PDS-FPGAs

Práctica P5

Escritura y lectura de registros de configuración

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Índice

1. Objetivos

Los objetivos de esta práctica son:

1. El objetivo de esta práctica es implementar el bloque de comunicación entre el PC y el dispositivo FPGA, que nos permitirá escribir y leer desde el PC, a través de un puerto serie RS232, los registros de configuración del modulador de FM/AM. El subsistema implementado deberá funcionar a una frecuencia de reloj de 25 MHz y se verificará utilizando la tarjeta DE2-115 escribiendo y leyendo los registros desde un PC.

2. Entender la estrategia de descomposición de máquinas de estados para simplificar el diseño del subsistema de control.

3. Saber usar la herramienta “Signal Tap Logic Analyzer” para depurar los módulos implementados.

2. Descripción del sistema

A continuación, se muestra el esquema que se utilizará en esta práctica para validar el subsistema de comunicación con el PC. Está compuesto de los siguientes bloques:

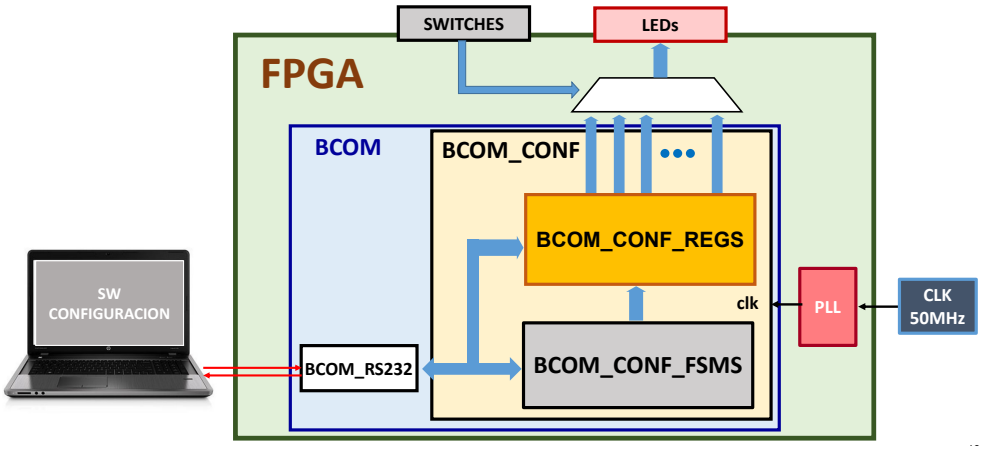
- *bcom_rs232*: este módulo (cuyo código HDL se proporciona) se encarga de realizar la capa física de la comunicación RS232.

- *bcom_conf_regs*: módulo que incluye los registros de configuración del modulador AM/FM y los registros necesarios para implementar la comunicación.

- *bcom_conf_fsms*: bloque encargado de gestionar la comunicación.

- PLL: Phase Locked-Loop del dispositivo Cyclone IV configurado para generar un reloj de f_{clk}=230 400 Hz a partir del reloj de 50 MHz de la tarjeta DE2-115.
- Interruptores y LEDs para visualizar los valores almacenados y facilitar la depuración.

Con el fin de facilitar la verificación de los diferentes módulos y su posterior uso en la siguiente práctica, los bloques *bcom_conf_regs* y *bcom_conf_fsms* están conectados entre sí formando *bcom_conf* y, este, está conectado a *bcom_rs232* en *bcom*.



2.1. Registros de configuración del modulador AM/FM

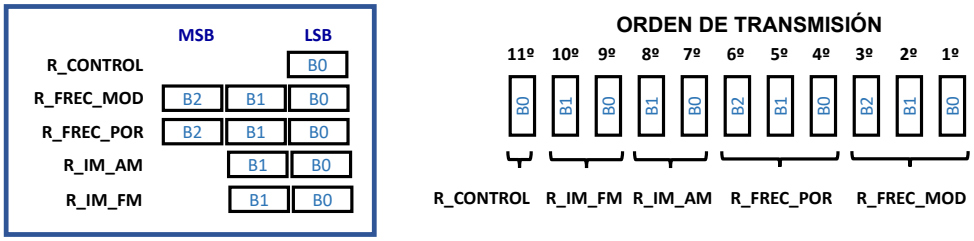
El funcionamiento del modulador se controla configurando 5 registros. Sus nombres, tamaño y la función que desempeñan se exponen en la siguiente tabla.

