Módulo 1: Problemas básicos de interfase

La metaestabilidad

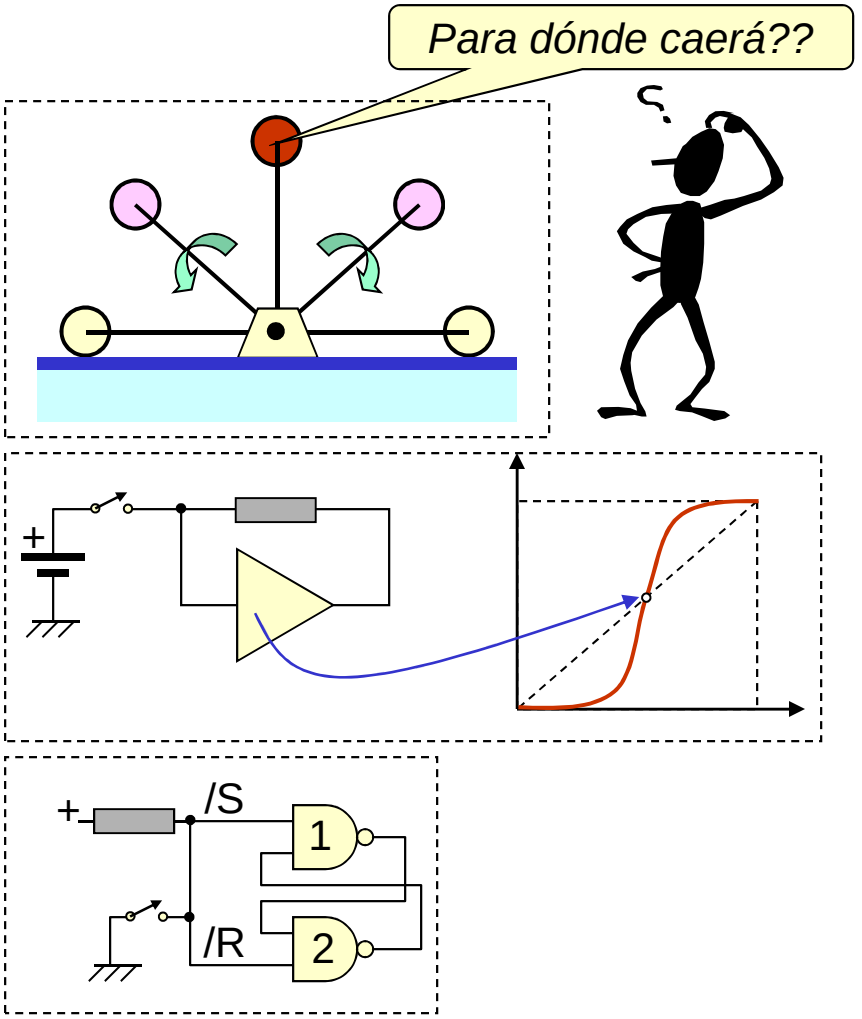
Contenidos del módulo 1

- Qué es la metaestabilidad.
- Influencia en los tiempos de conmutación de dispositivos de memoria.
- Sincronización de señales asíncronicas (estrategias para acotar los eventuales problemas de metaestabilidad).
- Detección sincrónica de transiciones de señales asincrónicas.
- Diseño digital robusto ante fenómenos de metaestabilidad.

EL PROBLEMA básico de interfase: la metaestabilidad

El primer problema a considerar al diseñar una interfase es la metaestabilidad.Hay varios ejemplos que pueden ser usados para comprender el problema:

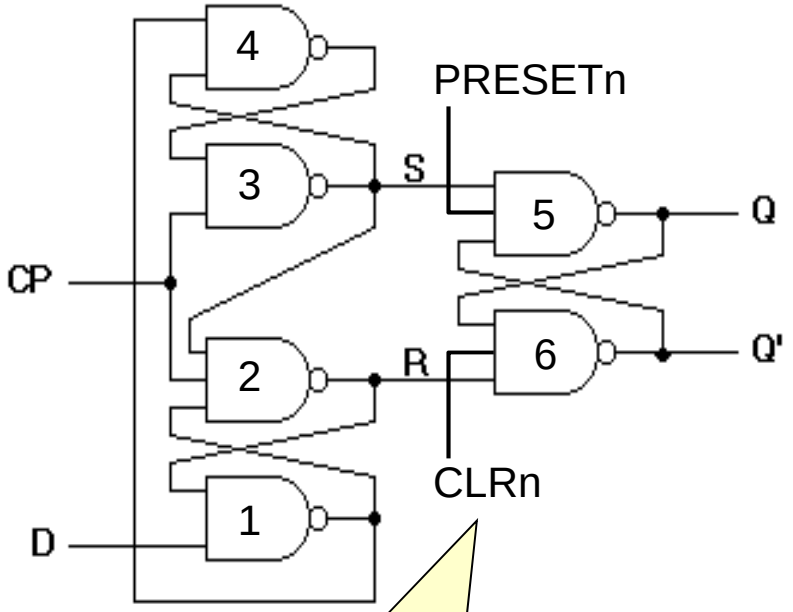
- ✓ el péndulo invertido, en la posición marcada en rojo
- ✓ el buffer NO-INVERSOR realimentado, cuando se lo lleva al potencial de cruce y se lo deja alibre
- ✓ el latch RS, cuando las dos entradas de control (/S y /R) son subidas simultáneamente



