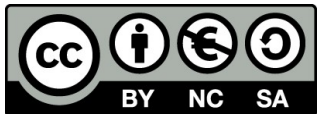
Laboratorio 4:

Validación e instrumentación

comunicación serie síncrona por bus PS-2

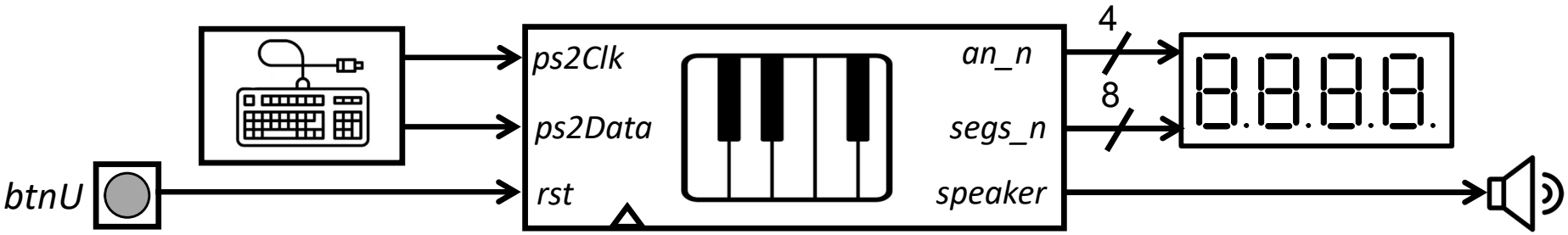
Diseño automático de sistemas

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.
  - Dispondrá de reset síncrono.
- El teclado y el altavoz se conectarán del siguiente modo:
  - Como teclado se usará un teclado USB convencional.
    - El scancode de cada tecla pulsada se mostrará en los displays centrales del display 7-segs
  - 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 1 del PMOD JC, el otro a 3.3V
    - Las notas se generan enviando a dicho pin una onda cuadrada de una cierta frecuencia.



Protocolo PS/2

algo de historia: teclados serie

- Los antiguos teclados o ratones de tipo AT-PS/2 utilizaban para comunicarse con un host un bus bidireccional serie síncrono.
  - Dispone de dos líneas a colector abierto con resistencias de pull-up 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

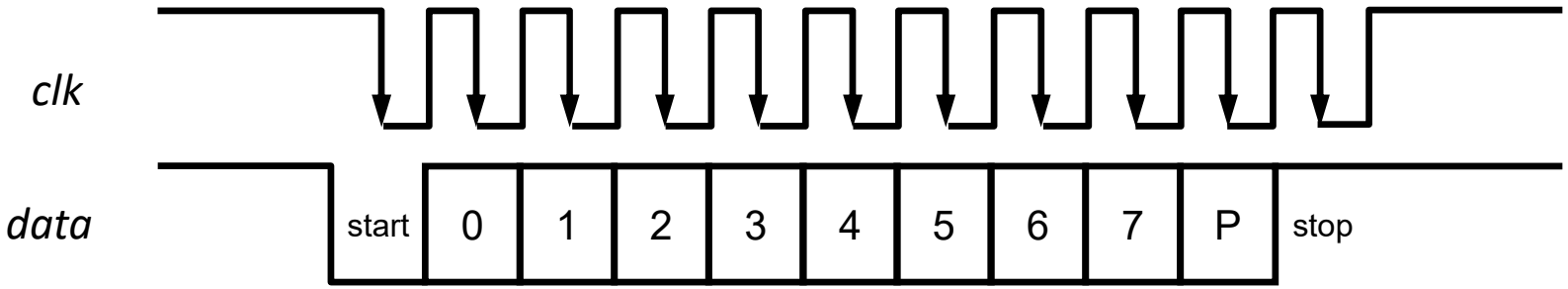
1. KBD Clock
2. GND
3. KBD Data
4. N/C
5. +5V (VCC)
6. N/C

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 si fija a baja CLK antes del 10º ciclo.
    - El dato cuya transmisión se aborta y siguientes se almacenan en el dispositivo.
    - Su retransmisión debe ser solicitada por el host.

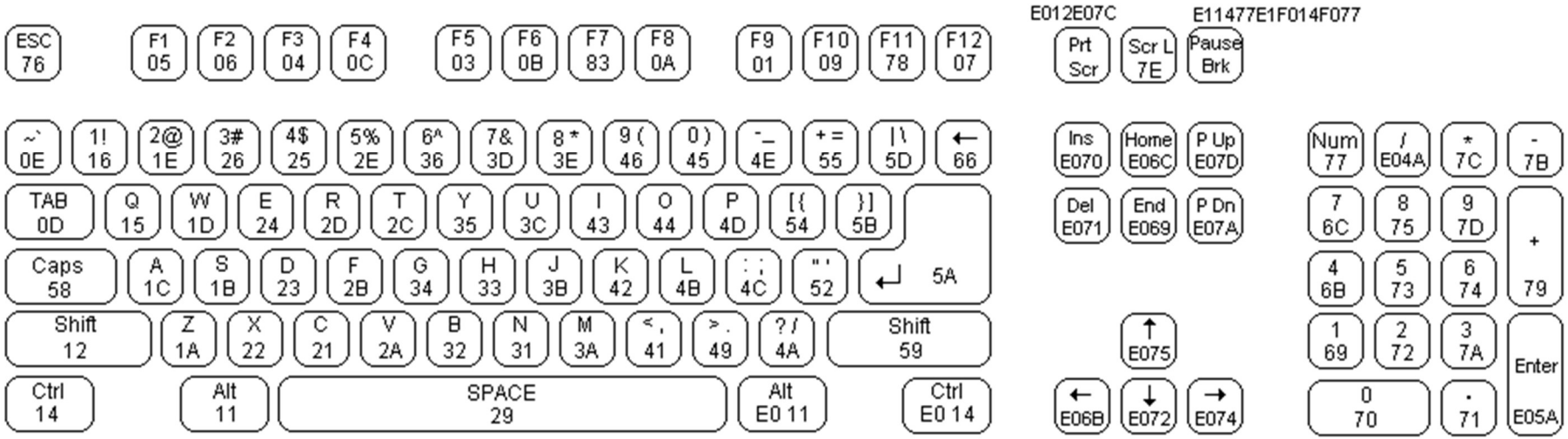


J.M. Mendías
2024

Protocolo PS/2

teclado: scancodes (i)

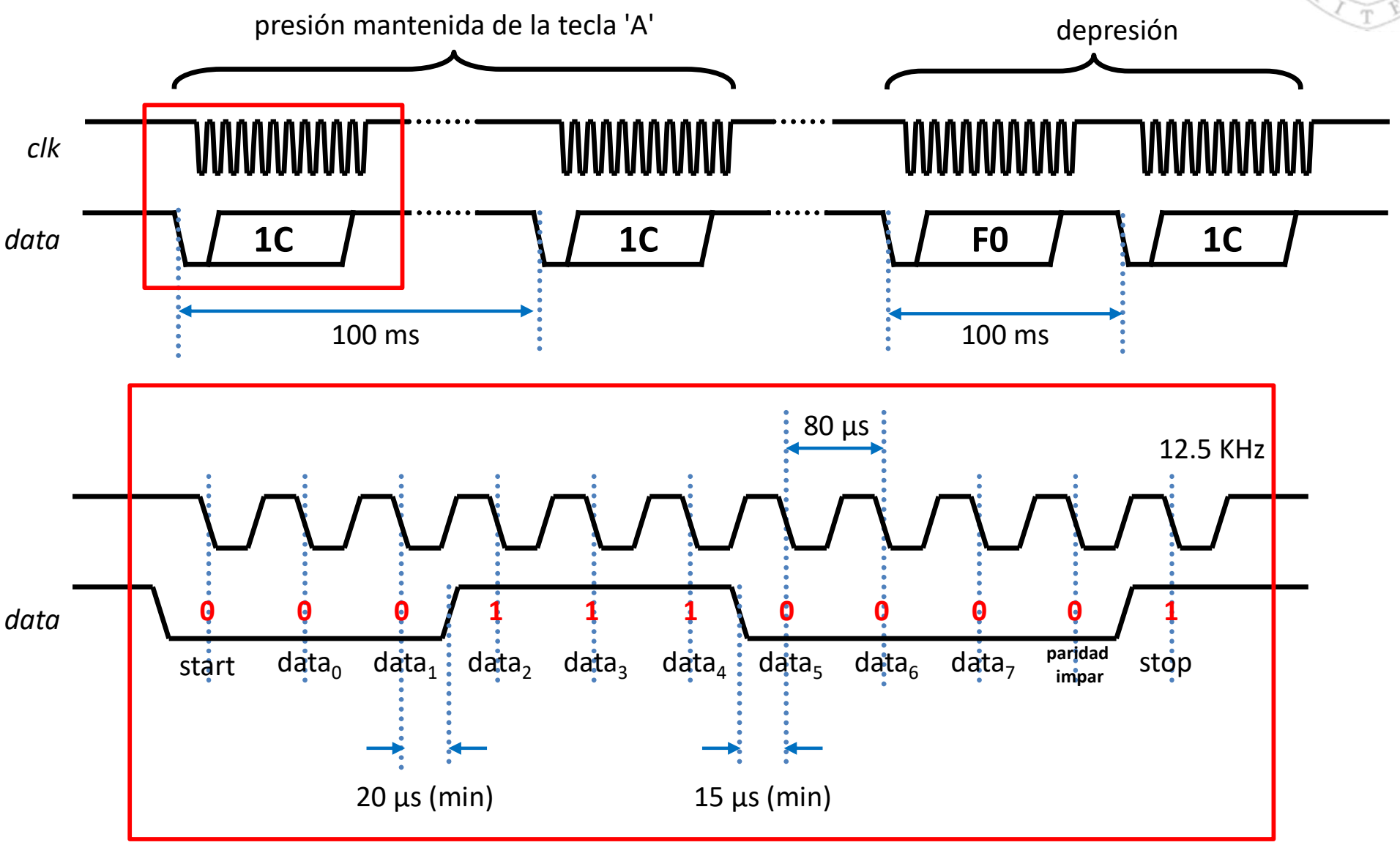
- Tras su enchufado, el teclado envía Power-on satisfactorio (AA).
- Después, cada vez que se presiona una tecla, envía de 3 a n códigos:
  - Scancode de presión: indica el momento de la presión y la tecla pulsada.
    - Si la tecla permanece pulsada el teclado reenvía el código periódicamente.
    - 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.



J.M. Mendías 2024

Protocolo PS/2

teclado: scancodes (ii)

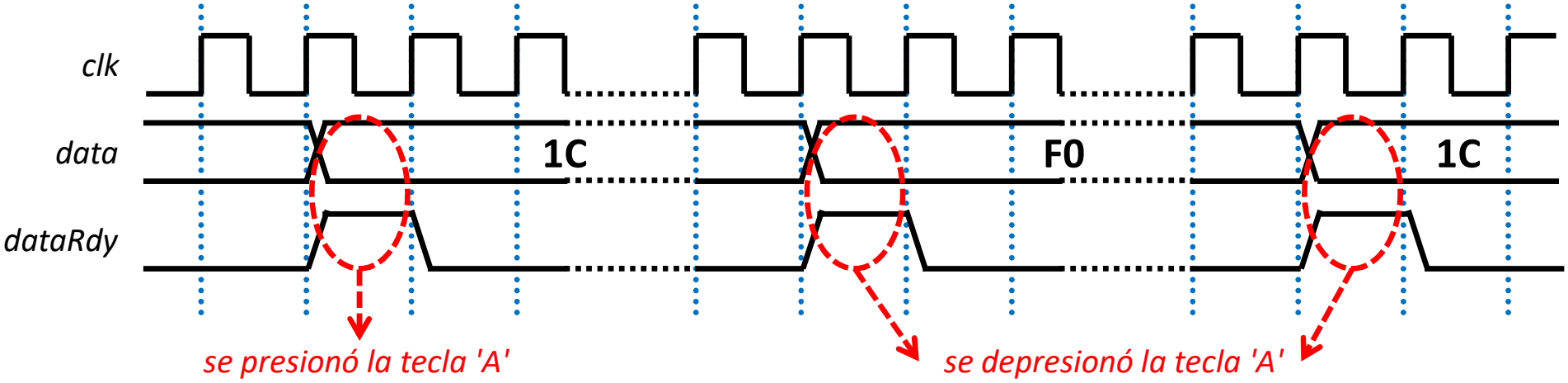


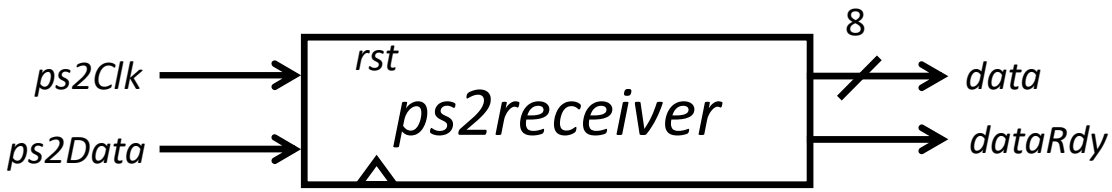
J.M. Mendías
2024

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.
  - Dispondrá de reset síncrono activo a alta.