Tabla 1: Registros de configuración del modulador

REGISTRO	TAMAÑO	FUNCIÓN
R_CONTROL	1 byte	Registro de control
R_FREC_MOD	3 bytes	Paso del DDS para generar la frecuencia de las señales de test
R_FREC_POR	3 bytes	Paso del DDS para generar la frecuencia portadora
R_IM_AM	2 bytes	Índice de modulación de AM
R_IM_FM	2 bytes	Índice de modulación de FM

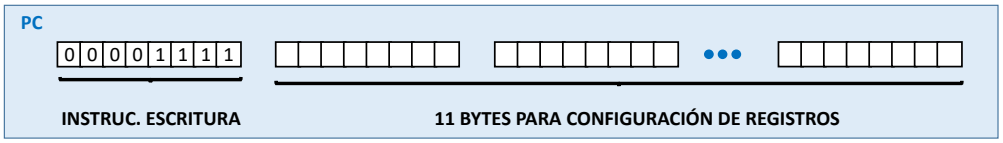
2.2. Procedimiento de comunicación

El puerto serie nos permite transmitir datos de tamaño byte. Por tanto, los datos de configuración de cada registro se tienen que descomponer en bytes para transmitirlos de forma independiente, respetando el orden de transmisión indicado en la siguiente figura.

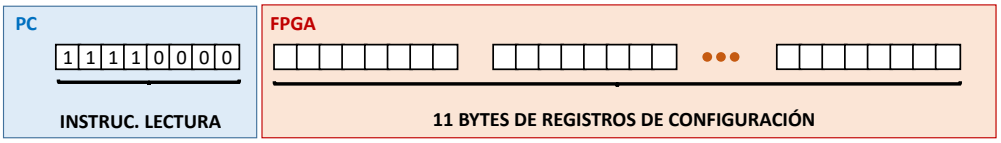


El procedimiento de comunicación es el siguiente:

- Para transmitir los 11 bytes desde el PC al dispositivo FPGA, primero se envía desde el PC un byte con el código "0Fh", que corresponde a la instrucción de escritura, y después, se envían los 11 bytes en el orden indicado anteriormente.



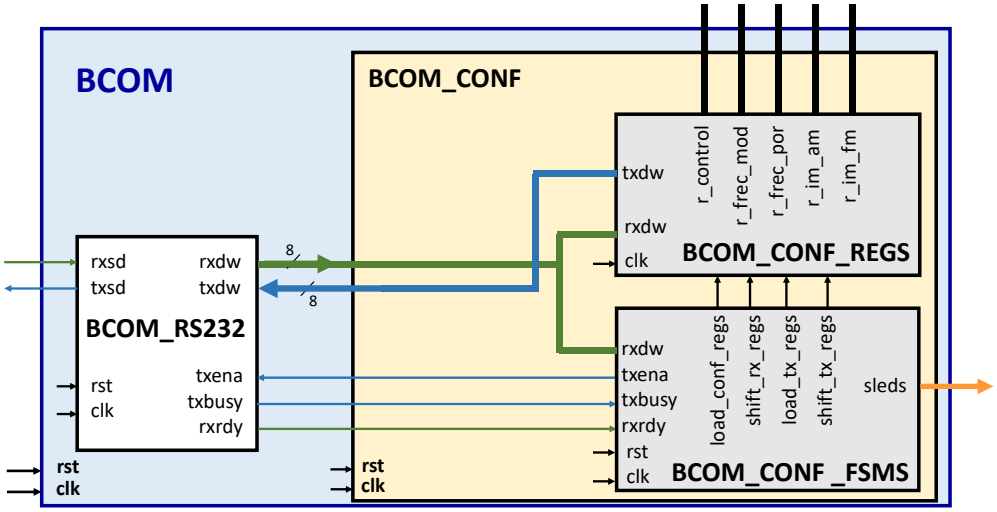
- Para transmitir desde el dispositivo FPGA al PC el contenido de los registros, primero se envía desde el PC un byte con el código "F0h", que corresponde a la instrucción de lectura, y después, se envían desde el dispositivo FPGA los 11 bytes manteniendo el orden indicado anteriormente.



