Laboratorio 5:

FSM con ruta de datos

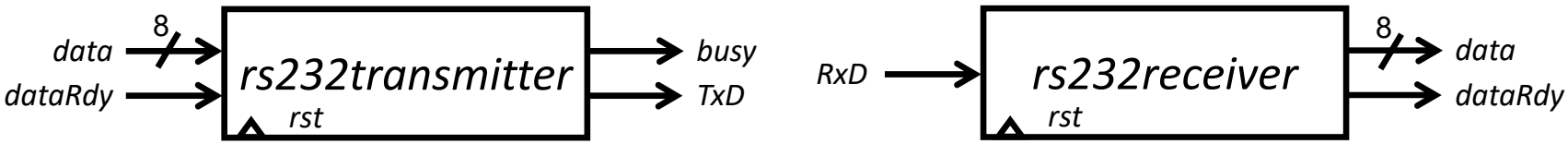
comunicación serie asíncrona por bus RS-232

Diseño automático de sistemas

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

Presentación

- Diseñar un par emisor-receptor RS-232 elemental:
  - Cada dato se transmite sin paridad en tramas de 10 bits con el siguiente formato:
    - 1 bit de start (0), 8 bits de datos (primero LSB), 1 bit de stop (1)
  - No realizará control de flujo.
- El transmisor RS-232:
  - Convertirá de paralelo a serie cada dato individual a transmitir por el bus RS-232.
  - Cada vez que la señal dataRdy se active, leerá data y comenzará a transmitirlo.
  - Durante la transmisión mantendrá activada la señal busy.
- El receptor RS-232:
  - Convertirá de serie a paralelo cada dato individual recibido por el bus RS-232.
  - Cada vez que reciba correctamente un dato, lo volcará en data.
  - Por cada nuevo dato volcado activará durante un ciclo la señal de strobe dataRdy.
- Ambos dispondrán de reset síncrono.



J.M. Mendías
2024

Protocolo RS-232

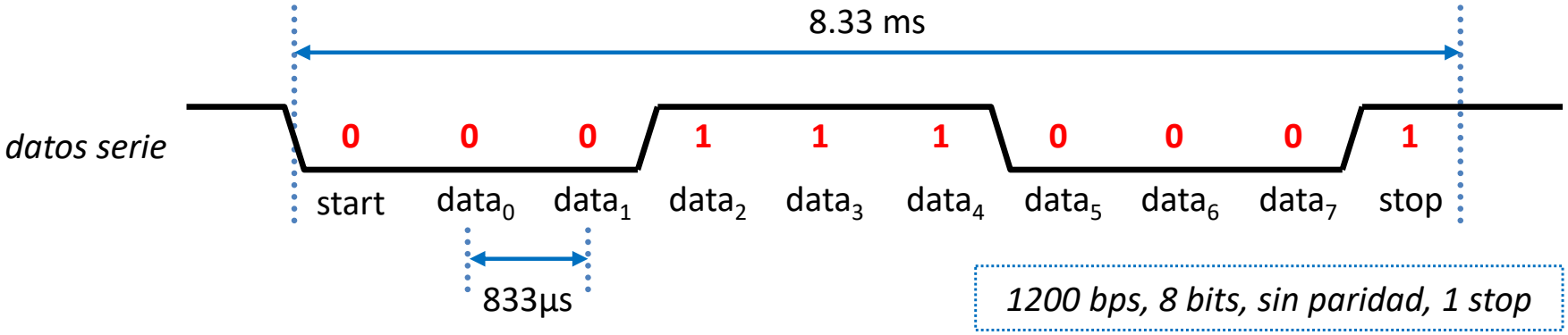
presentación

- El estándar RS-232 completo define un protocolo de comunicación serie que consta de 22 líneas unidireccionales.
  - Para conectar típicamente un computador (DTE) con un módem (DCE)
  - Cuando conecta un DTE con otro DTE o con un DCE distinto de módem, el conjunto de líneas usado es mucho menor.
- Algunos de los subconjuntos de líneas más comunes son los que usan:
  - Únicamente las 2 líneas de datos serie: TxD (salida) y RxD (entrada).
  - Las líneas de datos serie y 2 líneas para control de flujo: RTS (salida) y CTS (entrada).
  - Todas ellas transmiten información asíncronamente.
- La velocidad de transmisión, el formato de la trama y el mecanismo de control de flujo es configurable.
  - Velocidades: 1200, 2400, 4800, 9600, 14400,19200... 115200 baudios
  - Bits de datos: 5, 6, 7 u 8 (primero LSB)
  - Bits de stop: 1, 1.5 o 2
  - Paridad: sin paridad, par, impar...
  - Control de flujo: sin control de flujo, RTS/CTS, XON/XOFF...

Protocolo RS-232

transmisión/recepción

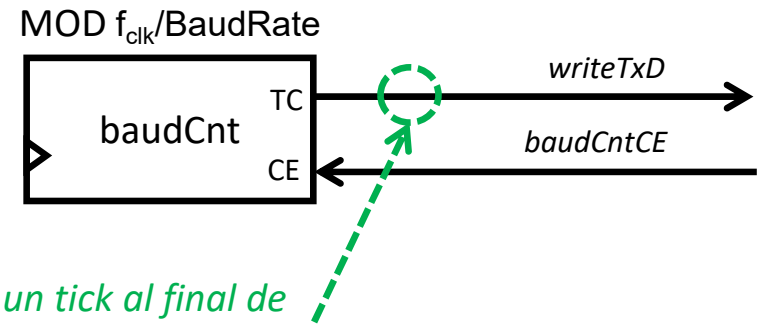
- En ausencia de control de flujo hardware:
  - Cualquiera de los 2 dispositivos conectados puede iniciar una transmisión.
    - Si la comunicación es full-duplex ambos pueden transmitir a la vez.
  - El emisor vuelca los datos serie a la velocidad convenida y en el siguiente orden:
    - bit de start (0), n bits de datos (primero LSB), bit paridad (opcional), bit/s de stop (1)
  - El receptor debe muestrear la línea serie a intervalos regulares dependientes de la velocidad de transmisión para extraer los datos.
- Si existe control de flujo, entonces:
  - El emisor solo puede transmitir si el receptor está preparado para aceptarlos.
    - Si la línea CTS está en BAJA (control hardware).
    - Si el receptor no ha enviado el dato XOFF (control software).



