# Static Timing Analysis

11 de noviembre de 2024

# 1. Introducción

## 1.1. Qué es Static Timing Analysis (STA)?

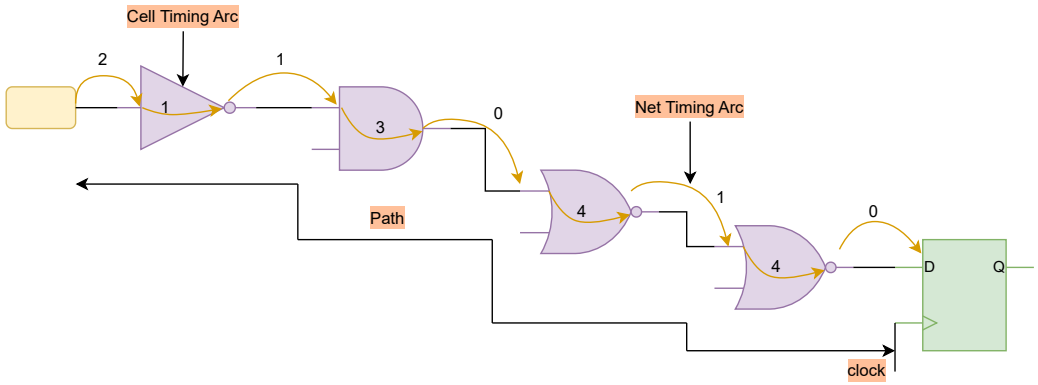



Figura 1: STA: Diseño simple donde se ven los timing arcs y los timing paths

$$\text{Path delay} = 2 + 1 + 1 + 3 + 0 + 4 + 1 + 4 + 0 = 16\text{time units} \tag{1}$$

El análisis estático de tiempos (STA) es el proceso de sumar los retrasos de las celdas y las redes de conexión para obtener los retrasos de las *paths*, y compararlos con las especificaciones de tiempo.

Cuando diseñas un circuito, es crucial asegurarte de que funcione correctamente. Existen varias maneras de lograr esto:

- **Simulación funcional**: Verifica que el circuito funcione como se espera, pero no analiza el rendimiento temporal.
- **Análisis SPICE**: Proporciona resultados precisos para cada trayectoria, pero es ineficiente ya que requiere analizar millones de *paths*.

El STA ofrece una solución eficiente al centrarse únicamente en los retrasos de las celdas y las redes de conexión, sin tener en cuenta marcos temporales específicos. Suma los retrasos obtenidos de bibliotecas o de otras fuentes, calcula los retrasos de los *paths* y los compara con las restricciones temporales definidas en las especificaciones.

## 1.2. Cuál es el objetivo de STA?

Primero, el STA calcula los retrasos de los *paths* que será utilizados por las herramientas de optimización. Luego, basándose en estos retrasos, las herramientas de optimización

seleccionan celdas de la biblioteca de tiempos para crear un circuito que cumpla con los requisitos temporales. Finalmente, el STA analiza los tiempos del circuito para verificar que funcione a la frecuencia especificada.

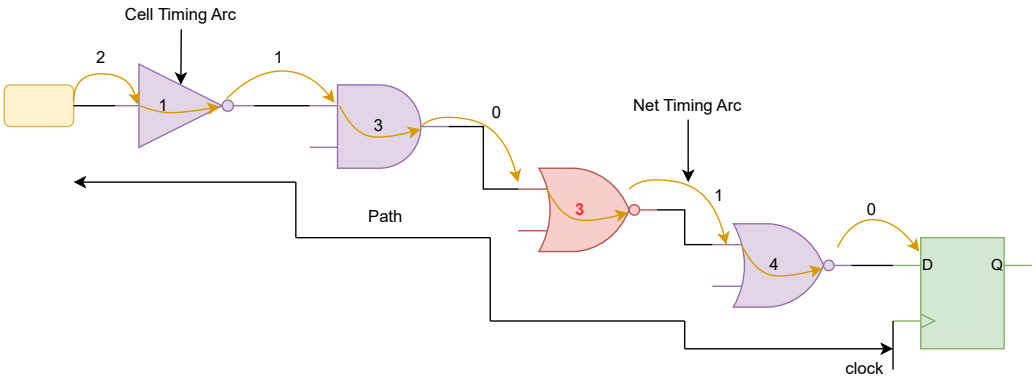
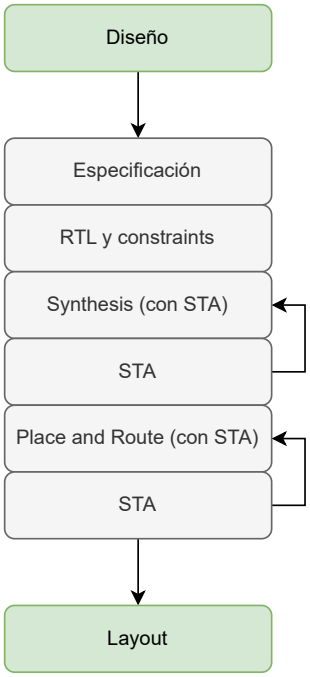


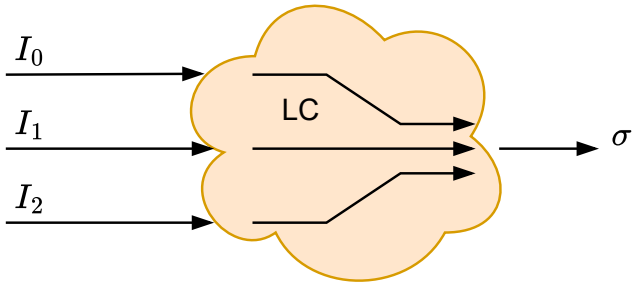

Figura 2: STA: La herramienta de síntesis puede cambiar celdas para cumpler el timing

### 1.3. En qué parte del diseño se hace?

- Se utiliza STA durante la síntesis, y luego se ejecuta después de la síntesis y nuevamente después de Place-and-route (PnR).
- Especificar la funcionalidad del circuito y las bibliotecas de tiempos que se van a utilizar.
- Desarrollar el RTL y escribir restricciones (*constraints*).
- Convertir el RTL a gates (celdas) usando síntesis para cumplir con las restricciones de tiempo.
- Ejecutar STA para verificar los tiempos de los resultados de la síntesis.
- Colocar las celdas en el *layout* y rutear las conexiones. La colocación y ruteo basados en tiempos usan STA para cumplir con los requisitos temporales.
- Ejecutar la verificación final utilizando herramientas de signoff STA.




## Timing Arcs and Markers



**Timing arcs:**

$$
\begin{array}{ll}
I_0 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=00)} & I_1 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=00)} \\
I_0 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=01)} & I_1 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=01)} \\
I_0 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=10)} & I_1 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=10)} \\
I_0 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=11)} & I_1 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=11)} \\
\vdots & \vdots \\
I_0 \downarrow \rightarrow \sigma \,|_{(I_1,I_2=00)} & I_2 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=00)} \\
I_0 \downarrow \rightarrow \sigma \,|_{(I_1,I_2=01)} & I_2 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=01)} \\
I_0 \downarrow \rightarrow \sigma \,|_{(I_1,I_2=10)} & I_2 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=10)} \\
I_0 \downarrow \rightarrow \sigma \,|_{(I_1,I_2=11)} & I_2 \uparrow \rightarrow \sigma \,|_{(I_1,I_2=11)} \\
\vdots &
\end{array}
$$

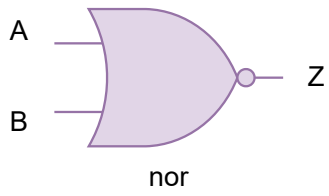
### Definición 1

*Un **timing arc** es positivo unate si una transición de subida en la entrada causa una transición de subida en la salida (o no cambia) y una transición de bajada en la entrada causa una transición de bajada en la salida (o no cambia), independientemente del estado actual de las entradas restantes. En caso inverso, el timing arc es negativo.*

Ejemplos:

- **AND, OR**: positive unate.
- **NAND, NOR**: negative unate.
- **XOR, XNOR**: non unate.

## 1.4. Actividad: Timing Arcs



1. ¿Cuántos timing arcs existen en esta celda NOR?
2. ¿Cuál es la monotonicidad (*unateness*) de los arcos en la celda NOR?

Una compuerta NOR tiene 2 combinaciones entrada-salida (2C1). Cada arco tiene dos transiciones válidas: ascendente (*rise*) y descendente (*fall*). Por lo tanto, hay 4 arcos temporales.

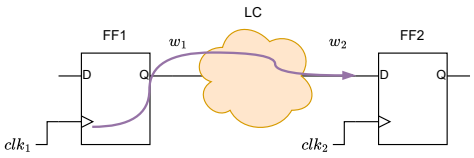
A partir de la tabla de verdad de la compuerta NOR, se observa lo siguiente:

- Cuando cualquiera de las entradas está en transición ascendente, la salida está en transición descendente, y viceversa.
- Por lo tanto, ambos arcos temporales de una compuerta NOR son **monotónicos negativos** (*negative unate*).

## Timing Paths

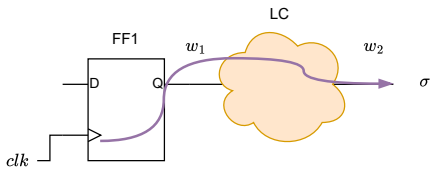
Existen 4 tipos de timing paths:

1. **reg2reg**:



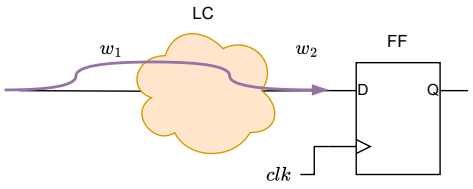

$$t_{path} = tp_{ck_1Q_1} + tp_{w1} + tp_{LC} + tp_{w2} \tag{2}$$

2. **reg2out**:



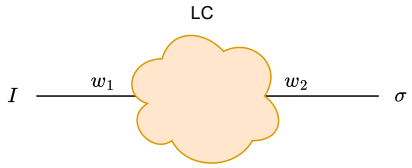

$$t_{path} = tp_{ck_1Q_1} + tp_{w1} + tp_{LC} + tp_{w2} \tag{3}$$

3. **in2reg**:




$$t_{path} = tp_{w1} + tp_{LC} + tp_{w2} \tag{4}$$

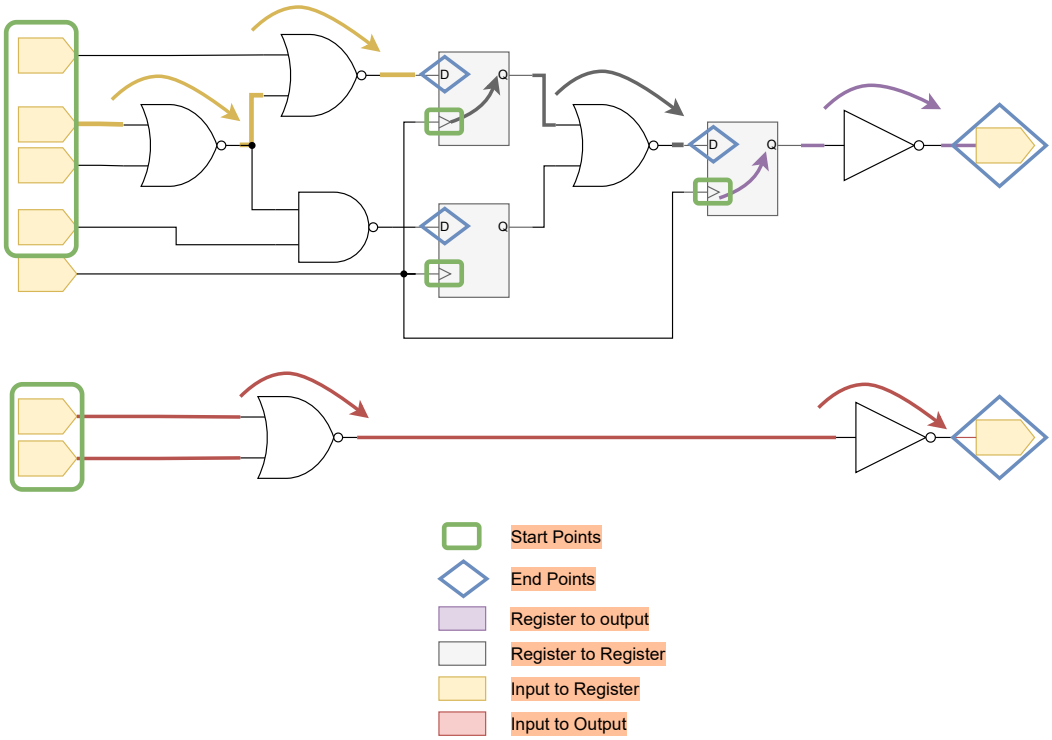
4. **in2out**:




No es realmente timing path ya que no hay clk, sólo puede medirse su retardo.

$$t_{path} = tp_{w1} + tp_{LC} + tp_{w2} \tag{5}$$

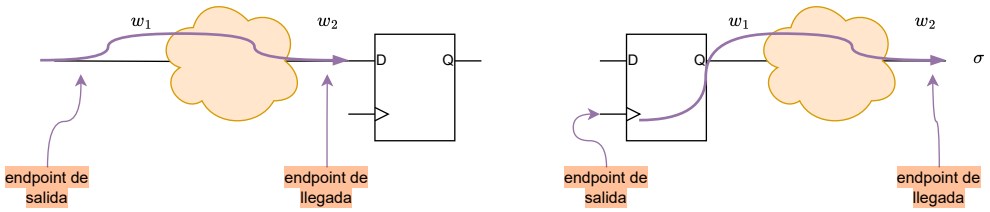
## 1.5. Resumen




**Def. 2:**

- Un **endpoint de salida** es o bien una entrada primaria del circuito o el pin de clk de un FFD.
- Un **endpoint de llegada** es o bien una salida primaria del circuito o el pin D de un FFD.