2.3. Módulo *bcom*

El módulo *bcom* es el que se encarga de gestionar la comunicación entre el PC y el dispositivo FPGA. Está formado por 2 bloques conectados entre sí: *bcom_rs232* y *bcom_conf*. El módulo *bcom_conf* es el que hay que diseñar e implementar en esta práctica. Lo forman los módulos *bcom_conf_regs* y *bcom_conf_fsms*. Sin embargo, para su diseño es necesario entender el funcionamiento del módulo *bcom_rs232*.

En las salidas del módulo *bcom* están accesibles los contenidos de todos los registros de configuración del modulador AM/FM. Estos registros se conectarán a la la ruta de datos completa del modulador, que se implementará en la práctica P6.



A continuación, se describe el interfaz y comportamiento de los 3 bloques necesarios para realizar la práctica: *bcom_rs232*, *bcom_conf_regs* y *bcom_conf_fsms*.

2.3.1. Módulo *bcom_rs232*

El módulo *bcom_rs232* se encarga de implementar la capa física de la comunicación entre el PC y el dispositivo FPGA. Está configurado para transmitir/recibir bytes a 57 600 baudios, incluyendo 1 bit de start y 1 bit de stop. Su interfaz se describe en la siguiente tabla:

Tabla 2: Interfaz del módulo bcom_RS232

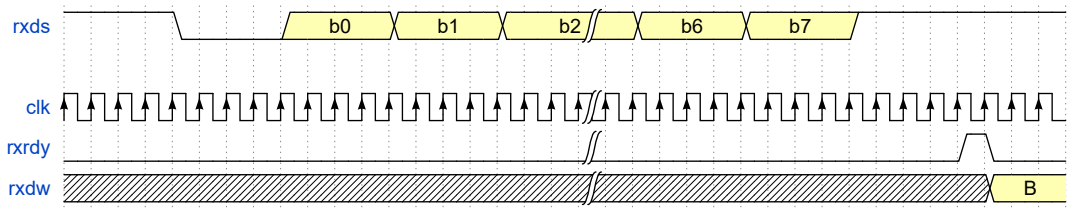
NOMBRE	TIPO	FORMATO	DESCRIPCIÓN
rxsd	in	bit	Dato serie recibido en el puerto RS232
txdw	in	8 bits	Dato (byte) a transmitir
txena	in	bit	Validación de transmisión, activo a nivel alto
rst	in	bit	Enta de reset, activa a nivel alto
clk	in	bit	Entrada de reloj
txsd	out	bit	Dato serie transmitido por el puerto RS232
rxdw	out	8 bits	Dato (byte) recibido
txbusy	out	bit	Flag de estado de transmisión, activo a nivel alto
rxrdy	out	bit	Flag de indicación de dato recibido, activo a nivel alto

El comportamiento de la entrada ***txena*** y los flags de estado ***txbusy*** y ***rxrdy*** se resume acontinuación:

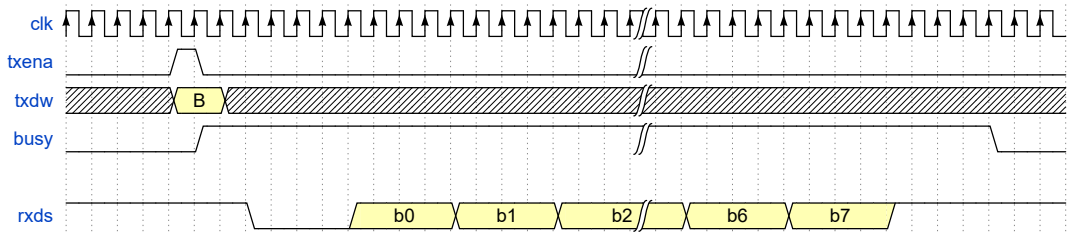
- Cuando la entrada de validación de transmisión ***txena*** se activa (a nivel alto) el byte presente en ***txdw*** se transmite en serie a través de la salida serie ***txsd***.
- El flag de estado de transmisión ***txbusy*** activado (a nivel alto) indica que el módulo *bcom_rs232* está ocupado transmitiendo un byte (y no se puede realizar otra transmisión). En ese caso, no se debe activar la entrada ***txena*** hasta que ***txbusy*** se haya desactivado.
- El flag ***rxrdy*** indica que se ha recibido un dato por el puerto serie y está disponible en ***rxdw***.

