

# Trabajo Final de Carrera

## Ingeniería Electrónica

Desarrollo e implementación de un puerto USB 2.0 de alta velocidad  
para FPGA a través de un puente específico

**Edwin Barragán**

Autór

**Mag. Ing. Cristian Sisterna**

**Mgtr. Ing. Martín Perez**

**Dr. Ing Marcelo Segura**

Asesores



Universidad Nacional  
de San Juan



Facultad de Ingeniería  
Universidad Nacional de San Juan

Cuando?No sabemos aún



# Agenda

- 1 Introducción
- 2 Implementación
- 3 Evaluación y validación
- 4 Resultados y conclusiones

# Agenda

## 1 Introducción

- Bus Serial Universal

## 2 Implementación

- Arquitectura del sistema
- Selección de los componentes del sistema
- Configuración de la Interfaz USB
- Desarrollo de la Máquina de Estados Finitos
- Circuito de interconexión

# Agenda

## 3 Evaluación y validación

- Desarrollo del sistema de pruebas
- Desarrollo de un programa de pruebas

## 4 Resultados y conclusiones

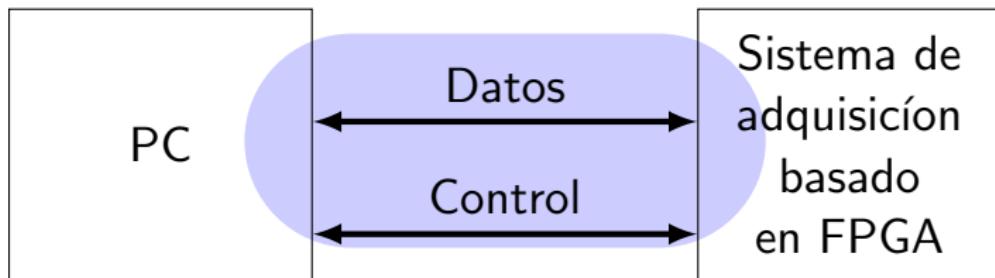
- Pruebas y resultados
- Conclusiones
- Trabajo futuro

# Agenda

## 1 Introducción

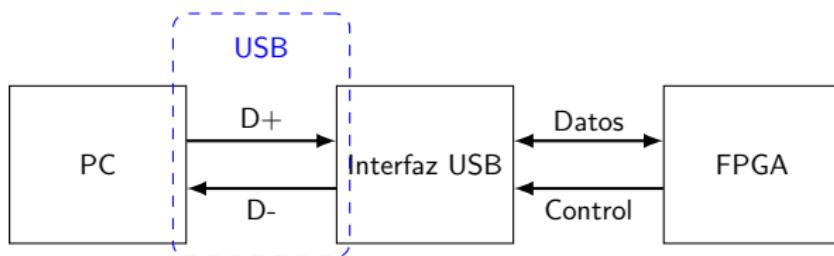
- Bus Serial Universal

# Objetivos



- Objetivo General
  - ▶ Realizar una comunicación entre un FPGA y una PC mediante USB 2.0.

# Objetivos

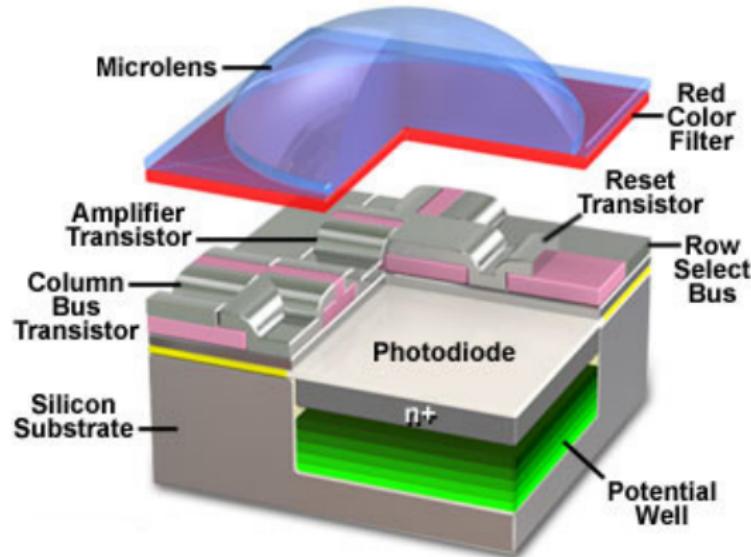


- Objetivos Particulares

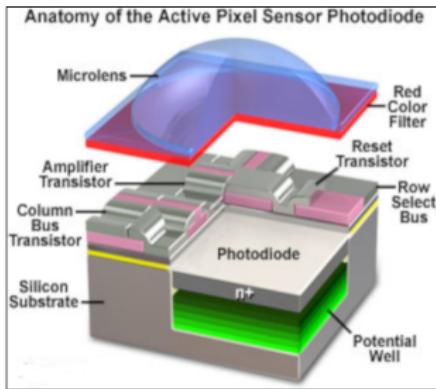
- ▶ Seleccionar la interfaz USB.
- ▶ Comprender el funcionamiento del kit de desarrollo CY3684 utilizado como interfaz USB y el framework provisto por Cypress.
- ▶ Realizar la configuración de la interfaz USB.
- ▶ Sintetizar un circuito en VHDL que sea capaz de interactuar con la interfaz USB.
- ▶ Sintetizar circuitos de prueba para realizar una verificación funcional.
- ▶ Validar el funcionamiento.

# Sensores basados en la tecnología CMOS

Anatomy of the Active Pixel Sensor Photodiode



# Nuevos sensores basados en la tecnología CMOS



## A Low Cost Environmental Ionizing Radiation Detector Based on COTS CMOS Image Sensors

Clara Lucía Galimberti<sup>1,2</sup>, Fabrício Alcalde Bessa<sup>1,3</sup>, Martín Pérez<sup>2</sup>, Mariano Gómez Berizzo<sup>1,3</sup>, Miguel Sofio Hueso<sup>1,3</sup>, Iván Siedlech<sup>1,3</sup>, Jerónimo Blonstein<sup>1,3</sup>, Hernán Astor<sup>1,2</sup> y José Lipovetsky<sup>1,3,4</sup>

<sup>1</sup>clara@ceis.unrc.edu.ar

<sup>2</sup>falcade@ib.edu.ar

<sup>3</sup>lip@ib.edu.ar

<sup>4</sup>División Baja Temperatura y Departamento de Física Mólica, Centro Atómico Bariloche, CNEA  
Buenos Aires 9500, San Carlos de Bariloche (8400), Argentina

<sup>1</sup>División Mediciones Nucleares, Centro Atómico Bariloche, CNEA  
Buenos Aires 9500, San Carlos de Bariloche (8400), Argentina

<sup>1</sup>CONICET

<sup>1</sup>Mutua Bariloche

**Abstract:** We present the development of a system for the detection of ionizing radiation based on the Omnisens OV9640 Commercial Off The Shelf image sensor. The data is recorded using a low cost Raspberry Pi board and a PIC 3 composite. The amount of charge and geometrical characteristics of the clusters of pixels excited when a particle interacts with the detector are used to identify the type of incoming particle, distinguishing between alpha particles and X-ray or gamma rays. The software was programmed in C using the OpenCV library. The system was tested with  $^{137}\text{Cs}$  and  $^{231}\text{Am}$  radiation sources.

The location of the measurement. The detector, mounted on different vehicles allowed the collaborative collection of data by citizens creating a dose-rate geographical distribution map of the affected region, and with the years of a many years.

In this work, we present a first prototype of an environmental radiation detector based on Commercial Off The Shelf (COTS) image sensors acquired using a low cost Raspberry Pi board. The data obtained with the sensor can be remotely read using an internet browser. Unlike the PuReGe and

## CMOS pixel sensor development: a fast read-out architecture with integrated zero suppression

Ch. Hu-Guo,<sup>a,1</sup> J. Baudot,<sup>a</sup> G. Bertolone,<sup>a</sup> A. Besson,<sup>a</sup> A.S. Brogna,<sup>a</sup> C. Colledani,<sup>a</sup> G. Claus,<sup>a</sup> R. De Masi,<sup>a</sup> Y. Degerli,<sup>a</sup> A. Dorokhov,<sup>a</sup> G. Doziere,<sup>a</sup> W. Dulinski,<sup>a</sup> X. Fang,<sup>a</sup> M. Gelin,<sup>a</sup> M. Goffe,<sup>a</sup> F. Guilloux,<sup>a</sup> A. Hamm,<sup>a</sup> K. Jaaskelainen,<sup>a</sup> M. Koziel,<sup>a</sup> F. Morel,<sup>a</sup> F. Orsini,<sup>a</sup> M. Specht,<sup>a</sup> Q. Sun,<sup>a</sup> I. Valin<sup>a</sup> and M. Winter<sup>a</sup>

<sup>a</sup>Institut Pluridisciplinaire Hubert Curien, University of Strasbourg, CNRS/IN2P3, 23 rue duless, BP 28, 67071 Strasbourg, France

<sup>b</sup>IRFU/SEEDL,  
CEA Saclay, 91191 Gif-sur-Yvette Cedex, France

E-mail: Christine.Hu@IReS.in2p3.fr

**ABSTRACT:** CMOS Monolithic Active Pixel Sensors (MAPS) have demonstrated their strong potential for tracking devices, particularly for flavour tagging. They are foreseen to equip several vertex detectors and beam telescopes. Most applications require high read-out speed, which imposes sensors to feature digital output with integrated zero suppression. The most recent development of MAPS at IPHC and IRFU addressing this issue will be reviewed. The design architecture, combining pixel array, column-level discriminators and zero suppression circuits, will be presented. Each pixel features a preamplifier and a correlated double sampling (CDS) micro-circuit reducing the temporal and fixed pattern noises. The sensor is fully programmable and can be monitored. It will equip experimental apparatus starting data taking in 2009/2010.

## 32 × 32 pH IMAGE SENSORS FOR REAL TIME OBSERVATION OF BIOCHEMICAL PHENOMENA

T. Hisawa<sup>1</sup>, J. Matsuo<sup>1</sup>, T. Ishida<sup>2,3</sup>, H. Takei<sup>3,4</sup>, H. Abe<sup>3</sup>, K. Sawada<sup>1,3,4</sup> and M. Ishida<sup>1,3,4</sup>  
<sup>1</sup>Dept. of Electrical and Electronic Eng., Toyohashi University of Technology, Toyohashi, JAPAN  
(Tel : +81-532-44-6739; E-mail: sawada@eeet.tuat.ac.jp)

<sup>2</sup>Orimacky Ltd., Toyohashi, JAPAN

<sup>3</sup>Intelligent Sensing System Research Center, Toyohashi University of Technology, Toyohashi, JAPAN

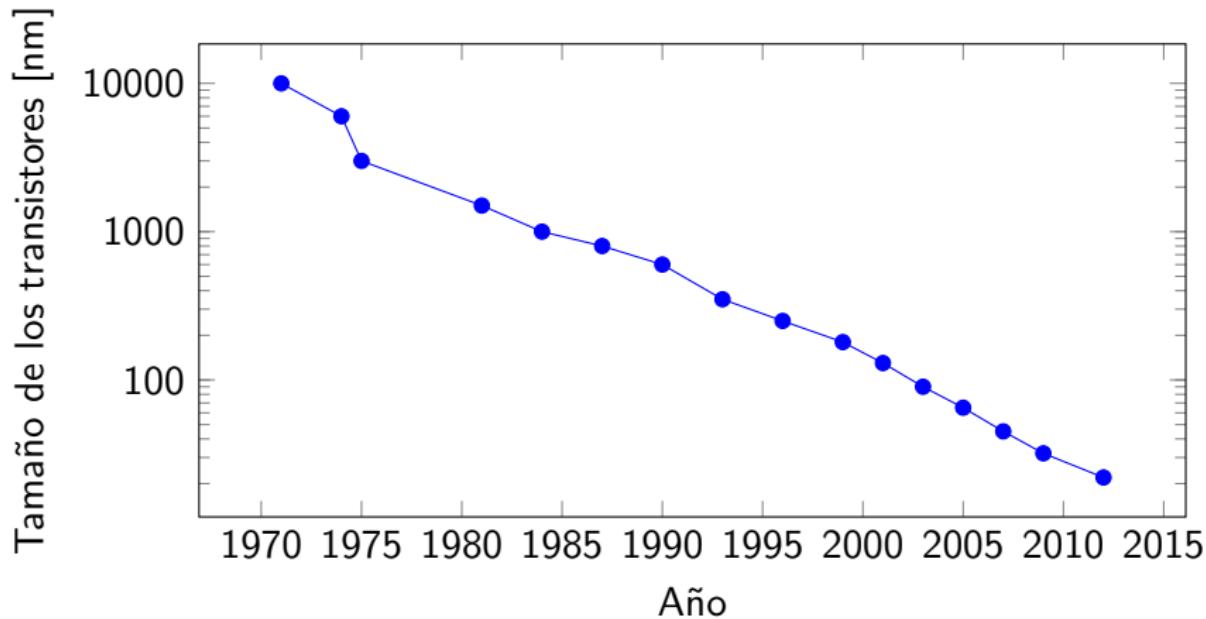
<sup>4</sup>Core System Research for Evolutional Science and Technology (CREST), JAPAN

<sup>b</sup>Biomedical Engineering Research Organization, Tohoku University, Sendai, JAPAN

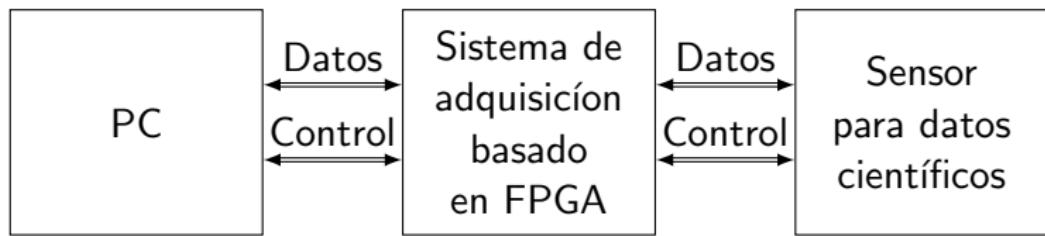
**Abstract:** 32 × 32 pH image sensors were successfully fabricated by using the CCD/CMOS image sensor technique, and real time imaging of a chemical reaction and pH distribution was carried out for the first time. The pH variations by a chemical reaction are observed by 200 ms step (i.e. 5 frames per sec). The pH image sensor was able to take a pH image of mouse stomach successfully. It means that the novel image sensor can be applied to a biomedical and biochemical field.

**Keywords:** pH, two-dimensional imaging, image sensor

# Nuevos sensores basados en la tecnología CMOS



# Comunicación USB 2.0 para sistema científicos implementados en FPGA



# Selección del protocolo de comunicación

Ethernet



Wi-Fi



USB

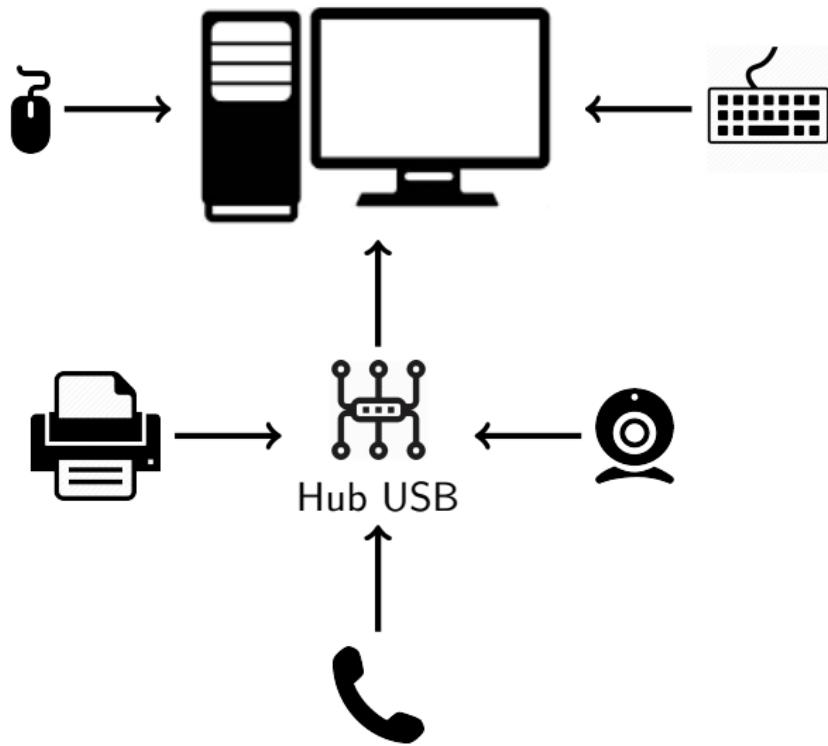


# Selección del protocolo de comunicación



USB 2.0 resulta la mejor alternativa para comunicar periféricos a una PC sin comprometer funcionalidades importantes.