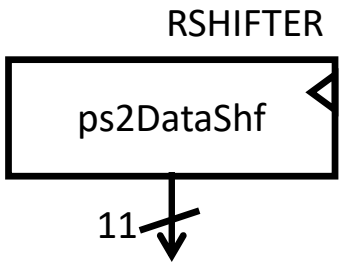




J.M. Mendías
2024

Interfaz PS/2

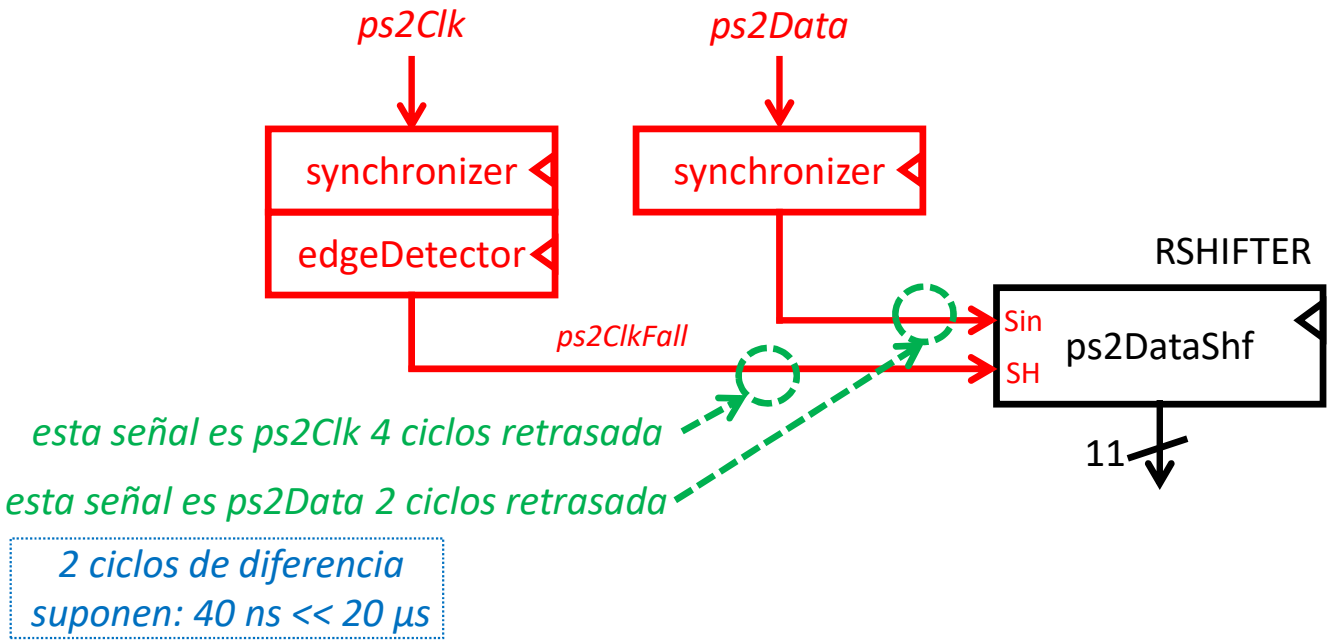
esquema RTL



J.M. Mendías 2024

Interfaz PS/2

esquema RTL

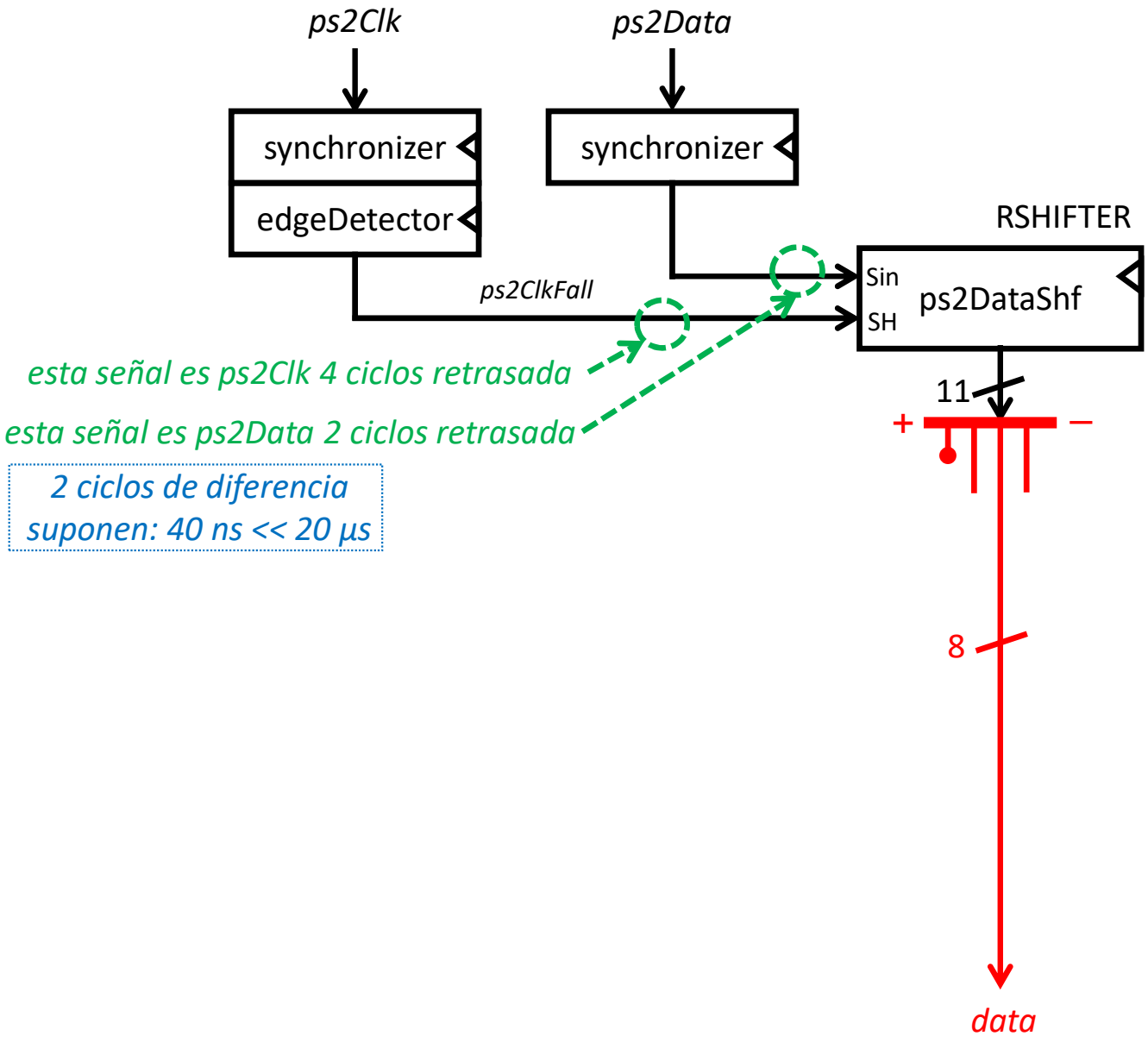


2 ciclos de diferencia suponen: 40 ns << 20 µs

J.M. Mendías
2024

Interfaz PS/2

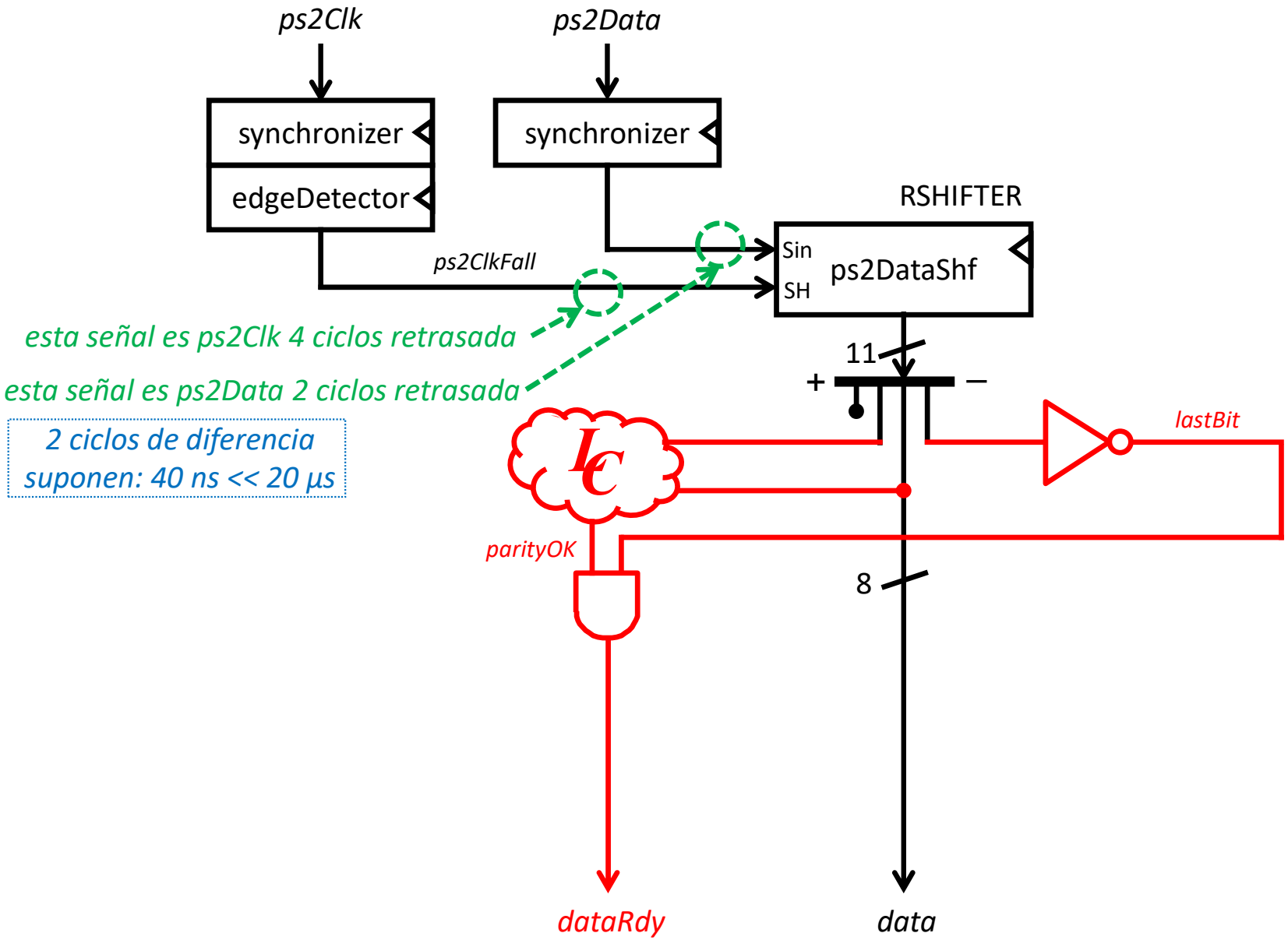
esquema RTL



J.M. Mendías
2024

Interfaz PS/2

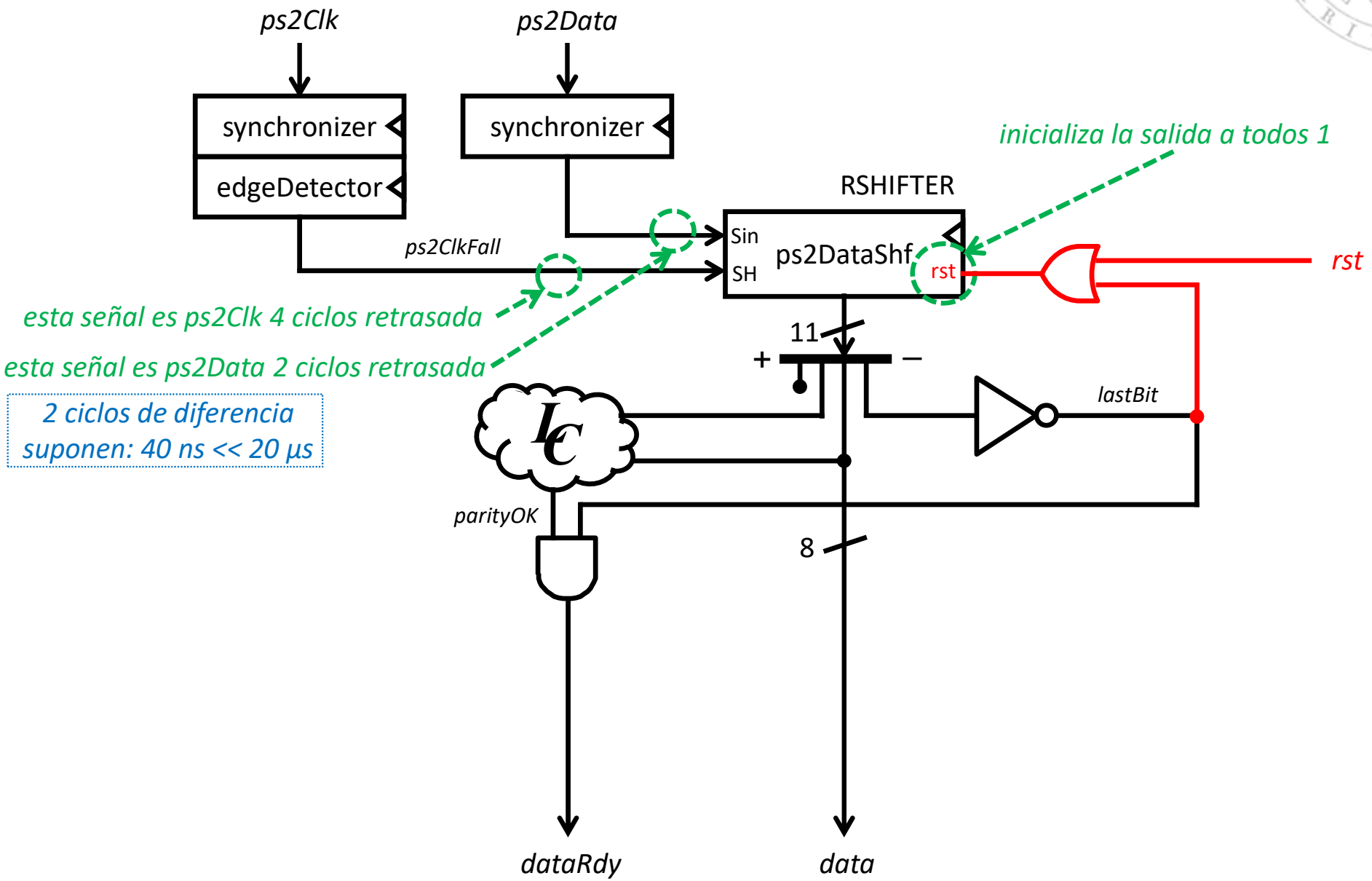
esquema RTL



J.M. Mendías 2024

Interfaz PS/2

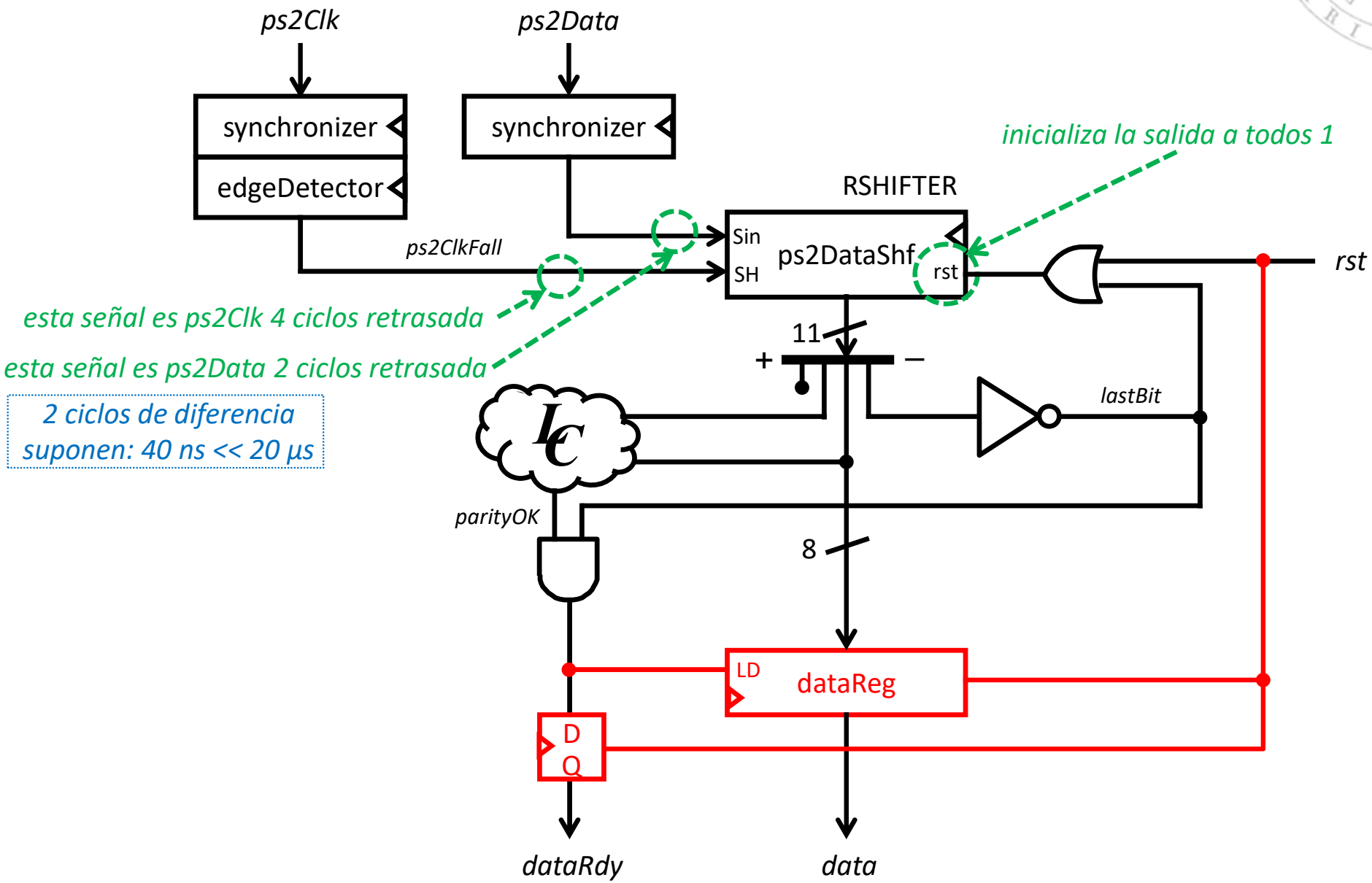
esquema RTL



J.M. Mendías
2024

Interfaz PS/2

esquema RTL



Interfaz PS/2

ps2receiver.vhd

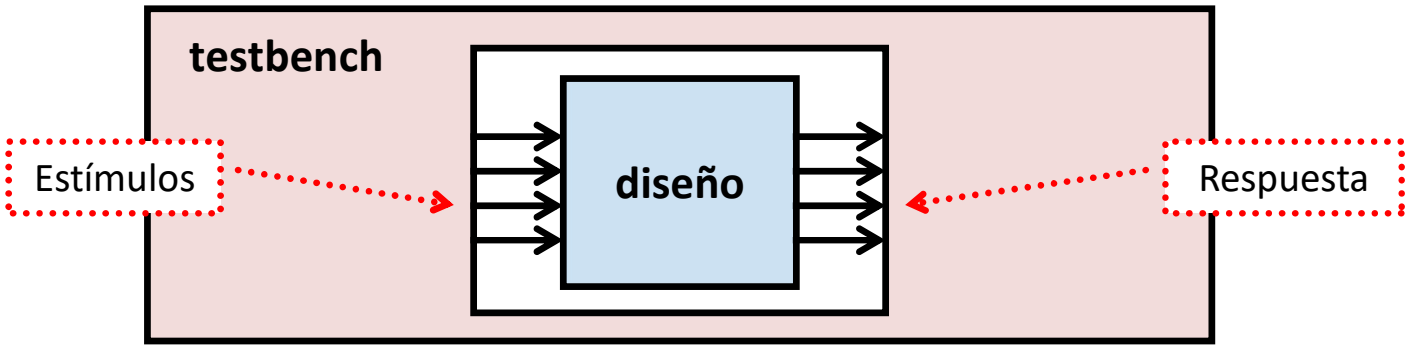
```
architecture syn of ps2receiver is
  signal ps2DataShf: std_logic_vector(10 downto 0) := (others =>'1');
  signal ps2ClkSync, ps2DataSync, ps2ClkFall: std_logic;
  signal lastBit, parityOK: std_logic;
begin
  ps2ClkSynchronizer  : synchronizer
    ...;
  ps2DataSynchronizer : synchronizer
    ...;
  ps2ClkEdgeDetector  : edgeDetector
    ...;
  ps2DataShifter:
  process (clk)
  begin
    if rising_edge(clk) then
      ...
    end if;
  end process;

  lastBitCheker :
  lastBit <= ...;

...
```

```
...
  oddParityCheker :
  process (ps2DataShf)
    variable aux : std_logic;
  begin
    aux := ...;
    for i in ... loop
      ...;
    end loop;
    parityOK <= aux;
  end;

  outputRegisters:
  process (clk)
    begin
      if rising_edge(clk) then
        ...
      end if;
    end process;

end syn;
```