**Ejemplos:**



## 1.6. Qué es el Slack?

- El **tiempo requerido** (required time) está definido por las restricciones de tiempo, como el período del clk.
- El **tiempo de llegada** (arrival time) es el momento en que la señal llega.
- El **slack** se define como la diferencia entre el tiempo requerido y el tiempo de llegada de una señal en el punto final.

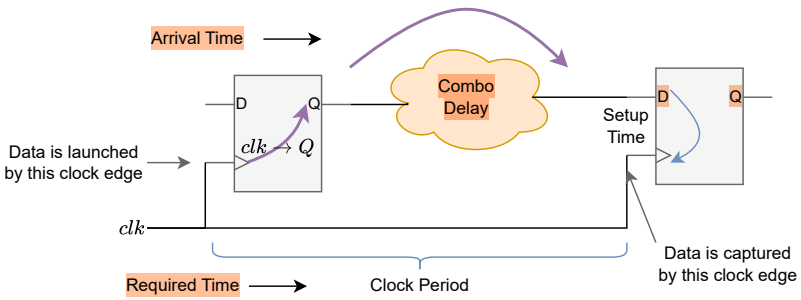
```
Slack = Tiempo Requerido  - Tiempo de Llegada (setup)
Slack = Tiempo de Llegada - Tiempo Requerido  (hold)
```

El motor de síntesis utiliza el valor del *slack* para identificar los *path* que necesitan más optimización para cumplir con los requisitos temporales o para reducir el área.

La ecuación de cálculo del *slack* varía según el tipo de *path*. El *slack* se incluye en todos los informes de tiempos.
Un *slack* positivo indica que el *path* cumple con las restricciones de tiempo, mientras que un *slack* negativo indica que el *path* no cumple con los requisitos.

Un diseño se considera exitoso si todas los *paths* cumplen con las restricciones de tiempo, es decir, si ningun *path* tiene un *slack* negativo.

## 1.7. Register to Register requerimientos setup



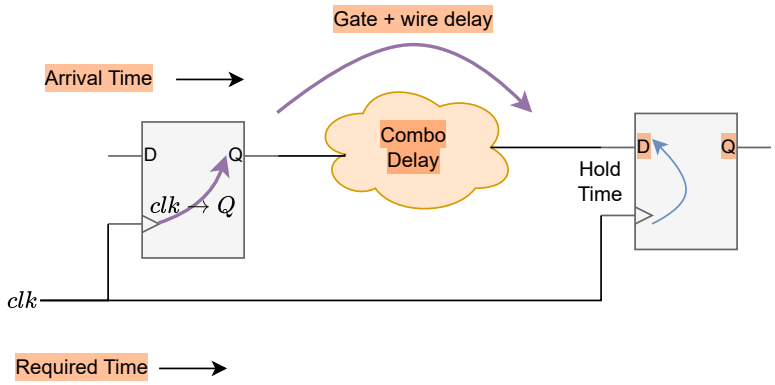

La siguiente ecuación representa el requisito de setup para el *path* de registro a registro:

$$t_{\text{clk\_to\_q}} + t_{\text{combo\_delay}} \leq T_{\text{clock\_period}} - t_{\text{reg\_setup}} \tag{6}$$

Donde:

$$\text{Required Time} = T_{\text{clock\_period}} - t_{\text{setup\_time}}$$
$$\text{Arrival Time} = t_{\text{CK}\rightarrow\text{Q}} + t_{\text{combo\_delay}}$$
$$\text{Slack} = \text{Required Time} - \text{Arrival Time}$$

## 1.8. Register to Register requerimientos hold




La siguiente ecuación representa el requisito de hold para el *path* de registro a registro:

$$t_{\text{clk\_to\_q}} + t_{\text{combo\_delay}} \geq \text{hold\_check}(0) + t_{\text{reg\_hold}} \tag{7}$$

Donde:

$$\text{Required time} = \text{Hold Check } (0) + t_{\text{Hold Time}}$$
$$\text{Arrival time} = t_{\text{CK}\rightarrow\text{Q}} + t_{\text{combo\_delay}}$$
$$\text{Slack} = \text{Arrival time} - \text{Required time}$$

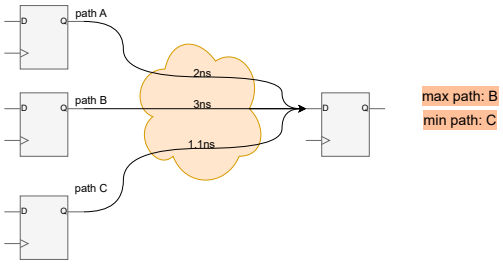
**Nota**: El hold check por defecto es en $t = 0$.

# Min and Max Timing Paths

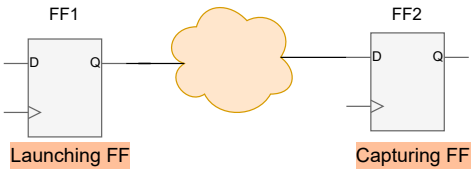
**Def. 3:**

- El **max path** (o late path) entre un endpoint de salida y uno de llegada, es el timing path más lento entre ellos.

- El **min path** (o early path) entre un endpoint de salida y uno de llegada, es el timing path más rápido entre ellos.




**Def. 4:** En un path reg2reg, el flip-flop D que lanza el dato se llama **launching FF** y el que captura el dato se llama **capturing FF**.




## Condiciones de Trabajo de las Celdas Estándar

**Def. 5:** *PVT conditions* son las condiciones de operación a las que está sometida una celda estándar y su apartamiento de la celda promedio del proceso de fabricación.
**PVT:** process-voltage-temperature.

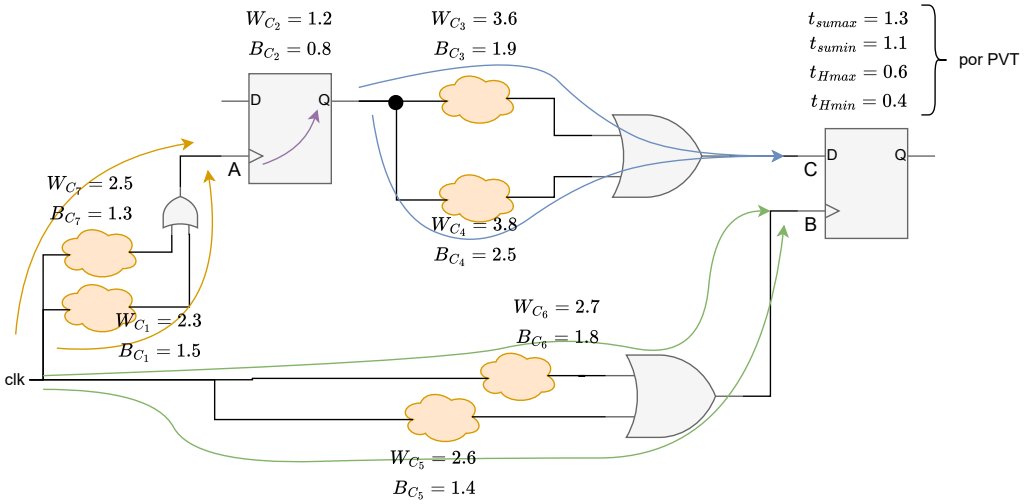
**Def. 6:** Se define como:

- **Worst case (WC):** a la celda más lenta del proceso operando a la máxima temperatura y al menor nivel de voltaje permitido (celdas más lentas que podemos encontrar para un tipo de celda dado).
- **Typical case (TT):** La celda promedio del proceso operando en condiciones típicas (25°C y voltaje nominal).
- **Best case (BC):** La celda más rápida del proceso operando a la mínima temperatura y al máximo voltaje permitido (celdas más rápidas que podemos encontrar para un tipo de celda dado).

**Def. 7:** Se define como:

- **Maximum leakage (ML):** al caso en el que el proceso es rápido, la temperatura es lo más alta posible y la tensión de alimentación es máxima.

# Timing Clocks: Circuitos Sincrónicos




Existen 2 formas de análisis:

1. **BC-WC** (best case - worst case)
2. **OCV** (on chip variation)

## BC-WC Setup Check

*(Solo se deben usar max delays)*

- Late path for launching clock usando **max delays**.
- Late path for arriving data usando **max delays**.
- Early path for capturing clock usando **max delays**.

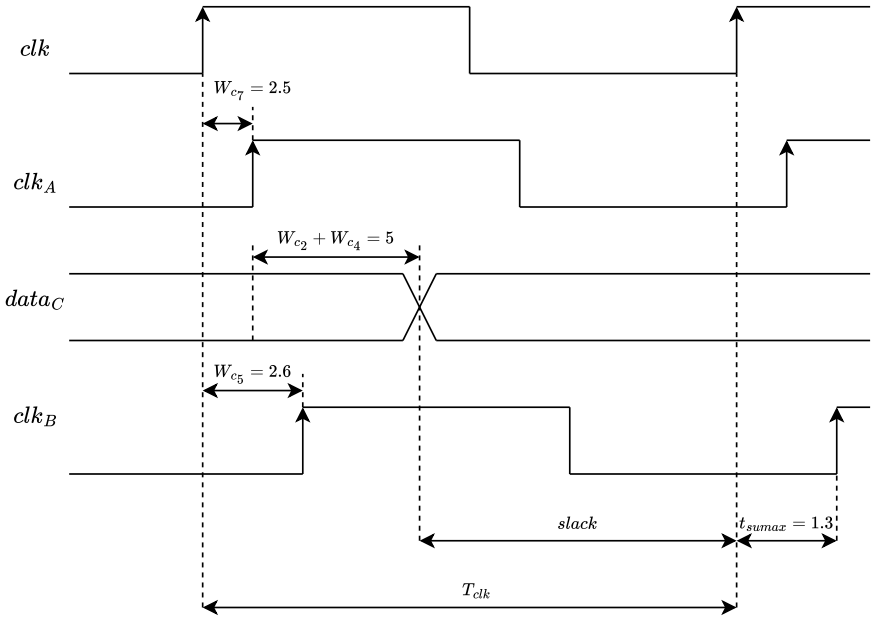
En este caso se tiene:

- Late path launching clock: $2{,}5(WC_7)$
- Late path arriving data: $1{,}2(WC_2) + 3{,}8(WC_4) = 5$
- Early path capturing clock: $2{,}6(WC_5)$
- $t_{\text{sumax}} = 1{,}3$

## Setup Slack Calculation

Luego, el **setup slack** se calcula como:

$$\begin{aligned}\text{Setup slack} &= T_{clk} - \Delta max_{clk_A} - (t_{clkA \to Q_{1_{max}}} + \Delta Q_{d_{max}}) + \Delta max_{clk_B} - t_{\text{sumax}_2} \\ &= T_{clk} - WC_7 - (WC_2 + WC_4) + WC_5 - t_{\text{sumax}_2} \\ &= T_{clk} - 2{,}5 - 1{,}2 - 3{,}8 \\ &= T_{clk} - 6{,}2\end{aligned} \tag{8}$$




Si $slack < 0 \Rightarrow$ *Timing violation.*
**Solución posible:**

- Más rápido la lógica en el data path
- Aumentar el delay entre clock siempre que $WC_5 < WC_7 + (WC_2 + WC_4)$
- Aumentar el período de clock $T_{clk}$

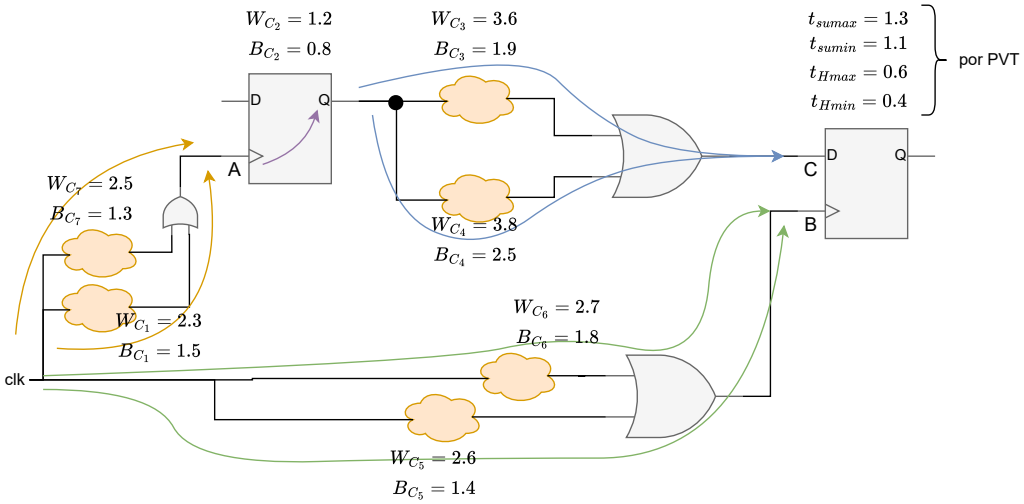
**Setup slack es dependiente de la frecuencia de clk.**

# BC-WC Hold Check

**(Solo se deben usar min delay)**

- **Early path for launching clock** usando **min delays**.
- **Early path for arriving data** usando **min delays**.
- **Late path for capturing clock** usando **min delays**.