# USB - Bus Serial Universal



## USB 2.0 - Velocidades de operación



Low-Speed: 1.5 Mbps

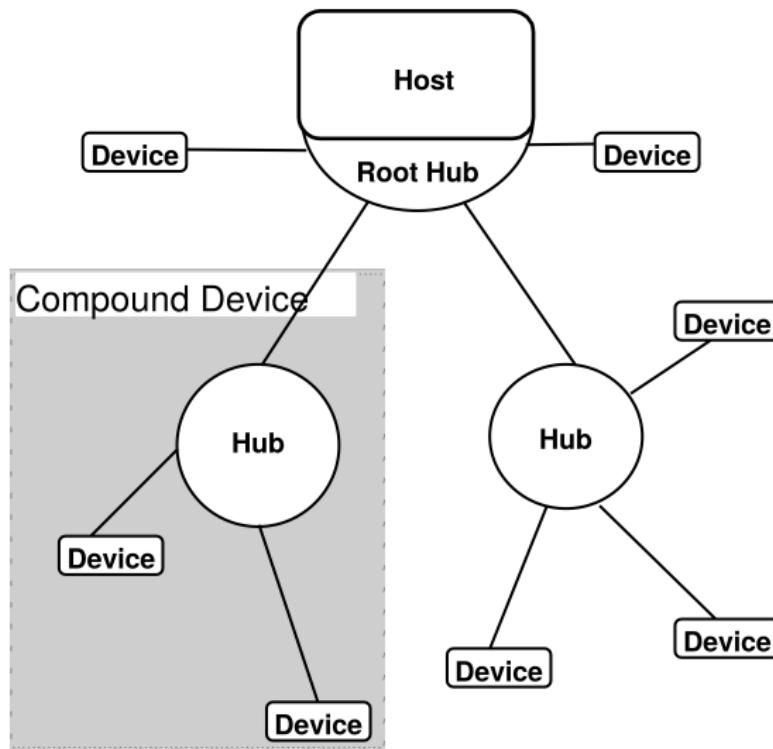


Full-Speed: 12 Mbps



High-Speed: 480 Mbps

# USB - Topología



# USB - Tipo de transferencias

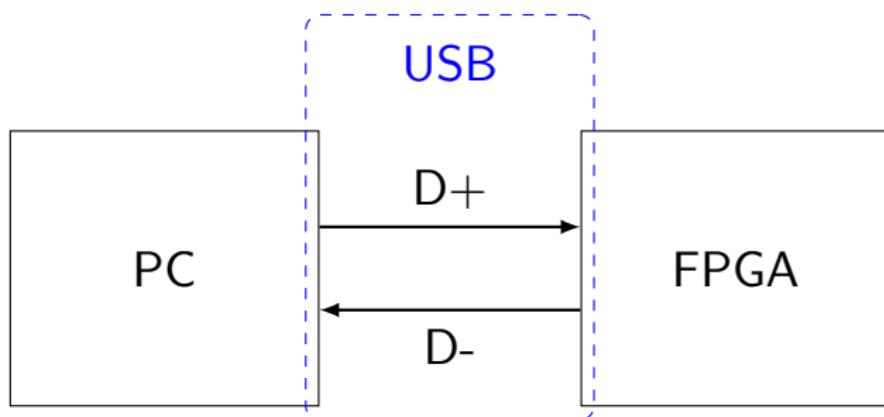
- Control: Destinada a comunicaciones de control y estado del bus entre el Host y los Dispositivos.
- Interrupción: Posee latencia mínima. Destinada a comunicaciones no periódicas.
- Isócronas: Acceso periódico al bus. Sin retransmisión de mensajes. Destinada a la comunicación que caduca con demoras en la entrega, como imágenes.
- en Masa: Destinada a maximizar la ocupación del bus.

# Agenda

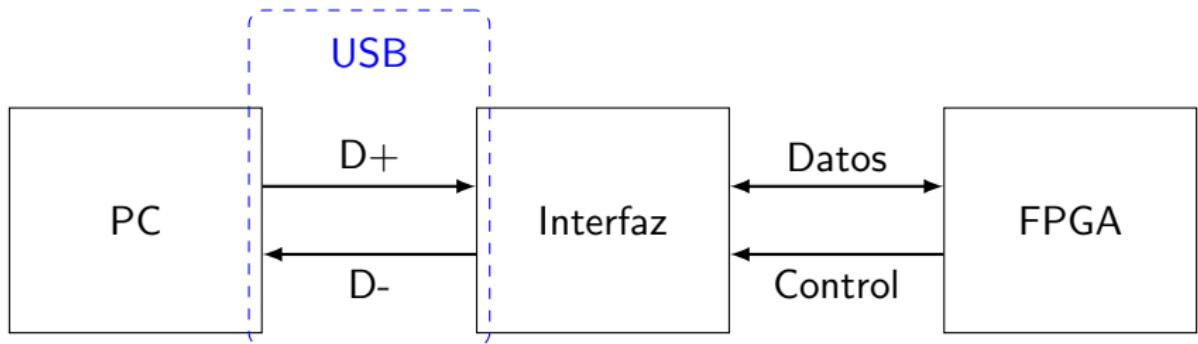
## 2 Implementación

- Arquitectura del sistema
- Selección de los componentes del sistema
- Configuración de la Interfaz USB
- Desarrollo de la Máquina de Estados Finitos
- Circuito de interconexión

# Arquitectura del sistema realizado

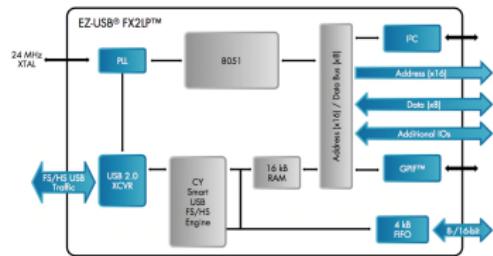


# Arquitectura del sistema realizado

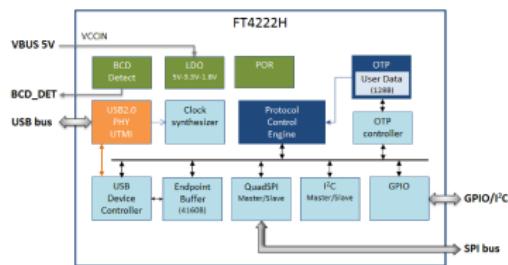


# Selección de la Interfaz USB

## Cypress FX2LP



## FTDI FT4222H



- Pros

- ▶ Comunicación paralela de 16 bits
- ▶ Microcontrolador 8051

- Contras

- ▶ Requiere esfuerzo en la configuración y software dedicado

- Pros

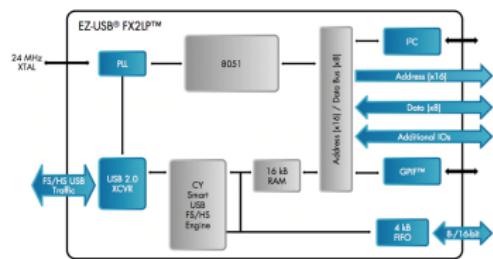
- ▶ No requiere programación
- ▶ Económico

- Contras

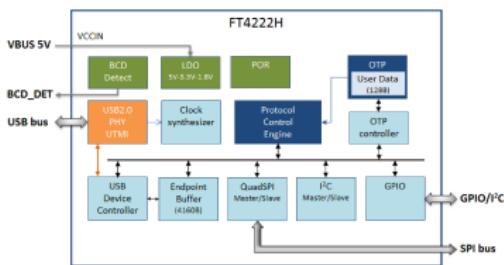
- ▶ Solo permite Transferencias en Masa
- ▶ Comunicación SPI cuádruple

# Selección de la Interfaz USB

## Cypress FX2LP



## FTDI FT4222H



- Pros

- ▶ Comunicación paralela de 16 bits
- ▶ Microcontrolador 8051

- Contras

- ▶ Requiere esfuerzo en la configuración y software dedicado

- Pros

- ▶ No requiere programación
- ▶ Económico

- Contras

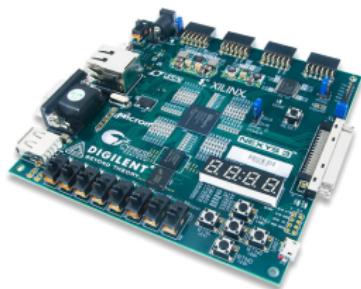
- ▶ Solo permite Transferencias en Masa
- ▶ Comunicación SPI cuádruple

# Selección del FPGA



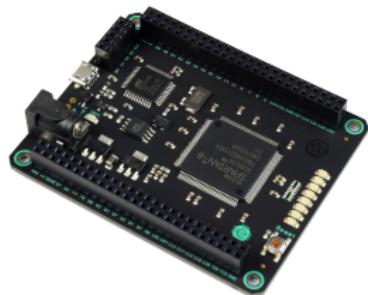
Spartan-3E Starter

- FPGA Spartan 3E de Xilinx
- Gran cantidad de periféricos



Nexys 3

- FPGA Spartan 6 de Xilinx
- Gran cantidad de periféricos



Movo v3

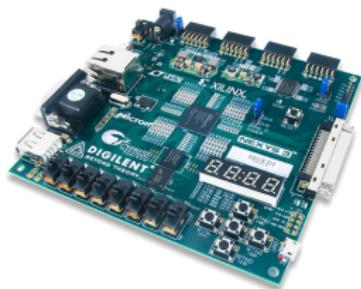
- FPGA Spartan 6 de Xilinx
- Muy económica

# Selección del FPGA



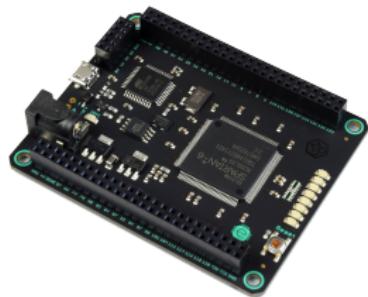
Spartan-3E Starter

- FPGA Spartan 3E de Xilinx
- Gran cantidad de periféricos



Nexys 3

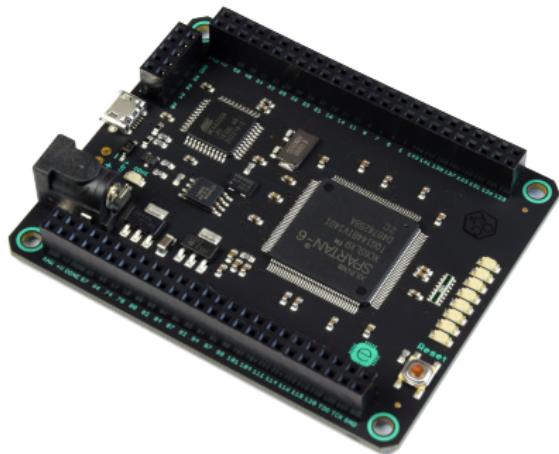
- FPGA Spartan 6 de Xilinx
- Gran cantidad de periféricos



Movo v3

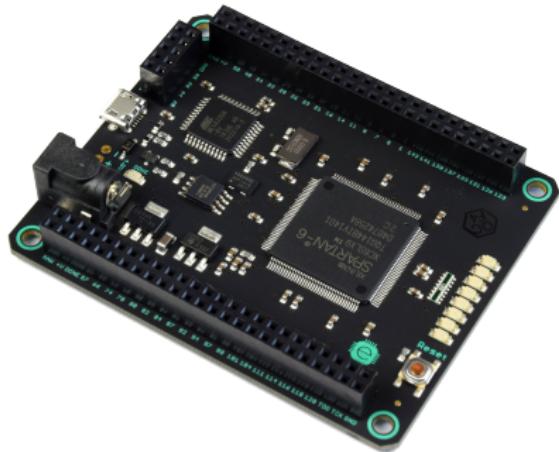
- FPGA Spartan 6 de Xilinx
- Muy económica

# La placa de desarrollo MOJO v3



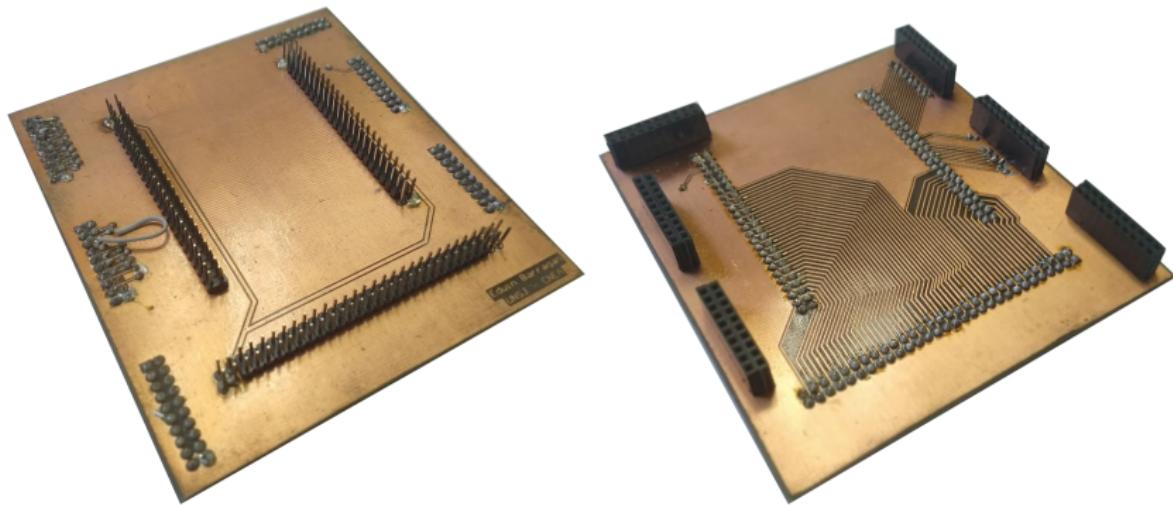
- FPGA Spartan 6 XC6SLX9 de Xilinx
- 84 pines IO digitales
- 8 entradas analógicas
- 8 LEDs de propósito general
- 1 pulsador de propósito general
- ATmega32U4 para configurar la FPGA y leer los pines analógicos.
- Memoria flash para almacenar la configuración de la FPGA.
- USB para programación y comunicación.

# La placa de desarrollo MOJO v3

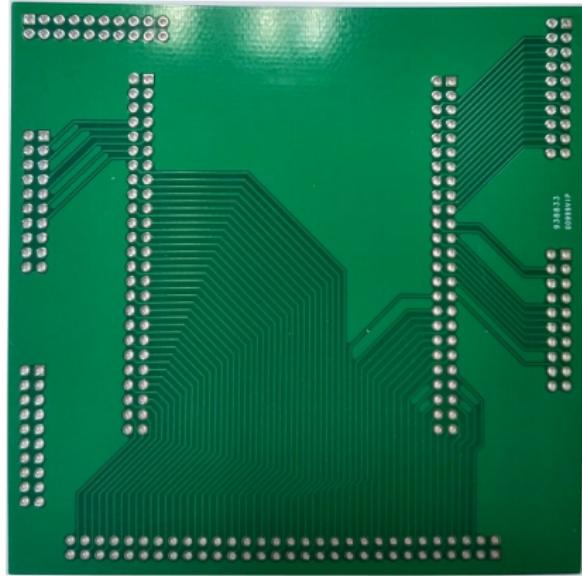
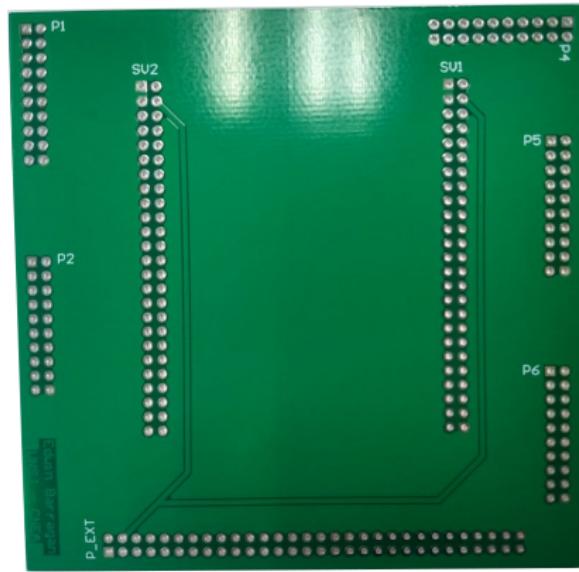


- USB para programación y comunicación.
  - ▶ USB 2.0 Full-Speed de 12 Mbps
  - ▶ Implementado a través de controlador ATmega32U4
  - ▶ Interfaz FPGA-ATmega32U4 via SPI de 8 Mbps.

# Circuito de interconexión



# Circuito de interconexión

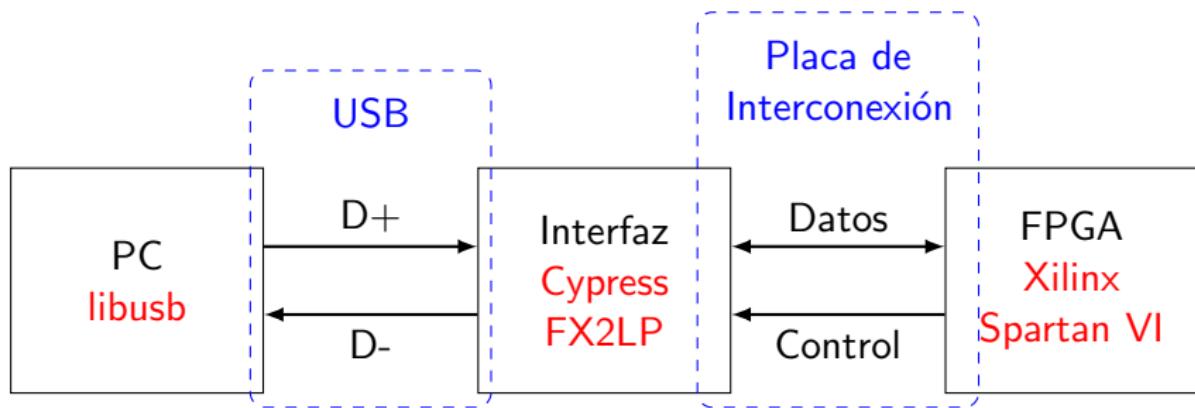


# Acceso al dispositivo desde la PC

Para acceder al puerto USB desde la PC se utilizó la biblioteca libusb-1.0:

- Software libre: Puede ser utilizado sin necesidad de comprar una licencia.
- API portable: Permite escribir código que puede ser compilado en múltiples Sistemas Operativos.
- Abundante documentación: Se encuentran en Internet manuales y tutoriales muy completos sobre su funcionamiento.

