Módulo 4:

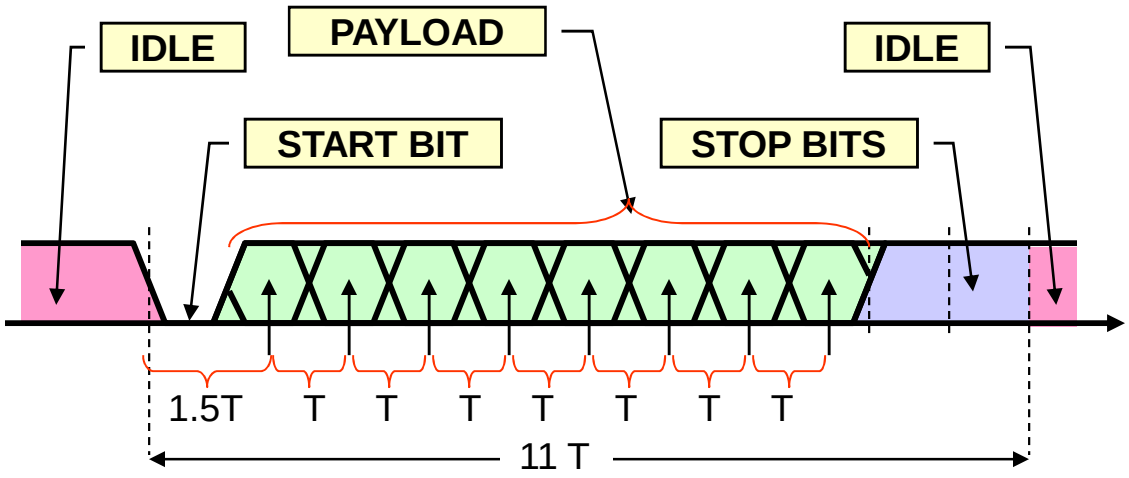
Circuitos de recuperación de reloj

Contenidos del módulo 4

- Posibles soluciones en función de la relación entre el reloj de datos y el reloj del sistema.
- Sistemas tipo serie asincrónico
- Start
- Sistemas tipo serie sincrónico
- PLL digital
- Encoding

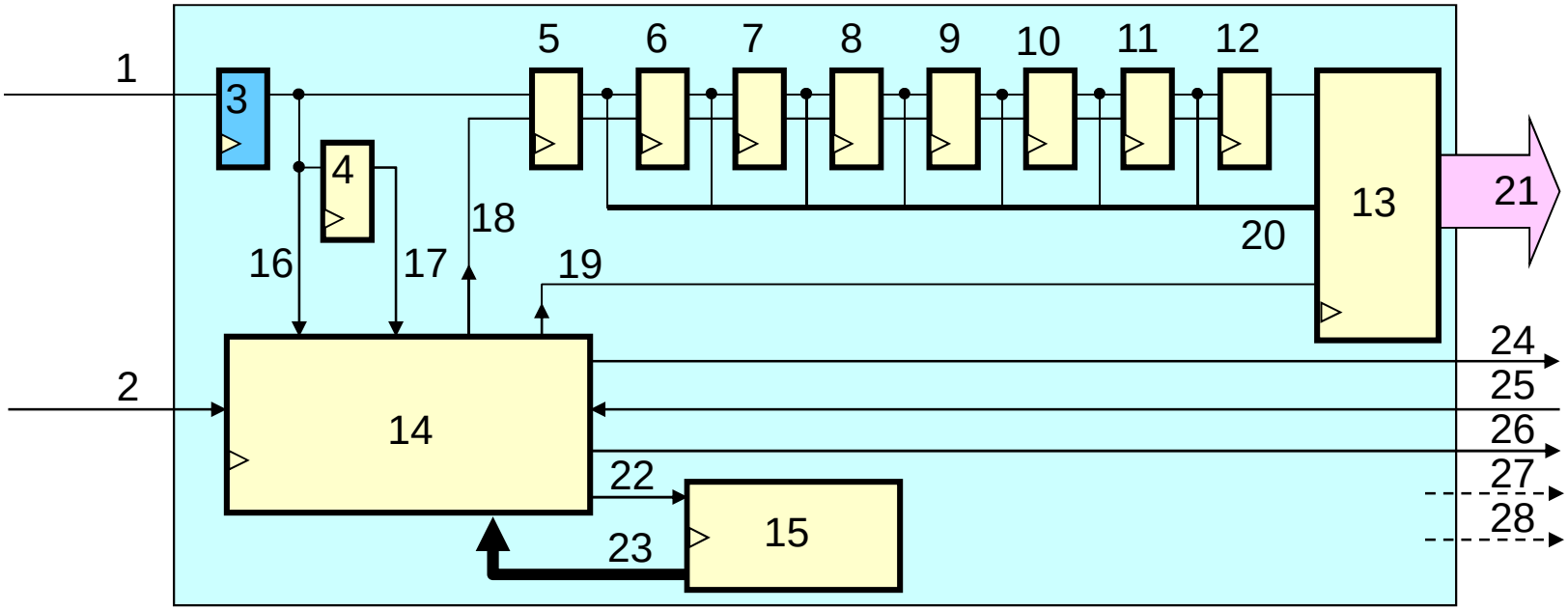
Procesos básicos de recuperación de reloj