Ejemplo:



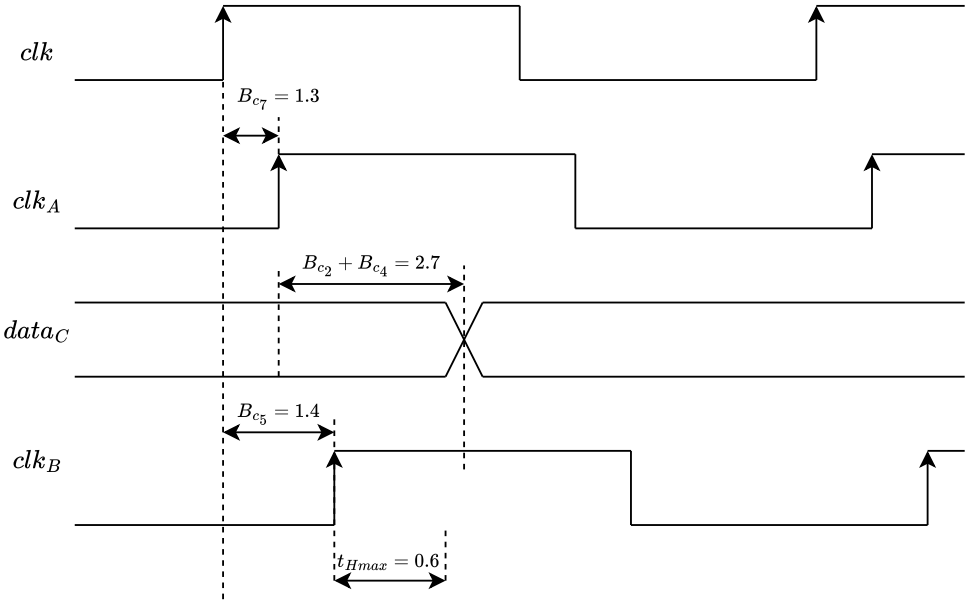

En este caso se tiene:

- Early path launching clock: 1,3 (BC7)
- Early path arriving data: 0,8 (BC2) + 1,9 (BC3) = 2,7
- Late path capturing clock: 1,4 (BC5)
- $t_{\text{sumin}} = 0{,}6$

## Hold Slack Calculation

El **hold slack** se calcula como:

$$\begin{aligned} \text{Hold slack} &= (\Delta_{clkA_min} + t_{clkA \to Q_{1_{max}}} + \Delta_{qdmin}) - (\Delta_{clk_{B_{min}}} + t_{H2_{max}}) \\ &= (BC_7 + BC_2 + BC_3) - (BC_5 + t_{\text{Hmax2}}) \\ &= (1{,}3 + 0{,}8 + 1{,}9) - (1{,}4 + 0{,}6) = 2 \end{aligned} \tag{9}$$




Si hold slack $< 0$ $\Rightarrow$ **Timing violation**.

## Solución Posible

1. Disminuir clock skew.
2. Incrementar arriving data delay.

**Hold slack es independiente de la frecuencia de clk.**

**Definición 8:** Se define como **clock skew** a la diferencia de tiempo de llegada del clk a dos flip-flops.

**Definición 9:** Se definen como registros consecutivos a todo par de registros que conformen un timing path **reg2reg**.

**Definición 10:** Se define como **skew local** a la diferencia de tiempo de arribo del clk a dos flip-flops consecutivos. Sean el registro $j$ un launching flop y el registro $i$ un capturing flop, entonces el skew local $ji$ está dado por:

$$SL_{ij} = t_{\mathrm{clk}_i} - t_{\mathrm{clk}_j} \tag{10}$$

**Definición 11:** Se define como **skew global** a la diferencia de tiempo de arribo máxima para dos registros cualesquiera, no necesariamente consecutivos.

$$SG = \max_{\substack{i<N \\ j<N}} |t_{\mathrm{clk}_i} - t_{\mathrm{clk}_j}| \tag{11}$$

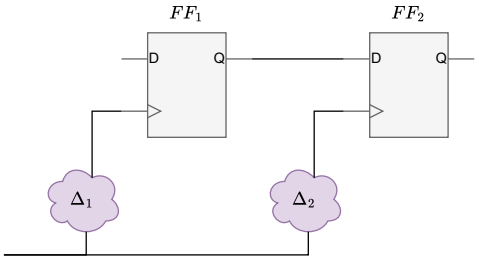
Siendo $N$ la cantidad de flip-flops que alimenta el clk.

Dado que $SL_{ij}$ es un caso específico de la diferencia de tiempo de llegada del reloj, su magnitud siempre es menor o igual al peor caso capturado por $SG$. Por lo tanto, se observa que si se minimiza $SG$, también se minimiza $|SL_{ji}|\,\forall j, i \in N$.

**Definición 12:** Dos medidas de la distribución del clk están dadas por la media y la varianza del $SL$:

$$\begin{aligned} \mu_{SL} &= \frac{1}{M}\sum SL_{ij}, \quad \text{donde } M \text{ es la cantidad de timing paths reg2reg.} \\ \sigma^2_{SL} &= \frac{1}{M}\sum (SL_{ji} - \mu_{SL})^2 \end{aligned} \tag{12}$$

- *Debe observarse que el peor escenario para **Hold Timing Violation** es el caso de los shift registers, ya que no hay retardo en la lógica del data path.*



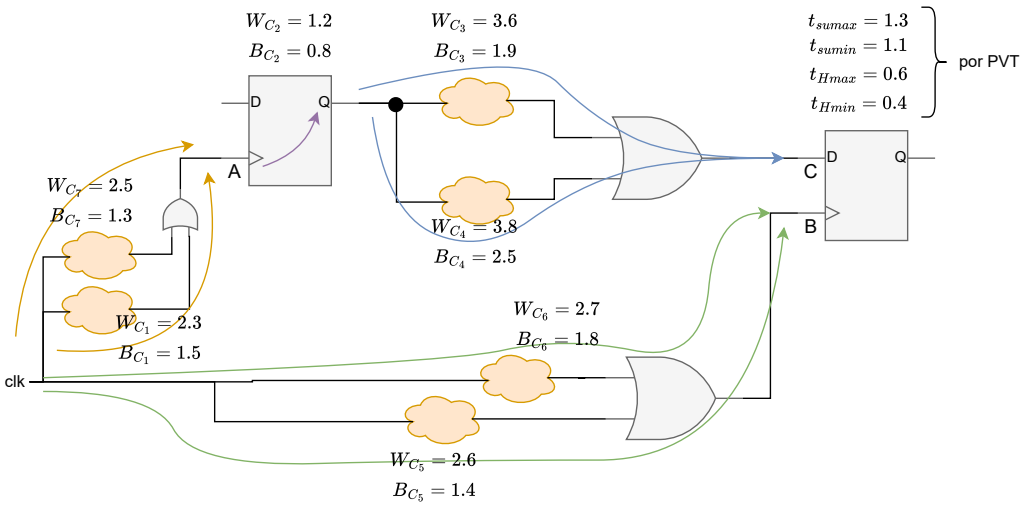

$$\text{Hold Slack} = (\Delta_1 + \Delta_3) - (\Delta_2 + t_{H2}) \tag{13}$$

- Debe observarse también que una falla del circuito por **Hold Timing Violation** es catastrófica dado que no hay frecuencia de operación a la cual el circuito pueda trabajar correctamente. Mientras que un fallo por **Setup Violation** puede ser siempre arreglado disminuyendo la frecuencia del clk.

## OCV Setup Check

*(max delays para late paths y min delays para early paths)*:

- Late path for launching clock usando **max delays**.
- Late path for arriving data usando **max delays**.
- Early path for capturing clock usando **min delays**.




En este caso se tiene:

- Late path launching clock: 2,5($WC_7$)
- Late path arriving data: 1,2($WC_2$) + 3,8($WC_4$) = 5
- Early path capturing clock: 1,4($BC_5$)
- tsumax = 1,3

Luego:

$$\text{Setup slack} = T_{clk} - (\Delta_{clk_A max} + t_{clk \to Q_1 max} + \Delta q_{d_{min}} + t_{su_2 max}) + \Delta t_{clk_B min} \quad (14)$$

Expandiendo:

$$\begin{aligned} \text{Setup slack} &= T_{\text{clk}} - (WC_7 + WC_2 + WC_4 + t_{su_2\text{max}}) + BC_5 \\ &= T_{\text{clk}} - (2{,}5 + 1{,}2 + 3{,}8 + 1{,}3) + 1{,}4 \\ &= T_{\text{clk}} - 7{,}4 \end{aligned} \quad (15)$$

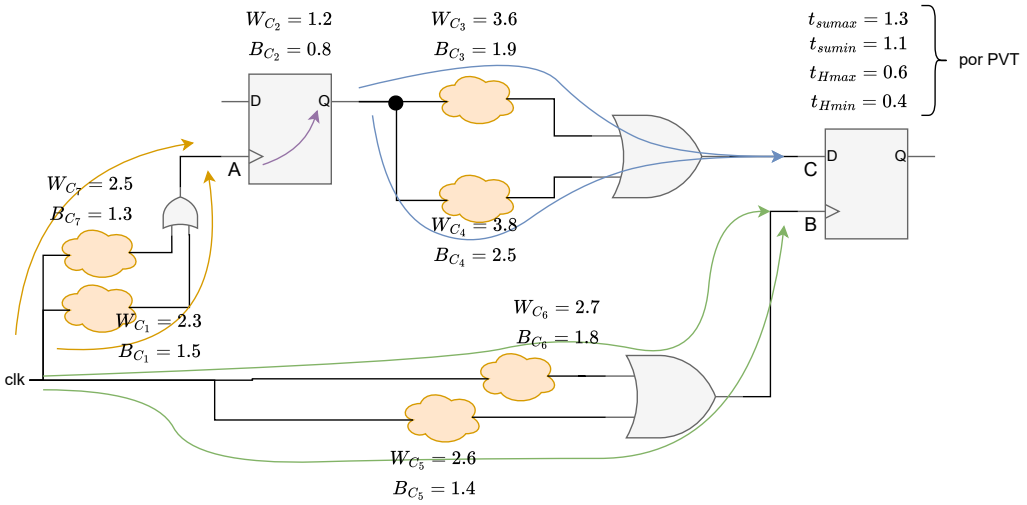
**Observamos que:**

**Setup slack OCV < Setup slack BC-WC**

## OCV Hold Check

*(Se deben usar max delays para late paths y min delays para early paths)*:

- Early path for launching clock usando **min delays**.
- Early path for arriving data usando **min delays**.
- Late path for capturing clock usando **max delays**.