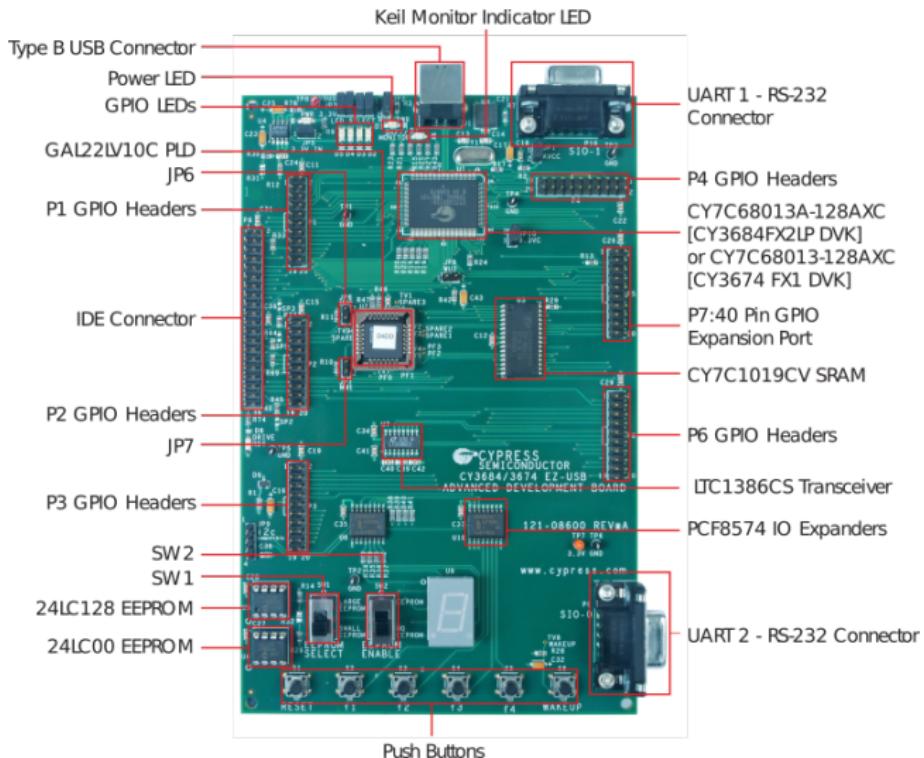
# Componentes del sistema



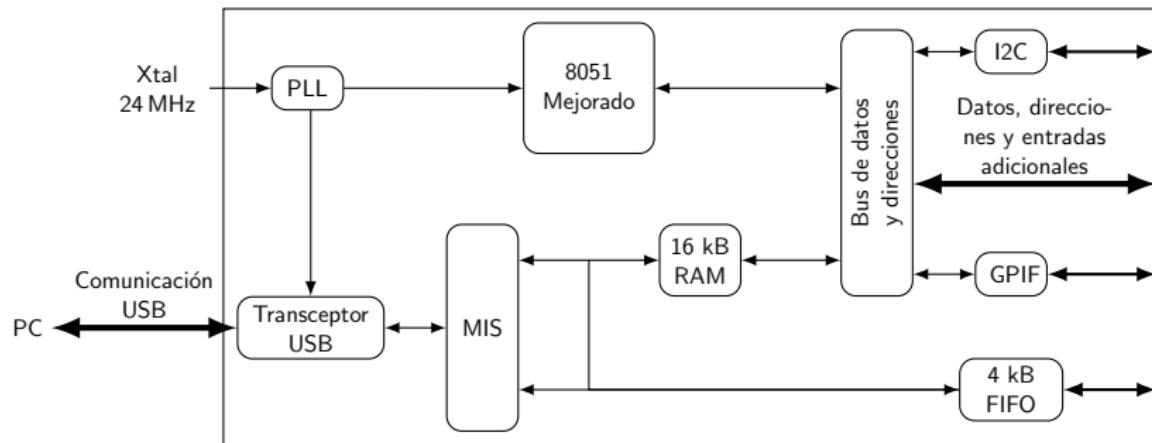
# Sistema completo



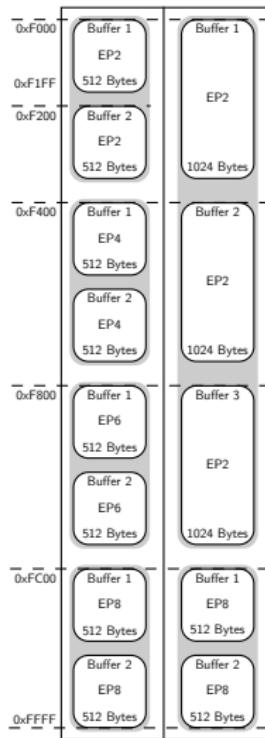
# Placa de desarrollo utilizada para la interfaz



# El circuito integrado FX2LP



# Configuración del dispositivo USB

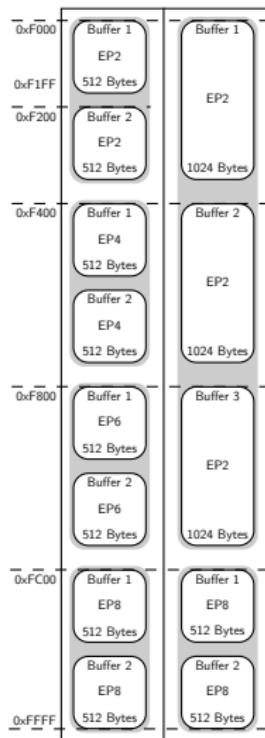


La configuración de las tuberías de comunicación se definieron con la finalidad de obtener el mayor ancho de banda posible a la entrada.

- Entrada:

- Extremo EP2
- Transferencias Isocrónicas
- 1024 Bytes máximo por transferencia
- 3 Buffers

# Configuración del dispositivo USB

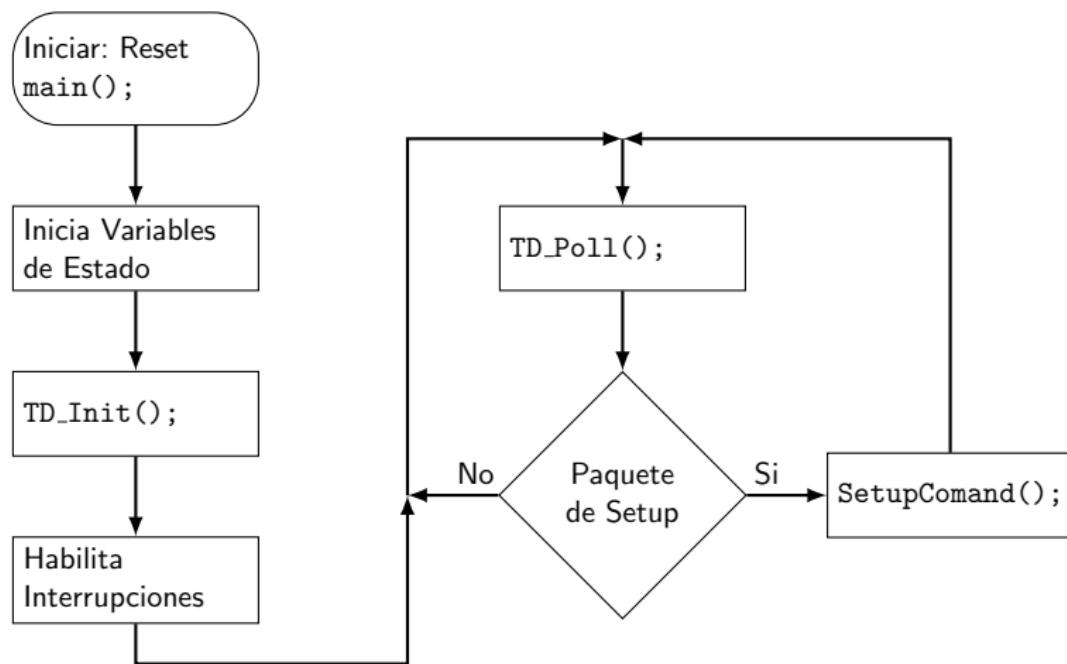


La configuración de las tuberías de comunicación se definieron con la finalidad de obtener el mayor ancho de banda posible a la entrada.

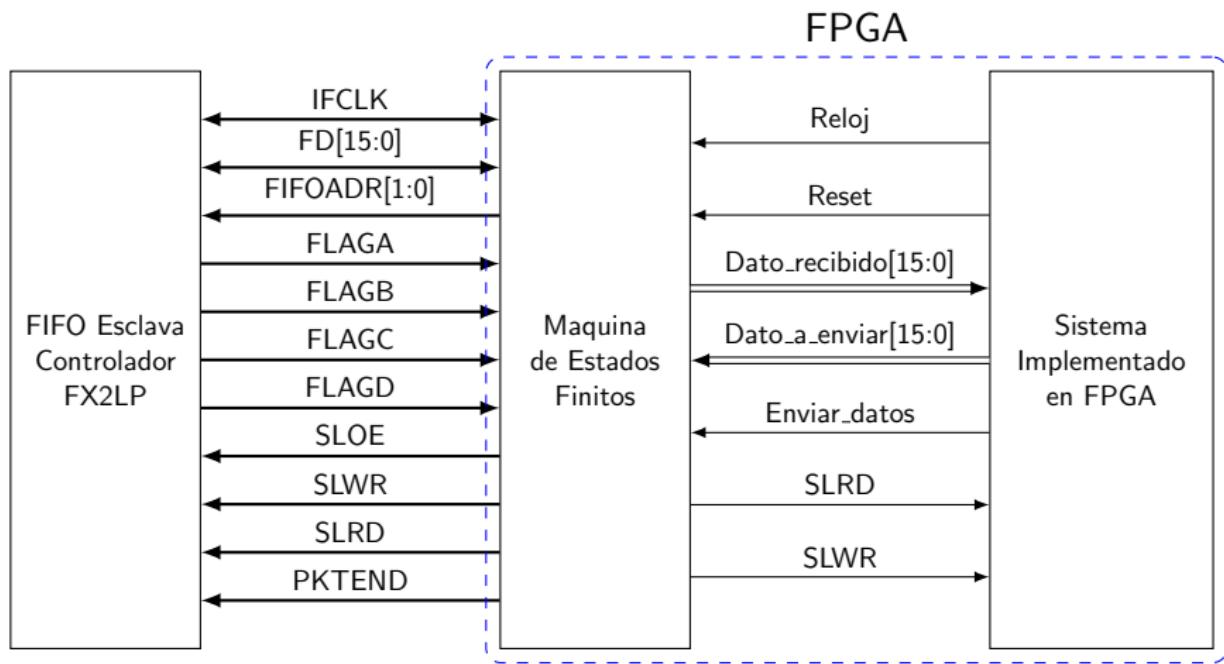
- Salida:

- ▶ Extremo EP8
- ▶ Transferencia por bultos
- ▶ 512 Bytes por transferencia
- ▶ 2 Buffers