J.M. Mendías
2024

Emisor RS-232

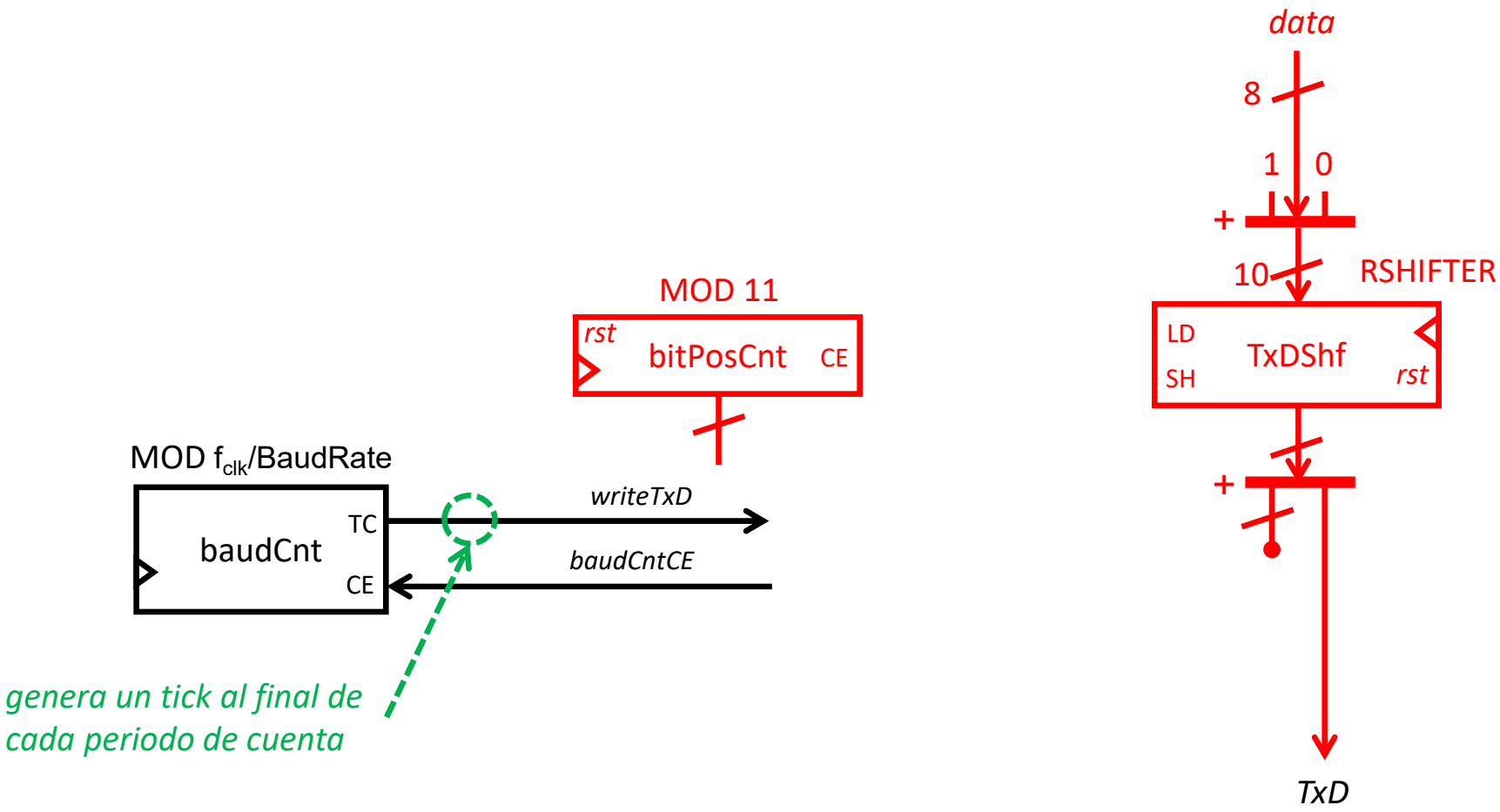
esquema RTL



J.M. Mendias
2024

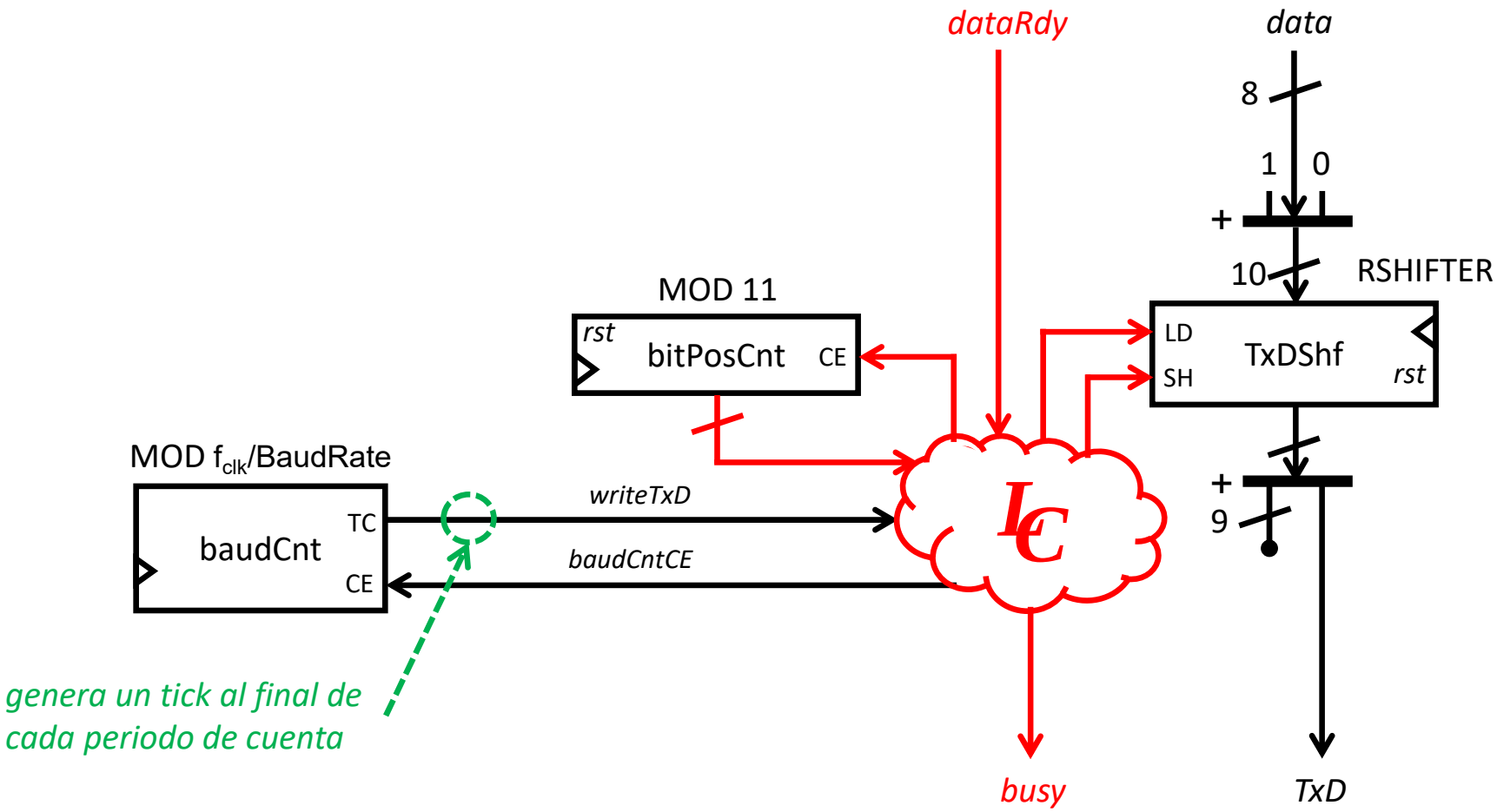
Emisor RS-232

esquema RTL



Emisor RS-232

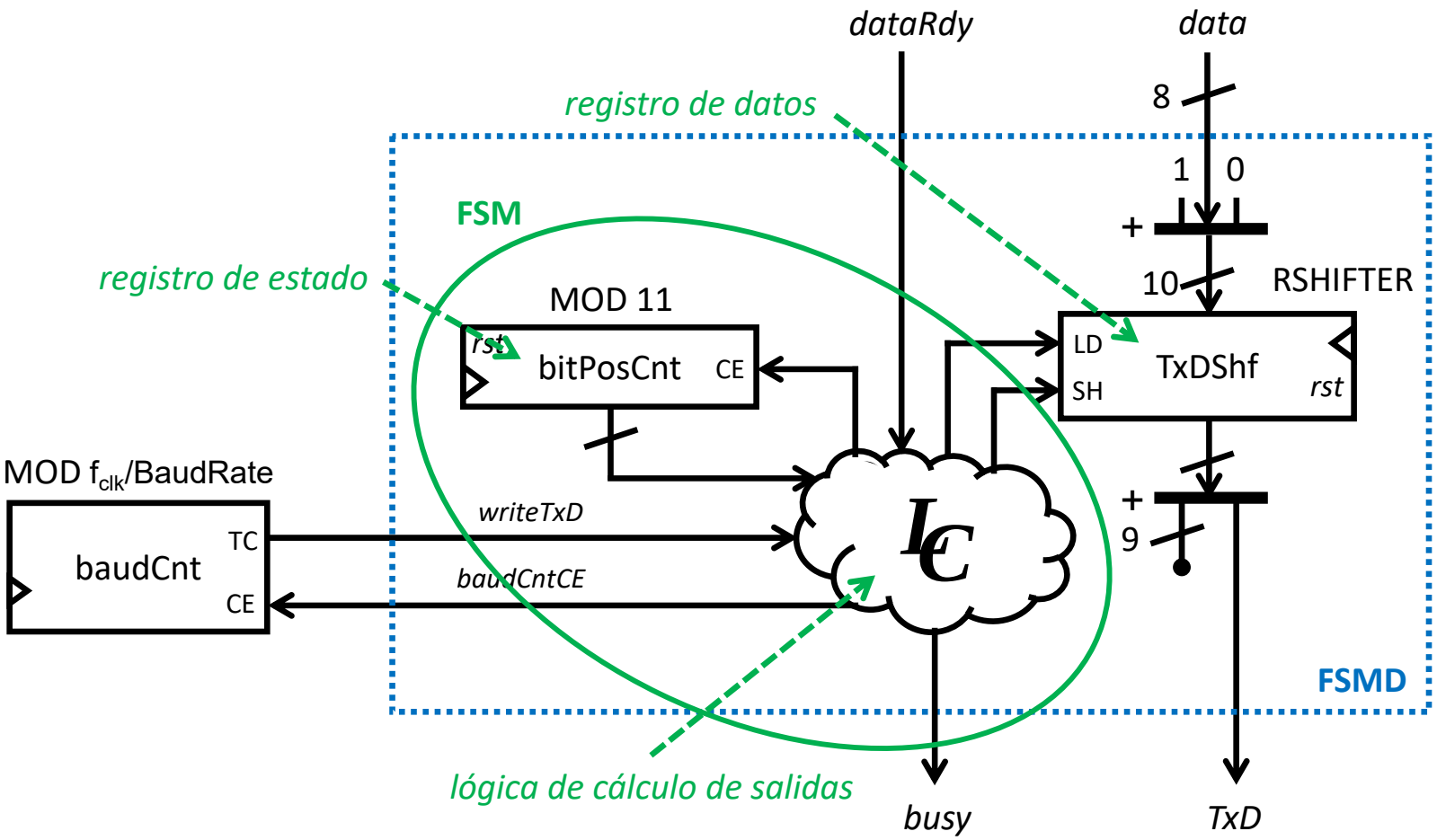
esquema RTL



J.M. Mendías
2024

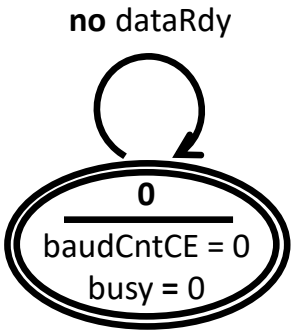
Emisor RS-232

esquema RTL



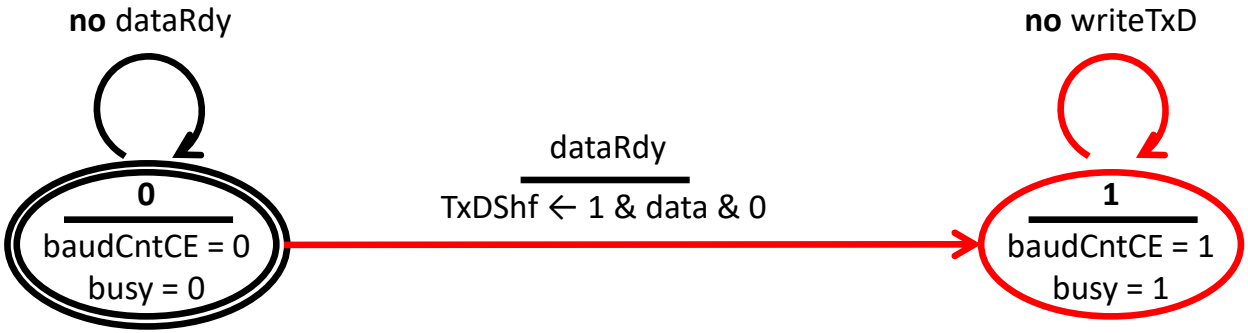
Emisor RS-232

FSMD



Emisor RS-232

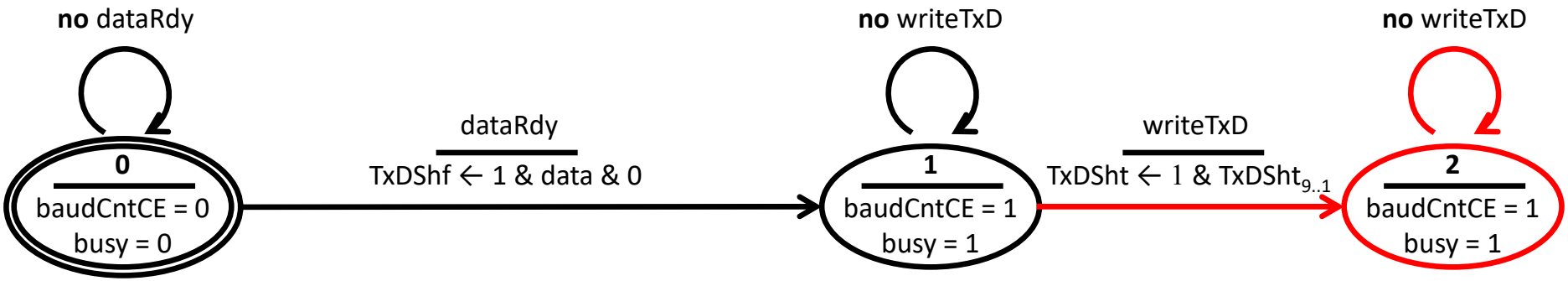
FSMD



J.M. Mendias
2024

Emisor RS-232

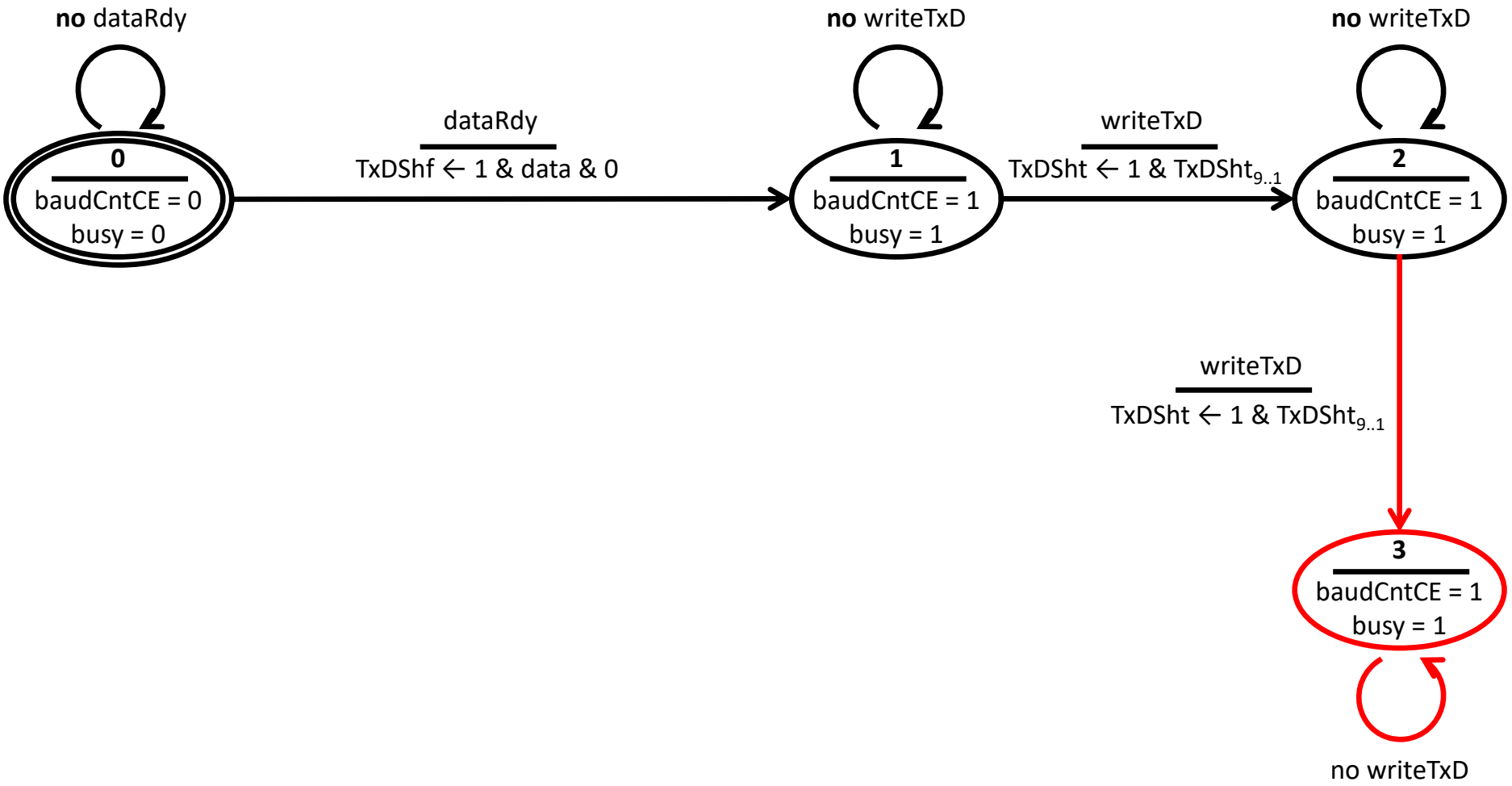
FSMD



J.M. Mendias
2024

Emisor RS-232

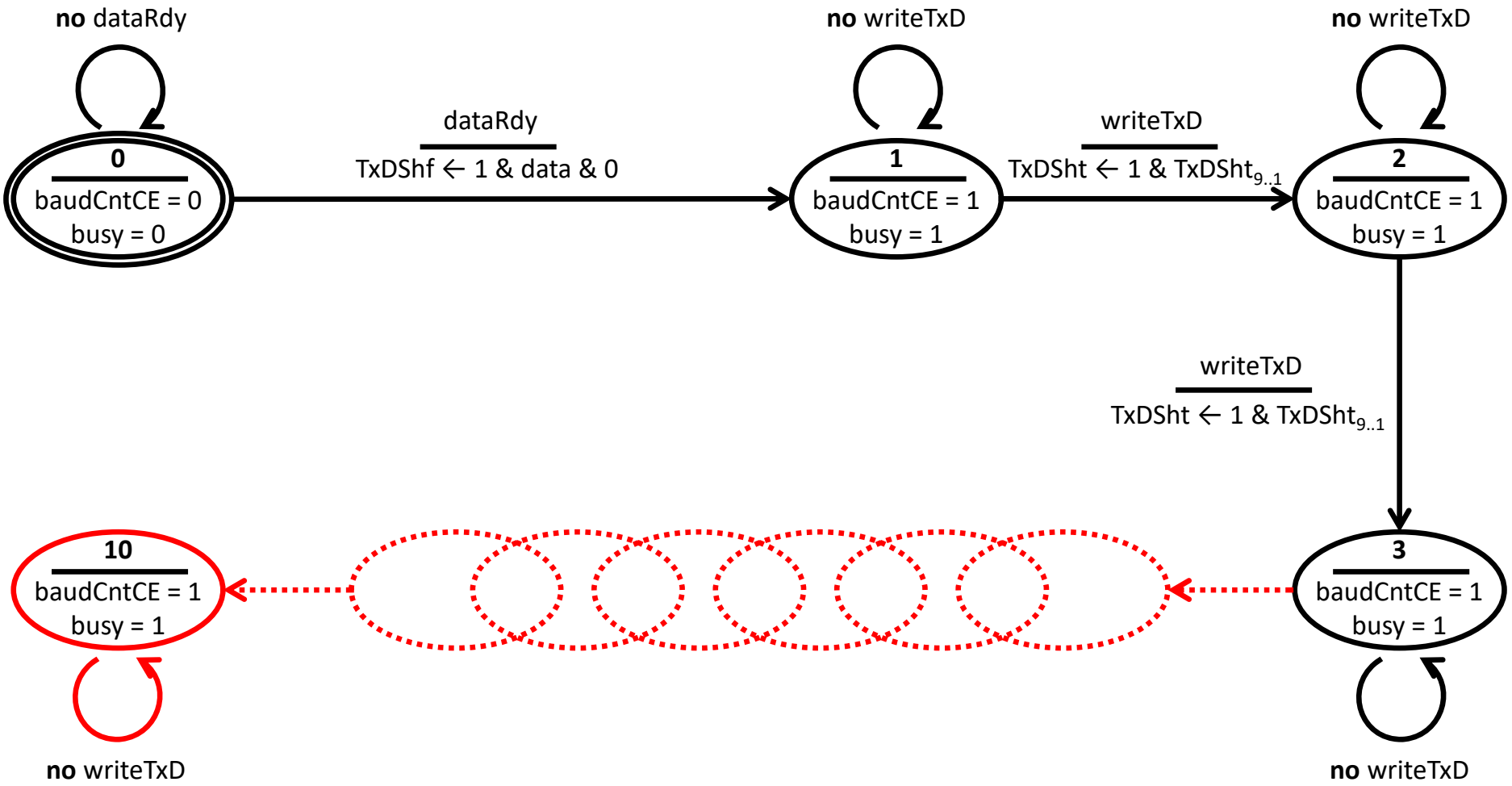
FSMD



J.M. Mendias
2024

Emisor RS-232

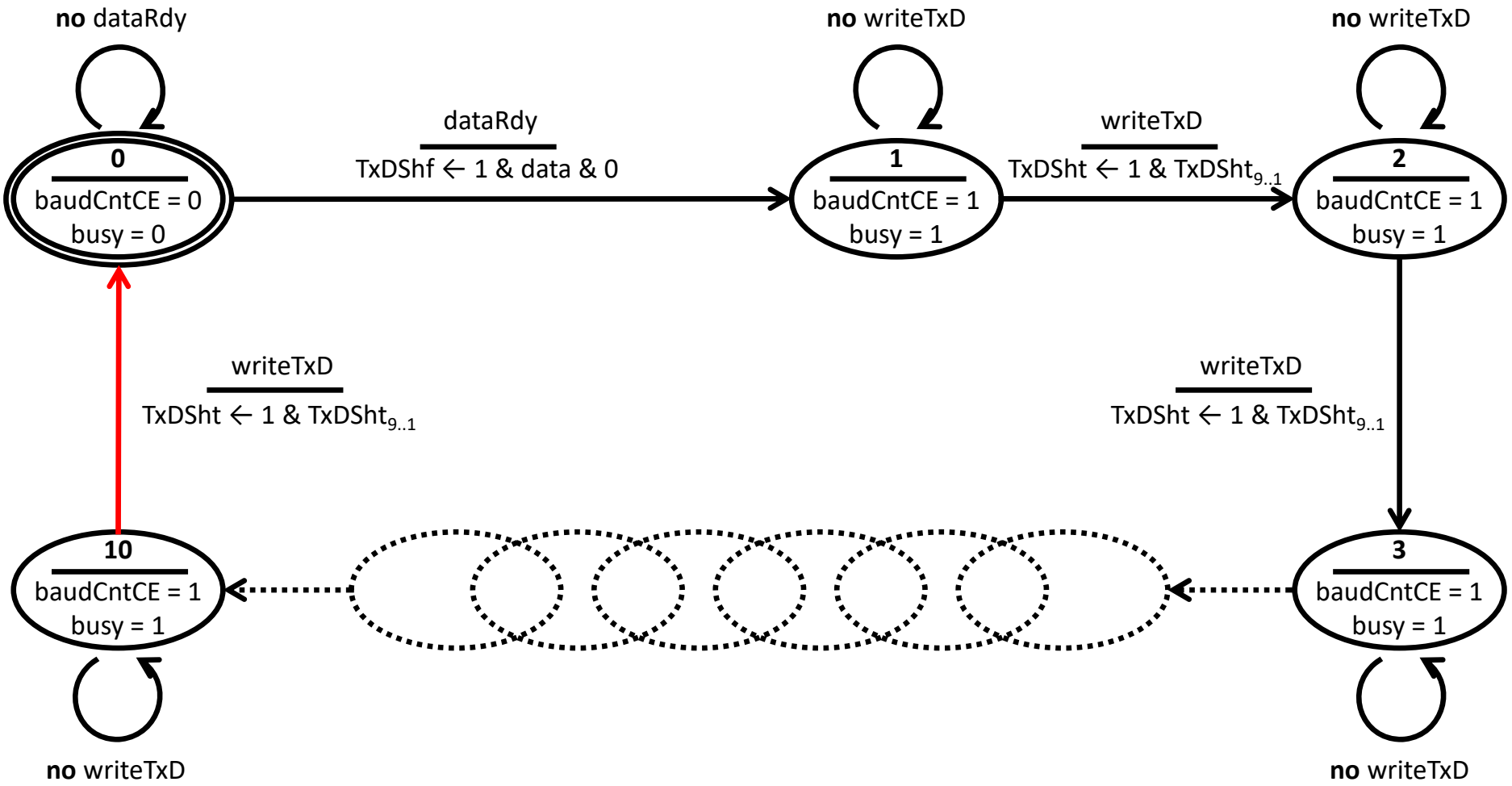
FSMD



J.M. Mendias
2024

Emisor RS-232

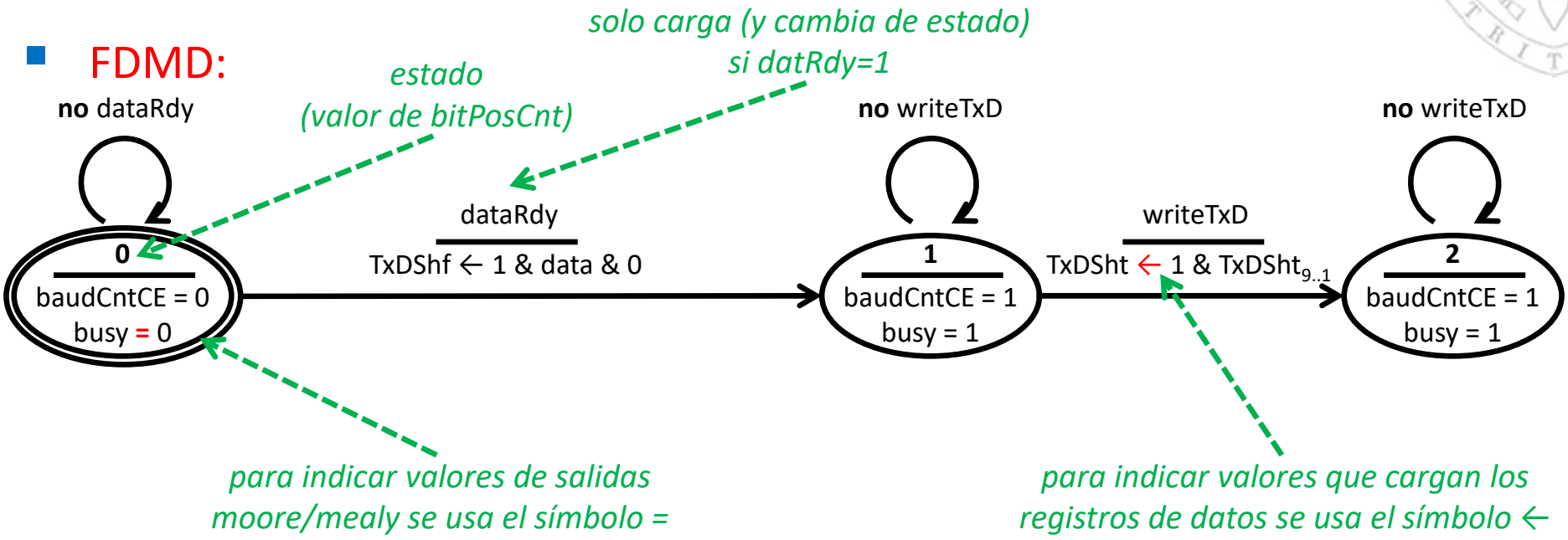
FSMD



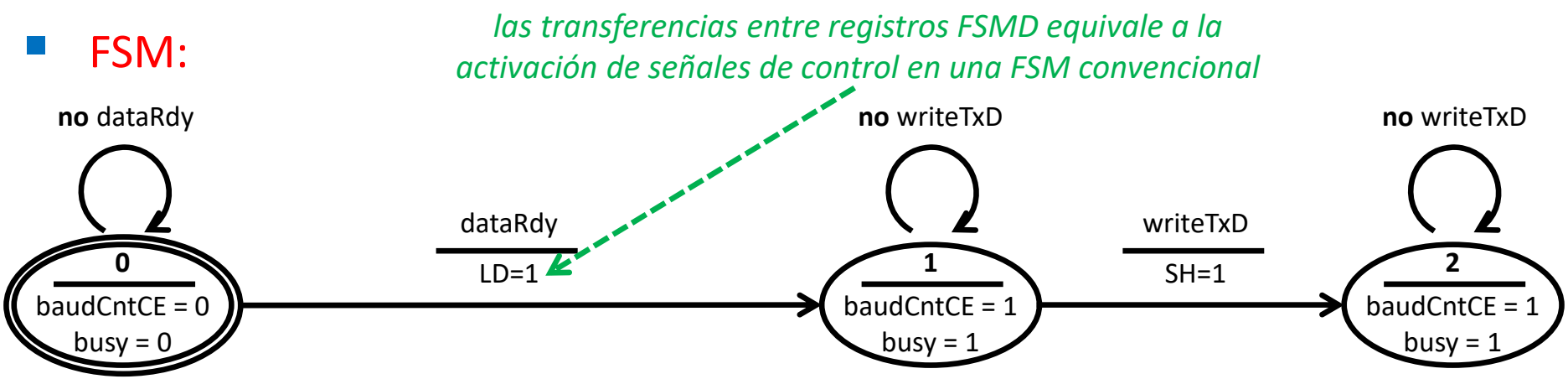
Emisor RS-232

FSMD vs FSM: nomenclatura

- FDMD:



- FSM:



Emisor RS-232

rs232transmitter.vhd

```
entity rs232transmitter is
  generic (
    FREQ_KHZ : natural;  -- frecuencia de operacion en KHz
    BAUDRATE : natural   -- velocidad de comunicacion
  );
  port (
    -- host side
    clk     : in  std_logic;   -- reloj del sistema
    rst     : in  std_logic;   -- reset síncrono del sistema
    dataRdy : in  std_logic;   -- se activa durante 1 ciclo cada vez que hay un nuevo
                               --   dato a transmitir
    data    : in  std_logic_vector (7 downto 0);   -- dato a transmitir