Uso de `for` *para especificación de lógica combinacional*

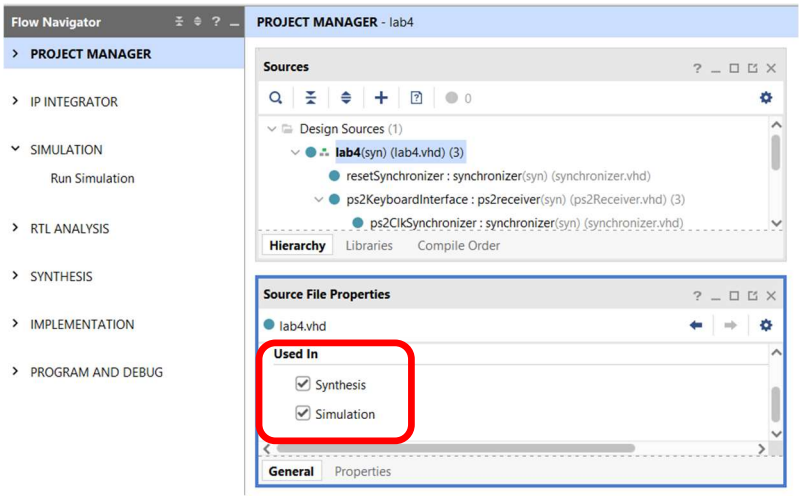
Validación por simulación

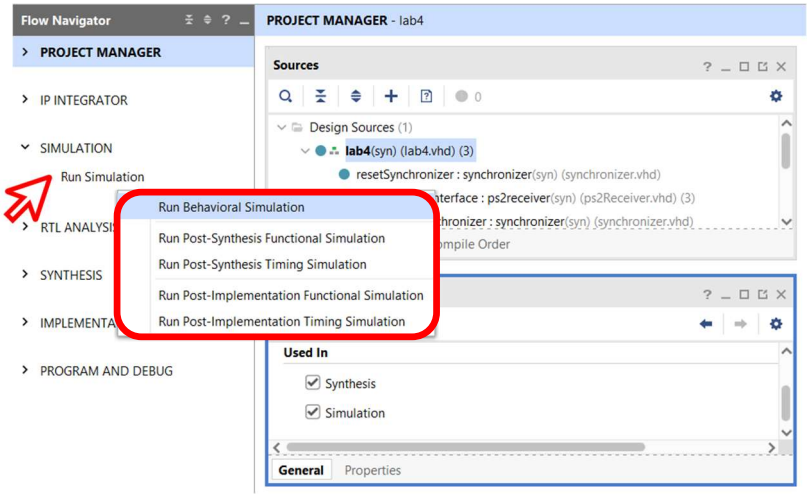
- Para validar un diseño es necesario estimular sus entradas y comprobar que sus salidas son las correctas.
  - Los simuladores facilitan entornos gráficos para el estímulo y visualización.
- Cuando se usa VHDL, se puede usar el lenguaje para todo:
  - Definir el comportamiento del diseño.
  - Definir 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.



Validación por simulación

- Vivado incorpora un simulador que permite efectuar:
  - Validaciones funcionales (nivel RT)
    - Simulaciones síncronas que ignoran los retardos de la lógica.
    - Pueden hacerse pre-síntesis, post-síntesis y post-implementación.
  - Validaciones temporales (nivel lógico)
    - Simulaciones que incorporan los retardos de la lógica.
    - Pueden hacerse post-síntesis y post-implementación.
- Por defecto, todos los ficheros pueden ser sintetizados y simulados:
  - Antes de arrancar la simulación debe indicarse cuales se usan para cada propósito.





J.M. Mendías
2024

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 100 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 (60 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.

J.M. Mendías
2024

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 := 10 ns;    -- Periodo del reloj (100 MHz)

  signal clk     : std_logic := '1';
  signal rst     : std_logic := '1';
  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;