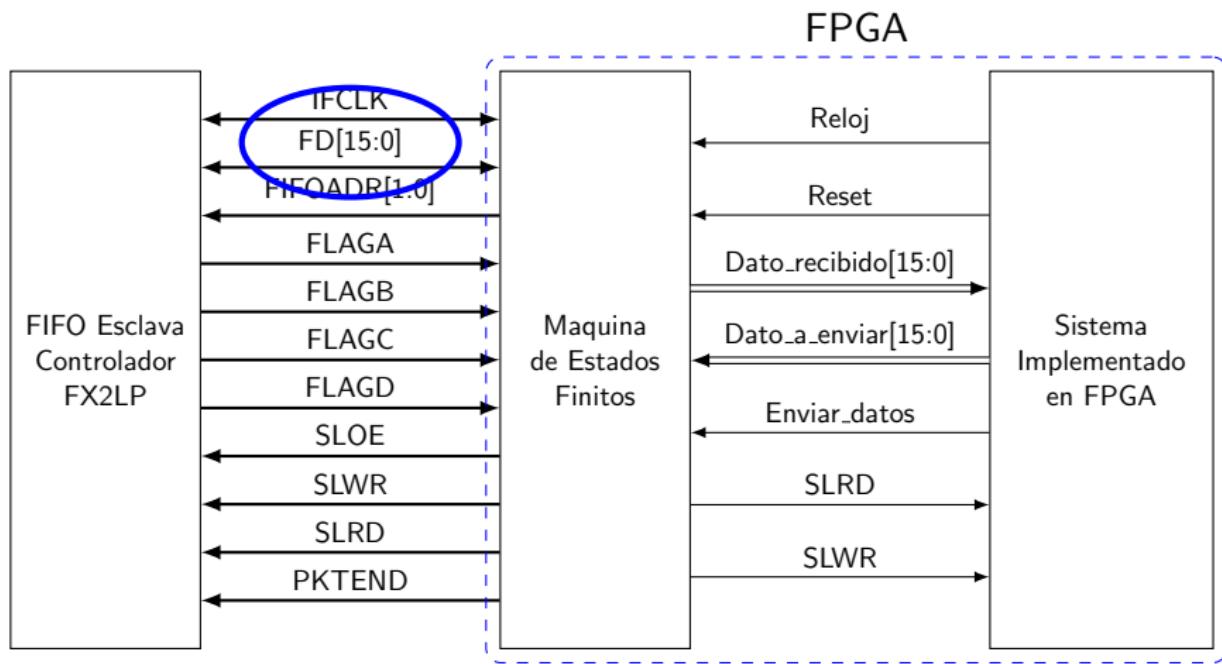
# Programación de la interfaz



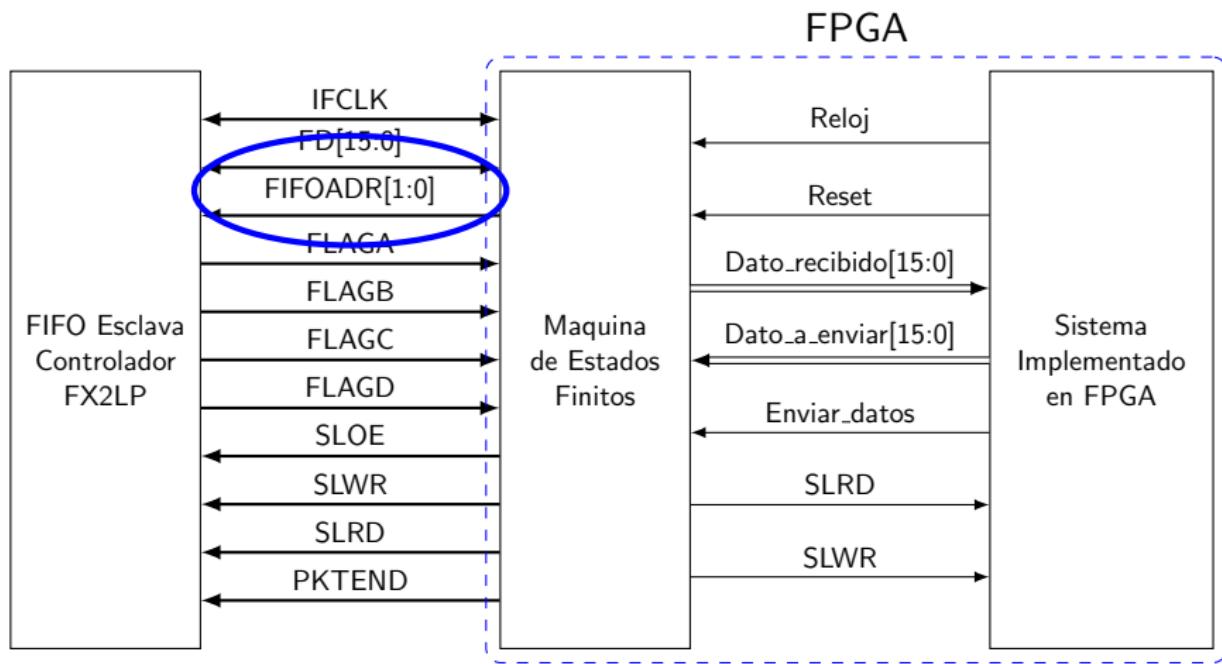
# Interfaz - FPGA



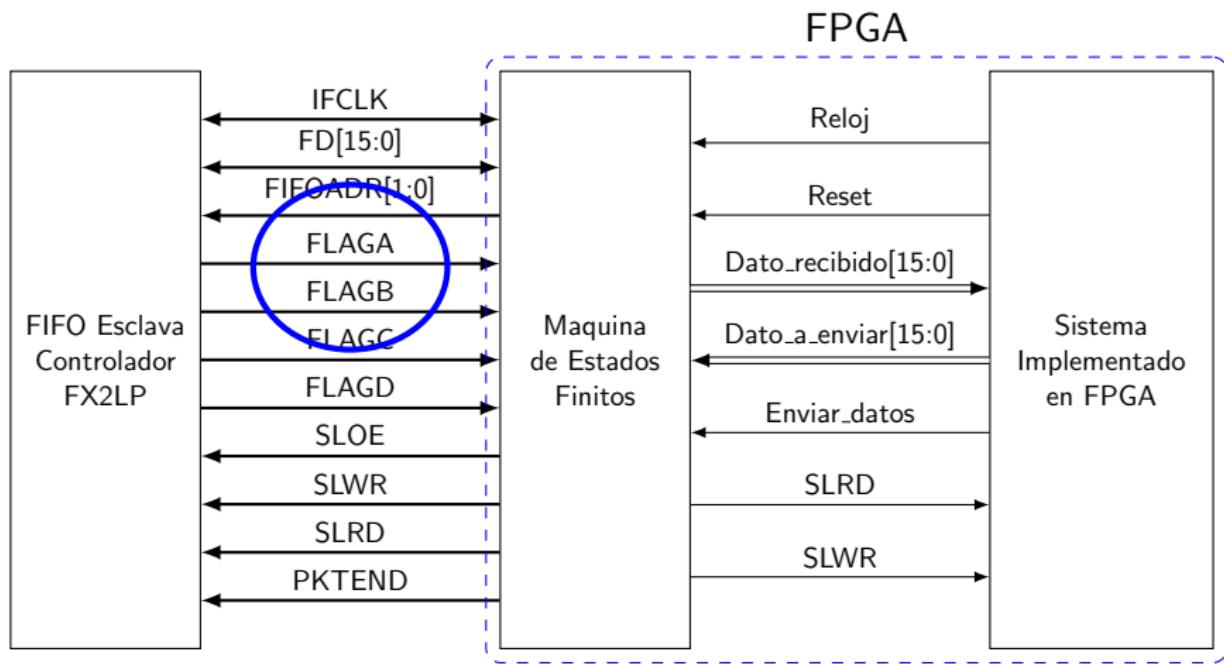
# Interfaz - FPGA



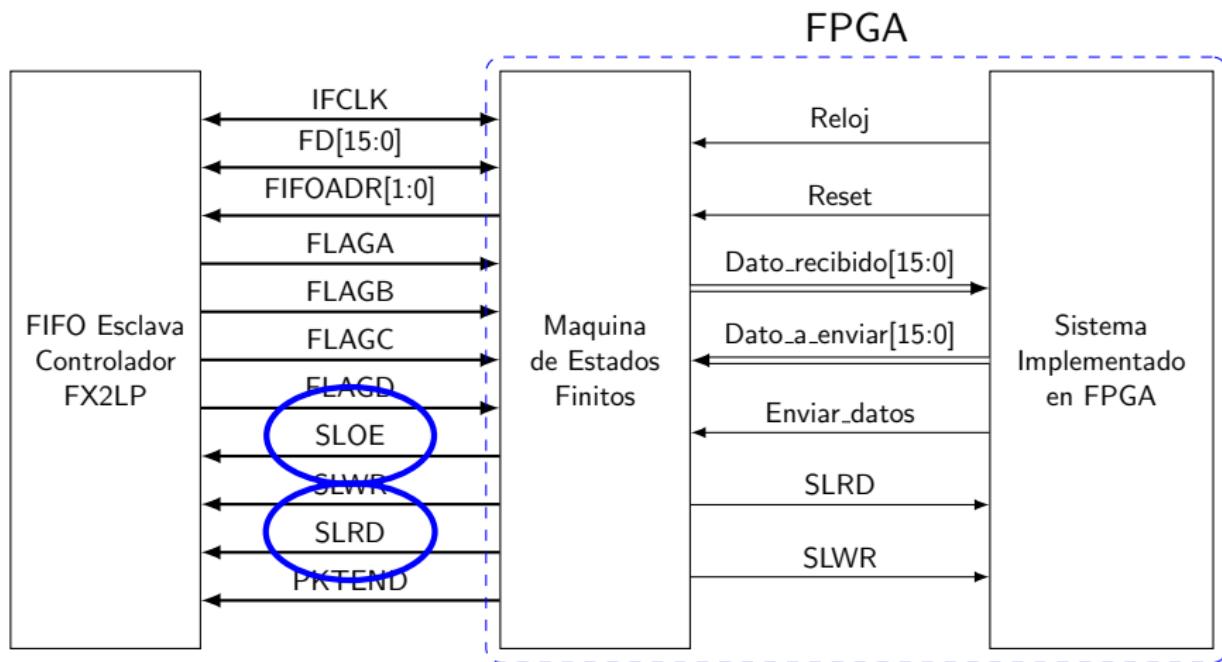
# Interfaz - FPGA



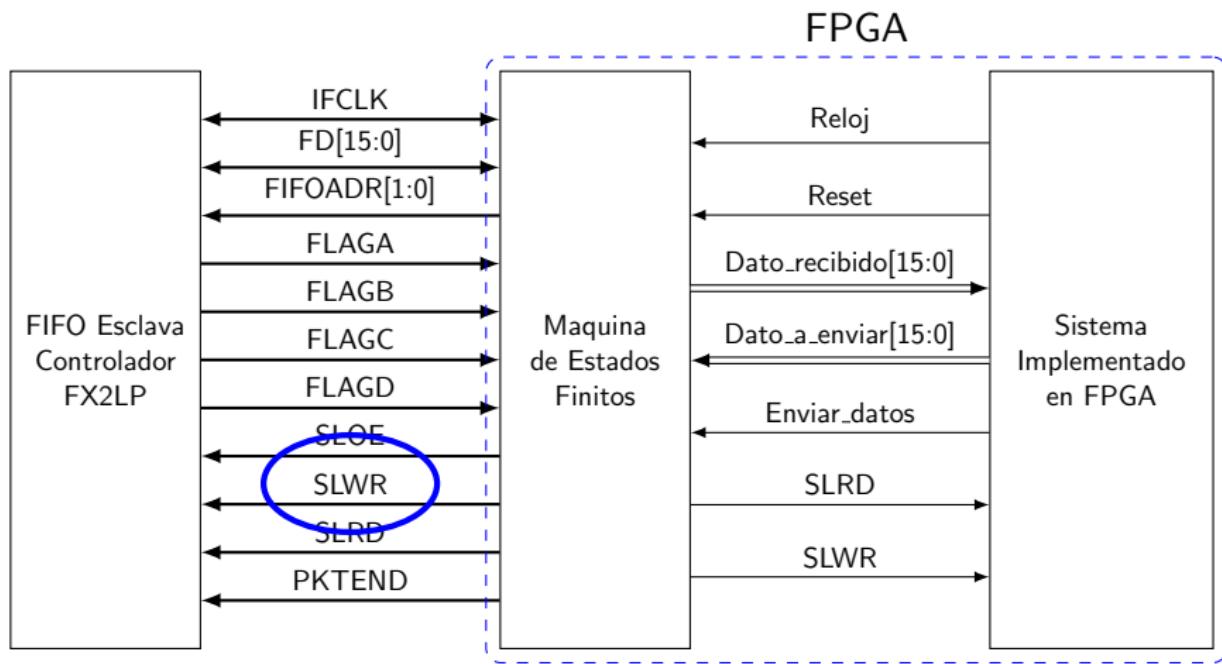
# Interfaz - FPGA



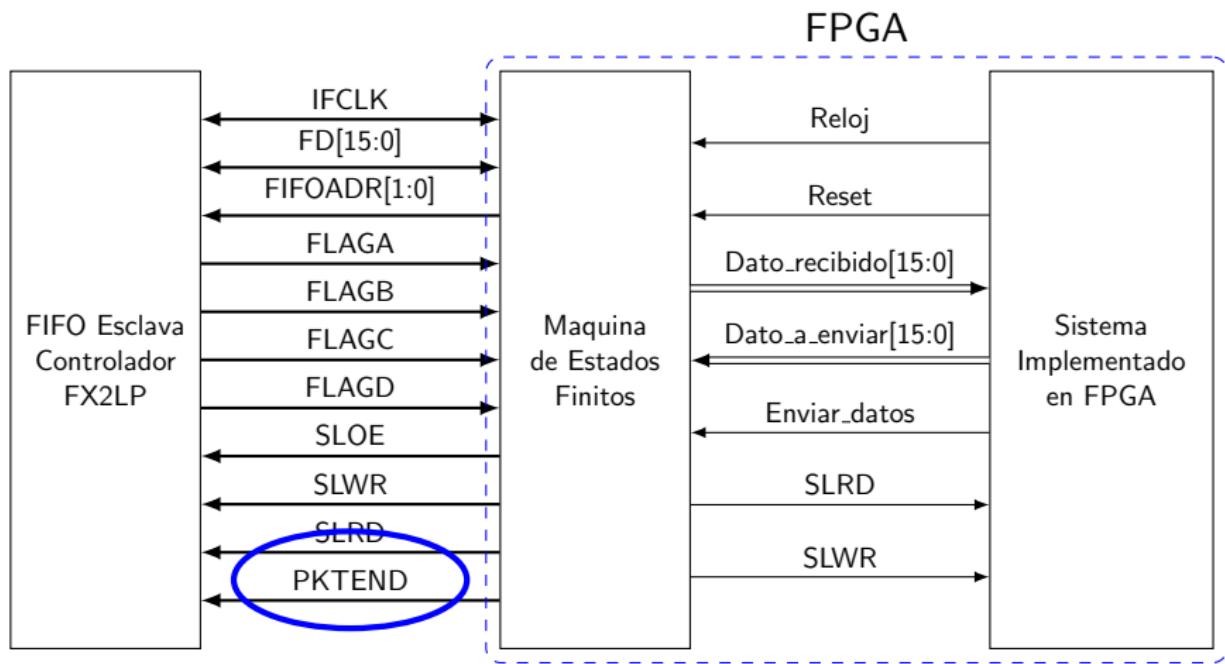
# Interfaz - FPGA



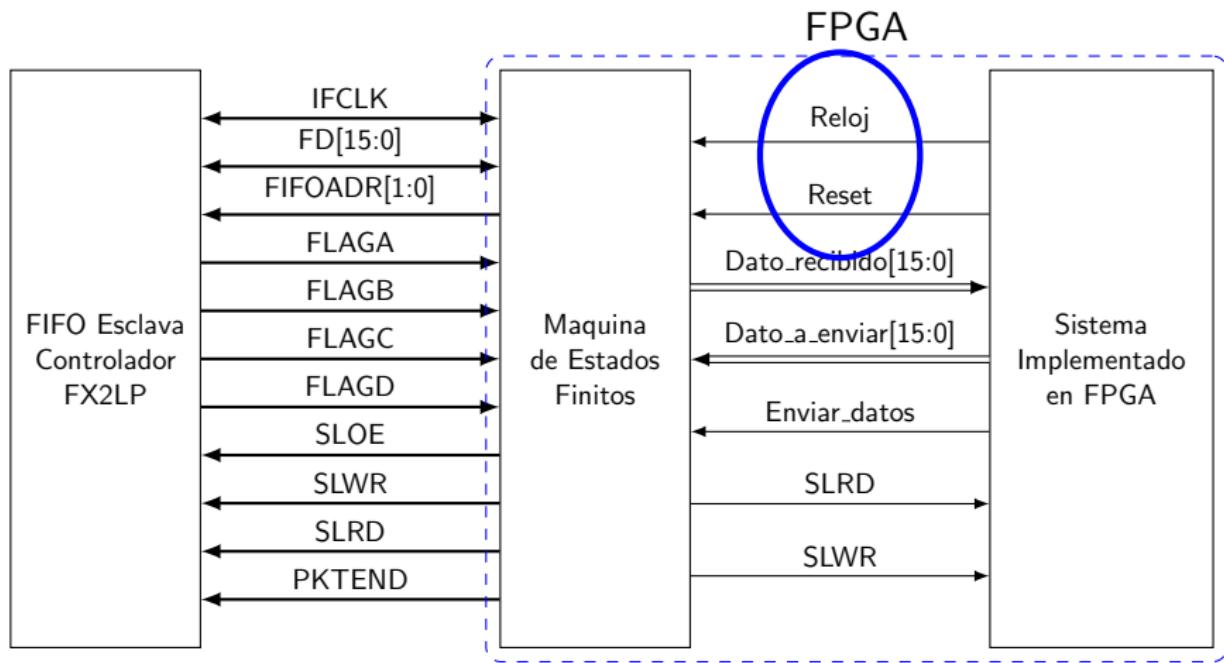
# Interfaz - FPGA



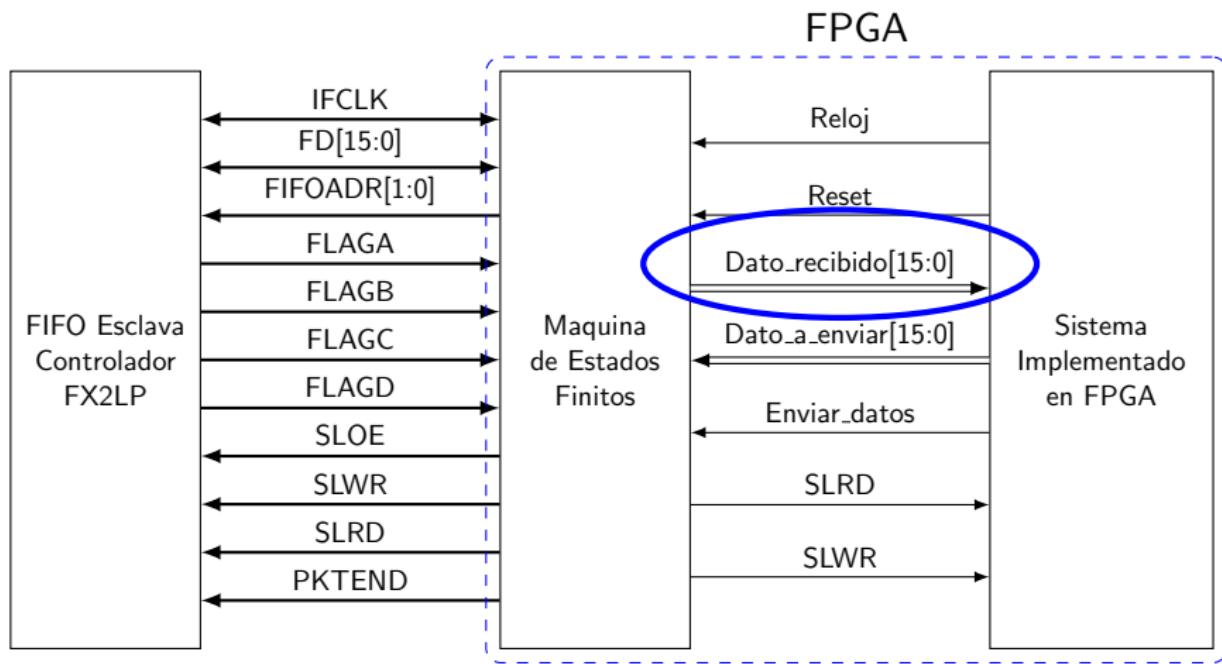
# Interfaz - FPGA



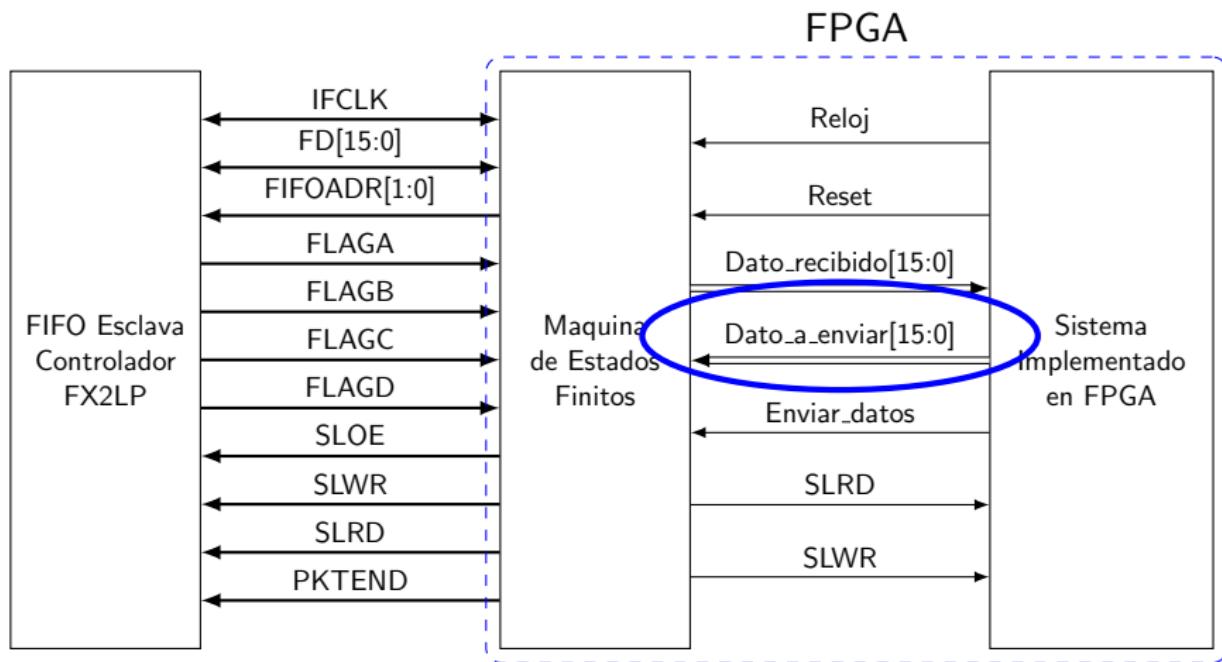
# Interfaz - FPGA



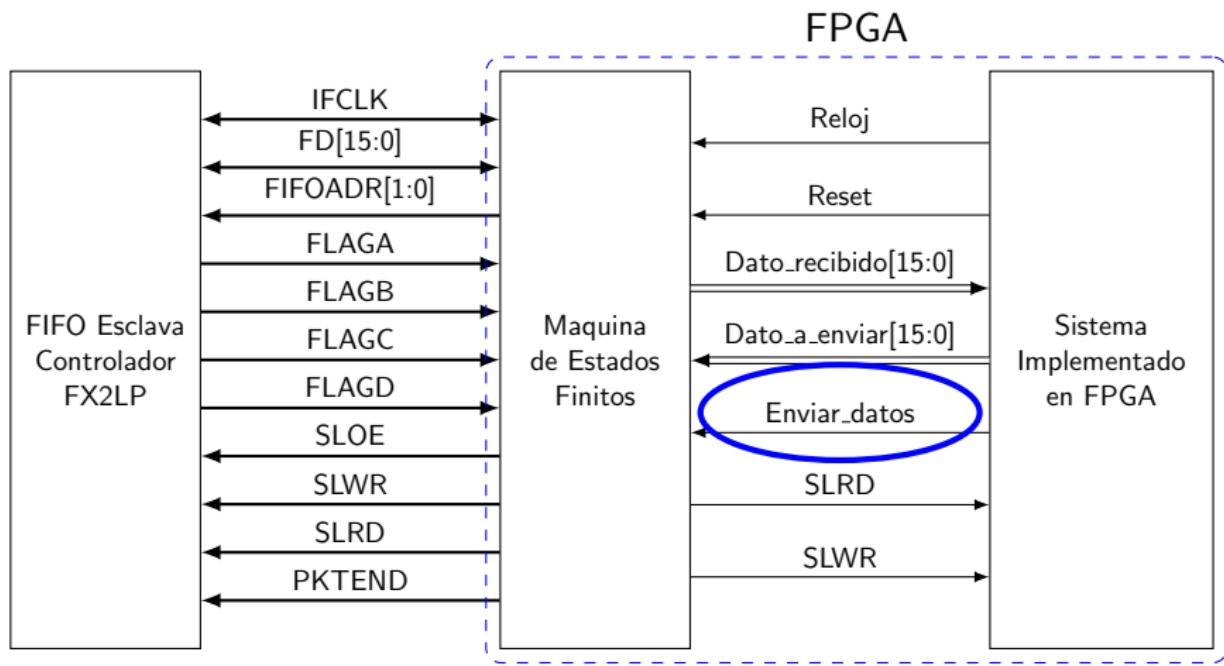
# Interfaz - FPGA



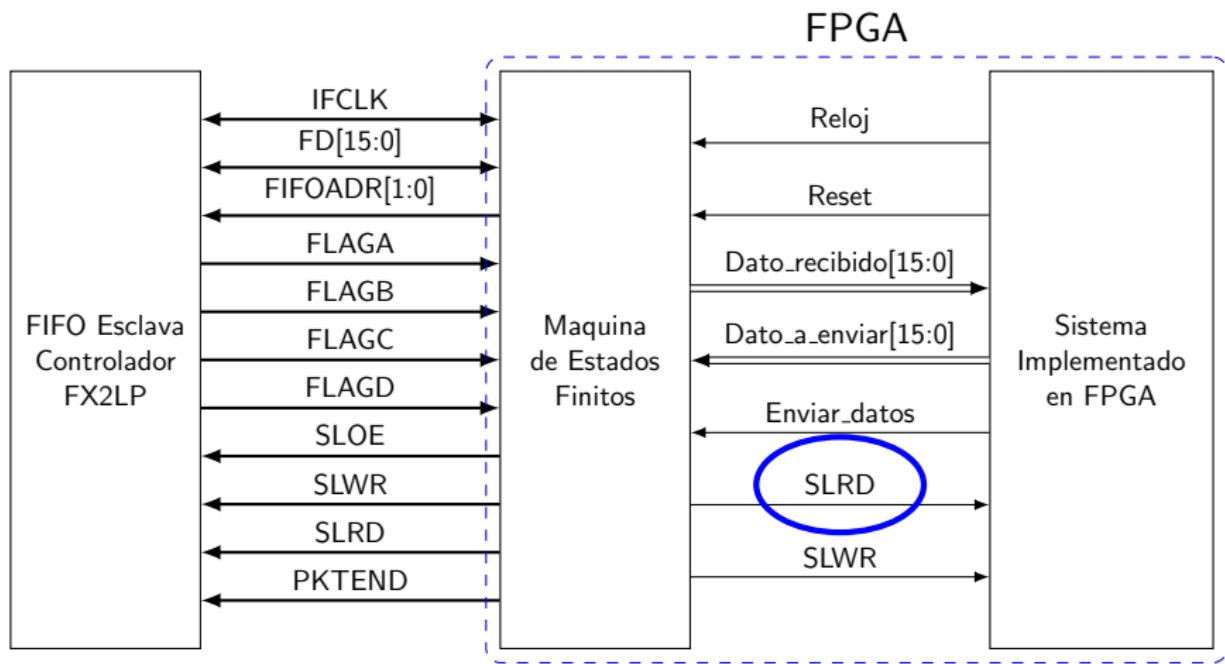
