Laboratorio 5:

FSM con ruta de datos

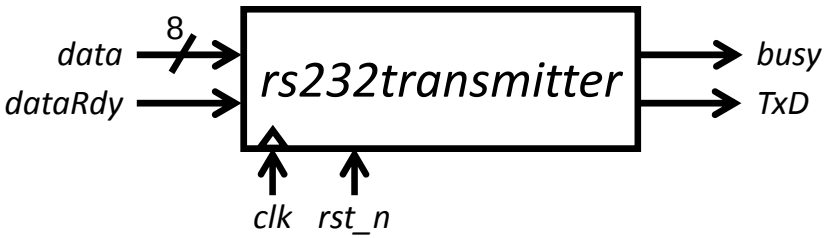
comunicación con un terminal a través de un bus RS-232

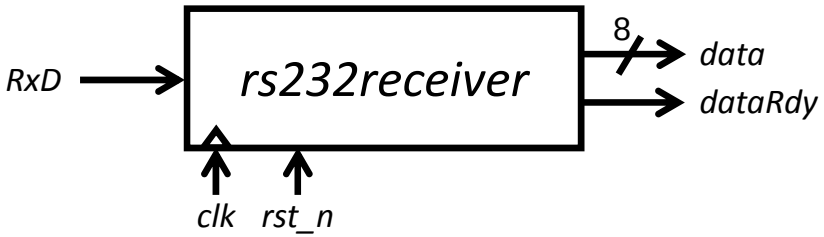
Diseño automático de sistemas

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 realizarán 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, comenzará a transmitir el dato leído de data.
  - 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.





J.M. Mendías 2015

Protocolo RS-232

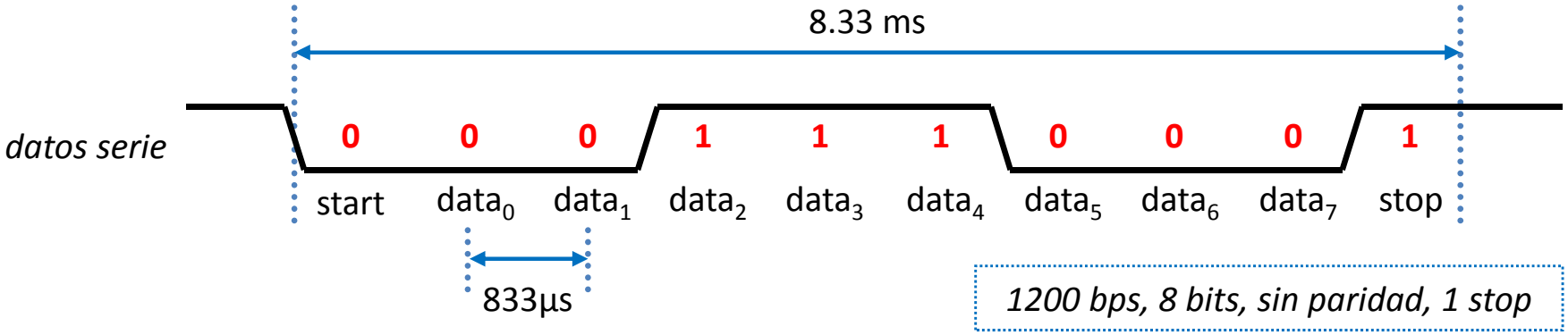
presentación

- 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).

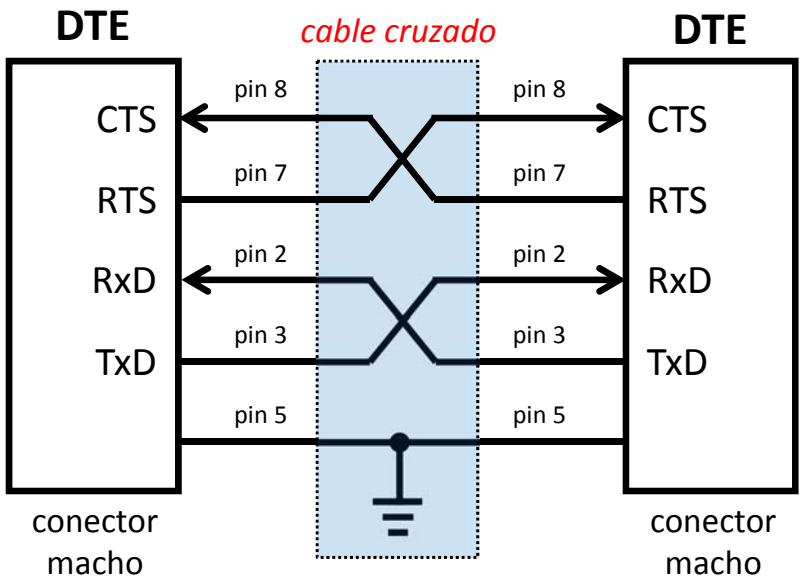


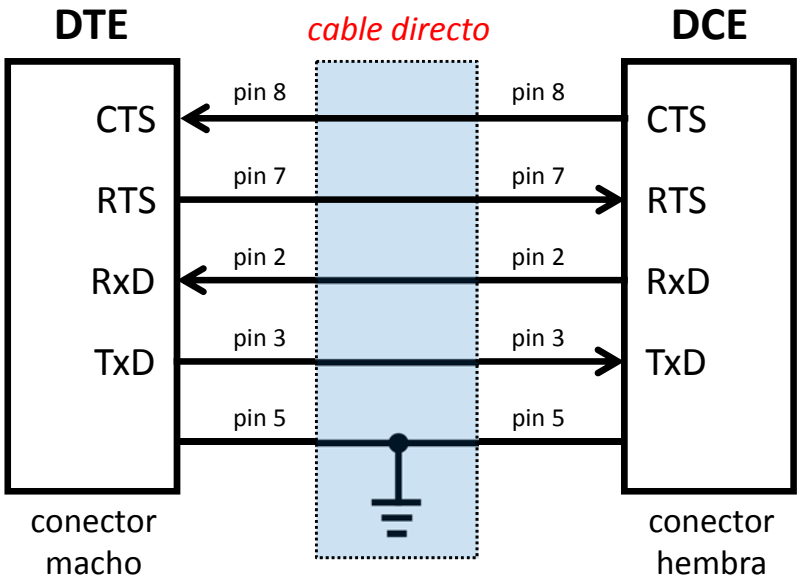
J.M. Mendías 2015

Protocolo RS-232

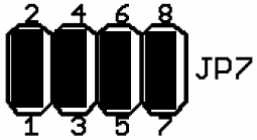
interconexionado

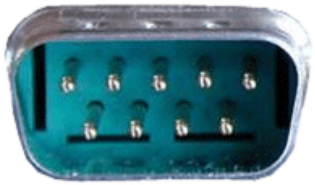
- El esquema de conexión, dependerá del tipo de elementos conectados
  - DTE = host, DCE = dispositivo





