



Laboratorio 4:

# Comunicación serie síncrona

simulación y lectura de un teclado PS/2

Diseño automático de sistemas

**José Manuel Mendías Cuadros**

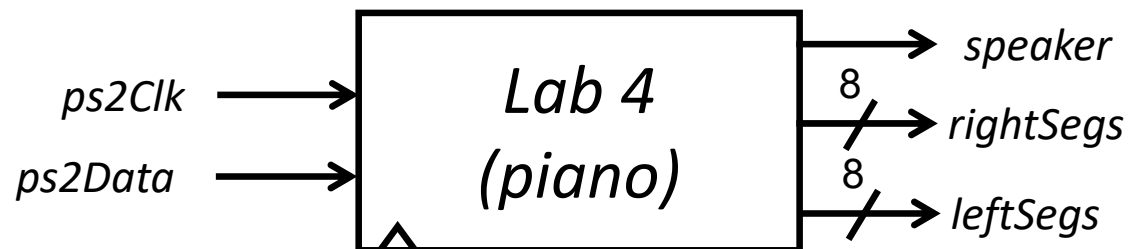
*Dpto. Arquitectura de Computadores y Automática  
Universidad Complutense de Madrid*



# Presentación



- Diseñar un **piano electrónico** con el siguiente comportamiento:
  - Generará el sonido a través de un **altavoz convencional** de 2 terminales.
  - Podrá generar cualquiera de las **13 notas que forman la 4ª octava** de un piano.
  - Cada **nota estará asociada a una tecla** y sonará mientras esté presionada.
  - Se generará **solo una nota simultáneamente**, por lo que si se presionan varias teclas solo sonará la primera que se pulsó.
- El teclado y el altavoz se conectarán del siguiente modo:
  - Como teclado se usará un **teclado PS/2** convencional conectado a la placa XSA-3S.
    - El **scancode** de cada tecla pulsada se mostrará **en los displays 7-segs** de la placa XST.
  - La **asociación de teclas** y notas será la siguiente:
    - **A**: DO, **W**: DO#, **S**: RE, **E**: RE#, **D**: MI, **F**: FA, **T**: FA#, **G**: SOL, **Y**: SOL#, **H**: LA, **U**: LA#, **J**: SI, **K**: DO
  - Un terminal del altavoz estará conectado al **pin L5** de la FPGA, el otro a **Vcc**.
    - Las notas se generan enviando a dicho pin una onda cuadrada de una cierta frecuencia.



# Protocolo PS/2

## presentación



- Un teclado o ratón tipo **AT-PS/2** utiliza para comunicarse con un host un **bus bidireccional serie síncrono**
  - Dispone de dos líneas a colector abierto con resistencias de pullup TTL +5V
    - **CLK**: para transmitir el reloj de sincronización.
    - **DATA**: para transmitir los datos serie.
    - Ambas por defecto están en alta.
  - La **señal de reloj** siempre debe ser generada por el **dispositivo**.
    - Con una frecuencia entre 10 y 30 KHz.
  - La **señal de datos** puede ser generados por el **dispositivo o por el host**.
  - La información se transmite en **tramas de 11 bits** con el siguiente formato:
    - 1 bit de start (0), 8 bits de datos (primero LSB), 1 bit paridad impar, 1 bit de stop (1)



**5 Pin DIN**

1. KBD Clock
2. KBD Data
3. N/C
4. GND
5. +5V (VCC)



**PS/2**

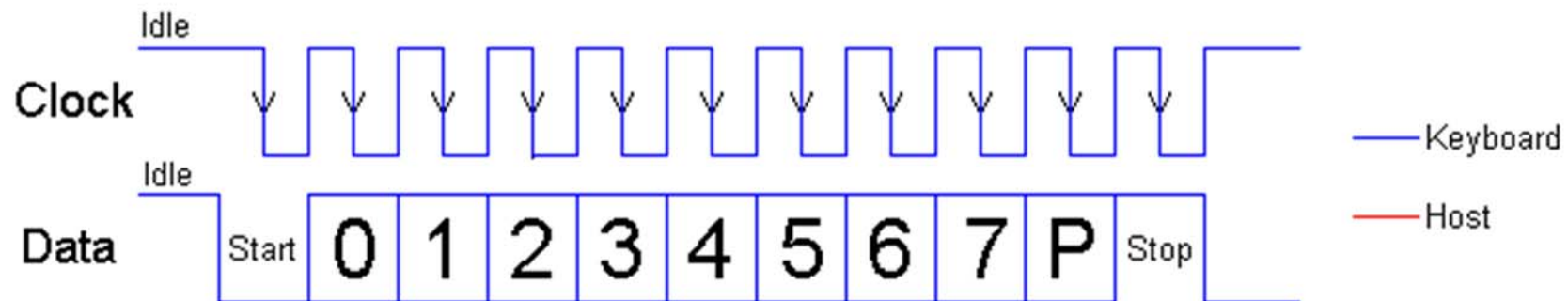
1. KBD Clock
2. GND
3. KBD Data
4. N/C
5. +5V (VCC)
6. N/C

# Protocolo PS/2

## transmisión device-to-host



- El dispositivo inicia una transmisión cuando:
  - Ha sucedido un evento en el periférico o ha recibido una solicitud del host.
  - Verifica que CLK y DATA están en ALTA.
- Una vez el dispositivo inicia la transmisión, el host:
  - Debe muestrear la señal de DATA a flancos de bajada de CLK.
  - Puede abortarla antes del 10º ciclo fijando CLK a baja.
    - La retransmisión de un dato abortado debe ser solicitada por el host.



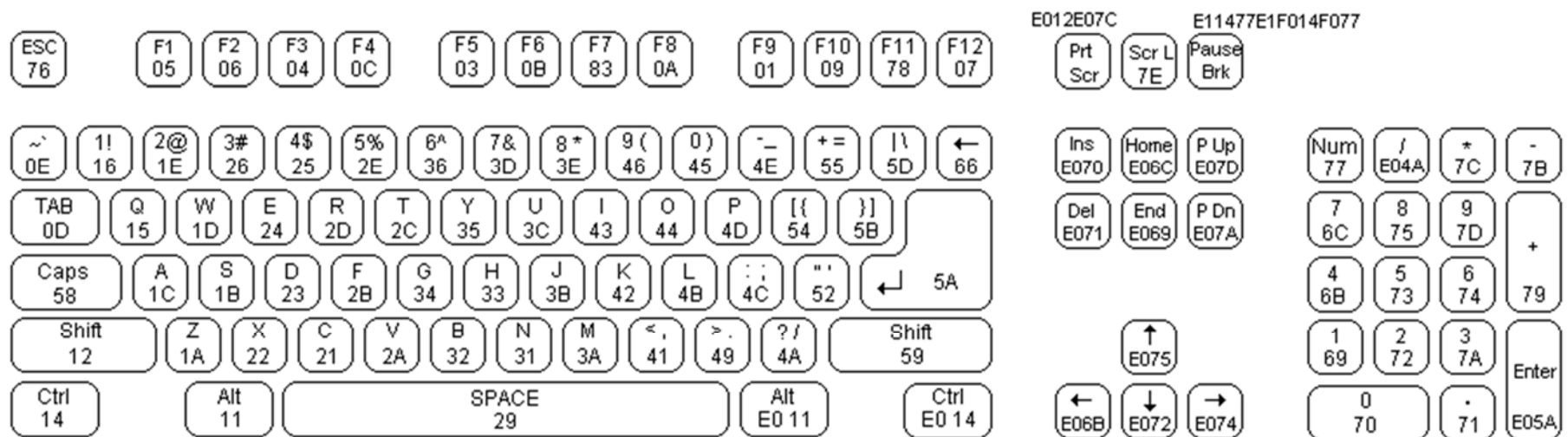
- El host además, en cualquier momento, puede:
  - Inhabilitar las transmisiones fijando CLK a baja
    - En cuyo caso los datos se almacenan temporalmente en el dispositivo.
  - Indicar al dispositivo su intención de enviarle información fijando DATA a baja.

# Protocolo PS/2

## teclado: scancodes (i)

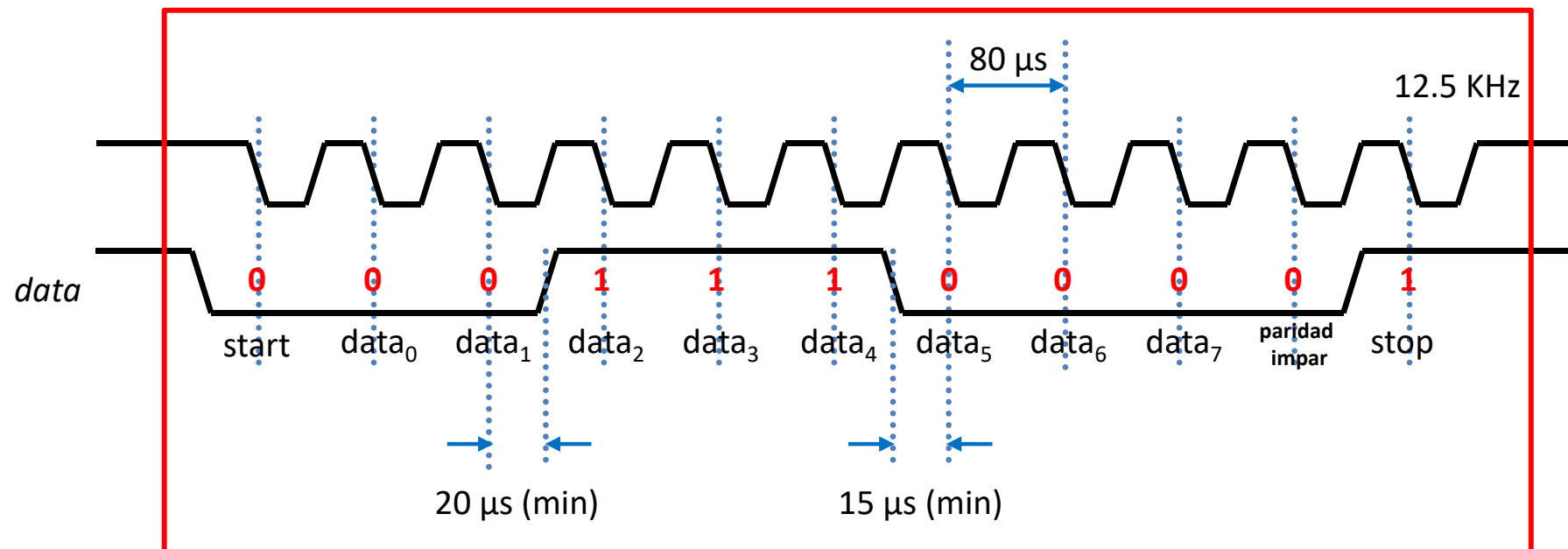
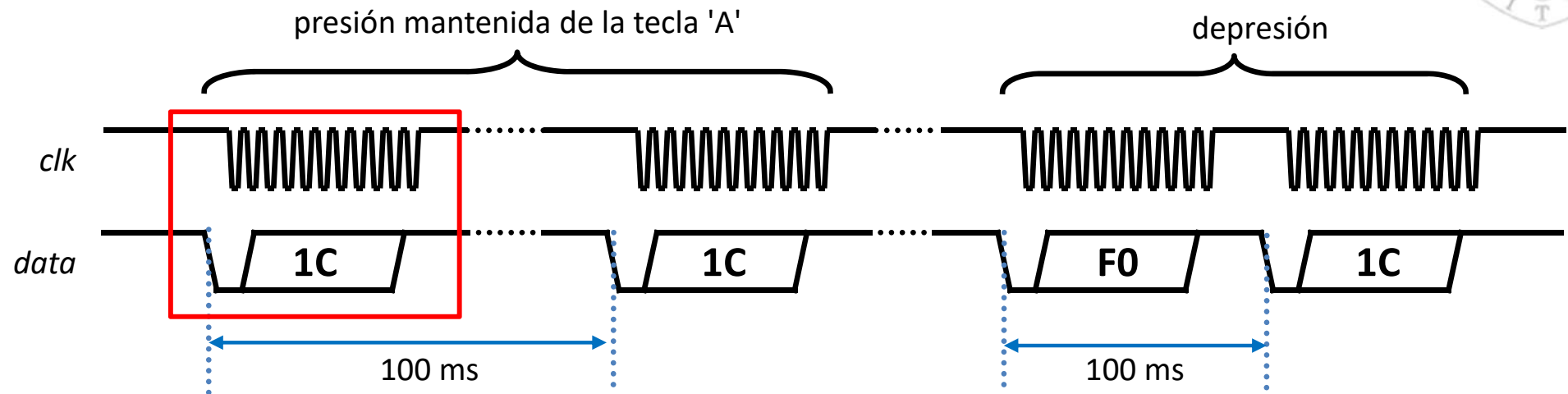


- Cada vez que se presiona una tecla, el teclado envía de 3 a n códigos:
  - **Scancode de presión**: indica el momento de la presión y la tecla pulsada.
    - Si la tecla permanece pulsada el teclado reenvía el código periódicamente.
    - Si el scancode es compuesto, los bytes que lo forman se envían en ráfaga.
  - **Código de depresión** (F0): indica el momento de la depresión de una tecla.
  - **Scancode de depresión**: indica la tecla que se ha dejado de pulsar.
- Cuando se presionan varias teclas:
  - Los scancodes de presión/depresión pueden intercalarse.
  - Pero el par (código de depresión, scancode de depresión) se envía en ráfaga.