- Un método simple de recuperación de reloj se emplea en las UART
- Se fuerza una transición desde un estado pasivo en “1” a “0” insertando un bit de arranque (***START BIT***)
- Se transmiten de 7 a 9 bits de datos
- Y se garantiza volver al estado pasivo insertando uno a dos bits de parada (***STOP BIT***) en “1”.
- En la recepción, a partir del flanco negativo del ***START BIT*** se espera 1.5 tiempos de bit para muestrear el primer dato, y luego un tiempo de bit para cada bit sucesivo
- Los bits de ***START*** y ***STOP*** reducen el aprovechamiento del canal
- Pero el error entre relojes puede ser de más del 5% (menor a 1/17)



- El método de transmisión empleado es llamado **NRZ (Non Return to Zero)**
- Según cuáles sean los datos a transmitir el valor medio de la señal varía
- No son aptos para su uso en enlaces donde se pierda la componente DC de la señal
- Cuanto más bits contiene el PAYLOAD mayor debe ser el apareamiento entre los relojes del transmisor y del receptor

Cómo sería la etapa receptora de una UART?

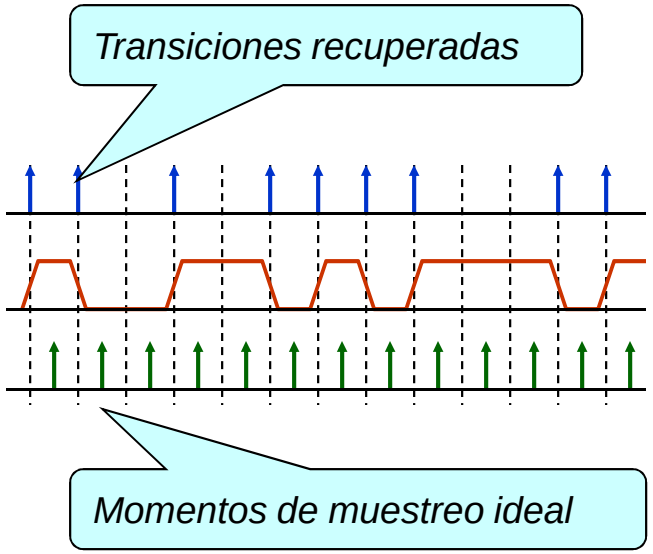


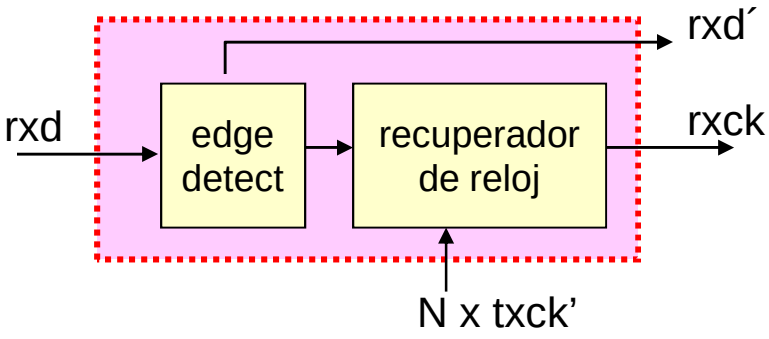
Diseñar el circuito en VHDL y simularlo
Asegurar la captura en el centro de Tbit