En los labs los cables son cruzados.
Posición de los jumpers JP7 de la placa XST



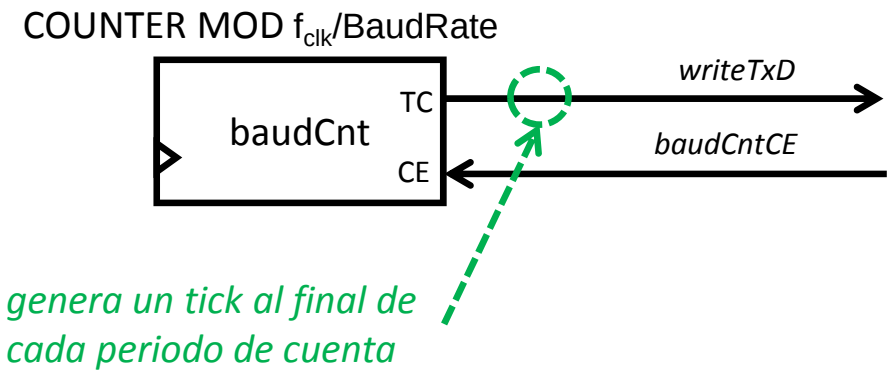


DB9

J.M. Mendías
2015

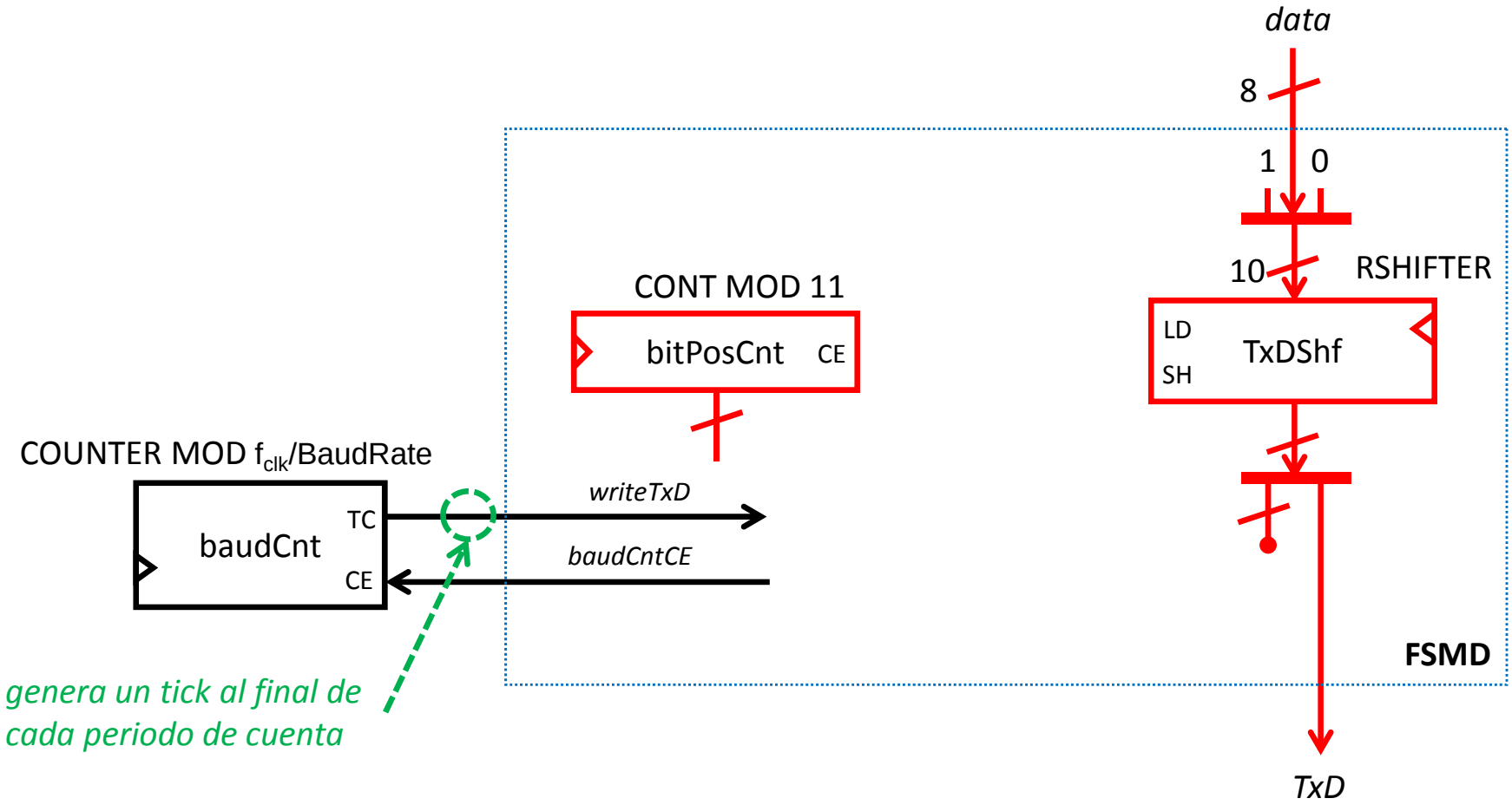
Emisor RS-232

diagrama RTL



Emisor RS-232

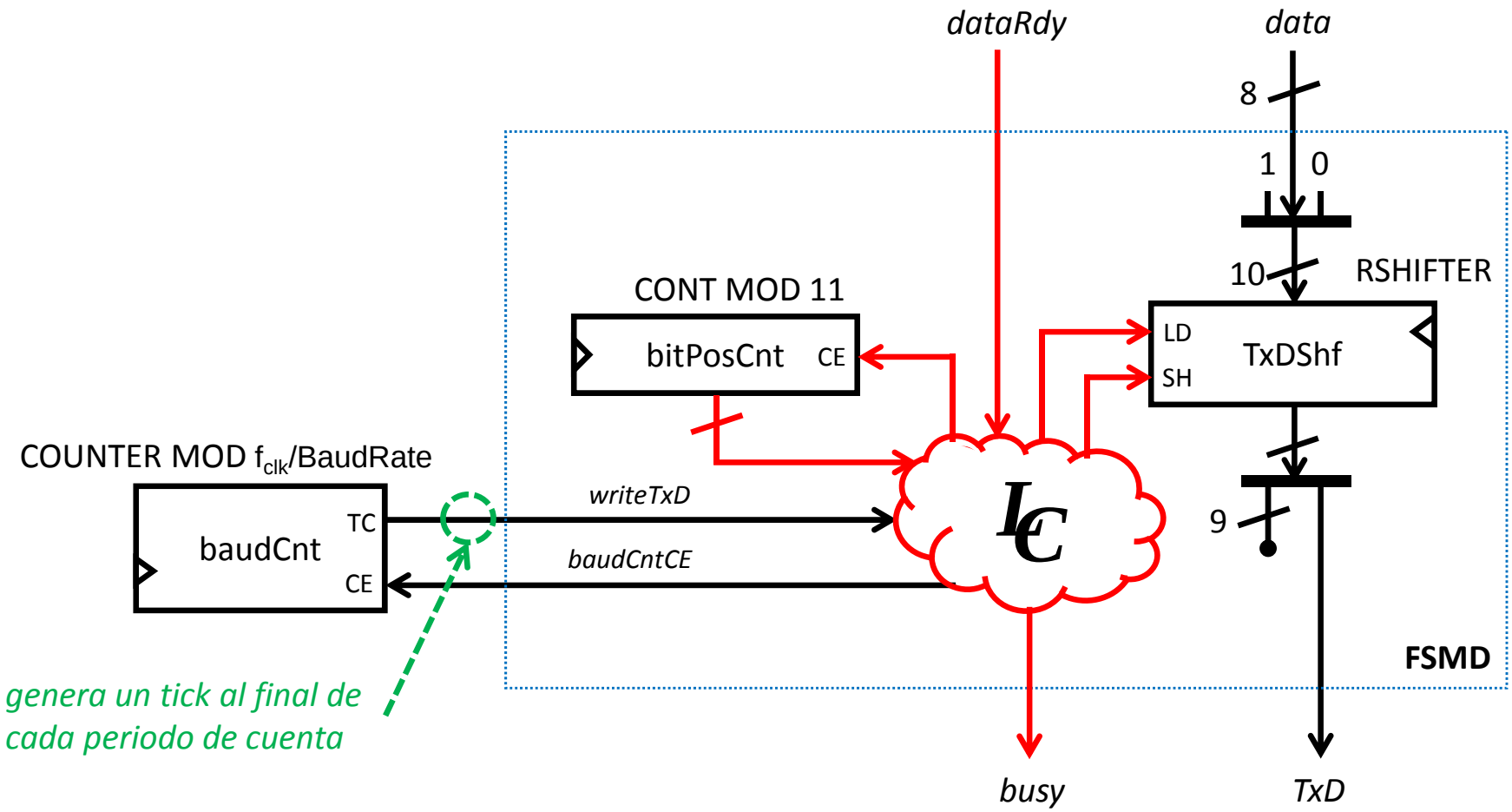
diagrama RTL



J.M. Mendías
2015

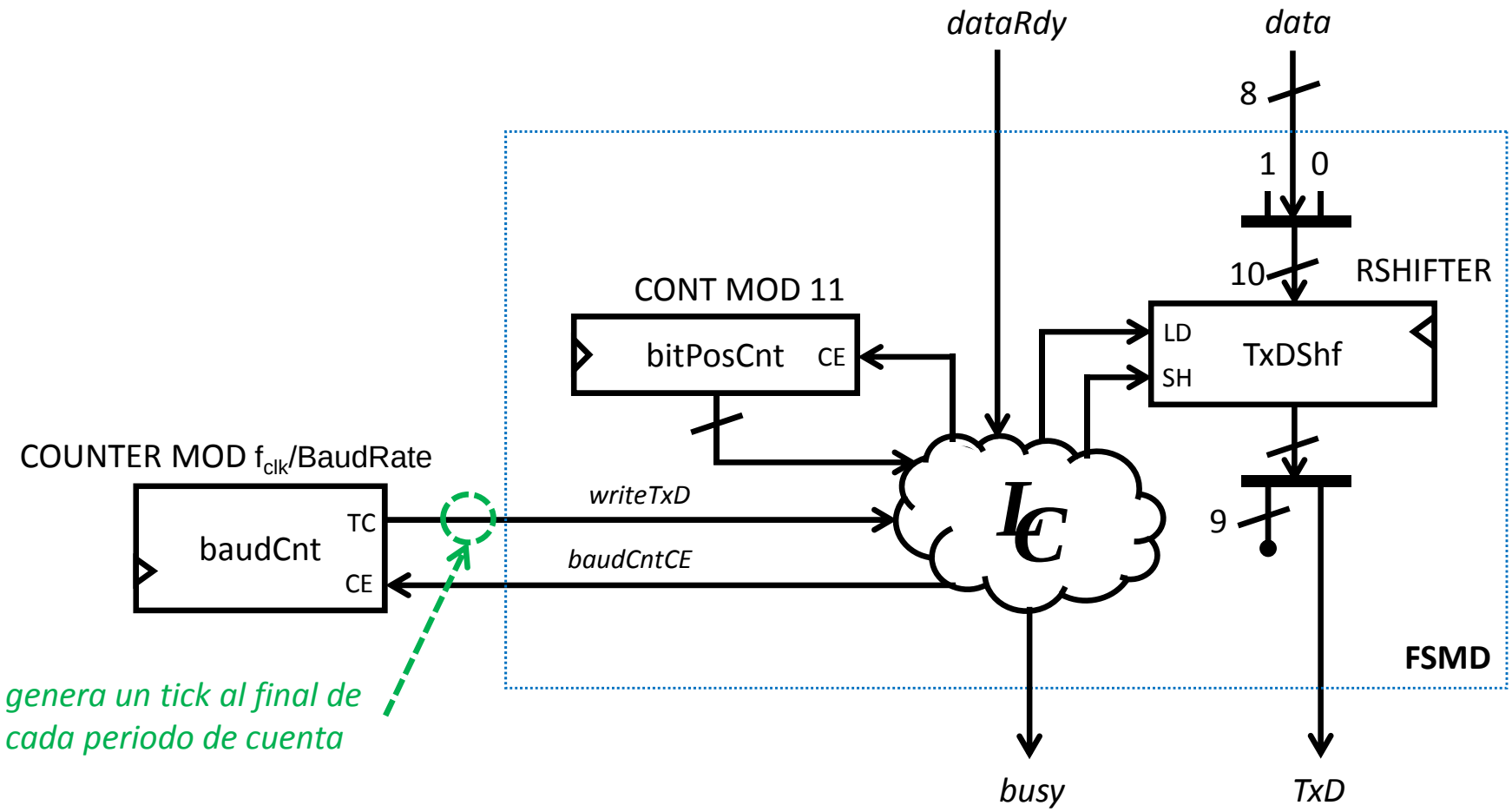
Emisor RS-232

diagrama RTL



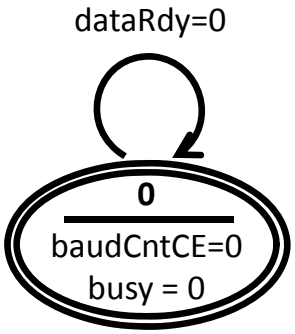
Emisor RS-232

diagrama RTL



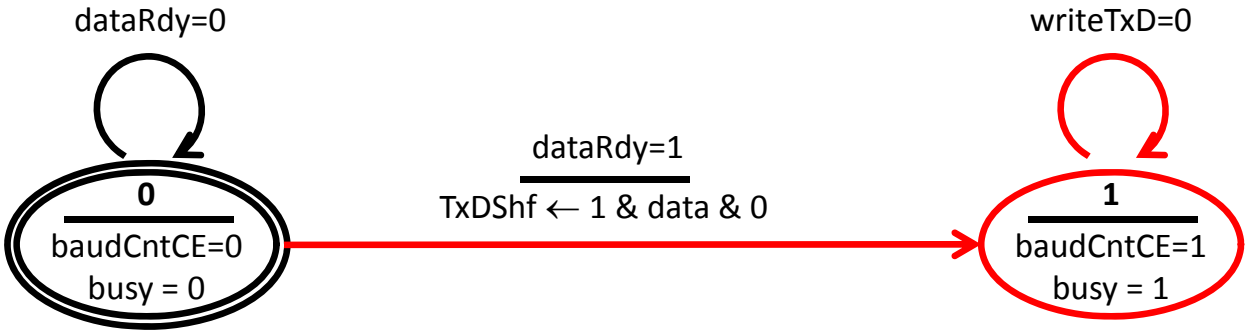
Emisor RS-232

FSMD



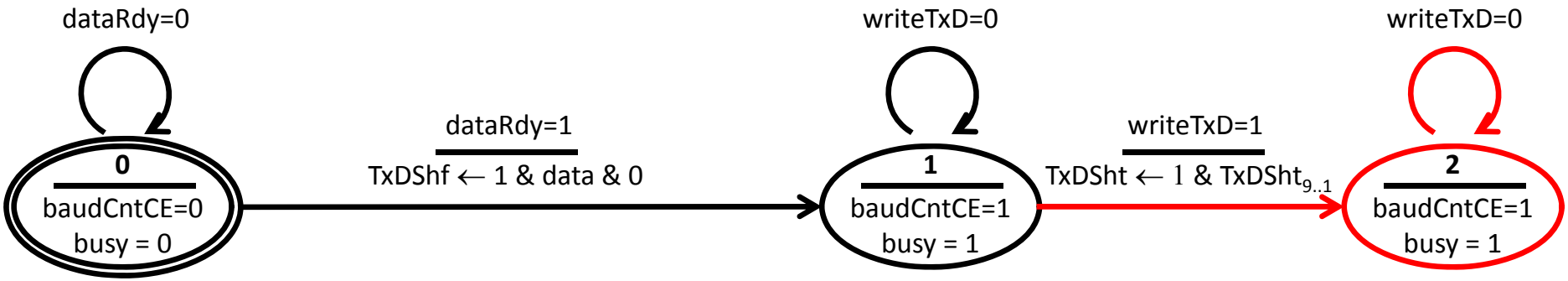
Emisor RS-232

FSMD



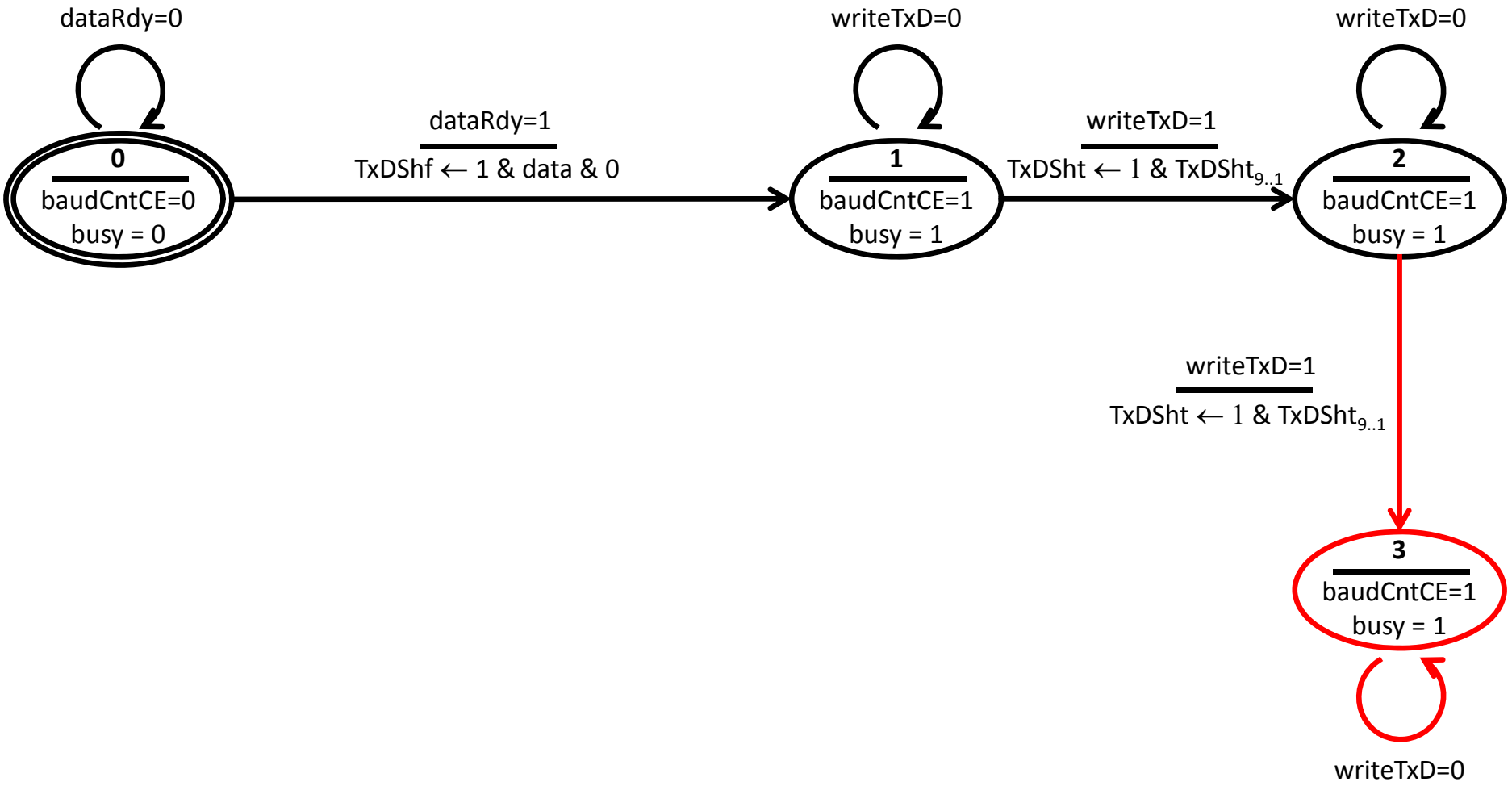
Emisor RS-232

FSMD



Emisor RS-232

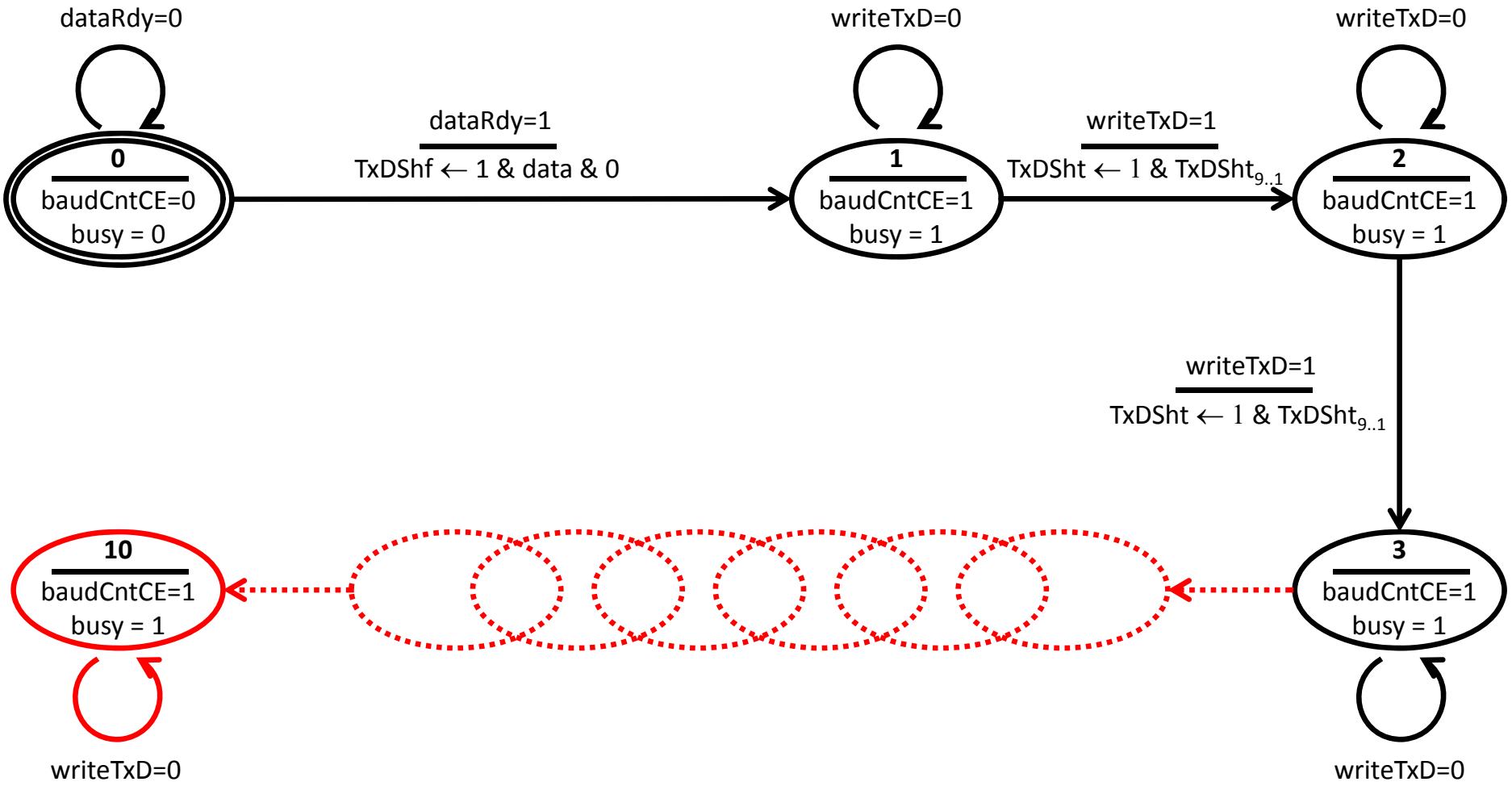
FSMD



J.M. Mendías
2015

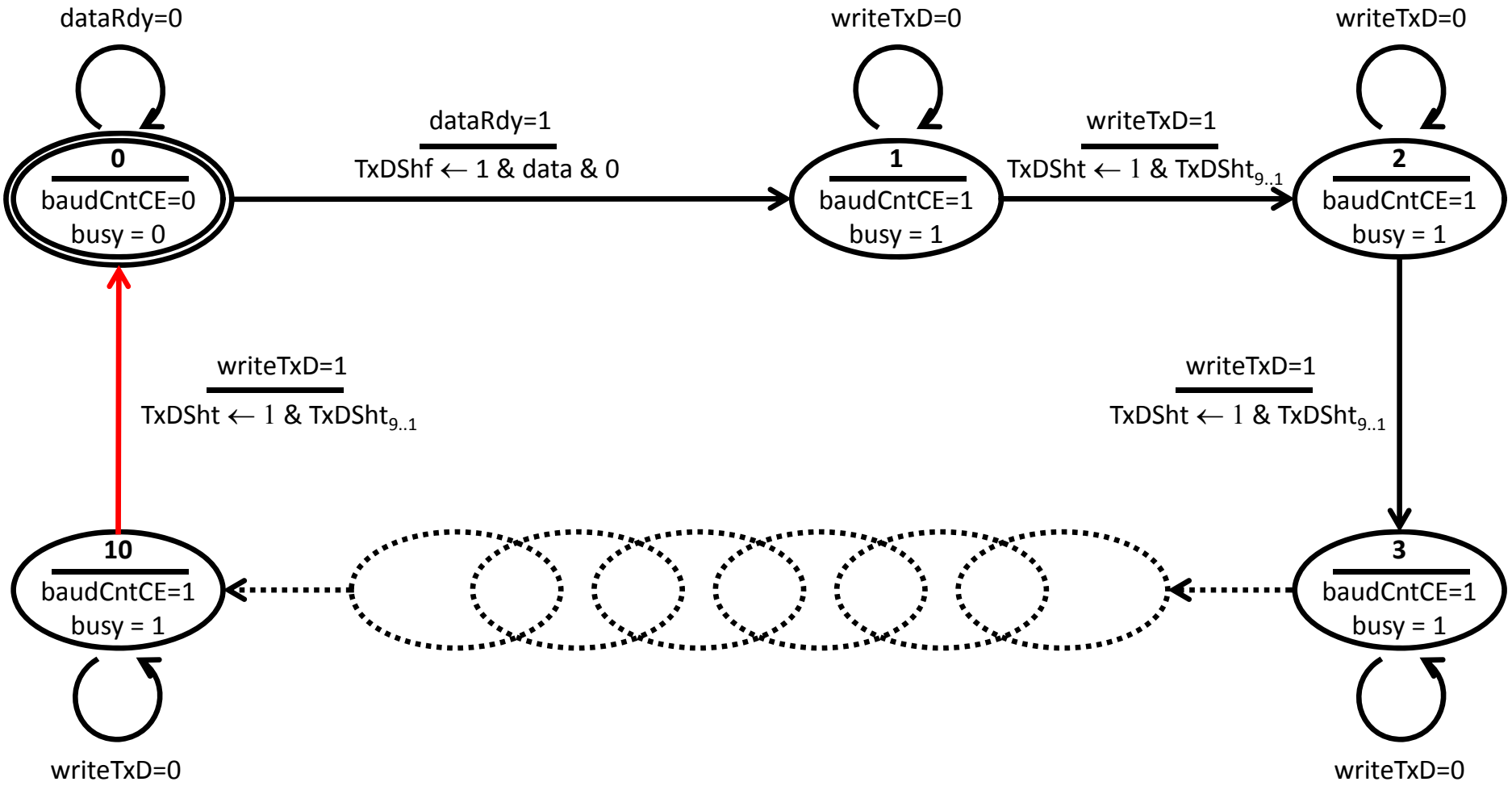
Emisor RS-232

FSMD



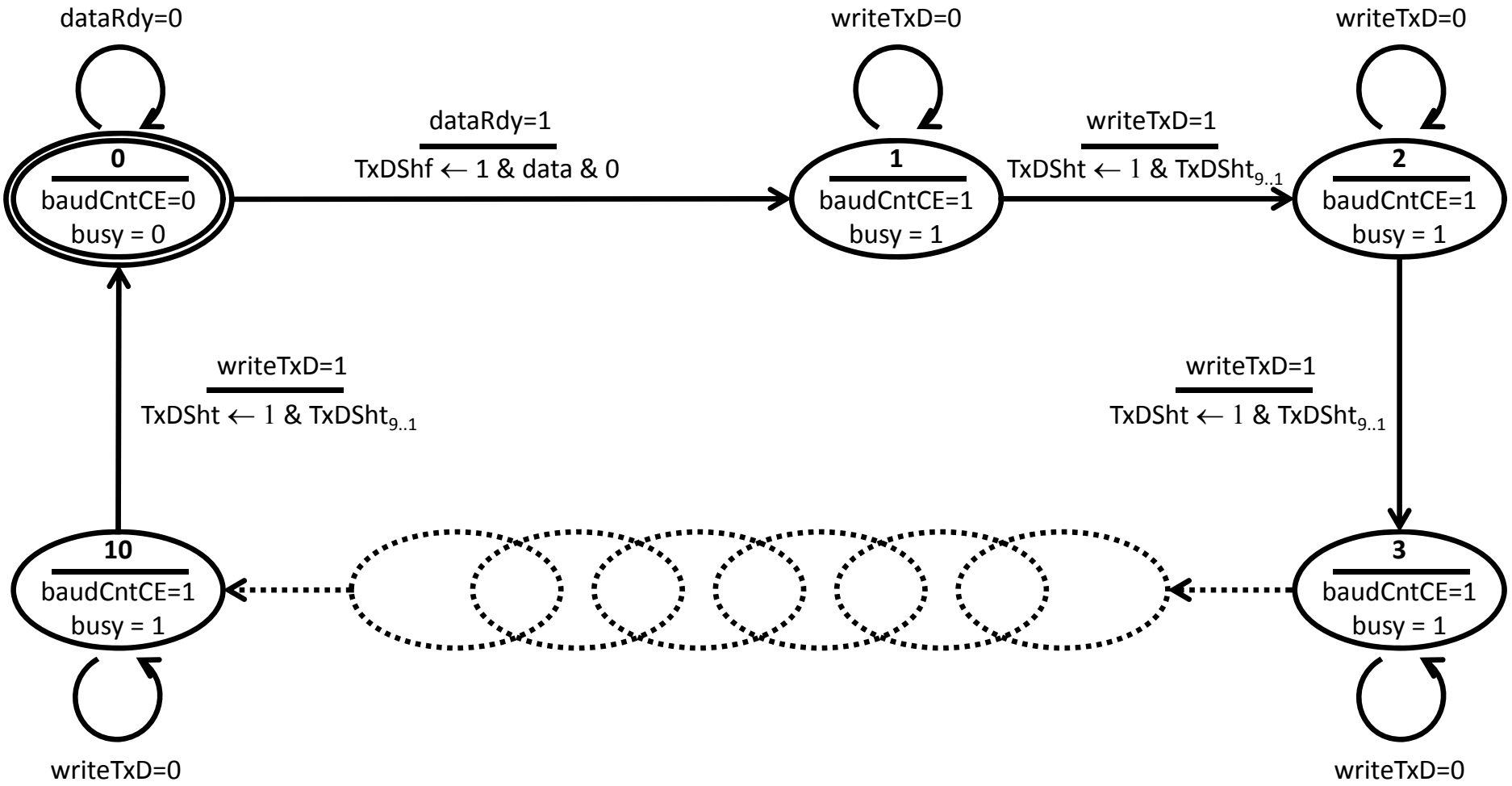
Emisor RS-232

FSMD



Emisor RS-232

FSMD



J.M. Mendías
2015

Emisor RS-232

rs232Transmitter.vhd

```
entity rs232Transmitter is
  generic (
    FREQ     : natural;  -- frecuencia de operacion en KHz
    BAUDRATE : natural   -- velocidad de comunicacion
  );
  port (
    -- host side
    rst_n   : in  std_logic;   -- reset asíncrono del sistema (a baja)