    busy    : out std_logic;   -- se activa mientras esta transmitiendo
    -- RS232 side
    TxD     : out std_logic    -- salida de datos serie del interfaz RS-232
  );
end rs232transmitter;

architecture syn of rs232transmitter is

  signal baudCntCE, writeTxD : boolean;

begin

  ...
end syn;
```

las señales internas tener tipos distintos a std_logic

J.M. Mendias
2024

Emisor RS-232

rs232transmitter.vhd

salidas Moore/Mealy

```
fsmd:
process (clk)
  variable bitPos : natural range 0 to 10 := 0;
  variable TxDShf : std_logic_vector (9 downto 0) := (others =>'1');
begin
  TxD       <= TxDShf(0);
  baudCntCE <= ...;
  if baudCntCE then
    busy <= '1';
  else
    busy <= '0';
  end if;
  if rising_edge(clk) then
    if rst='1' then
      TxDShf := (others =>'1');
      bitPos := 0;
    else
      case bitPos is
        when 0 =>
          if dataRdy='1' then
            TxDShf := "1" & data & "0";
            bitPos := 1;
          end if;
        when others =>
          ...
      end case;
    end if;
  end if;
end process;
```

transferencia entre registros

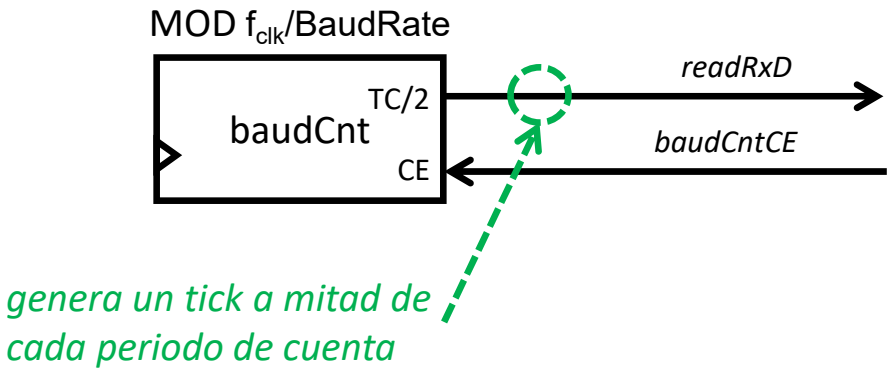
```
baudCnt:
process (clk)
  constant CYCLES : natural := (FREQ_KHZ*1000)/BAUDRATE;
  variable count  : natural range 0 to CYCLES-1 := 0;
begin
  writeTxD <= (count=CYCLES-1);
  if rising_edge(clk) then
    if baudCntCE then
      ...
    else
      count := 0;
    end if;
  end if;
end process;
```

