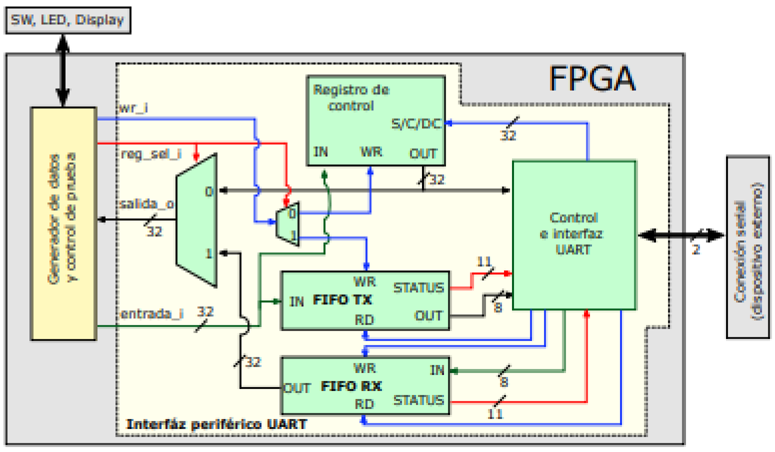
Part 1 - FIFO

1) Implement a FIFO using the "FIFO Generator" IP core. This FIFO should have an 8-bit word width and a length of 16 words. Additionally, it should include outputs for FIFO full, FIFO empty, and the number of present words. Do not register the output of the FIFO. Use the "Common Clock Block RAM" implementation.

2) Write a TOP module that includes the FIFO and a PLL with an output frequency of 16 MHz. Simulate this block and demonstrate the operation of the different input and output signals.

Part 2)



Registro de control - reg_sel = 0

31 ··· 29	28 ··· 20	19 ··· 17	16 ··· 8	7 ··· 4	3	2	1	0
N/A	Bytes TX (RO)	N/A	Bytes RX (RO)	N/A	RXAV (RO)	leer (WC)	FTXF (RO)	enviar (WC)

Registros de datos - reg_sel = 1

31 ··· 8	7 ··· 0
N/A	DATO (RW)

*Por simplicidad mantener campos N/A siempre en cero

RW - read/write
RO - read only
WC - write / clear when done

1) Develop the UART peripheral as shown in Figure 1. For this peripheral, there is a control register and a pair of FIFO memories for input and output data.

- Control Register: The contents of the control register can be written or read by an external agent when reg sel i=0. In addition, the UART control unit is always reading the contents of this register to handle requests and update states. There is a bus called S/C/DC, which sends word counts available in the FIFOs and allows setting the state bits (RXAV, FTXD, read, and send) to one or zero, depending on the actions being performed by the control. The content of the S/C/DC bus takes priority over the WR signal derived from the bus coming from the external agent. Here is a description of the fields in this register:

    - **Send:** A high value in this bit triggers the data transfer process through the serial port. The bit remains high during the process and is lowered to zero at the end of the operation. During the transfer, the system sends the data stored in the transmit FIFO (TX FIFO). At the end of the transaction, the internal control unit must clear the send bit (set it to zero) to indicate the end of the operation.

    - **Read:** A high value in this bit allows extracting a byte from the receive FIFO (RX FIFO). The byte will be available when reading the peripheral using reg sel i=1. Note that if the RX FIFO is empty, the system must ignore the request to extract a byte from the FIFO. Regardless of whether the operation generates output data or not, the system must automatically clear the read bit after processing the request.

    - **FTXF:** This bit indicates that the transmit FIFO (TX) is full and cannot accept new data to transmit.

- **RXAV:** This bit indicates that the UART peripheral has received at least one byte from the external system connected to the FPGA. Each time the UART peripheral receives data, it must store it in the RX FIFO. To indicate that there is data available in the RX FIFO, the internal control unit must set this field (RXAV) to 1. The bit must self-clear when there is no data available in the RX FIFO (all data has been read).

- **Bytes TX and Bytes RX:** These are two 9-bit vectors that indicate the number of unprocessed words present in the TX FIFO and RX FIFO, respectively. These vectors must be automatically incremented each time a new word is received in the FIFO and decremented each time a word is extracted from the FIFO.

- Data FIFOs:

  - The UART peripheral module has two FIFOs with a word width of 8 bits. To write or read from these registers from outside the module, the reg sel i signal must be set to one. The external agent can only read data from the RX FIFO, while any write attempt is made to the TX FIFO. Note that the write action only requires reg sel i = 1 and WE i=1, while reading requires setting the read bit of the control register to one beforehand.

  - Each FIFO has a length of 512 words, with each word being 8 bits wide. Note that the input and output buses of the peripheral are 32 bits, so adjustments must be made to maintain both the bus width and the input/output widths of the FIFOs according to Figure 1.

2) The UART peripheral to be developed must be capable of handling bidirectional serial communication at a speed of 115200 baud. Along with this guide, you have been provided with a UART interface module that allows the management of transmission and reception protocols. This block can be used as part of the "UART Control and Interface."

3) Develop an appropriate testbench to demonstrate the functionality of your module. You can use the same UART inputs to generate your stimulus generator block.

4) Develop an appropriate test block that allows sending and receiving binary data to and from a computer connected to the FPGA. In the case of sending information to the computer, the system must have a mechanism to capture an arbitrary amount of data, which must be sent in bursts when a button is pressed. Similarly, the system must be able to receive an arbitrary amount of data from the PC and display it one by one when a button is pressed. Use LEDs and a seven-segment display to show values in the register.

NOTES)
CLK and FIFO are made with IP catalog (Clocking Wizard and Fifo Generator)
Both parts use a CLKW with 16MHz output and 100MHz input
Has to be in System Verilog and FPG number is xc7a35tcpg236-1