Flip flop Edge Triggered

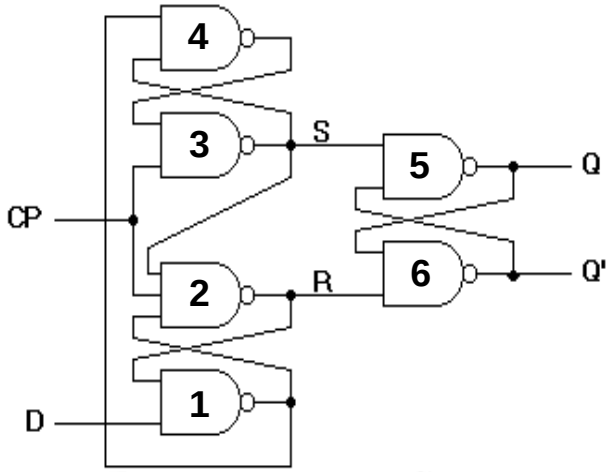
El flip flop activado por flanco repite el problema de metaestabilidad cuando hay simultaneidad entre el cambio del reloj y la entrada de datos

- En el caso de la figura, si la señal de datos (D) va a ‘1’ al mismo tiempo que el reloj (CP) va a ‘1’, el latch formado por las compuertas 1+2 entra en situación metaestable.
- Estos procesos de metaestabilidad están relacionados entre si por el vínculo de la compuerta 3 a la 2 (con dos retardos de compuerta respecto a la señal D), y hacen imprevisible cuál de las señales de control del latch de salida (S y R) será la última activa



Habrá problemas de metaestabilidad referidos a PRESETn y CLRn?

Flip flop Edge Triggered



Name	Type	0 20 40
D	bit	
CP	bit	
s1	bit	
s2	bit	
s3	bit	
s4	bit	
s5	bit	
s6	bit	

```
ENTITY ffd IS
END ENTITY ffd;

ARCHITECTURE a OF ffd IS
  CONSTANT td1, td2, td3, td4 : time := 2000ps;
  CONSTANT td5, td6 : time := 2000ps;
  signal D,CP: bit := '0';
  signal s1,s2,s3,s4,s5,s6 : bit := '0';
BEGIN
  s1 <= NOT(s2 AND D) after td1;
  s2 <= NOT(CP AND s1 AND s3) after td2;
  s3 <= NOT(CP AND s4) after td3;
  s4 <= NOT(s1 AND s3) after td4;
  s5 <= NOT(s3 AND s6) after td5;
  s6 <= NOT(s5 AND s2) after td6;
  D   <= '0', '1' after 100ns;
  CP <= '0', '1' after 100ns;
END ARCHITECTURE a;
```

Simular y verificar otras posibles condiciones de metaestabilidad

Qué sucede si los td[1:6] son todos diferentes?

EL PROBLEMA básico de interfase: la metaestabilidad

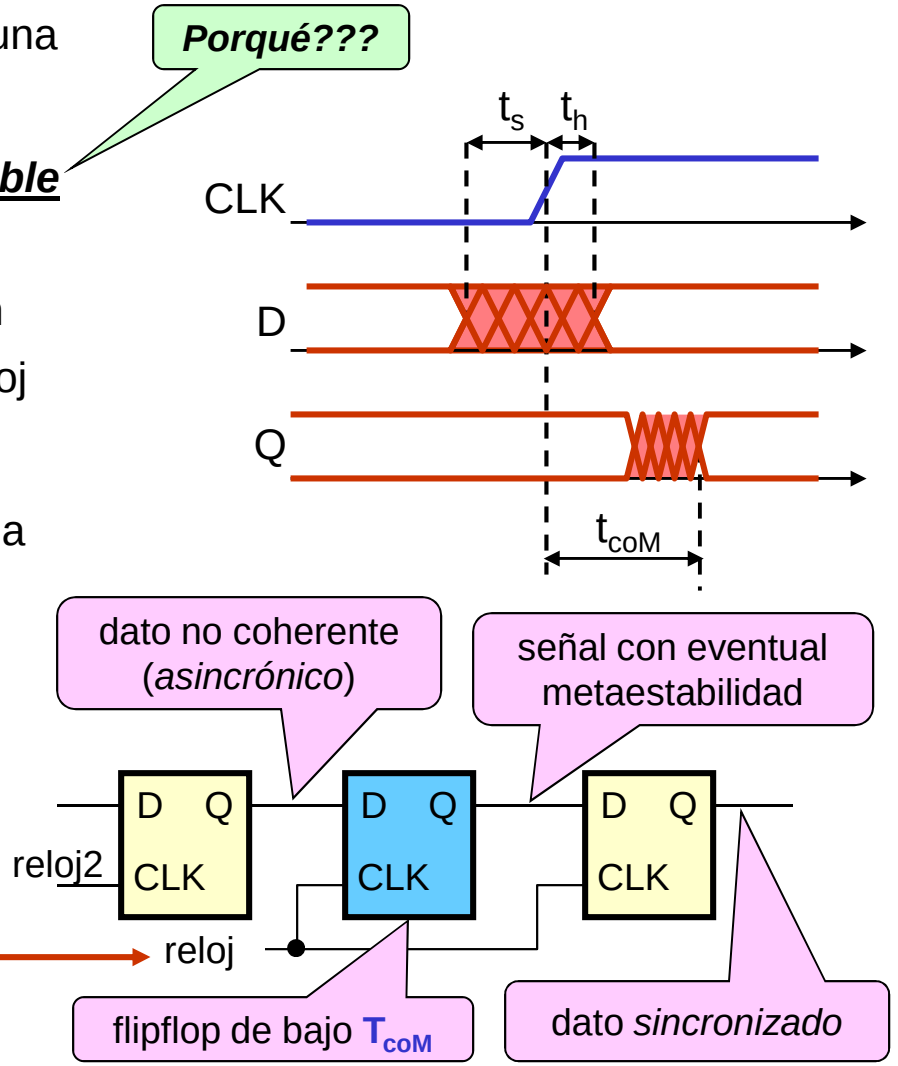
El primer problema a considerar en el diseño de una interfase es la metaestabilidad

