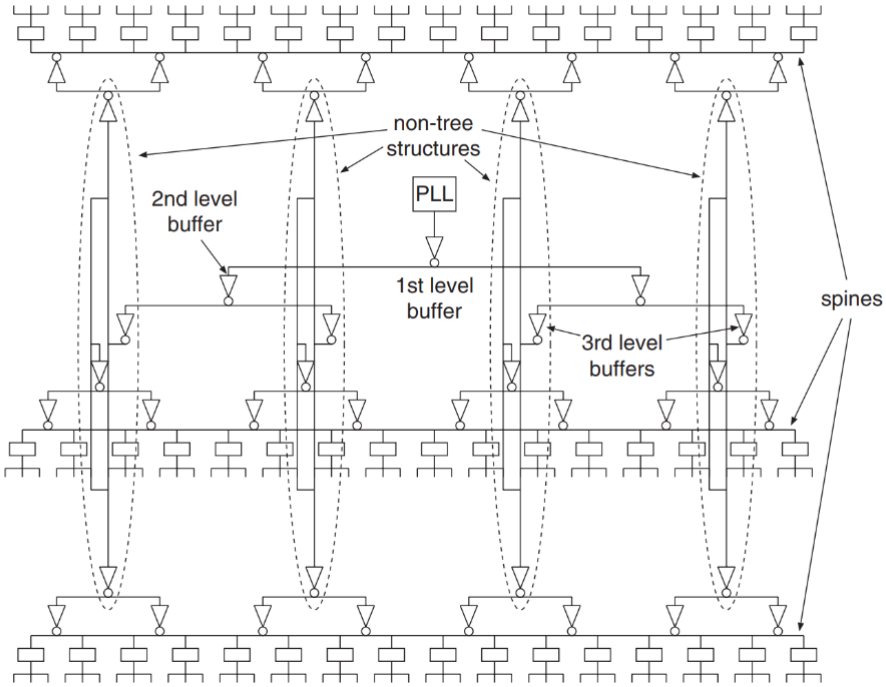
7.1. Clock tree estructurado

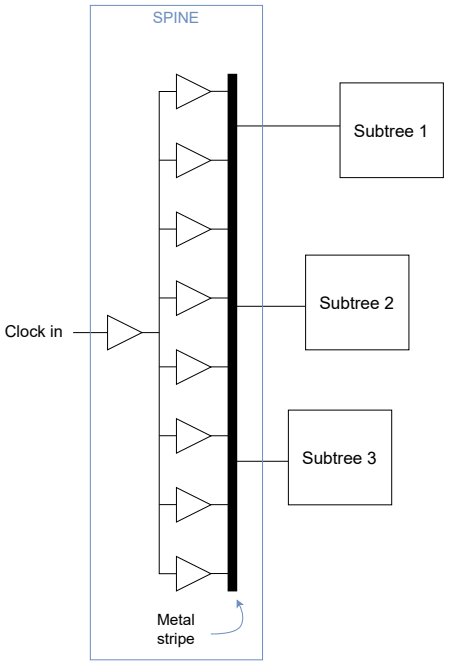
En la figura siguiente se observa dos estructuras (H o X) que logran mismo retardo (en promedio, sin considerar variaciones PVT) en todas las hojas del árbol.

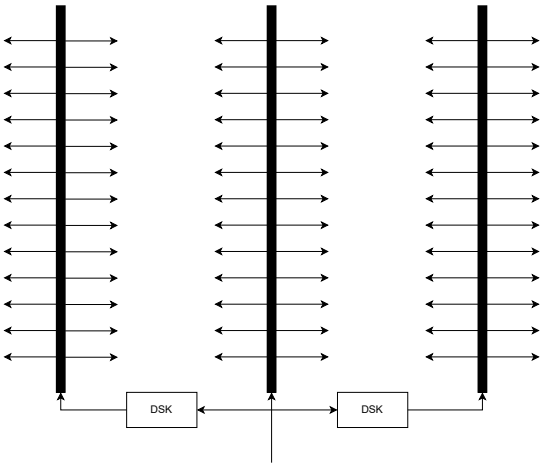
Sin embargo, la principal limitación que poseen estas estructuras (más allá de la limitación de la incertidumbre PVT que la tienen todas las estructuras y siempre limita la profundidad del árbol) es la necesidad de un layout regular a lo largo de todo el die.