En las siguientes figuras se ilustran los cronogramas que muestran el funcionamiento del interfaz en el caso de recepción y transmisión serie del byte B=[b7 b6 b5 b4 b3 b2 b1 b0]:

- Ejemplo recepción de dato serie: el PC envía el byte B=[b7 b6 b5 b4 b3 b2 b1 b0] al dispositivo FPGA



- Ejemplo envío de dato serie: El dispositivo FPGA envía el byte B=[b7 b6 b5 b4 b3 b2 b1 b0] al PC. Mientas el flag ***txbusy*** está activo (a 1), no se puede enviar otro dato.

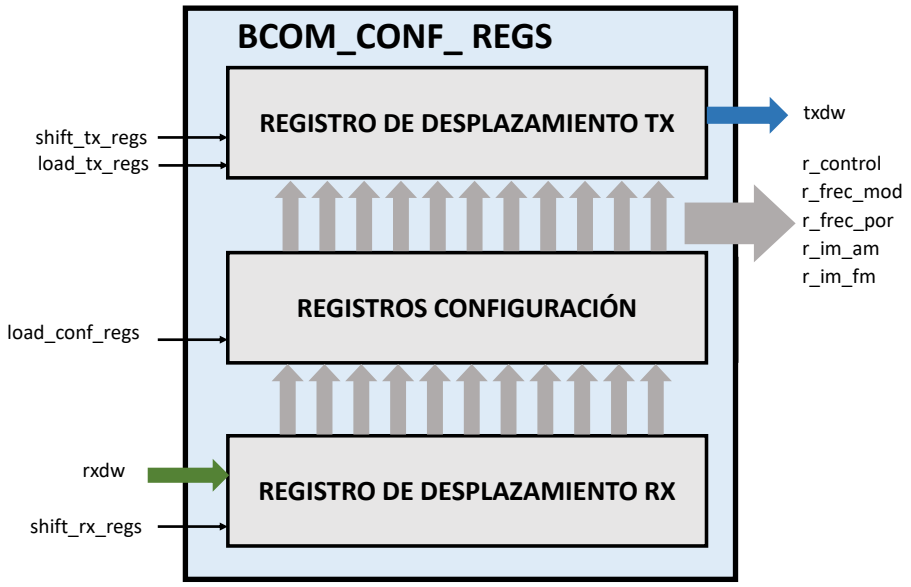


2.3.2. Módulo *bcom_conf_regs*

El módulo *bcom_conf_regs*, cuyo esquema se muestra en la siguiente figura, contiene los registros de desplazamientos necesarios para la recepción y transmisión serie, byte a byte, de los bytes de configuración y los registros de configuración:

- Registro de desplazamiento RX (SR-RX): es un registro de desplazamiento de 11 registros de tamaño byte, con carga serie y salida paralela. Se encarga de ir almacenando y desplazando los 11 bytes cuando lo indique la señal de desplazamiento ***shift_rx_regs***, manteniendo el orden en el que son recibidos.
- Registros de configuración: lo forman 11 registros de tamaño byte que se cargarán a la vez, cuando lo indique la señal de carga ***load_conf_regs***.

- Registro de desplazamiento de TX (SR-TX): es un registro de desplazamiento de 11 registros de tamaño byte, con carga paralela y salida serie. Se encarga de capturar los valores de los registros de configuración cuando lo indique la señal ***load_tx_regs*** y de ir desplazándolos cuando lo indique la señal de desplazamiento ***shift_tx_regs*** para transmitirlos al PC.



Su interfaz es el siguiente:

Tabla 3: Interfaz del módulo bcom_conf_regs

NOMBRE	TIPO	FORMATO	DESCRIPCIÓN
rxdw	in	8 bits	Dato (byte) recibido
shift_rx_regs	in	bit	Activación (a nivel alto) del desplazamiento del SR-RX
load_conf_regs	in	bit	Carga (a nivel alto) el registro de configuración
load_tx_regs	in	bit	Carga (a nivel alto) el SR-TX
shift_tx_regs	in	bit	Activación (a nivel alto) del desplazamiento del SR-TX
clk	in	bit	Entrada de reloj
txdw	out	8 bits	Dato (byte) a transmitir
r_control	out	8 bits	Registro de control
r_frec_mod	out	U[24,24]	Paso del DDS para generar la frecuencia moduladora
r_frec_por	out	U[24,24]	Paso del DDS para generar la frecuencia portadora
r_im_am	out	U[16,15]	Índice de modulación de AM
r_im_fm	out	U[16,15]	Índice de modulación de FM