# Interfaz - FPGA



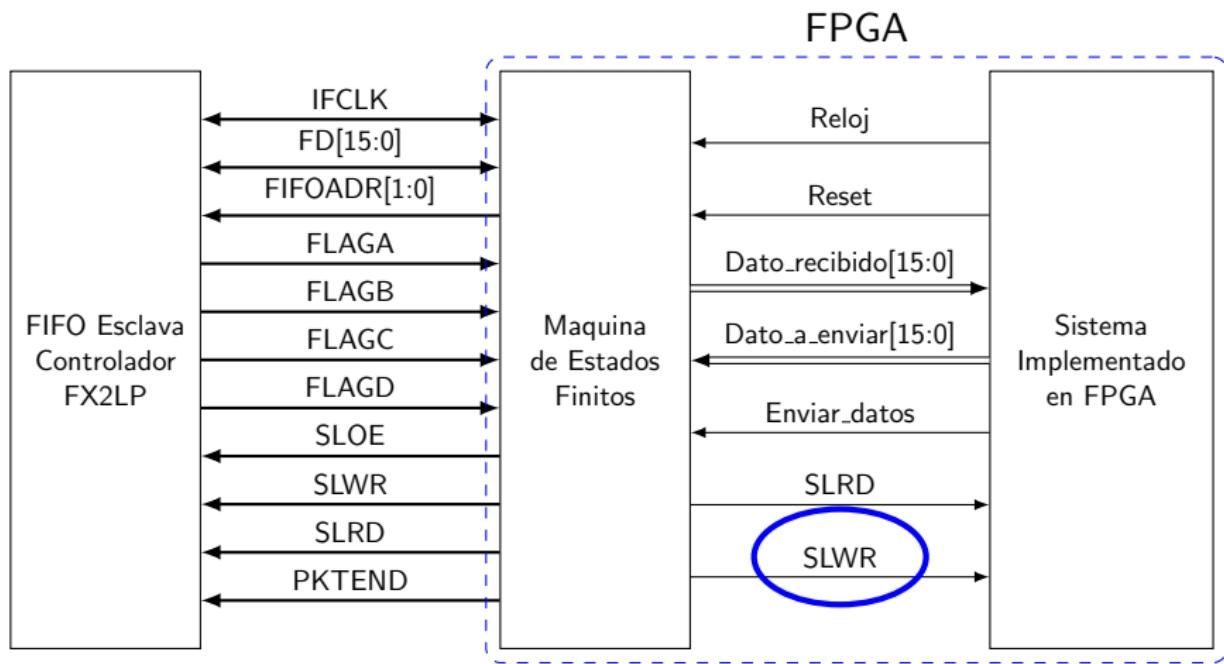
# Interfaz - FPGA



# Interfaz - FPGA

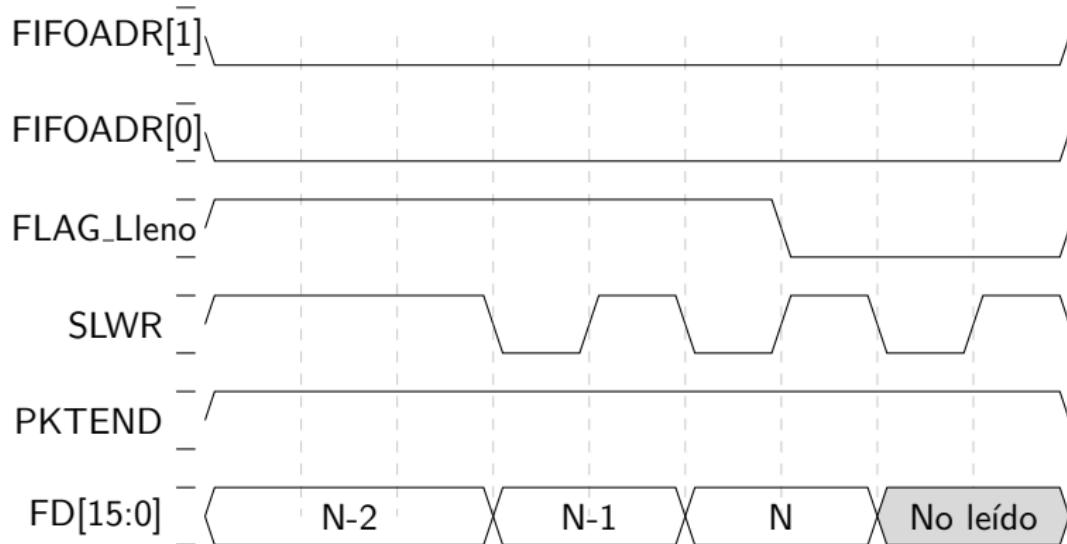


# Interfaz - FPGA



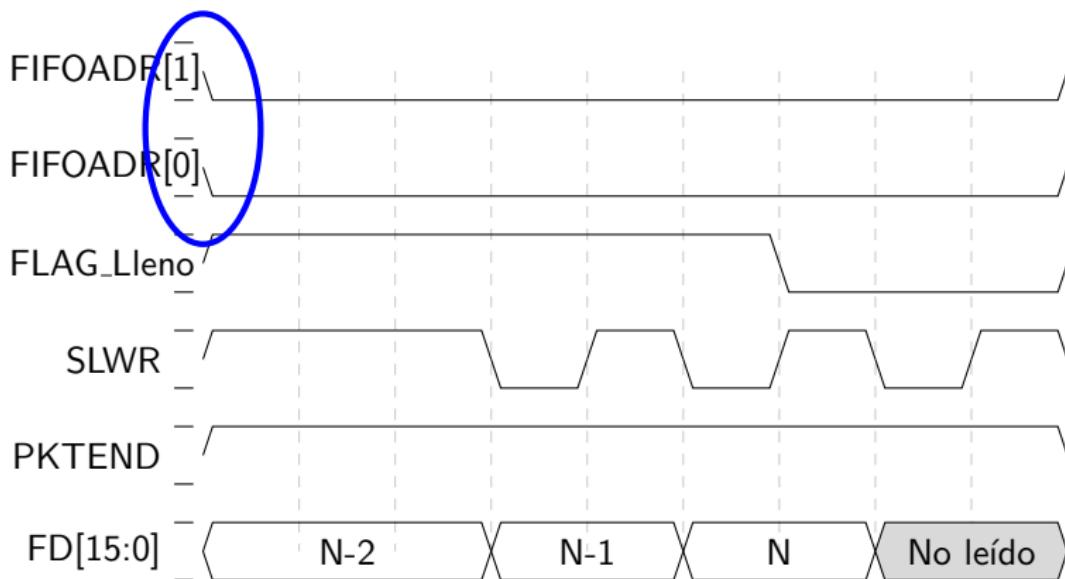
# Operaciones en la FIFO

## Escritura Asíncrona



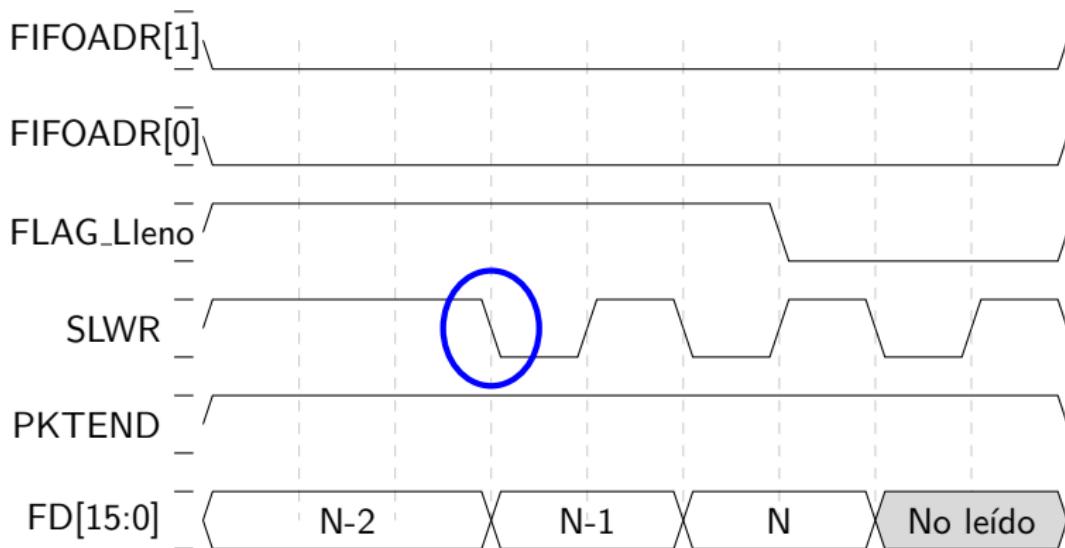
# Operaciones en la FIFO

## Escritura Asíncrona



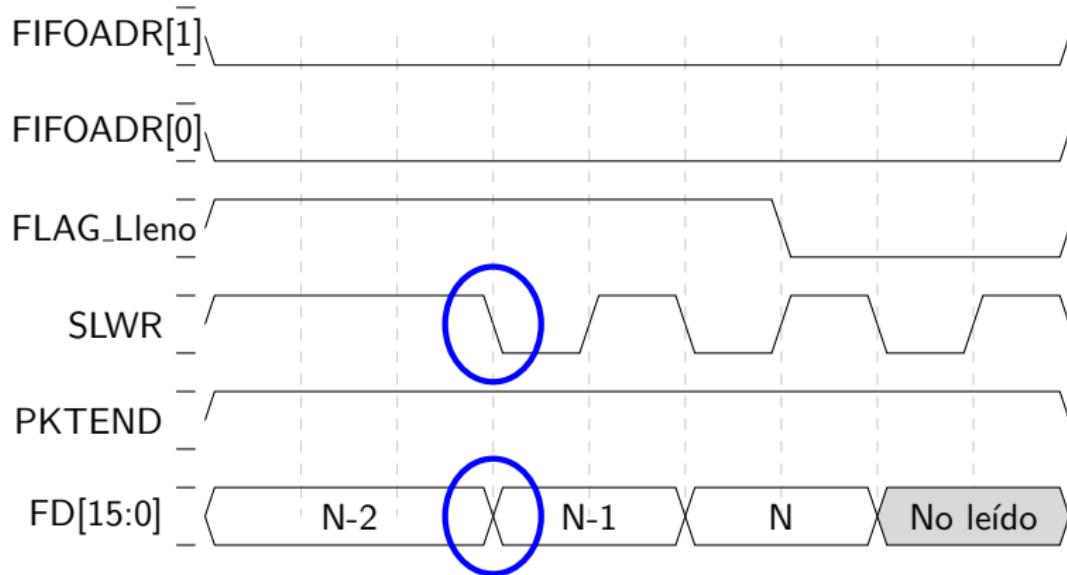
# Operaciones en la FIFO

## Escritura Asíncrona



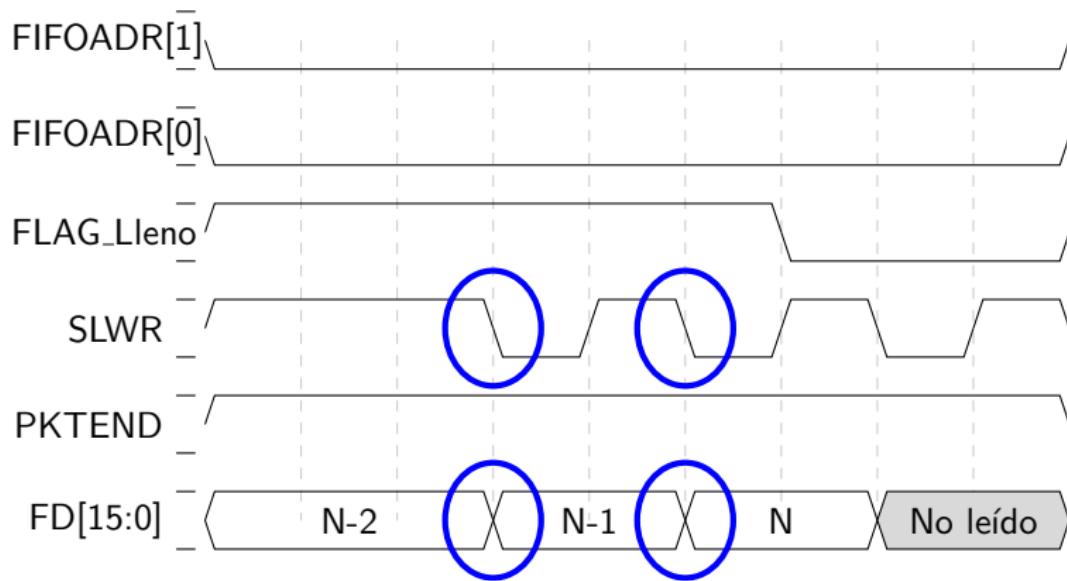
# Operaciones en la FIFO

## Escritura Asíncrona



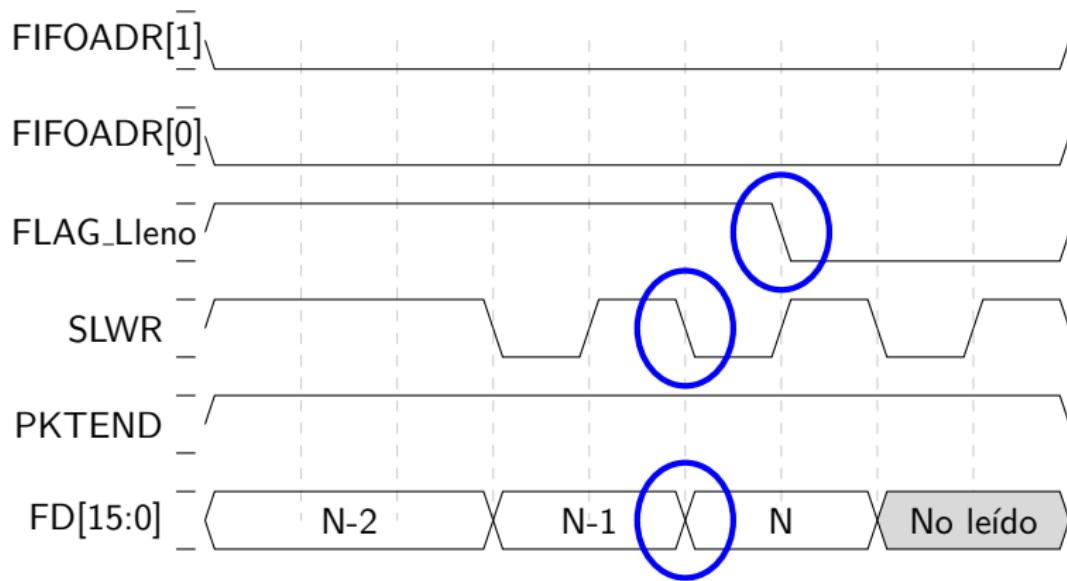
# Operaciones en la FIFO

## Escritura Asíncrona



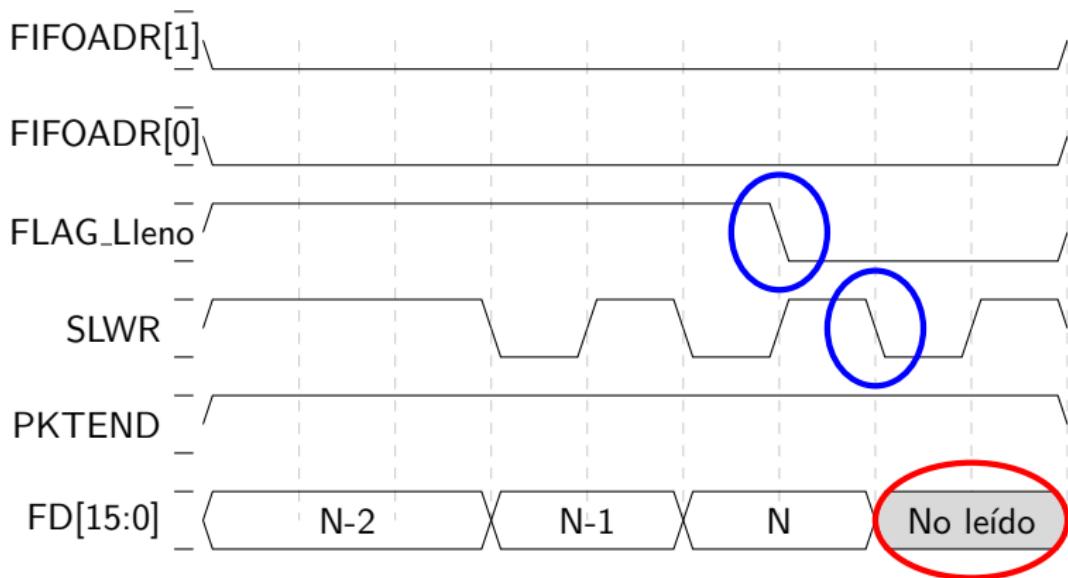
# Operaciones en la FIFO

## Escritura Asíncrona



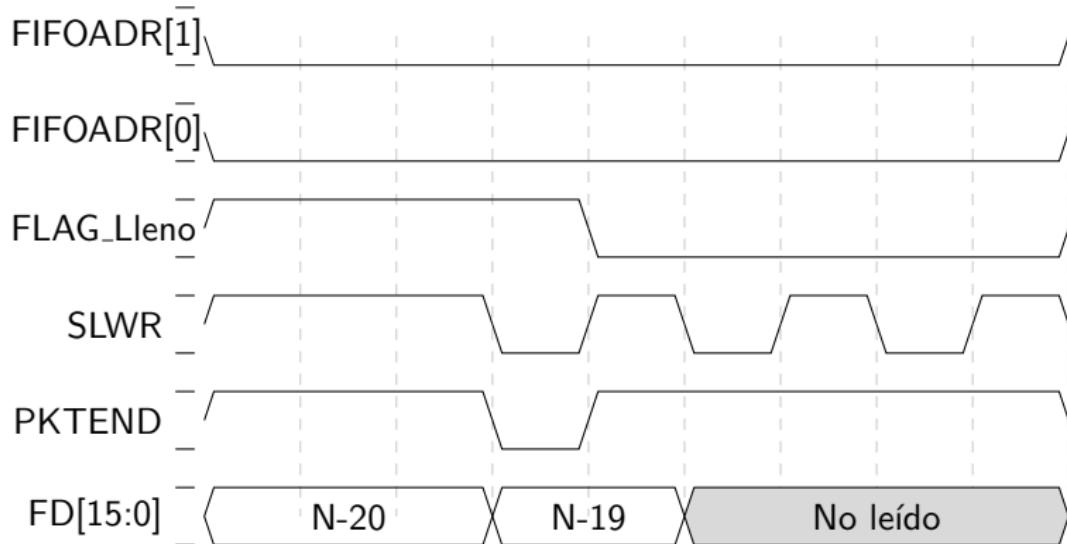