genera un tick al final de cada periodo de cuenta

J.M. Mendías
2024

Receptor RS-232

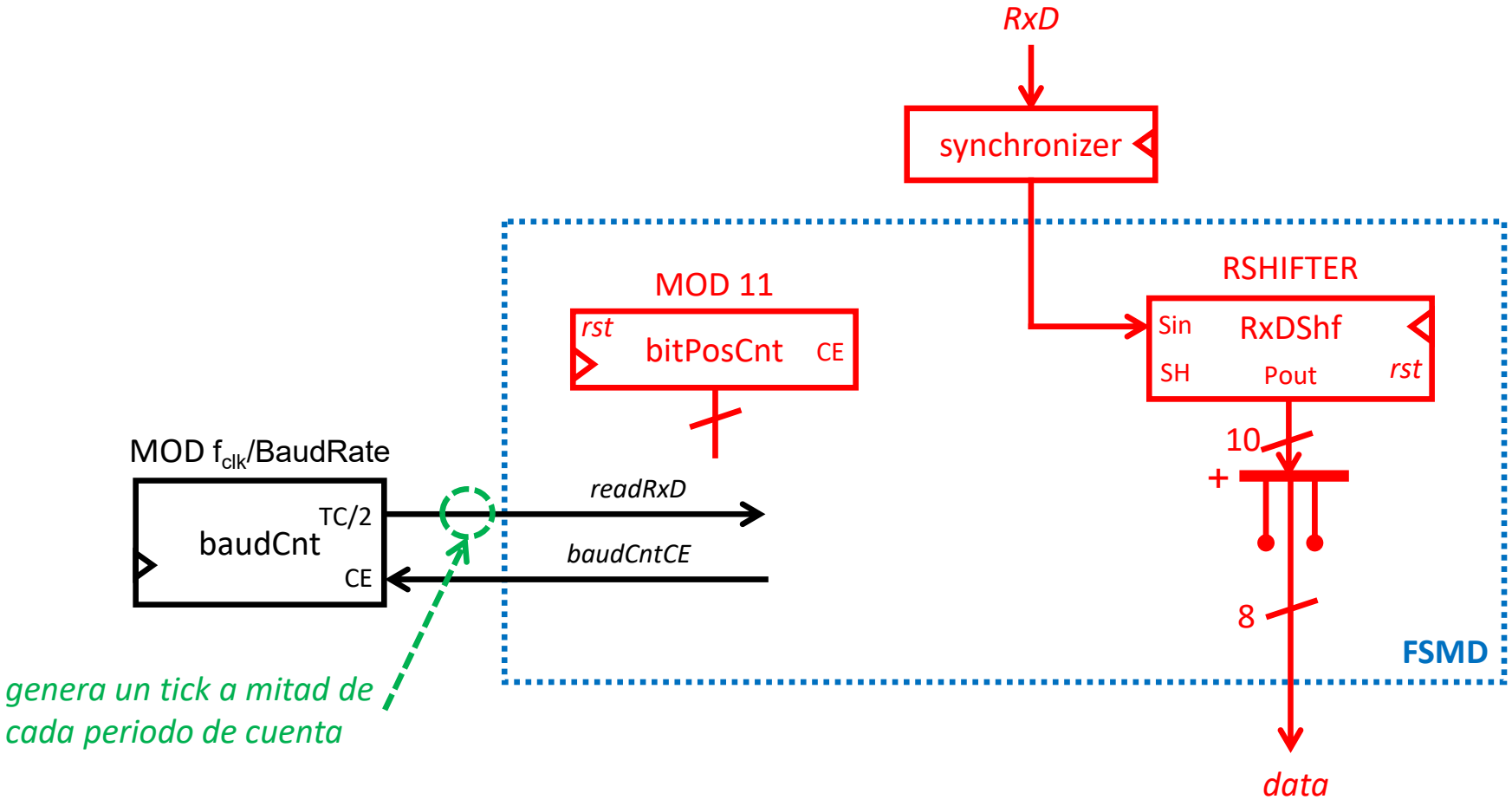
esquema RTL



J.M. Mendias
2024

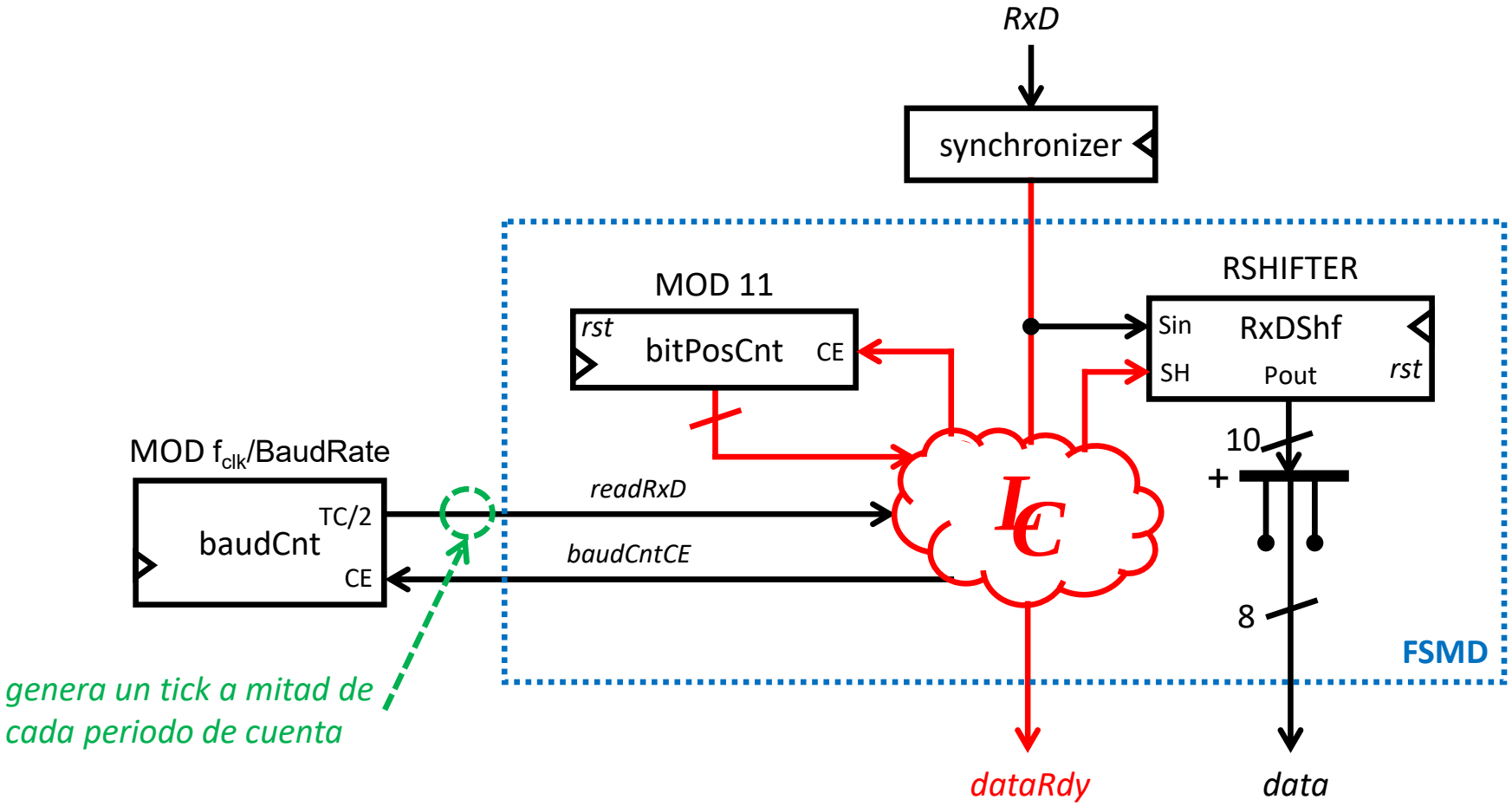
Receptor RS-232

esquema RTL



Receptor RS-232

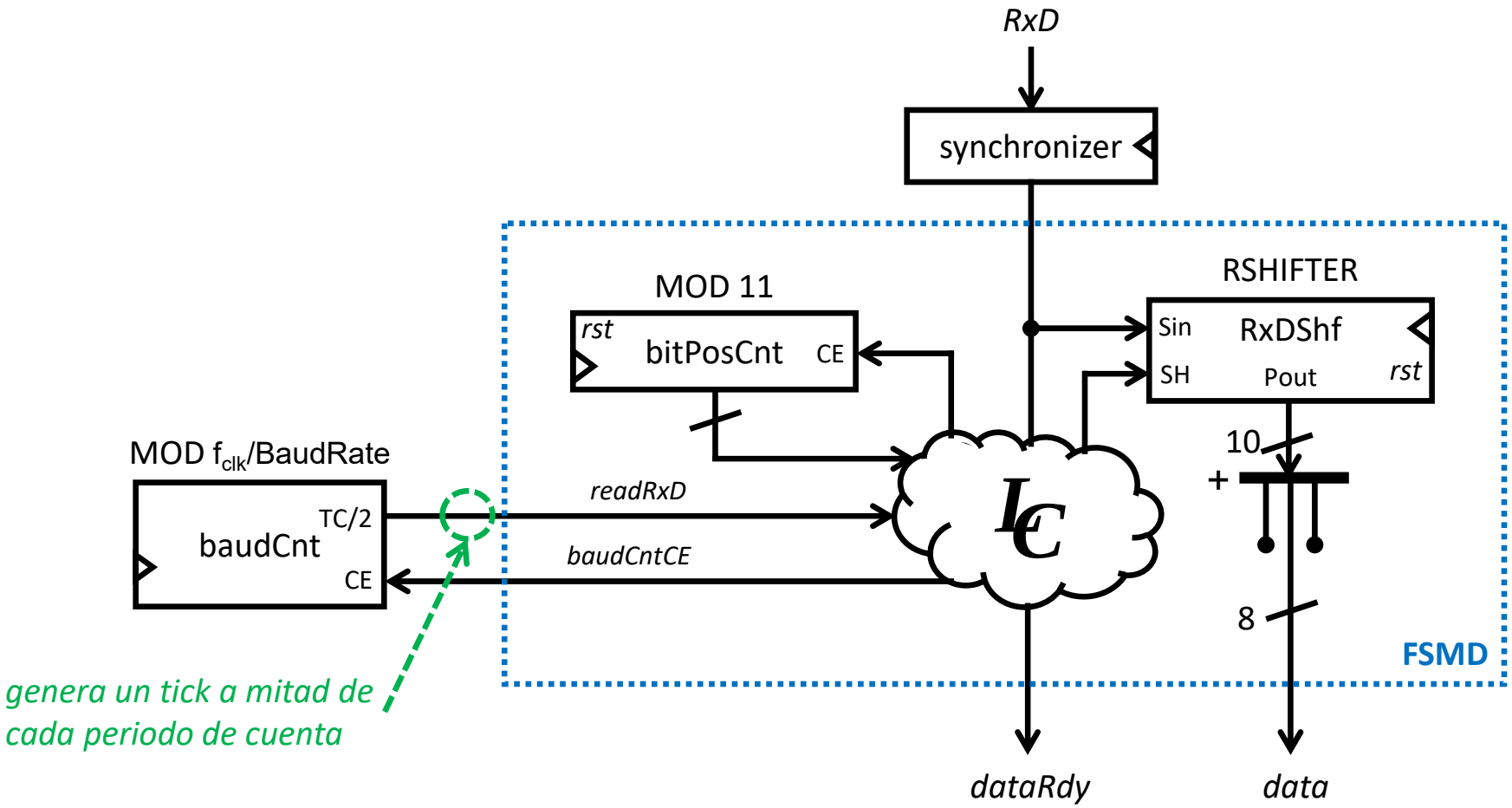
esquema RTL



J.M. Mendias
2024

Receptor RS-232

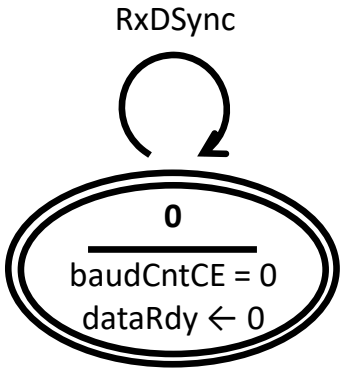
esquema RTL



J.M. Mendias
2024

Receptor RS-232

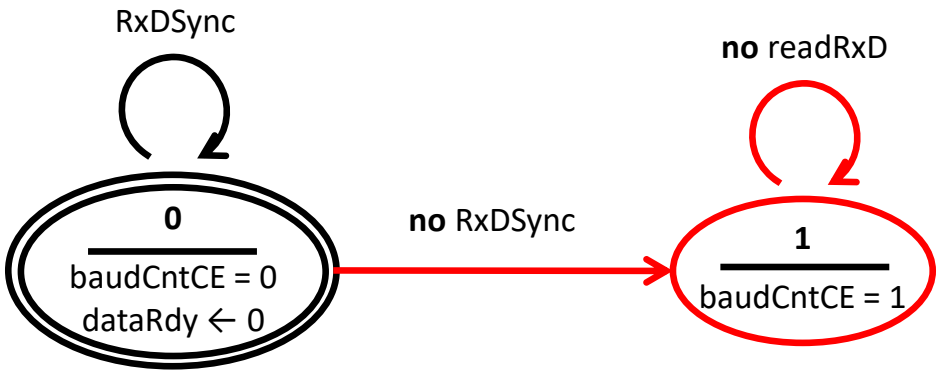
FSMD



J.M. Mendias
2024

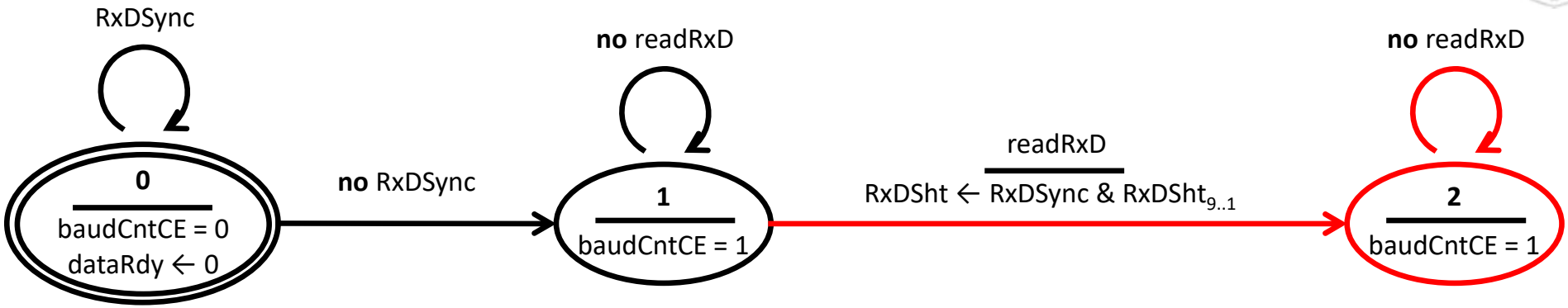
Receptor RS-232

FSMD



Receptor RS-232

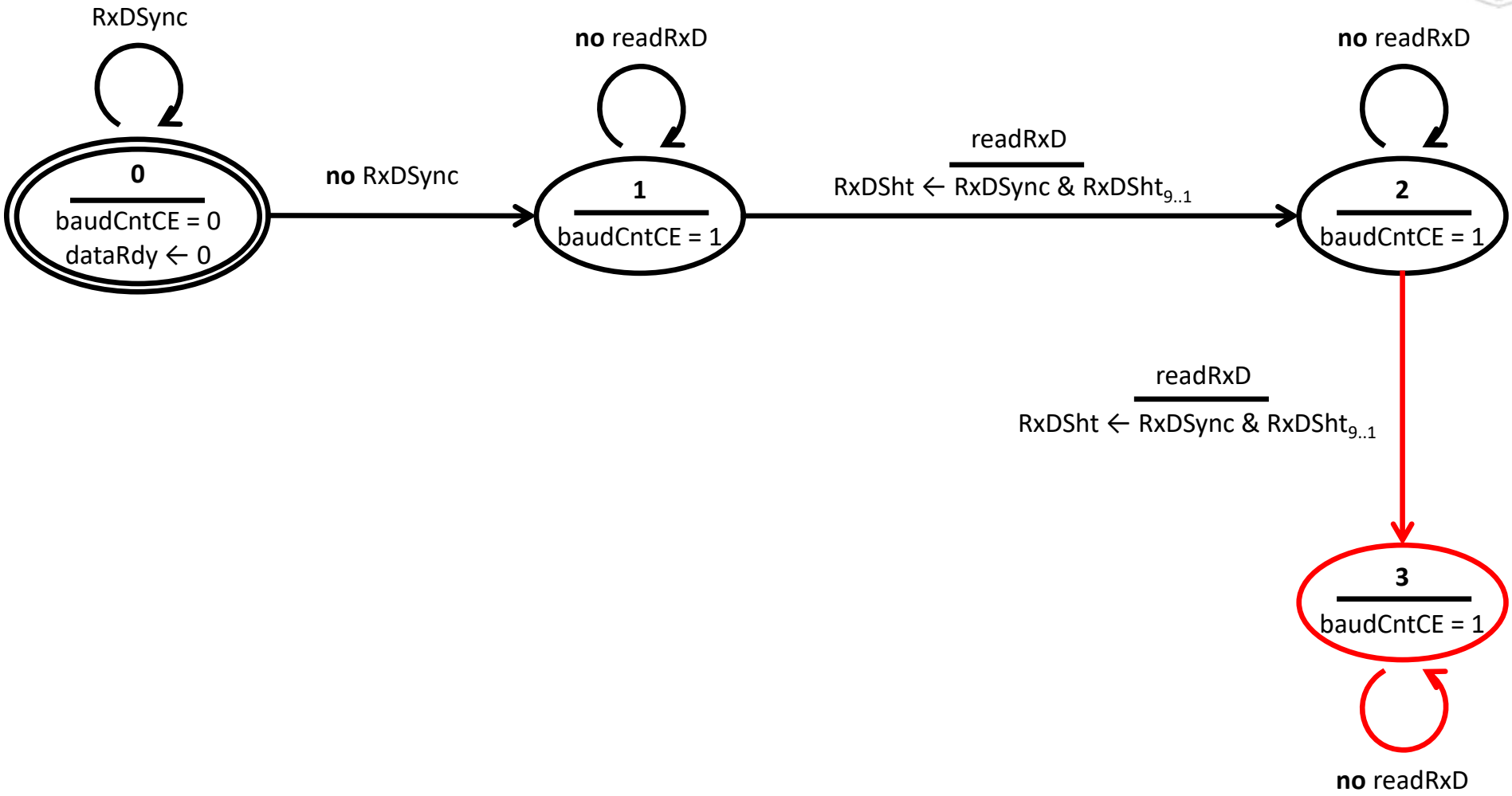
FSMD



J.M. Mendias
2024

Receptor RS-232

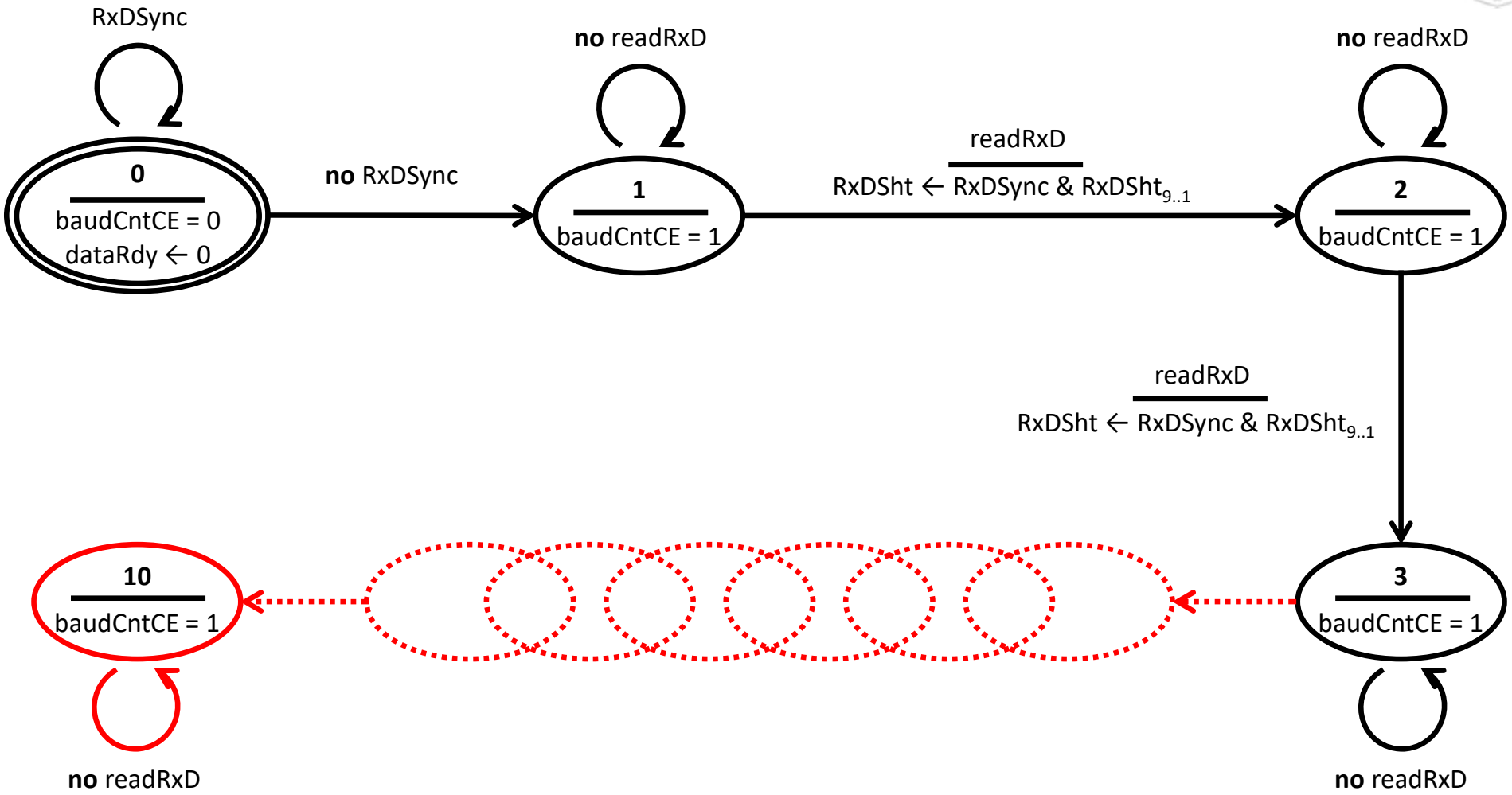
FSMD



J.M. Mendias
2024

Receptor RS-232

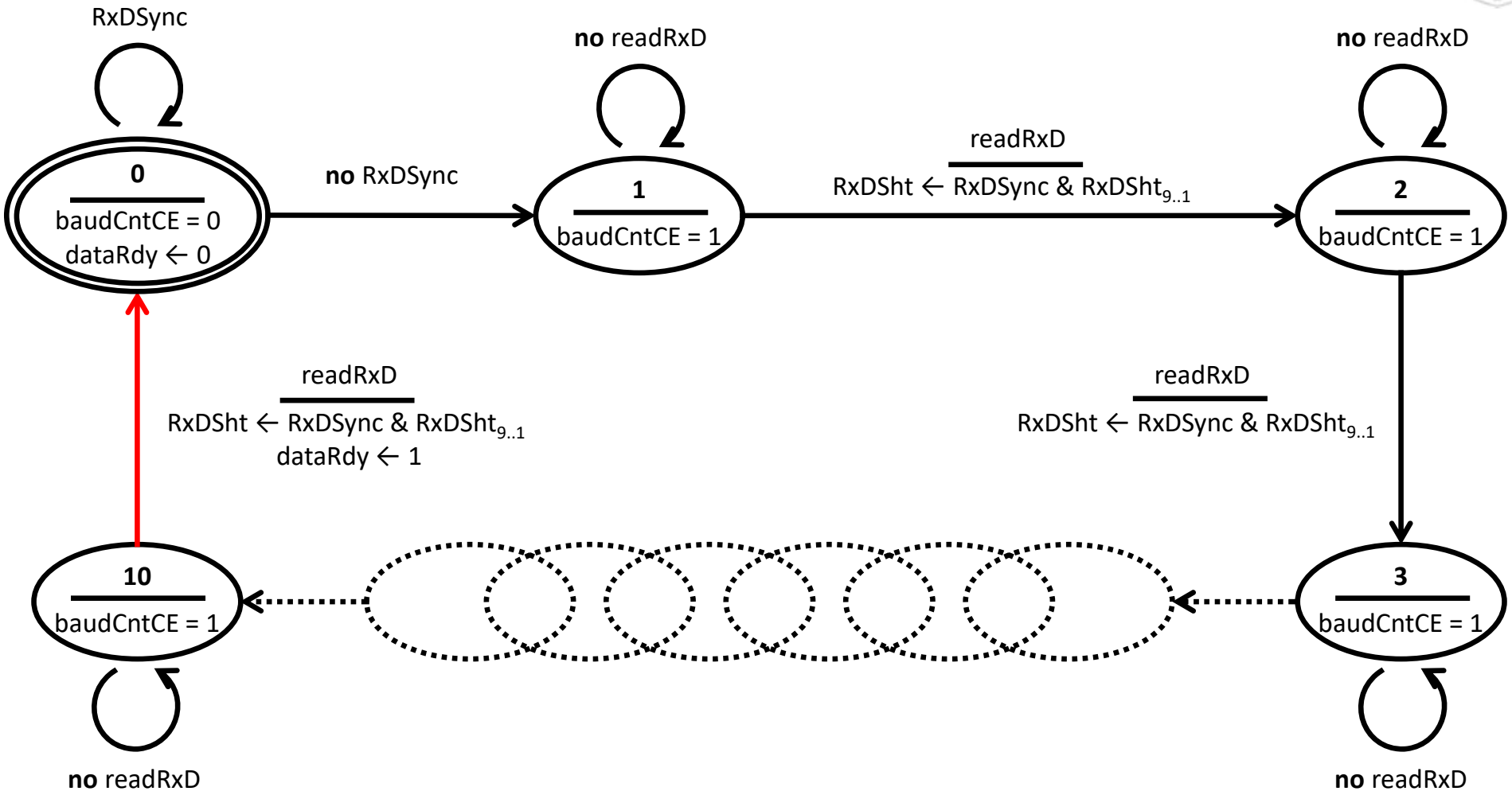
FSMD



J.M. Mendias
2024

Receptor RS-232

FSMD



Receptor RS-232

rs232receiver.vhd

```
entity rs232receiver is
  generic (
    FREQ     : natural;  -- frecuencia de operacion en KHz
    BAUDRATE : natural   -- velocidad de comunicacion
  );
  port (
    -- host side
    clk     : in  std_logic;   -- reloj del sistema
    rst     : in  std_logic;   -- reset síncrono 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
    -- RS232 side
    RxD     : in  std_logic    -- entrada de datos serie del interfaz RS-232
  );
end rs232receiver;

architecture syn of rs232receiver is

  signal RxDSync : std_logic;
  signal readRxD, baudCntCE : boolean;

begin

  ...
end syn;
```

J.M. Mendias
2024

Receptor RS-232

rs232receiver.vhd


```
RxDsynchronizer : synchronizer
...

fsmd:
process (clk)
  ...
begin
  data      <= ...;
  baudCntCE <= ...;
  if rising_edge(clk) then
    if rst='1' then
      ...
    else
      case bitPos is
        when 0 =>
          dataRdy   <= '0';
          ...
        when others =>
          if bitPos = 10 then
            dataRdy <= '1';
          end if;
          ...
      end case;
    end if;
  end if;
end process;
```

```
baudCnt:
process (clk)
  ...
begin
  readRxD <= (count=CYCLES/2-1);
  if rising_edge(clk) then
    ...;
  end if;
end process;
```

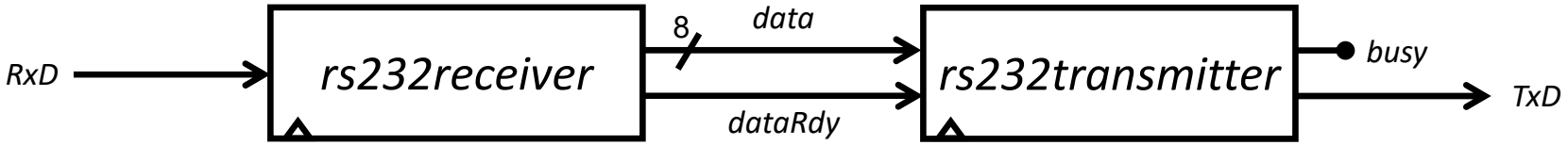
genera un tick a mitad de cada periodo de cuenta

