Contadores — Electrónica Digital

Grado en Ing. Electrónica, Robótica y Mecatrónica — Universidad de Sevilla

Apuntes propios (tema 09). Estilo *worked-example primero*: cada concepto se acompaña de su **cronograma** (clock + bits Q_i) generado con `matplotlib` y de su **circuito** dibujado con `schemdraw`.

Índice

---

1. Qué es un contador

> **Definición.** Un contador genera **números naturales en secuencia**, avanzando **uno por cada ciclo de reloj**.

Es un circuito **secuencial síncrono** construido con biestables (flip-flops). Con n biestables se cuenta de 0 a $2^n - 1$ (módulo 2^n).

Ejemplo: contador binario de 3 bits ($Q_2 Q_1 Q_0$), secuencia $000 \to 001 \to \cdots \to 111 \to 000$:

paso	Q_2	Q_1	Q_0	decimal
0	0	0	0	0
1	0	0	1	1
2	0	1	0	2
3	0	1	1	3
4	1	0	0	4
5	1	0	1	5
6	1	1	0	6
7	1	1	1	7

Observa el patrón: Q_0 conmuta cada flanco, Q_1 cada 2, Q_2 cada 4. Cada bit conmuta a la **mitad de frecuencia** del anterior (de ahí su uso como divisor de frecuencia).

Utilidad de dibujo de cronogramas

Definimos una función reutilizable que dibuja un cronograma digital: una señal de reloj y los bits Q_i. La usaremos durante todo el notebook.

```
draw_timing lista. Flanco activo marcado en verde.
```

Generador de la cuenta binaria

Función auxiliar que produce, para cada bit, su secuencia de valores a lo largo de los pasos. Sirve para ascendente y descendente.

```
Secuencia ascendente 3 bits: [0 1 2 3 4 5 6 7]
```

---

2. Contadores asíncronos (ripple)

> **Definición.** En un contador **asíncrono** la señal de reloj **NO es común** a todos los biestables. Cada biestable se dispara con la salida del anterior.

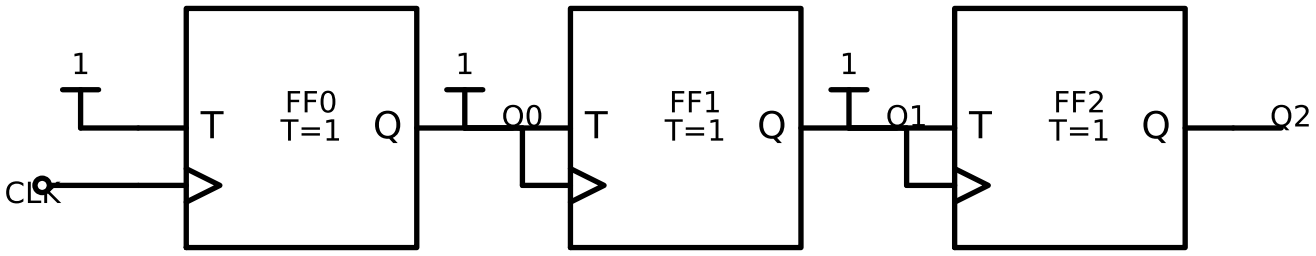
Se conectan en cascada biestables **T** (o JK con $J = K = 1$, que conmutan en cada flanco):