En este caso se tiene:

- Early path launching clock: 1,3($BC_7$)
- Early path arriving data: 0,8($BC_2$) + 1,9($BC_3$) = 2,7
- Late path capturing clock: 2,7($WC_6$)
- Además $t_{H_2\max} = 0{,}6$

Luego:

$$\text{Hold slack} = (\Delta t_{clk_A min} + t_{clk \to Q_1 min} + \Delta q_{d_{min}}) - (\Delta t_{clk_B max} + t_{H_2 max}) \tag{16}$$

Expandiendo:

$$\begin{aligned} \text{Hold slack} &= (BC_7 + BC_2 + BC_3) - (WC_6 + t_{H2\max}) \\ &= (1{,}3 + 0{,}8 + 1{,}9)ns - (2{,}7 + 0{,}6)ns \\ &= 4ns - 3{,}3ns = 0{,}7ns \end{aligned} \tag{17}$$

**Se observa que:**

**Hold Slack OCV < Hold Slack BC WC**

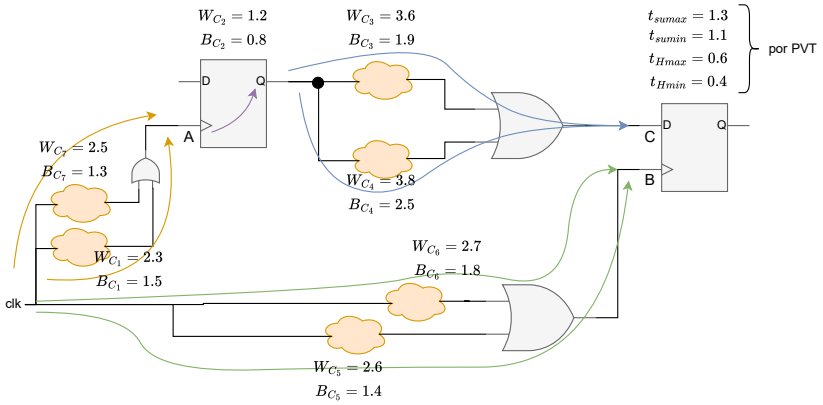
## Conclusión

- **BC-WC** modela el comportamiento de un circuito como si todas las celdas son del mismo tipo (o bien BC o bien WC). Esto se debe a que si el proceso resultó rápido todas las celdas del modelo son similares (100 % de correlación).
- **OCV** modela el comportamiento del circuito en las peores condiciones. En este caso, una celda es lo más lenta posible para **setup check** y lo más rápida para **hold check** (esto no cumple en la práctica, la celda tiene un retardo fijo).

## Opción Más Realista

- Usar BC-WC análisis y aplicar **de-rating factor**.
- Sea $D_E$ el derating factor para **early paths** $O < D_E < 1{,}0$.
- Sea $D_L$ el derating factor para **late paths** $D_L > 1{,}0$.

### BC-WC Derated Setup Check




Entonces para el ejemplo anterior se tiene que:

- Late path launching clock: $WC_7 \times D_L = 2{,}5 \times D_L$
- Late path arriving data: $(WC_2 + WC_4) \times D_L = 5 \times D_L$
- Early path capturing clock: $WC_5 \times D_E = 2{,}6 \times D_E$
- Además $t_{sumax} = 1{,}3$

El **setup slack** se calcula como:

$$\begin{aligned}\text{Setup slack} &= T_{clk} - (\Delta_{clk_A max} + t_{clk_1 \to q_1 max} + \Delta q_{\text{d max}}) \times D_L + \Delta_{clk_B min} \times D_E - t_{su_B max} \\ &= T_{clk} - (2{,}5 + 1{,}2 + 3{,}8) \times D_L + 2{,}6 \times D_E - 1{,}3\end{aligned} \tag{18}$$

Supongamos:

$$D_L = 1{,}1, \quad D_E = 0{,}9$$

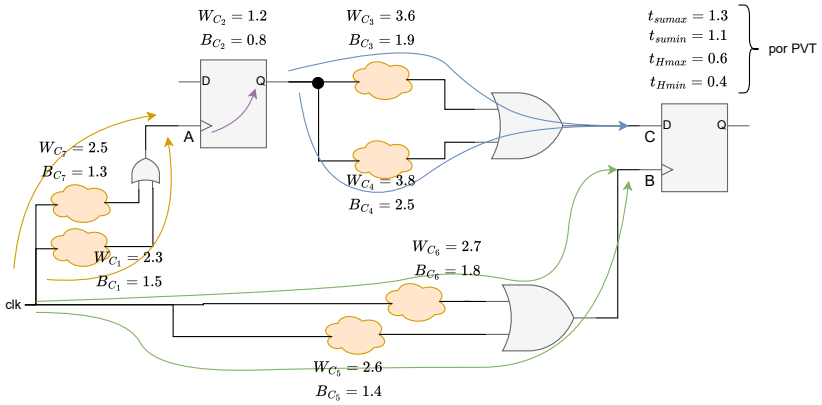
Entonces:

$$\begin{aligned}\text{Setup slack} &= T_{\text{clk}} - 7{,}5 \times 1{,}1 + 2{,}6 \times 0{,}9 - 1{,}3 \\ &= T_{\text{clk}} - 8{,}25 + 2{,}34 - 1{,}3 \\ &= T_{\text{clk}} - 7{,}22\end{aligned} \tag{19}$$

**Se observa que:**

**Setup slack derated < Setup slack BC-WC**

## BC-WC Derated Hold Check




- Early path launching clock: $BC_7 \times D_E = 1{,}3 \times D_E$
- Early path arriving data: $(BC_2 + BC_3) \times D_E = 2{,}7 \times D_E$
- Late path capturing clock: $BC_5 \times D_L = 1{,}4 \times D_L$
- Además $t_{H_2max} = 0{,}6$

El **hold slack** se calcula como:

$$\begin{aligned}\text{Hold slack} &= (\Delta_{clk_A\ \min} + t_{clk_A \to q_1\ \min} + \Delta q_{d\ \min}) \times D_E - \Delta_{clk_B\ \max} \times D_L - t_{H_2\max} \\ &= (BC_7 + BC_2 + BC_3) \times D_E - 1{,}4 \times D_L - 0{,}6 \\ &\text{Supongamos } D_L = 1{,}1 \text{ y } D_E = 0{,}9, \text{ entonces:} \\ &= (1{,}3 + 0{,}8 + 1{,}9) \times 0{,}9 - 1{,}4 \times 1{,}1 - 0{,}6 \\ &= 3{,}6 - 1{,}54 - 0{,}6 = 1{,}46\end{aligned} \tag{20}$$

**Se observa que:**

**Hold Slack derated < Hold Slack BC WC**

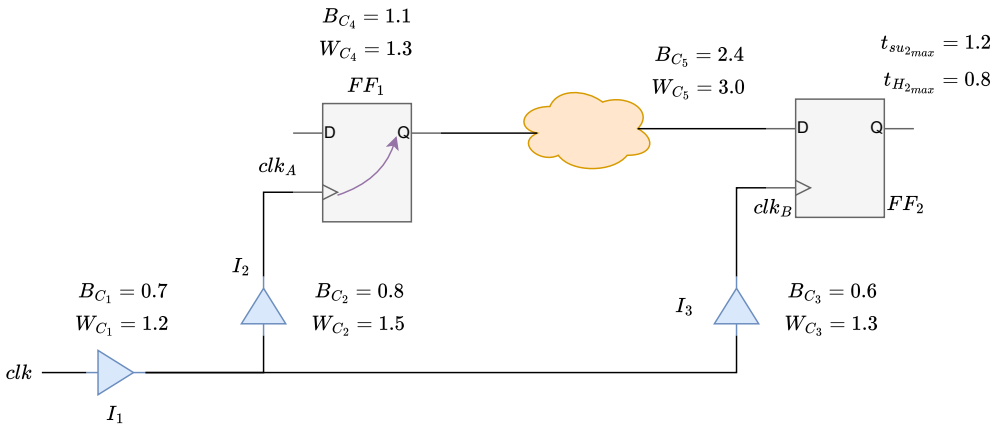
## Conclusión

*Derated timing analysis da un margen de robustez al diseño respecto de BC-WC analysis, pero no tan pesimista como OCV.*

# Clock Path Pessimism Removal (CPPR)

Clock Path Pessimism Removal (CPPR), o también llamado Clock Reconvergence Pessimism Removal (CRPR), es el proceso de identificar y remover el pesimismo introducido por clock paths que tienen segmentos en común.

# CPPR Application Example




## OCV Setup Check

$$\begin{aligned}\text{Setup slack} &= T_{clk} - (\Delta_{clk_A max} + t_{clk_A \to Q_1 max} + \Delta q_{1d_2 max}) + \Delta_{clk_B min} - t_{su_2 max} \\ &= T_{clk} - (\overbrace{WC_1 + WC_2}^{} + WC_4 + WC_5) + (\overbrace{BC_1 + BC_3}^{}) - t_{su_2 max} \\ &= T_{clk} - (1{,}2 + 1{,}5 + 1{,}3 + 3{,}0) + (0{,}7 + 0{,}6) - 1{,}2 \\ &= T_{clk} - 7{,}0 + 1{,}3 - 1{,}2 = T_{clk} - 6{,}9\end{aligned} \tag{21}$$

## Aplicando CPPR

La celda $I_1$ presenta el mismo retardo (desconocido) para el early y el late path:

$$\begin{aligned}\text{Setup slack} &= T_{clk} - (\cancel{\Delta I_1} + WC_2 + WC_4 + WC_5) + (\cancel{\Delta I_1} + BC_3) - t_{su_2max} \\ &= T_{clk} - (1{,}5 + 1{,}3 + 3{,}0) + (0{,}6) - 1{,}2 \\ &= T_{clk} - 5{,}8 + 0{,}6 - 1{,}2 \\ &= T_{clk} - 6{,}4\end{aligned} \tag{22}$$

**Se observa que:**

**Setup slack CPPR $\geq$ Setup slack sin CPPR**

## OCV Hold Check

$$\begin{aligned}\text{Hold slack} &= (\Delta_{clk_A\min} + t_{clk_A \to q_1\min} + \Delta q_{1d_2\min}) - (\Delta_{clk_B\max} + t_{H_2\max}) \\ &= (\overbrace{BC_1 + BC_2} + BC_4 + BC_5) - (\overbrace{WC_1 + WC_3} + t_{H_2\max}) \\ &= (0{,}7 + 0{,}8 + 1{,}1 + 2{,}4) - (1{,}2 + 1{,}3 + 0{,}8) \\ &= 5 - 3{,}3 = 1{,}7\end{aligned} \tag{23}$$

## Aplicando CPPR

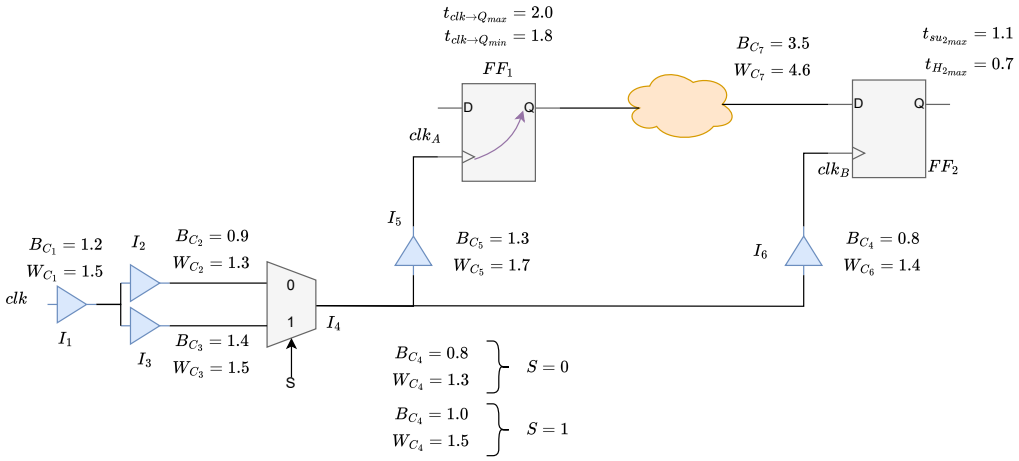
La celda $I_1$ comparte el mismo retardo (desconocido) para el early y el late path:

$$\begin{aligned}\text{Hold slack} &= (\cancel{\Delta I_1} + BC_2 + BC_4 + BC_5) - (\cancel{\Delta I_1} + WC_3 + t_{su_2max}) \\ &= (0{,}8 + 1{,}1 + 2{,}4) - (1{,}3 + 0{,}8) \\ &= 4{,}3 - 2{,}1 = 2{,}2\end{aligned} \tag{24}$$

**Se observa que:**

**Hold slack CPPR $\geq$ Hold slack sin CPPR**

## BC-WC Setup Check (solo con WC delays)




$$
\begin{aligned}
\text{Setup slack} &= T_{clk} - (\Delta_{clk_1\max} + t_{clk_1\to Q_1\max} + \Delta q_{1d_2\max}) + \Delta clk_{2\min} - t_{su_2\max} \\
&= T_{clk} - (WC_1 + WC_3 + WC_4\big|_{S=1} + WC_5 + t_{clk_1\to Q_1\max} + WC_7) \\
&+ (WC_1 + WC_2 + WC_4\big|_{S=0} + WC_6) - t_{su_2\max} \\
&= T_{clk} - (1{,}5 + 1{,}5 + 1{,}5 + 1{,}7 + 2{,}0 + 4{,}6) + (1{,}5 + 1{,}3 + 1{,}3 + 1{,}4) - 1{,}1 \\
&= T_{clk} - 12{,}8 + 5{,}5 - 1{,}1 \\
&= T_{\text{clk}} - 8{,}4
\end{aligned}
\tag{25}
$$