# Protocolo PS/2

teclado: scancodes (ii)

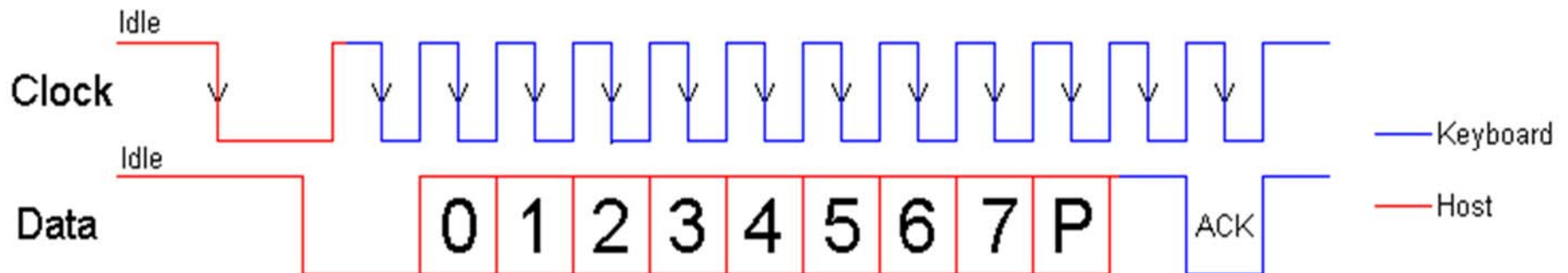




# Protocolo PS/2

## transmisión host-to-device

- El **host puede iniciar una transmisión** fijando DATA a baja
  - Estando DATA a baja y tras poner CLK a ALTA, el dispositivo comenzará a generar la señal de reloj antes de 10 ms, provocando el primer flanco de bajada
  - Para evitar evitar colisiones con los posibles datos en transmisión es conveniente fijar primero CLK a BAJA durante al menos 60  $\mu$ s y no después del 10º ciclo.
- Una vez el host **inicia la transmisión**, el dispositivo:
  - Muestreará la señal de DATA a **flancos de bajada de CLK**
    - Por lo que el host deberá poner un nuevo bit cuando CLK está en alta, como máximo 1  $\mu$ s después del flanco de subida.
- El host **finaliza la transmisión**:
  - Enviando el bit de paridad en el 9º ciclo y liberando DATA en el 10º (que subirá a alta)
  - El dispositivo reconoce la recepción poniendo a baja la línea DATA en el 11º ciclo.







# Protocolo PS/2

## teclado: comandos y respuestas

- El host, entre otros, puede enviar al teclado los siguientes **comandos**:
  - **Reset (FF)**, resetea el dispositivo, inicia power-on test, responde con el power-on byte
  - **Set/reset indicadores de estado (ED)**, controla los LEDs de estado del teclado
    - Responde con ACK y espera el byte de estado: (b0,b1,b2,0,0,0,0,0) donde b0 scrollLock, b1, NumLock, b2 CapsLock, mandando un 1 cambia el indicador.
  - **Disable (F5)**, resetea teclado, suspende escaneado, devuelve ACK
  - **Echo (EE)**, solicita que el dispositivo responda con ECHO .
  - **Enable (F4)**, borra buffer, devuelve ACK.
  - **Resend (FE)**, solicita la retransmisión del último scancode.
  - **Intervalo/retardo de autorepetición (F3)**,
    - Responde con ACK y espera con un byte de b6-b7 (00=250 ms, 11=1000ms), b4..b0 (00000=30x/s, 11111=2x/s), vuelve a responder con ACK.
- El teclado, entre otros, puede enviar al host las siguientes **respuestas**:
  - **ACK (FA)**
  - **Power-on satisfactorio (AA)**
  - **ECHO (EE)**
  - **Error (00 ó FF)**, error o desbordamiento del buffer

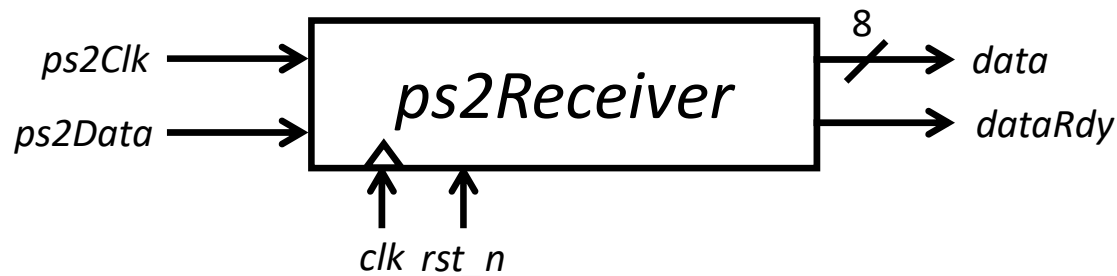
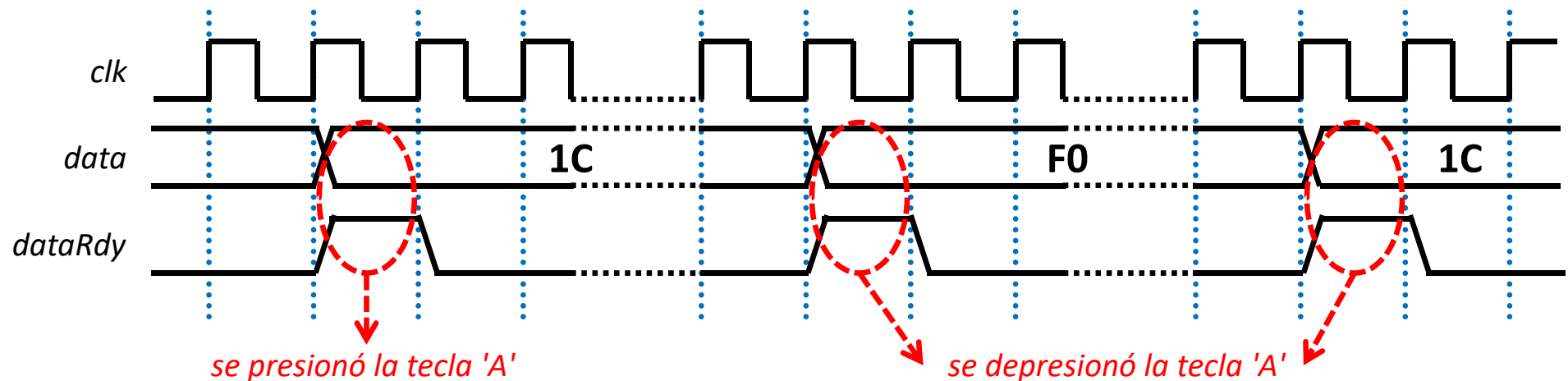


# Interfaz PS/2

## presentación

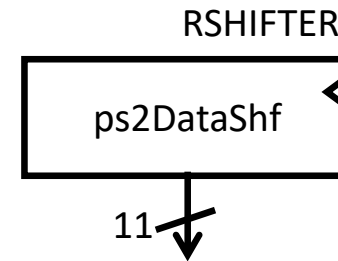


- Diseñar un receptor PS/2 elemental:
  - Convertirá de **serie a paralelo** cada dato individual recibido desde un bus PS/2.
  - Cada vez que reciba correctamente una trama, **volcará el dato en data**.
    - Si hay error de paridad, ignorará la trama recibida.
  - Por cada nuevo dato volcado **activará durante un ciclo** la señal de strobe **dataRdy**.



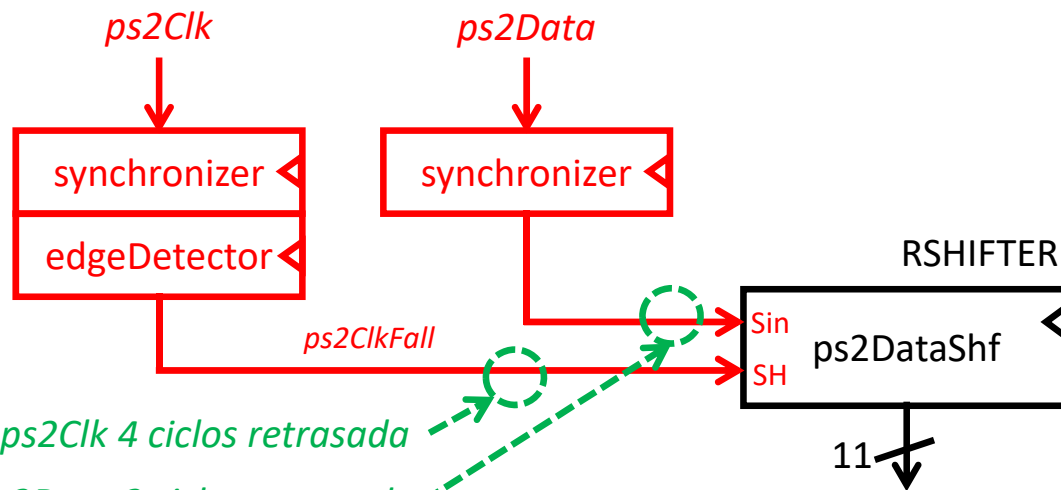
# Interfaz PS/2

## diagrama RTL



# Interfaz PS/2

## diagrama RTL



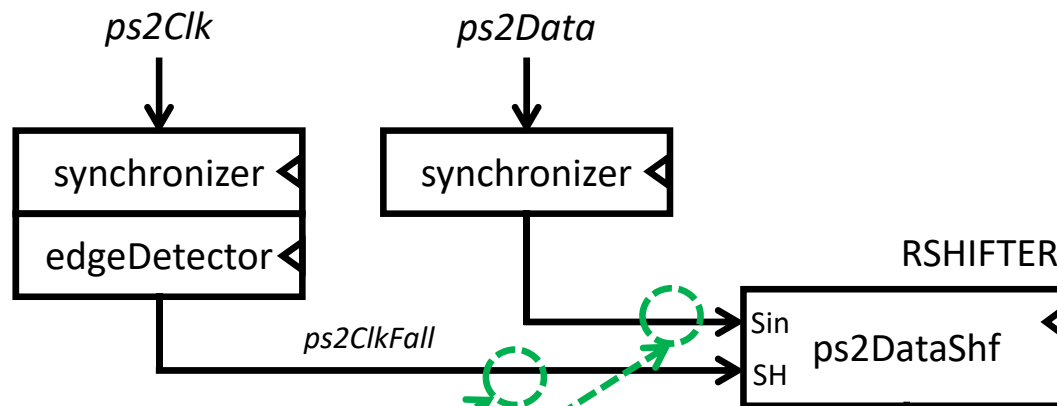
*esta señal es ps2Clk 4 ciclos retrasada*