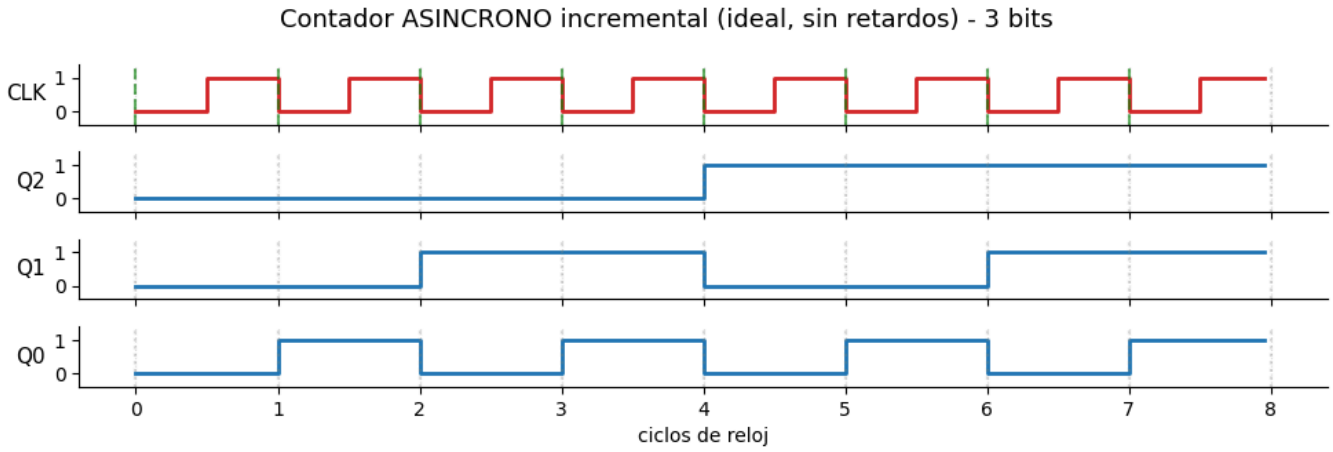
J	K	Q_{n+1}	→	T	Q_{n+1}
0	0	Q_n		0	Q_n
0	1	0		1	$\overline{Q_n}$
1	0	1			
1	1	$\overline{Q_n}$			

Asíncrono incremental (FF sensibles a flanco de bajada):

- Q_0 se conecta al reloj externo.
- Cada Q_i alimenta el reloj de Q_{i+1}.
- **Condición de cambio:** flanco de **bajada** del bit anterior.

$$000 \to 001 \to 010 \to 011 \to 100 \to 101 \to 110 \to 111 \to 000$$



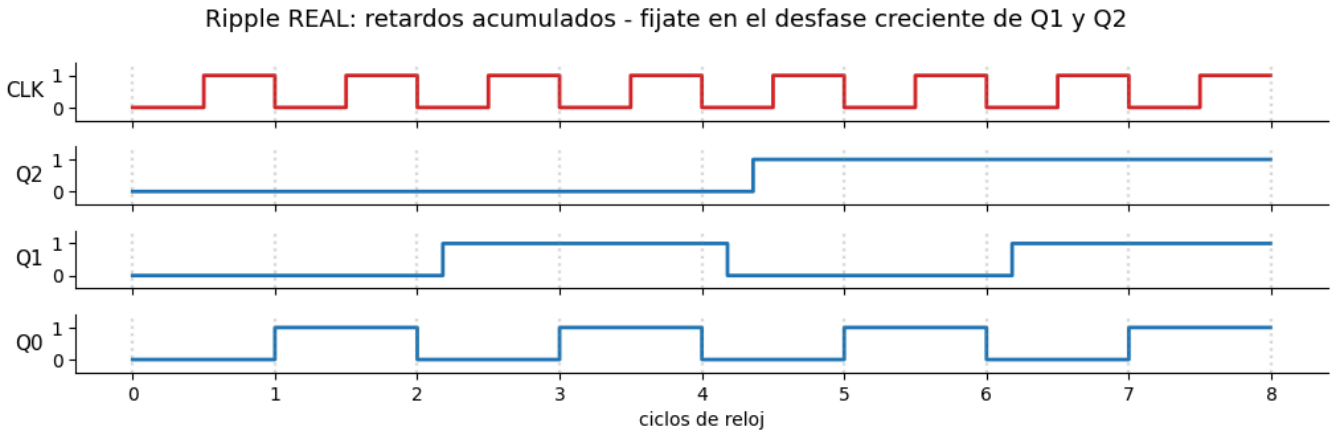


El problema de la asincronía: acumulación de retardos

El cronograma anterior es el **ideal**. En la realidad cada biestable tiene un retardo de propagación t_{pd}. Como Q_0 dispara a Q_1, Q_1 a Q_2, etc., los flancos **NO ocurren simultáneamente**: el retardo se **acumula** ("ripple", ondulación).

> En el peor caso (p.ej. $111 \rightarrow 000$) el bit más significativo cambia tras $n \cdot t_{pd}$. Durante ese transitorio aparecen **estados intermedios falsos** (glitches), que pueden disparar lógica aguas abajo.

Vamos a visualizar el efecto exagerando t_{pd}.