J.M. Mendías
2024

Emisor/receptor RS-232

aplicación al diseño

- Para realizar un loopback bastaría:

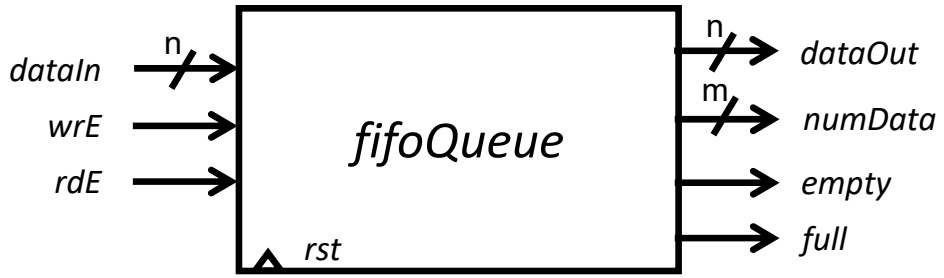


- Sin embargo, aunque emisor y receptor funcionen al mismo ancho de banda nominal:
  - Los host que usan el canal RS-232 suelen transmitir datos con cadencia irregular.
  - Es común añadir una FIFO intermedia para absorber los picos de carga.
  - Si el emisor y receptor tienen anchos de banda diferentes, la FIFO no solventa nada.

FIFO

presentación

- Diseñar una FIFO genérica en anchura y profundidad:
  - Aceptará peticiones de lectura y escritura (que podrán ser simultáneas):
    - Cuando active wrE, leerá dataIn y almacenará el dato internamente.
    - Cuando active rdE, escribirá en dataOut el dato más antiguo almacenado.
  - Señalizará mediante empty y full el estado de la FIFO
    - Una vez llena, ignorará peticiones de escritura adicionales.
    - Una vez vacía, ignorará peticiones de lectura adicionales.
  - Si la FIFO no está llena, numData indicará el número de datos almacenados.
    - Si la anchura de numData será el log2 de la profundidad.
  - Dispondrá de reset síncrono.

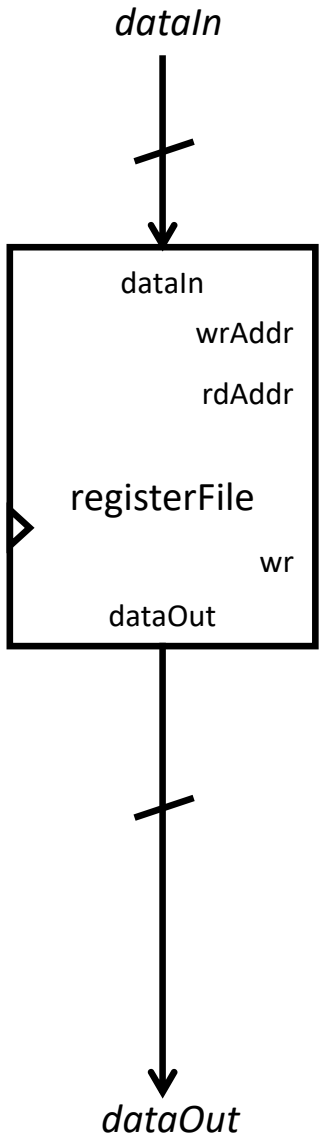


FIFO
diagrama RTL (i)

- Una FIFO consta de los siguientes elementos:
  - Un banco de registros (o una RAM) como almacén de datos.
    - Conceptualmente será gestionado como un buffer circular.
  - Dos punteros circulares (contadores)
    - Uno de escritura apuntando a la dirección en donde se escribirá el siguiente dato.
    - Otro de lectura apuntando a la dirección en donde leer el siguiente dato.
    - Solo avanzarán, respectivamente, a cada petición de escritura/lectura solo si la FIFO no esta llena/vacía.
    - La diferencia entre ambos índica el número de datos almacenados.
  - Un flag que indica si la FIFO está vacía.
    - Se activará cuando el puntero de lectura alcance al de escritura.
    - Se desactivará cuando se haga una escritura.
  - Un flag que indica si la FIFO está llena.
    - Se activará cuando el puntero de escritura alcance al de lectura.
    - Se desactivará cuando se haga una lectura.

FIFO

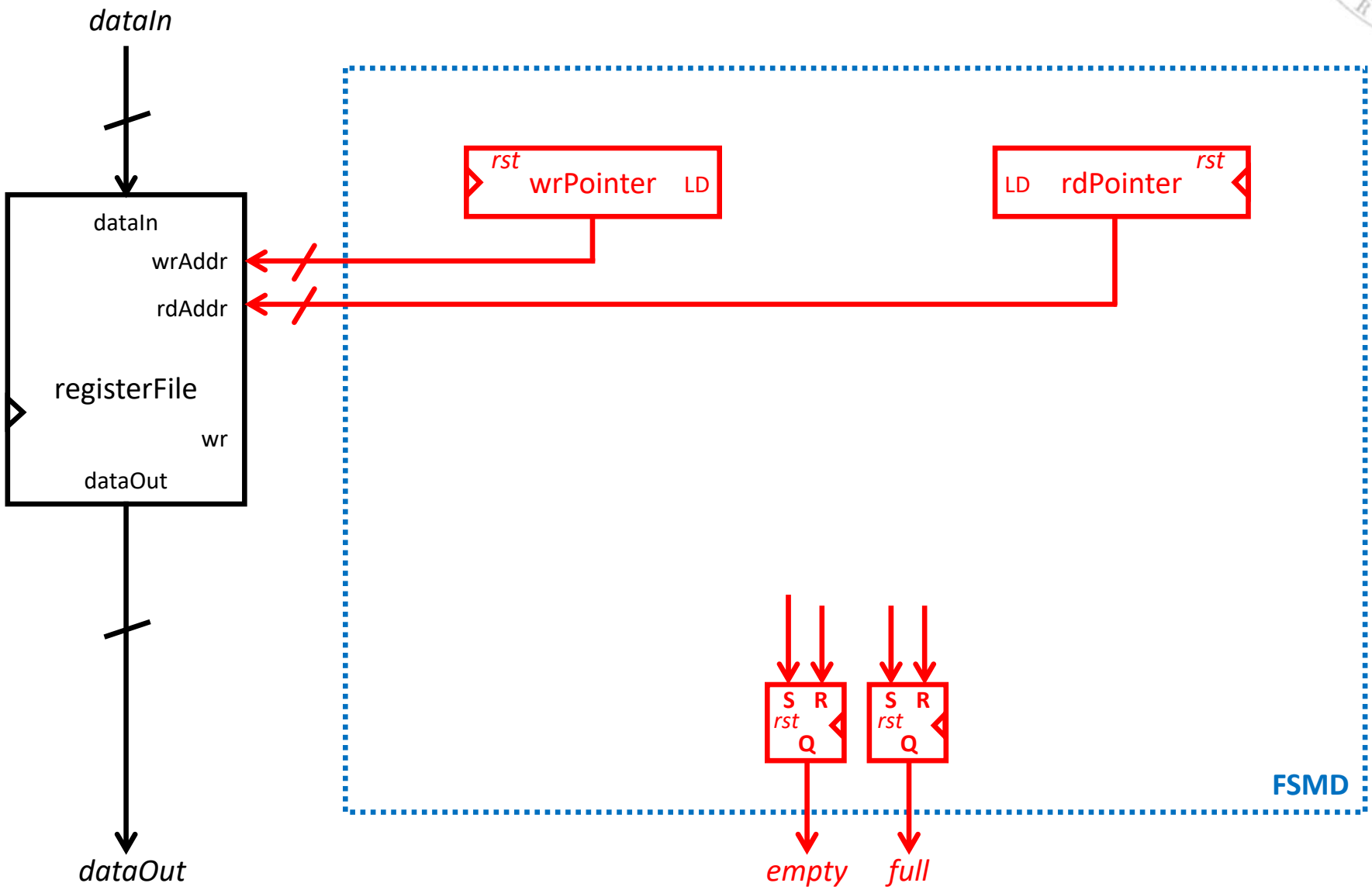
diagrama RTL (ii)