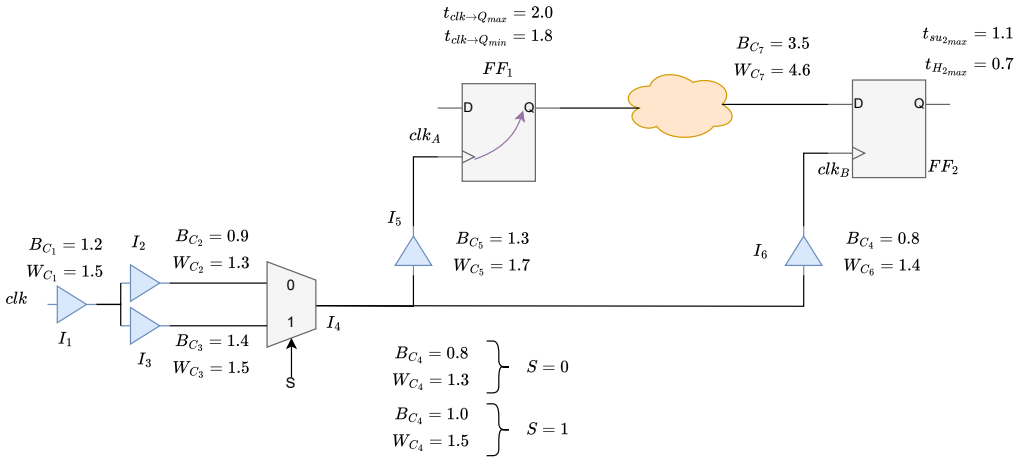
## BC-WC Setup Check con CPPR (solo con WC delays)

$$
\begin{aligned}
\text{Setup slack} &= T_{clk} - (WC_5 + t_{clk_1\to Q_1\max} + WC_7) + (WC_6) - t_{su_2\max} \\
&= T_{clk} - (1{,}7 + 2{,}0 + 4{,}6) + 1{,}4 - 1{,}1 \\
&= T_{clk} - 8{,}3 + 1{,}4 - 1{,}1 \\
&= T_{clk} - 8
\end{aligned}
\tag{26}
$$

**Se observa que:**

| **Setup slack con CPPR $\geq$ Setup slack sin CPPR** |
|---|

## BC-WC Hold Check (usando solo BC delays)




$$
\begin{aligned}
\text{Hold slack} &= (\Delta t_{clk_1\min} + t_{clk_1\to Q_1\min} + \Delta q_{1d_2\min}) - \Delta clk_{2\max} - t_{H_2\max} \\
&= (\underbrace{BC_1 + BC_2 + BC_4\big|_{S=0} + BC_5}_{\Delta\text{clk}_1\text{min}} + t_{clk_1\to Q_1\min} + BC_7) \\
&\quad - (\underbrace{BC_1 + BC_3 + BC_4\big|_{S=1} + BC_6}_{\Delta\text{clk}_2\text{max}}) - t_{H_2\max} \\
&= (1{,}2 + 0{,}9 + 0{,}8 + 1{,}3 + 1{,}8 + 3{,}5) - (1{,}2 + 1{,}4 + 1{,}0 + 0{,}8) - 0{,}7 \\
&= 9{,}5 - 4{,}4 - 0{,}7 = 4{,}4
\end{aligned} \tag{27}
$$

## BC-WC Hold Check con CPPR (solo usando BC delays)

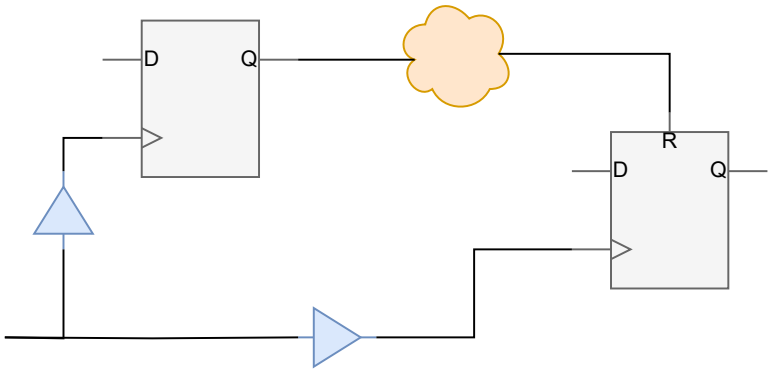
$$
\begin{aligned}
\text{Hold slack} &= (BC_5 + t_{clk_1\to Q_1\min} + BC_7) - (BC_6) - t_{H_2\max} \\
&= (1{,}3 + 1{,}8 + 3{,}5) - 0{,}8 - 0{,}7 \\
&= 6{,}6 + 0{,}8 + 0{,}7 = 5{,}1
\end{aligned} \tag{28}
$$

**Se observa que:**

**Hold slack con CPPR $\geq$ Hold slack sin CPPR**

## Removal and Recovery Checks

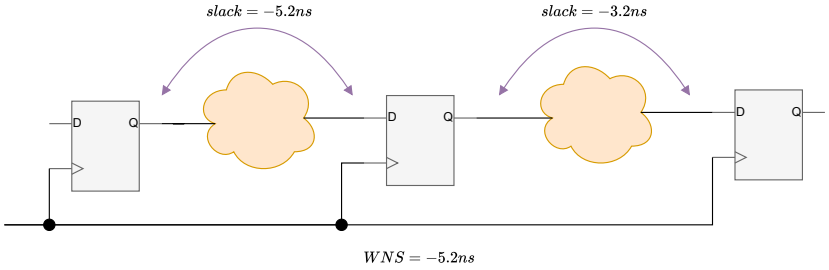
El análisis de *removal time* y de *recovery time* son idénticos a los límites del *hold time* y *setup time* respectivamente.

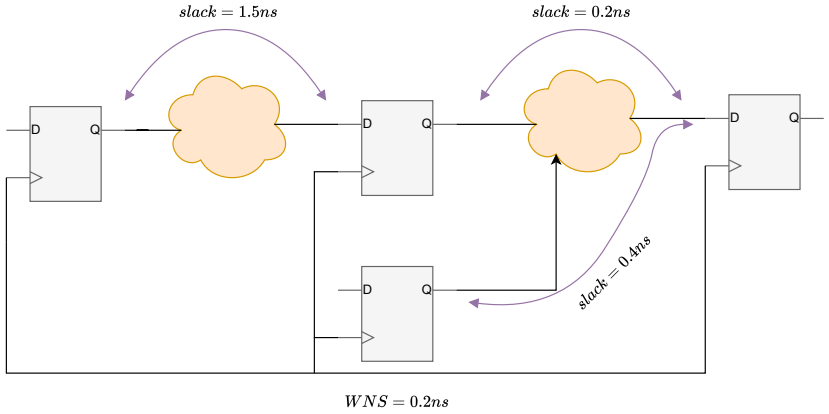



**Nota:** *No se recomienda alimentar de forma directa un flip-flop por una señal sincrónica funcional.* Este pin sólo debería ser usado para *Power on Reset.*
Sin embargo, herramientas de STA permiten realizar este análisis.

## Definición: Worst Negative Slack (WNS)

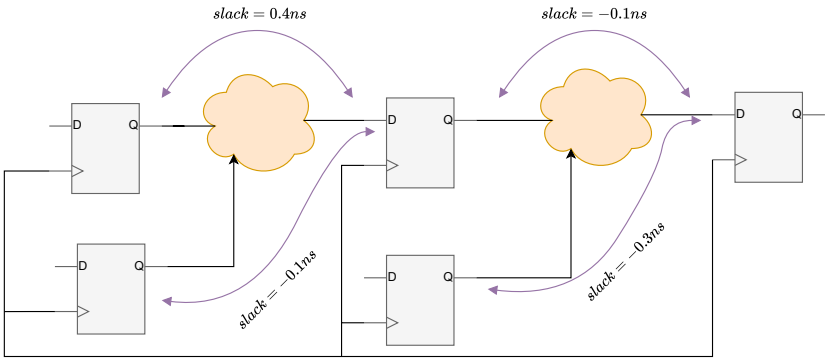
Se define como **Worst Negative Slack** (ya sea *setup slack* o *hold slack*) al *slack más negativo* encontrado en el circuito.




## Definición: Total Negative Slack (TNS)

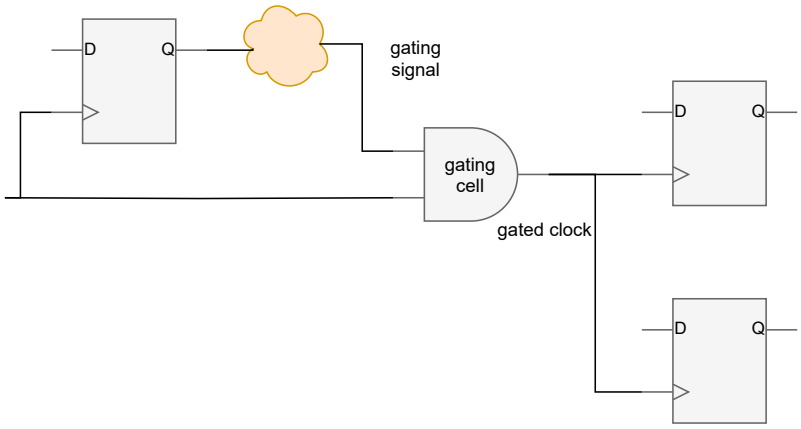
Se define como **Total Negative Slack (TNS)** a la suma de todos los WNS en cada endpoint.




## Clock Gating Checks

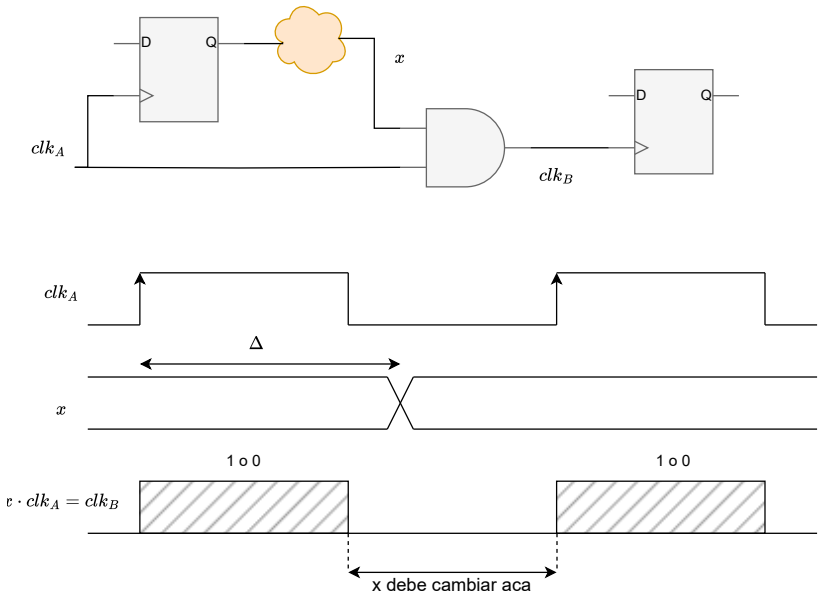
Se utiliza cuando una señal puede controlar a un *clock*.

- Ejemplo de circuito de **gating signal** y **gating cell**.




**Nota:** Esta práctica debe ser evitada a toda costa. Sin embargo, las herramientas de STA permiten su análisis.

### Active High Clock Gating

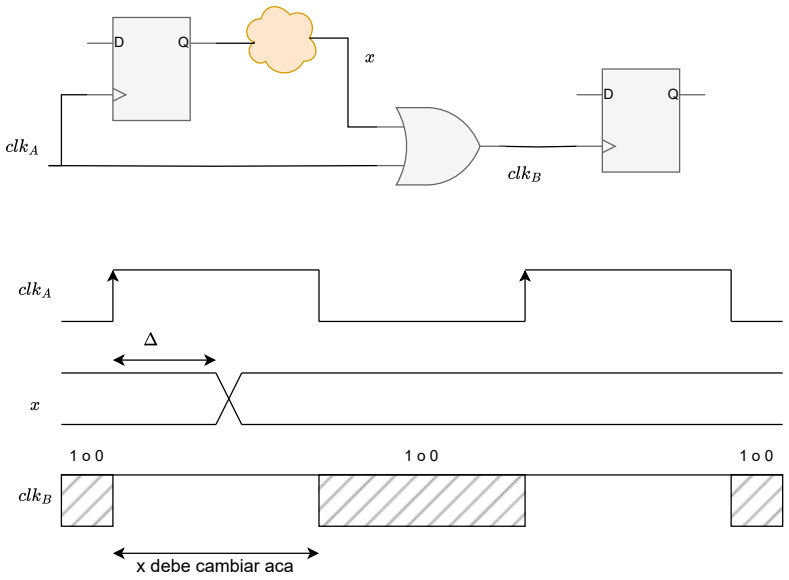



La celda de **gating** es de tipo **AND** o **NAND**. La señal (*gating signal*) debe cambiar mientras $clk_A = 0$ de forma tal de no afectar el duty cycle ni producir glitches en $clk_B$. Por medio de STA se encuentran los valores $\Delta_{min}$ y $\Delta_{max}$. Dichos valores deben ser:

$$\begin{aligned} \frac{1}{2}T_{clk_A} &< \Delta_{min} \\ \Delta_{max} &< T_{clk_A} \end{aligned} \tag{29}$$

De no cumplirse estas condiciones, hay una violación de *clock gating timing*.