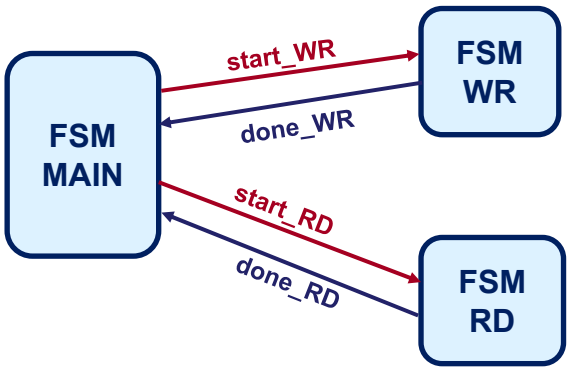
2.3.3. Módulo *bcom_conf_fsms*

El módulo *bcom_conf_fsms* es el que gestiona la comunicación. Tendrá que detectar el byte de instrucción enviado por el PC y proceder a dar las órdenes a los módulos *bcom_rs232* y *bcom_conf_regs*, para ejecutar la instrucción detectada. Su comportamiento es el siguiente:

- Si se recibe una instrucción de escritura deberá esperar a que se reciban los 11 bytes e ir almacenándolos en el registro de desplazamiento SR-RX. Una vez han llegado todos los bytes debe cargarlos en los registros de configuración y volver a un estado que le permita decodificar otra instrucción.
- Si se recibe una instrucción de lectura deberá almacenar los contenidos de los registros de configuración en el registro de desplazamiento SR-TX e ir transmitiendo los 11 bytes teniendo en cuenta que el módulo *bcom_rs232* no esté en estado ocupado. Una vez se hayan transmitido todos los bytes debe volver a un estado que le permita decodificar otra instrucción.
- Si no se recibe ni la instrucción de escritura ni de lectura deberá ir a un estado de error, del que podrá salir si se le envía una instrucción correcta.

Para simplificar el diseño del módulo *bcom_conf_fsms* se ha dividido en 3 máquinas de estado (FSM) que interactúan entre sí, tal y como se muestra en la figura:

- *bcom_conf_fsms_main* es la FSM encargada de decodificar la instrucción y de iniciar las tareas de lectura o escritura o generar el estado de error de instrucción;
- *bcom_conf_fsms_wr* genera las señales de control necesarias para completar el proceso de escritura de los bytes recibidos en los registros de configuración;
- *bcom_conf_fsms_rd* genera las señales de control necesarias para completar el proceso de lectura de los registros de configuración y su transmisión por el puerto serie.



El interfaz del módulo bcom_conf_fsms es el siguiente:

Tabla 4: Interfaz del módulo *bcom_conf_fsms*

NOMBRE	TIPO	FORMATO	DESCRIPCIÓN
rxdw	in	8 bits	Dato (byte) recibido
txbusy	in	bit	Flag de estado de transmisión, activo a nivel alto
rxrdy	in	bit	Flag de indicación de dato recibido, activo a nivel alto
rst	in	bit	Enta de reset, activa a nivel alto
clk	in	bit	Entrada de reloj
sleds	out	9 bits	Monitorización de estados de las FSMs
txena	out	bit	Validación de transmisión, activo a nivel alto
shift_rx_regs	out	bit	Activación (a nivel alto) del desplazamiento del SR-RX
load_conf_regs	out	bit	Carga (a nivel alto) el registro de configuración
load_tx_regs	out	bit	Carga (a nivel alto) el SR-TX
shift_tx_regs	out	bit	Activación (a nivel alto) del desplazamiento del SR-TX

3. Ficheros

Ficheros necesarios para realizar las prácticas:

- `P5.zip`: Proyecto de Quartus para verificación con la tarjeta *DE2_115*
- `top_test_bcom_DE2_115.sv`: Modelo Verilog completo
- `bcom.sv`: Modelo Verilog completo
- `bcom_conf.sv`: Modelo Verilog completo
- `bcom_conf_regs.sv`: Interfaz Verilog del módulo *bcom_conf_regs*
- *`bcom_conf_regs_tsb.sv`: Banco de pruebas del módulo *bcom_conf_regs*
- `bcom_conf_fsms.sv`: Modelo Verilog del módulo *bcom_conf_fsms*
- *`bcom_conf_fsms_tsb.sv`: Banco de pruebas del módulo *bcom_conf_fsms*
- `bcom_conf_fsms_main.sv`: Interfaz Verilog del módulo *bcom_conf_fsms_main*
- *`bcom_conf_fsms_main_tsb.sv`: Banco de pruebas del módulo *bcom_conf_fsms_main*
- `bcom_conf_fsms_wr.sv`: Interfaz Verilog del módulo *bcom_conf_fsms_wr*
- *`bcom_conf_fsms_wr_tsb.sv`: Banco de pruebas del módulo *bcom_conf_fsms_wr*