*esta señal es ps2Data 2 ciclos retrasada*

*2 ciclos de diferencia  
suponen:  $40\text{ ns} \ll 20\text{ }\mu\text{s}$*

# Interfaz PS/2

## diagrama RTL



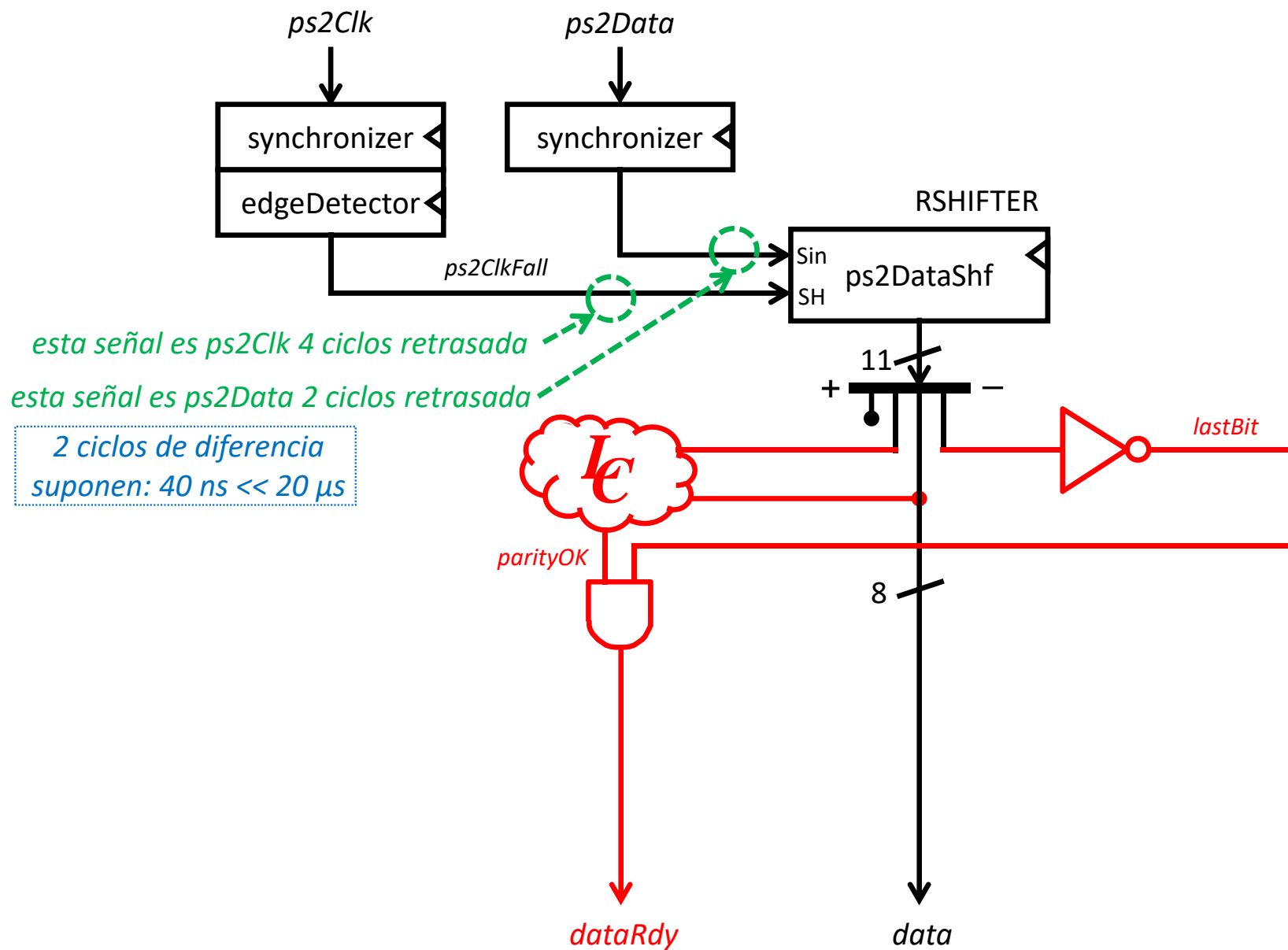
esta señal es ps2Clk 4 ciclos retrasada

esta señal es ps2Data 2 ciclos retrasada

2 ciclos de diferencia  
suponen:  $40\text{ ns} \ll 20\text{ }\mu\text{s}$

# Interfaz PS/2

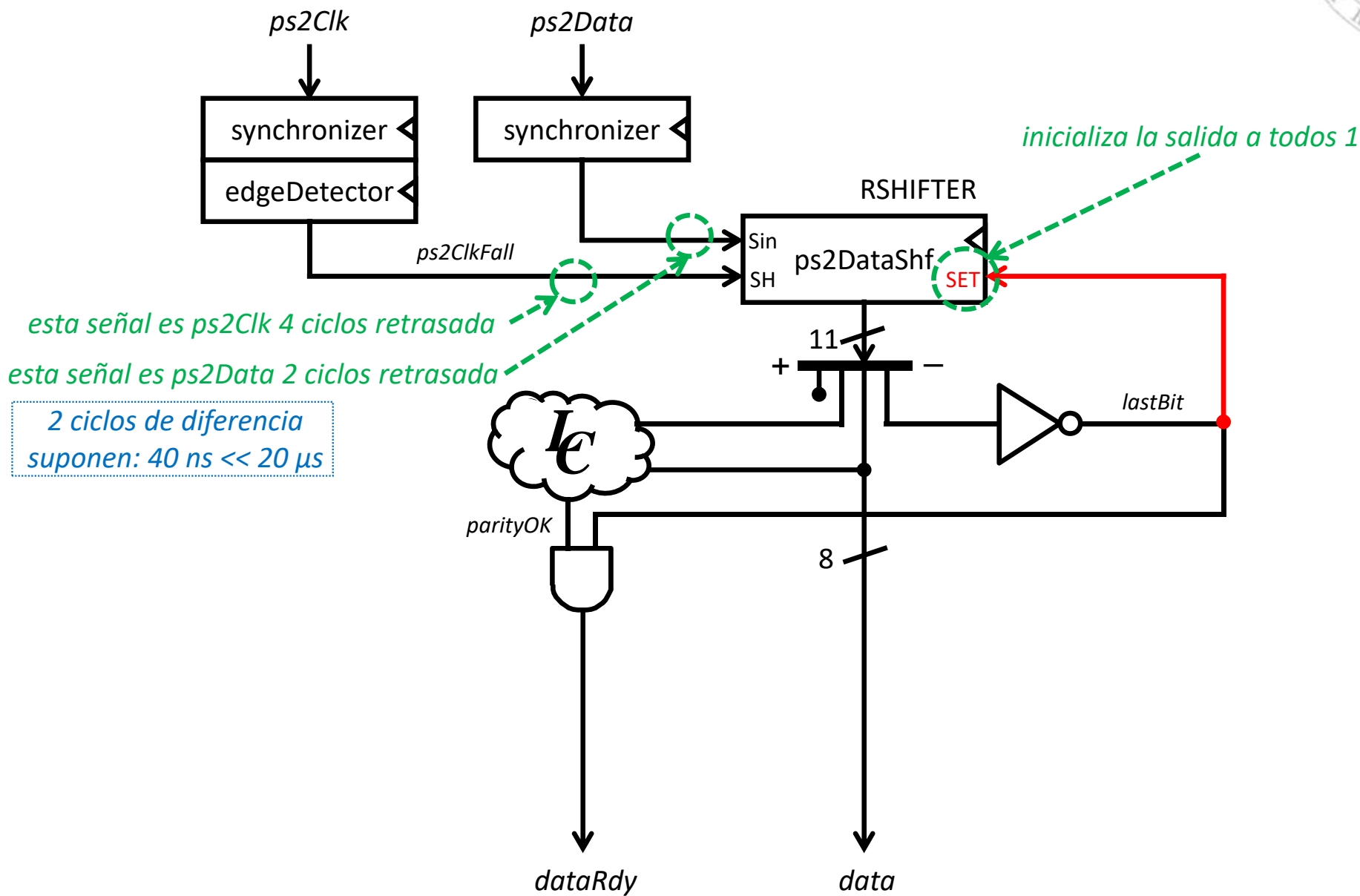
## diagrama RTL





# Interfaz PS/2

# diagrama RTL

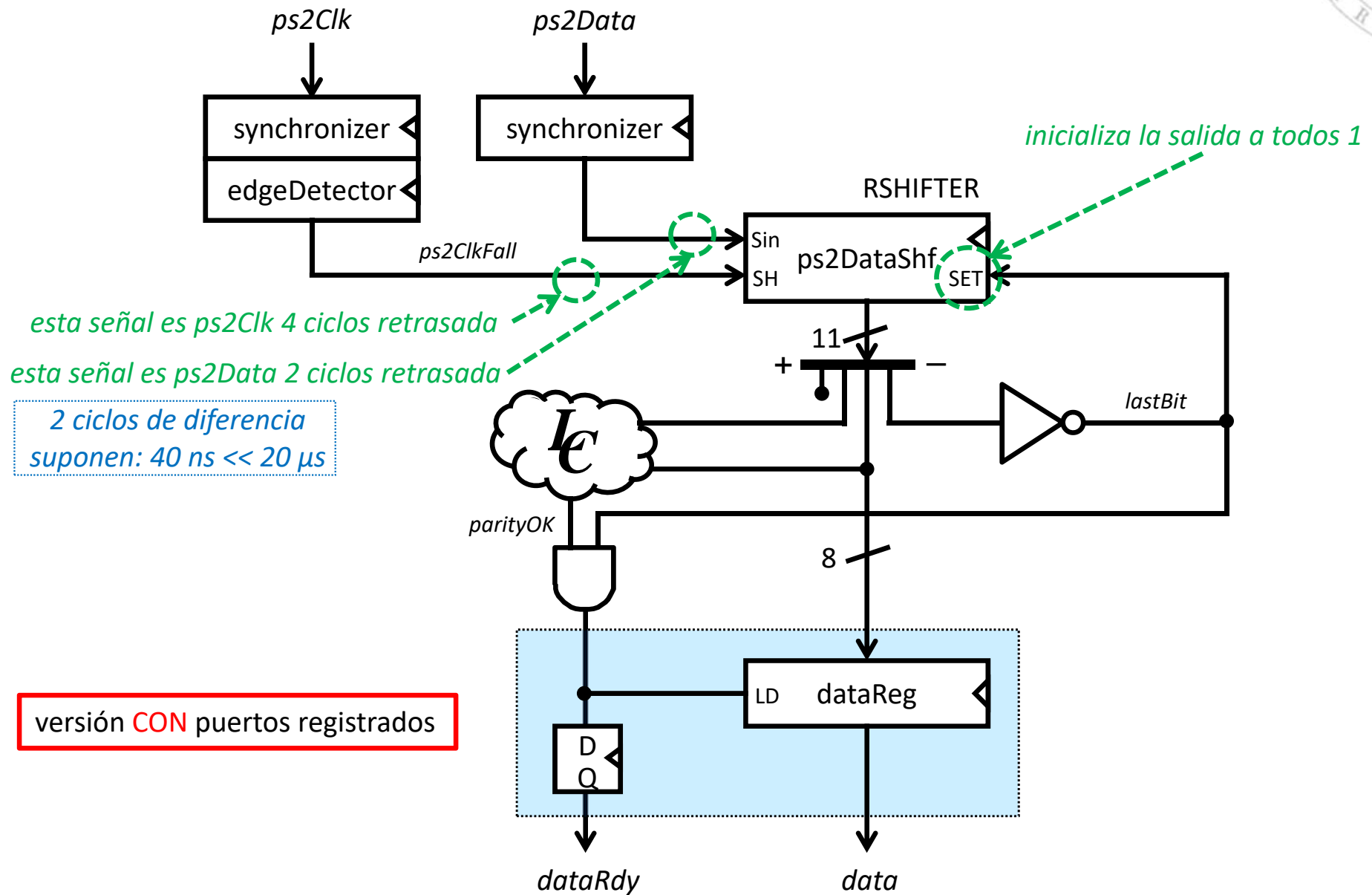


versión SIN puertos registrados



# Interfaz PS/2

## diagrama RTL



# Interfaz PS/2

## ps2Receiver.vhd



```
library ieee;
use ieee.std_logic_1164.all;

entity ps2Receiver is
  generic (
    REGOUTPUTS : boolean    -- registra o no las salidas
  );
  port (
    -- host side
    rst_n      : in  std_logic;    -- reset asíncrono del sistema (a baja)
    clk        : in  std_logic;    -- reloj del sistema
    dataRdy    : out std_logic;    -- se activa durante 1 ciclo cada vez que hay
                                   -- un nuevo dato recibido
    data       : out std_logic_vector (7 downto 0); -- dato recibido
    -- PS2 side
    ps2Clk     : in  std_logic;    -- entrada de reloj del interfaz PS2
    ps2Data    : in  std_logic     -- entrada de datos serie del interfaz PS2
  );
end ps2Receiver;
```

# Interfaz PS/2

## ps2Receiver.vhd



```
architecture syn of ps2Receiver is
    signal ps2ClkSync, ps2DataSync, ps2ClkFall: std_logic;
    signal ps2DataShf: std_logic_vector(10 downto 0);
    signal lastBit, parityOK: std_logic;
begin
    ps2ClkSynchronizer : synchronizer
        ...;
    ps2DataSynchronizer : synchronizer
        ...;
    ps2ClkEdgeDetector : edgeDetector
        ...;

    ps2DataShifter:
    process (rst_n, clk)
    begin
        if rst_n='0' then
            ps2DataShf <= (others => '1');
        elsif rising_edge(clk) then
            ...
        end if;
    end process;

    oddParityCheker :
    parityOK <= ...;

    lastBitCheker :
    lastBit <= ...;
    ...

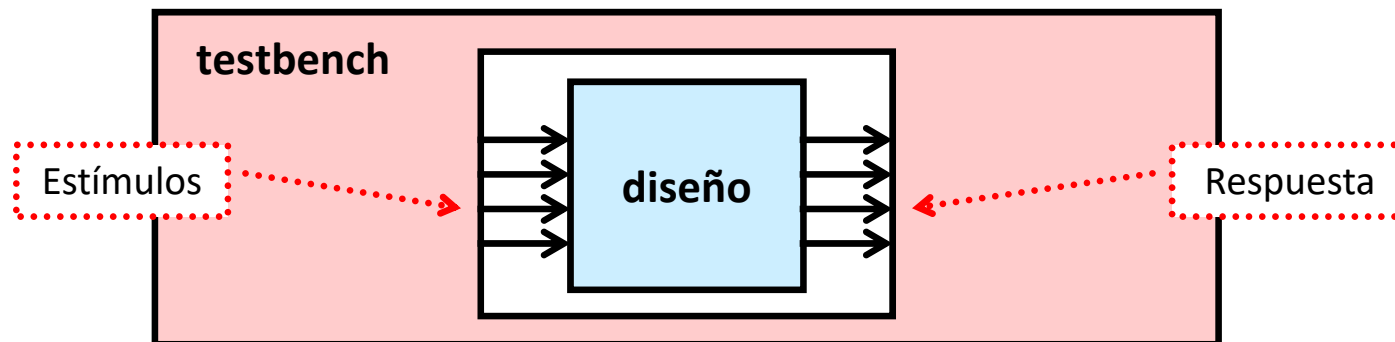
    ...
    outputRegisters:
    if REGOUTPUTS generate
        process (rst_n, clk)
        begin
            if rst_n='0' then
                dataRdy <= '0';
                data <= (others=>'0');
            elsif rising_edge(clk) then
                ...
            end if;
        end process;
    end generate;

    outputSignals:
    if not REGOUTPUTS generate
        dataRdy <= ...;
        data <= ...;
    end generate;
end syn;
```



# Validación por simulación

- Para **validar una diseño** es necesario estimular sus entradas y comprobar que sus salidas son las correctas.
  - Los simuladores facilitan entornos gráficos para el estímulo y visualización.
- Cuando se usa VHDL, se puede **usar el lenguaje para todo**:
  - Definir el comportamiento del diseño.
  - Definir una colección de estímulos de complejidad arbitraria.
  - Chequear autónomamente la respuesta del diseño.
  - De manera que no sean necesarios ni entornos gráficos ni comandos adicionales.
- En VHDL se simulan entidades cerradas (sin puertos) formadas por:
  - **Diseño a validar** (unit under test)
  - **Banco de pruebas** (testbench) que estimula el circuito y analiza su respuesta.