    clk     : in  std_logic;   -- reloj 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
  -- Registros
  signal bitPos : natural range 0 to 10;
  signal TxDShf : std_logic_vector (9 downto 0);
  -- Señales
  signal baudCntCE, writeTxD : std_logic;
begin
  ...
end syn;
```

Emisor RS-232

rs232Transmitter.vhd

```
fsmd :
process (rst_n, clk, bitPos, TxDShf)
begin
  TxD       <= TxDShf(0);
  baudCntCE <= '1';
  busy      <= '1';
  if ... then
    baudCntCE <= '0';
    busy      <= '0';
  end if;
  if rst_n='0' then
    TxDShf <= (others =>'1');
    bitPos <= 0;
  elsif rising_edge(clk) then
    case bitPos is
      when 0 =>
        ...
      when 10 =>
        ...
      when others =>
        ...
    end case;
  end if;
end process;
```

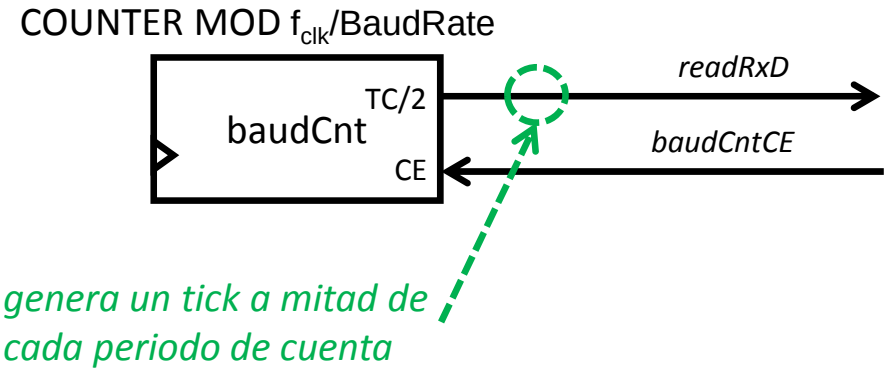
```
baudCnt:
process (rst_n, clk)
  constant numCycles : natural := (FREQ*1000)/BAUDRATE;
  constant maxValue  : natural := numCycles-1;
  variable count     : natural range 0 to maxValue;
begin
  writeTxD <= '0';
  if count=maxValue then
    writeTxD <= '1';
  end if;
  if rst_n='0' then
    count := 0;
  elsif rising_edge(clk) then
    if baudCntCE='0' then
      count := 0;
    else
      ...
    end if;
  end if;
end process;
```