Observa cómo Q_1 va retrasado respecto a Q_0, y Q_2 aún más. En la transición $111 \rightarrow 000$ el sistema pasa fugazmente por $110, 100, 000$: **estados espurios**. Por esto:

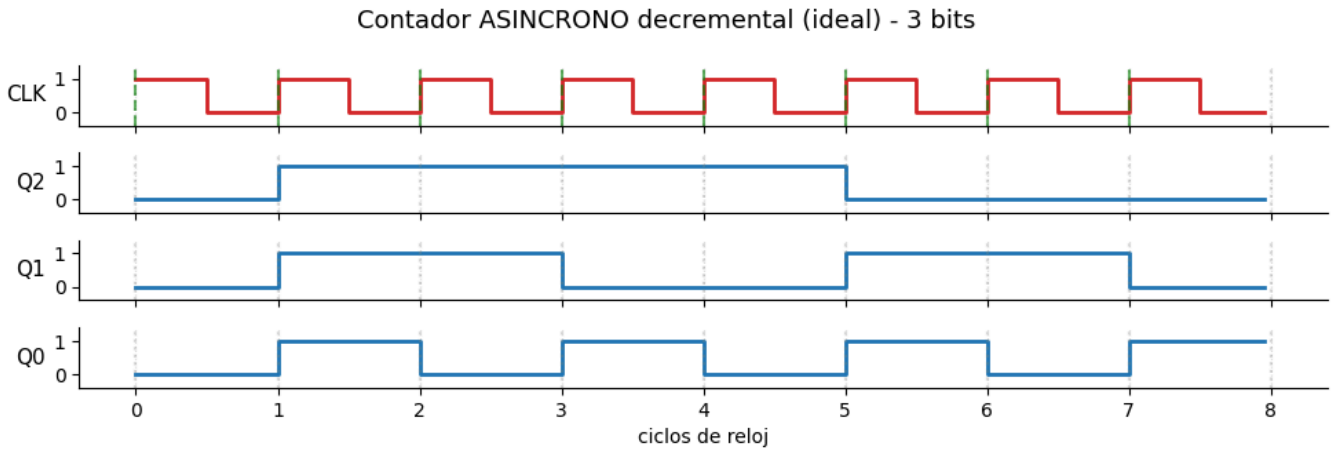
> **Conclusión del tema (literal de los apuntes): ¡No usar contadores asíncronos!**

Variantes asíncronas (para completar)

- **Asíncrono incremental con FF en flanco de subida:** condición de cambio = flanco de **subida del negado** ($\overline{Q}$) del bit anterior. Equivale a tomar el reloj de $\overline{Q_i}$.
- **Asíncrono decremental (descendente):** secuencia $000 \to 111 \to 110 \to \cdots \to 001 \to 000$. Condición = flanco de **subida** del bit anterior. (Si los FF son sensibles a flanco de bajada, se usan las señales $\overline{Q}$.)
- **Asíncrono reversible:** un selector de **Modo** elige entre tomar Q (subir) o $\overline{Q}$ (bajar). Problema: **glitch de cuenta al cambiar de modo**; se mejora con una señal **CUENTA** que bloquea el contador cuando vale 0. Aun así, sigue siendo asíncrono → **evitar**.

Cronograma del decremental (ideal):

```
Secuencia descendente: [0 7 6 5 4 3 2 1]
```



---

3. Contadores síncronos

> Solo usaremos **circuitos síncronos**: **todos** los biestables comparten **la misma** señal de CLK, que se usa *únicamente* para sincronizar. Esto elimina la acumulación de retardos: todos los bits que deban cambiar lo hacen **a la vez**.

Idea de diseño con biestables T: un bit Q_i debe conmutar cuando **todos los bits inferiores están a 1** (ascendente). Por tanto la entrada T_i es:

$$T_i = Q_0 \cdot Q_1 \cdots Q_{i-1} = \prod_{j=0}^{i-1} Q_j$$

con $T_0 = 1$ (el LSB conmuta siempre).