# Test del interfaz PS/2

## presentación

- Codificar un test del receptor PS/2 elemental:
  - Generará una señal de reset
  - Generará una señal de reloj a 50 MHz
  - Generará las formas de onda de las señales ps2Clk y ps2Data emulando la presión mantenida y la depresión de la tecla A
    - Reproduciendo el cronograma mostrado en una transparencia anterior.
    - Este patrón se repetirá indefinidamente
  - Analizará las formas de onda de las señales data y dataRdy para verificar que:
    - A cada activación de dataRdy en data se encuentra el correspondiente código.
    - La señal dataRdy cuando se activa lo hace durante solo 1 ciclo de reloj.
    - Tras la activación del reset, dataRdy y data valen 0.
  - Deberá simularse, al menos, durante 600 ms (30 millones de ciclos de reloj).
- Téngase en cuenta que codificar un buen testbench puede llegar a ser tan complejo como codificar el propio diseño.
  - Además los errores detectados en la simulación pueden deberse a errores del diseño o a errores del testbench.

# Test del interfaz PS/2

## ps2ReceiverTest.vhd



```
entity ps2ReceiverTest is
end ps2ReceiverTest;

library ieee;
use ieee.std_logic_1164.all;
use work.common.all;

architecture sim of ps2ReceiverTest is

    constant clkPeriod : time := 20 ns;    -- Periodo del reloj (50 MHz)

    signal clk          : std_logic := '1';
    signal rst_n        : std_logic := '0';
    signal ps2clk       : std_logic := '1';
    signal ps2Data      : std_logic := '1';
    signal data         : std_logic_vector(7 downto 0) := (others => '0');
    signal dataRdy      : std_logic := '0';

    type stimulusT is
        record
            ps2clk    : std_logic;
            ps2data   : std_logic;
        end record;

    type stimuliT is array (natural range <>) of stimulusT;

    ...
end architecture;
```

# Test del interfaz PS/2

## ps2ReceiverTest.vhd



```
...
-- Trama PS/2 correspondiente al scancode de la tecla A
constant Astimuli : stimuliT(1 to 23) :=
(
  ( '1', '0' ),    -- start (0)
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- paridad (0)
  ( '0', '0' ),
  ( '1', '1' ),    -- stop
  ( '0', '1' ),
  ( '1', '1' )    -- reposo
);
```

```
...
-- Trama PS/2 correspondiente al código de depresión
constant F0stimuli : stimuliT(1 to 23) :=
(
  ( '1', '0' ),    -- start (0)
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '0' ),    -- 0
  ( '0', '0' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '1' ),    -- 1
  ( '0', '1' ),
  ( '1', '1' ),    -- paridad (1)
  ( '0', '1' ),
  ( '1', '1' ),    -- stop
  ( '0', '1' ),
  ( '1', '1' )    -- reposo
);
```



# Test del interfaz PS/2

## ps2ReceiverTest.vhd



```
begin

    uut : ps2Receiver
        generic map ( REGOUTPUTS => true );
        port map ( rst_n => rst_n, clk => clk, dataRdy => dataRdy,
                    data => data, ps2Clk => ps2Clk, ps2Data => ps2Data );

    rstGen :
    rst_n <=
        '1' after (50 us + 5 ns),
        '0' after (500 ms + 5 ns),
        '1' after (500 ms + 50 us + 5 ns);

    clkGen :
    clk <= not clk after clkPeriod/2;

    ...
```

# Test del interfaz PS/2

## ps2ReceiverTest.vhd



```
stimuliGen :
process
begin
    wait for 5 ns;  -- Evita que coincidan los flancos de clk y de los estímulos
    loop
        wait for 100 ms;
        for i in Astimuli'range loop          -- Genera scancode de presión de A
            ps2clk <= Astimuli(i).ps2clk;
            ps2data <= Astimuli(i).ps2data;
            wait for 40 us;
        end loop;
        wait for 100 ms;
        for i in Astimuli'range loop          -- Genera scancode de repetición de A
            ps2clk <= Astimuli(i).ps2clk;
            ps2data <= Astimuli(i).ps2data;
            wait for 40 us;
        end loop;
        wait for 100 ms;
        for i in F0stimuli'range loop         -- Genera código de depresión
            ps2clk <= F0stimuli(i).ps2clk;
            ps2data <= F0stimuli(i).ps2data;
            wait for 40 us;
        end loop;
        wait for 100 ms;
        for i in Astimuli'range loop         -- Genera scancode de depresión de A
            ps2clk <= Astimuli(i).ps2clk;
            ps2data <= Astimuli(i).ps2data;
            wait for 40 us;
        end loop;
        wait for 500 ms;
    end loop;
end process;
```

# Test del interfaz PS/2

## ps2ReceiverTest.vhd



```
...
dataCheck :
process
begin
    wait until dataRdy='1';
    assert data=X"1C"
        report "La uut ha leído erróneamente el scancode de presión de la tecla A"
        severity error;
    wait until dataRdy='1';
    assert data=X"1C"
        report "La uut ha leído erróneamente el scancode de repetición de la tecla A"
        severity error;
    wait until dataRdy='1';
    assert data=X"F0"
        report "La uut ha leído erróneamente el código de depresión"
        severity error;
    wait until dataRdy='1';
    assert data=X"1C"
        report "La uut ha leído erróneamente el scancode de depresión de la tecla A"
        severity error;
end process;
...
```

# Test del interfaz PS/2

## ps2ReceiverTest.vhd



```
...
rstCheck:
process (rst_n'delayed)
begin
    if rst_n='0' then
        assert dataRdy='0' and data=(others=>'0')
        report "La uut no se resetea adecuadamente"
        severity warning;
    end if;
end process;

dataRdyCheck :
process (dataRdy)
begin
    if dataRdy='0' then
        assert dataRdy'delayed'last_event <= clkPeriod
        report "La uut activa durante más de un ciclo la señal dataRdy"
        severity warning;
    end if;
end process;

end sim;
```

# Sonido



- Un **sonido** es una onda longitudinal mecánica con **una frecuencia comprendida entre 20 y 20000 Hz**.
  - el sonido lo percibimos como una variación periódica de la presión en nuestro tímpano generada por la oscilación periódica de las partículas del aire.
- **Magnitudes objetivas del sonido:**
  - **Frecuencia:** número de oscilaciones completas por unidad de tiempo.
  - **Amplitud:** valor máximo de la elongación de una oscilación.
  - **Intensidad:** cantidad media de energía transportada por una onda, por unidad de superficie y tiempo. Es proporcional al cuadrado de la amplitud de la onda.
    - Dado el amplio intervalo de intensidades perceptibles, se utiliza una escala logarítmica relativa denominada *nivel de intensidad* que se expresa en decibelios.
- **Magnitudes subjetivas del sonido:**
  - **Volumen:** percepción humana de la intensidad, entre otros factores, depende de la amplitud y de la frecuencia de la onda.
  - **Tono:** percepción humana de la frecuencia de una onda, entre otros factores depende de la amplitud y de la frecuencia de la onda.
  - **Timbre:** percepción humana de la forma de la onda, es decir, del número, intensidad y distribución temporal de los armónicos que forman un sonido.

# Notas musicales



- Una **nota musical** (pura) es una onda sinusoidal de frecuencia y amplitud definidas:
  - Se **agrupan en escalas y octavas** de modo que sus **frecuencias guardan una relación matemática exacta**.
- En la **escala uniformemente temperada** cada octava incluye 13 notas:
  - Cualesquiera 2 notas consecutivas difieren entre sí en un semitono, y sus frecuencias siempre guardan una relación de  $\sqrt[12]{2}$
  - La frecuencia de cualquier nota puede calcularse a partir de la frecuencia de una de ellas que se toma como referencia (LA 440Hz) según la fórmula:

$$f_{nota} = f_{LA} \cdot \sqrt[12]{2^n} = 440 \cdot \sqrt[12]{2^n} \approx 440 \cdot 1.06^n$$

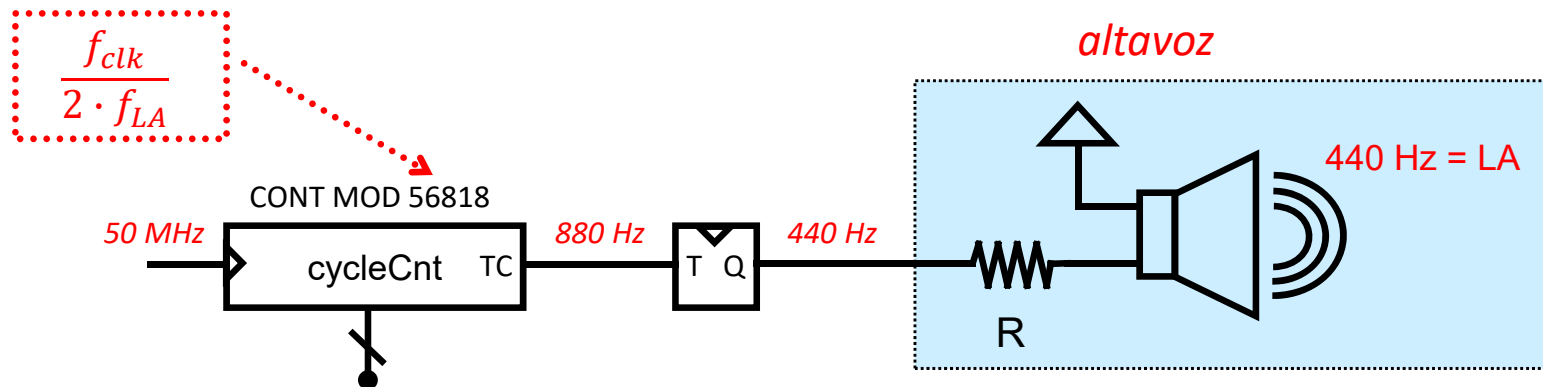
donde  $n$  es el número de notas que la separan del LA (negativo hacia la izquierda)

	DO#		RE#				FA#		SOL#		LA#			
	277.2 Hz		311.1 Hz				370.0 Hz		415.3 Hz		466.2 Hz			
$\sqrt[12]{2}$	DO central	RE		MI		FA		SOL		LA		SI		DO
	261.6 Hz	293.7 Hz		329.6 Hz		349.2 Hz		392.0 Hz		440 Hz		493.9 Hz		523.3 Hz
	$\sqrt[6]{2}$ $\sqrt[6]{2}$ $\sqrt[12]{2}$ $\sqrt[6]{2}$ $\sqrt[6]{2}$ $\sqrt[6]{2}$ $\sqrt[6]{2}$ $\sqrt[12]{2}$													

# Generación digital de sonido



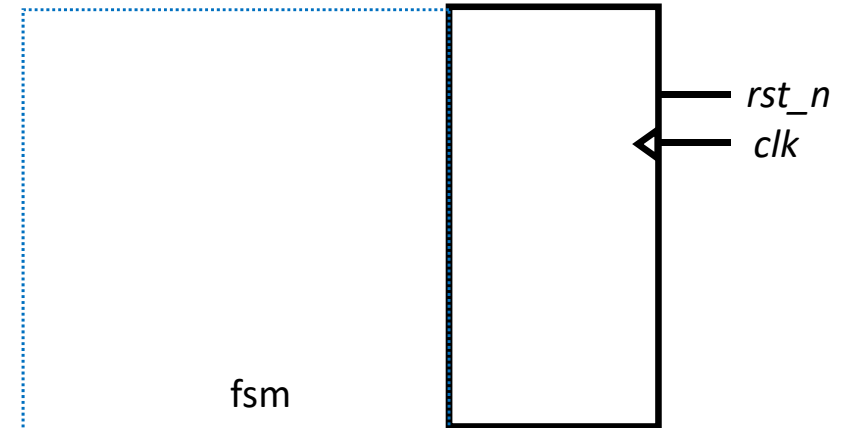
- Un **altavoz**, es un transductor capaz de **generar una onda sonora análoga** (en frecuencia y amplitud) a **una señal eléctrica dada**.
- Se compone de una membrana elástica unida a una bobina móvil que se monta dentro del campo magnético de un imán permanente.
  - La fuerza de atracción entre la bobina y el imán es función de la intensidad y el sentido de la corriente que circule por la bobina.
  - **Cambios en la corriente**, provocan movimientos en la bobina que se traducen en **vibraciones de la membrana**. Si estas vibraciones tienen la frecuencia adecuada, se escucha un sonido.
- Un **sistema digital puede generar sonidos** a través de un altavoz.
  - **Generando una señal digital periódica** a través de uno de sus pines.