- `bcom_conf_fsms_rd.sv`: Interfaz Verilog del módulo *bcom_conf_fsms_rd*
- *`bcom_conf_fsms_wr_tsb.sv`: Banco de pruebas del módulo *bcom_conf_fsms_rd*
- `bcom_conf_tsb.sv`, `bcom_conf_tsb.do` y `bcom_conf_tsb.tcl`: Ficheros para verificar el módulo *bcom_conf*.
- `test_serial_port_bcom.m`: script de Matlab con un ejemplo de configuración, escritura y lectura del puerto serie

NOTA: Los ficheros marcados con un "*" no están disponibles en Poliformat y los tendrá que generar el alumno.

4. Tareas a realizar

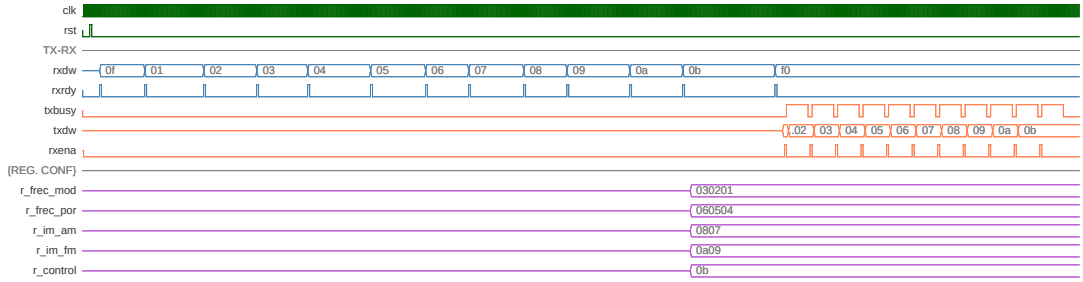
Teniendo en cuenta las especificaciones y descripción del sistema realizada anteriormente:

1. Modele usando System Verilog el módulo *bcom_conf_regs* utilizando el interfaz definido en `bcom_conf_regs.sv` y realice un sencillo banco de pruebas para validar su comportamiento.

2. Abra el fichero `bcom_conf_fsms.sv` y dibuje el esquema de conexiones entre los módulos *bcom_conf_fsms_main*, *bcom_conf_fsms_wr* y *bcom_conf_fsms_rd*. Tenga en cuenta las conexiones entre dichos bloques para realizar la siguiente tarea.

3. Para cada uno de los tres módulos que forman *bcom_conf_fsms* (*bcom_conf_fsms_main*, *bcom_conf_fsms_wr* y *bcom_conf_fsms_rd*): dibuje el diagrama de estados que describe su comportamiento, modélelo con System Verilog y compruebe su funcionamiento escribiendo un sencillo banco de pruebas. Tenga en cuenta que las salidas ***sled*** de cada módulo se han incluido para facilitar la depuración del sistema. A ellas debe asignar un código que indique en qué estado está la máquina de estados.