Síncrono **paralelo**

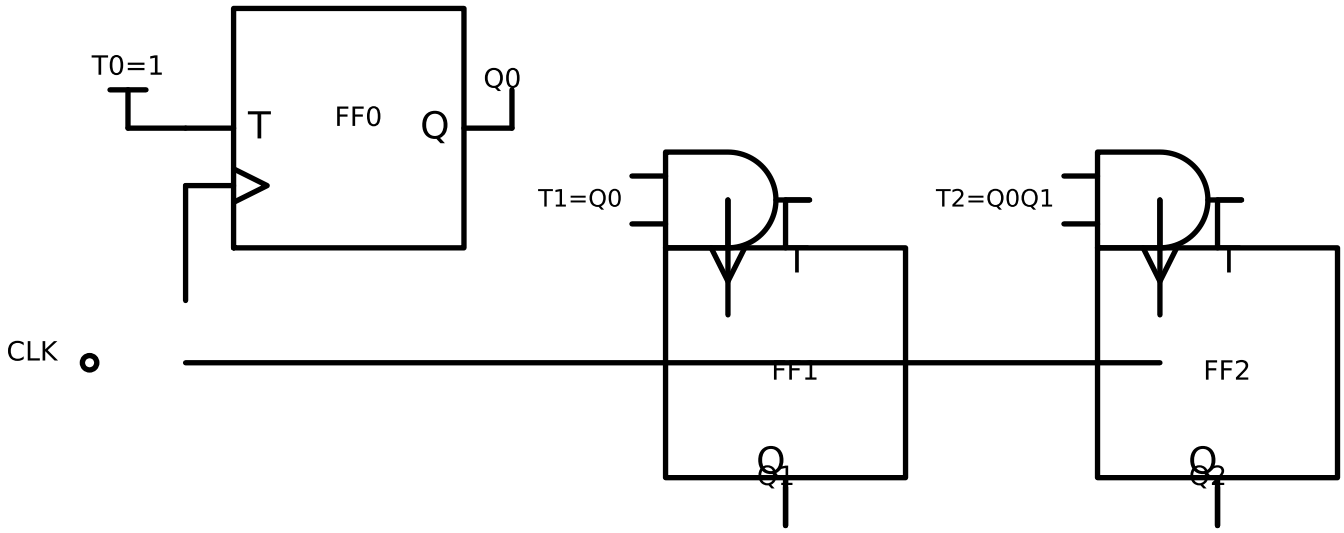
Cada T_i se calcula con una puerta AND que toma **directamente** todos los Q_j inferiores. Rápido, pero las AND crecen en nº de entradas con i.

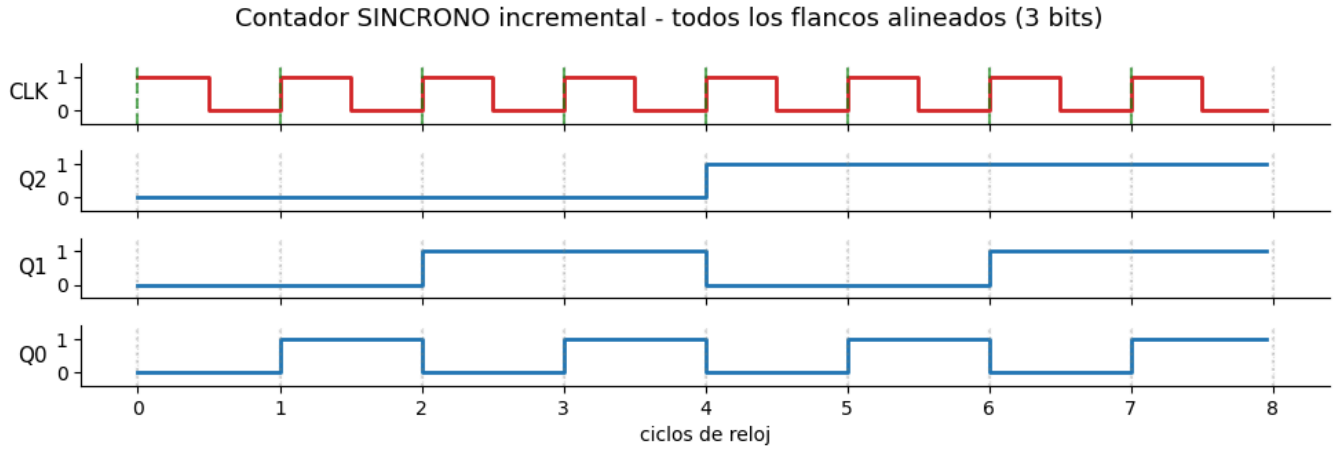
Síncrono **serie**

Se reaprovecha el cálculo anterior encadenando:

$$T_i = T_{i-1} \cdot Q_{i-1}$$

Se simplifica la circuitería (AND de 2 entradas en cascada) a costa de un pequeño retardo combinacional en cadena.