Asignar funcionalidades a las señales y objetos numerados de 1 a 26

Circuitos de recuperación de reloj

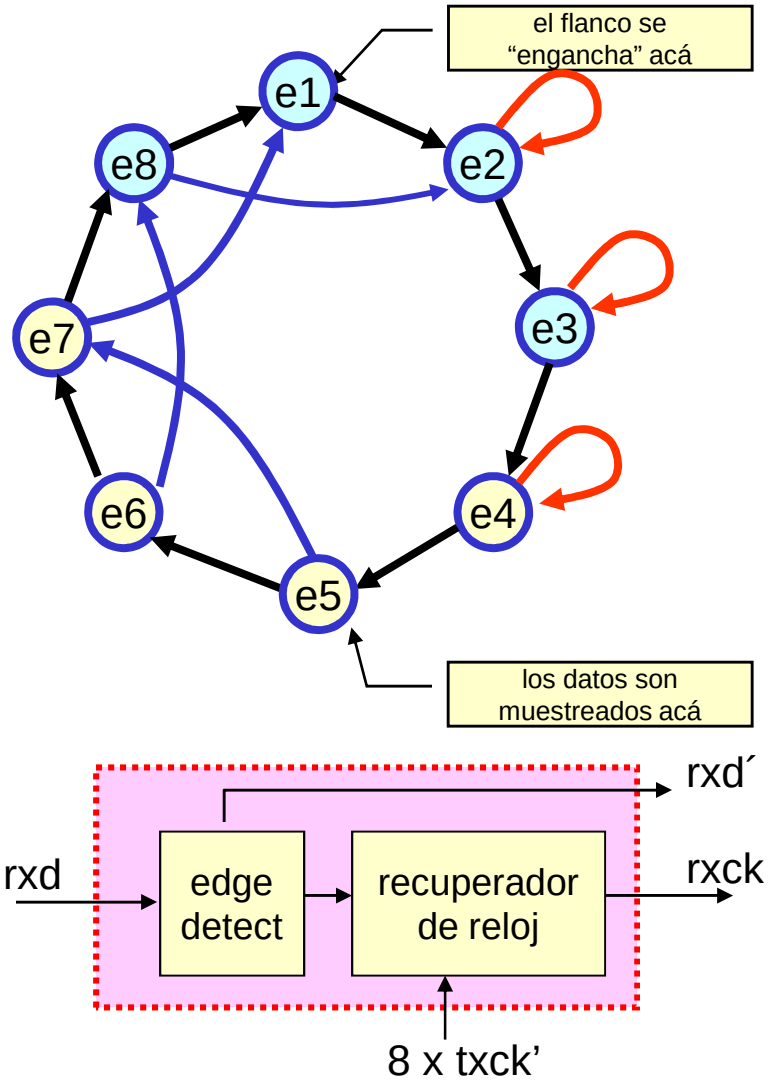
- En muchas aplicaciones dos sistemas intercambian información permanentemente y a una dada velocidad (medida en bps)
- Si los sistemas emplean relojes no coherentes, aunque hayan acordado una frecuencia es inevitable que entre ambos relojes exista una cierta diferencia
- Como alternativa a que el transmisor envíe al receptor, además de los datos, el reloj empleado para serializarlos, la información sobre el reloj del transmisor puede ser “recuperada” en el receptor a partir de las transiciones de los mismos datos
- El esquema básico de un **recuperador de reloj** emplea un reloj local con una frecuencia N veces mayor que la frecuencia *estimada* de transmisión.
- Y en base a los flancos de los datos que ingresan actúa sobre un divisor por N para generar el reloj de recepción **rxck**.