genera un tick al final de cada periodo de cuenta

J.M. Mendías
2015

Receptor RS-232

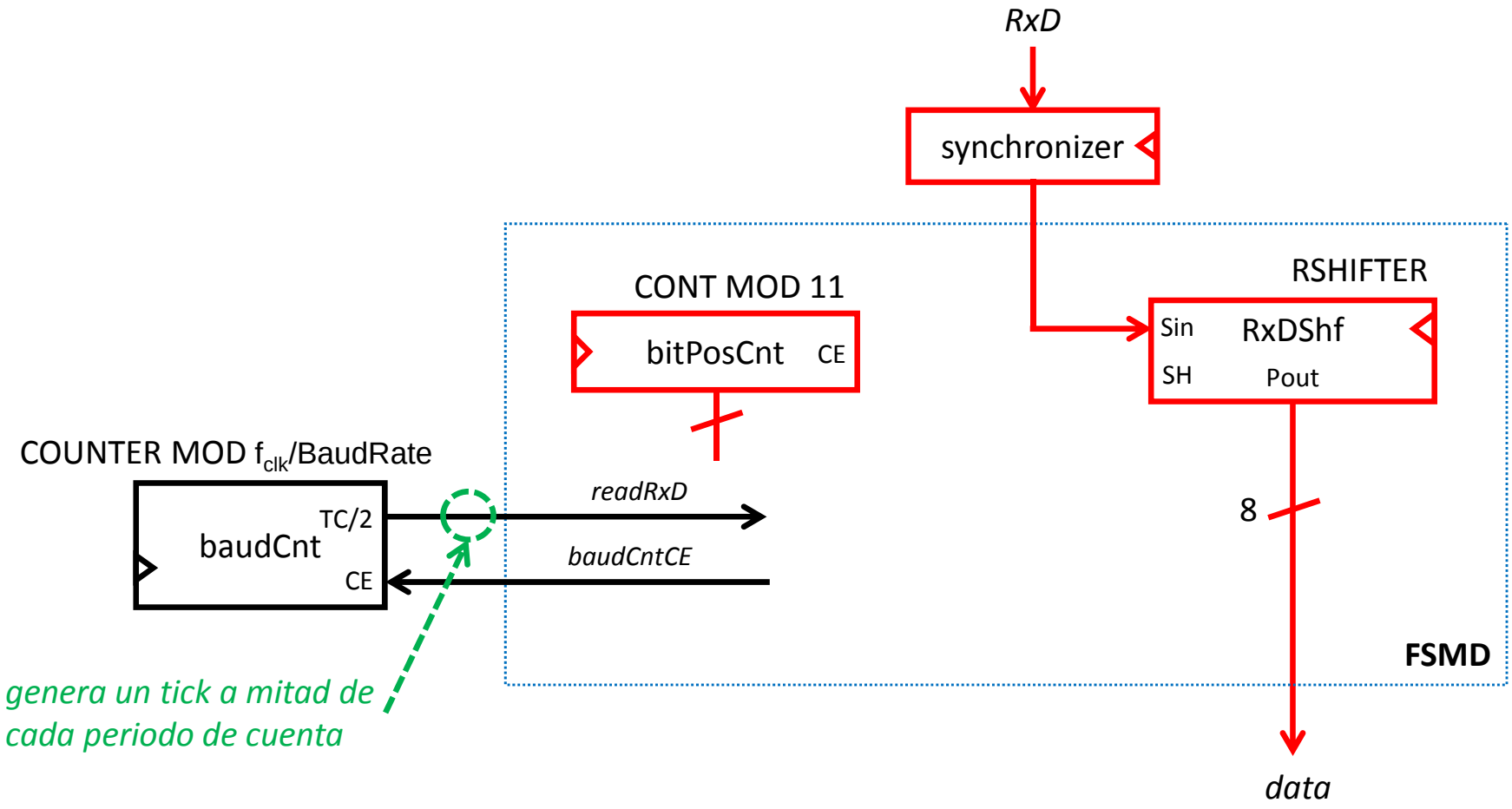
diagrama RTL



J.M. Mendías
2015

Receptor RS-232

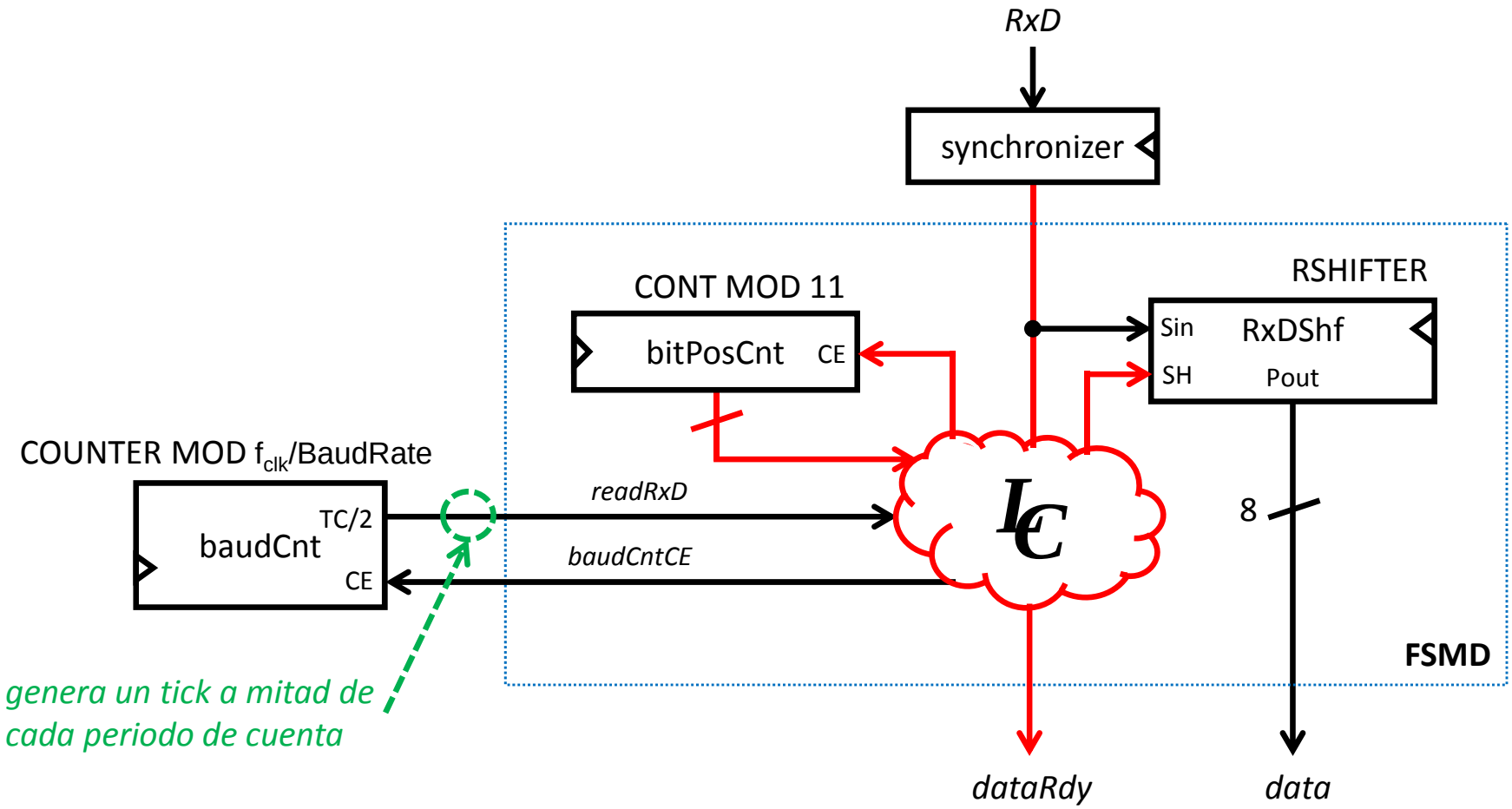
diagrama RTL



J.M. Mendías
2015

Receptor RS-232

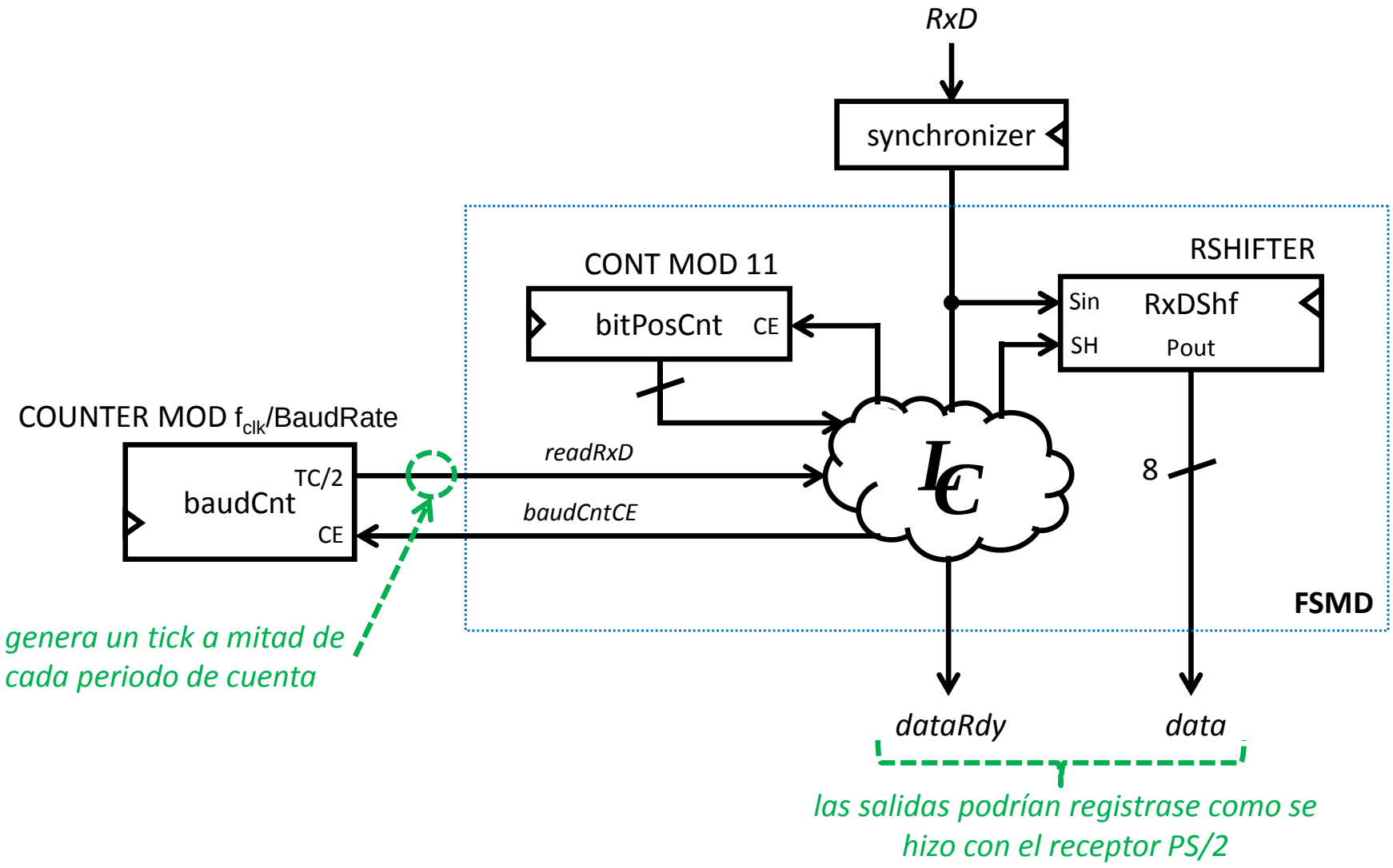
diagrama RTL



J.M. Mendías
2015

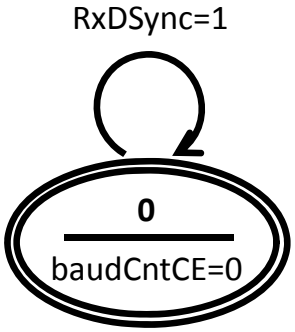
Receptor RS-232

diagrama RTL



Receptor RS-232

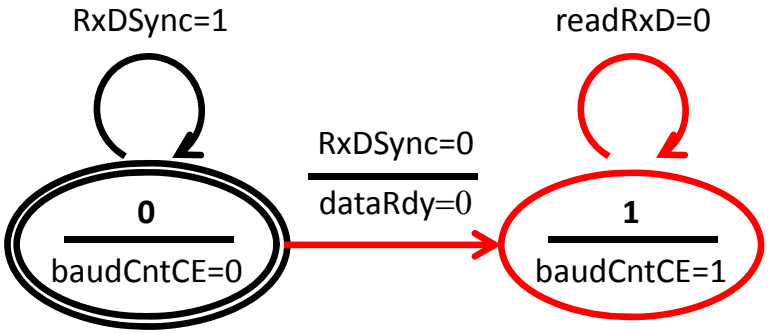
FSMD



J.M. Mendías
2015

Receptor RS-232

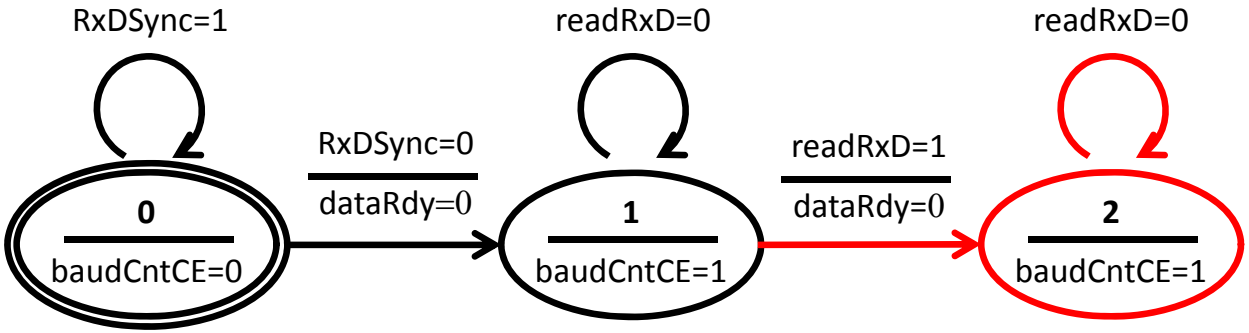
FSMD



J.M. Mendías
2015

Receptor RS-232

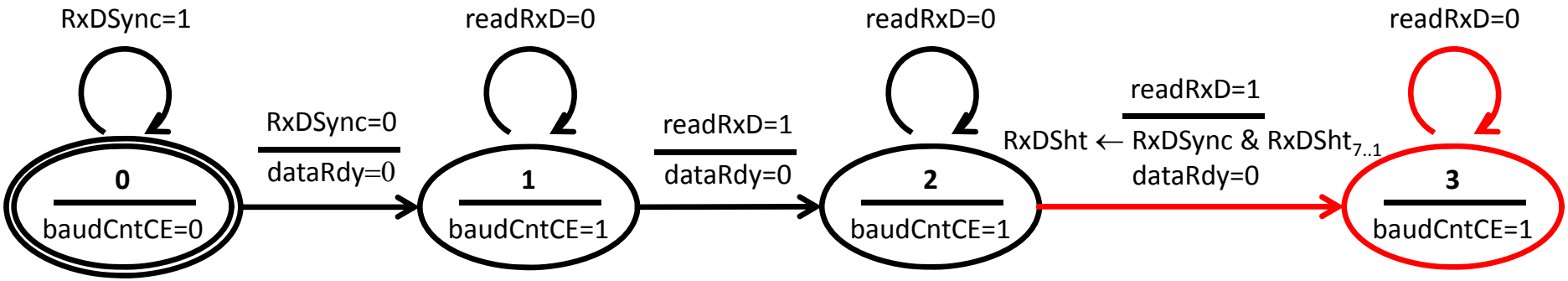
FSMD



J.M. Mendías
2015

Receptor RS-232

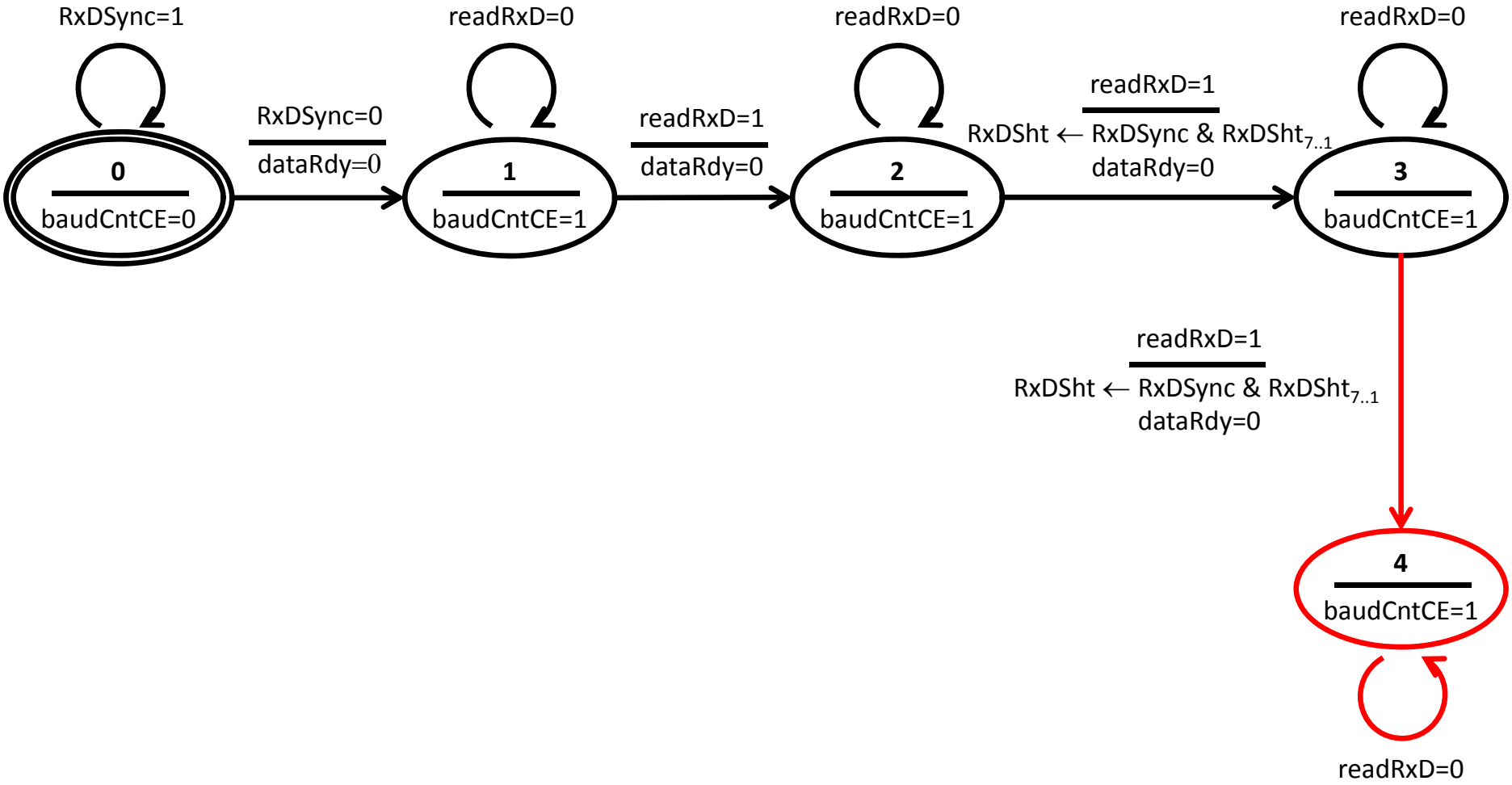
FSMD



J.M. Mendías
2015

Receptor RS-232

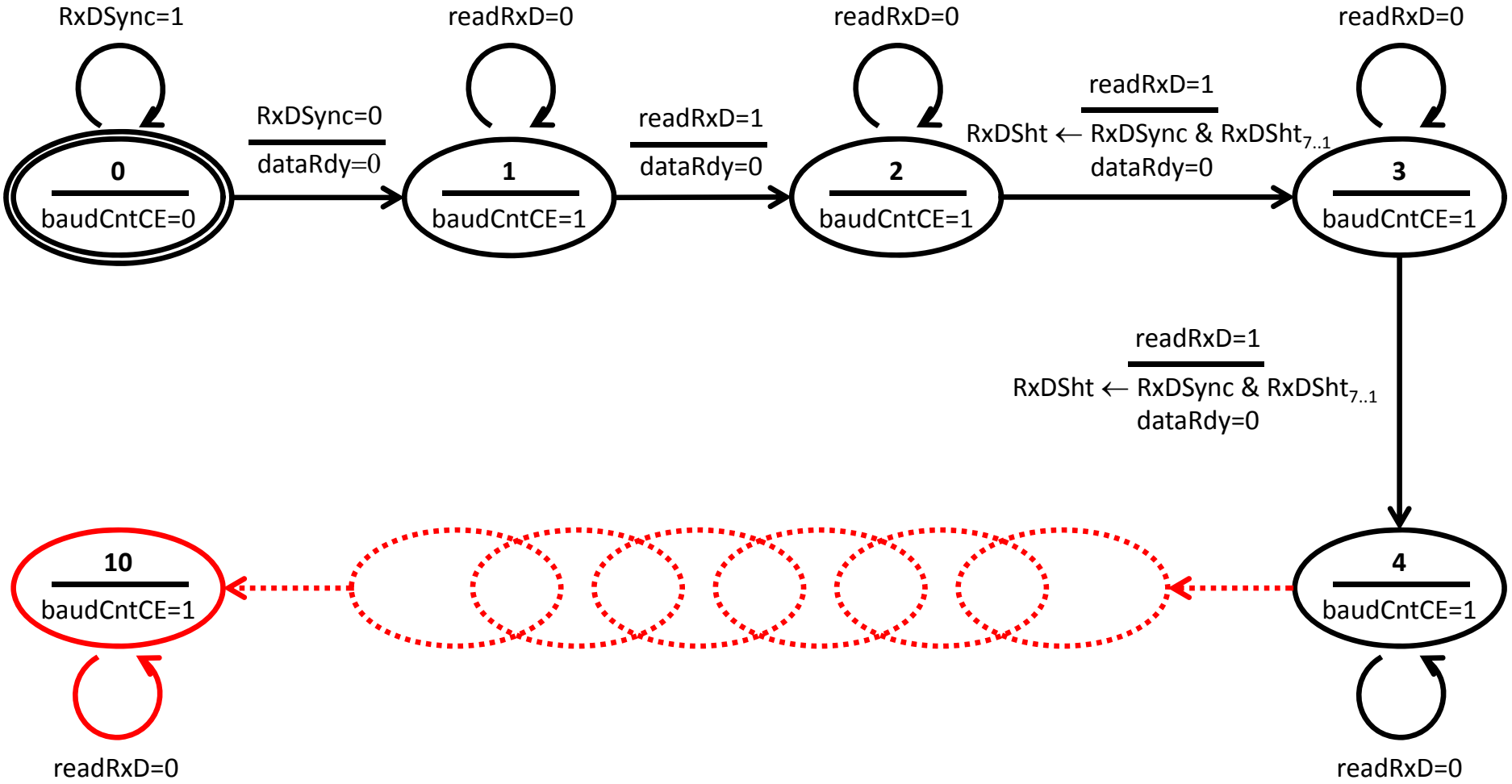