Comparación clave: en el síncrono, en $111 \rightarrow 000$ los tres bits caen **exactamente** en el mismo flanco de reloj (mismo t_{pd} medido desde el reloj común). No hay estados espurios acumulativos como en el ripple.

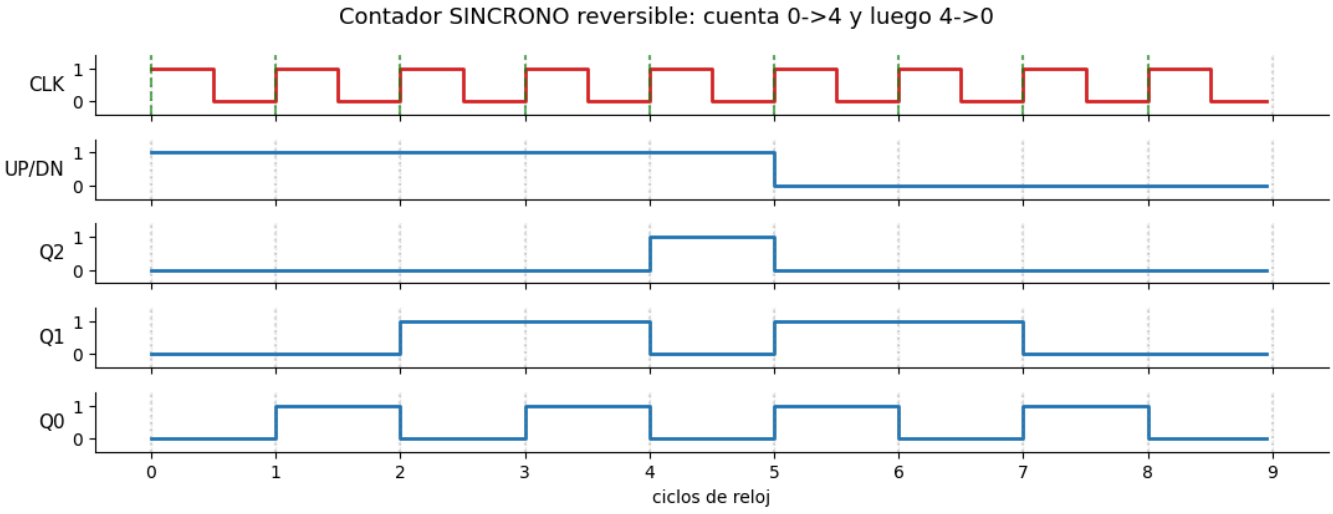
Síncrono **decremental**

Misma estructura pero la condición de conmutación es "todos los bits inferiores a **0**", lo que equivale a usar las señales $\overline{Q}$: $T_i = \prod_{j<i} \overline{Q_j}$.

Síncrono **reversible**

Una señal $UP/\overline{DOWN}$ selecciona, mediante multiplexores, si cada T_i usa los Q_j (subir) o los $\overline{Q_j}$ (bajar). Al ser síncrono, no hay glitch de cambio de modo si el control se respeta en el flanco.

```
Secuencia reversible: [0 1 2 3 4 3 2 1 0]
```



---

4. Contadores de módulo N

Un contador de n bits cuenta hasta 2^n. Para un **módulo N** arbitrario (p.ej. N=10, *década*) hay que **forzar el retorno a 0** al alcanzar N.

4.1 Reset asíncrono — ¡NO HACER!

Usar el `CLR` asíncrono de los biestables (detectando la cuenta = N con una AND que ataca el clear) provoca **transitorios**: el estado N aparece fugazmente antes de borrarse → glitch en la salida. Además depende de retardos. **¡NO HACER!**

Tampoco vale "sincronizar el reset con un biestable extra": sigue siendo un parche.