# Operaciones en la FIFO

## Escritura Asíncrona



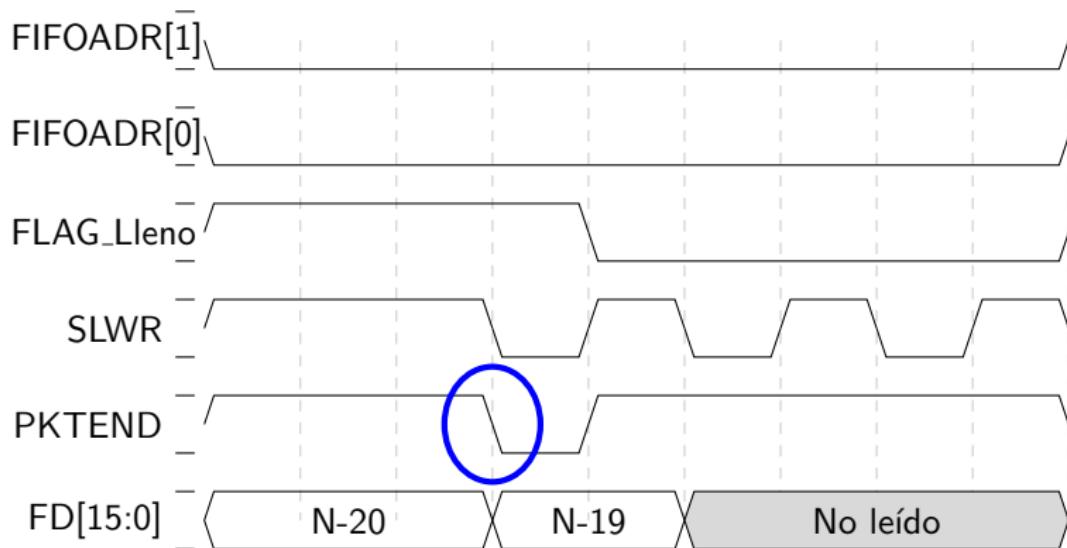
# Operaciones en la FIFO

## Escritura Asíncrona



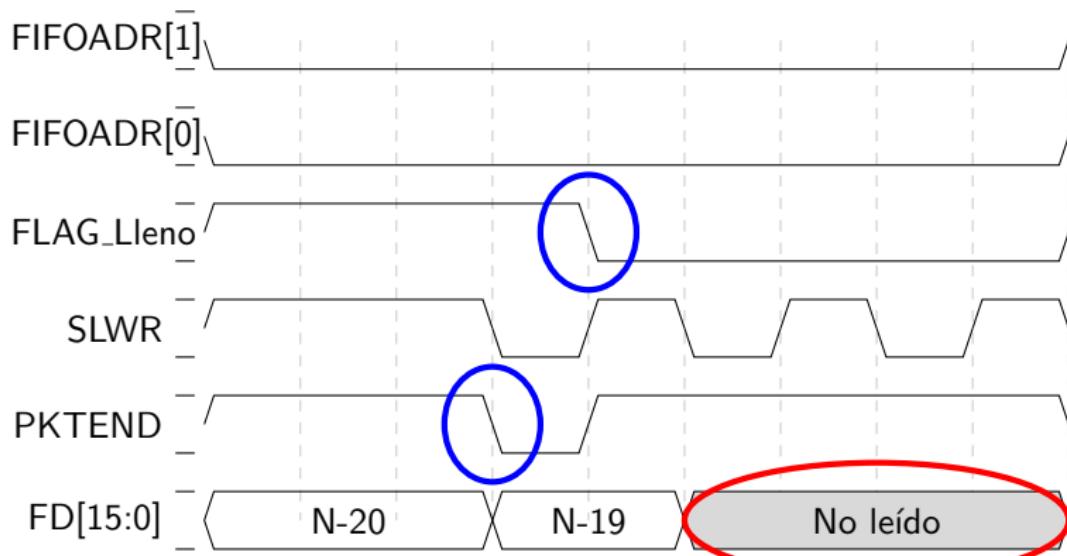
# Operaciones en la FIFO

## Escritura Asíncrona



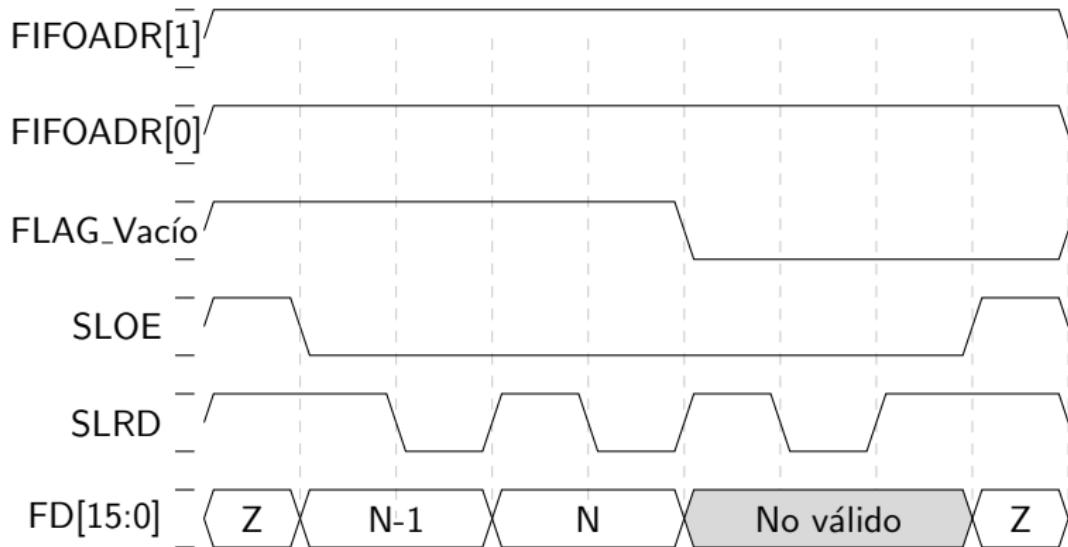
# Operaciones en la FIFO

## Escritura Asíncrona



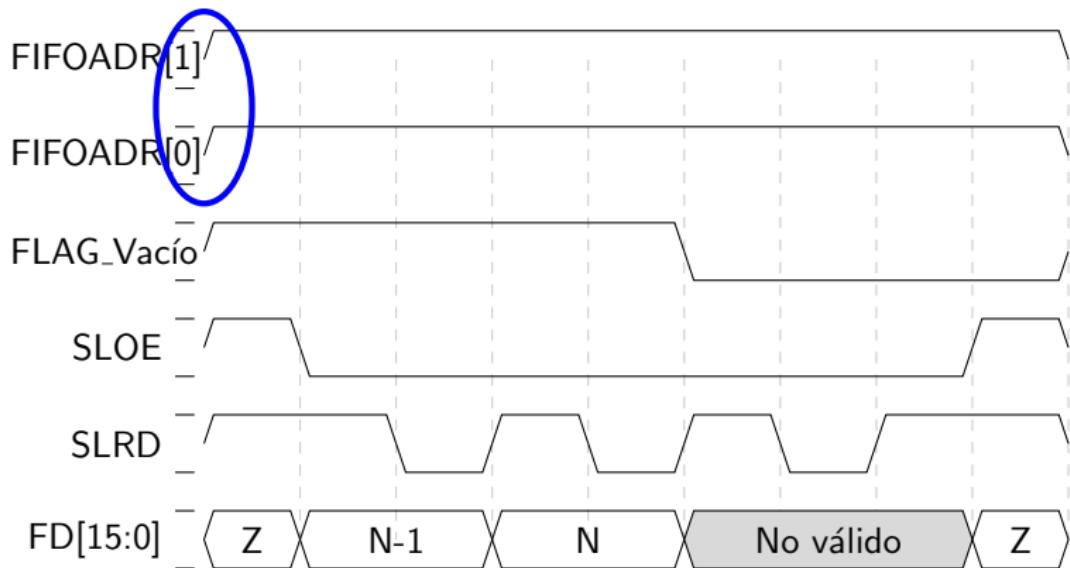
# Operaciones en la FIFO

## Lectura Asíncrona



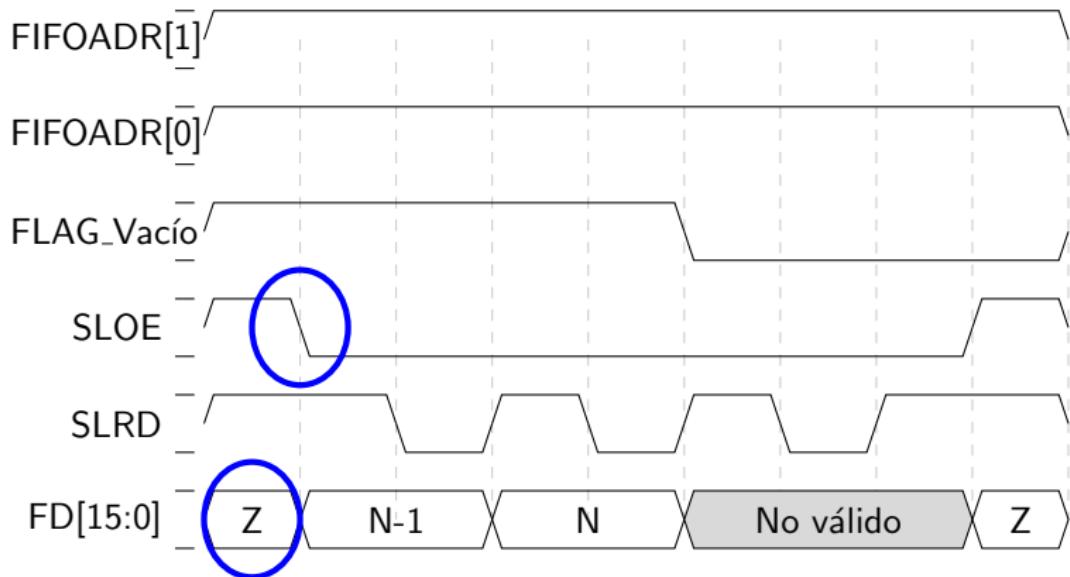
# Operaciones en la FIFO

## Lectura Asíncrona



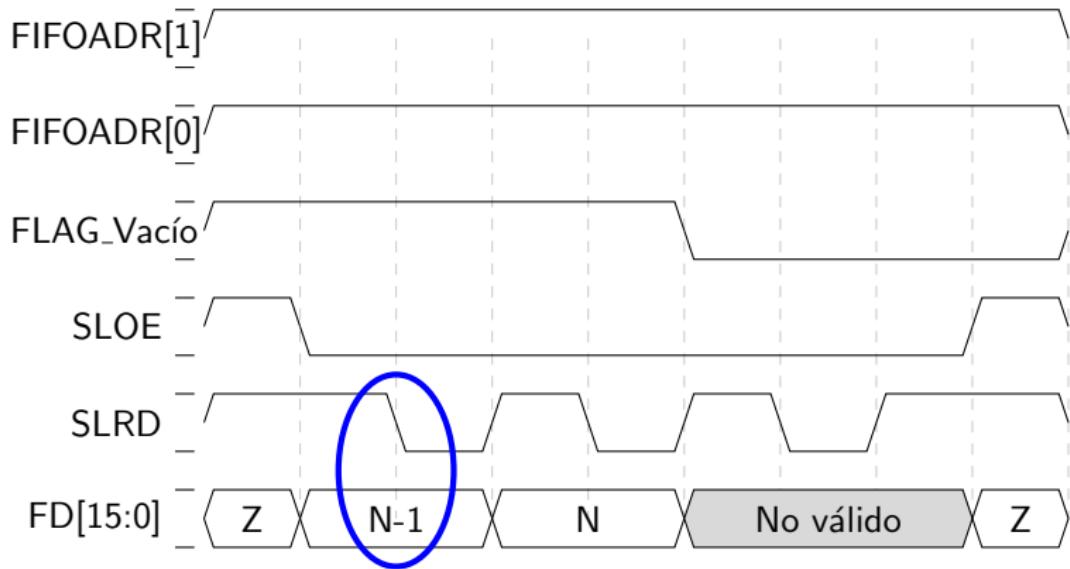
# Operaciones en la FIFO

## Lectura Asíncrona



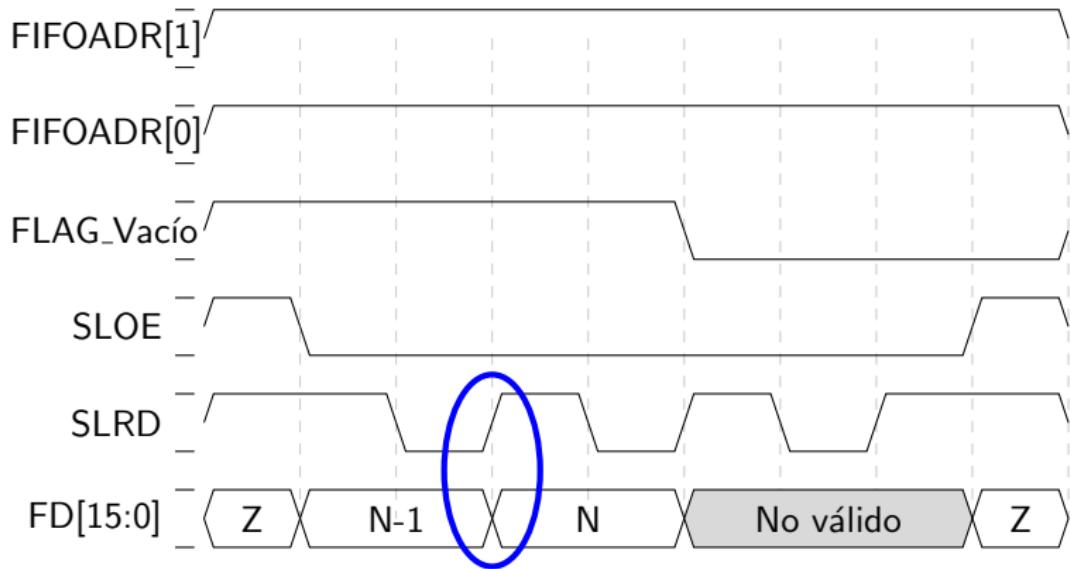