FSMD



J.M. Mendías
2015

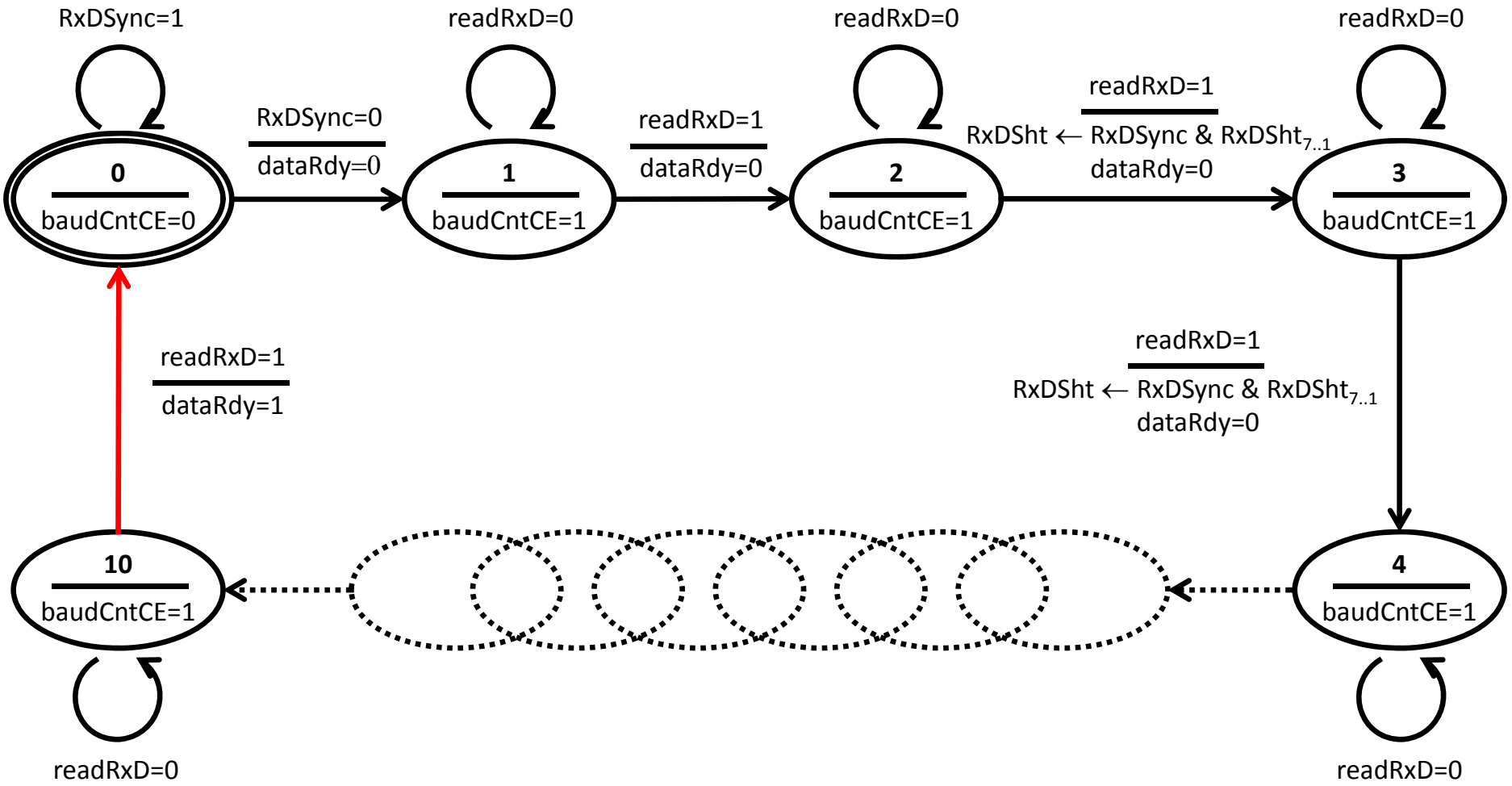
Receptor RS-232

FSMD



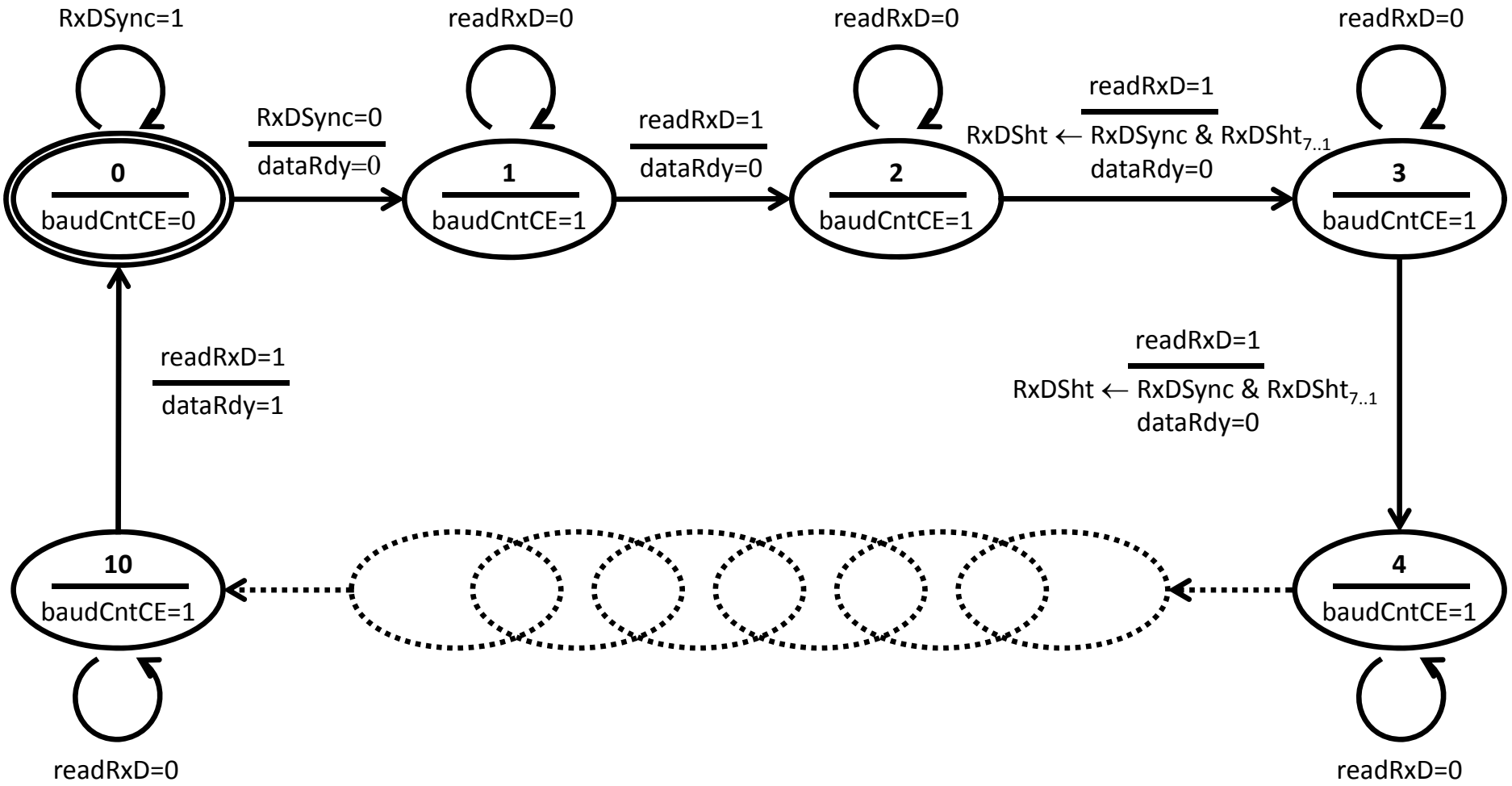
Receptor RS-232

FSMD



Receptor RS-232

FSMD



J.M. Mendías
2015

Receptor RS-232

rs232Receiver.vhd

```
entity rs232Receiver is
  generic (
    FREQ     : natural;  -- frecuencia de operacion en KHz
    BAUDRATE : natural   -- velocidad de comunicacion
  );
  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
    -- RS232 side
    RxD     : in  std_logic    -- entrada de datos serie del interfaz RS-232
  );
end rs232Receiver;

architecture syn of rs232Transmitter is
  -- Registros
  signal bitPos : natural range 0 to 10;
  signal RxDShf : std_logic_vector (7 downto 0);
  -- Señales
  signal RxDSync : std_logic;
  signal readRxD, baudCntCE : std_logic;
begin
  ...
end syn;
```