4.2 Reset **síncrono** — forma correcta

Hay que actuar sobre las **entradas síncronas** (T, J/K, D) mediante lógica combinacional, de modo que **en el siguiente flanco** la salida sea 0. Idea general (válida para cualquier secuencial):

> Mira qué entrada necesita cada biestable para que tras el flanco $Q_i = 0$.

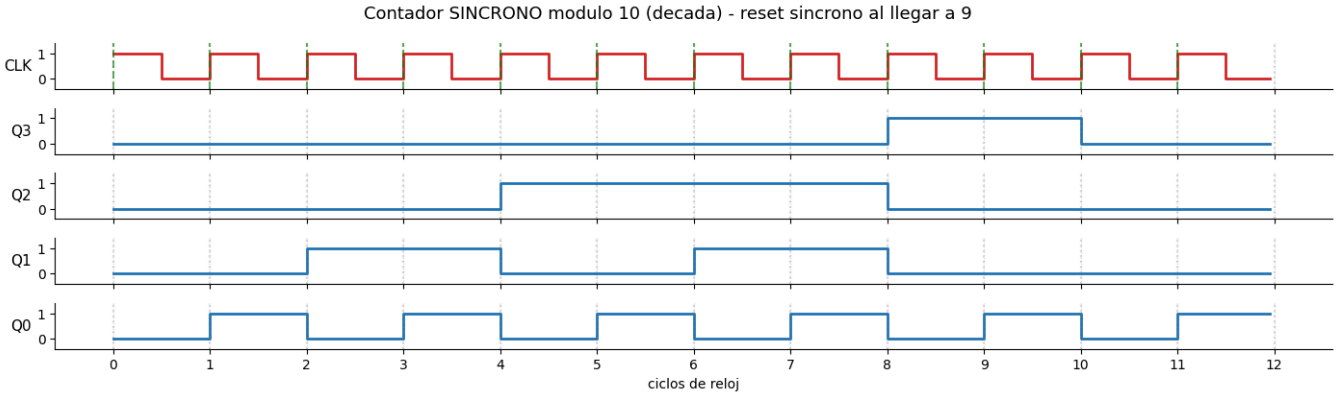
Con biestables **T** y `reset` activo a **nivel bajo**:

- Si `reset = 0` (queremos resetear): debe quedar $Q_i^+ = 0$.
  - Si ahora $Q_i = 0$ → mantener → $T_i = 0$.
  - Si ahora $Q_i = 1$ → conmutar → $T_i = 1$.
  - En ambos casos: $\boxed{T_i = Q_i}$.
- Si `reset = 1` (no resetear): la cuenta continúa con su T_i normal.

Esto se implementa con un **multiplexor**/lógica a la entrada de cada FF (celda básica de contador con reset síncrono). La misma idea sirve para **preset** (cargar un valor, útil en decrecientes) e **inhibición** síncrona (congelar la cuenta).

Ejemplo: contador de módulo 10 (década), $0 \to 9 \to 0$, reset síncrono al detectar 9 (o equivalentemente carga de 0).

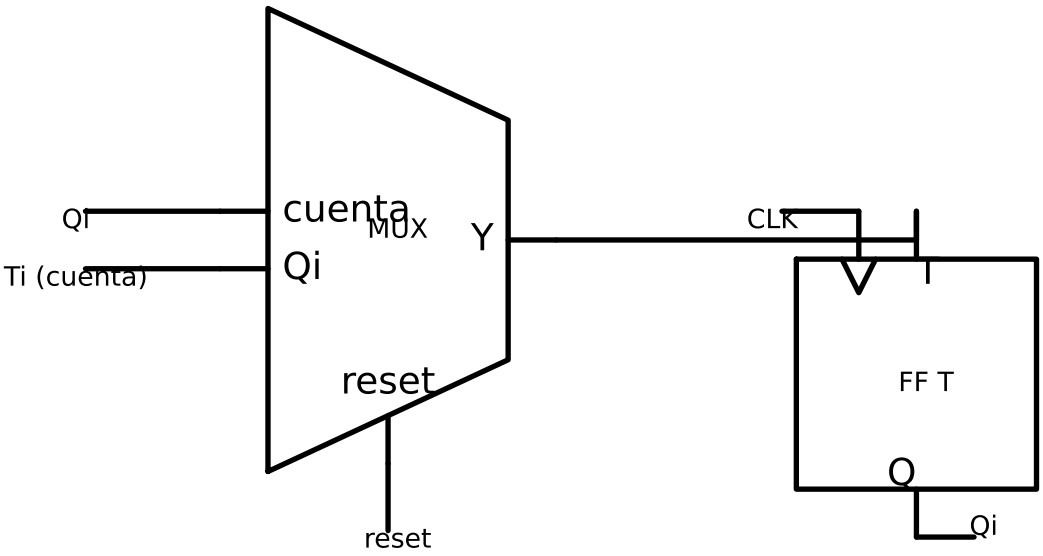
```
Secuencia modulo 10: [0 1 2 3 4 5 6 7 8 9 0 1]
```