...
```

J.M. Mendías
2024

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
    port map ( clk => clk, rst => rst, dataRdy => dataRdy,
               data => data, ps2Clk => ps2Clk, ps2Data => ps2Data );

  rstGen :
  rst <=
    '0' after (50 us + 5 ns),
    '1' after (500 ms + 5 ns),
    '0' 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 (clk)
  begin
    wait until clk'delayed='1';
    if rst='1'then
      assert dataRdy='0' and data=X"00"
        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

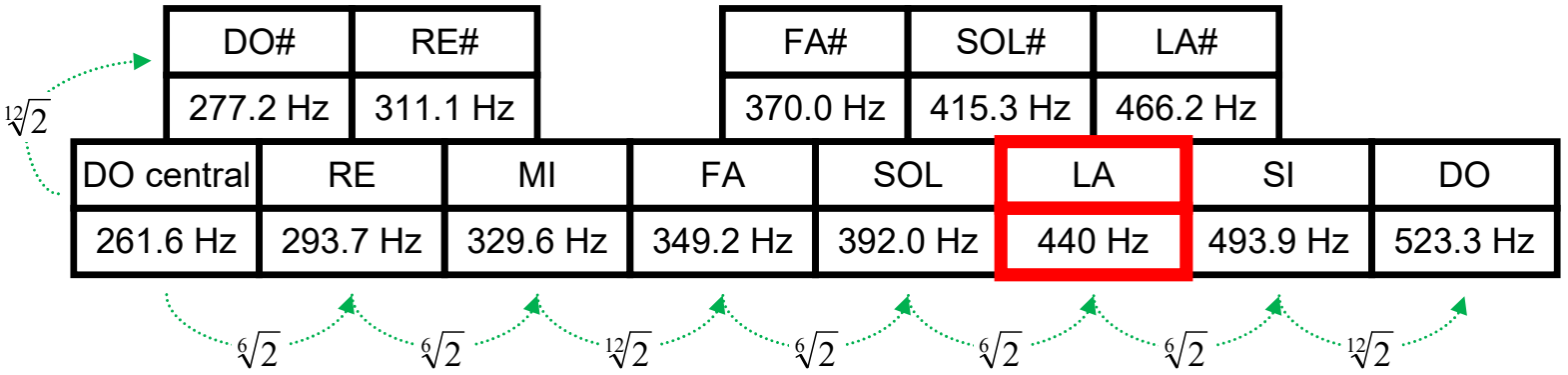
- Un sonido es una onda longitudinal mecánica con una frecuencia comprendida entre 20 y 20000 Hz.
  - El sonido lo percibimos como una variación periódica de la presión en nuestro tímpano generada por la oscilación periódica de las partículas del aire.
- Magnitudes objetivas del sonido:
  - Frecuencia: 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:

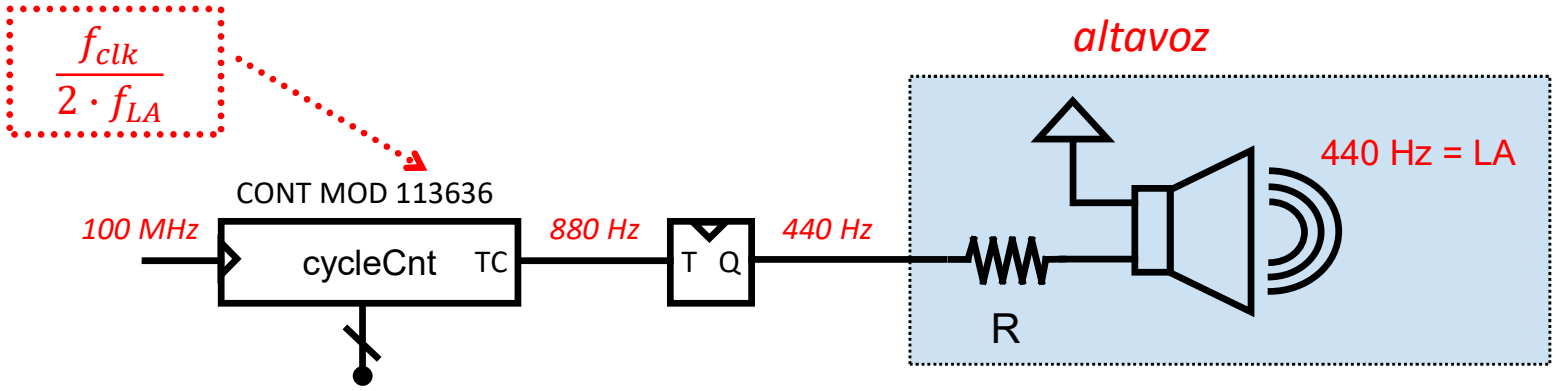
$$f_{nota} = f_{LA} \cdot \sqrt[12]{2^n} = 440 \cdot \sqrt[12]{2^n} \approx 440 \cdot 1.06^n$$

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