# Operaciones en la FIFO

## Lectura Asíncrona



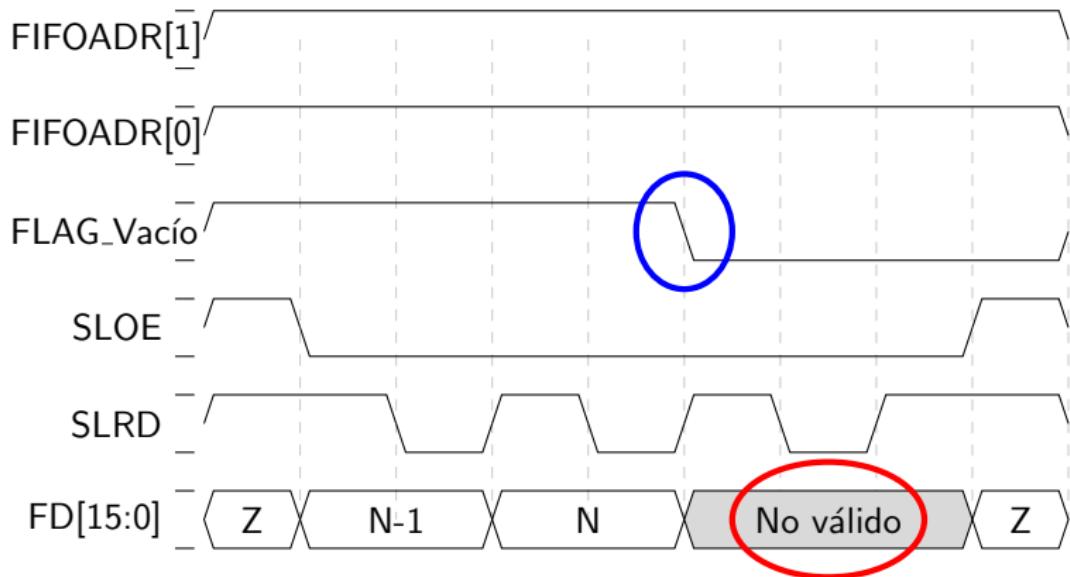
# Operaciones en la FIFO

## Lectura Asíncrona

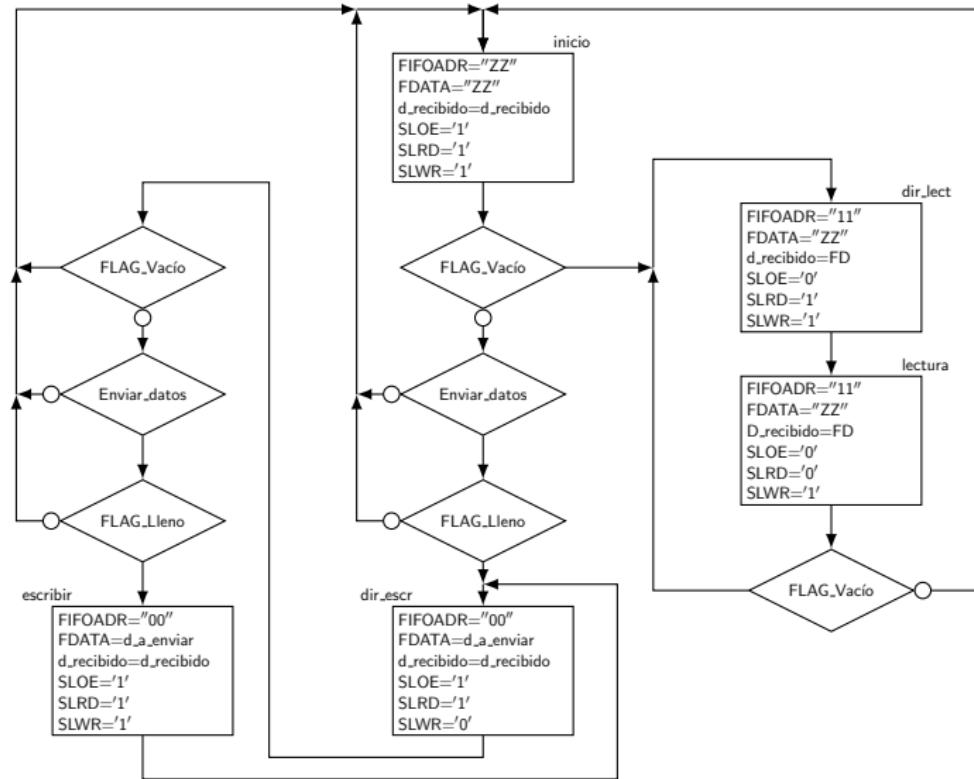


# Operaciones en la FIFO

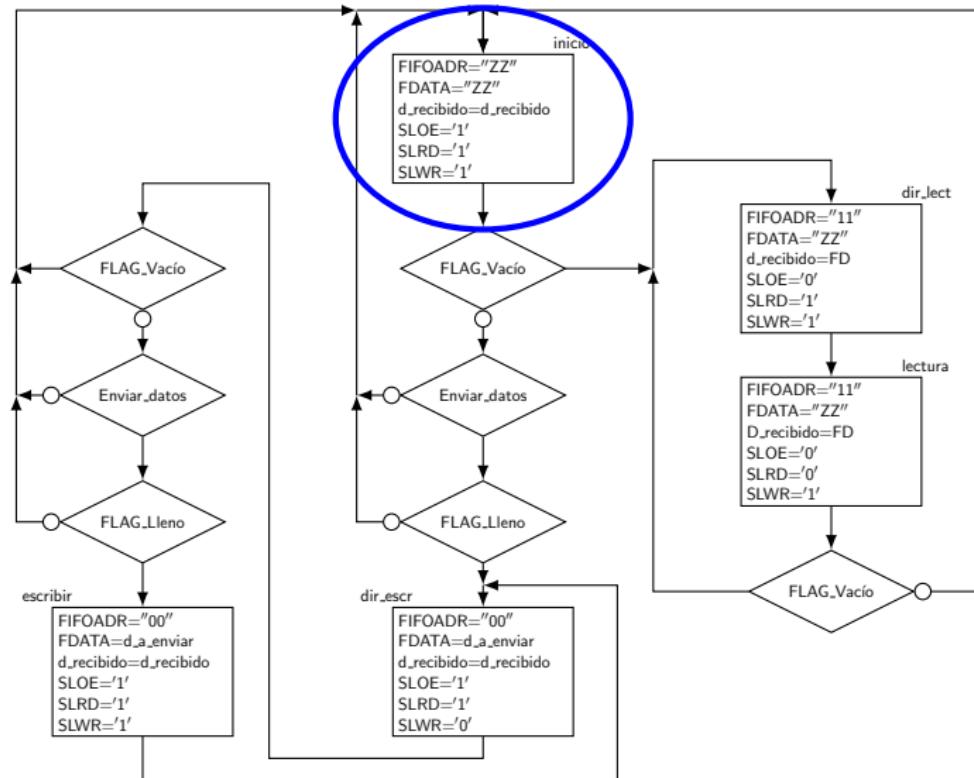
## Lectura Asíncrona



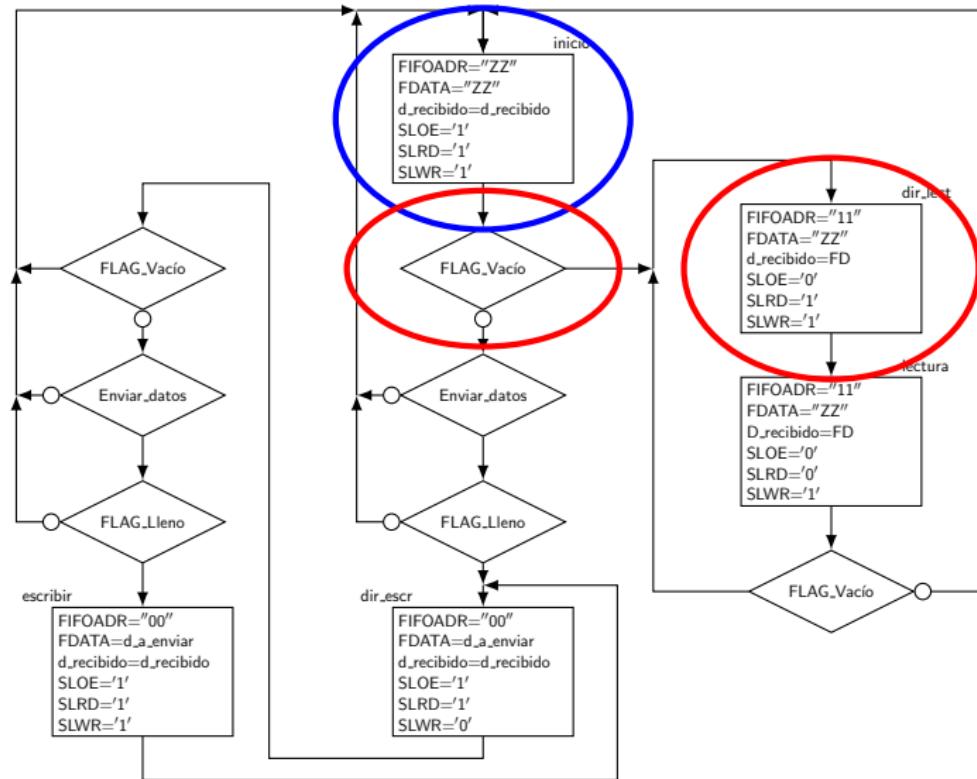
# Máquina de estados algorítmica de la interfaz



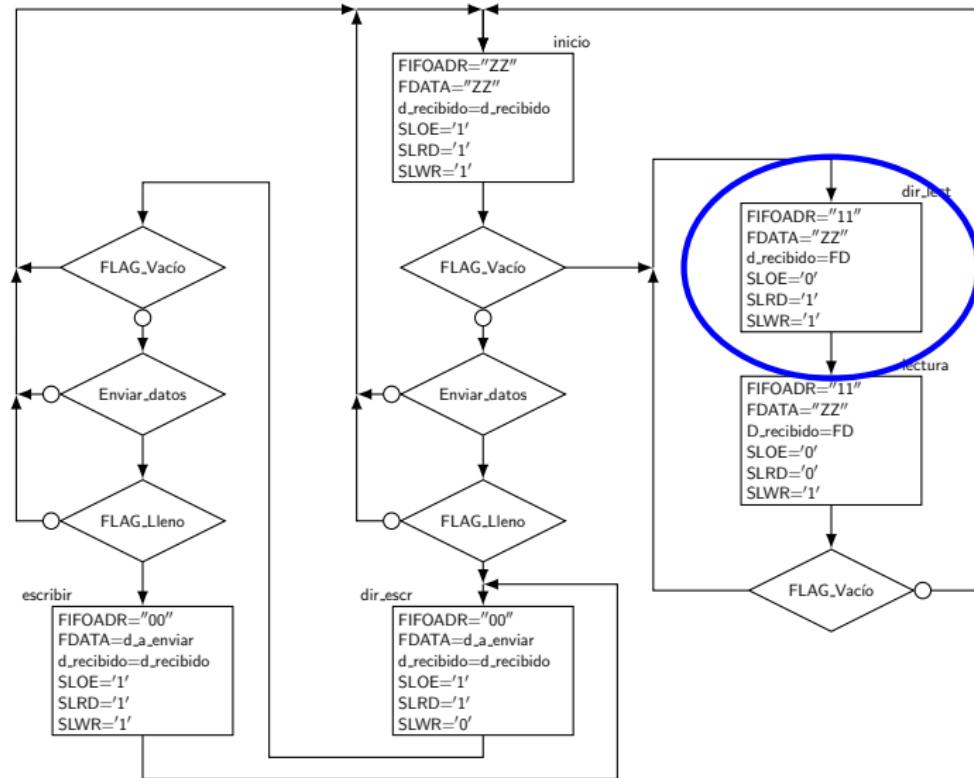
# Máquina de estados algorítmica de la interfaz



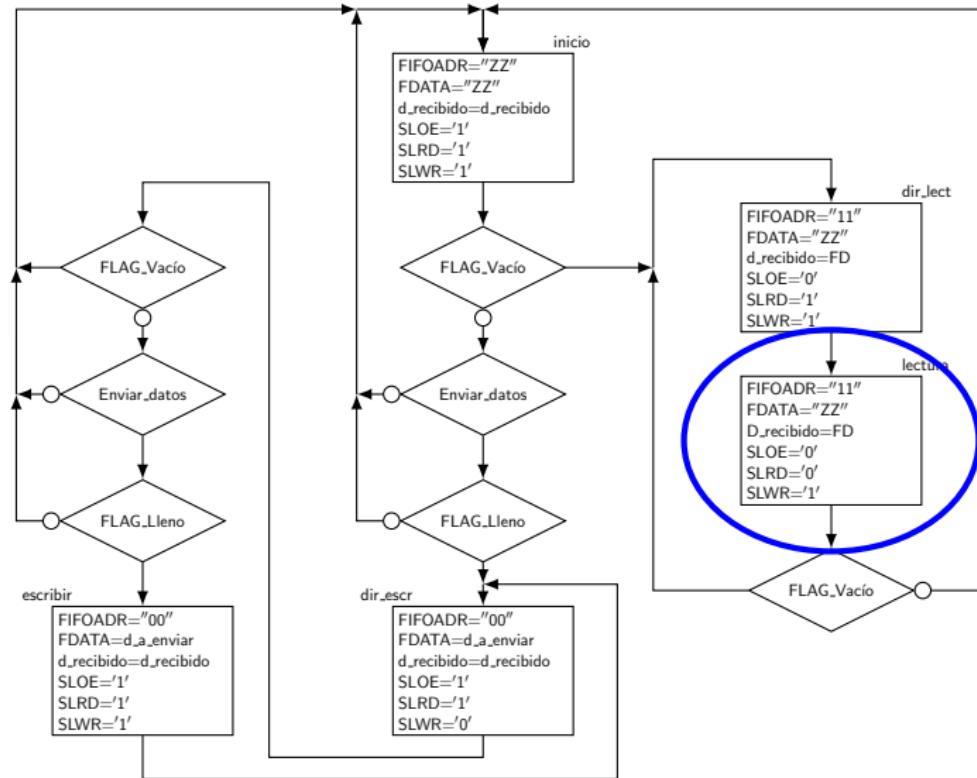
# Máquina de estados algorítmica de la interfaz



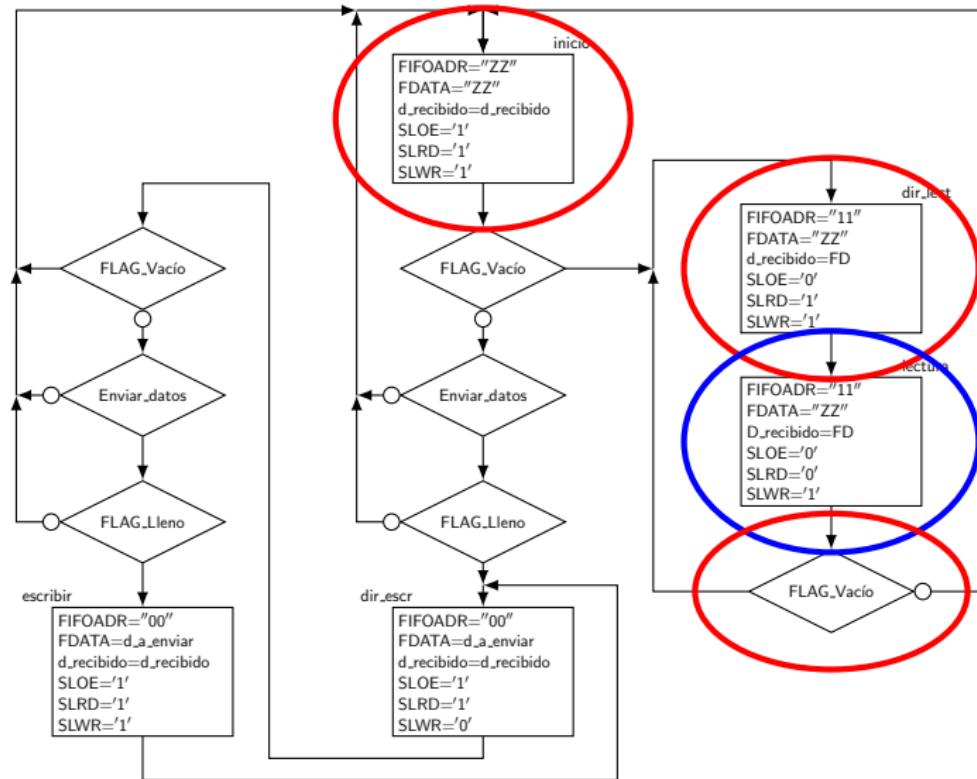
# Máquina de estados algorítmica de la interfaz



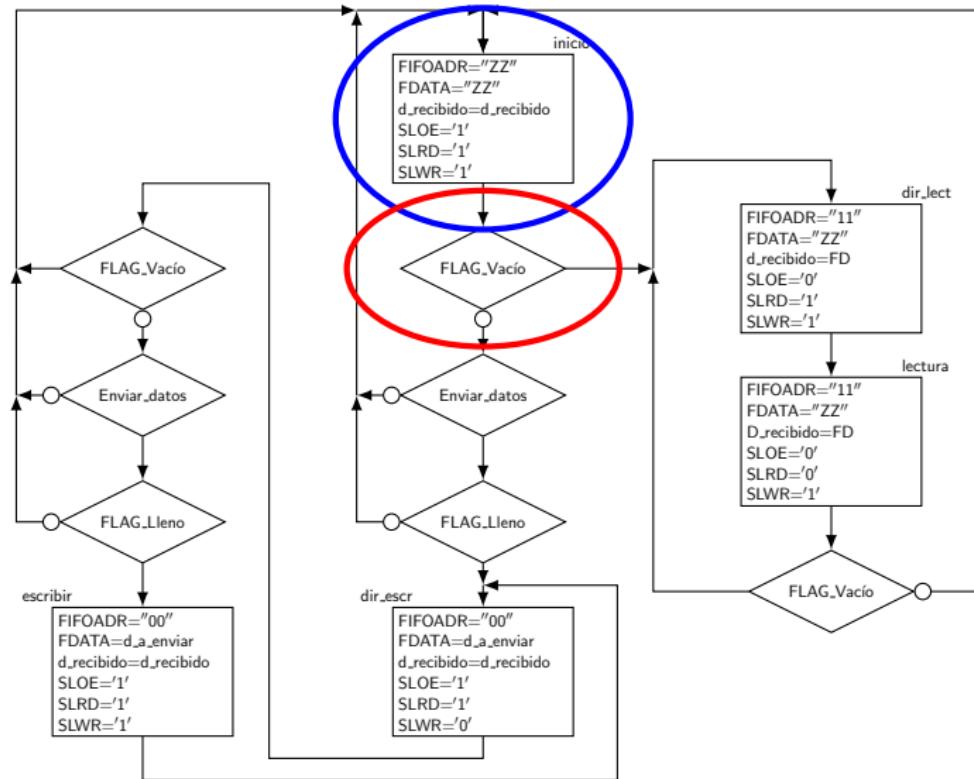
# Máquina de estados algorítmica de la interfaz