Circuitos de recuperación de reloj

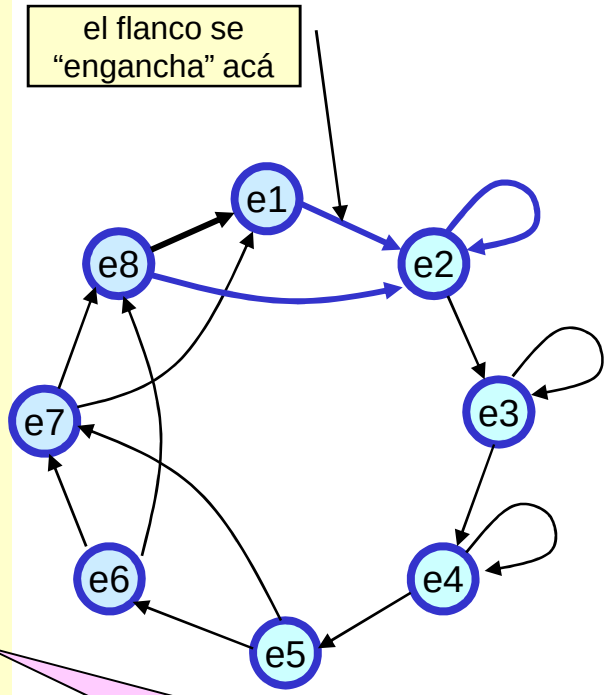
- En este ejemplo, una serie de datos ingresa en forma sincrónica aunque no coherente
- El recuperador de reloj emplea un reloj local con una frecuencia de 8 veces la frecuencia estimada de transmisión
- Mientras no hay flancos en los datos ese reloj se divide por 8 (flechas negras)
- Si aparecen flancos la división se acelera (flechas azules) o demora (flechas rojas) tratando que esas transiciones sucedan siempre entre e1 y e2
- Con lo que el momento ideal de captura de datos es entre e5 y e6.
- La limitación de este circuito es el corrimiento de fase cuando los datos contienen largas secuencias de ceros o unos sucesivos (cuanto más precisos y estables son txck y txck' más largas pueden ser)



Recuperador de reloj “behavioral”

```
ENTITY recupclk IS PORT (rxd,rxck_8x: IN bit; rxck: OUT bit);
END ENTITY recupclk;

ARCHITECTURE a OF recupclk IS
 SIGNAL rxdly,edge : bit;
 SIGNAL e: bit_vector (8 DOWNTO 1);
BEGIN
 edge <= rxdly XOR rxd; -- detector de transicion
 PROCESS (rxck_8x) IS
  BEGIN
   IF (rxck_8x='1') THEN
    rxdly <= rxd; -- rxd demorado un ciclo
    CASE (e) IS
     WHEN X"02" => IF edge='1' THEN e <= X"02"; ELSE e <= X"04"; END IF;
     WHEN X"04" => IF edge='1' THEN e <= X"04"; ELSE e <= X"08"; END IF;
     WHEN X"08" => IF edge='1' THEN e <= X"08"; ELSE e <= X"10"; END IF;
     WHEN X"10" => IF edge='1' THEN e <= X"40"; ELSE e <= X"20"; END IF;
     WHEN X"20" => IF edge='1' THEN e <= X"80"; ELSE e <= X"40"; END IF;
     WHEN X"40" => IF edge='1' THEN e <= X"01"; ELSE e <= X"80"; END IF;
     WHEN X"80" => IF edge='1' THEN e <= X"02"; ELSE e <= X"01"; END IF;
     WHEN OTHERS => e <= X"02";
    END CASE;
   END IF;
  END PROCESS;
 rxck <= e(5);
END ARCHITECTURE a;
```

Descubrir los errores
Hay errores graves!!!