J.M. Mendías
2015

Receptor RS-232

rs232Receiver.vhd

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

```
baudCnt:
process (rst_n, clk)
  ...
begin
  readRxD <= '0';
  if count=maxValue/2 then
    readRxD <= '1';
  end if;
  if rst_n='0' then
    ...;
  elsif rising_edge(clk) then
    ...;
  end if;
end process;

RxDSynchronizer : synchronizer
  ...
```

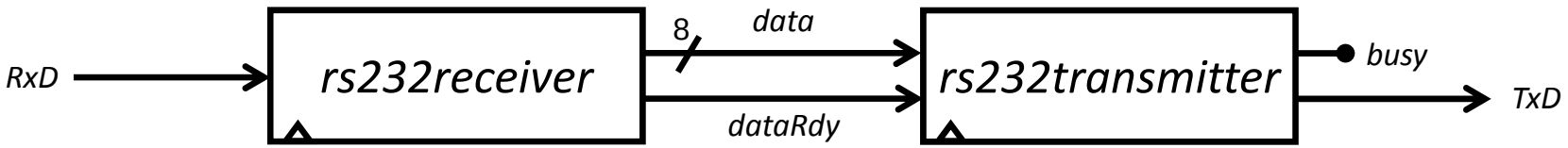
genera un tick a mitad de cada periodo de cuenta

J.M. Mendías
2015

Emisor/receptor RS-232

aplicación al diseño

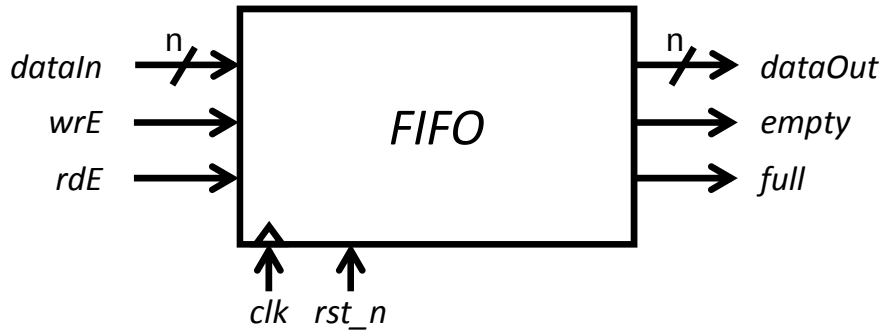
- Para realizar un loopback bastaría:



FIFO
presentación

- Cuando un canal de comunicación tiene un cierto ancho de banda nominal, pero los datos no se transmiten con una cadencia regular:
  - Suele ser necesario 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.

- 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.

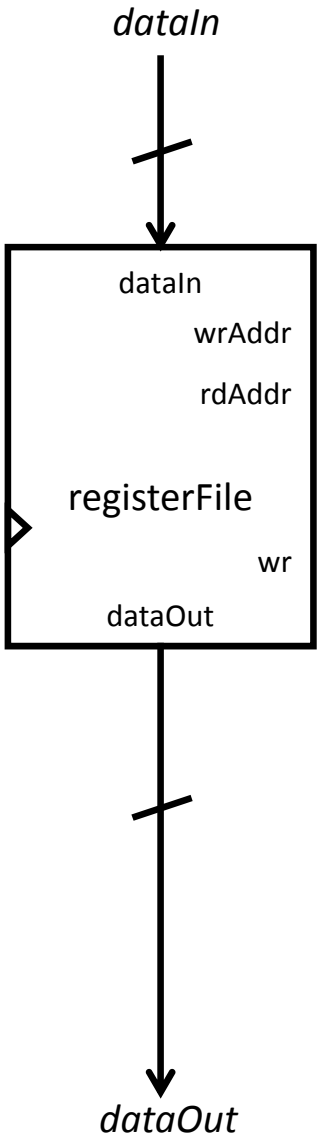


FIFO
diagrama RTL (i)

- 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.
  - 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.

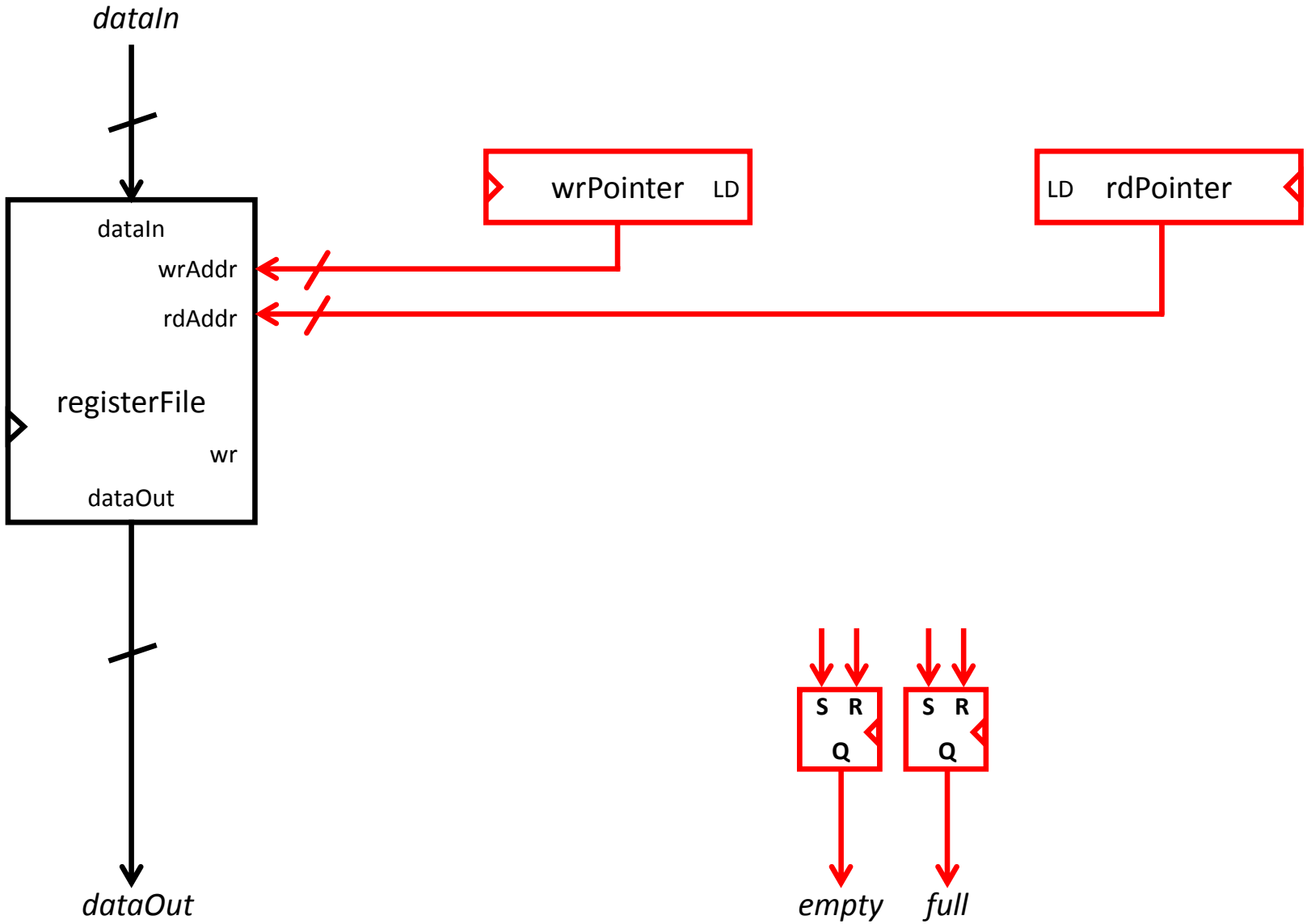
FIFO

diagrama RTL (ii)



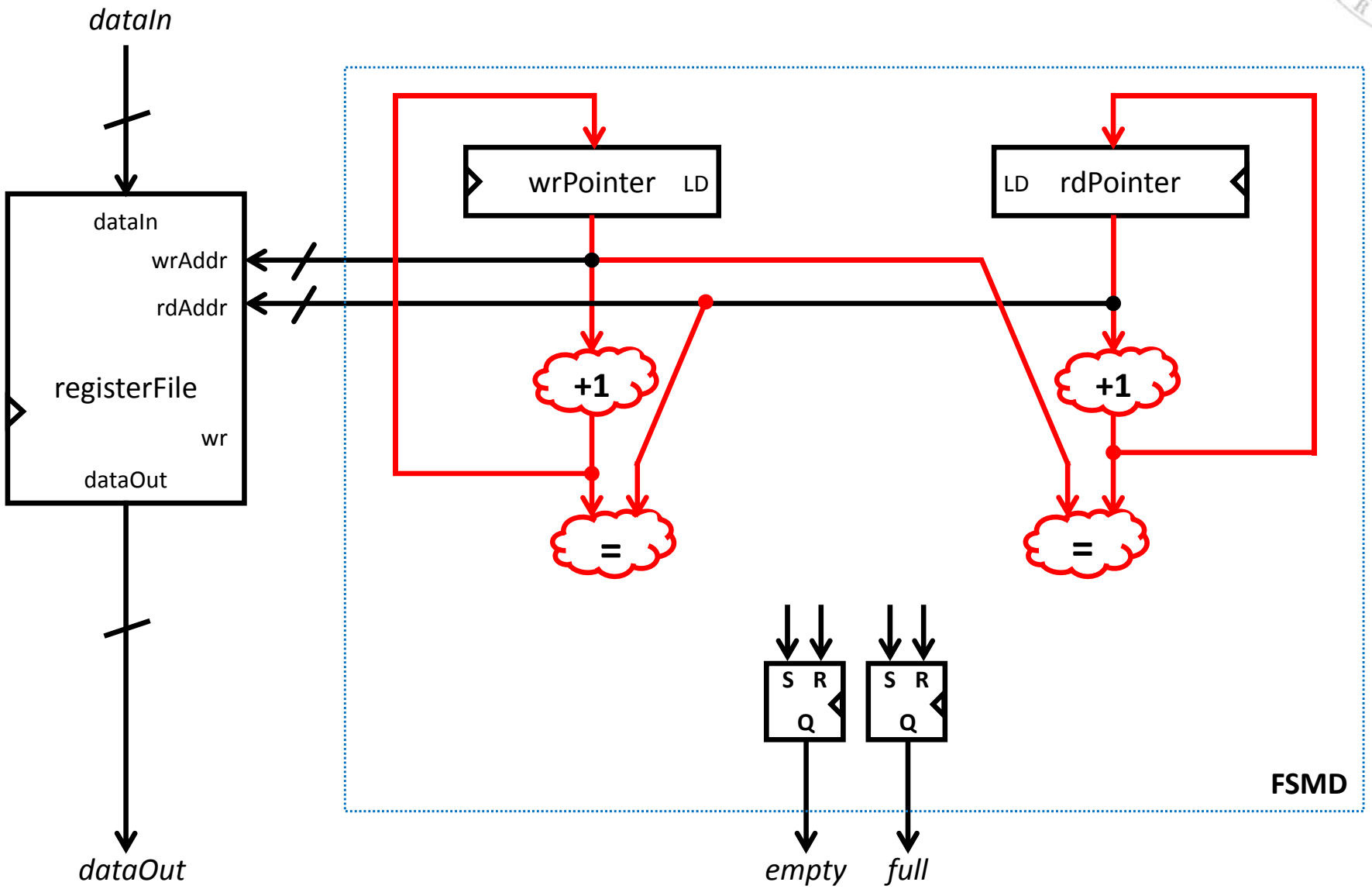
FIFO

diagrama RTL (ii)



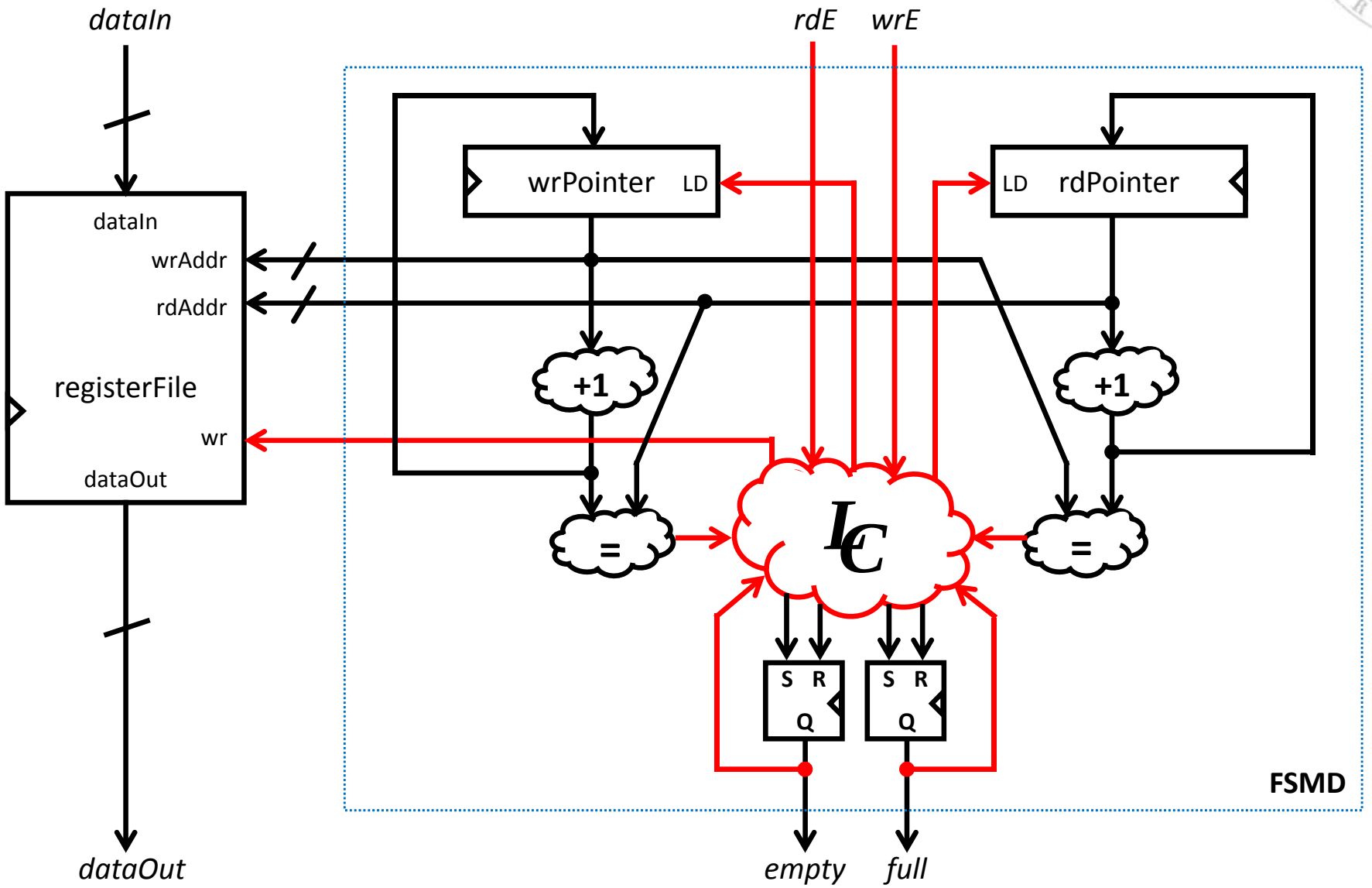
FIFO

diagrama RTL (ii)