### Active Low Clock Gating




Se utiliza cuando la celda de *gating* es de tipo **OR** o **NOR**.

La señal X (*gating signal*) debe cambiar mientras $clk_A = 1$ de forma tal de no afectar el *duty cycle* ni producir glitches en $clk_B$.

**Nota:** El procedimiento de STA se encarga de encontrar los valores $\Delta_{\min}$ y $\Delta_{\max}$. Dichos valores debe ser:

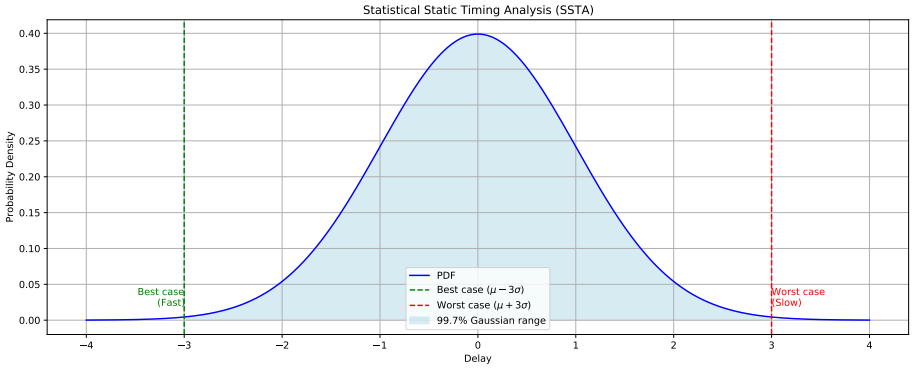
$$\begin{aligned} 0 &< \Delta_{min} \\ \Delta_{max} &< \frac{T_{clk_A}}{2} \end{aligned} \tag{30}$$

Esto asegura que no haya **clock gating violations**.

## Statistical Static Timing Analysis (SSTA)

Cuando se hace STA, el análisis se basa en los dos casos extremos de las condiciones **PVT** (process-voltage-temperature). Cuando se realiza el modelado de las celdas, se las caracteriza entonces para estos dos extremos. Sin embargo, sabemos que estos dos extremos **PVT** corresponden en realidad a $\pm 3\sigma$ de los parámetros de la celda.

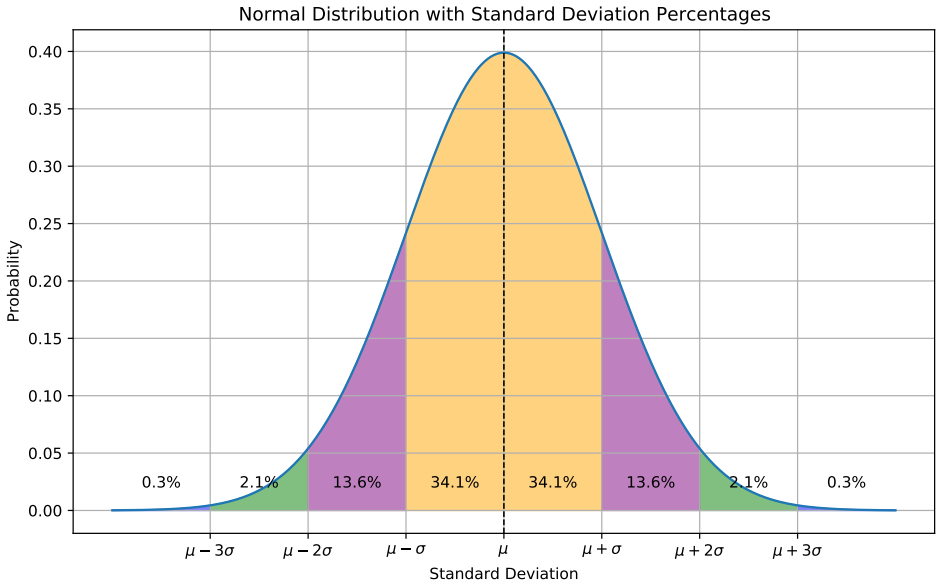
**Ejemplo:**




- Distribución Normal (Gaussiana):

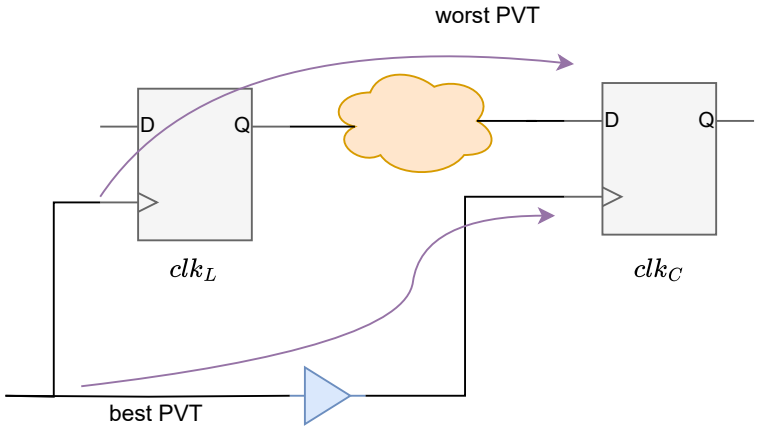
$$\mu - 3\sigma \quad \text{(Fast case)}$$
$$\mu + 3\sigma \quad \text{(Slow case)}$$

- 99,7 % de los casos para procesos gaussianos.

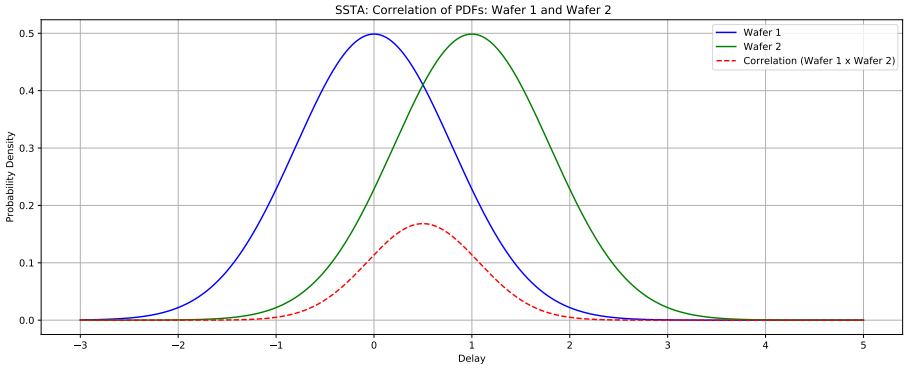


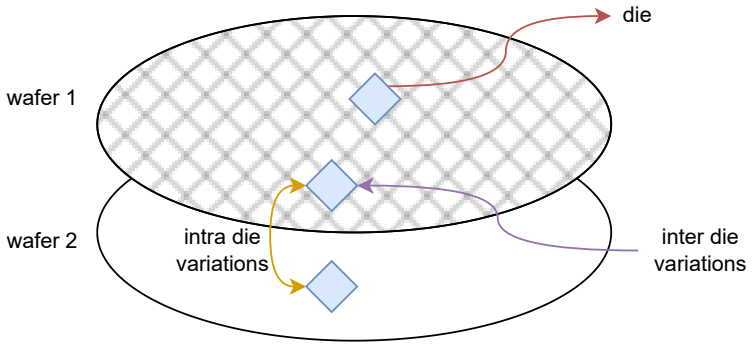

Cuando se realiza el STA en modo **BC-WC**, en realidad no se está considerando todos los extremos de las condiciones **PVT**, mientras que en modo **OCV** sí se hace. El problema

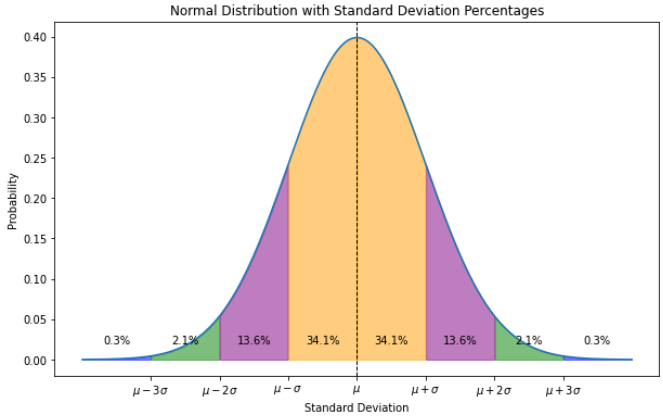
es que OCV es extremadamente pesimista ya que, por ejemplo, en el siguiente circuito para setup análisis consideraríamos que el data path time las peores condiciones **PVT** y el *capturing clock* las mejores (muy poco probable).

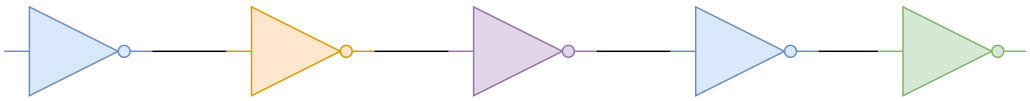



Es de esperar que al menos paths geométricamente cercanos operen bajo condiciones **PVT** similares, lo cual se modela por una función de correlación.





SSTA es una técnica de modelado de variaciones que calcula el impacto de las variaciones locales del proceso en el delay y la pendiente (slew) de cada instancia en el diseño, en un punto de variación global específico. SSTA propaga el sigma de los tiempos de llegada y requeridos a través del grafo de temporización, y luego calcula las características estadísticas del slack en todos los pines de temporización.

SSTA soluciona algunas limitaciones del análisis OCV:

- Ineficiencia en operaciones de ultrabajo voltaje (V operación $\sim$ V umbral).
- Dependencia del tiempo de las celdas con respecto a las pendientes (slews) y cargas.

SSTA se puede realizar utilziando librerías de variación en formato Liberty Variation Format (LVF). LVF incluye variaciones absolutas a nivel de arco en los retrasos de las

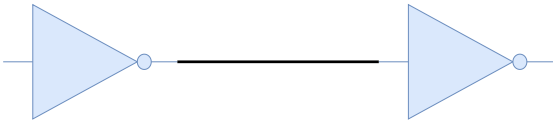
celdas, transiciones de salida y verificaciones de temporización como funciones de la pendiente de entrada y la carga.

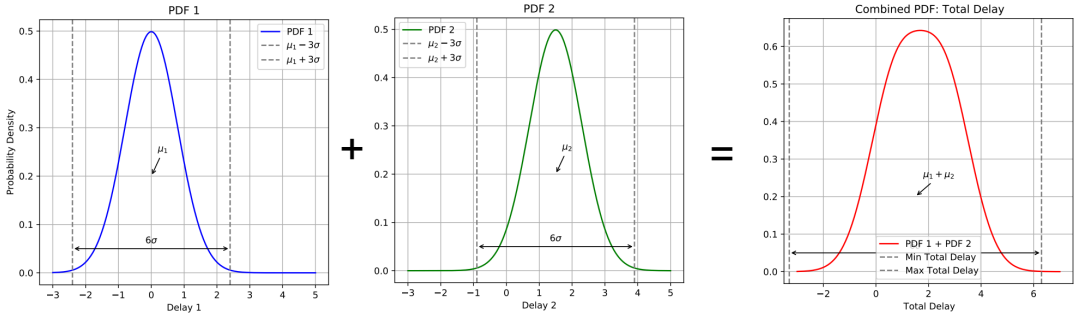
## Conclusión:

El análisis SSTA ofrece una representación estadística más realista, comparado con los métodos tradicionales BC-WC o OCV.

En SSTA, los parámetros *delay* y *slew out* de las celdas, así como también los valores $R$ y $C$ de la interconexión, son modelados estadísticamente.