La ***METAESTABILIDAD*** es un fenómeno ***inevitable*** producido cuando no se respetan los tiempos de Setup t_s y de Hold t_h en la entrada de datos de un flipflop "edge triggered" respecto a la señal de reloj

Este problema introduce dos efectos:

- incertidumbre sobre el valor que aparece en la salida Q
- un mucho mayor tiempo de propagación clock_to_output (t_{coM}) que el t_{co} cuando los tiempos t_s y t_h han sido respetados

Cuando es inevitable se resuelve mediante dos flipflops que opera con un reloj con $t_{clk} > t_{coM} + t_{su}$



EL PROBLEMA básico de interfase: la metaestabilidad

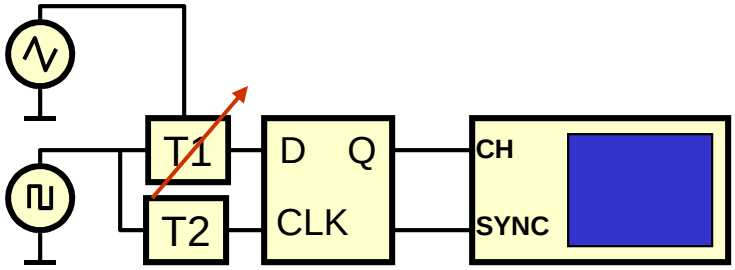
En (*) se detalla un banco de prueba para medir el problema de la metaestabilidad

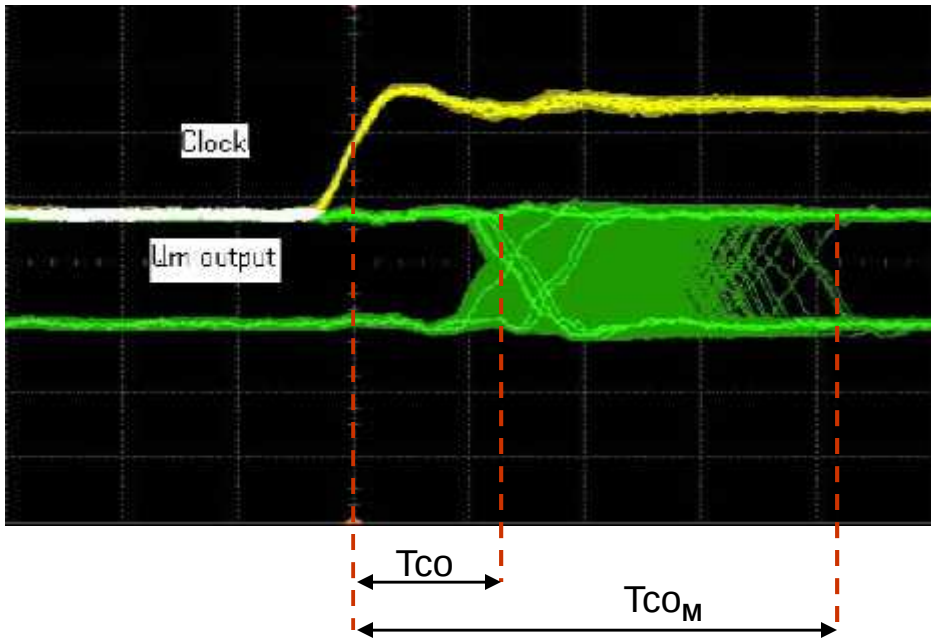
En este banco la salida de un generador es pasada por dos cadenas de demora,

- una fija (T2)
- otra variable (T1), donde T1 puede variar de T1<T2-Tsetup a T1>T2+Thold

Y con ellas se excita un flipflop 74HC74.

Se evidencia cómo el tiempo Tco de algo más de un cuadro se llega a triplicar al ocurrir la metaestabilidad.