J.M. Mendias
2024

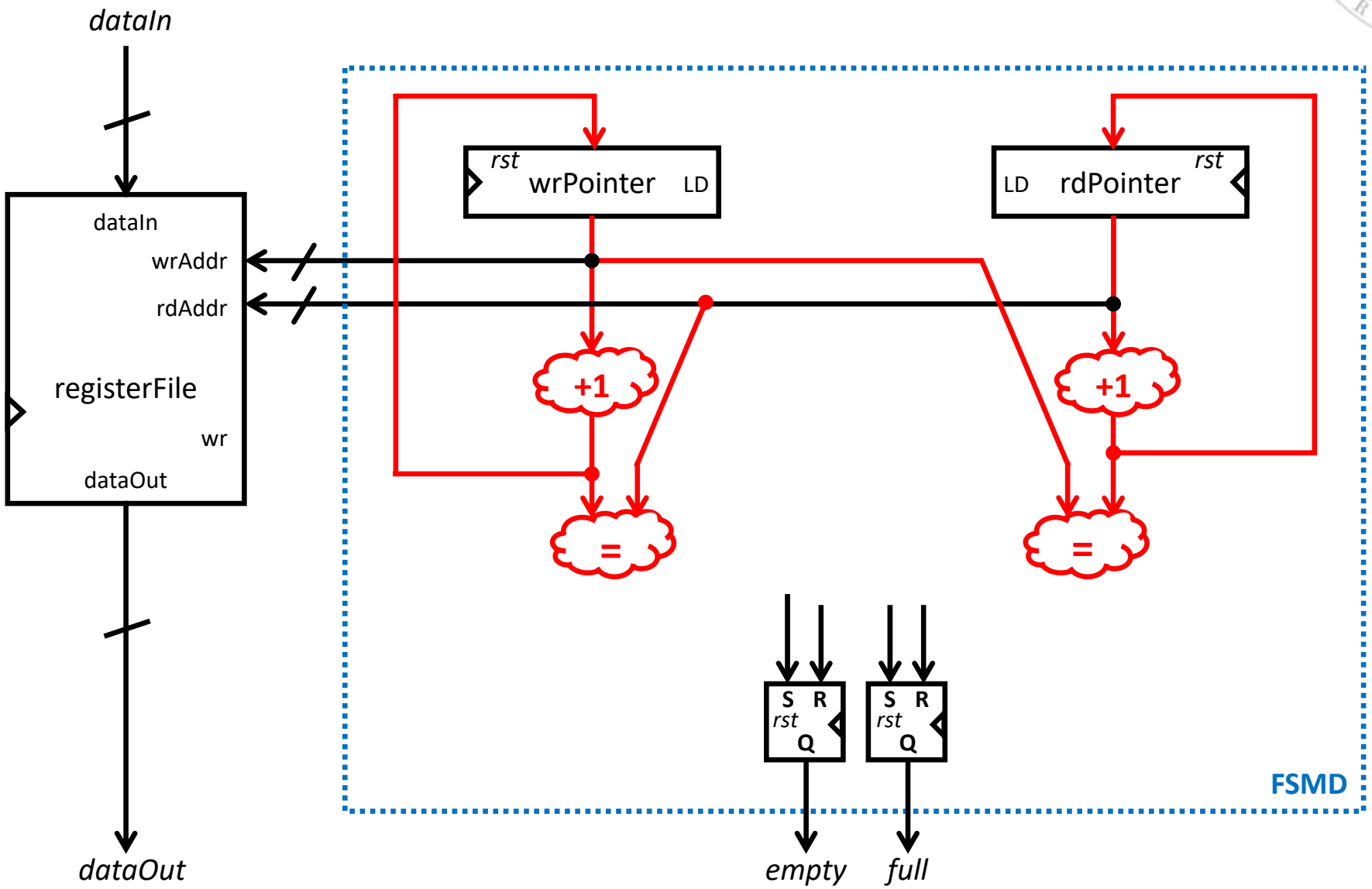
FIFO

diagrama RTL (ii)



FIFO

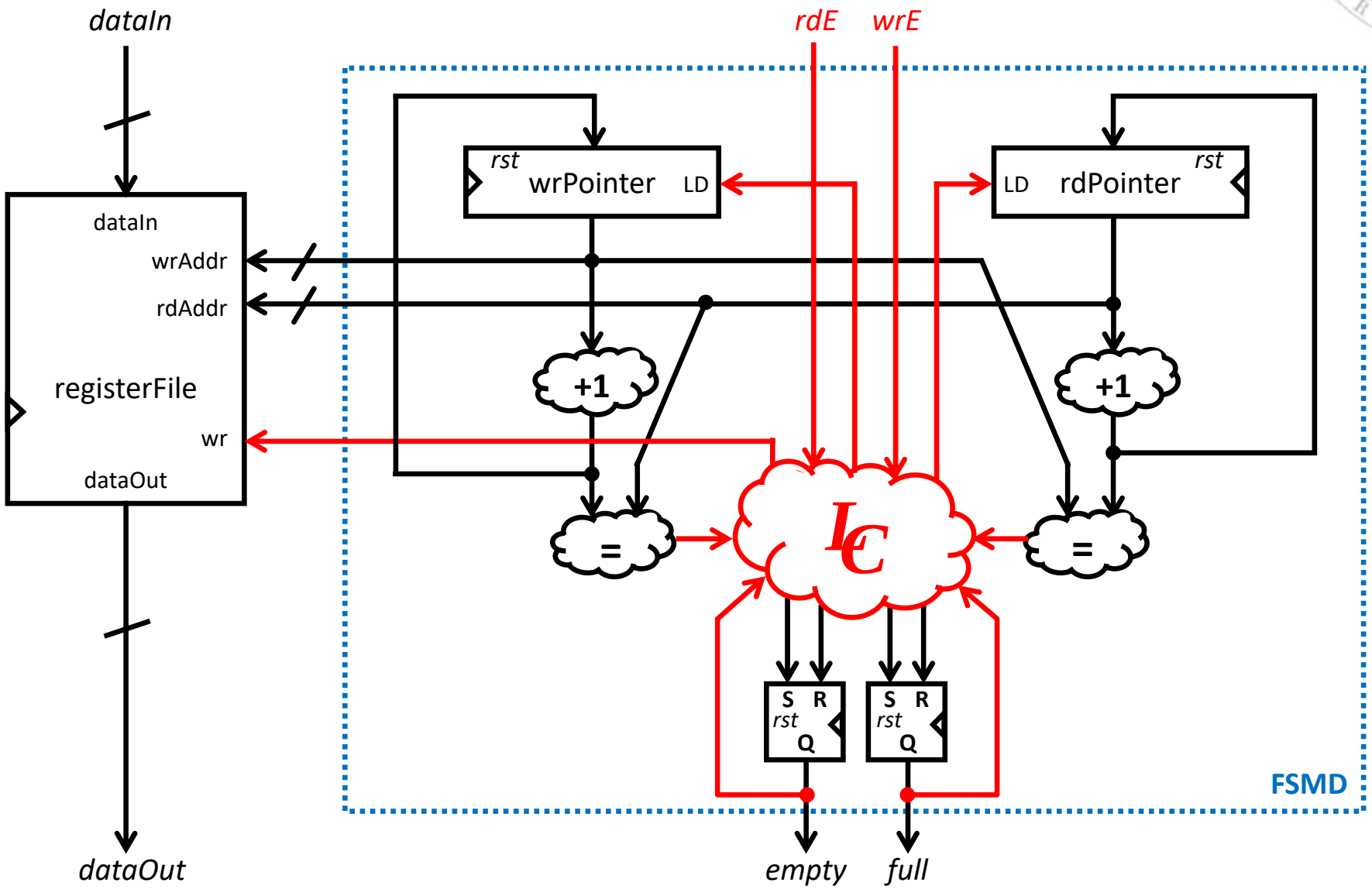
diagrama RTL (ii)



J.M. Mendias
2024

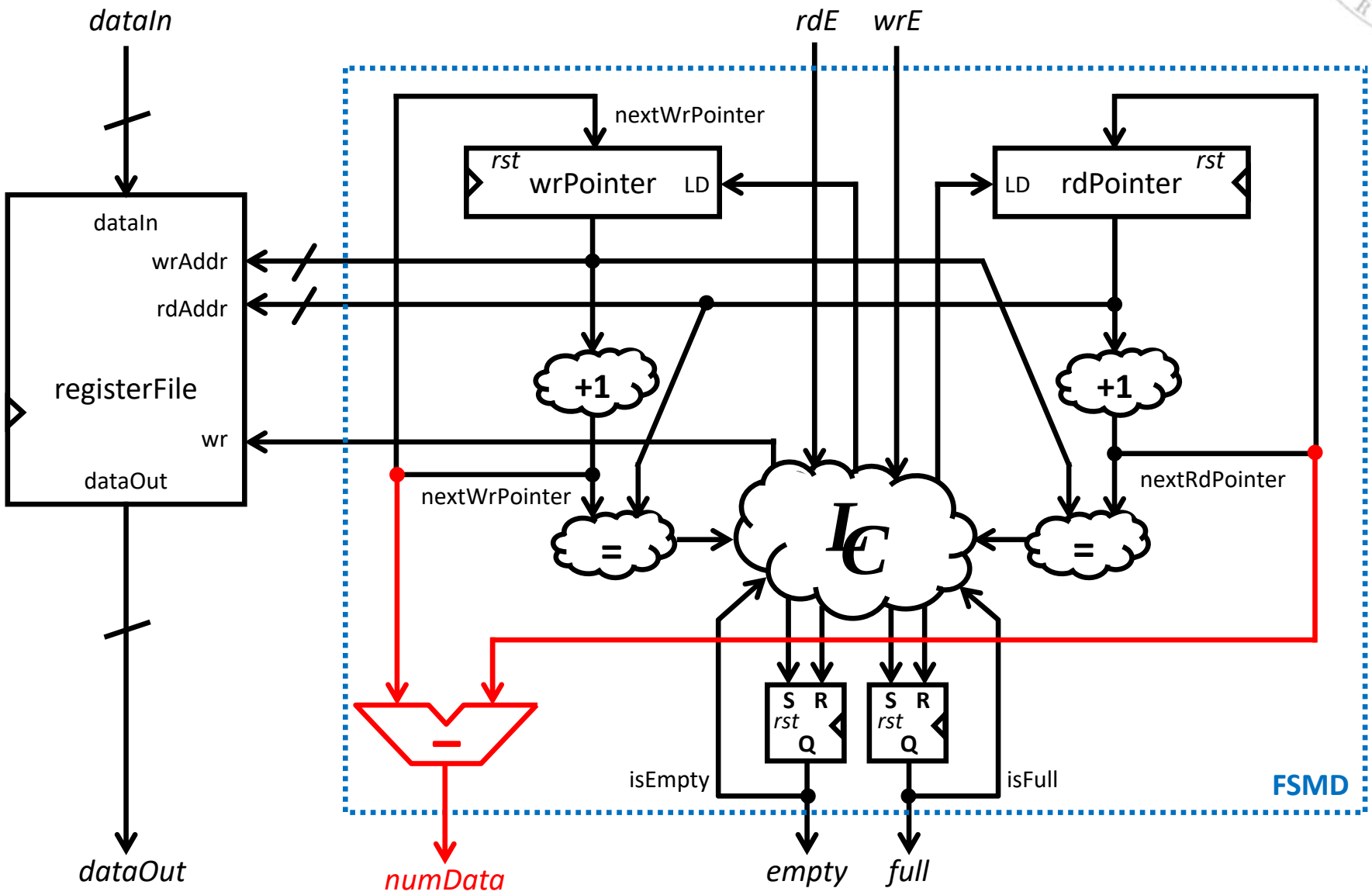
FIFO

diagrama RTL (ii)



FIFO

diagrama RTL (ii)



FIFO

fifoQueue.vhd

```
entity fifoQueue is
  generic (
    WL    : natural;   -- anchura de la palabra de fifo
    DEPTH : natural    -- numero de palabras en fifo
  );
  port (
    clk     : in  std_logic;   -- reloj del sistema
    rst     : in  std_logic;   -- reset síncrono del sistema
    wrE     : in  std_logic;   -- se activa durante 1 ciclo para escribir un dato en la fifo
    dataIn  : in  std_logic_vector(WIDTH-1 downto 0);   -- dato a escribir
    rdE     : in  std_logic;   -- se activa durante 1 ciclo para leer un dato de la fifo
    dataOut : out std_logic_vector(WIDTH-1 downto 0);   -- dato a leer
    numData : out std_logic_vector(log2(DEPTH)-1 downto 0);   -- numero de datos almacenados
    full    : out std_logic;   -- indicador de fifo llena
    empty   : out std_logic    -- indicador de fifo vacia
  );
end fifo;

architecture syn of fifo is
```

un banco de registro es un array de arrays de bits

```
  type regFileType is array (0 to DEPTH-1) of std_logic_vector(WIDTH-1 downto 0);
  signal regFile : regFileType := (others => (others => '0'));

  signal isFull  : std_logic := '0';
  signal isEmpty : std_logic := '1';

  signal wrPointer, rdPointer : natural range 0 to DEPTH-1 := 0
  signal nextWrPointer, nextRdPointer : natural range 0 to DEPTH-1;

  signal rdFifo, wrFifo : std_logic;
```

las señales internas tener tipos distintos a std_logic_vector

```
begin
  ...
end syn;
```