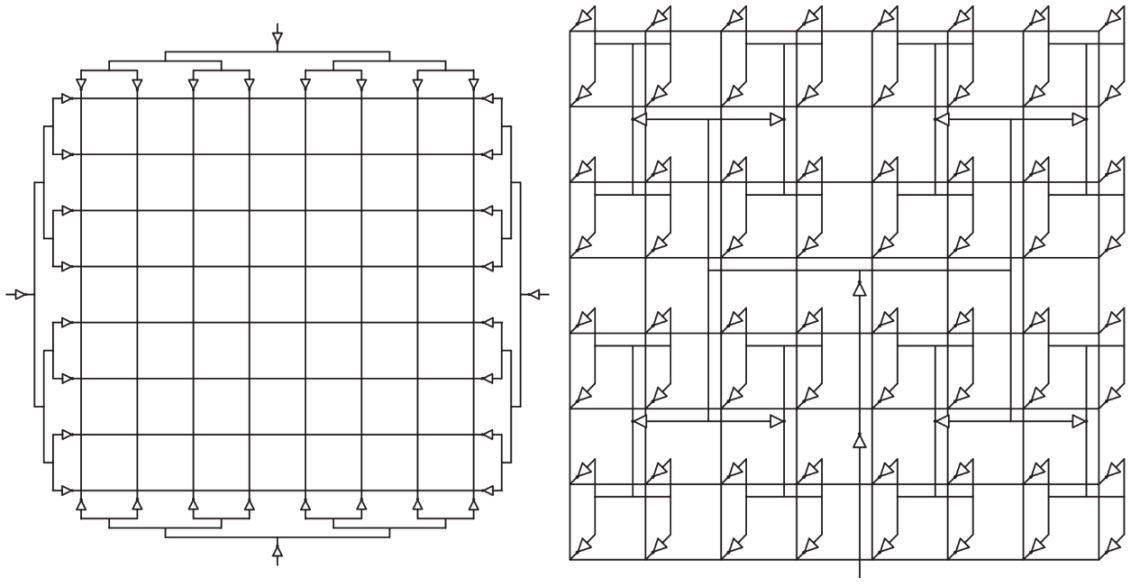
Un enoque utilizado, cuando no hay una estructura tan regular del layout es el concepto de *spine* (Pentium 4).



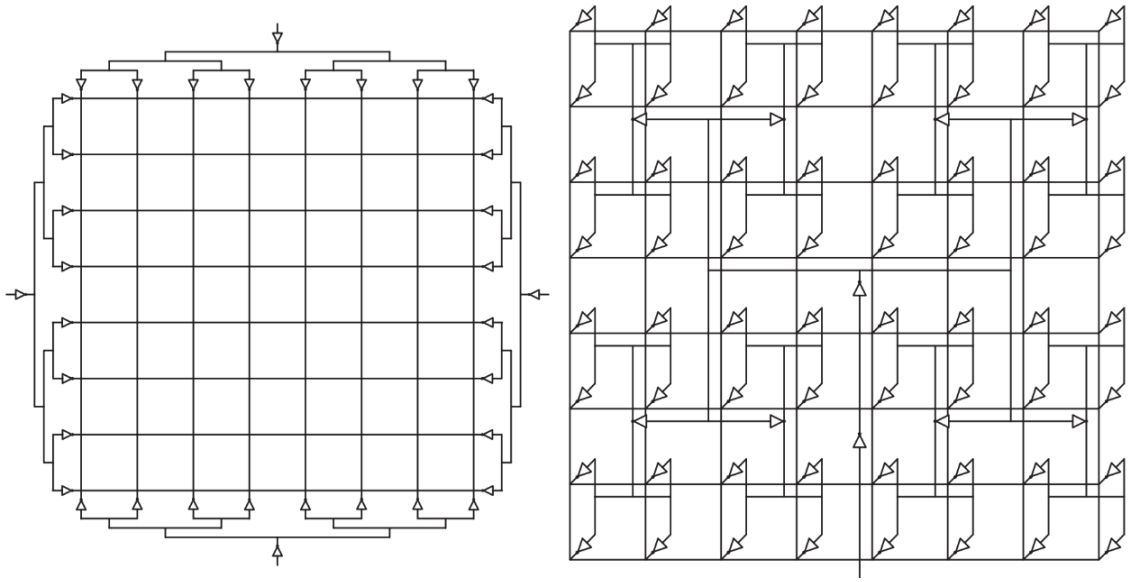




DSK: Deskewing.