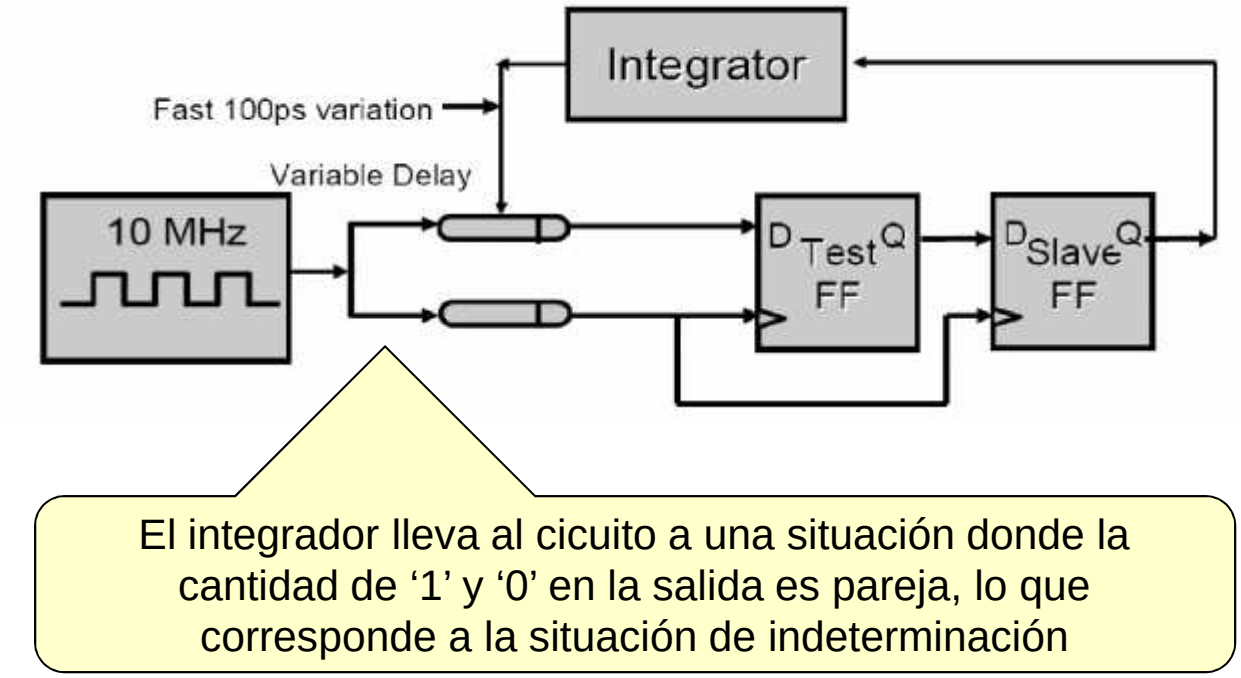




(*) http://www.siue.edu/~gengel/pdf/MeasuringMetastability.pdf

Forzando la metaestabilidad

En (*) se detallan formas de medir la metaestabilidad y estimar el MTBF de sincronizadores



El integrador lleva al cicuito a una situación donde la cantidad de ‘1’ y ‘0’ en la salida es pareja, lo que corresponde a la situación de indeterminación

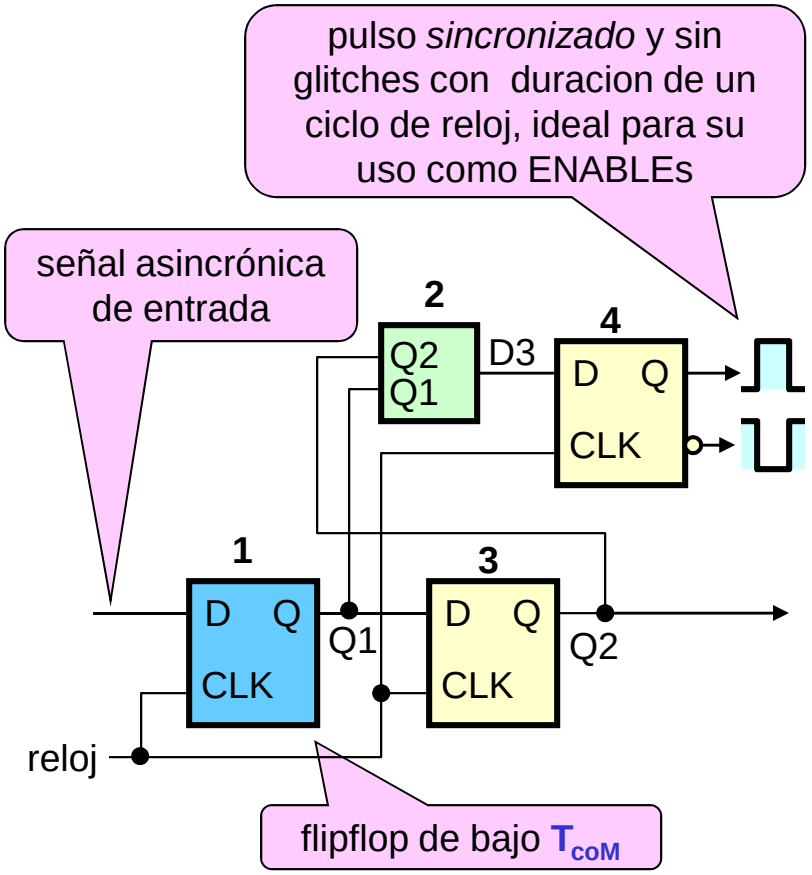
(*) http://tima.imag.fr/conferences/async/Technical_Program/Monday/Session_1/David_Kinniment_talk_1_11h00.pdf