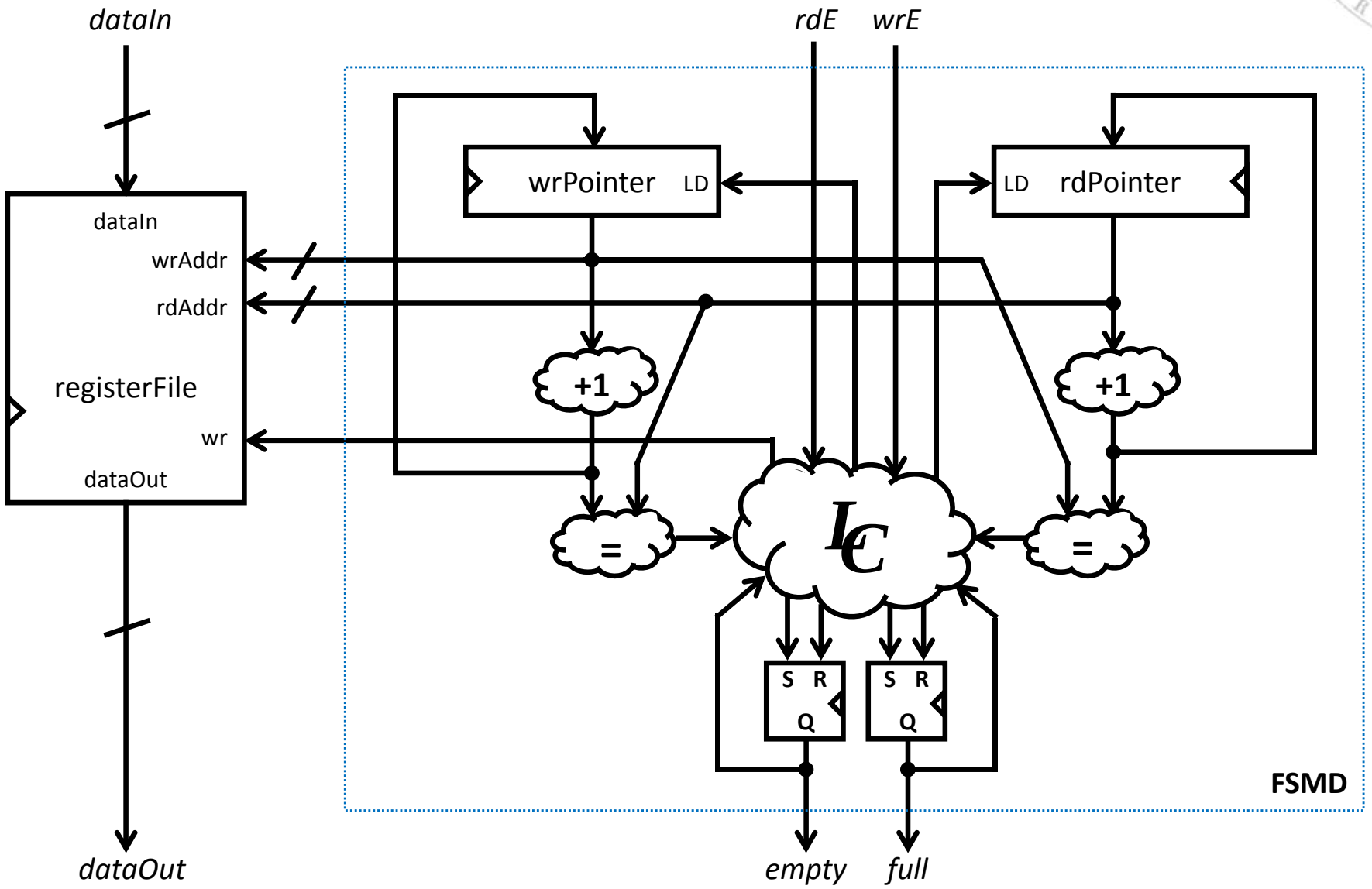
FIFO

diagrama RTL (ii)



FIFO

diagrama RTL (ii)



FIFO

fifo.vhd

```
entity fifo is
  generic (
    WIDTH : natural;   -- anchura de la palabra de fifo
    DEPTH : natural    -- numero de palabras en fifo
  );
  port (
    rst_n   : in  std_logic;   -- reset asíncrono del sistema (a baja)
    clk     : in  std_logic;   -- reloj 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
    full    : out std_logic;   -- indicador de fifo llena
    empty   : out std_logic    -- indicador de fifo vacia
  );
end fifo;

architecture syn of fifo is

  constant maxValue : natural := DEPTH-1;
  type   regFileType is array (0 to maxValue) of std_logic_vector(WIDTH-1 downto 0);
  -- Registros
  signal regFile : regFileType;
  signal wrPointer, rdPointer : natural range 0 to maxValue;
  signal isFull : std_logic;
  signal isEmpty : std_logic;
  -- Señales
  signal nextWrPointer, nextRdPointer : natural range 0 to maxValue;
  signal rdFifo, wrFifo : std_logic;

begin
  ...
end syn;
```