FIFO

fifoQueue.vhd

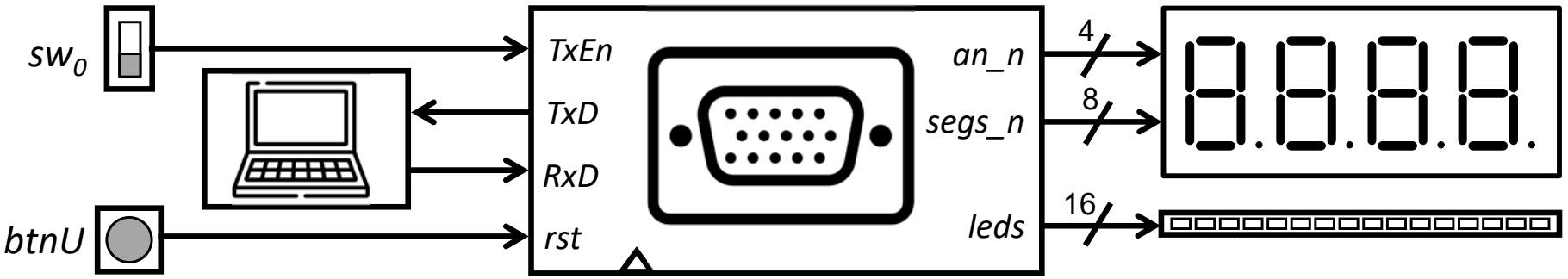
```
registerFile :
process (clk, rdPointer, regFile)
begin
  dataOut <= ...;
  if rising_edge(clk) then
    if wrFifo='1' then
      ...;
    end if;
  end if;
end process;
```

```
full    <= isFull;
empty   <= isEmpty;

numData <= ...;

wrFifo  <= ...;
rdFifo  <= ...;

nextWrPointer <= ...;
nextRdPointer <= ...;

fsmd :
process (clk)
begin
 if rising_edge(clk) then
    if rst='1' then
      wrPointer <= 0;
      rdPointer <= 0;
      isFull    <= '0';
      isEmpty   <= '1';
    else
      if wrFifo='1' then
        ...
      end if;
      if rdFifo='1' then
        ...
      end if;
    end if;
  end if;
end process;
```

J.M. Mendías
2024

Diseño principal

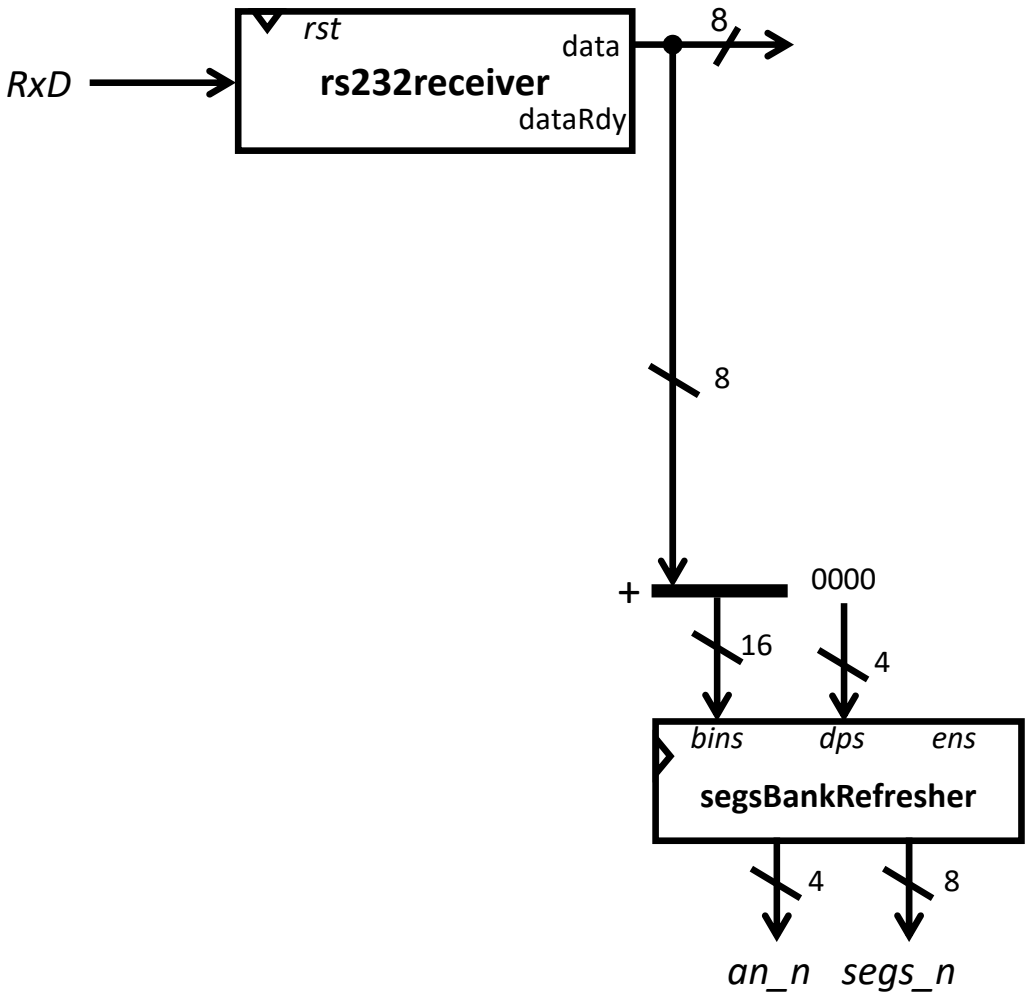
- Para testar el par emisor-receptor realizaremos un diseño que retransmita lo que reciba por el bus RS-232 previo paso por una FIFO.
  - El PC enviará y recibirá los datos a través del cable USB.
  - En el PC correrá un emulador de terminal (Termite) con la siguiente configuración:
    - Baud Rate: 1200, Data bits: 8, Stop bits: 1, Parity: none, Flow Control: none
  - El receptor almacenará los datos en la FIFO y el transmisor los tomará de la FIFO.
    - La FIFO tendrá una capacidad de 16B.
  - La salida del receptor se visualizará en los displays 7-segmentos más significativos.
  - El estado de la FIFO vacía/llena se mostrará (E/F) en el display 7-segmentos derecho.
    - Además deberán estar encendidos tantos leds como datos contenga la FIFO.
  - El transmisor podrá inhabilitarse desactivando el switch 0.
    - El receptor seguirá recibiendo datos y almacenándolos en la FIFO hasta llenarla.