### Ejemplo:






- Distribución individual de los delays:

$$\mu_1, \sigma_1 \quad \text{(delay 1)}$$

$$\mu_2, \sigma_2 \quad \text{(delay 2)}$$

- Distribución combinada de los delays:

$$\text{Total delay: } \mu_{total} = \mu_1 + \mu_2, \quad \sigma^2_{total} = \sigma^2_1 + \sigma^2_2$$

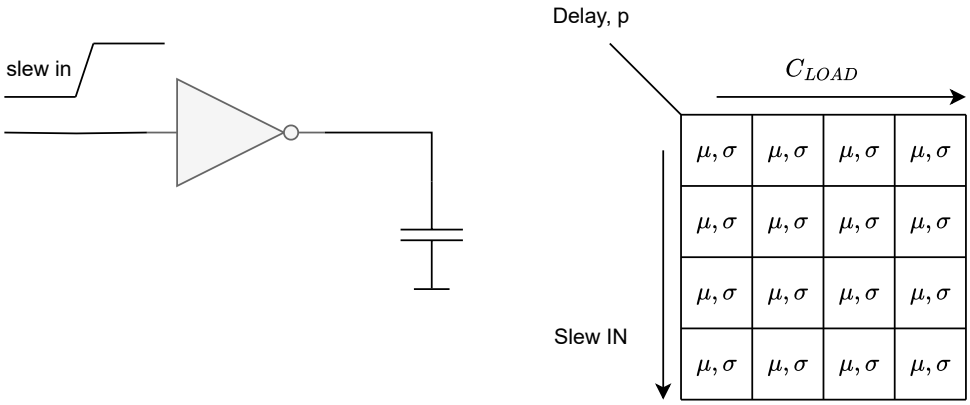
Si se consideran que las variables no están correlacionadas:

$$\sigma_T = \sqrt{\sum \sigma_i^2} \tag{31}$$

$\sigma_T$ depende de la correlación entre $\text{delay}_1$ y $\text{delay}_2$

## Funciones de Correlación

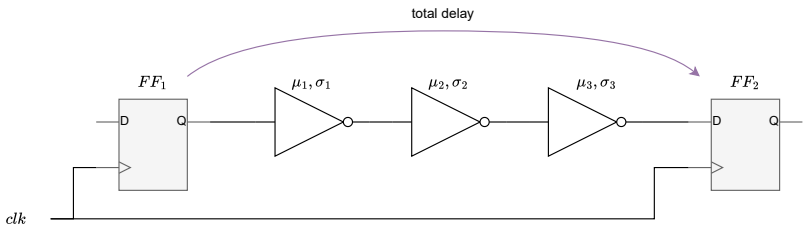
Cada parámetro en el *Liberty File* tendrá asociado un valor de media, varianza, y una función de correlación en función de la distancia. La interpolación bilineal se hará ahora para obtener los valores de $\mu$ y $\sigma$ del retardo de la celda.

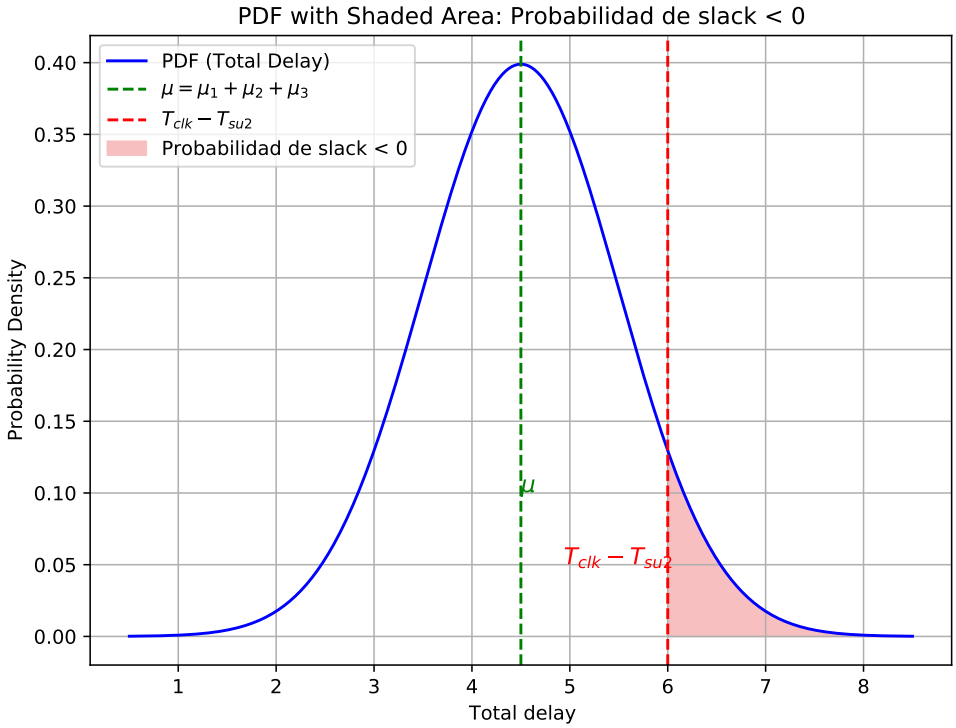



Algunos modelos de correlación frecuentemente usados son:

- Sea $y =$ distancia
- Constante: $\rho(y) = \begin{cases} 1, & y = 0 \\ \alpha, & y \neq 0 \quad \alpha < \alpha \leq 1 \end{cases}$
- Lineal: $\rho(y) = 1 - \alpha \cdot y$
- Exponencial: $\rho(y) = e^{-\alpha y}; \quad \rho(y) = (1-\beta)e^{-\alpha y} + \beta$
- Bessel: sea $K$ función de Bessel modificada de segundo tipo. y $\Gamma$ función Gamma $\rho(y) = 2\frac{by}{e}^{y-1} K_{y-1}(by)\Gamma_{y-1}^{-1}$

Con los valores de $\mu_i$ y $\sigma_i$, a lo largo del path $y$ de $\rho$, se computa la probabilidad de *slack negativo*, y se busca por ejemplo que la probabilidad de $slack < 0$ sea menor al 5 %.






## Introducción a Synopsys Design Constraints (SDC)

Es un lenguaje casi estandarizado que permite definir las condiciones sobre las celdas sobre las cales se realiza el *Static Timing Analysis (STA)* y se optimiza el diseño. De esta forma, existen tres grupos mayoritarios de constraints:

- **Design rule constraints:** transition time, capacitance, fanout.
- **Optimization constraints:** clocks, delays.
- **Settings:** library, PVT conditions, wire load model.

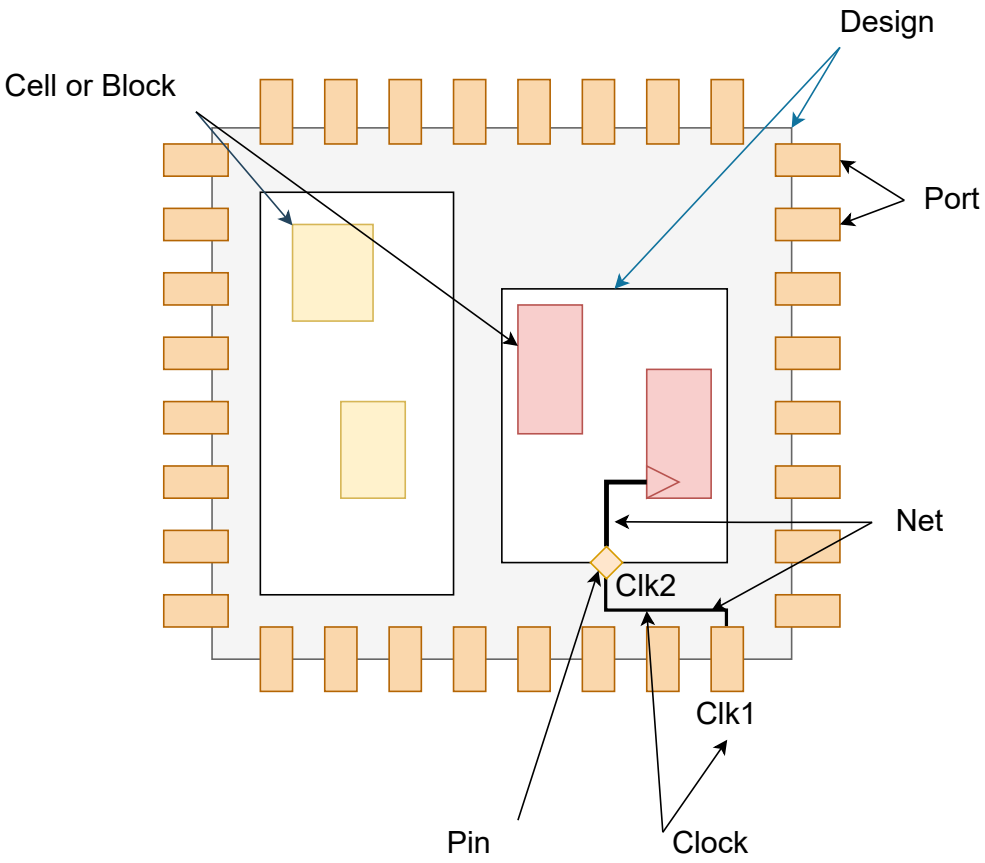
## Objetos del Diseño

El conjunto de constraints puede ser aplicado sobre los siguientes objetos de diseño:

- **design:** un *contenedor lógico* del diseño actual.

- **clock:** un objeto que representa una señal de clk.
- **port:** un objeto que representa un puerto de un bloque o diseño.
- **cell:** un objeto que representa una instancia del diseño.
- **pin:** un puerto de un tipo *cell*.
- **net:** un objeto que representa una conexión entre dos objetos tipo pin o port.
- **library:** un objeto que representa la biblioteca de celdas estándar.

## 1.9. Resumen objetos



## Acceso a los Objetos del Diseño

Los siguientes comandos se utilizan para tener acceso a los objetos antes mencionados:

| **Objeto** | **Comando** |
|---|---|
| design | current_design |
| clock | get_clocks<br>all_clocks |
| port | get_ports<br>all_inputs<br>all_outputs |
| cell | get_cells |
| pin | get_pins |
| net | get_nets |
| library | get_libs |

## Optimization Constraints

Son las constraints que se utilizan para establecer el **STA** y configurar la optimización de timing de la herramienta de síntesis.

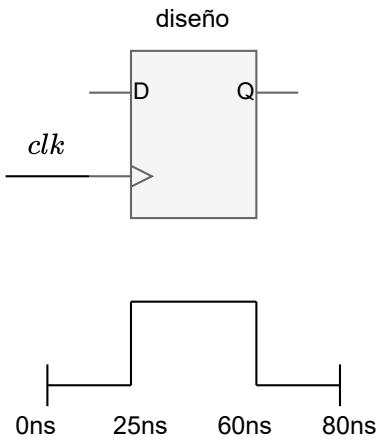
### Clocks

**Especificación de clocks:**

```
create_clock -name <clock-object-name>\
  -period <time-units>\
  -waveform {<time-rise> <time-fall>}\
  [get_ports <port-name>]
```

**Ejemplo:**

```
create_clock -name my_clock\
  -period 20\
  -waveform {25 60}\
  [get_ports clk]
```




**Notas:**

- Si no se especifica la opción `waveform`, el valor por defecto es: waveform $= \{0, \text{period}/2\}$
- El valor `period` significa que el periodo es en unidades de tiempo. El valor por defecto de las unidades de tiempo es nanosegundos.

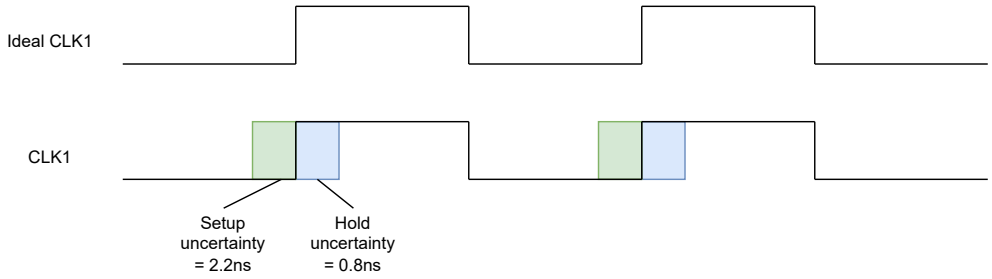
### Clock Uncertainty

Permite modelar la incertidumbre en el clk ya sea por jitter, skew o ambos.

**Ejemplo:**

```
# set_clock_uncertainty [-from from_clock] [-to to_clock] \
#                       [-rise] [-fall]    \
#                       [-setup] [-hold]   \
#                        uncertainty       \
#                       [clock_objects]

set_clock_uncertainty -setup 2.2 [get_clocks my_clock]
set_clock_uncertainty -hold 0.8 [get_clocks my_clock]
```



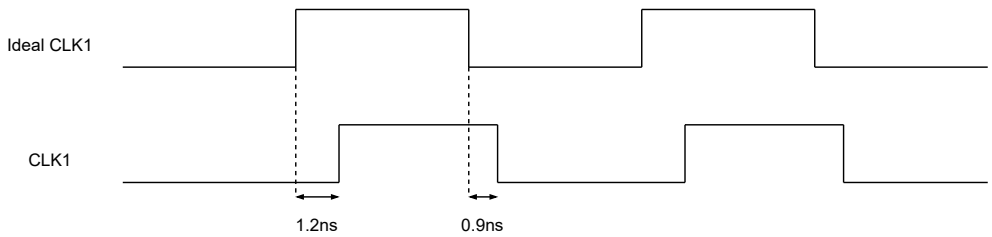

## Clock Latency

Además del (*clock slew*) y la incertidumbre del clk (*clock uncertainty*), la latencia del clk (*clock latency*) se establece en el puerto del clk para tener en cuenta los retrasos en la señal de clk desde el punto de definición del clk hasta el pin de clk de un registro. Puedes definir tanto la latencia de la fuente como la latencia de la red utilizando el comando `set-clock-latency`. Cuando especificas *rise* o *fall*, estos incluirán también los tiempos de transición de ascenso y descenso. Sin esas opciones, simplemente se modelaría la latencia sin considerar los tiempos de transición. La latencia de la fuente se especifica utilizando la opción `-source`. Si no se especifica la opción `-source`, se está modelando la latencia de la red. En este gráfico de ejemplo que muestra el clk ideal frente al clk real, el desplazamiento visible en la forma de onda del clk es causado por las latencias.