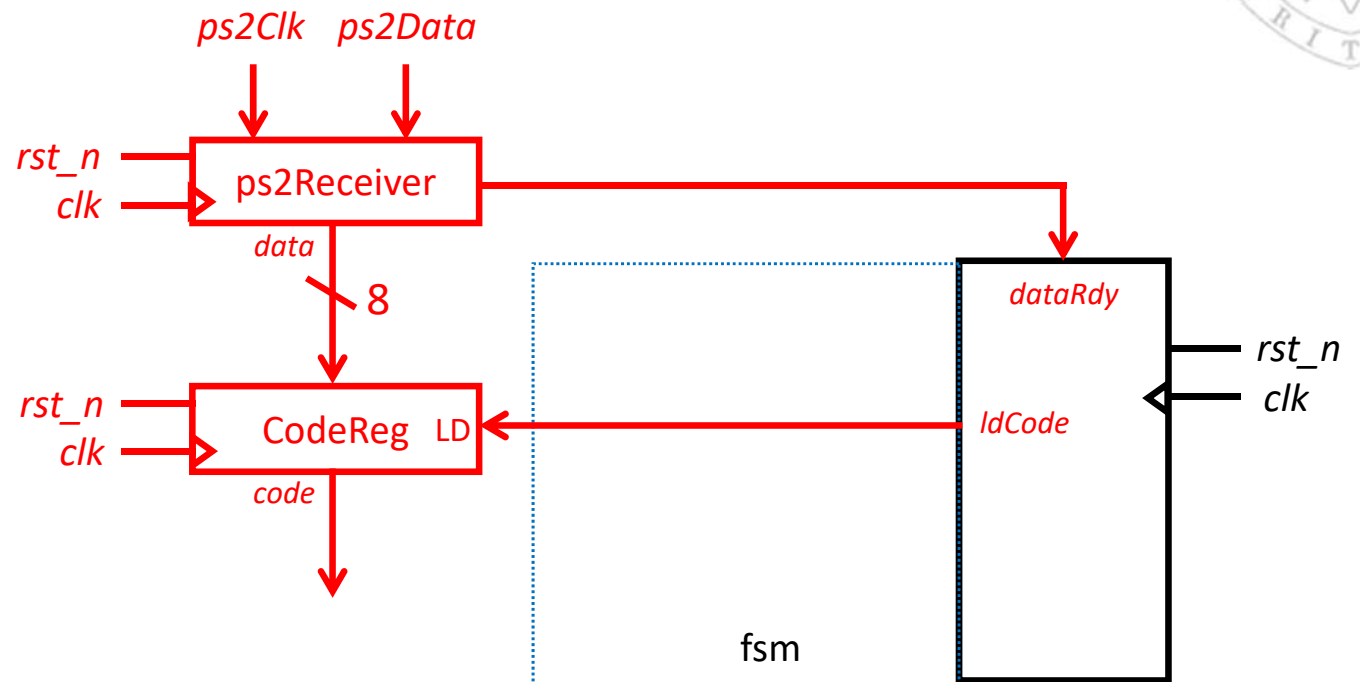
# Diseño principal

## diagrama RTL (i)



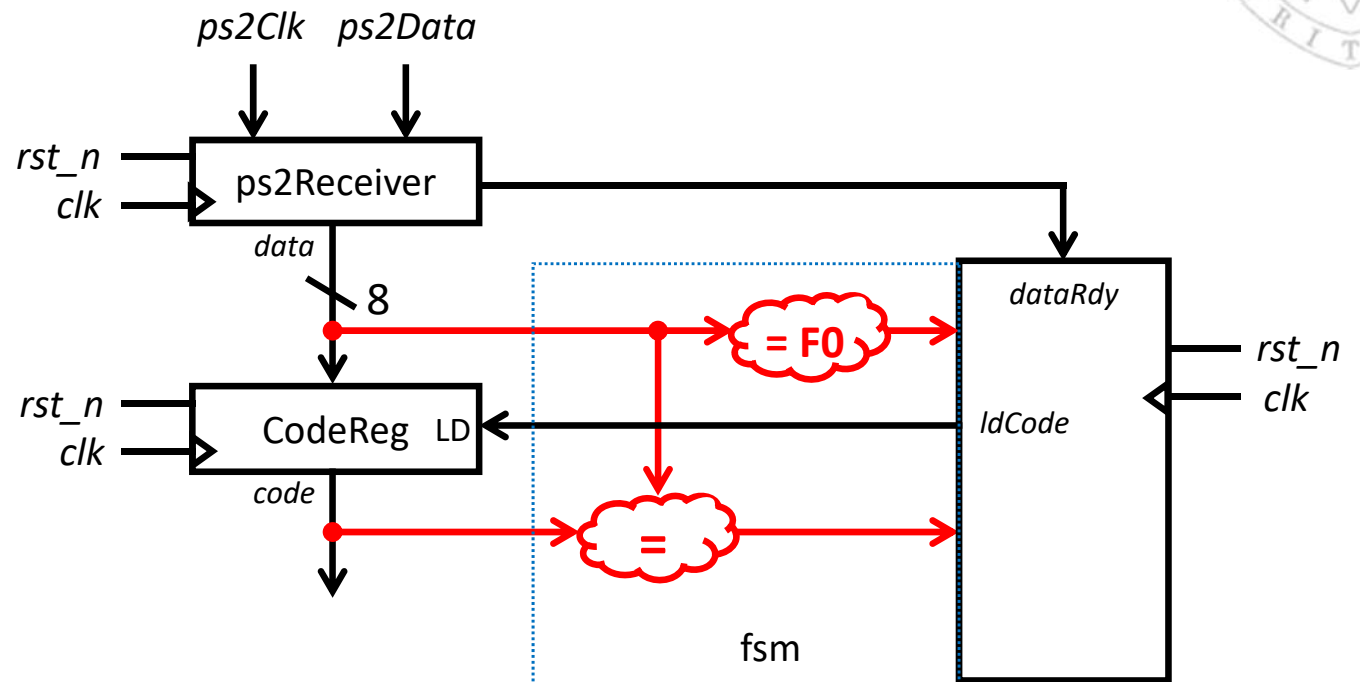
# Diseño principal

## diagrama RTL (i)



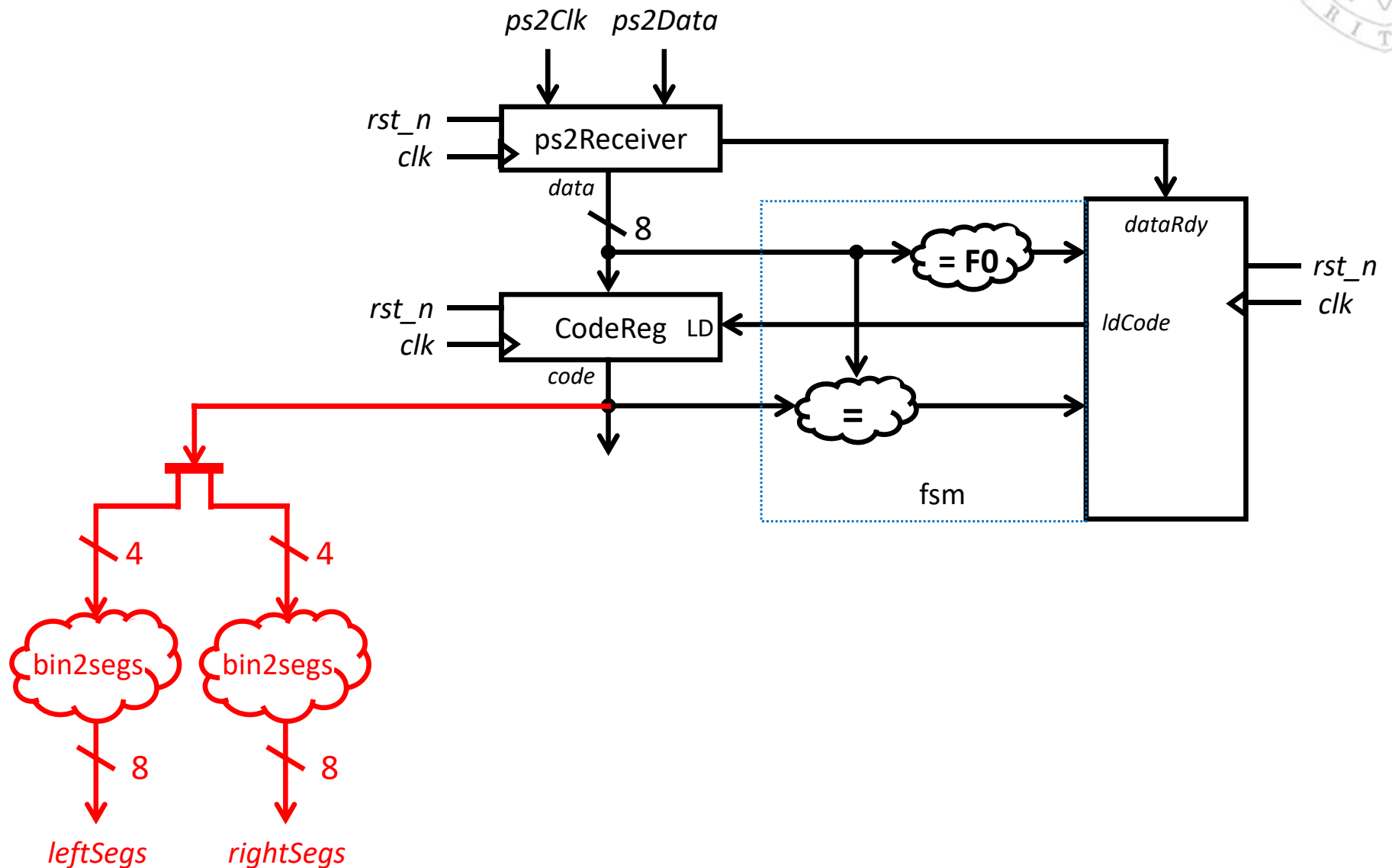
# Diseño principal

## diagrama RTL (i)



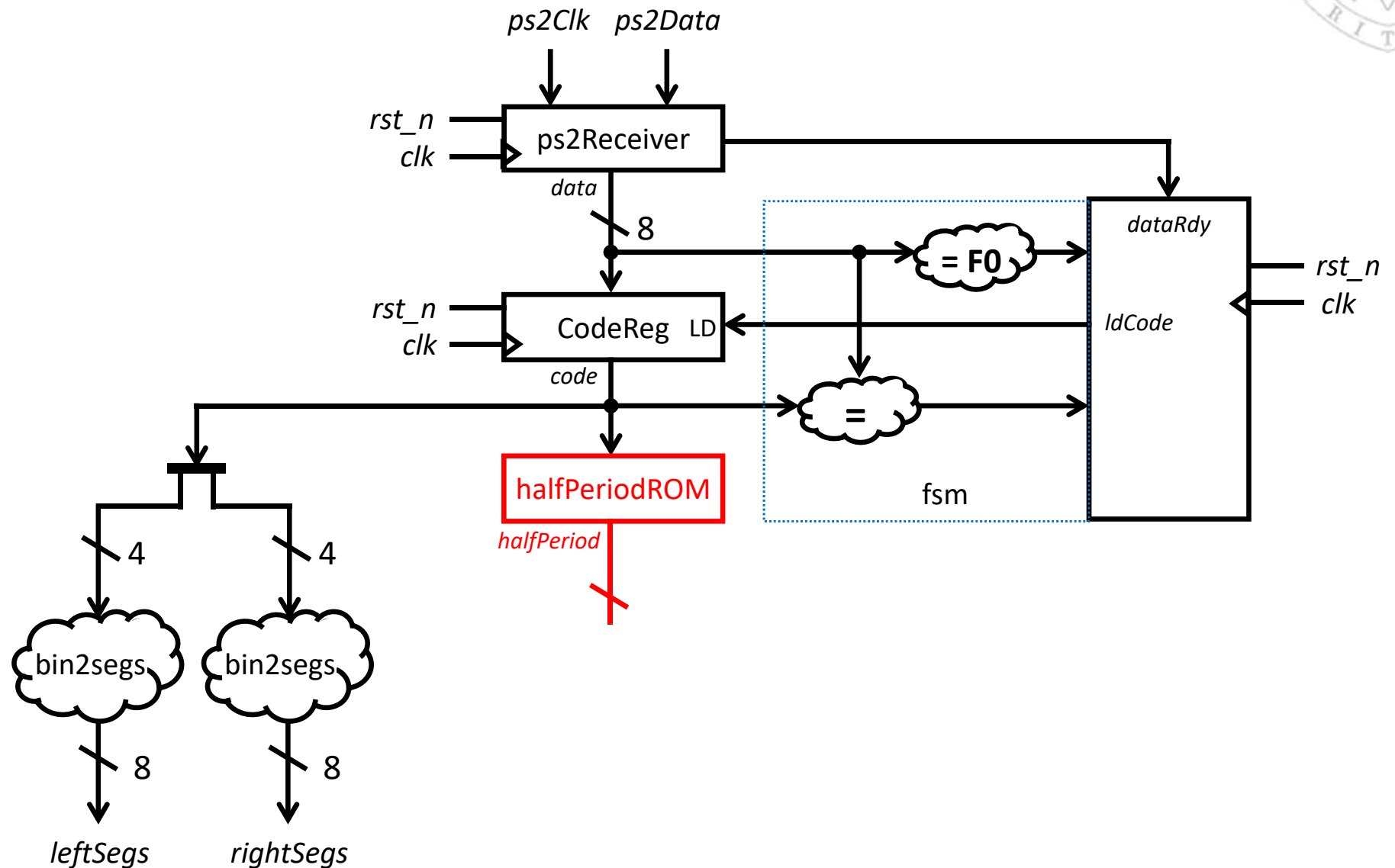
# Diseño principal

## diagrama RTL (i)



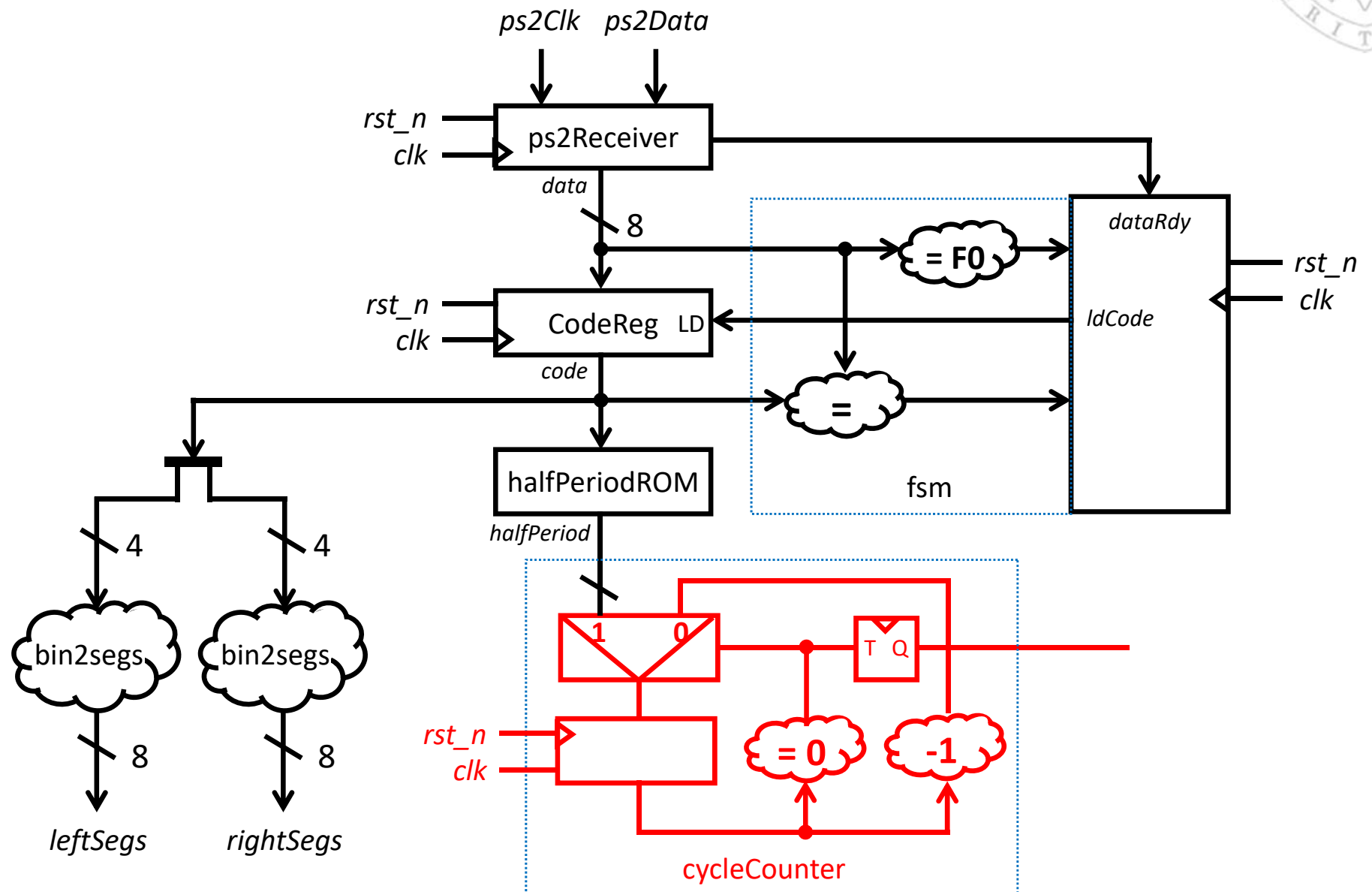