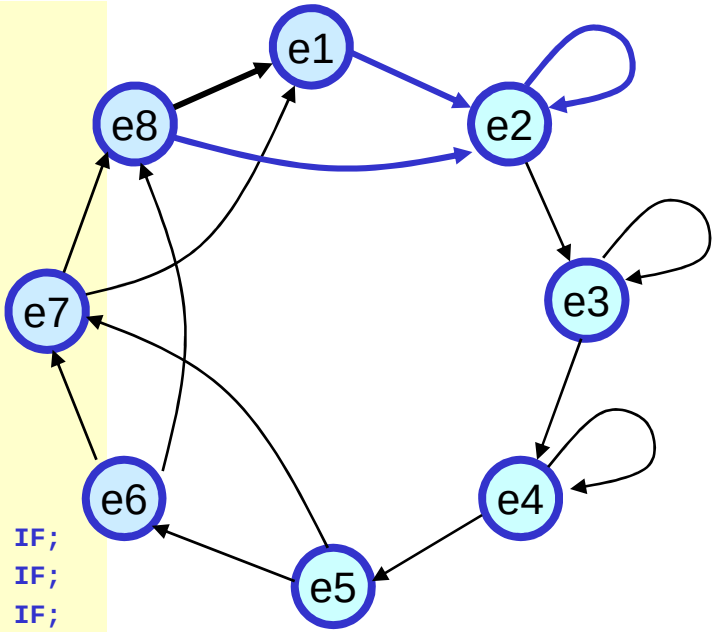


Este diseño, compilado requiere 32 LEs

Recuperador de reloj “behavioral”

```
ENTITY recupclk IS PORT (rxd,rxck_8x: IN bit; rxck: OUT bit);
END ENTITY recupclk;

ARCHITECTURE a OF recupclk IS
 SIGNAL rxdly,edge : bit;
 SIGNAL e: bit_vector (8 DOWNTO 1);
BEGIN
 edge <= rxdly XOR rxd; -- detector de transicion
 PROCESS (rxck_8x) IS
  BEGIN
   IF rxck_8x='1' THEN
    rxdly <= rxd; -- rxd demorado un ciclo
    CASE e IS
     WHEN X"02" => IF edge='1' THEN e <= X"02"; ELSE e<= X"04"; END IF;
     WHEN X"04" => IF edge='1' THEN e <= X"04"; ELSE e<= X"08"; END IF;
     WHEN X"08" => IF edge='1' THEN e <= X"08"; ELSE e<= X"10"; END IF;
     WHEN X"10" => IF edge='1' THEN e <= X"40"; ELSE e<= X"20"; END IF;
     WHEN X"20" => IF edge='1' THEN e <= X"80"; ELSE e<= X"40"; END IF;
     WHEN X"40" => IF edge='1' THEN e <= X"01"; ELSE e<= X"80"; END IF;
     WHEN X"80" => IF edge='1' THEN e <= X"02"; ELSE e<= X"01"; END IF;
     WHEN OTHERS => e <= X"02";
    END CASE;
   END IF;
  END PROCESS;
 rxck <= e(5);
END ARCHITECTURE a;
```

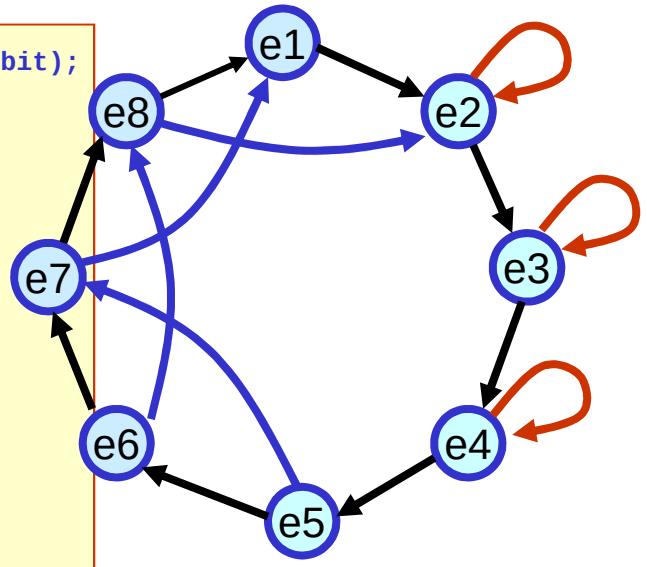
errores graves!!!