Nota cómo tras $1001_2 = 9$ la cuenta vuelve a 0000 en el flanco siguiente, **sin** pasar por 10..15. Los estados $1010..1111$ nunca aparecen en la salida (no hay glitch porque el reset es síncrono).

Celda básica con reset síncrono (esquema)

Se elige, con un **multiplexor** gobernado por `reset`, entre la entrada normal de cuenta (T_i) y la condición de reset (Q_i), y el resultado ataca la entrada T del biestable.



---

5. Aplicaciones y CI comerciales

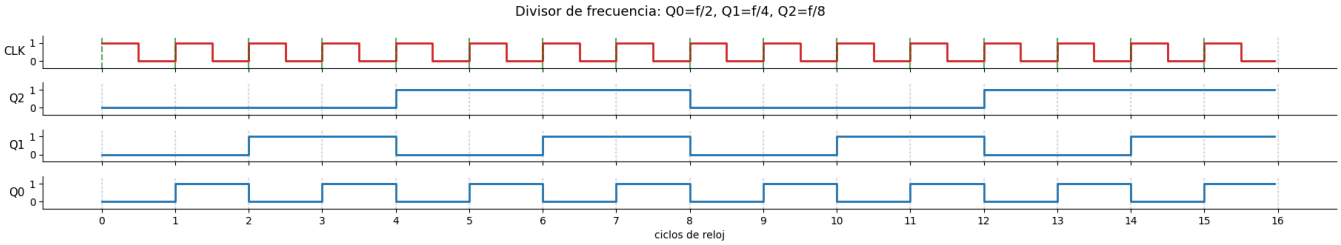
Aplicaciones típicas de un contador:

- Contador de **eventos / sucesos**.
- **Divisor de frecuencia** (cada bit divide por 2; el bit i da $f_{CLK}/2^{i+1}$).
- **Multiplexación** de información en el tiempo (selección de canal).
- **Medición del tiempo** de un evento.
- Generación de **números aleatorios** (con LFSR).
- Generación de **pulsos de duración definida**.

CI comerciales (familia 74xx):

CI	Tipo	Clear	Carga
74160	década síncrono asc.	asíncrono	asíncrona (paralela)
74161	4 bits síncrono asc.	asíncrono	—
74162	década síncrono asc.	**síncrono**	—
74163	4 bits síncrono asc.	**síncrono**	—
74168/190	década reversible	—	asíncrona (paralela)
74169/191	4 bits reversible	—	señal única UP/DOWN
74192	década reversible	—	relojes independientes UP/DOWN
74193	4 bits reversible	asíncrono	asíncrona, relojes UP/DOWN indep.

Demostración: divisor de frecuencia. Cada bit de un contador es media frecuencia del anterior.



---

6. Skew y jitter de reloj

En los contadores síncronos *suponíamos* que el reloj llega a todos los biestables a la vez. En la práctica no es así.

Skew (sesgo)

> El **skew** es la **diferencia entre los instantes de llegada** del flanco de subida del reloj a los distintos biestables.

Causa: los tres (o N) caminos físicos que recorre la señal de reloj hasta cada FF son **diferentes** (longitudes/cargas distintas). Aunque todos estén "conectados al mismo reloj", el flanco no llega simultáneamente. Un skew excesivo puede hacer que el sistema **no registre los datos correctamente** (viola tiempos de setup/hold).