# Diseño principal

## diagrama RTL (i)



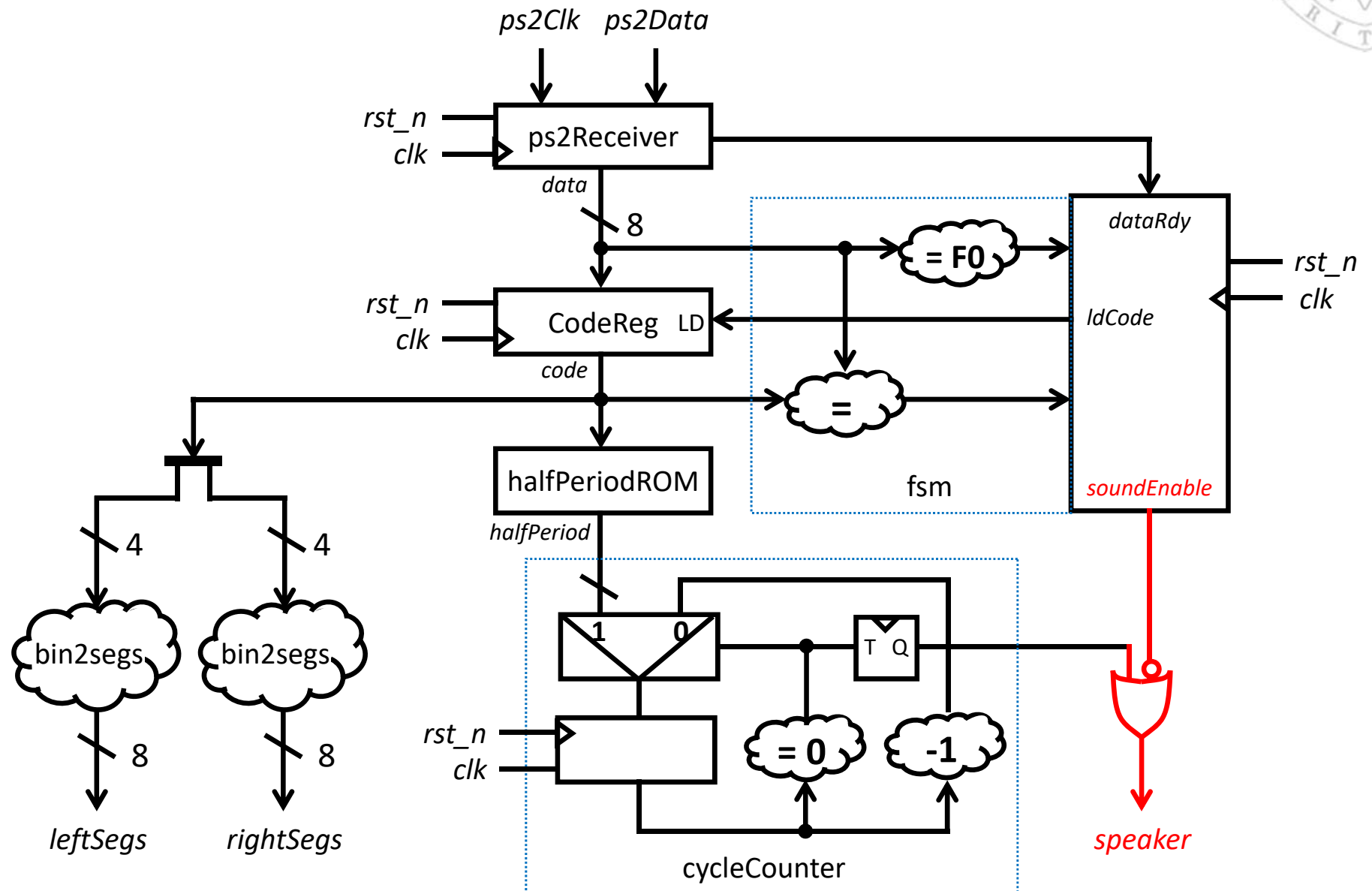
# Diseño principal

## diagrama RTL (i)



# Diseño principal

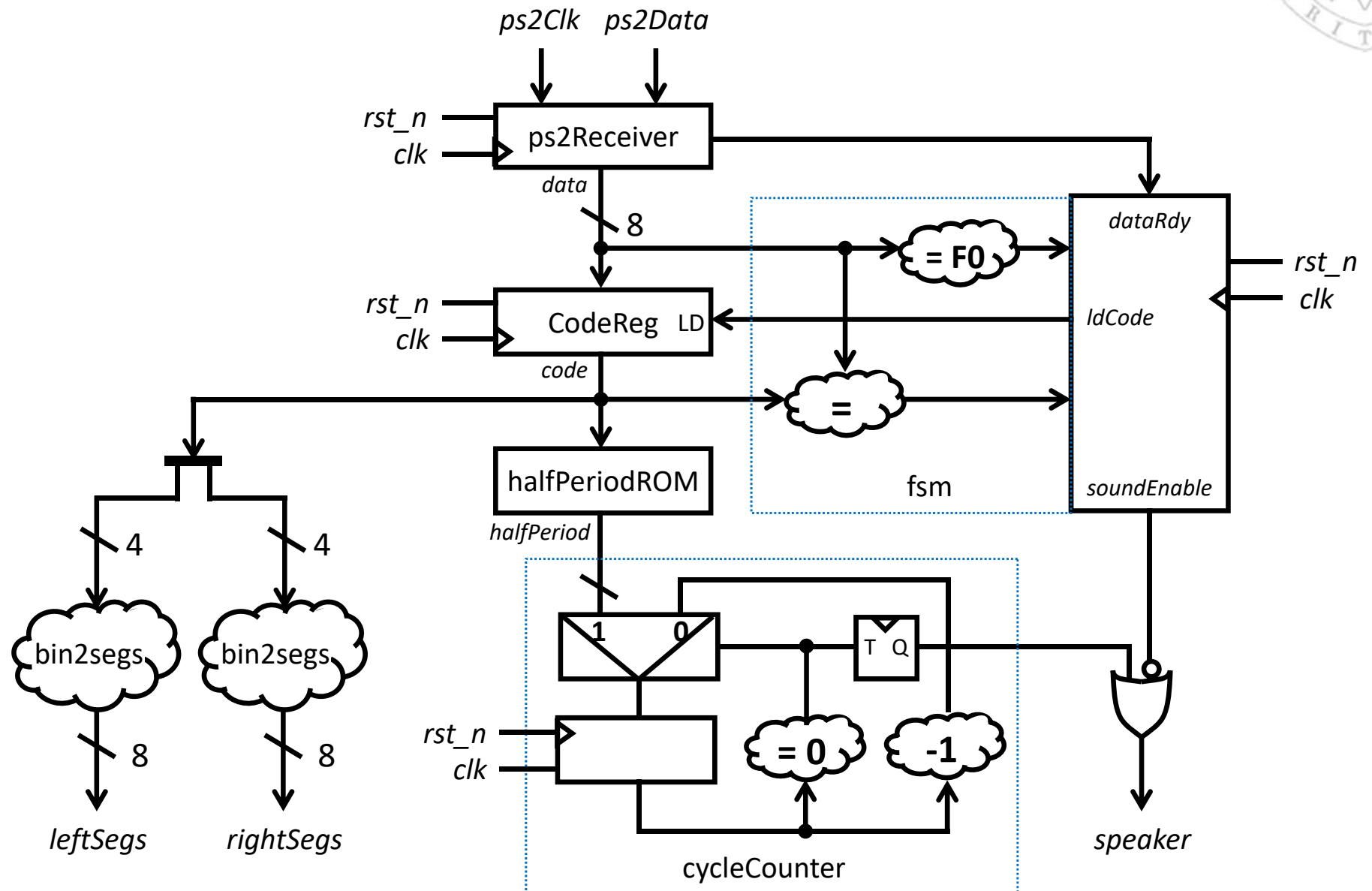
## diagrama RTL (i)





# Diseño principal

## diagrama RTL (i)

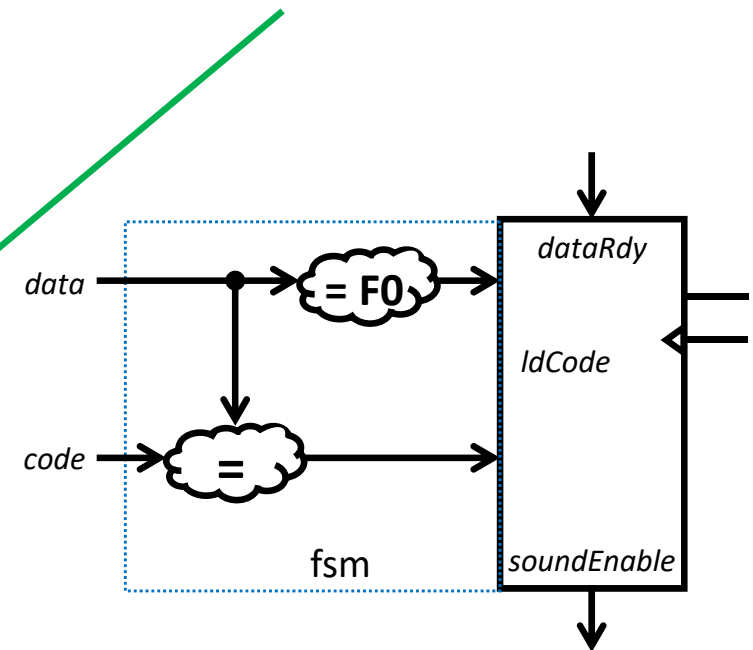
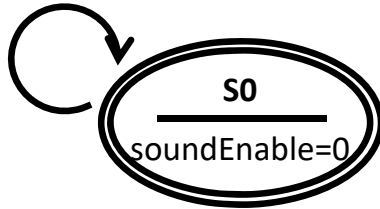