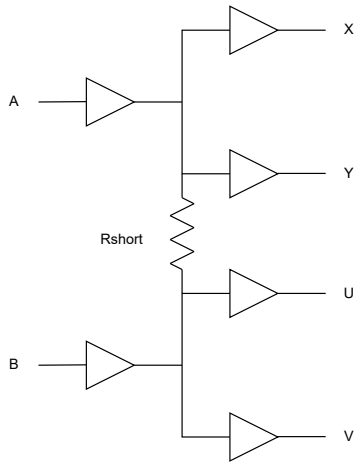


Clock grid



Distribución híbrida: Tree + Mesh.

7.2. Efecto del shorting (Grid/Spine)



Supongamos que:

$$Skew(A, B) = |T_{clk_A} - T_{clk_B}|$$

$$|T_{clk_A} - T_{clk_X}| = |T_{clk_A} - T_{clk_Y}|$$

y también que

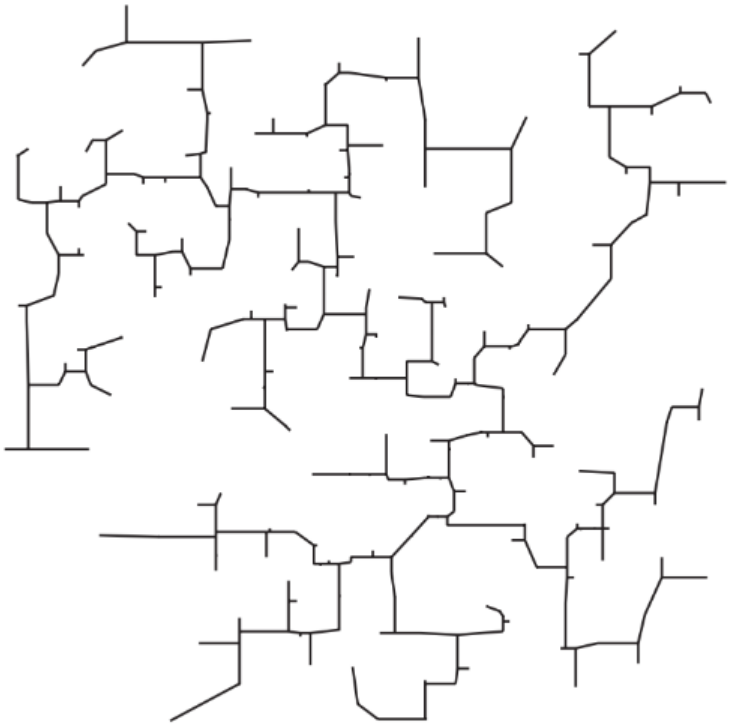
$$|T_{clk_B} - T_{clk_U}| = |T_{clk_B} - T_{clk_Y}|$$

- Si $R_{short} = 0$, entonces $Skew(Y, U) = 0$.
- Si $R_{short} = \infty$, entonces $Skew(X, U) = Skew(X, V) = Skew(Y, U) = Skew(Y, V) = Skew(A, B)$.
- Si $0 < R_{short} < \infty$, entonces $Skew(Y, U) < Skew(A, B)$.

Observación: El shorting si bien reduce el skew, tiene la contraparte de aumentar la corriente de corto circuito durante el toggling del clock.

7.3. Clock tree no estructurado

Generalmente, la distribución local dentro de un bloque se hace con un árbol no estructurado, automáticamente sintetizado por la herramienta de PnR.



Observando la eq. (2), se puede deducir que una forma de resolver el clock scheduling es forzando $Skew_{min} = Skew_{max} = 0$.

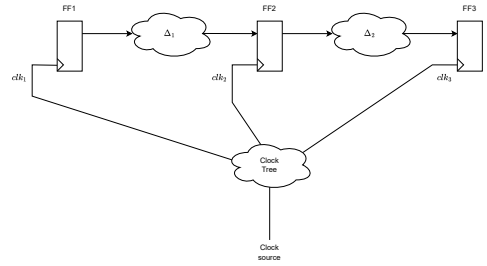
Algunos de los primeros algoritmos desarrollados para la implementación automática de árboles de reloj son los algoritmos zero-skew routing (ZSR) y el deferred-merge embedding (DME). Dada una topología abstracta del arból de reloj y una disposición geométrica de las hojas (DFFs), estos algoritmos logran el ruteo del árbol de forma tal de lograr cero skew.