Problemas básicos de interfase: detección de flancos

- En muchos casos una señal externa *asincrónica* debe actuar en uno de sus flancos sobre algún circuito interno como clear, preset o reloj.
- En esos casos se usan circuitos de detección y sincronización de flancos, para adaptar la señal externa al dominio de un único (en lo posible) reloj interno, usándola como clear o preset sincrónico, o como enable.
- Estos sólo sirven si el evento externo es de mucha menor velocidad que el reloj interno
- La ecuación lógica del bloque combinatorio marcado en verde define la operación:
  - deteccion flanco + : **D3 <= Q1 AND NOT Q2**
  - deteccion flanco - : **D3 <= Q2 AND NOT Q1**
  - deteccion de ambos flancos: **D3 <= Q1 XOR Q2**
- La ecuación a cumplir es:

$$T_{coM} + T_{comb} + T_{setup} < T_{reloj}$$

1 2 3

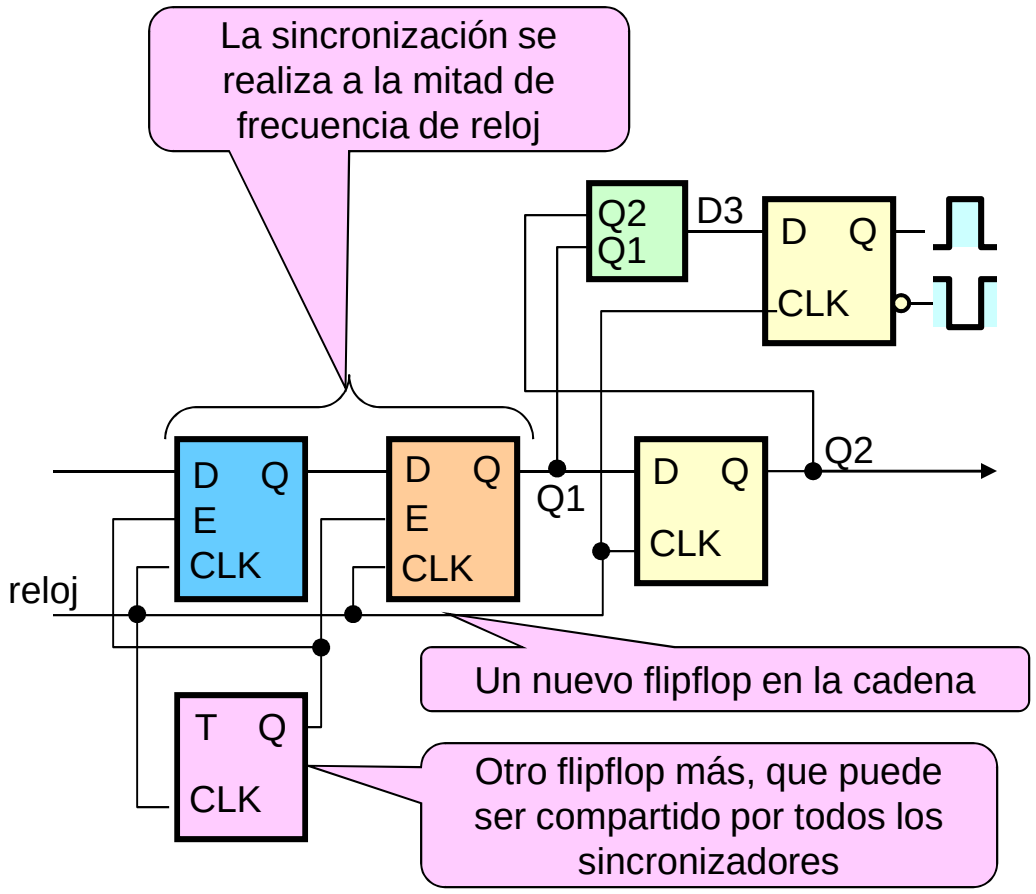


Diseñar el circuito en VHDL y simularlo

Y si no quiero limitar mi frecuencia interna?

- Cómo hago si deseo que Fclk sea más rápido, para no tener que esperar TcoM+Tsetup ???
- Aprovecho que los flipflops cuentan con señal de Enable
- Y a los dos primeros flipflops del sincronizador los habilito cada DOS ciclos de reloj
- La señal de salida del flipflop celeste dispone ahora de 2xTreloj para estabilizarse
- Es decir, ahora debe cumplirse:

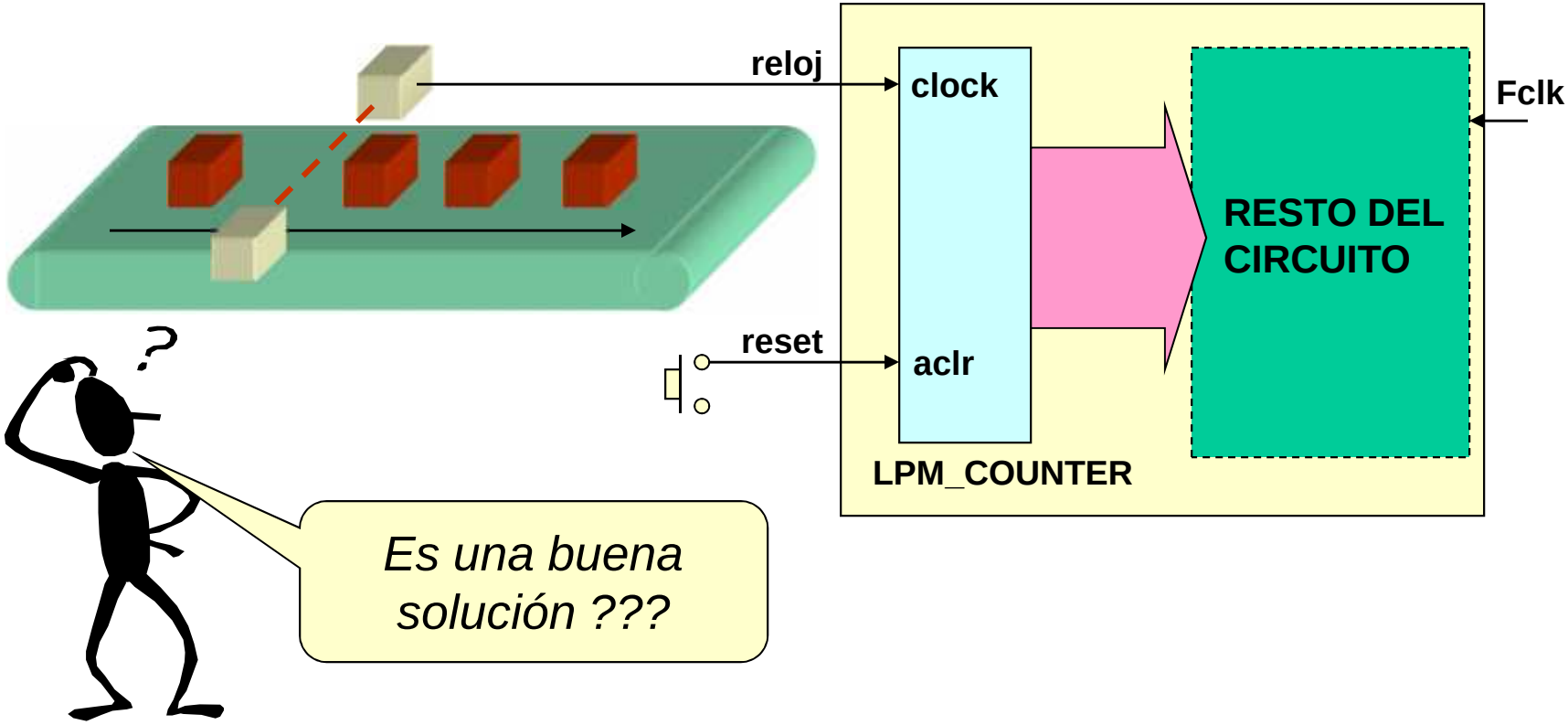
$$T_{coM} + T_{setup} < 2 \times T_{reloj}$$



Diseñar el circuito en VHDL y simularlo

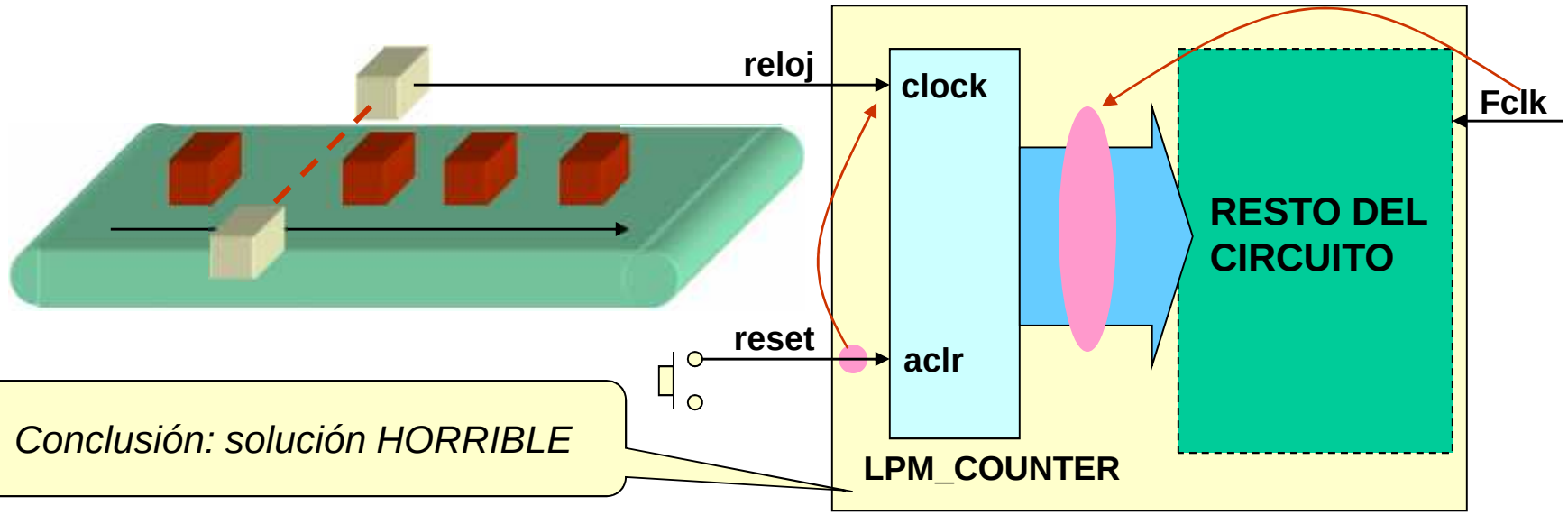
Un ejemplo: una solución “naif” y la solución realista