Generación digital de sonido

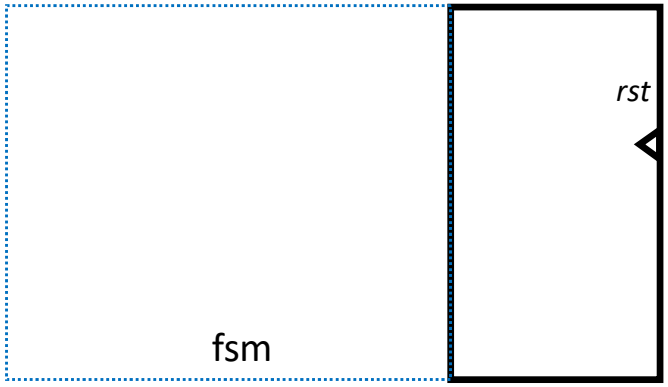
- Un altavoz, es un transductor capaz de generar una onda sonora análoga (en frecuencia y amplitud) a una señal eléctrica dada.
- Se compone de una membrana elástica unida a una bobina móvil que se monta dentro del campo magnético de un imán permanente.
  - La fuerza de atracción entre la bobina y el 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.



J.M. Mendías
2024

Diseño principal

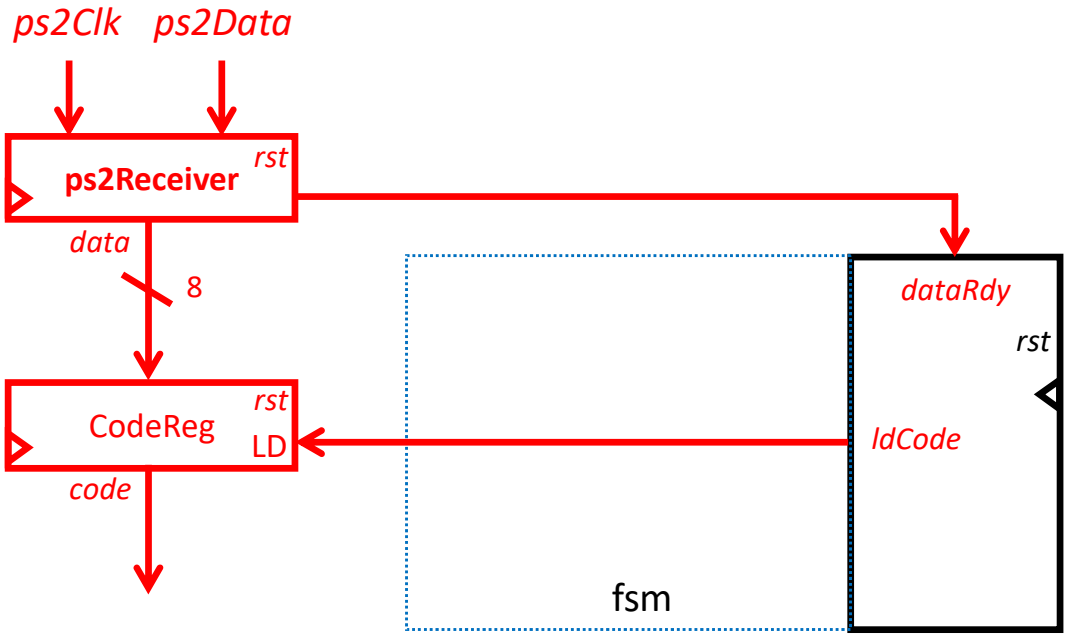
esquema RTL (i)



J.M. Mendías
2024

Diseño principal

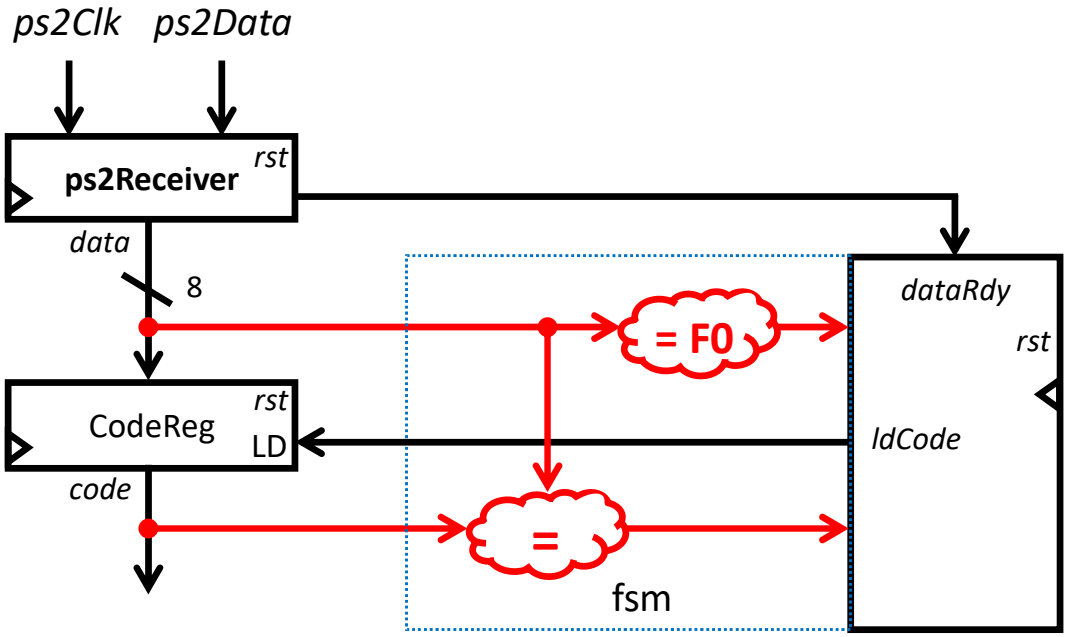
esquema RTL (i)



J.M. Mendías
2024

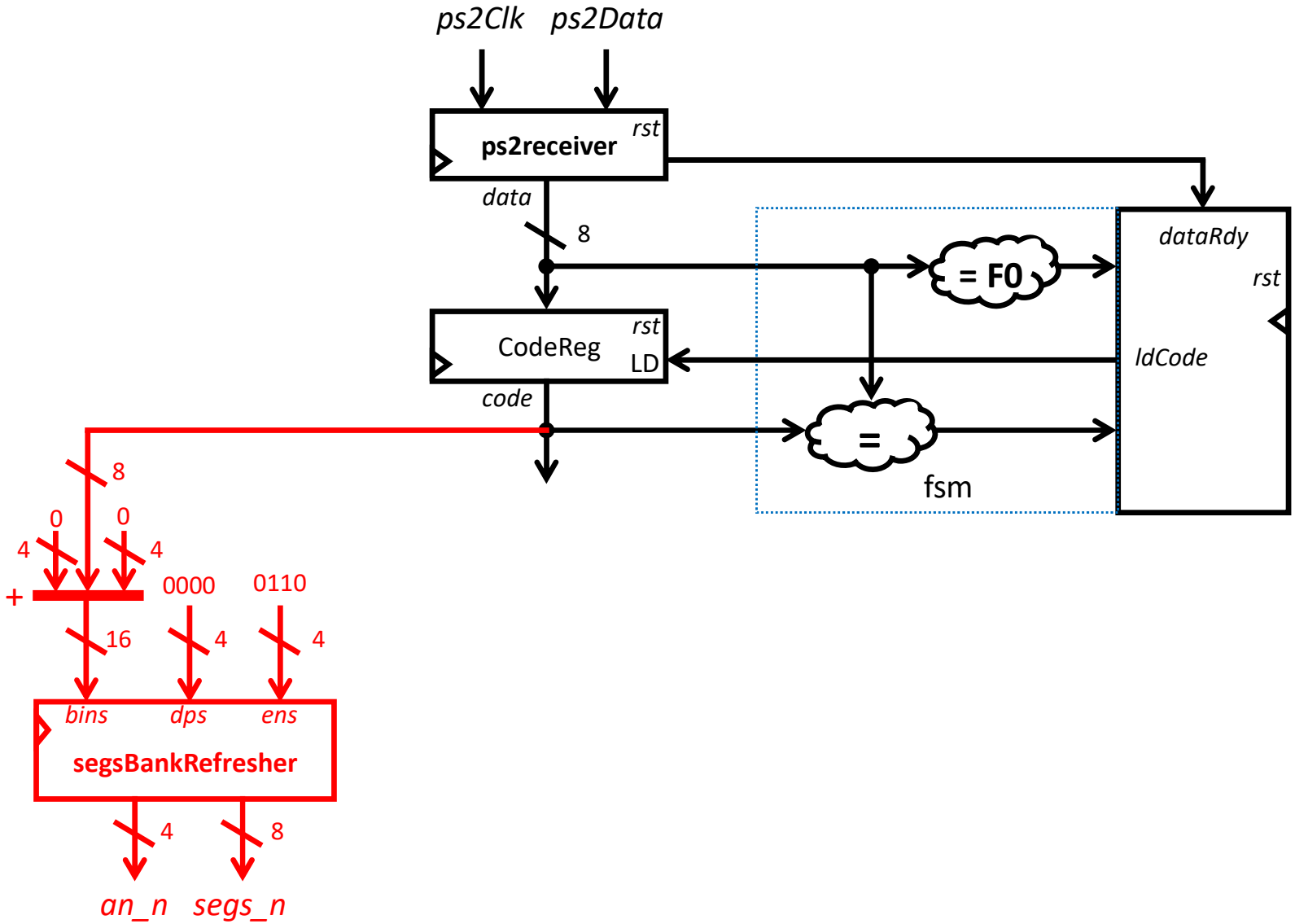
Diseño principal

esquema RTL (i)



Diseño principal

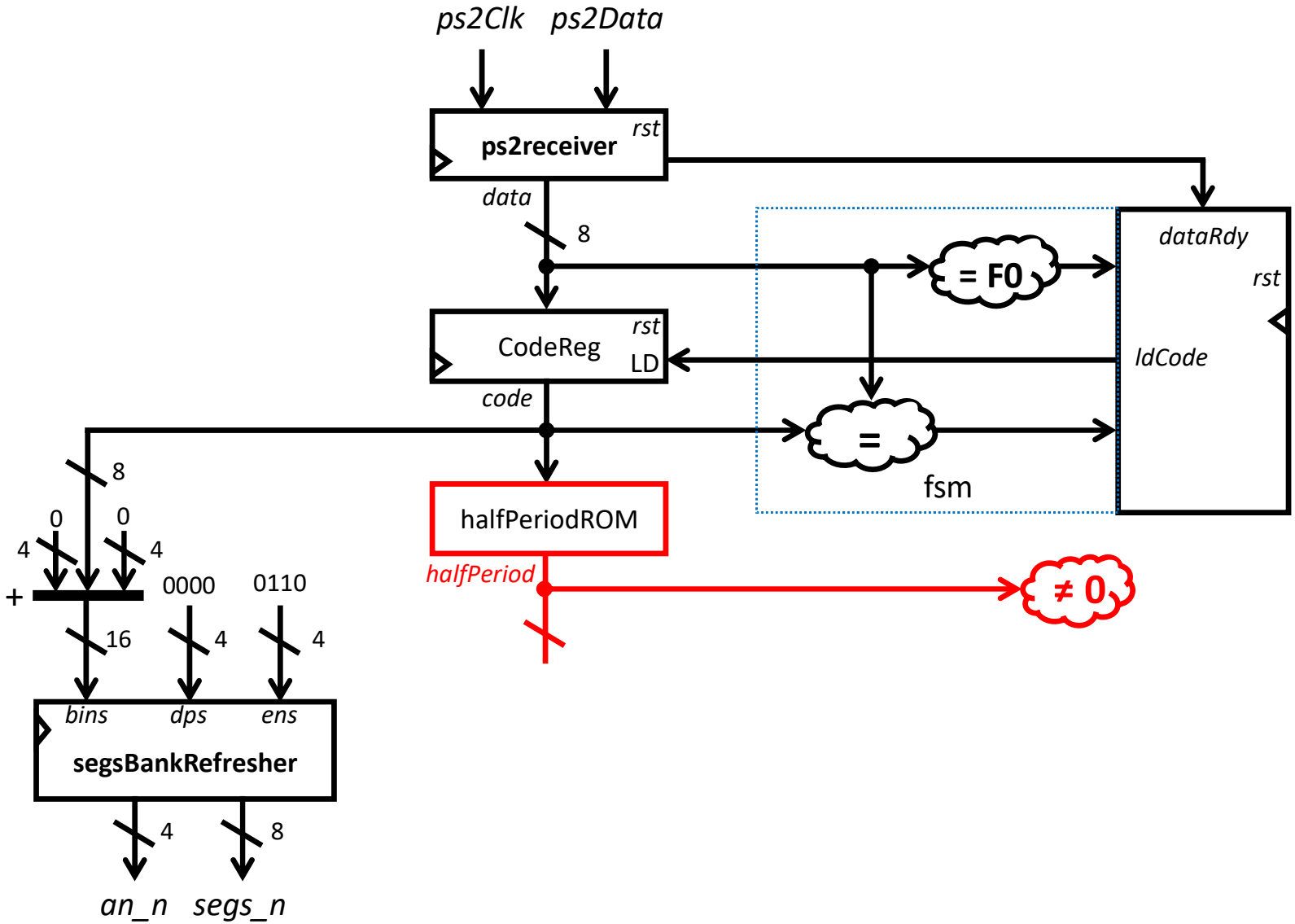
esquema RTL (i)



J.M. Mendías
2024

Diseño principal

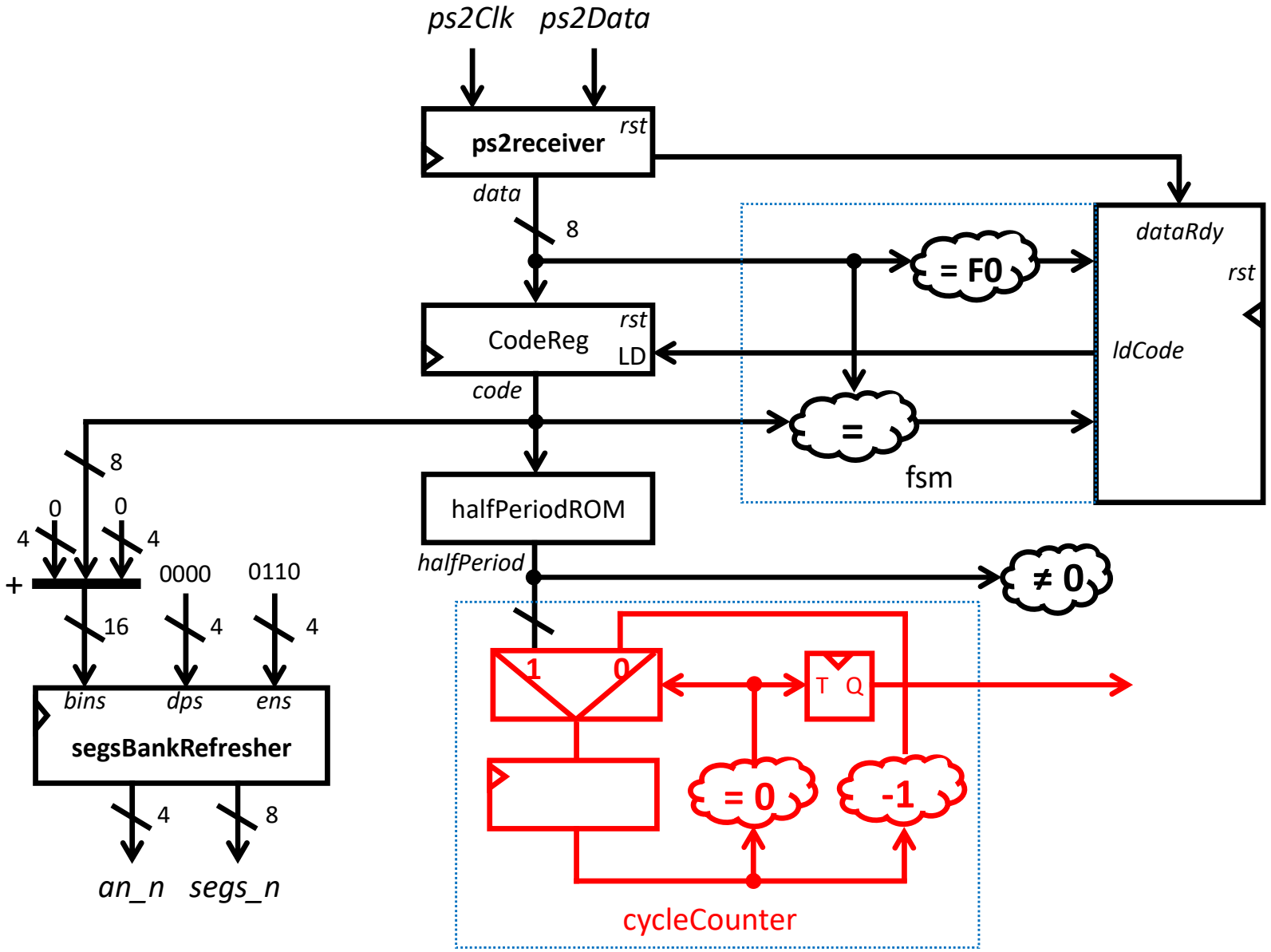
esquema RTL (i)



J.M. Mendías
2024

Diseño principal

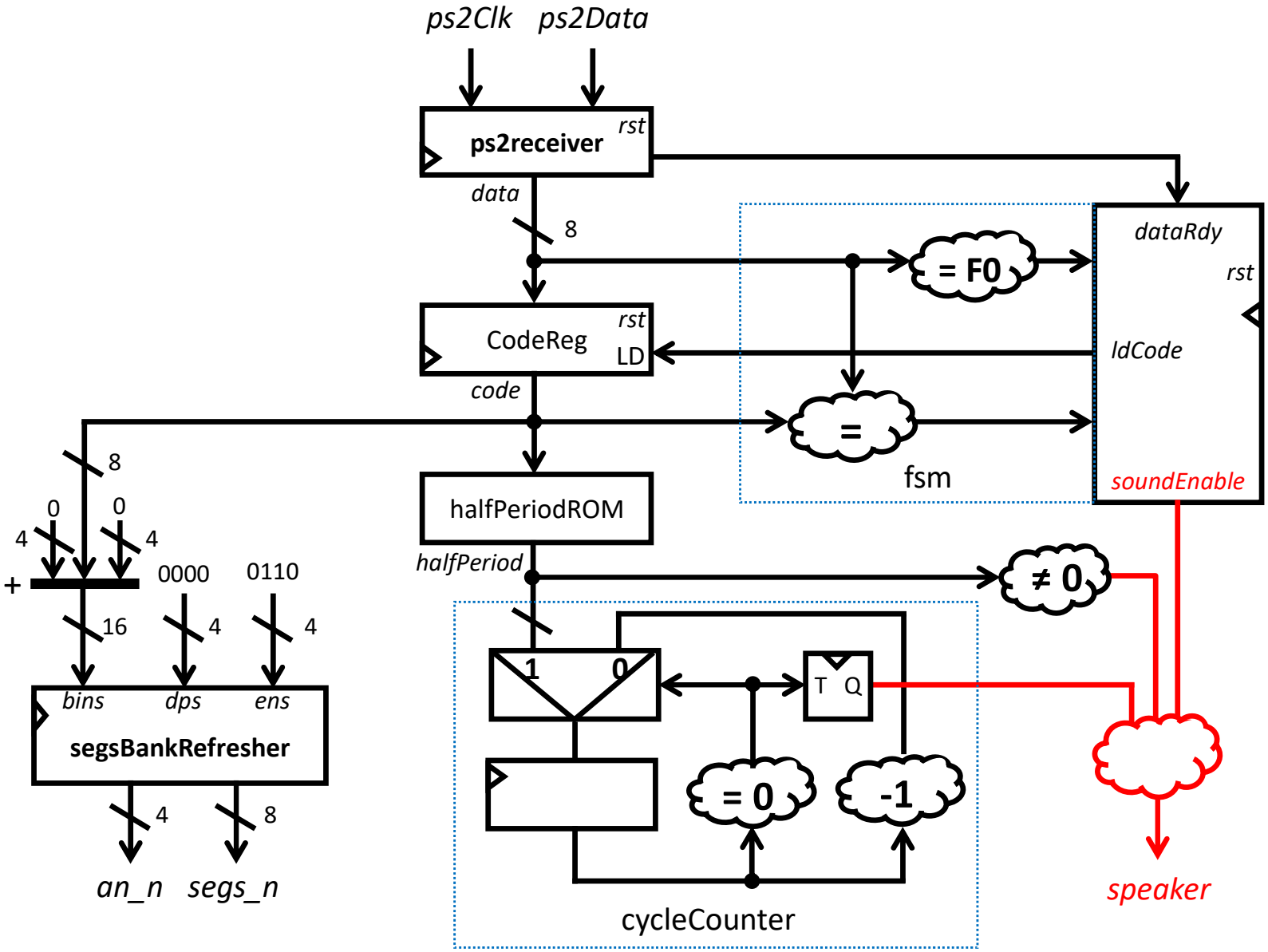
esquema RTL (i)



J.M. Mendías
2024

Diseño principal

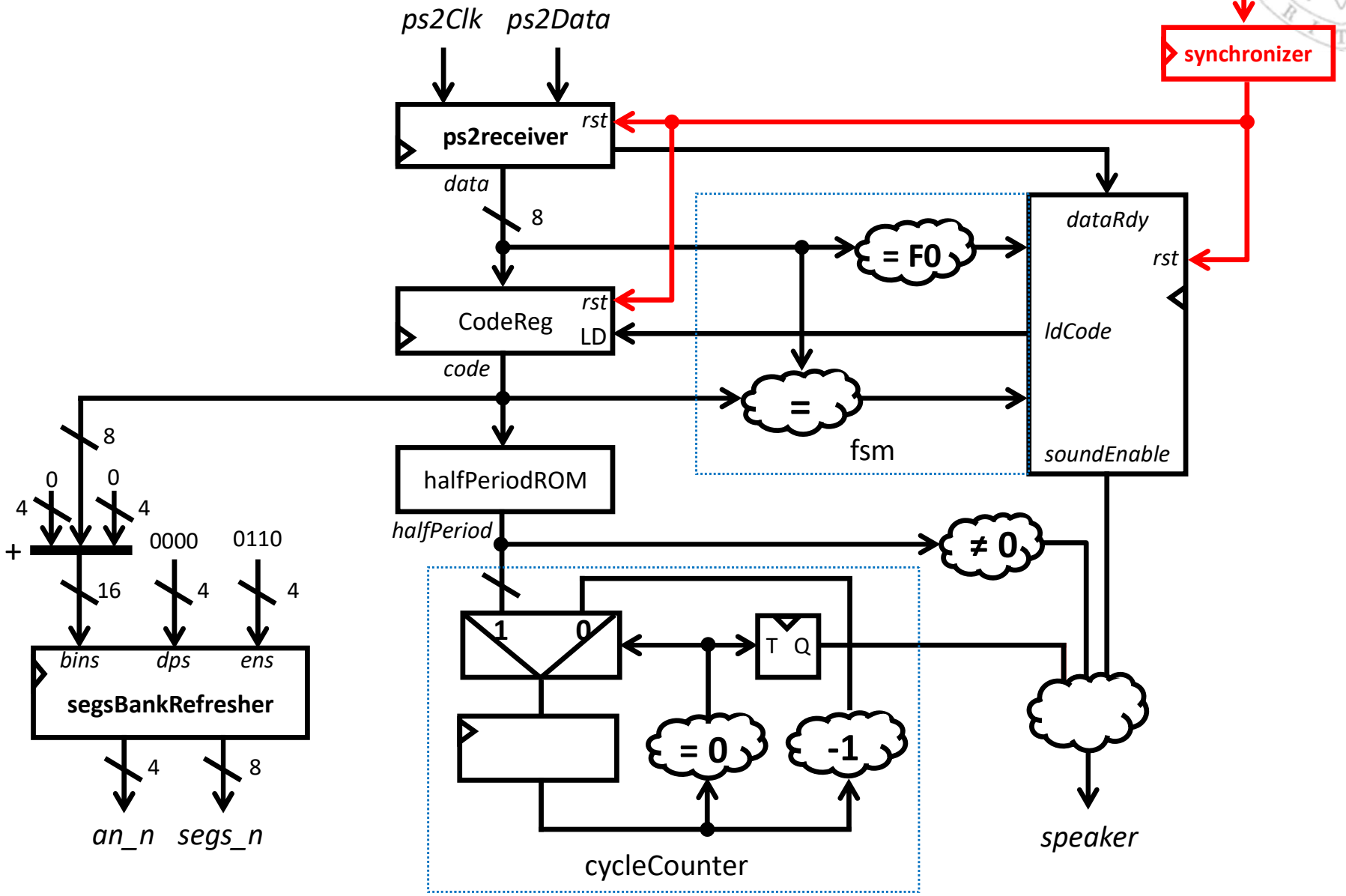
esquema RTL (i)



J.M. Mendías
2024