# Diseño principal

## diagrama RTL (ii)

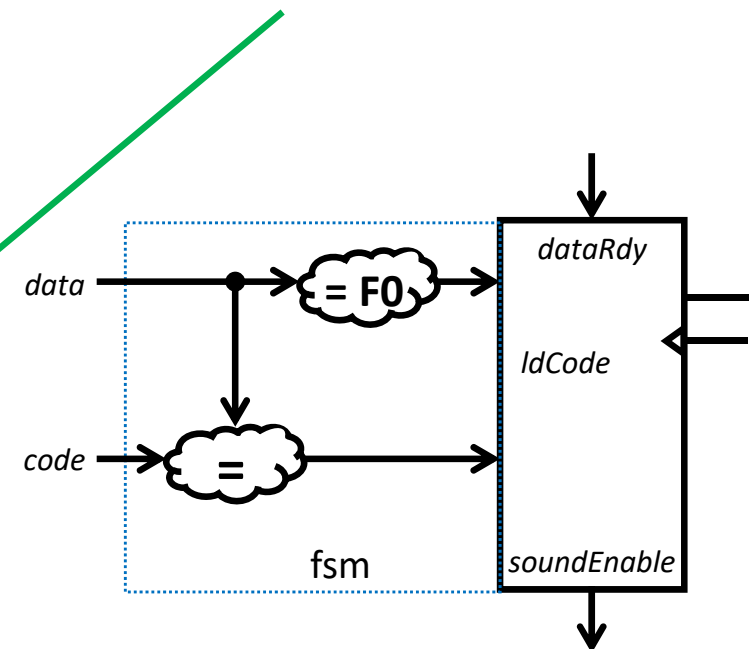
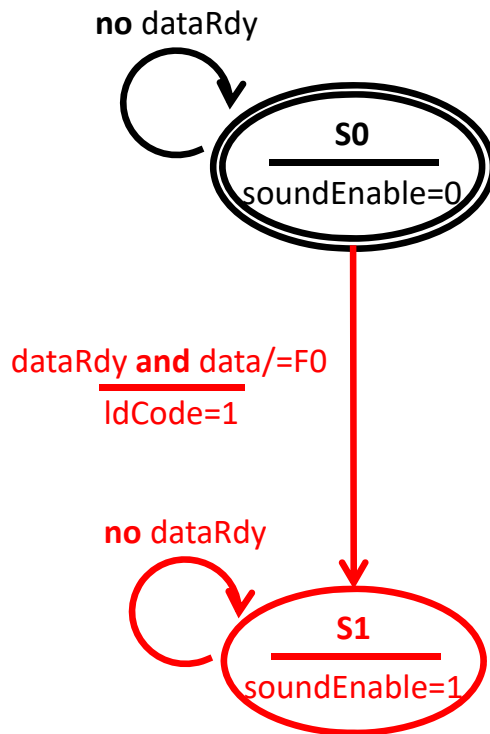


no dataRdy



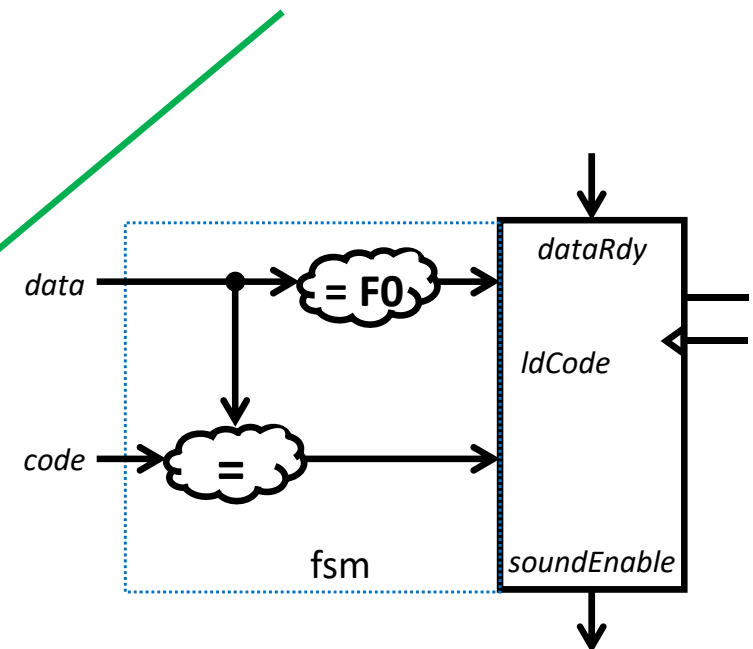
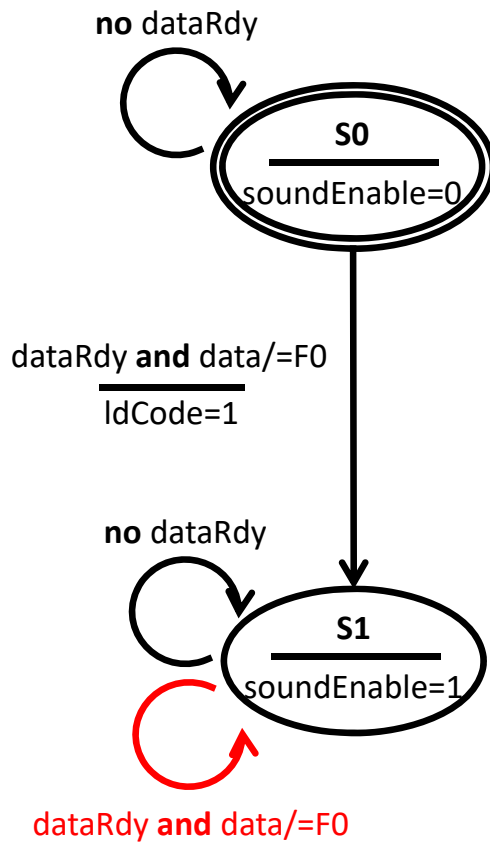
# Diseño principal