- Tengo una señal de reset y una señal de reloj externas, y deseo contar los pulsos de reloj que se producen y hacer accesible esa cuenta a circuitos internos que operan a mucha mayor velocidad



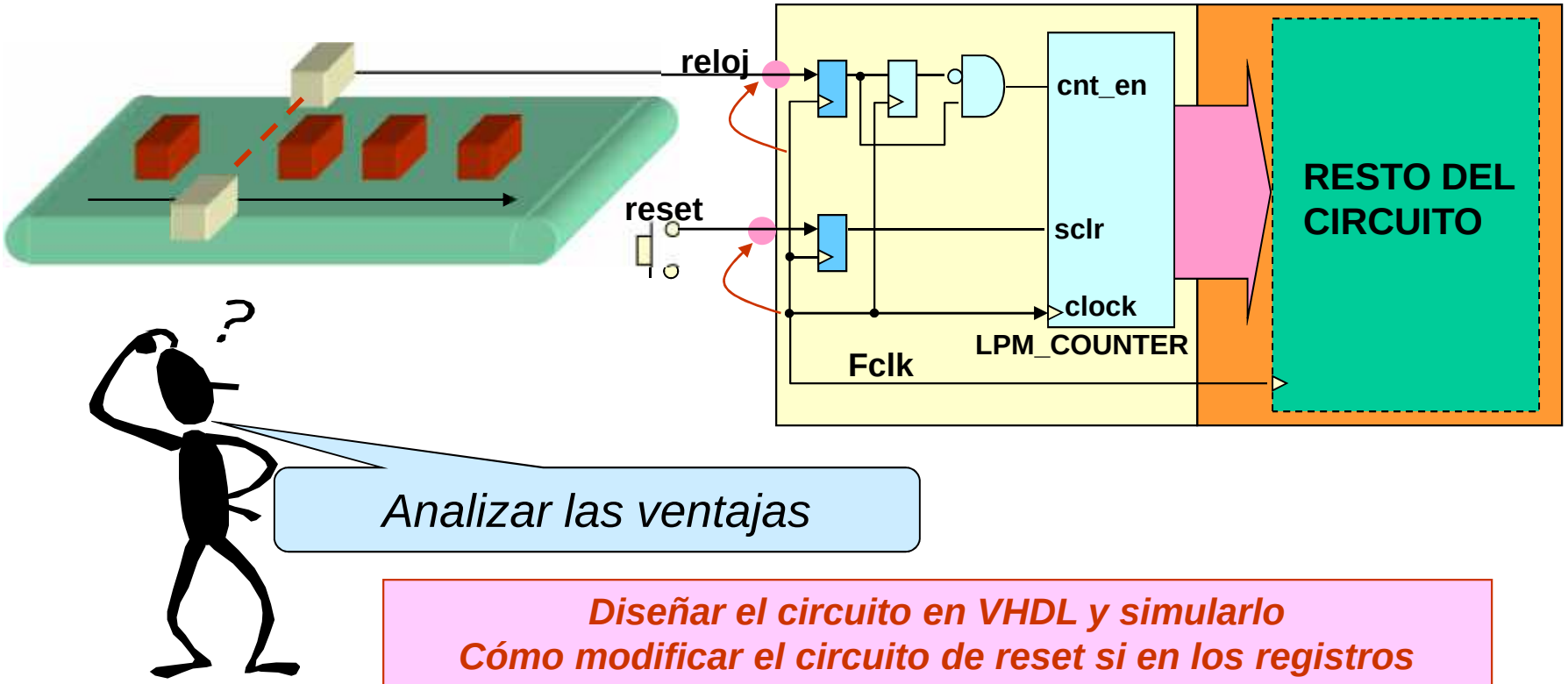
Un ejemplo: una solución “naif” y la solución realista

- Problemas:
  1. Reset (aclr) es asincrónica respecto a Reloj (clock) del LPM_COUNTER, por lo que su accionamiento sin respetar Tsetup y Thold puede generar comportamientos indeseados en el contador
  2. Las salidas del contador son asincrónicas respecto a Fclk, por lo que el uso de las salidas del contador por el RESTO DEL CIRCUITO está sujeto a problemas de metaestabilidad



Un ejemplo: una solución “naif” y la solución realista

La solución correcta elimina las metaestabilidades y asincronismos

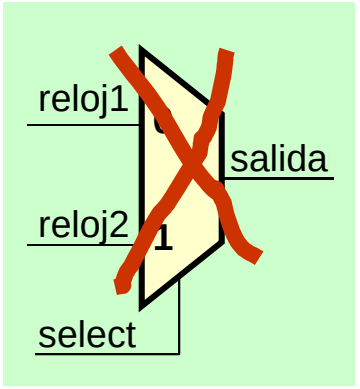


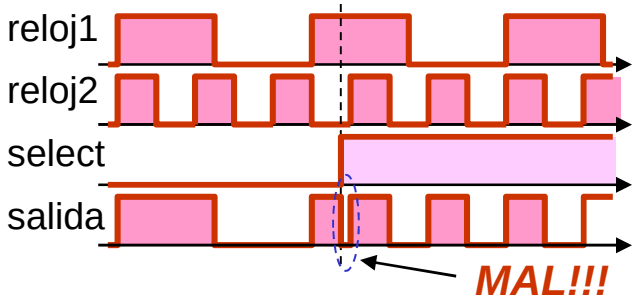
Diseñar el circuito en VHDL y simularlo
Cómo modificar el circuito de reset si en los registros internos se desea usar los CLRn asincrónicos??