FIFO

fifo.vhd

```
registerFile :
process (rst_n, clk, rdPointer, regFile)
begin
  dataOut <= ...;
  if rst_n='0' then
    regFile <= (others => (others => '0'));
  elsif rising_edge(clk) then
    ...
  end if;
end process;

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

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

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

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

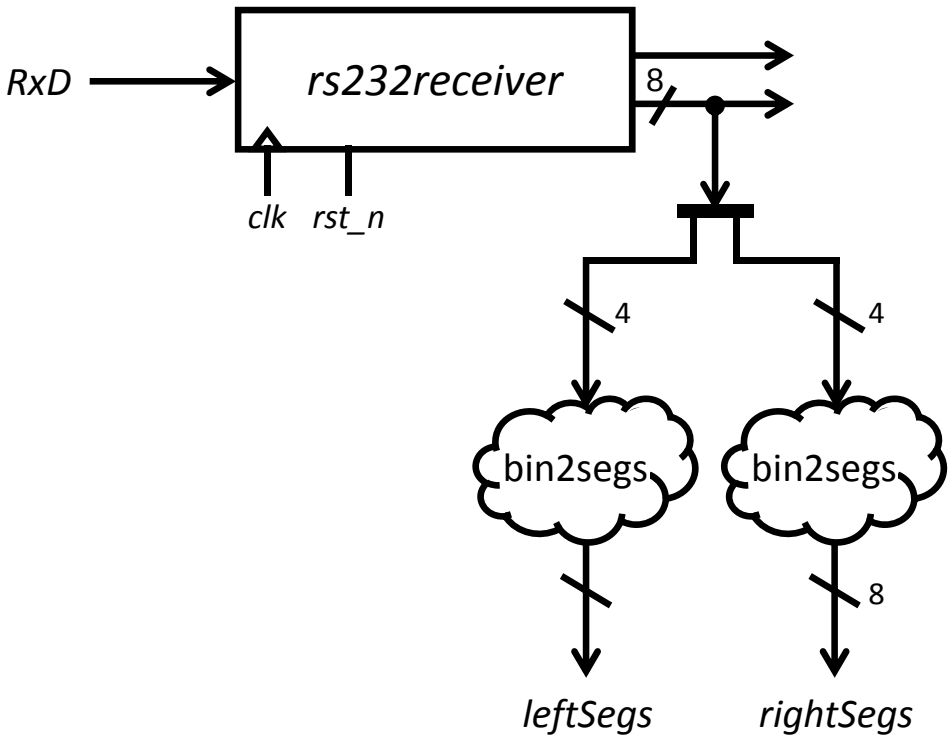
Diseño principal

- Para testar el par emisor-receptor realizaremos un diseño que retransmita lo que reciba por el bus RS-232 previo paso por una FIFO.
  - Conectaremos la FPGA a través de un cable serie al PC, sobre el que correremos un emulador de terminal (Termite) para enviar/recibir información
  - El emulador deberá configurarse:
    - 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.

- Para facilitar la prueba, además:
  - El estado de la FIFO vacía/llena se mostrará (E/F) en el display 7-segs de la placa XST.
  - La salida del receptor estará conectada a los displays 7-segs de la placa XST.
  - El transmisor podrá inhabilitarse desactivando el DIPSW-8 de la placa XST.
    - El receptor seguirá recibiendo datos y almacenándolos en la FIFO hasta llenarla.

Diseño principal

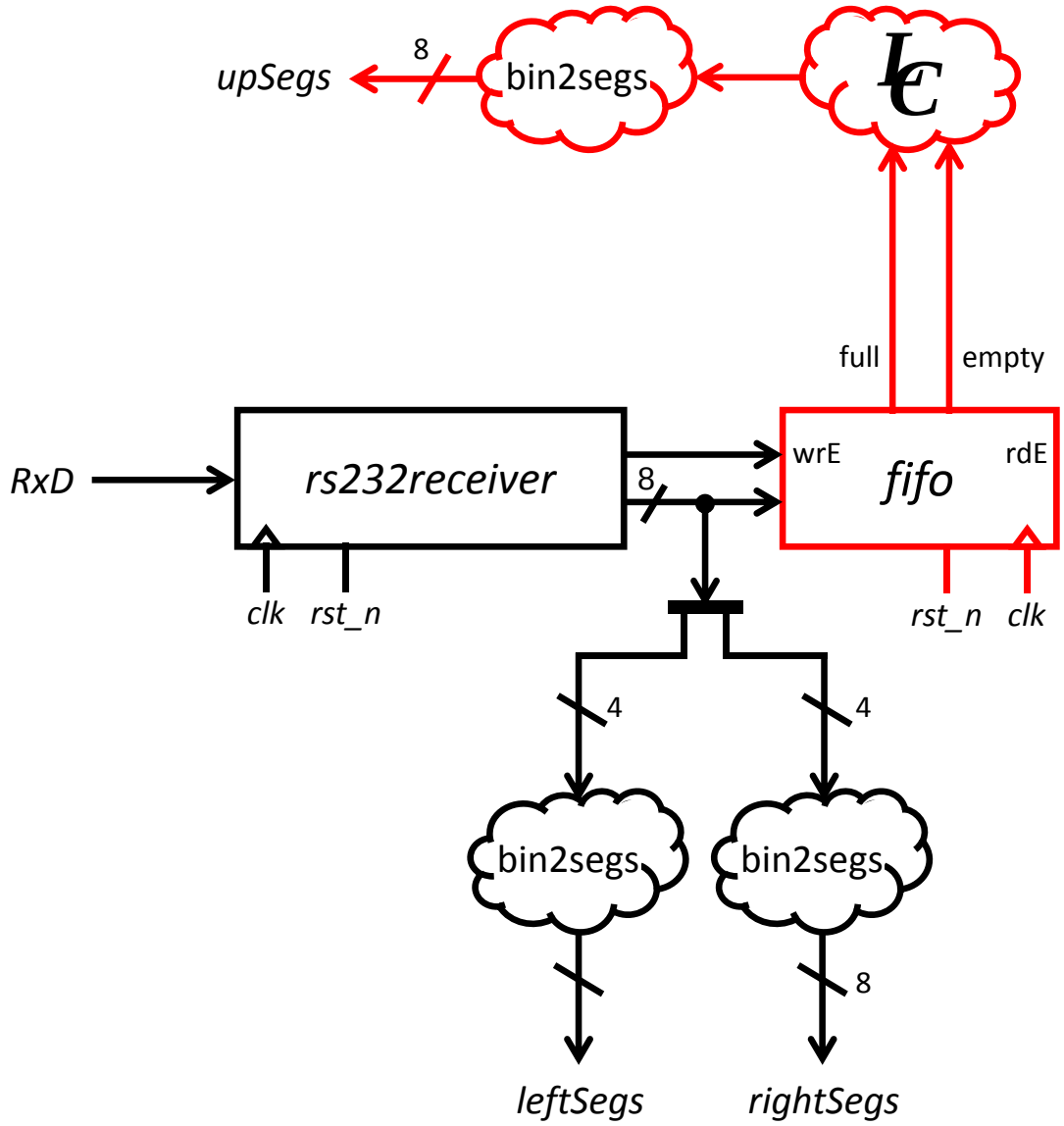
diagrama RTL



J.M. Mendías
2015

Diseño principal

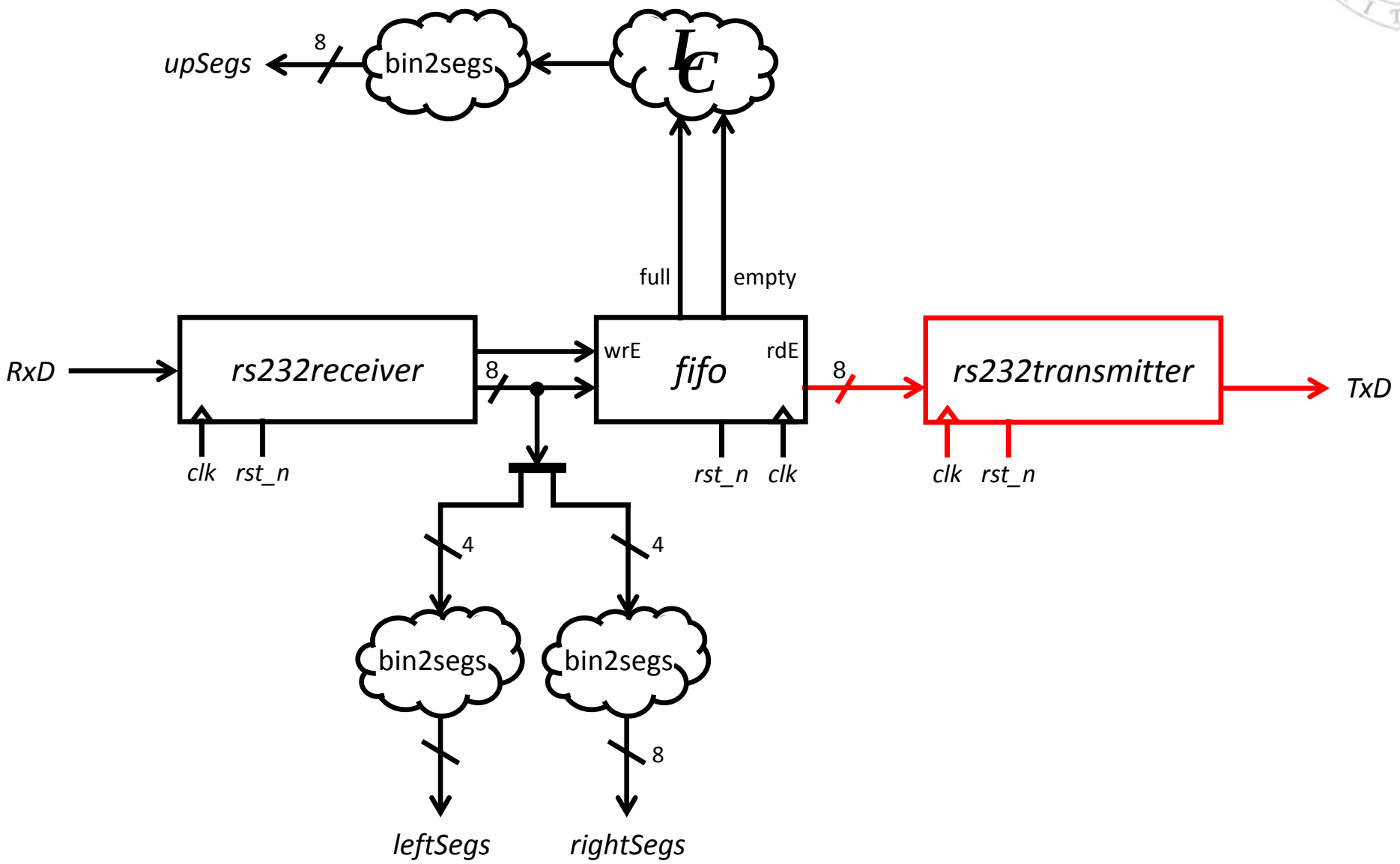
diagrama RTL



J.M. Mendías
2015

Diseño principal

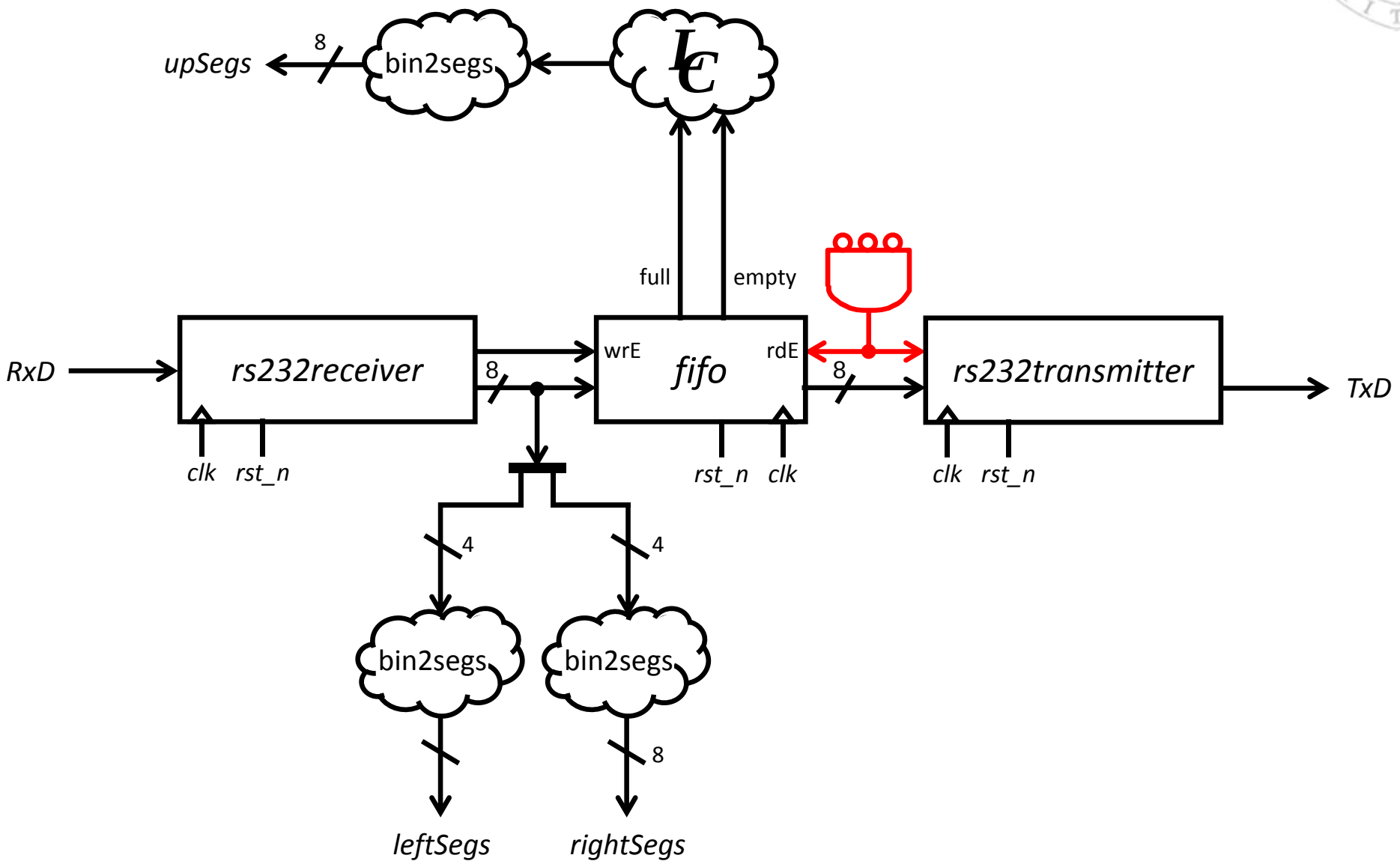
diagrama RTL



J.M. Mendías
2015

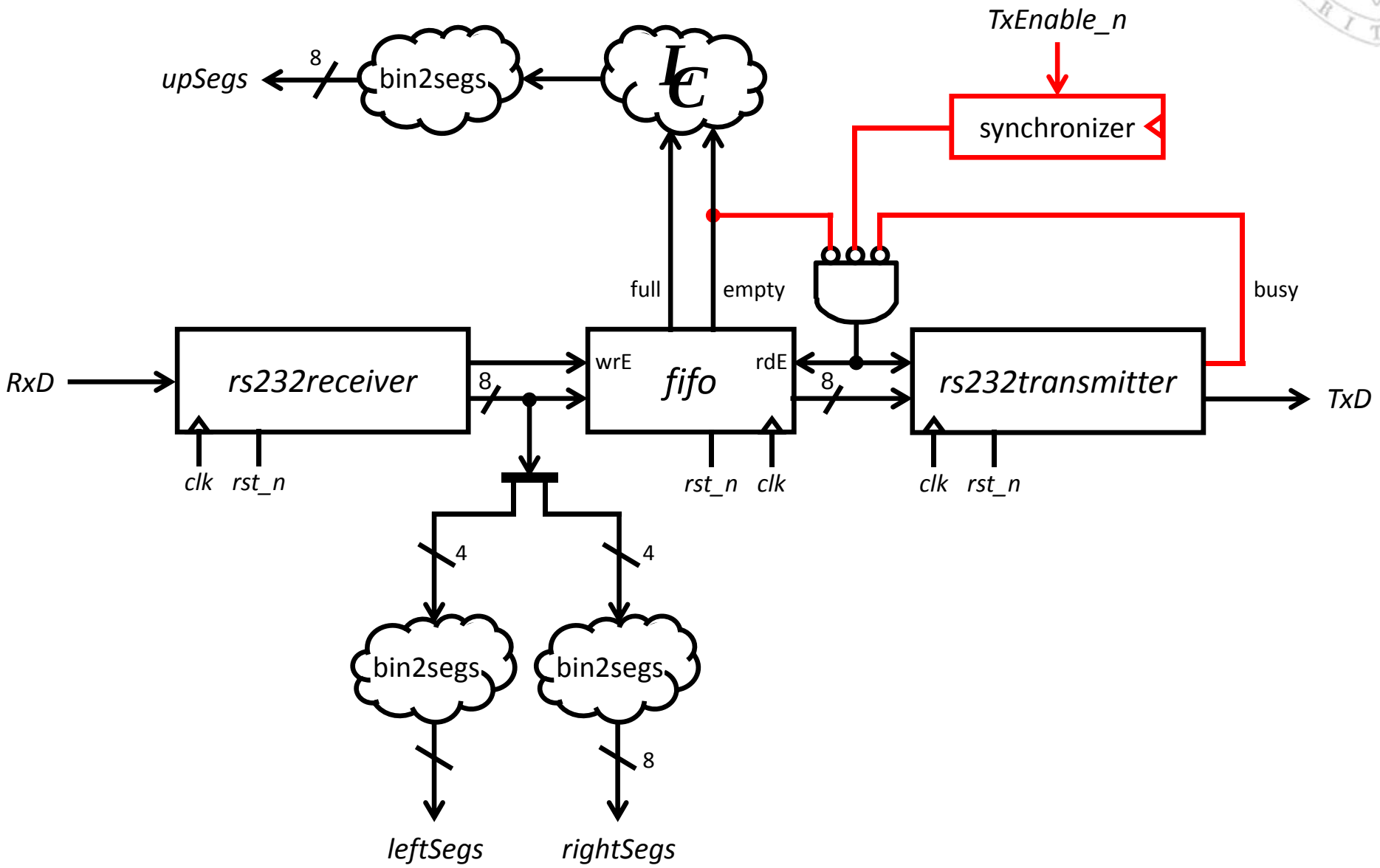
Diseño principal

diagrama RTL



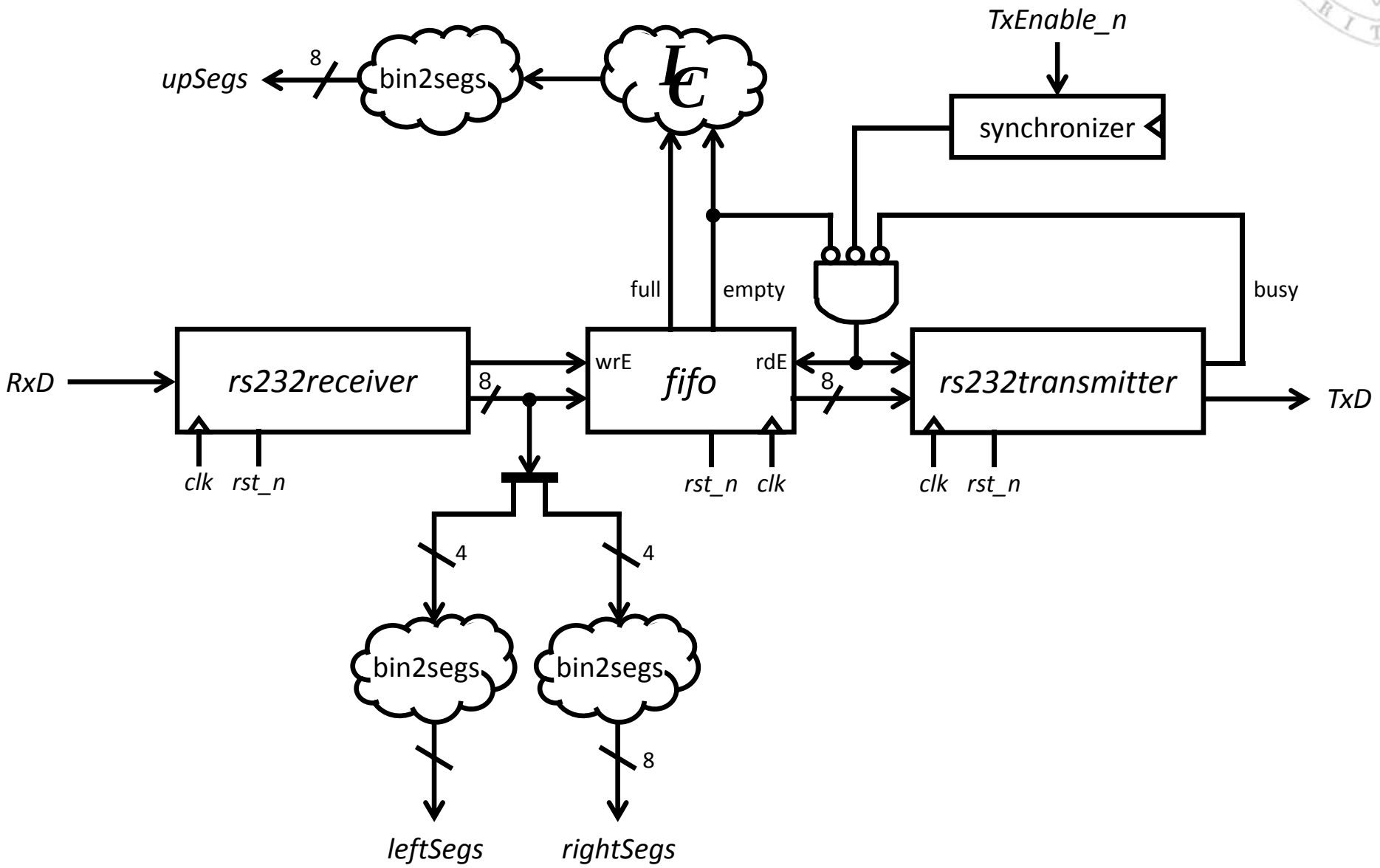
Diseño principal

diagrama RTL



Diseño principal

diagrama RTL



J.M. Mendías
2015

Tareas

1. Crear el proyecto **`lab5`** en el directorio **DAS**
2. Descargar de la Web en el directorio **`common`** los ficheros:
   - **`fifo.vhd`**, **`rs232receiver.vhd`** y **`rs232transmitter.vhd`**
3. Descargar de la Web en el directorio **`lab5`** los ficheros:
   - **`lab5.vhd`** y **`lab5.ucf`**
4. Completar el fichero **`common.vhd`** con la declaración de los tres nuevos componente reusables.
5. Completar el código omitido en los ficheros:
   - **`fifo.vhd`**, **`rs232receiver.vhd`** y **`rs232transmitter.vhd`**
6. Añadir al proyecto los ficheros:
   - **`common.vhd`**, **`bin2segs.vhd`**, **`synchronizer.vhd`**, **`rs232receiver.vhd`**, **`rs232transmitter.vhd`**, **`lab5.vhd`** y **`lab5.ucf`**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Conectar el cable RS-232 a la placa y encenderla.
9. Descargar el fichero **`lab5.bit`**
10. Arrancar un emulador de terminal (Termite) en el PC.

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/