**Ejemplo:**

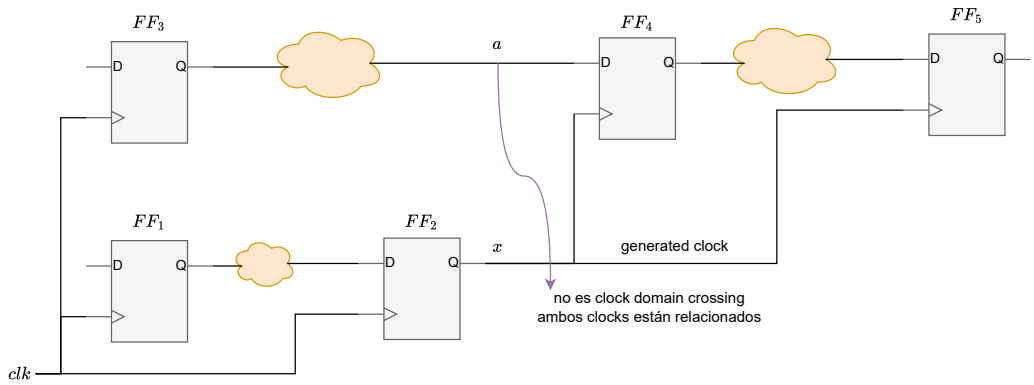
```
# set_clock_latency [-source] [-clock clock_list]    \
#                   latency obj_list                 \
#                   [-min ] [-max] [-rise ] [-fall]  \
#                   [-early] [-late] [-clock_gate]


set_clock_latency 1.2 -rise [get_clocks CLK1]
set_clock_latency 0.9 -fall [get_clocks CLK1]
```

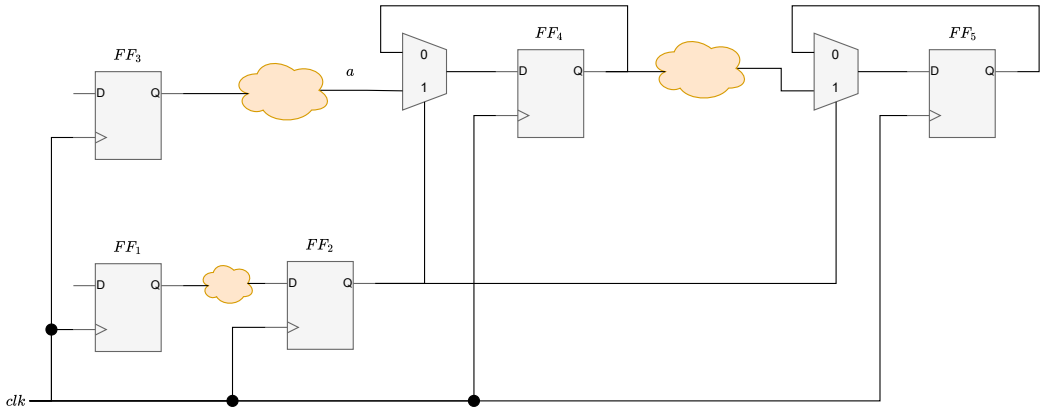


### Generated Clocks

**Definición:** Un **generated clock** es todo pin Q de salida de un flip-flop D que sirve como clock para otro flip-flop D en el diseño.




**Nota:** Esta práctica **NO** es recomendable, aunque **SDC** permite su modelado. Ejemplo: En un diseño donde un nodo $a$ genera un clock derivado de una señal principal.



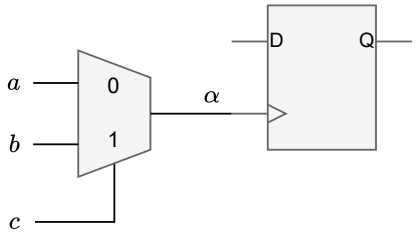

## Generated Clocks

```
create_generated_clock -name my_clock_div \
  -source [get_pins clk] \
  -divide_by 2 \
  [get_pins FF2/Q]
```

**Nota:** El diseñador debe especificar por cuánto se divide el clock principal, ya que el

entorno *STA* no analiza el *RTL*.

**Ejemplo:**

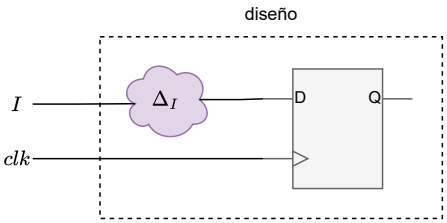



$\alpha$ no es *generated clock*, ya que no es la salida de un pin $Q$ de un flip-flop.

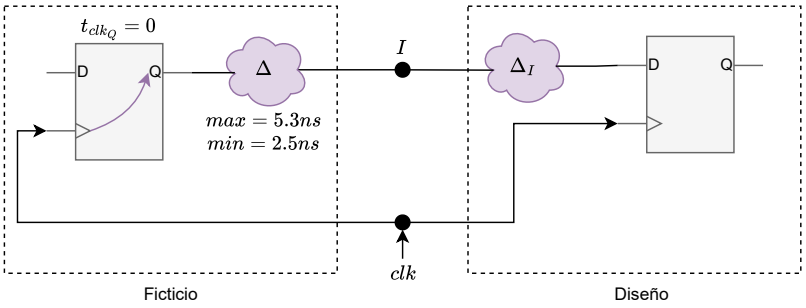
### Nota:

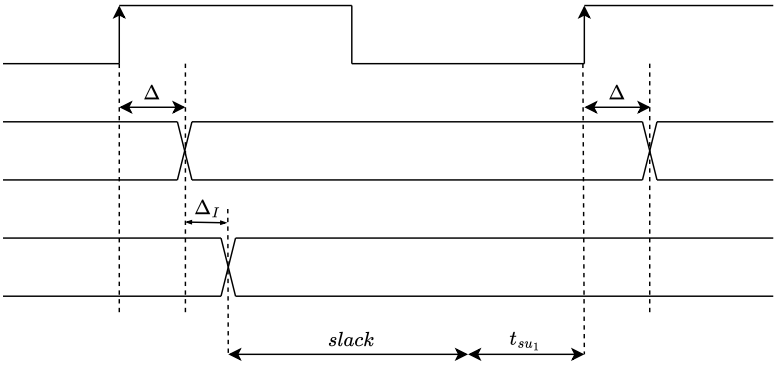
Al definir todos los clocks y los *generated clocks* de un diseño, automáticamente quedan restringidos todos los *paths reg2reg* del diseño. Es decir, el entorno *STA* conoce cómo analizar las violaciones de setup y hold de todos los *reg2reg paths*. Esto sucede con los *paths in2reg* y *reg2out*, que deben ser restringidos de manera separada.

## Inputs




Si la entrada $I$ es sincrónica con $clk$, siempre debemos declarar la relación temporal entre ambas señales.
Para restringir la entrada $I$ respecto de $clk$, se asume que existe un flip-flop y una lógica que genera la señal $I$ de la siguiente forma:




```
set_input_delay -clock my_clk -max 5.3 [get_ports I]  # Setup analysis
set_input_delay -clock my_clk -min 2.5 [get_ports I]  # Hold analysis
```

Este comando asegura el análisis temporal correcto respecto a la señal de entrada.

## 2. Path Exceptions

Cuando se necesita que la herramienta STA cree excepciones a los cálculos normales de *timing*, se hace a través de (*path exceptions*). Estas son las más comunes:

- *Multicycle paths*
- *False paths*
- *Disabling timing arcs*
- *Case analysis*
- *Path delay limits*

## 2.1. Multicycle Paths

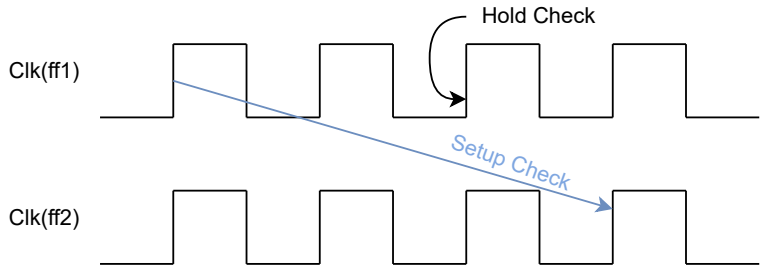
Por defecto, las herramientas de análisis estático de tiempos asumen que todos los *path* temporales cumplirán con los tiempos en un solo ciclo. Aquellos *paths* que requieren más de un período de clock para su ejecución se denominan *multicycle paths*.

El comando `set_multicycle_path` especifica los *path* temporales designados en el diseño actual cumplan con las verificaciones de *setup* y *hold* en múltiples ciclos de reloj.

```
set_multicycle_path [-setup | -hold] <valor> [-start | -end] \
                    [-from <lista>] [-through <lista>] [-to <lista>]
```

El siguiente comando indica a la herramienta que verifique una violación de *setup* tres ciclos después del flanco de lanzamiento del reloj. Todas los *paths* entre `ff1` y `ff2` son *path* de 3 ciclos.

```
set_multicycle_path -setup 3 -from {ff1} -to {ff2}
```

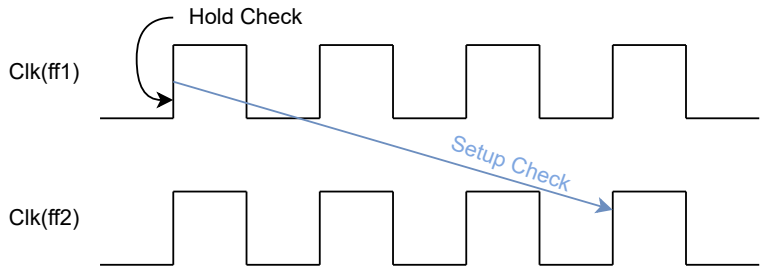



### Multicycle Path: Restableciendo Hold

Cuando deseas configurar un *path* de múltiples ciclos solo para *setup* pero no para *hold*, utiliza ambos comandos juntos:

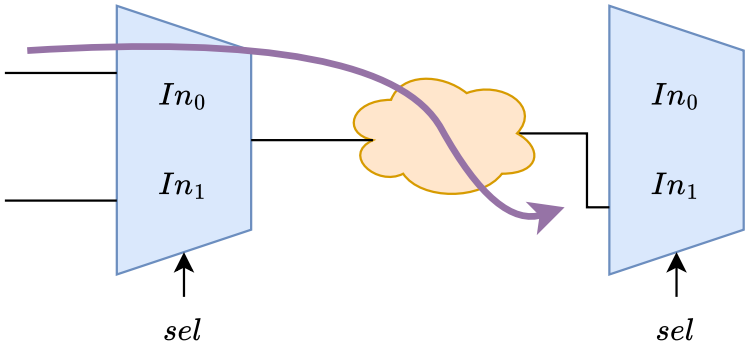
```
set_multicycle_path -setup 3 -from {ff1} -to {ff2}
set_multicycle_path -hold 2  -from {ff1} -to {ff2}
```

Estos comandos configuran todas los *path* entre `ff1` y `ff2` como *path* de 3 ciclos para *setup*, pero ahora la verificación de *hold* se mueve al flanco de lanzamiento del reloj. Al configurar el *hold*, en efecto, estás cancelando el *path* de múltiples ciclos de *hold* que fue establecida por el comando `set_multicycle_path -setup`.




## 2.2. False Path

Un **false path** es un camino que nunca puede ser cronometrado correctamente en el circuito real. Estos *false paths* son aquellos que lógicamente no son factibles, o cuyo análisis de tiempos no es viable. Por ejemplo, los caminos provenientes de un puerto de reset asíncrono (*asynchronous reset port*) suelen configurarse como *false paths*, así como también el path entre 2 FFs de un sincronizador.




En STA, es importante evitar analizar *false paths* para ahorrar tiempo (y área) en la optimización.

```
set_false_path [-setup] [-hold]          \
               [-rise] [-fall]           \
               [-from from_list]         \
               [-to to_list]             \
               [-through through_list]
```