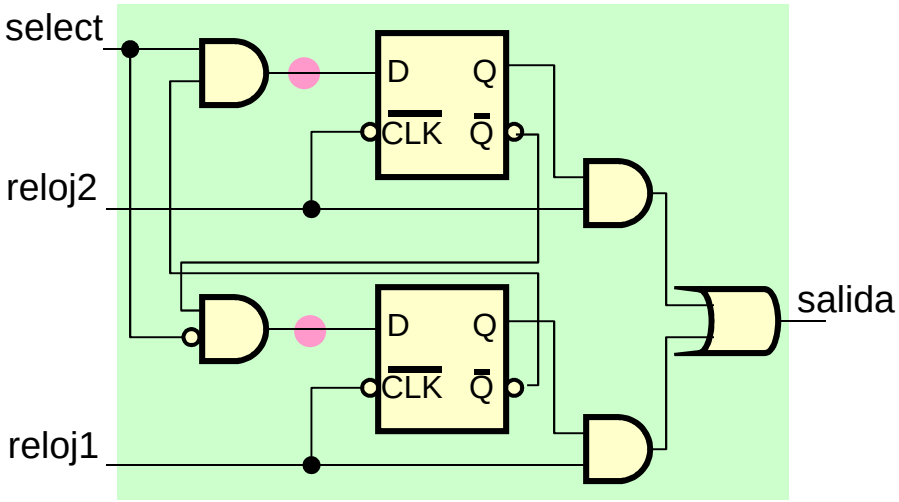
Problemas básicos de interfase: conmutación de relojes coherentes

- Cuando en una aplicación debe conmutarse entre dos posibles señales de reloj, el uso de un simple multiplexer no es conveniente, debido al riesgo de generar pulsos de duración T_{ON} o T_{OFF} inaceptables
- Una solución robusta requiere que la señal de control de la conmutación ***select*** se active en el momento indicado
- Si los relojes son coherentes (tienen relacional racional entre ellos) el circuito mostrado garantiza que la conmutación se inicie al comenzar el tiempo T_{OFF} del reloj inicial y que la salida permanezca en cero al menos el tiempo T_{OFF} de la señal final
- Esto presume que ***select*** respeta los tiempos t_s y t_h respecto a los relojes

Ref: techniques to make clock switching glitch free. Rafey Mahmud, EEdesign, 2003

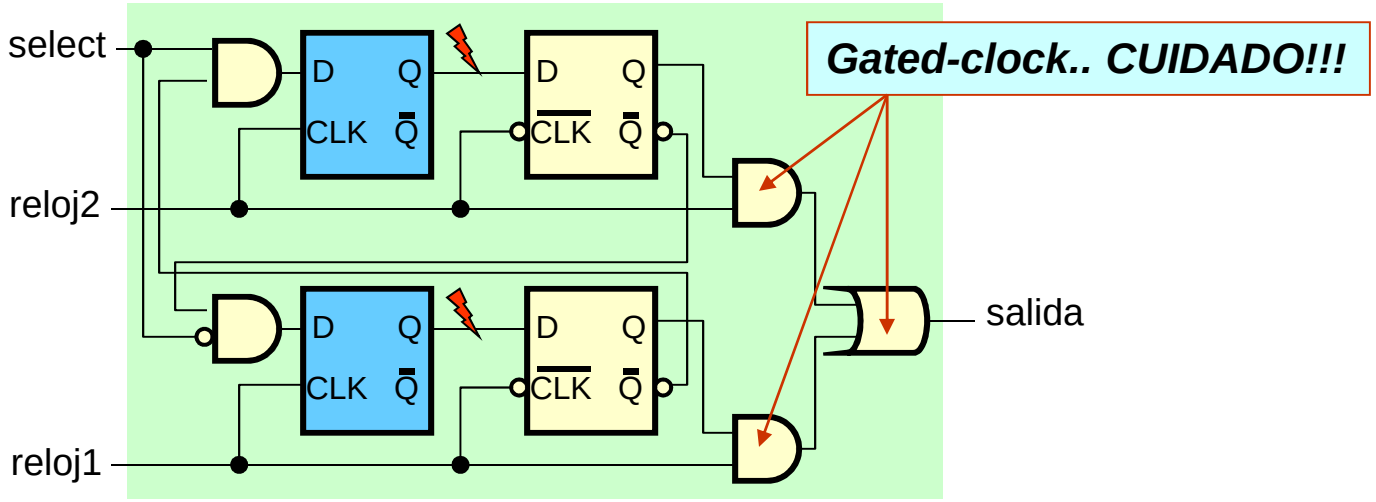






Diseñar el circuito en VHDL y simularlo

Conmutación de relojes no coherentes



- Cuando en una aplicación debe conmutarse entre dos posibles señales de reloj no coherentes, es imposible garantizar que la señal select no viole los tiempos t_s y t_h
- Una solución robusta requiere que la conmutación se sincronize
- Por eso es necesario agregar dos flipflops donde $t_{coM}+t_s$ sea menor que el tiempo T_{ON} de cualquiera de los relojes

Ref: techniques to make clock switching glitch free. Rafey Mahmud, EEdesign, 2003

Diseñar el circuito en VHDL y simularlo