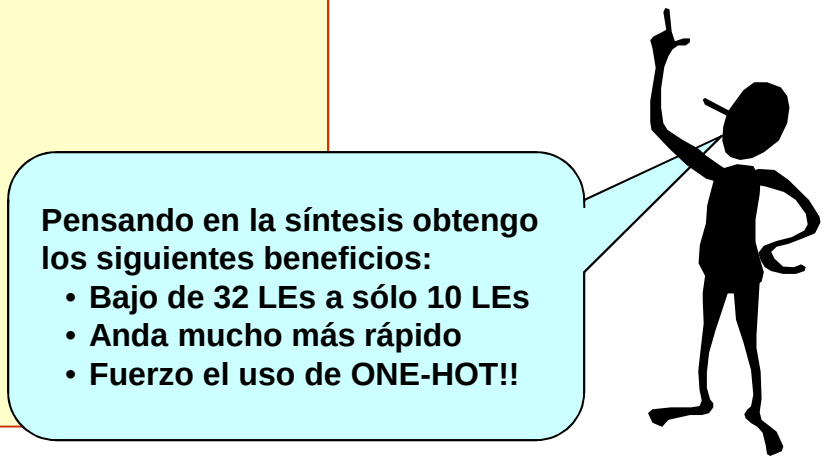


Recuperador de reloj pensando en la síntesis

```
ENTITY recupclka IS PORT (rxd,rxck_8x,nreset: IN bit; rxck,rxdsyn: OUT bit);
END ENTITY recupclka;

ARCHITECTURE a OF recupclka IS
 SIGNAL rxdly,edge : bit;
 SIGNAL e: bit_vector (8 DOWNTO 1);
BEGIN
 PROCESS (rxck_8x,nreset) IS
  BEGIN
   IF nreset='0' THEN e <= X"00";
   ELSIF rxck_8x'EVENT and rxck_8x='1' THEN
    rxdly <= rxd; -- rxd demorado un ciclo
    edge <= rxdly XOR rxd; -- detector de transicion
    e(1) <= NOT((e(8) AND NOT(edge)) OR (e(7) AND edge));
    e(2) <= NOT(e(1)) OR ((e(8) OR e(2)) AND edge);
    e(3) <= (e(2) AND NOT(edge)) OR (e(3) AND edge);
    e(4) <= (e(3) AND NOT(edge)) OR (e(4) AND edge);
    e(5) <= (e(4) AND NOT(edge));
    e(6) <= (e(5) AND NOT(edge));
    e(7) <= (e(6) AND NOT(edge)) OR (e(5) AND edge);
    e(8) <= (e(7) AND NOT(edge)) OR (e(6) AND edge);
   END IF;
  END PROCESS;
 rxck <= e(4); rxdsyn <= rxdly;
END ARCHITECTURE a;
```

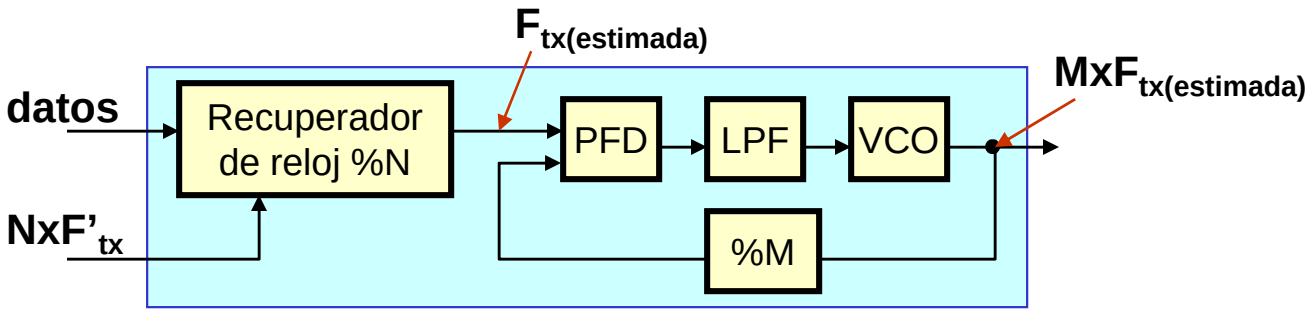
Hay metaestabilidades?
Buscar .....queda una!!!