Diseño principal

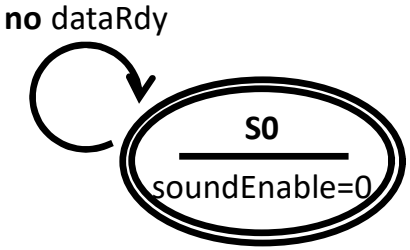
esquema RTL (i)

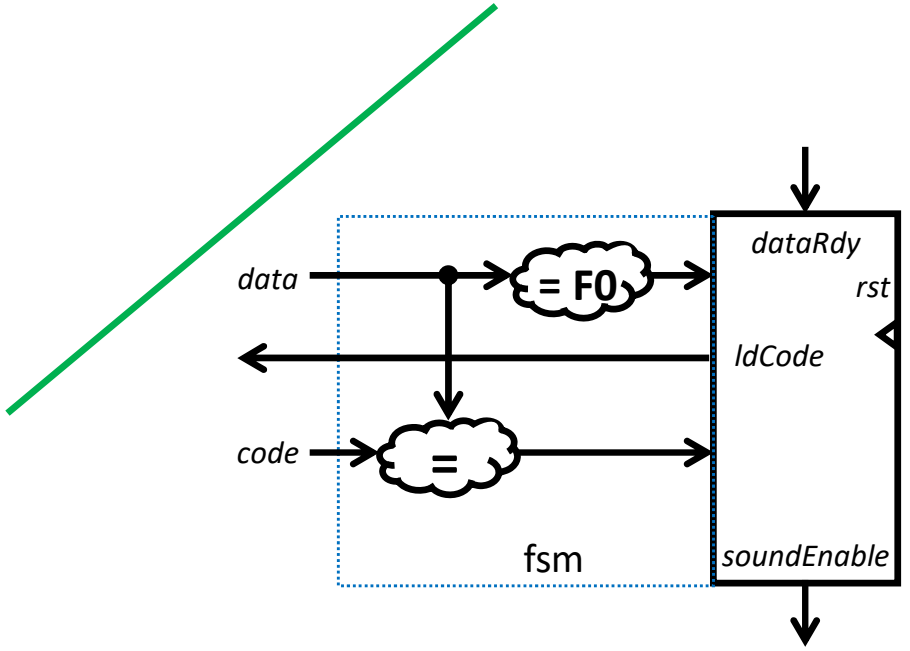


J.M. Mendías
2024

Diseño principal

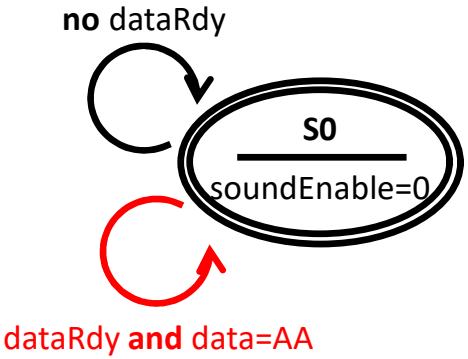
esquema RTL (ii)

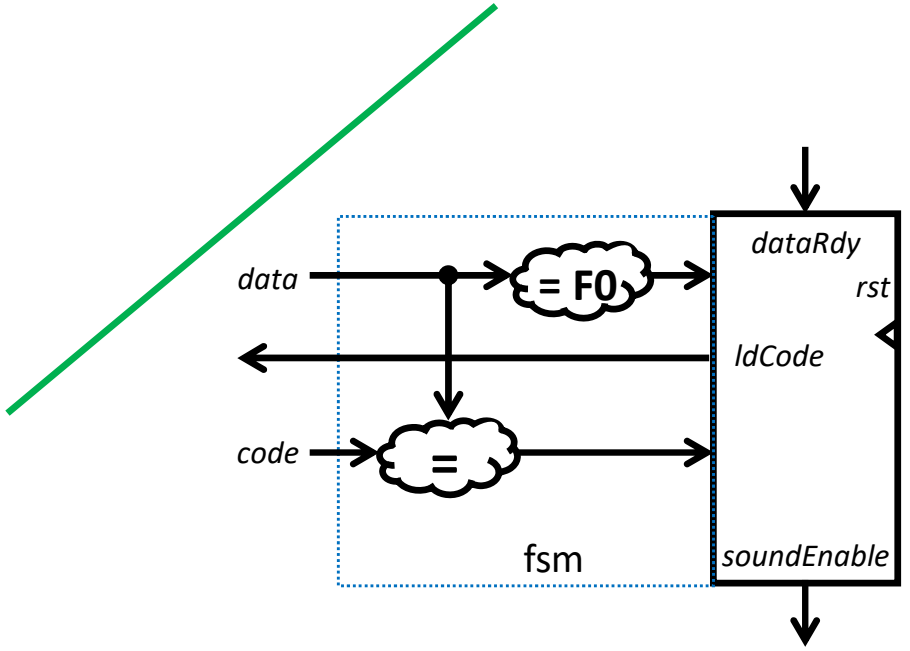




Diseño principal

esquema RTL (ii)

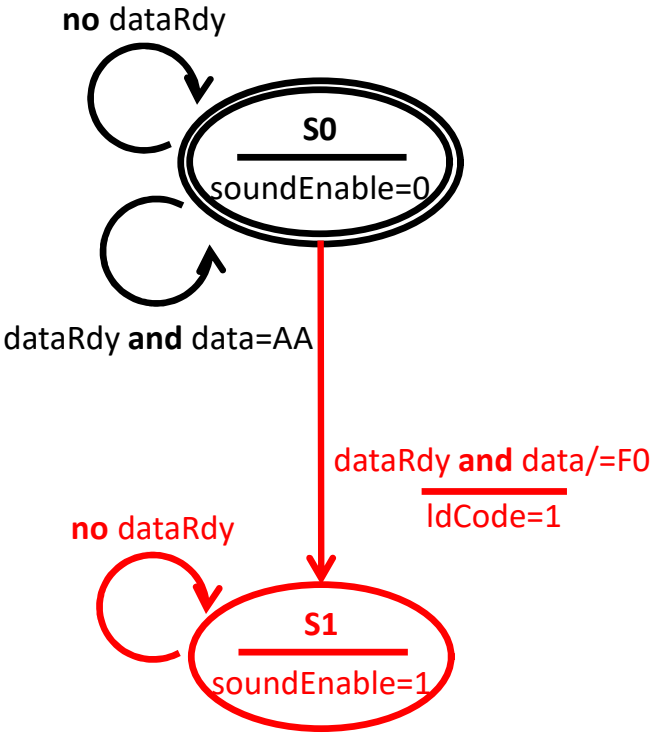


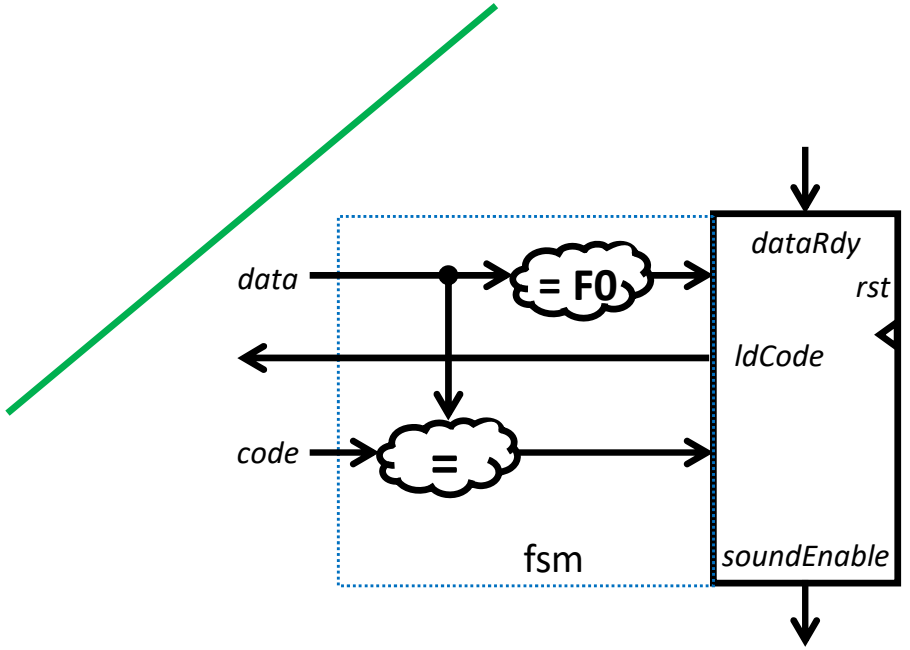


J.M. Mendías
2024

Diseño principal

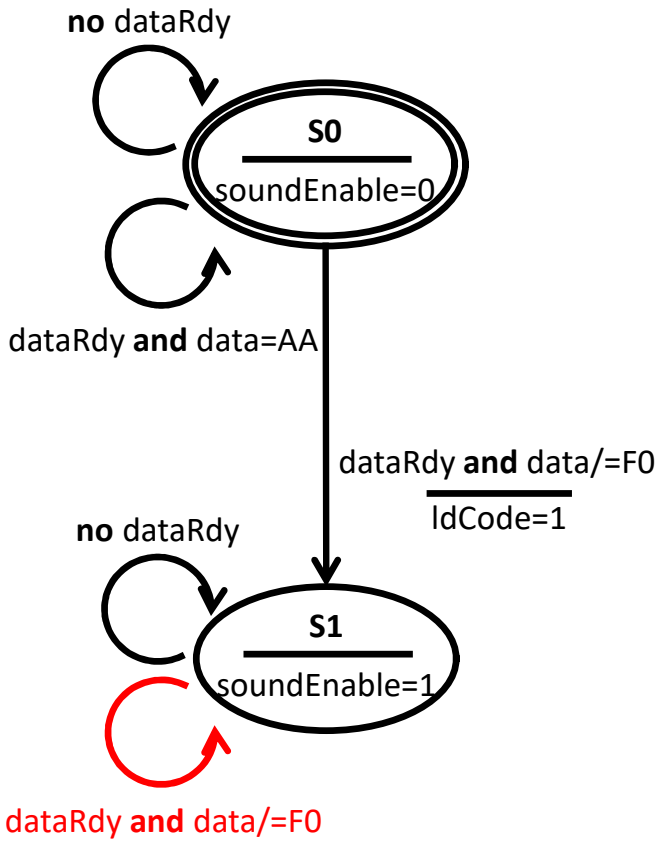
esquema RTL (ii)

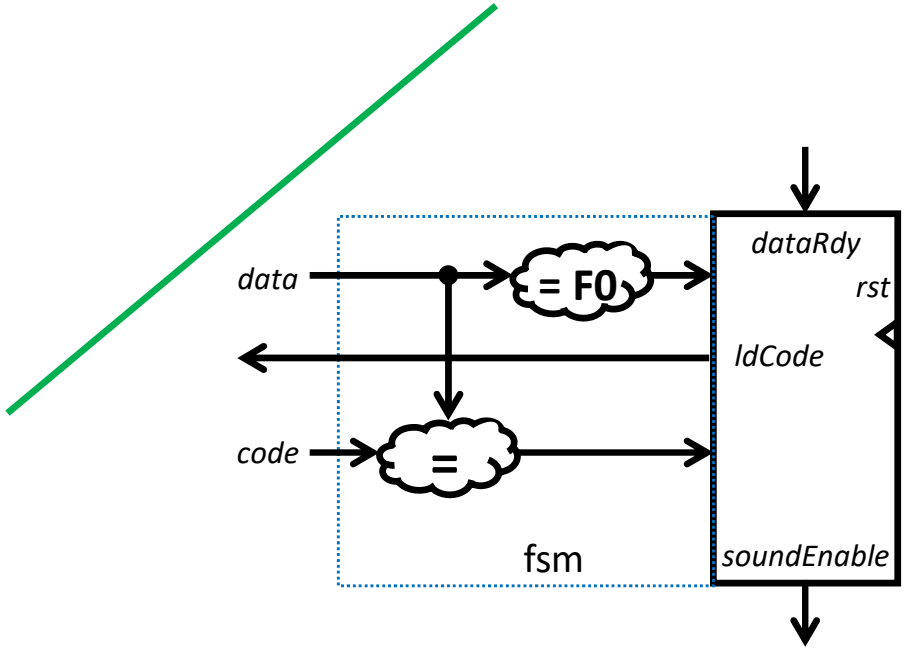




Diseño principal

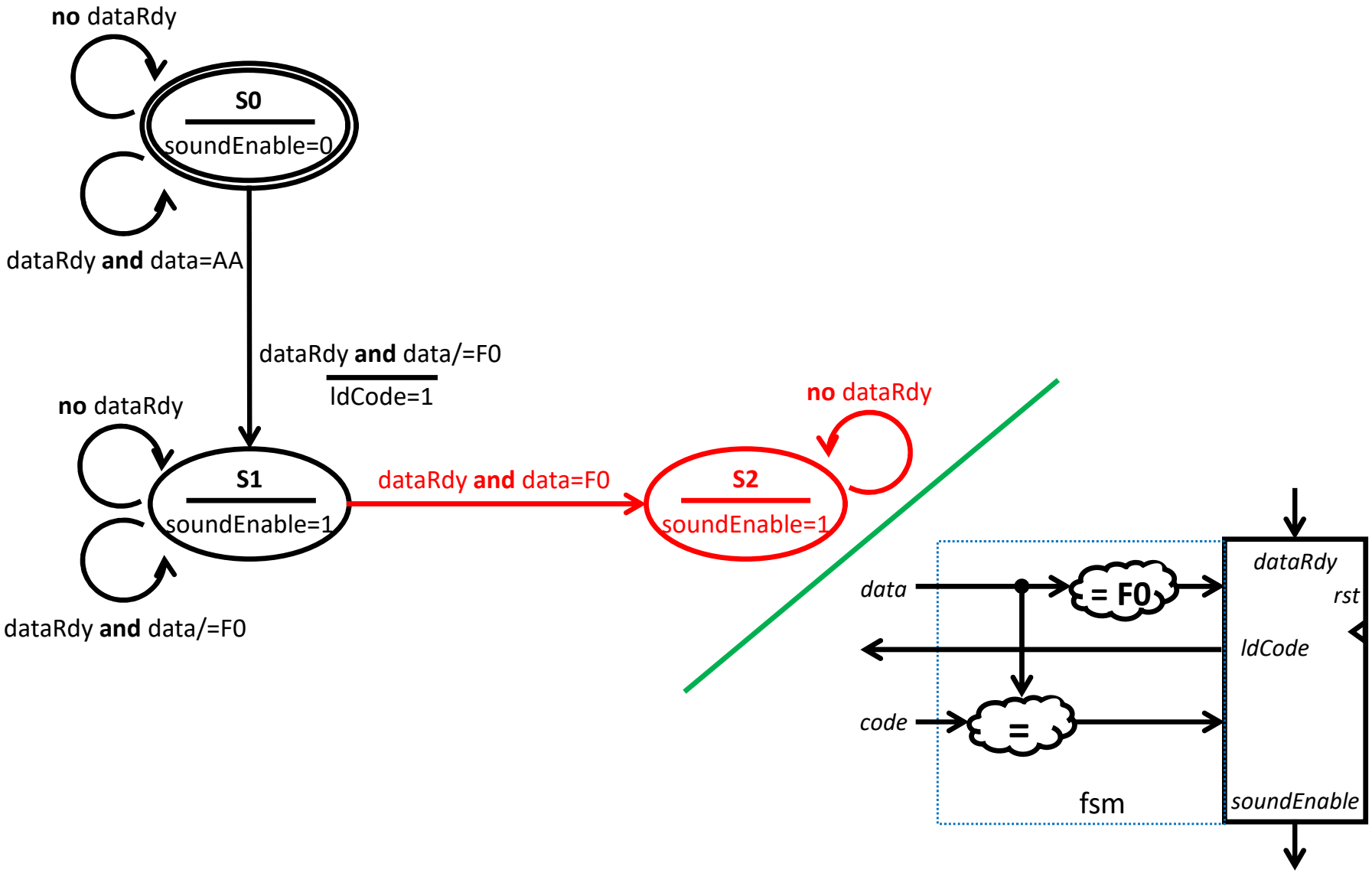
esquema RTL (ii)





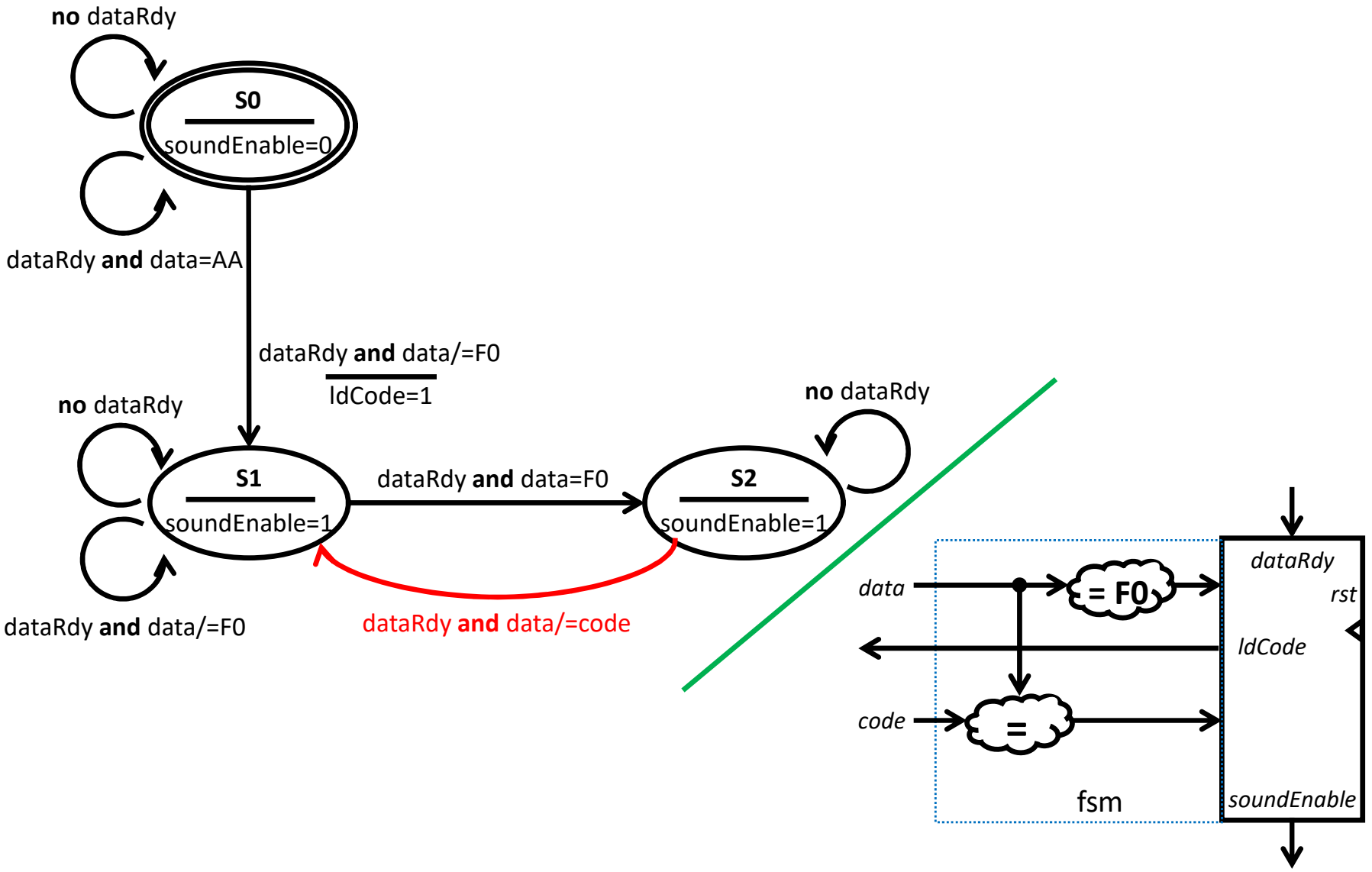
Diseño principal

esquema RTL (ii)



Diseño principal

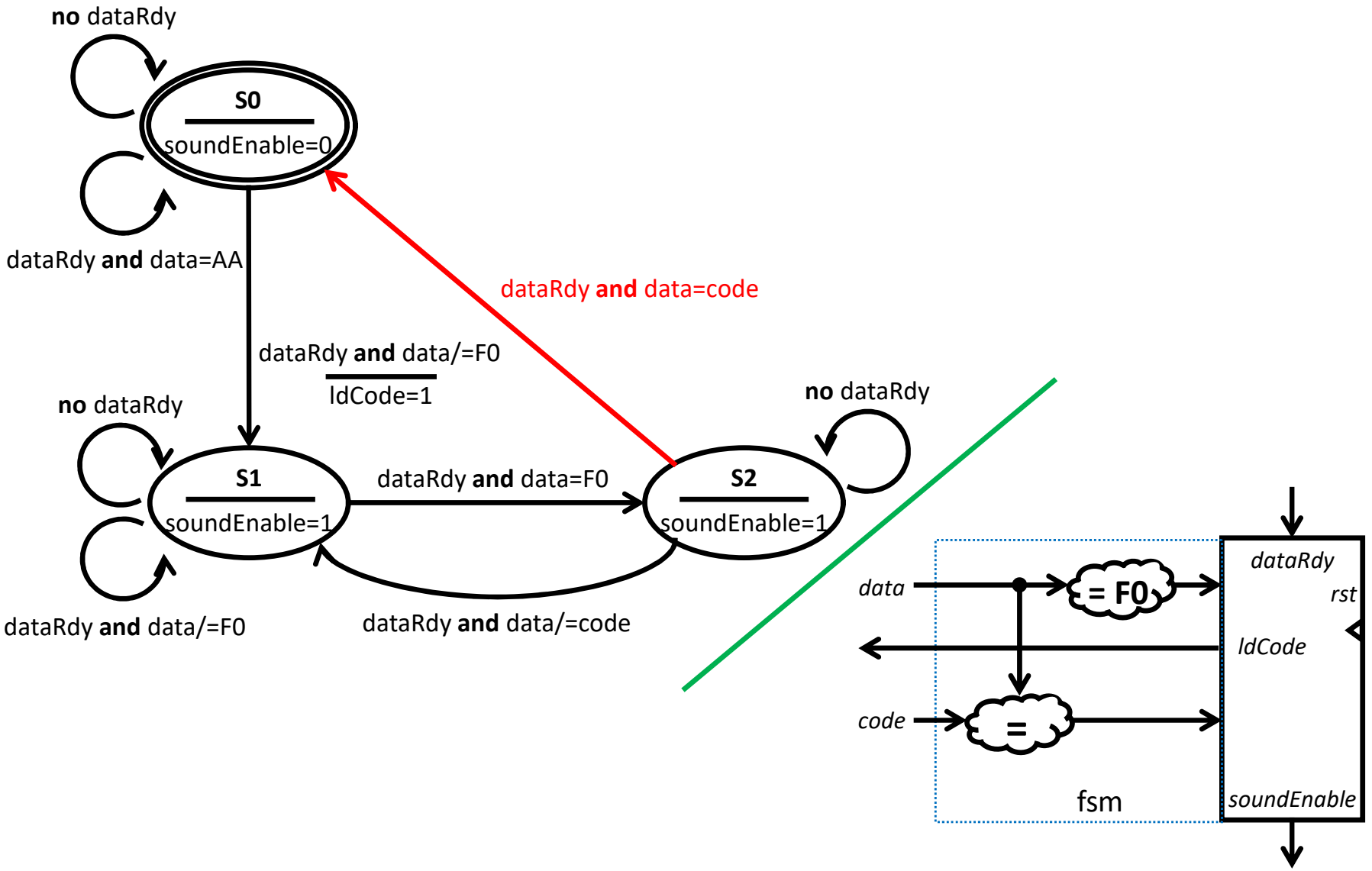
esquema RTL (ii)



J.M. Mendías
2024

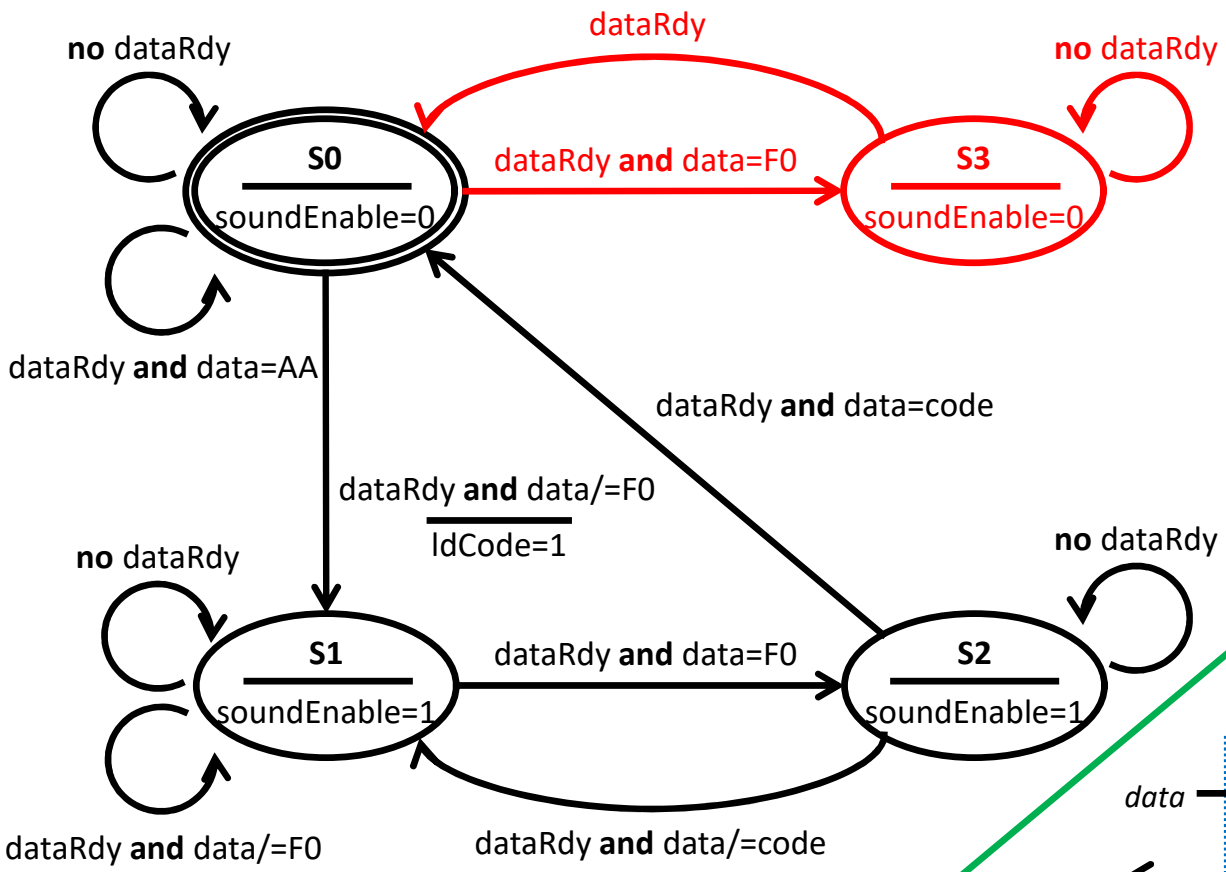
Diseño principal

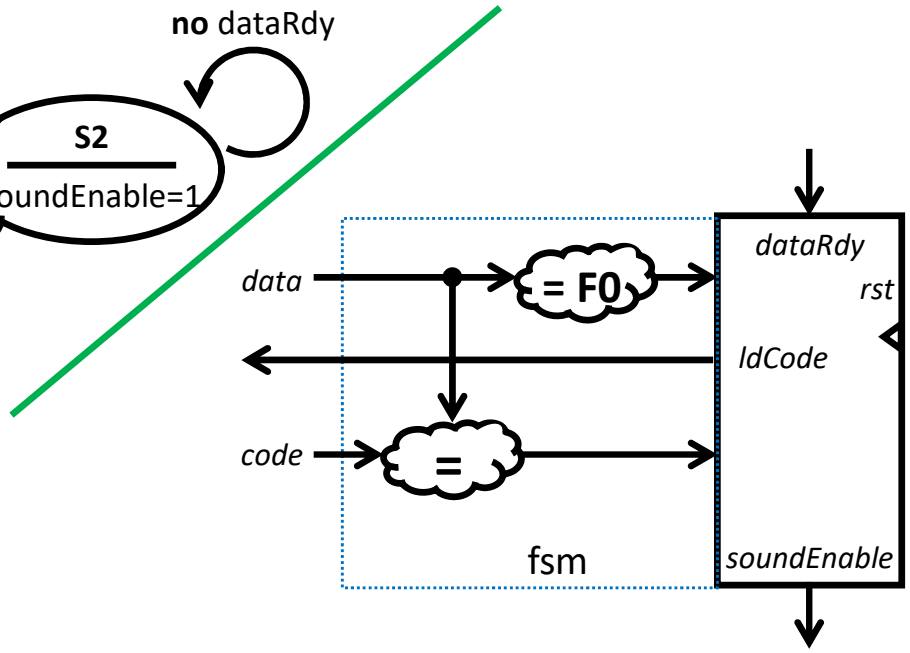
esquema RTL (ii)



Diseño principal

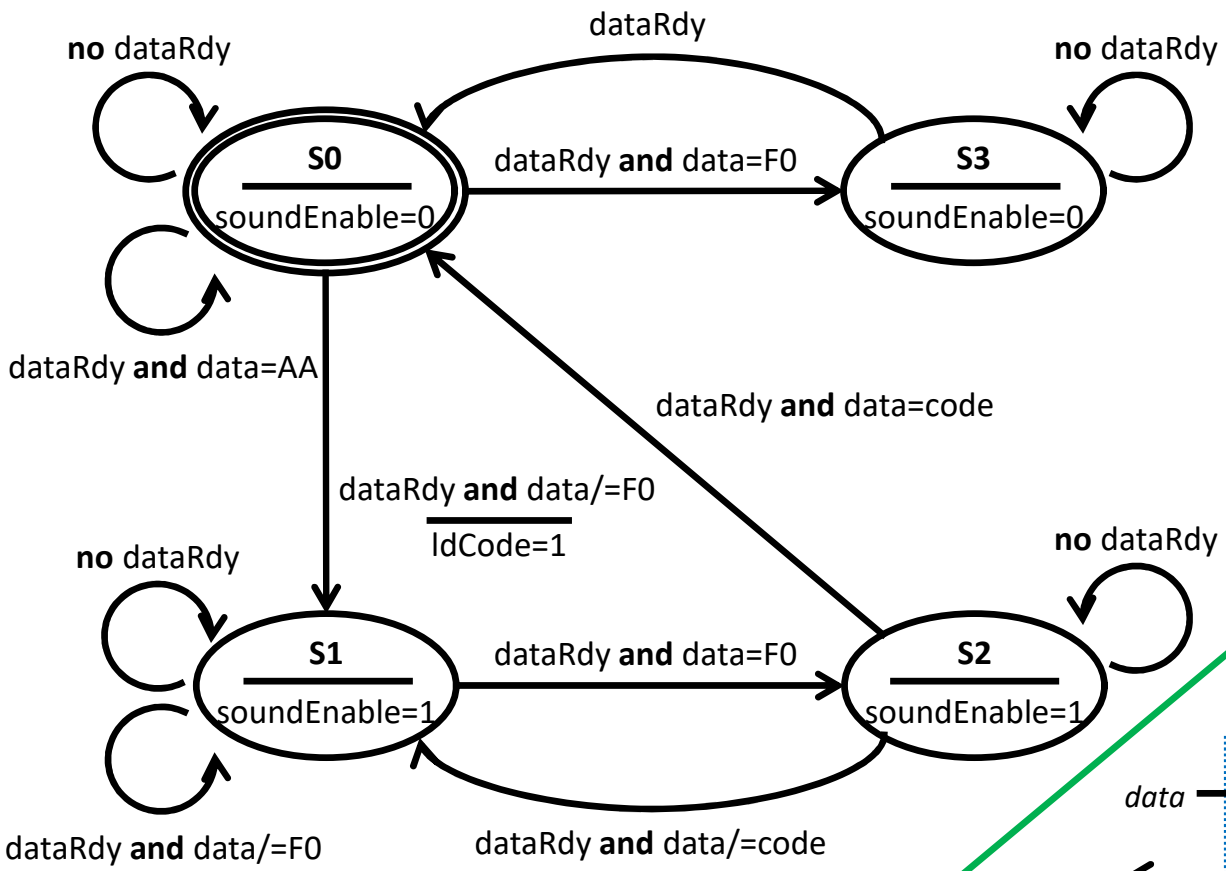
esquema RTL (ii)

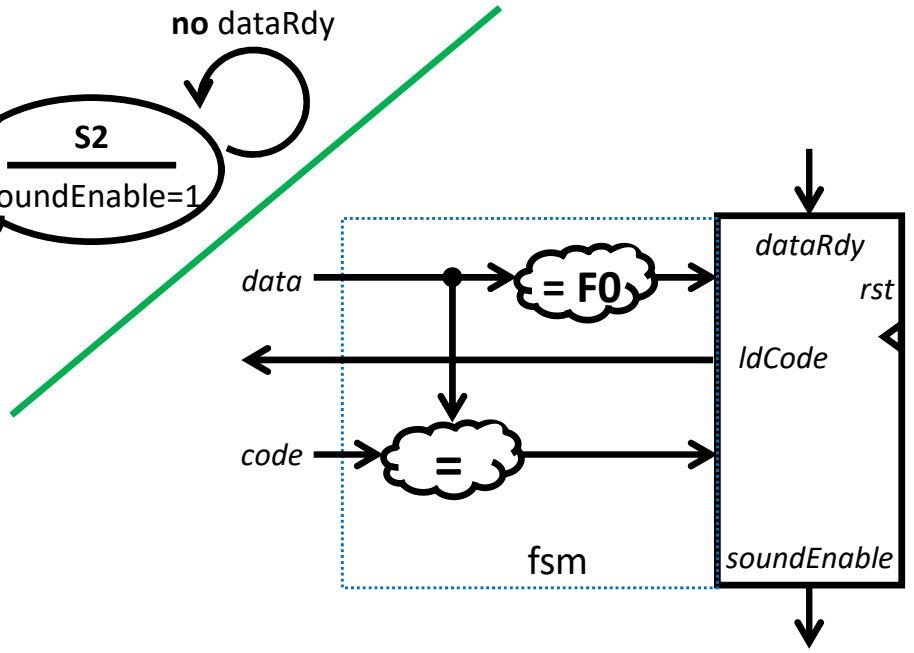




Diseño principal

esquema RTL (ii)





J.M. Mendías 2024

Diseño principal

lab4.vhd