Jitter (fluctuación)

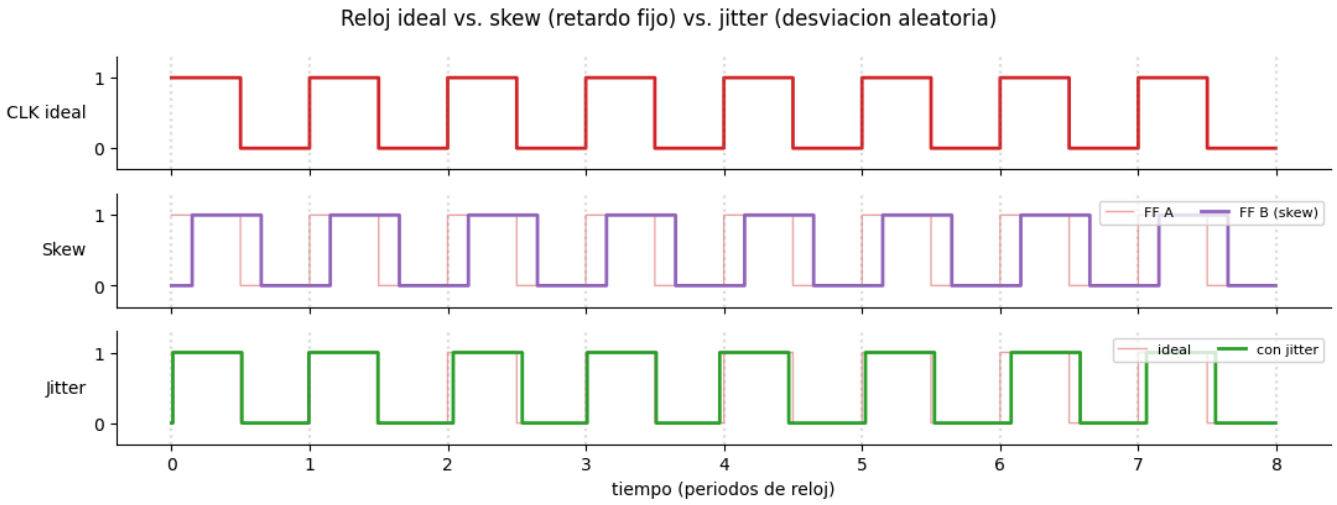
> El **jitter** es la **desviación de un reloj respecto de su posición ideal** en el tiempo.

Causas: ruido **electromagnético**, ruido **térmico**, variación en las **fuentes de alimentación**, **condiciones de carga**.

Tipos de medida de jitter:

- **Average Period Jitter**: diferencia *promedio* entre el periodo medido y el ideal.
- **Cycle-to-Cycle Jitter**: diferencia entre periodos de dos ciclos **consecutivos**.
- **Period Jitter**: desviación del instante de transición tras un número **grande** de ciclos.
- **Time Interval Error (TIE)**: desviación de un flanco respecto a una **posición de referencia**.

Visualizamos un reloj ideal vs. uno con skew y otro con jitter:



Resumen / chuleta de examen

Concepto	Clave
Asíncrono (ripple)	Reloj NO común; cada FF disparado por el anterior. Retardos se **acumulan** → glitches. **NO USAR.**
Condición cambio asínc. incremental	flanco de **bajada** del bit anterior (FF flanco bajada)
Síncrono	Reloj **común** a todos. Bits cambian a la vez. $T_i = \prod_{j<i} Q_j$
Síncrono **paralelo**	AND directa de todos los Q_j inferiores (rápido, ANDs grandes)
Síncrono **serie**	$T_i = T_{i-1} \cdot Q_{i-1}$ (menos hardware)
Decremental	usar $\overline{Q}$ en vez de Q
Reversible	MUX selecciona $Q/\overline{Q}$ según UP/DOWN
Módulo N	Reset **asíncrono** → **NO** (glitch). Reset **síncrono** → **SÍ**: con T, $T_i = Q_i$ cuando reset activo
Skew	diferencia de **llegada** del flanco a cada FF (caminos distintos)
Jitter	desviación temporal del flanco (ruido EM/térmico, alimentación, carga)

Regla de oro: *solo circuitos síncronos*. Para módulo N y reset/preset/inhibición, actuar **sobre las entradas síncronas** de los biestables, nunca sobre el clear asíncrono.

Apuntes propios — Tema 09, Electrónica Digital (GIERM, US). Fuente: diapositivas ED9-Contadores.