J.M. Mendías
2024

Diseño principal

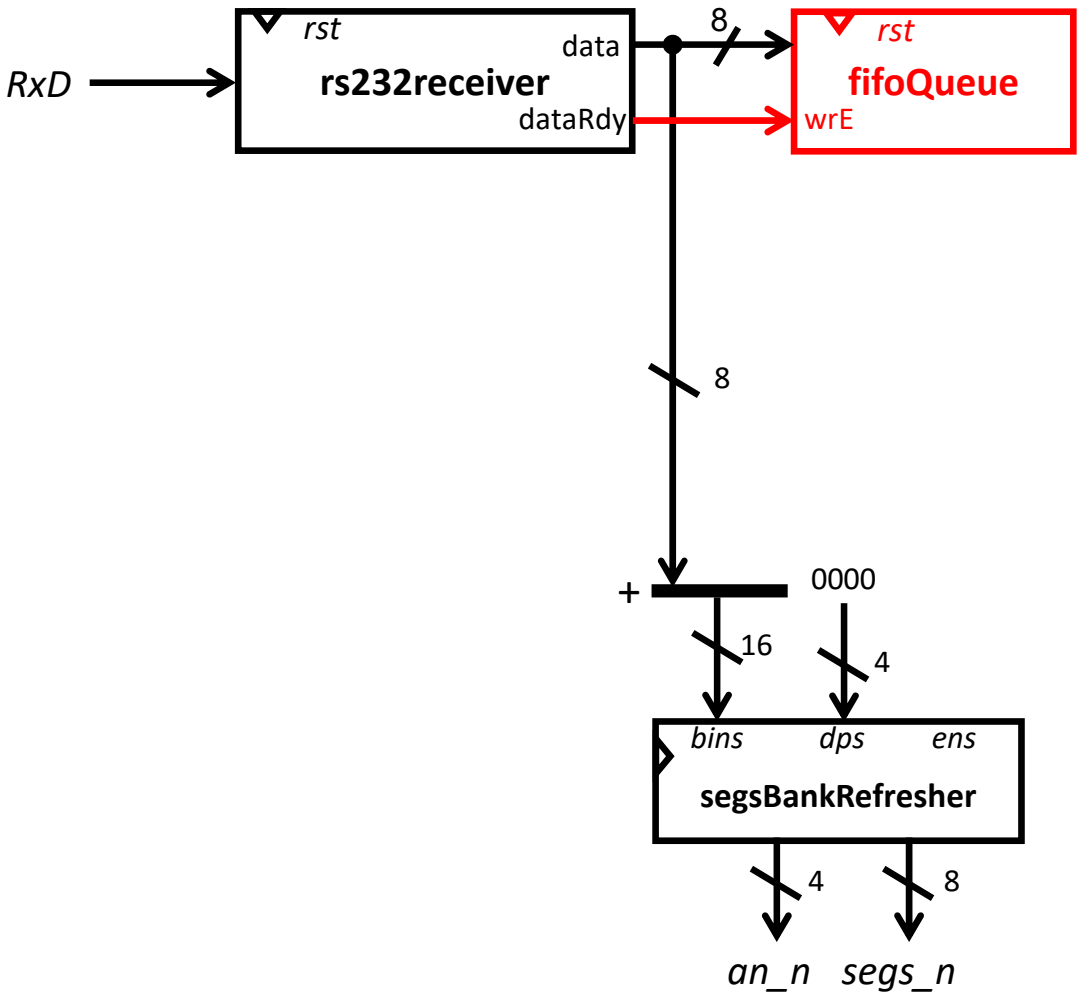
esquema RTL



J.M. Mendias
2024

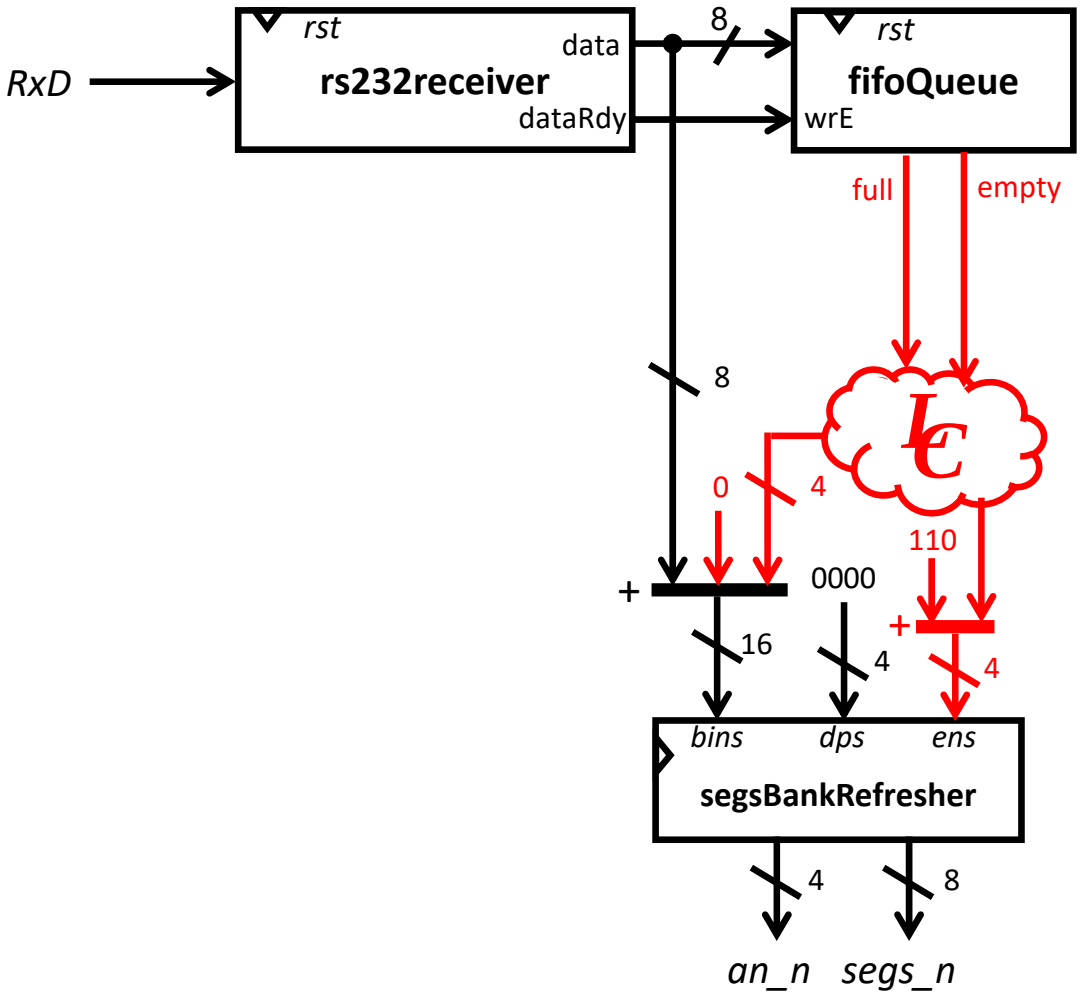
Diseño principal

esquema RTL



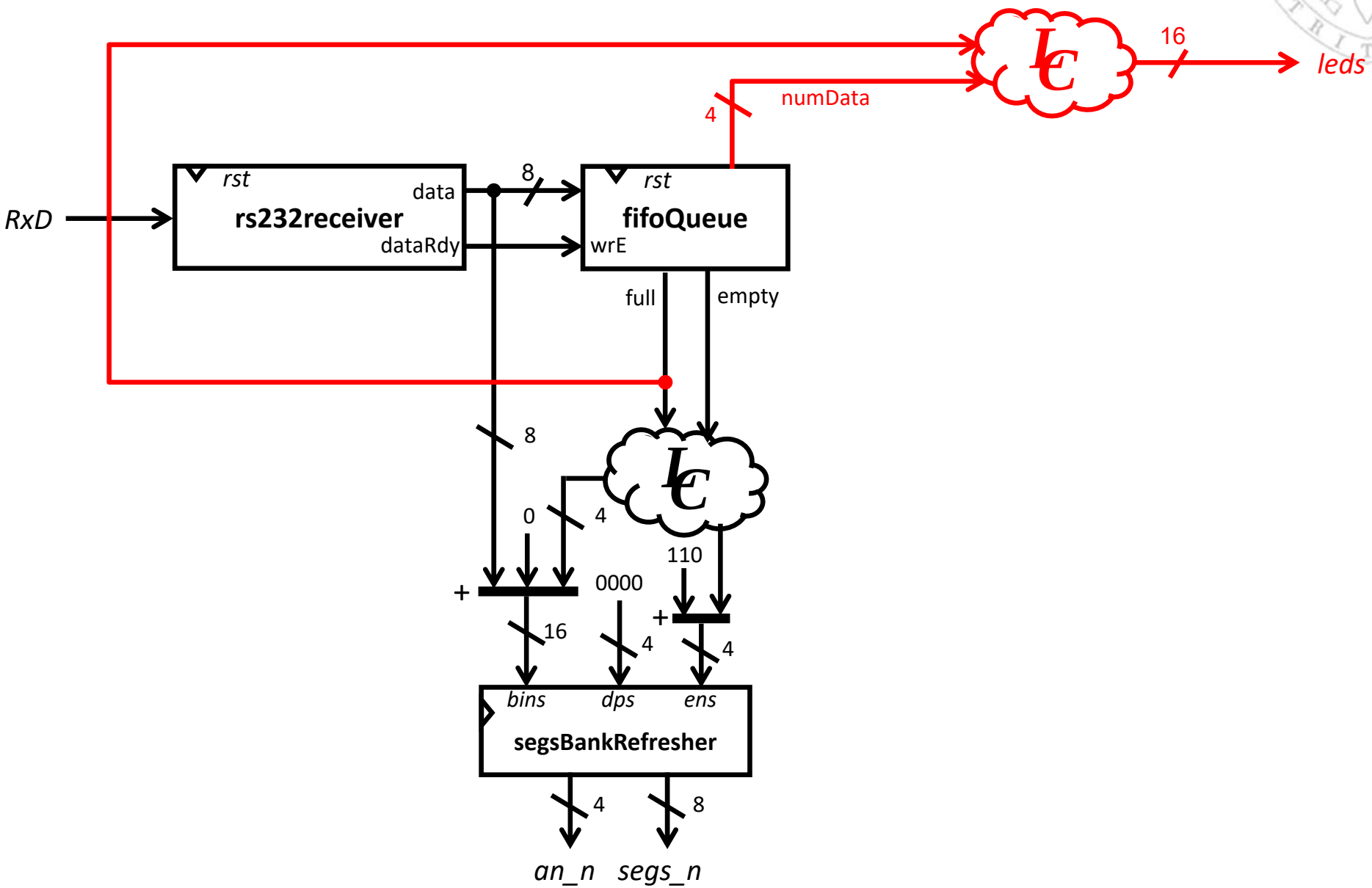
Diseño principal

esquema RTL



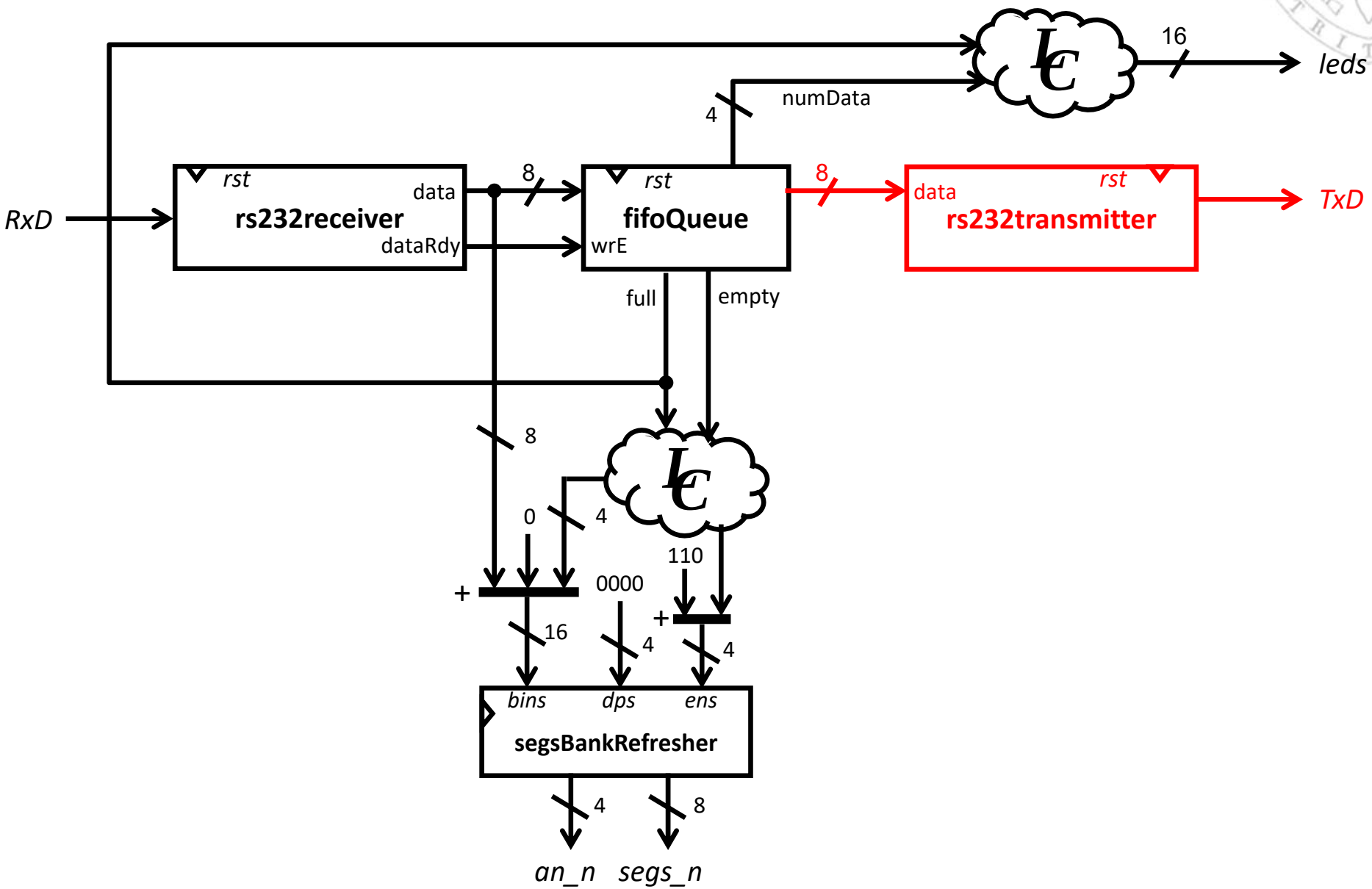
Diseño principal

diagrama RTL



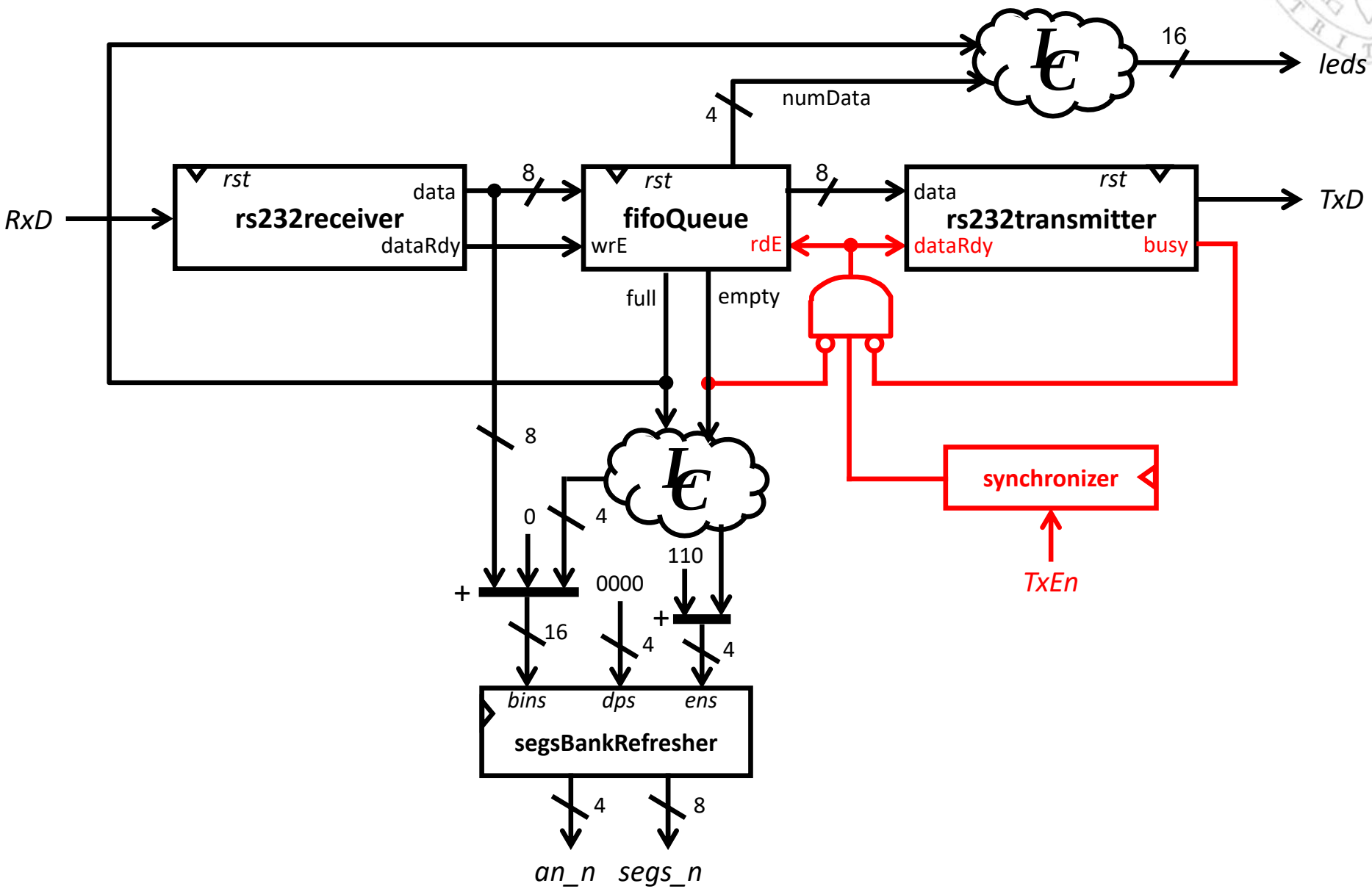
Diseño principal

diagrama RTL



Diseño principal

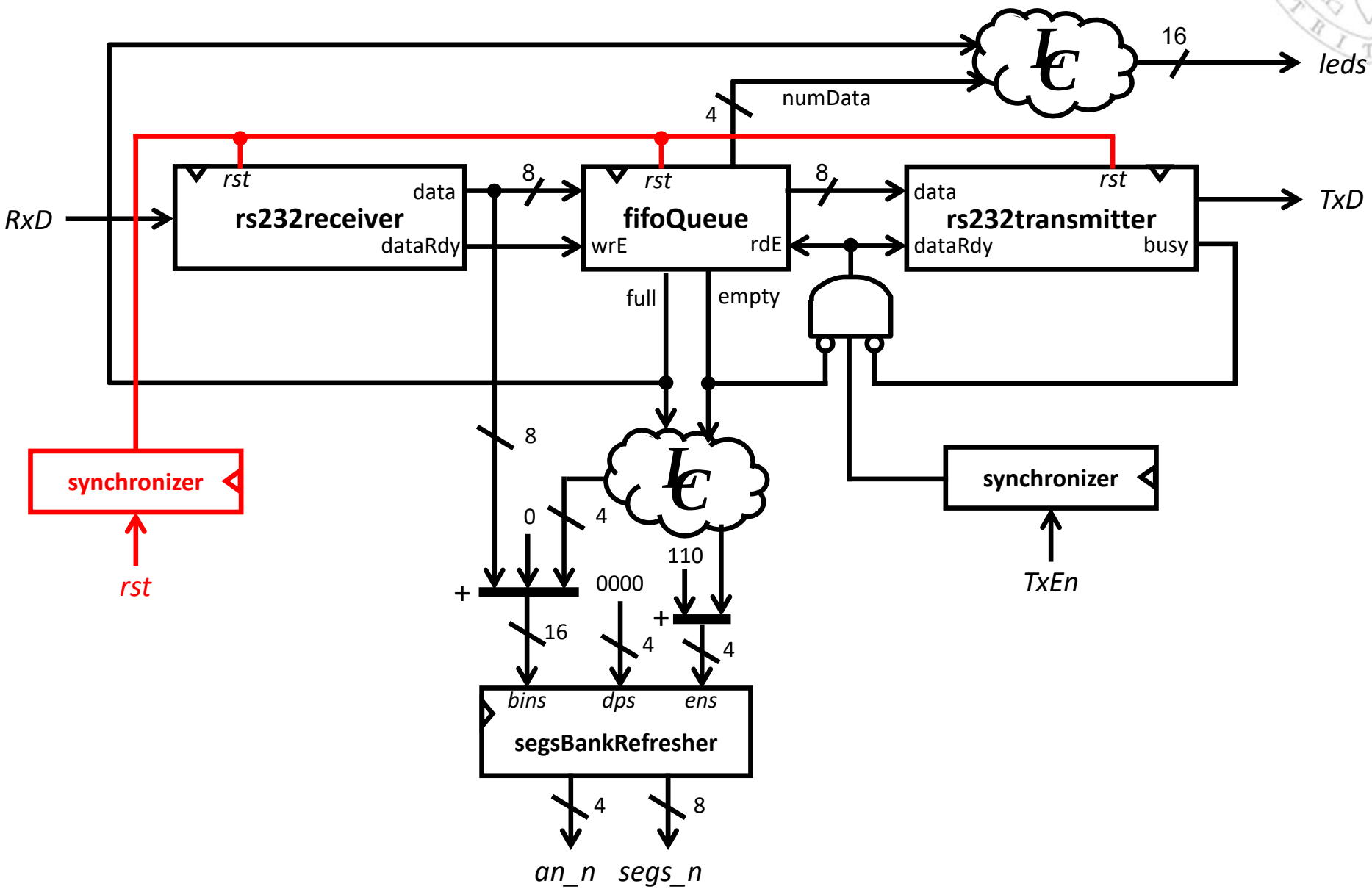
esquema RTL



J.M. Mendias
2024

Diseño principal

esquema RTL



J.M. Mendías
2024

Tareas

loopback sin FIFO

1. En la carpeta **`DAS/source`** crear las carpeta **`lab5loopback`**.
2. Descargar de la Web los ficheros en:
   - **`common:`** **`rs232receiver.vhd`** y **`rs232transmitter.vhd`**
   - **`lab5:`** **`lab5loopback.vhd`** y **`lab5loopback.xdc`**
3. Completar **`common.vhd`** con la declaración de los nuevos componentes.
4. Completar el código omitido en los ficheros:
   - **`rs232receiver.vhd`** y **`rs232transmitter.vhd`**
5. En la carpeta **`DAS/projects`** crear el proyecto **`lab5loopback`**.
6. Incluir en el proyecto (sin copiarlos) los siguientes fuentes:
   - **`common.vhd`**, **`synchronizer.vhd`**, **`rs232receiver.vhd`**, **`rs232transmitter.vhd`**, **`lab5loopback.vhd`** y **`lab5loopback.xdc`**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Arrancar un emulador de terminal (Termite) en el PC.
   - Baud Rate: 1200, Data bits: 8, Stop bits: 1, Parity: none, Flow Control: none
9. Conectar la placa y encenderla.
10. Volcar el fichero de configuración **`lab5loopback.bit`**

J.M. Mendías
2024

Tareas

loopback con FIFO

1. En la carpeta **DAS/source** crear las carpeta **lab5fifo**.
2. Descargar de la Web los ficheros en:
   - **common:** **fifoQueue.vhd**
   - **lab5:** **lab5fifo.vhd** y **lab5fifo.xdc**
3. Completar **common.vhd** con la declaración del nuevo componentes.
4. Completar el código omitido en los ficheros:
   - **fifoQueue.vhd** y **lab5fifo.vhd**
5. En la carpeta **DAS/projects** crear el proyecto **lab5fifo**.
6. Incluir en el proyecto (sin copiarlos) los siguientes fuentes:
   - **common.vhd**, **bin2segs.vhd**, **segsBankRefresher.vhd**, **synchronizer.vhd**, **fifoQueue.vhd**, **rs232receiver.vhd**, **rs232transmitter.vhd**, **lab5fifo.vhd** y **lab5fifo.xdc**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Arrancar un emulador de terminal (Termite) en el PC.
   - Baud Rate: 1200, Data bits: 8, Stop bits: 1, Parity: none, Flow Control: none
9. Conectar la placa y encenderla.
10. Volcar el fichero de configuración **lab5fifo.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/