```
architecture syn of lab4 is
  constant FREQ_KHZ : natural := 100_000;        -- frecuencia de operación en KHz
  constant FREQ_HZ : natural := FREQ_KHZ*1000;   -- frecuencia de operación en Hz

  signal code : std_logic_vector(7 downto 0) := (others => '0');
  signal speakerTFF : std_logic := '0';

  signal rstSync     : std_logic;
  signal dataRdy     : std_logic;
  signal ldCode      : std_logic;
  signal halfPeriod  : natural;
  signal data        : std_logic_vector(7 downto 0);
  signal soundEnable : std_logic;
begin
  rstSynchronizer : synchronizer
    ...;
  ps2KeyboardInterface : ps2receiver
    ...;
  displayInterface : segsBankRefresher
    ...;
  codeRegister :
  process (clk)
  begin
    if rising_edge(clk) then
      ...
    end if;
  end process;
```

Registros (con valor inicial)

Conexiones (sin valor inicial)

al ser un rst síncrono se trata como cualquier otra señal externa de datos y se usa un sincronizador convencional

Diseño principal

lab4.vhd

al ser una expresión computable se precalcula para no incluir lógica aritmética

```
  halfPeriodROM :
  with code select
      halfPeriod <=
        FREQ_HZ/(2*262) when X"1c",   -- A = Do
            ...         when X"1d",   -- W = Do#
            ...
         0 when others;

  cycleCounter :
  process (clk)
    variable count : natural;
  begin
    if rising_edge(clk) then
      ...
    end if;
  end process;

  fsm:
  process (clk, dataRdy, data, code)
    type states is (S0, S1, S2, S3);
    variable state: states;
  begin
    ...
  end process;

  speaker <= speakerTFF when ... else ...;
end syn;
```

Instrumentación

- Vivado permite analizar en tiempo real el comportamiento temporal de cualquier señal/es de un diseño volcado sobre una FPGA.
  - Incorporando al diseño un analizador lógico de muestreo síncrono:
    - Captura el valor de las señales seleccionadas en los flancos del reloj del sistema.

- Así, añade de manera transparente y no intrusiva a la netlist diseñada:
  - Sondas que leen los valores de las señales a cada flanco de reloj.
  - Memorias que almacena los valores leídos en una serie de ciclos consecutivos.
  - Lógica de disparo para indicar la condición que debe cumplirse para comenzar la captura de muestras, captura que finalizará cuando las memorias se llenen.
  - Lógica de volcado de las memorias a interfaz un gráfico de Vivado.

- El circuito una vez volcado podrá funcionar normalmente:
  - La captura de datos y su visualización podrán hacerse en paralelo.

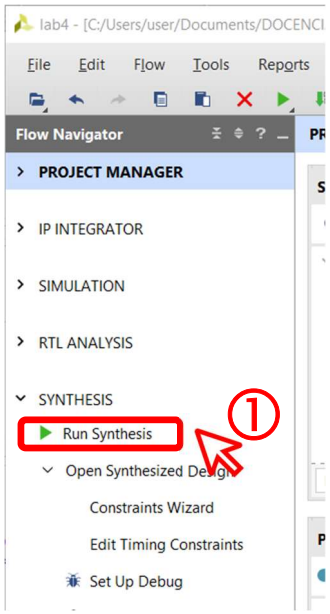
J.M. Mendías
2024

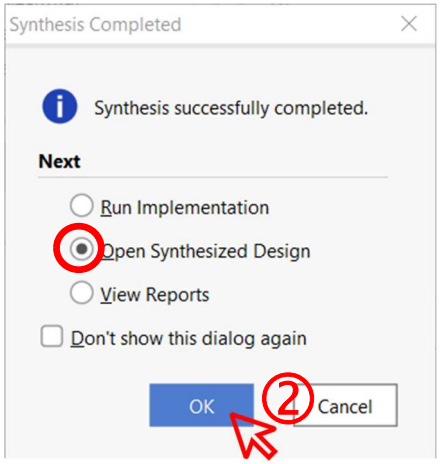
Instrumentación

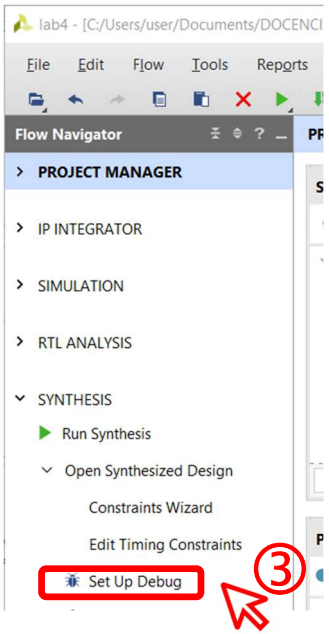
selección de señales