## diagrama RTL (ii)



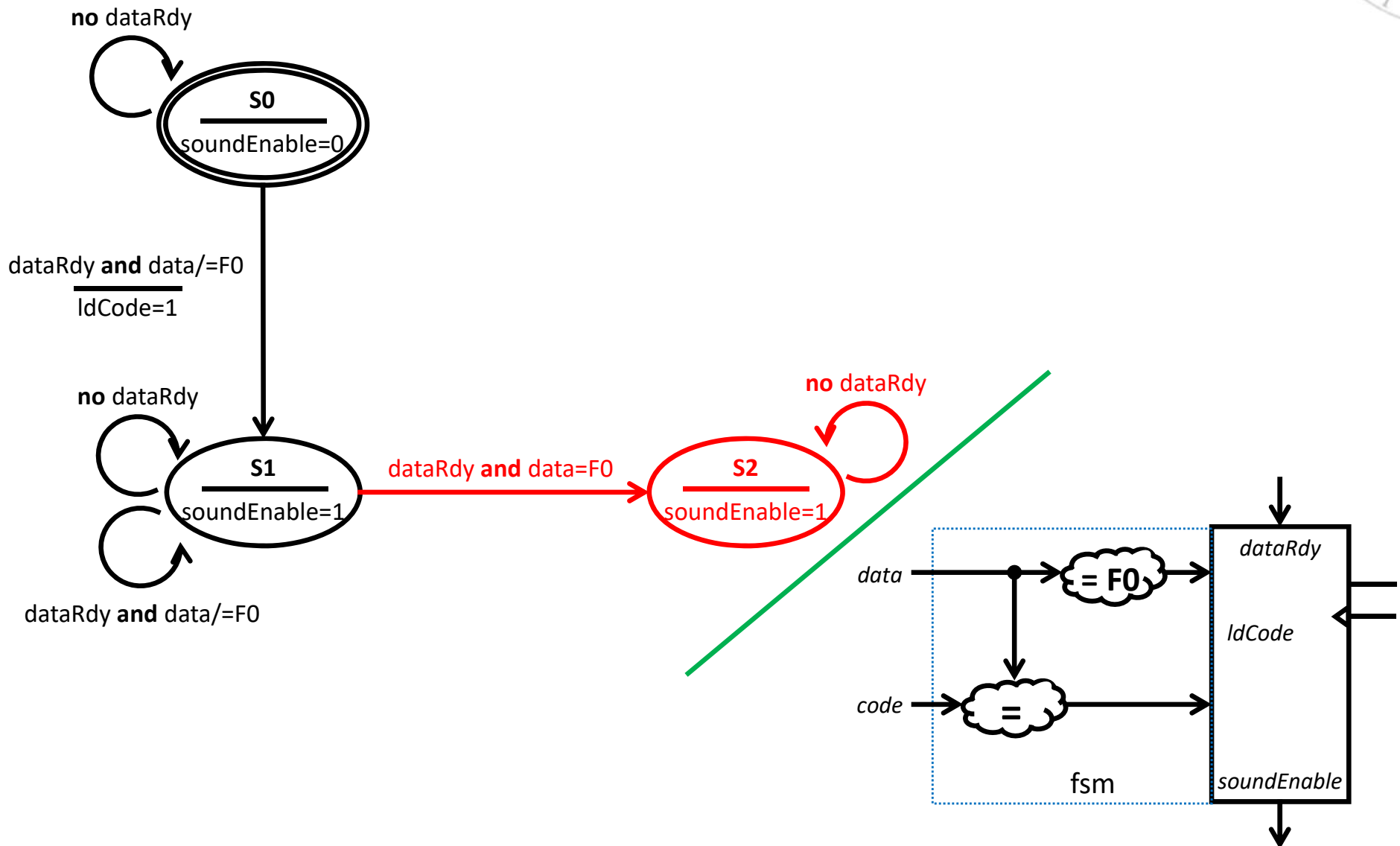
# Diseño principal

## diagrama RTL (ii)



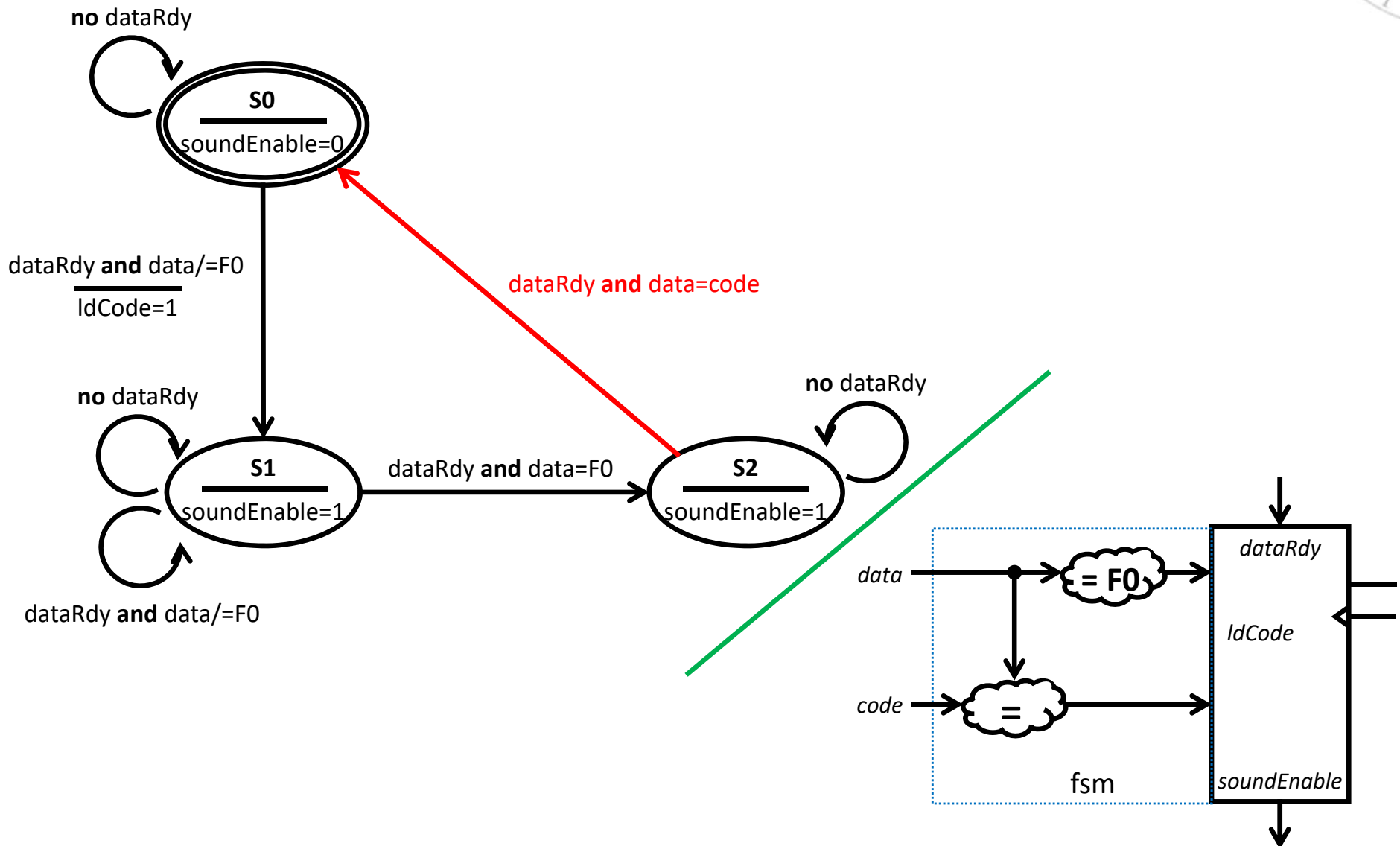
# Diseño principal

## diagrama RTL (ii)



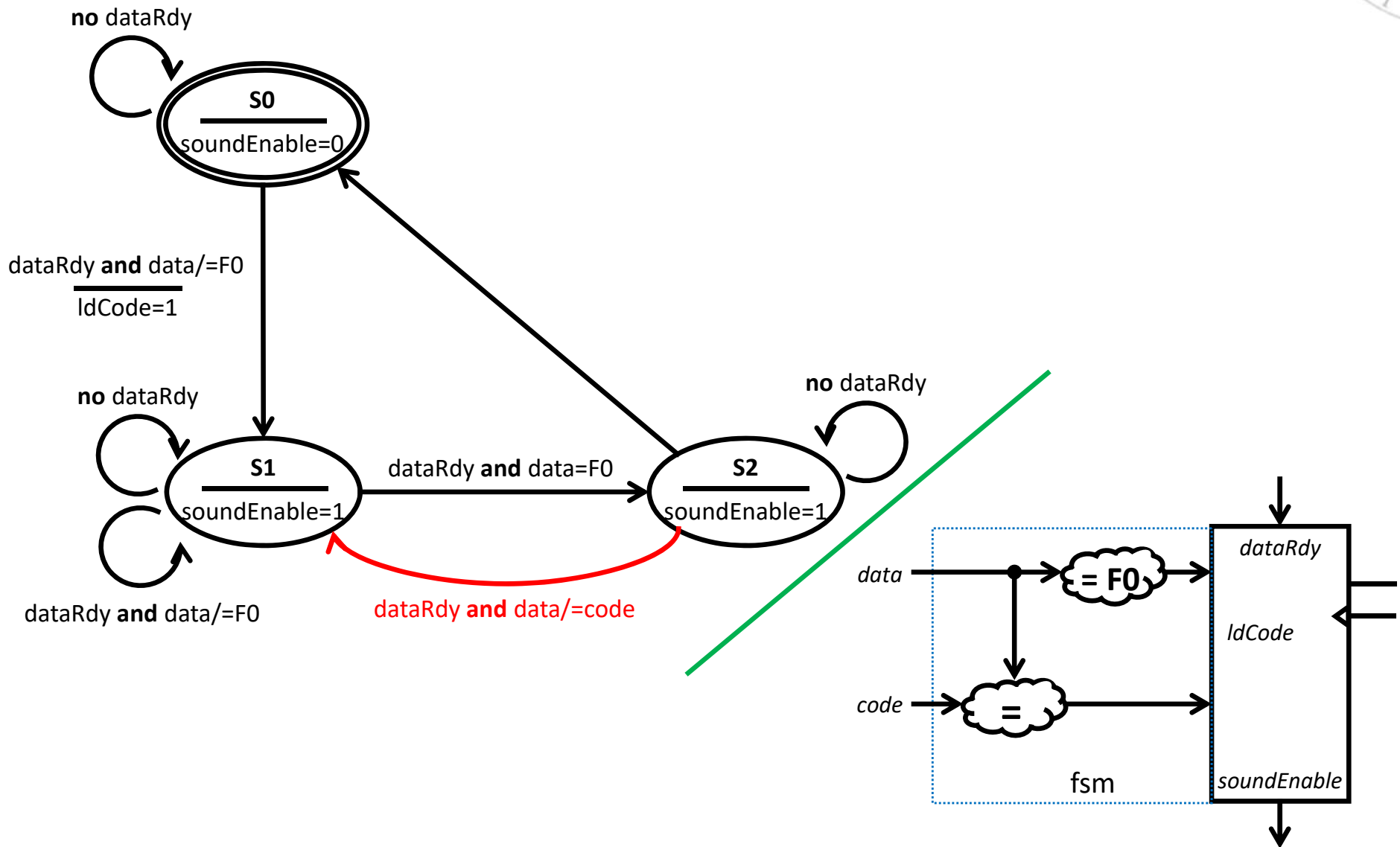
# Diseño principal

## diagrama RTL (ii)



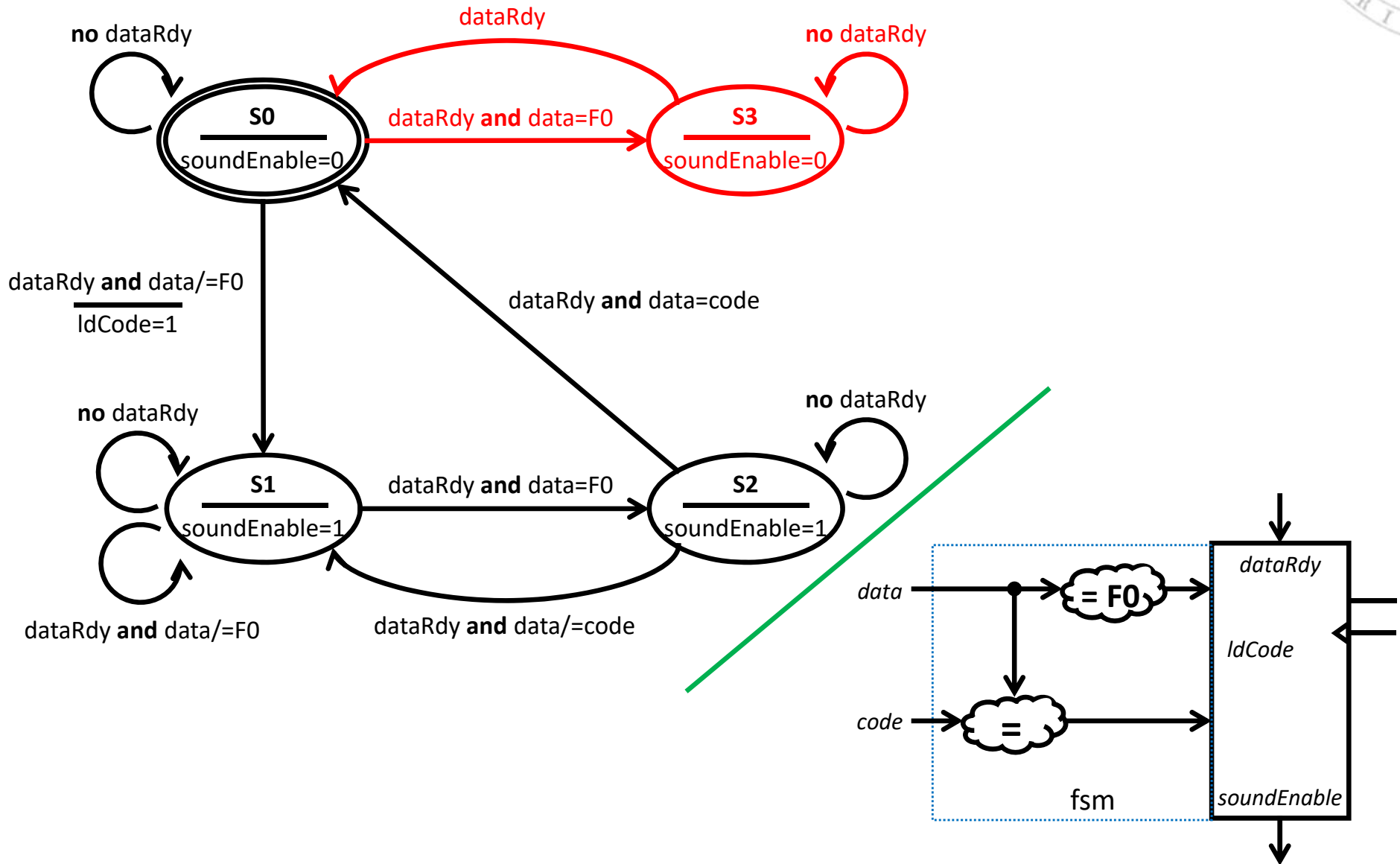
# Diseño principal

## diagrama RTL (ii)



# Diseño principal

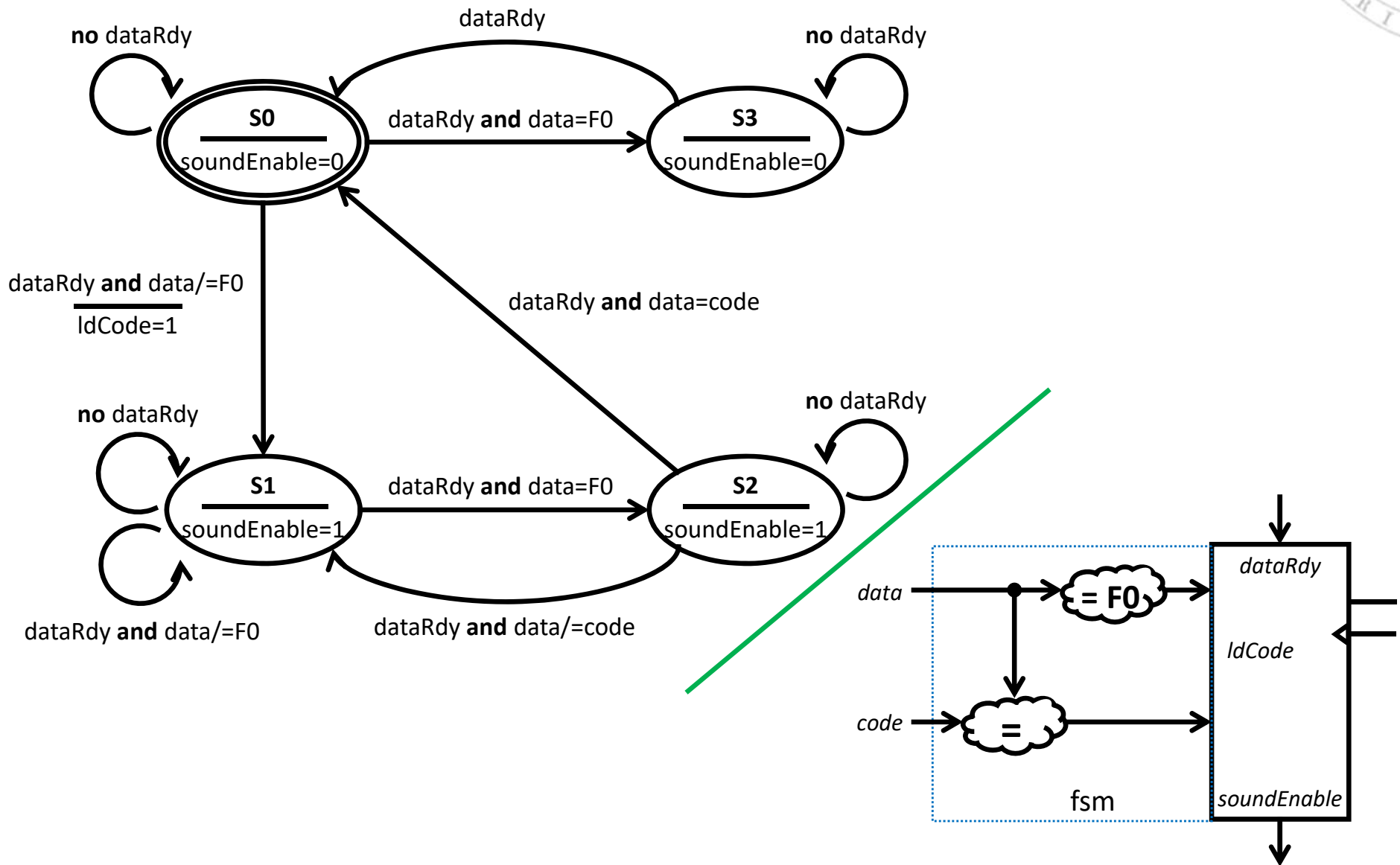
## diagrama RTL (ii)





# Diseño principal

## diagrama RTL (ii)



# Diseño principal

## lab4.vhd



```
architecture syn of lab4 is
    constant CLKFREQ : natural := 50_000_000;  -- frecuencia del reloj en MHz
    signal dataRdy : std_logic;
    signal code, data : std_logic_vector(7 downto 0);
    signal ldCode : std_logic;
    signal halfPeriod : natural;
    signal speakerTFF, soundEnable: std_logic;
begin

    ps2KeyboardInterface : ps2Receiver
        ...;

    codeRegister :
    process (rst_n, clk)
    begin
        if rst_n='0' then
            code <= ...;
        elsif rising_edge(clk) then
            ...
        end if;
    end process;

    leftConverter : bin2segs
        ...;

    righthConverter : bin2segs
        ...;
```

# Diseño principal

## lab4.vhd



```
halfPeriodROM :  
with code select  
    halfPeriod <=  
        FREQ/(2*262) when X"1c", -- A = Do  
        ...          when X"1d", -- W = Do#  
        ...  
        0 when others;  
  
cycleCounter :  
process (rst_n, clk)  
    variable count : natural;  
begin  
    if rst_n='0' then  
        speakerTFF <= ...;  
        count := ...;  
    elsif rising_edge(clk) then  
        ...  
    end if;  
end process;  
  
fsm:  
process (rst_n, clk, dataRdy, data, code)  
    type states is (S0, S1, S2, S3);  
    variable state: states;  
begin  
    ...  
end process;  
  
speaker <= speakerTFF when ... else ...;  
end syn;
```

# Tareas



1. Crear el proyecto **lab4** en el directorio **DAS**
2. Descargar de la Web en el directorio **common** el fichero **ps2receiver.vhd**
3. Descargar de la Web en el directorio **lab4** los ficheros:
  - **ps2receivertest.vhd**, **lab4.vhd** y **lab4.ucf**
4. Completar el fichero **common.vhd** con la declaración del nuevo componente reusable.
5. Completar el código omitido en los ficheros:
  - **ps2receiver.vhd** y **lab4.vhd**
6. Añadir al proyecto los ficheros:
  - **Asociado a simulación:** **ps2receivertest.vhd**
  - **Asociado a implementación:** **bin2segs.vhd**, **lab4.vhd** y **lab4.ucf**
  - **Asociado a ambas:** **common.vhd**, **synchronizer.vhd**, **edgedetector.vhd**, y **ps2receiver.vhd**
7. Simular durante 600 ms la entidad cerrada **ps2receiverTester**
8. Sintetizar, implementar la entidad **lab4** y generar el fichero de configuración.
9. Conectar el teclado y el altavoz a placa y encenderla.
10. Descargar el fichero **lab4.bit**



# Acerca de *Creative Commons*



## ■ Licencia CC (*Creative Commons*)

- Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:



**Reconocimiento** (*Attribution*):

En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.



**No comercial** (*Non commercial*):

La explotación de la obra queda limitada a usos no comerciales.



**Compartir igual** (*Share alike*):

La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: <https://creativecommons.org/licenses/by-nc-sa/4.0/>