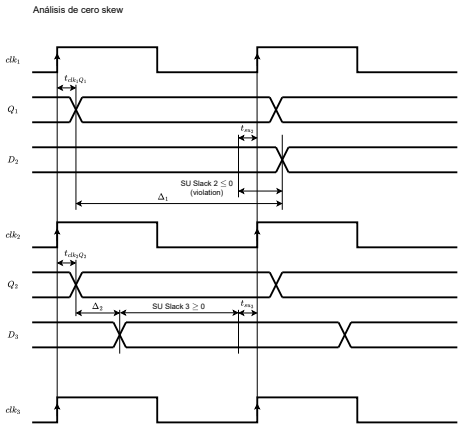
7.4. *Useful skew*

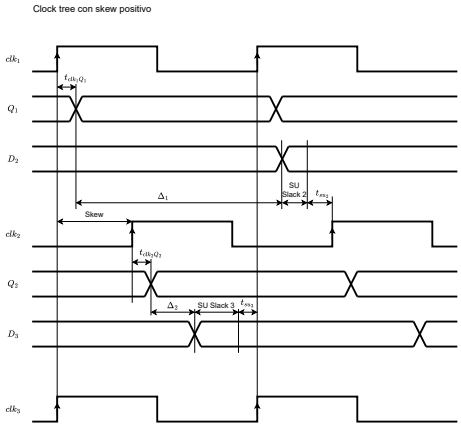
Distintos nombres tiene esta técnica (*useful skew*, *cycle stealing*, árbol de skew positivo), mediante la cual, a diferencia de los algoritmos de cero skew, intentan sintetizar el clock tree de forma de aprovechar el skew para compensar la falta de slack.

Obviamente, cuando se busca un diseño de máxima frecuencia de operación (alta performance), los retardo combinacionales deben hacerse de valores similares ($\Delta_1 = \Delta_2$) y por lo tanto queda poco margen para aplicar la técnica de useful skew.

Ejemplo:







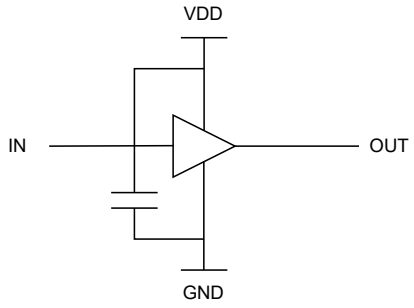
Observar como se reduce el SU Slack 3. Otra opción sería aumentar T_{clk} a un valor

$T_{clk} \geq t_{clk,Q_1} + \Delta_1 + t_{su_2} + U_{su_2}$ manteniendo la condición de cero skew.

8. Clock buffers

En general, buffers en un proceso CMOS se construyen, si bien los PMOS en general son más anchos que los NMOS, de forma no simétrica con el fin de reducir el área. Sin embargo, los buffers que conforman el clock tree tienden a ser lo más simétrico posibles de forma de tener los mismos rise time y fall time. Caso contrario, una cadena de buffers a lo largo del clock tree podría alterar significativamente el duty cycle de la señal de reloj y violar el ancho mínimo de pulso en uno o varios DFFs del diseño limitando la profundidad del clock tree (similar a lo que produce el jitter y el skew).

Otra característica deseada es que también sean más insensibles a variaciones PVT y no aumenten el jitter del clock que ya tiene la fuente del reloj, motivo por el cual es buena práctica también agregarles una capacidad entre VDD-GND.



9. Reset Tree

El reset tree tiene algunas restricciones similares al clock tree y otras significativamente distintas. En cuanto a las similares, se puede decir que el reset tree también es otra señal de máximo fanout (carga capacitiva) ya que alimenta todos los DFFs del circuito por lo cual un árbol de distriución de reset debe ser construido de forma de poder manejar tanta capacidad de carga.

Respecto a las restricciones temporales, estas son similares pero más relajadas respecto de un clock tree debido a que no hay lógica funcional en el camino del reset desde la salida de la compuerta OR hasta la entrada R de cada DFF_i.

$$\begin{aligned} \text{Rec Slack } i &= (T_{clk} + \Delta_{clk_i}) - (\Delta + \Delta_{rst_i} + t_{rec_i} + U_{rec_i}) \geq 0 \\ \text{Rem Slack } i &= (\Delta + \Delta_{rst_i}) - (\Delta_{clk_i} + t_{rem_i} + U_{rem_i}) \geq 0 \end{aligned}$$