```
architecture syn of lab4 is
  ...
  attribute mark_debug : string;
  attribute mark_debug of ps2Clk  : signal is "true";
  attribute mark_debug of ps2Data : signal is "true";
  attribute mark_debug of dataRdy : signal is "true";
  attribute mark_debug of data    : signal is "true";
begin
  ...
end syn;
```

Indica las señales a muestrear

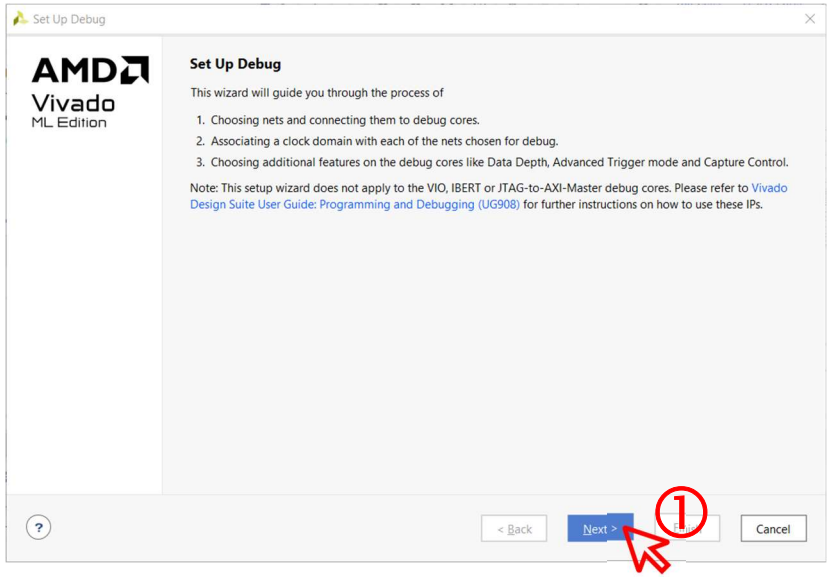


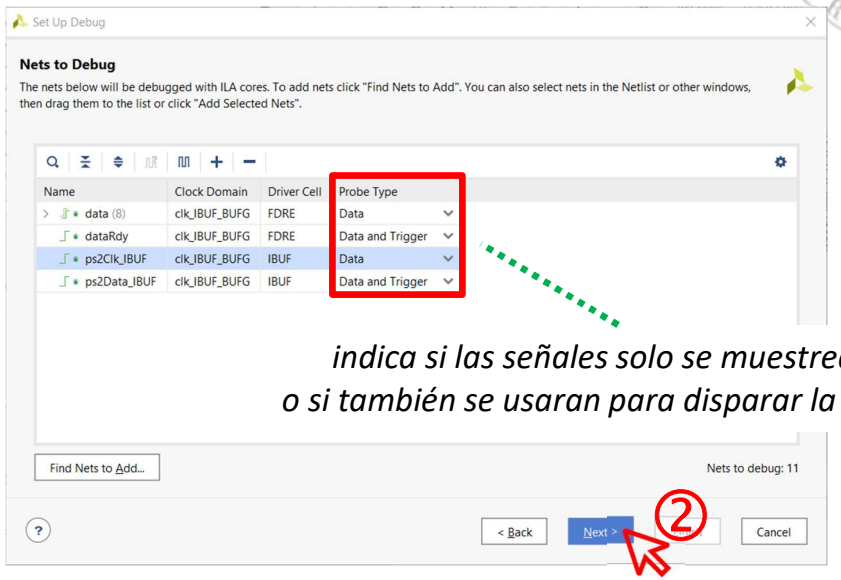




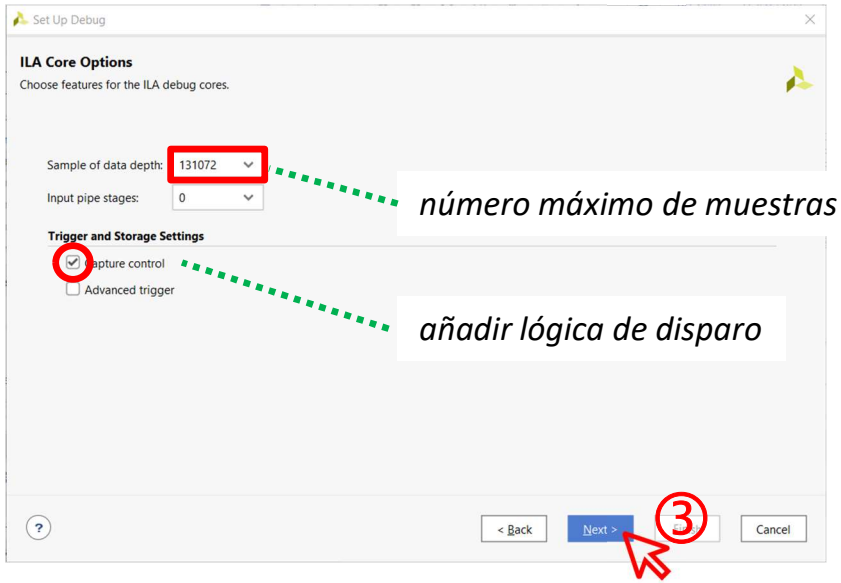
Instrumentación

características del muestreo



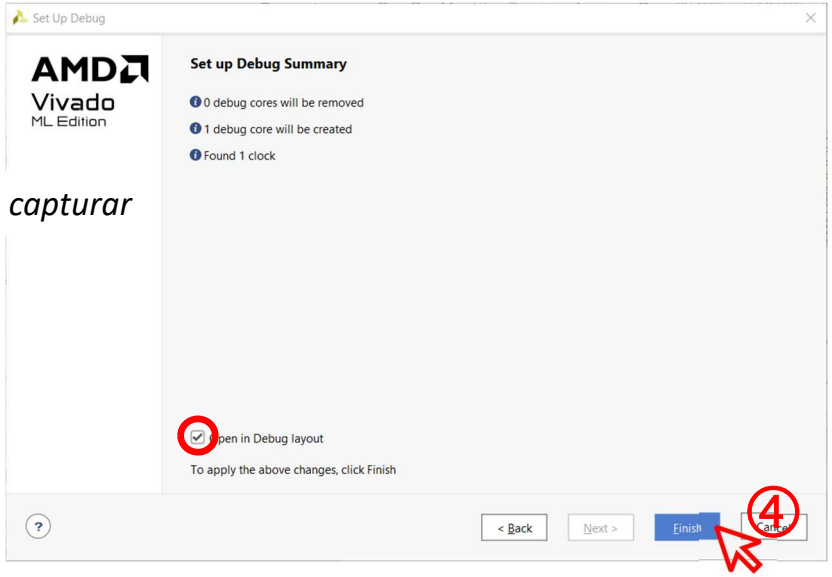


indica si las señales solo se muestrearan
o si también se usaran para disparar la captura



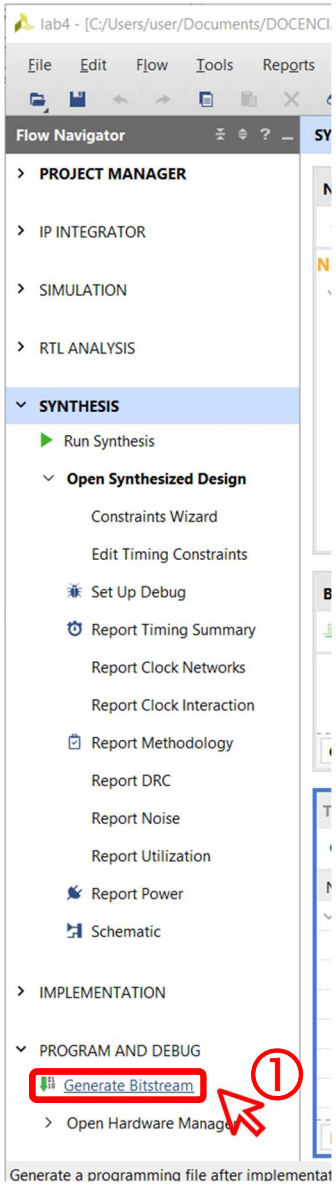
número máximo de muestras a capturar

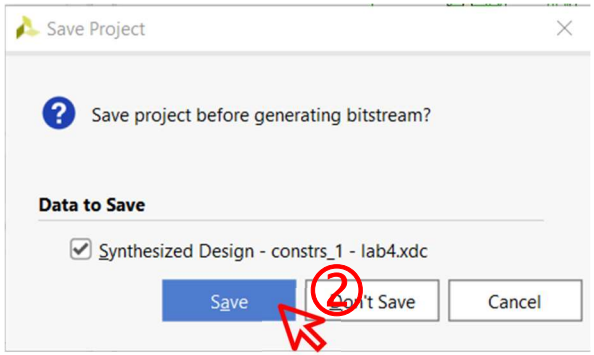
añadir lógica de disparo

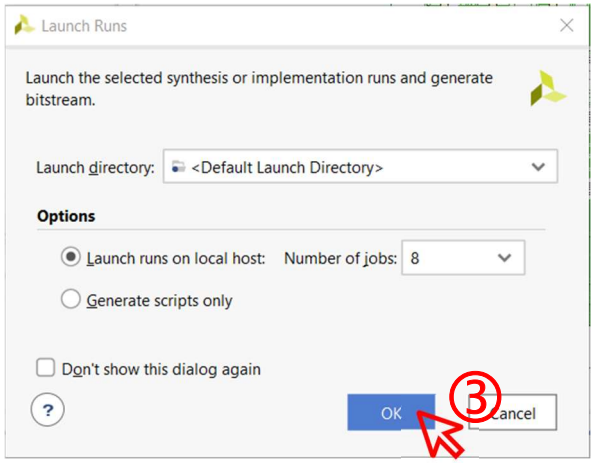


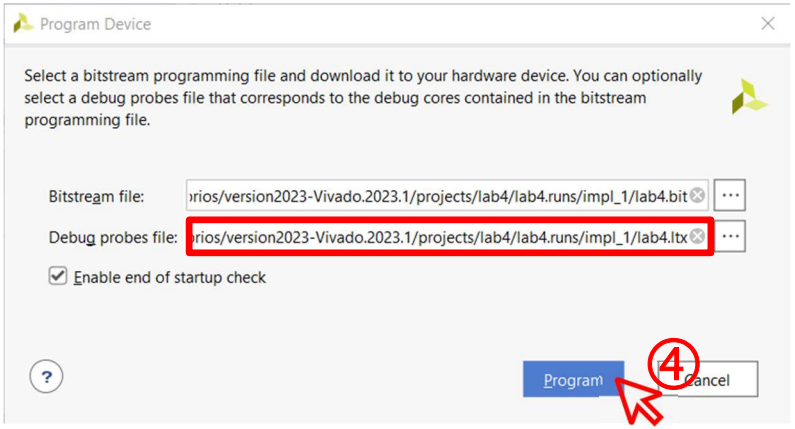
Instrumentación

generación y volcado del bitstream





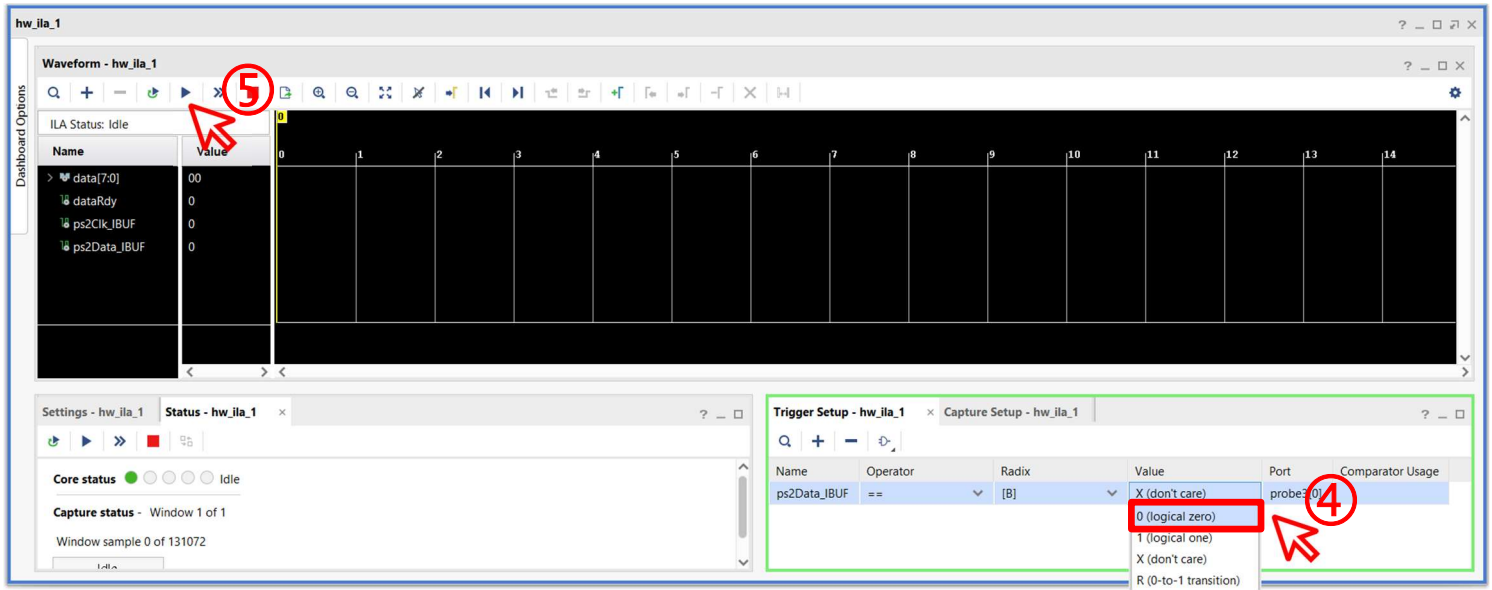




J.M. Mendías
2024

Instrumentación

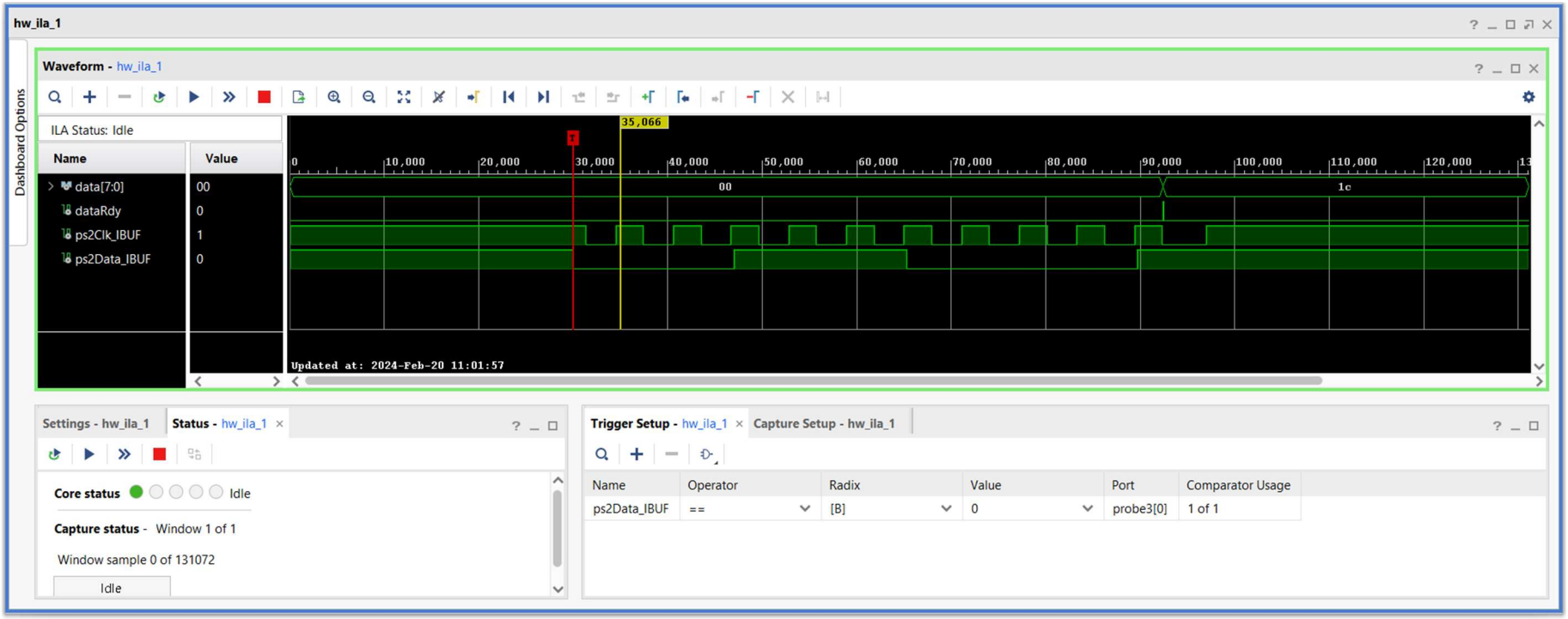
condición de disparo



Instrumentación

análisis

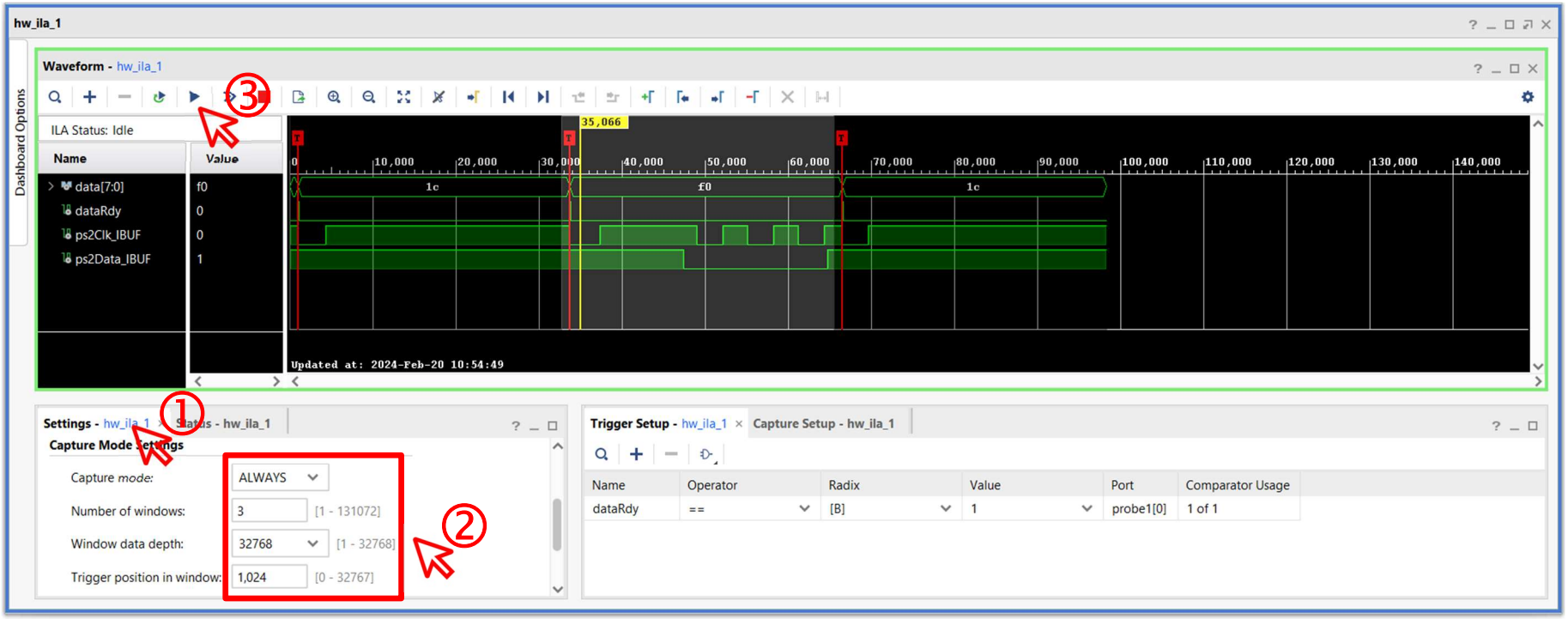
- Tras conectar el teclado, resetear el circuito y pulsar una tecla:
  - Se visualizará el cronograma capturado.



Instrumentación

disparo múltiple

- Pueden visualizarse también tras varios disparos consecutivos
  - Solo se visualizan ventanas de muestras tras el disparo



J.M. Mendías
2024

Tareas

1. En la carpeta `DAS/source` crear las carpeta `lab4`.
2. Descargar de la Web los ficheros en:
   - `common:` `ps2receiver.vhd`
   - `lab4:` `ps2receiverTest.vhd`, `lab4.vhd` y `lab4.xdc`
3. Completar `common.vhd` con la declaración del nuevo componente reusable.
4. Completar el código omitido en `ps2receiver.vhd` y `lab4.vhd`
5. En la carpeta `DAS/projects` crear el proyecto `lab4`.
6. Incluir en el proyecto (sin copiarlos) los siguientes fuentes:
   - `common.vhd`, `bin2segs.vhd`, `segsBankRefresher.vhd`, `synchronizer.vhd`, `edgeDetector.vhd`, `ps2receiver.vhd`, `ps2receiverTest.vhd`, `lab4.vhd` y `lab4.xdc`
7. Modificar las propiedades de los siguientes ficheros:
   - solo síntesis: `bin2segs.vhd`, `segsBankRefresher.vhd` y `lab4.vhd`
   - solo simulación: `ps2receiverTest.vhd`

Tareas

8. Realizar una simulación funcional presíntesis (behavioral) durante 600 ms de `ps2receiverTest`

9. Sintetizar, implementar y generar el fichero de configuración.
10. Conectar el teclado y el altavoz a placa y encenderla.
11. Volcar el fichero de configuración `lab4.bit`

12. Instrumentar el diseño.
13. Implementar y generar el fichero de configuración.
14. Volcar el fichero de configuración `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/