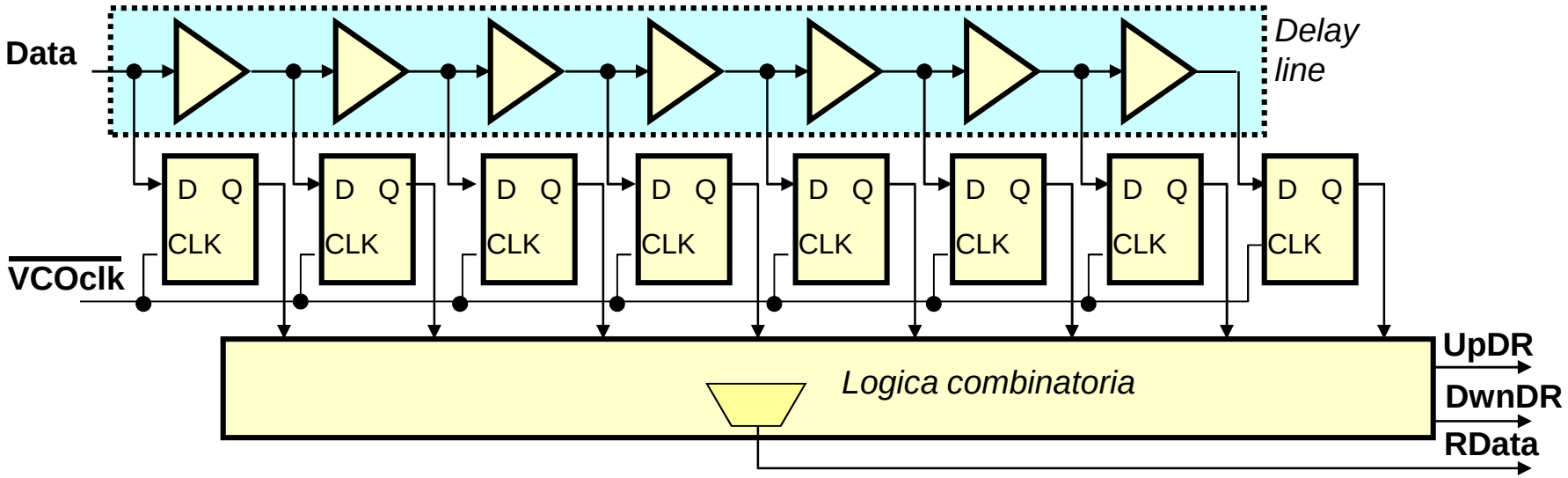


Uso de PLLs para recuperar reloj desde los datos

- A diferencia del uso del PLL para obtener una relación entre frecuencias, en el caso de tomar la información de reloj de los datos, las transiciones pueden estar o no presentes en el momento asociado al flanco de reloj a recuperar
- En ese caso la ausencia de información no debe generar alteraciones
- Un recuperador de reloj que genere $F_{tx(estimada)}$ como referencia puede servir para generar un $MxF_{tx(PLL)}$ con mucho menor ruido de fase



Recuperación de reloj en enlaces de alta velocidad



- La etapa de retiming de datos muestrea los datos ingresados a la frecuencia de reloj, solo que emplea una línea de retardo para tomar muestras demoradas de esos datos (la línea de retardo total tiene una duración apoximada a un tiempo de bit)
  - Si dentro del tiempo de bit el dato está centrado, los flipflops copiarán mitad '1' y mitad '0' (o viceversa)
  - Si en cambio los datos están corridos puede verse si están atrasados o adelantados y en base a eso corregir la fase del VCO

Diseñar el circuito en VHDL y simularlo.
Usar la library ALTERA y los LCELL para realizar la línea de retardo