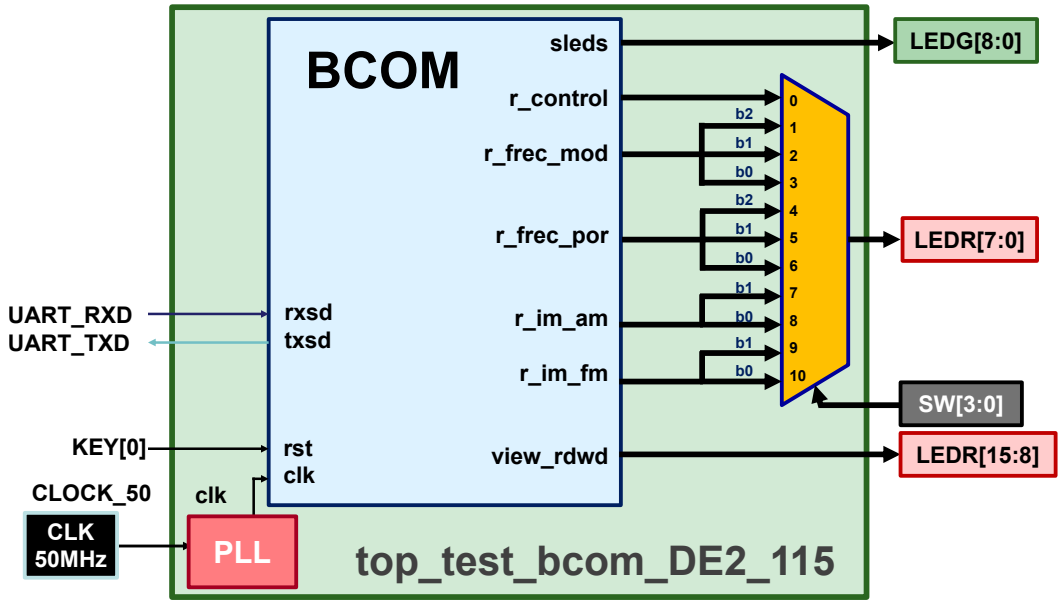
4. Depure el funcionamiento del módulo *bcom_conf*, que realiza las conexiones entre *bcom_conf_regs* y *bcom_conf_fsms*, utilizando el banco de pruebas del fichero `bcom_conf_tsb.sv`. Los resultados del test se visualizan utilizando el fichero `com_conf_tsb.do`. El banco de pruebas transmite una trama de escritura enviando los valores del 1 al 11 en cada byte transmitido y, posteriormente, inicia una trama de lectura (ver la siguiente figura) y comprueba que se han recibido correctamente los 11 bytes. También, envía ciertos mensajes por la consola indicando si se han transmitido correctamente los bytes desde la FPGA y si los registros quedan bien configurados.



5. El fichero `P5.zip` contiene el proyecto de Quartus `test_bcom_DE2_115.qsf` preparado para su ejecución. Abra el proyecto y compile el módulo `top_test_bcom_DE2_115` y compruebe que no aparecen errores de sintaxis y que el circuito puede funcionar a una frecuencia de al menos 25 MHz. Encienda la tarjeta DE2-115, configure el dispositivo FPGA y pase a la siguiente tarea.

6. La verificación del funcionamiento del módulo *bcom_conf* se va a realizar utilizando la tarjeta DE2-115. Para transimitir y recibir los datos se requiere el script de Matlab `test_serial_port_bcom.m`. Este script contiene un ejemplo de configuración y uso del puerto serie para escribir y leer bytes. Lea el script y compruebe que se configura el puerto serie y se escriben y leen 11 bytes, precedidos, en ambos casos, por la escritura del código de instrucción ("0Fh" para escritura y "F0h" para lectura). Abra el administrador de dispositivos de su PC y compruebe en qué puerto "COM" está conectado el dispositivo. Úselo para comprobar el módulo *bcom_conf*. Si no funciona, deberá usar para su depuración la información monitorizada con los LEDs de la tarjeta DE2-115 (que se describe a continuación) y la herramienta "*Signal Tap Logic Analyzer*".

   Para comprobación del funcionamiento del sistema tenga en cuenta que el módulo *bcom* se ha instanciado en el módulo `top_test_bcom_DE2_115` añadiendo un multiplexor para visualizar, a través de los LEDs, los valores de los diferentes bytes que forman los registros y el byte recibido. Su esquema se muestra a continuación. Como puede observarse, las salidas sled, que monitorizan el estado del bloque *bcom_conf_fsms*, están conectadas a los LEDs verdes, los bytes de los registros de configuración están conectados, a través de un multiplexor controlado por los interruptores SW[3:0], a los LEDs rojos LEDR[7:0] y el byte recibido por el puerto serie está conectado a los LEDs rojos LEDR[15:8].


BCOM
sleds
r_control
r_frec_mod
r_frec_por
r_im_am
r_im_fm
view_rdwd
rxsd
txsd
rst
clk
clk
PLL
top_test_bcom_DE2_115
LEDG[8:0]
LEDR[7:0]
SW[3:0]
LEDR[15:8]
UART_RXD
UART_TXD
KEY[0]
CLOCK_50
CLK 50MHz
b2
b1
b0
0
1
2
3
4
5
6
7
8
9
10