Circuitos CDR en enlaces de alta velocidad: data retiming

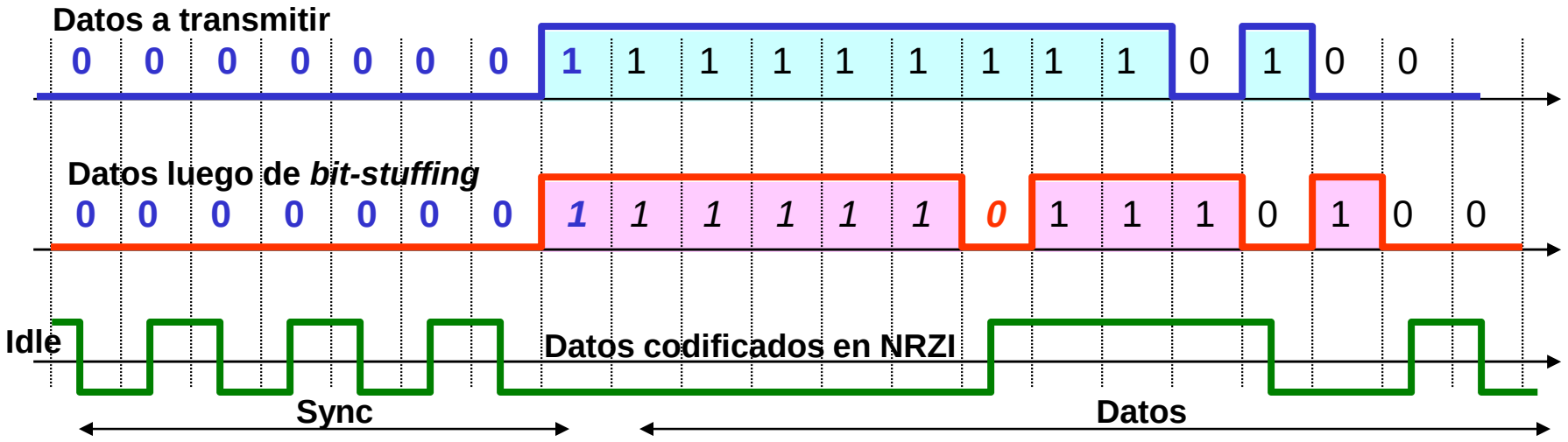
```
LIBRARY ieee; USE ieee.std_logic_1164.ALL; LIBRARY altera; USE altera.maxplus2.all;

ENTITY data_align IS PORT
 (data, vco_clk : IN std_logic; UpDr, DwnDr, Rdata : OUT std_logic;
  delay_taps : OUT std_logic_vector (7 DOWNTO 0));
END ENTITY data_align;

ARCHITECTURE a OF data_align IS
  SIGNAL delay_line,reg_chain  : std_logic_vector (7 DOWNTO 0);
BEGIN
  delay_line(0) <= data;
  gendelayline: FOR i IN 0 to 6 GENERATE
    delay_tap: lcell PORT MAP (A_IN => delay_line(i), A_OUT => delay_line (i+1));
  END GENERATE;
---
  PROCESS (vco_clk) IS BEGIN IF (vco_clk = '1') THEN reg_chain <= delay_line; END IF; END PROCESS;
---
  PROCESS (reg_chain) IS BEGIN
    CASE (reg_chain) IS
      WHEN "00001111"|"11110000" => UpDr <= '0'; DwnDr <= '0'; -- en fase; dejo al VCO como está
      WHEN "10000000"|"01111111"|"11000000"|"00111111"|"11100000"|"00011111"
                                 => UpDr <= '0'; DwnDr <= '1'; -- datos adelantados; freno al VCO
      WHEN "11111000"|"00000111"|"11111100"|"00000011"|"11111110"|"00000001"
                                 => UpDr <= '1'; DwnDr <= '0'; -- datos retrasados;  apuro al VCO
      WHEN OTHERS                => UpDr <= '0'; DwnDr <= '0'; -- no se sabe que hacer
    END CASE;
  END PROCESS;
---
  Rdata <= reg_chain(0); delay_taps <= delay_line;
END ARCHITECTURE a;
```

Vale para Quartus!

Enlaces pseudo sincrónicos: Bit-Stuffing

Cuando se desea evitar la transmisión de largas secuencias de bits idénticos debe agregarse información, y un método posible es llamado *bit stuffing* (es usado en USB, junto con NRZI):

- **NRZI:** método de codificación diferencial donde un '0' es representado mediante una transición (cambio de 1 a 0 o de 0 a 1) en tanto un '1' es representado mediante la ausencia de cambios
- **Bit stuffing:** inserción en el transmisor y remoción en el receptor de un '0' (una transición) cada seis '1' sucesivos (seis tiempos de bit sin cambios) de modo de garantizar la existencia de transiciones en la señal y la consiguiente recuperación de reloj. El uso de ***bit-stuffing*** permite transmitir los datos junto a la señal de reloj, y para ello se coloca un campo SYNC precediendo a cada paquete para que el receptor sincronice su reloj de recepción.



Violación de códigos en bit-stuffing

Una vez que se agrega información adicional, ésta puede ser aprovechada para otras tareas, tales como sincronismos de trama.

- Por ejemplo, en bit-stuffing pueden enviarse 7 unos seguidos (en vez de insertar el '0' de bit stuffing) para indicar un inicio de trama.
- Y si se usa NRZI conviene poner antes de esa violación de sincronismo algunos ceros de modo de garantizar la recuperación de reloj