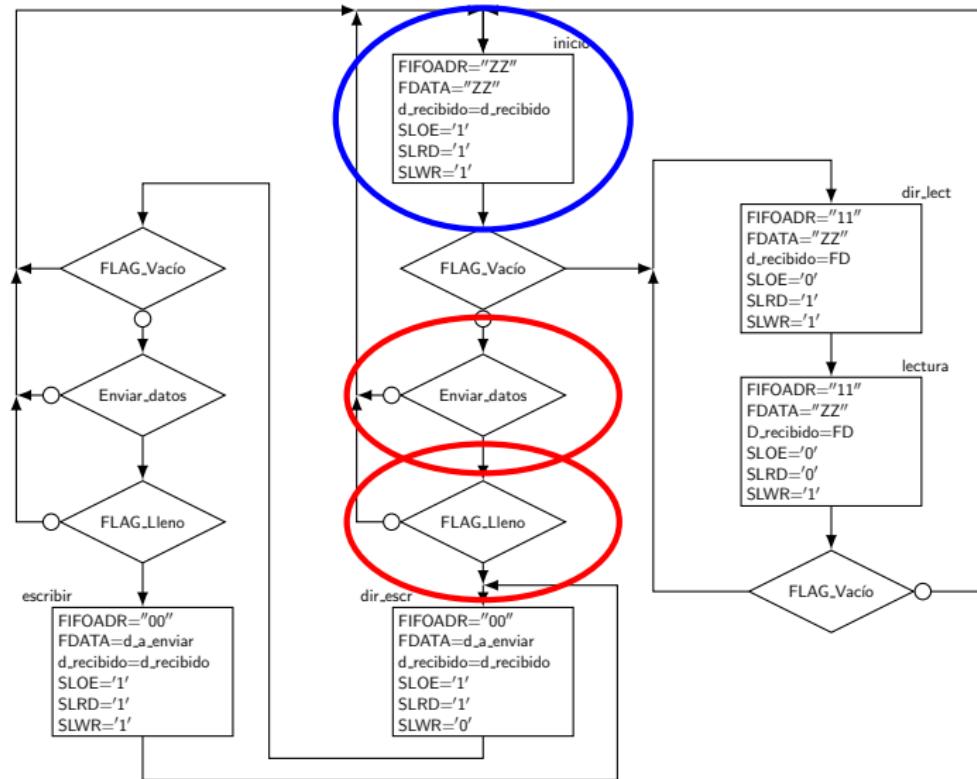
# Máquina de estados algorítmica de la interfaz



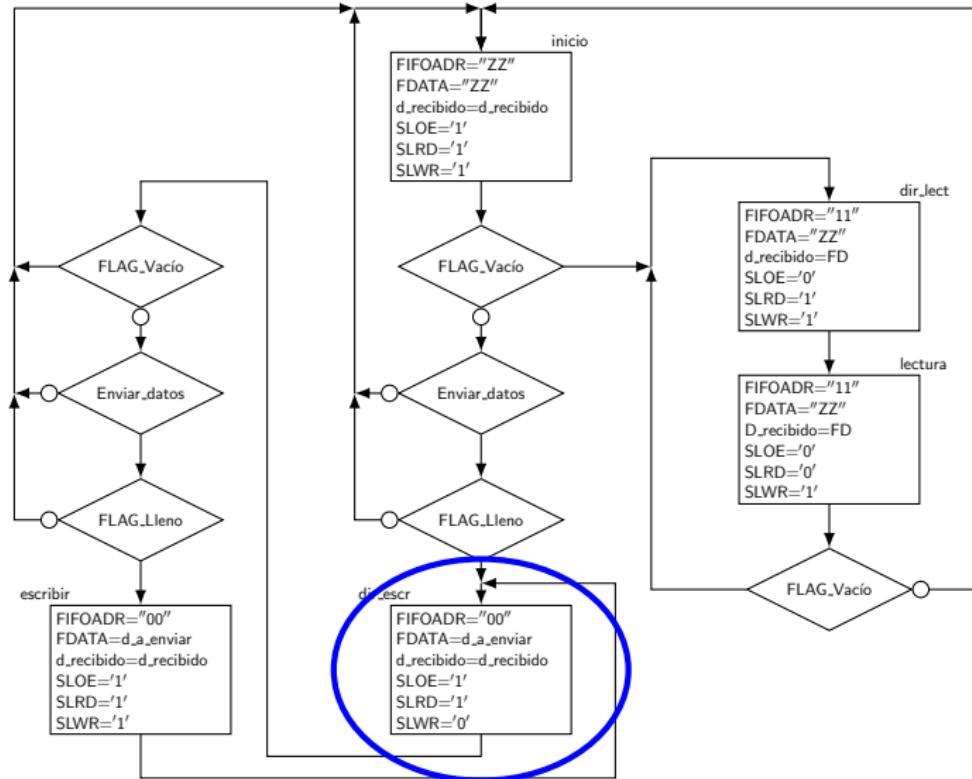
# Máquina de estados algorítmica de la interfaz



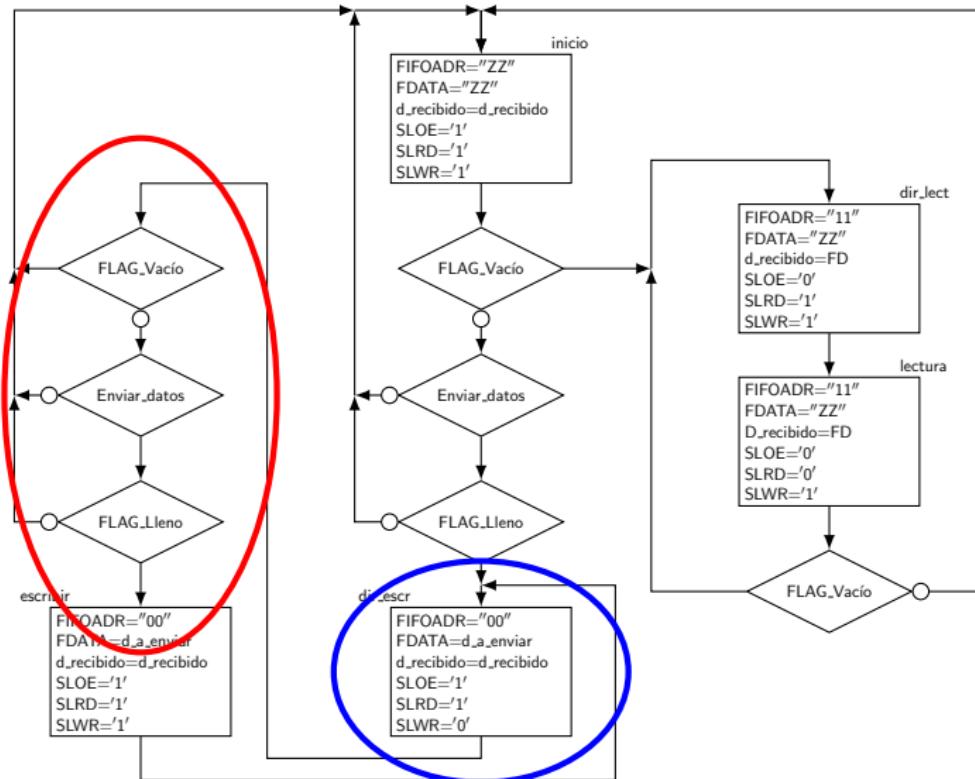
# Máquina de estados algorítmica de la interfaz



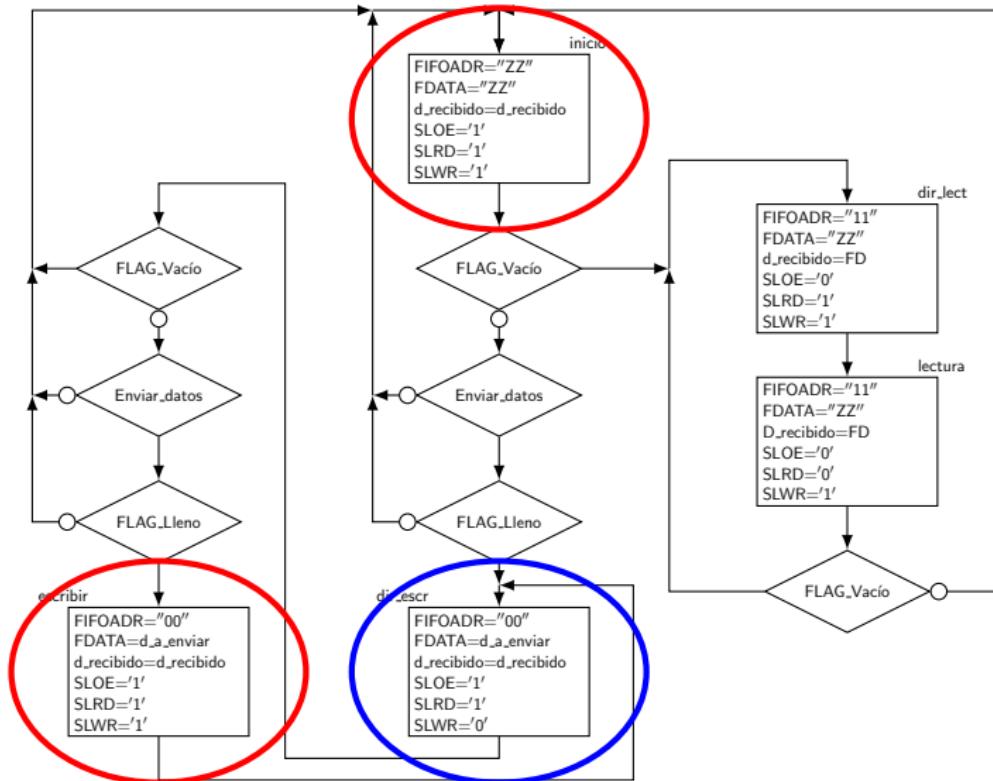
# Máquina de estados algorítmica de la interfaz



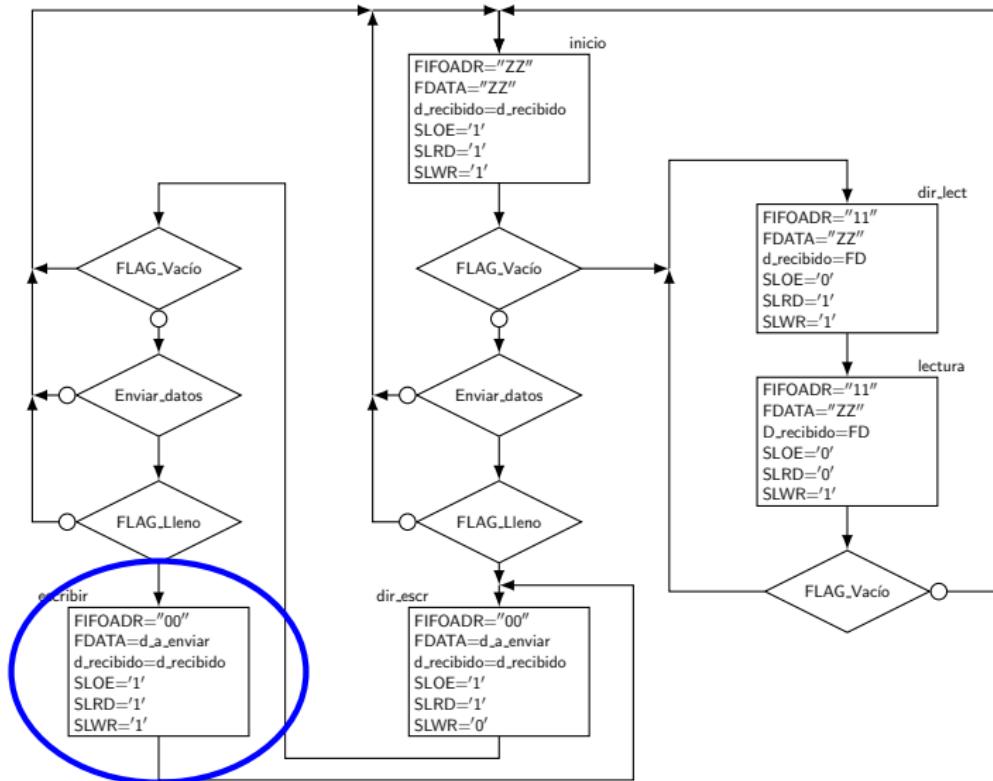
# Máquina de estados algorítmica de la interfaz



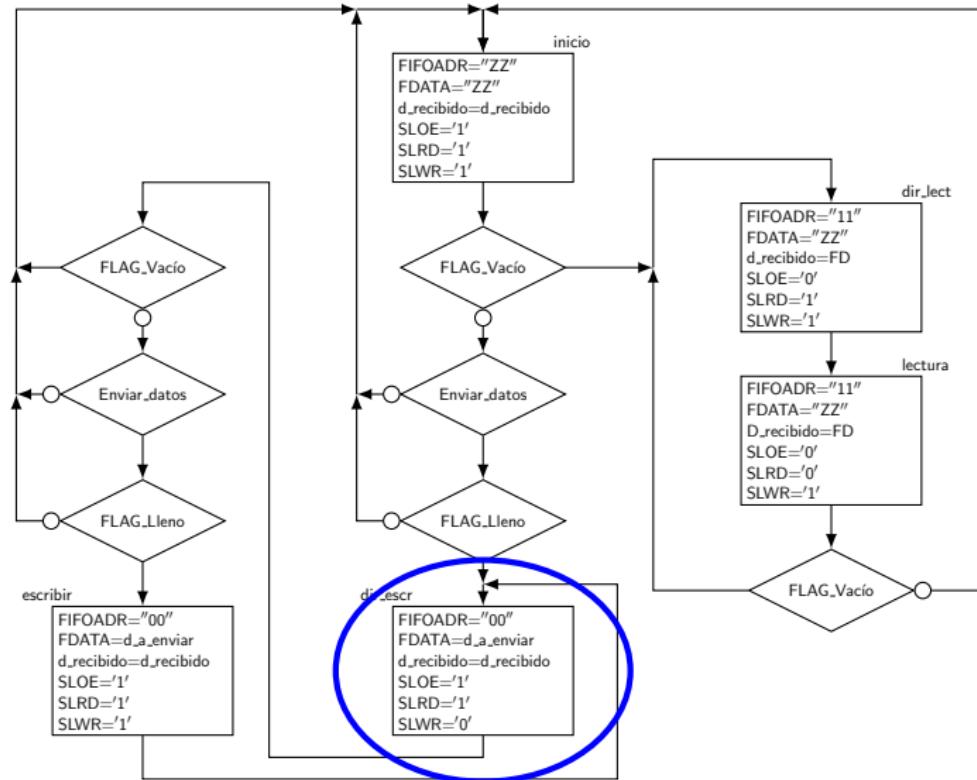
# Máquina de estados algorítmica de la interfaz



# Máquina de estados algorítmica de la interfaz



# Máquina de estados algorítmica de la interfaz



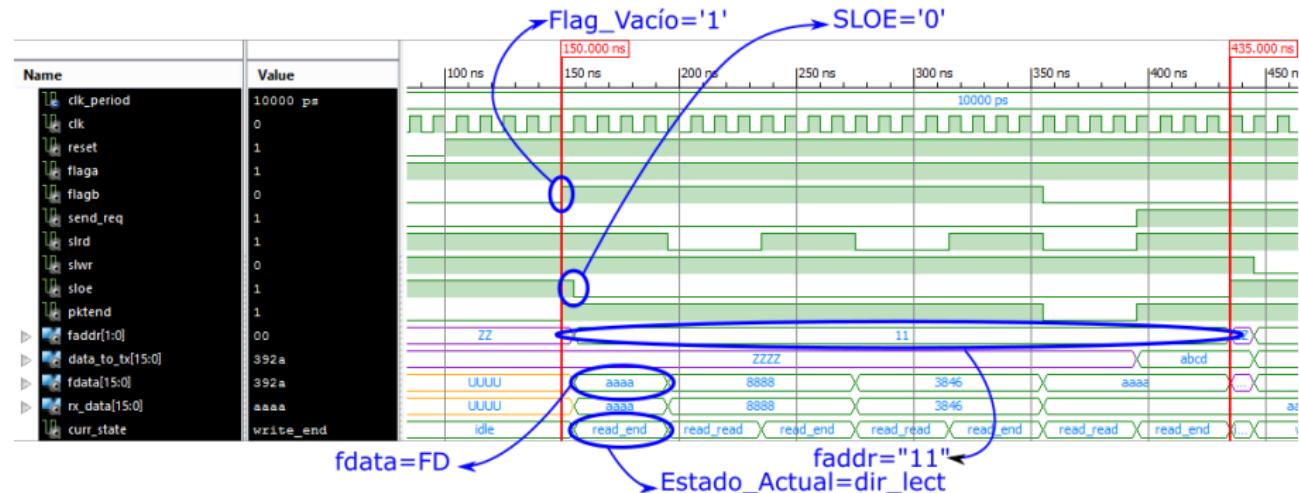
# Verificación Funcional

## Operación de lectura



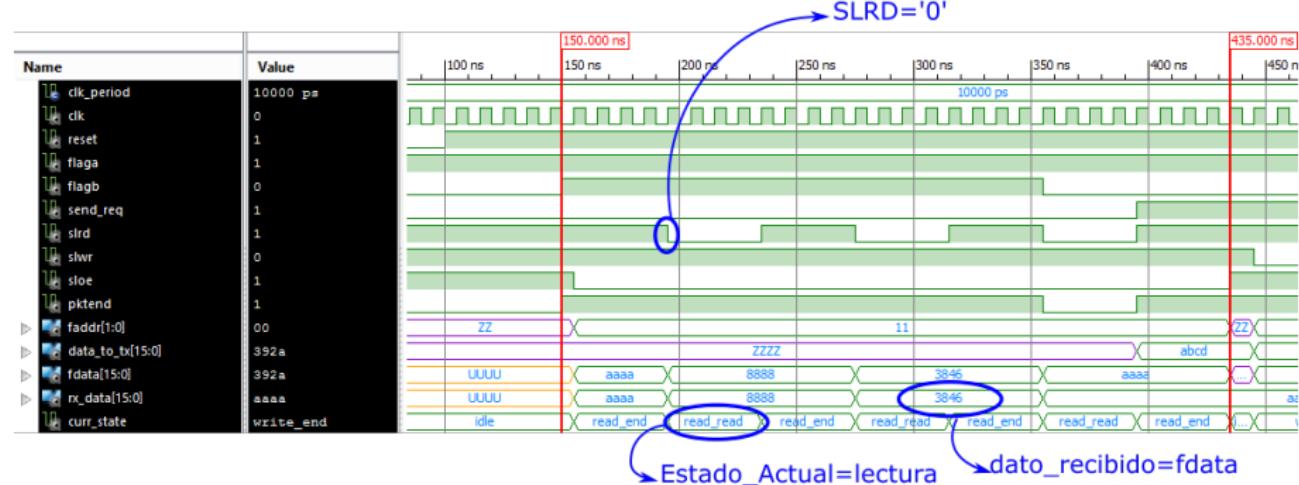
# Verificación Funcional

## Operación de lectura



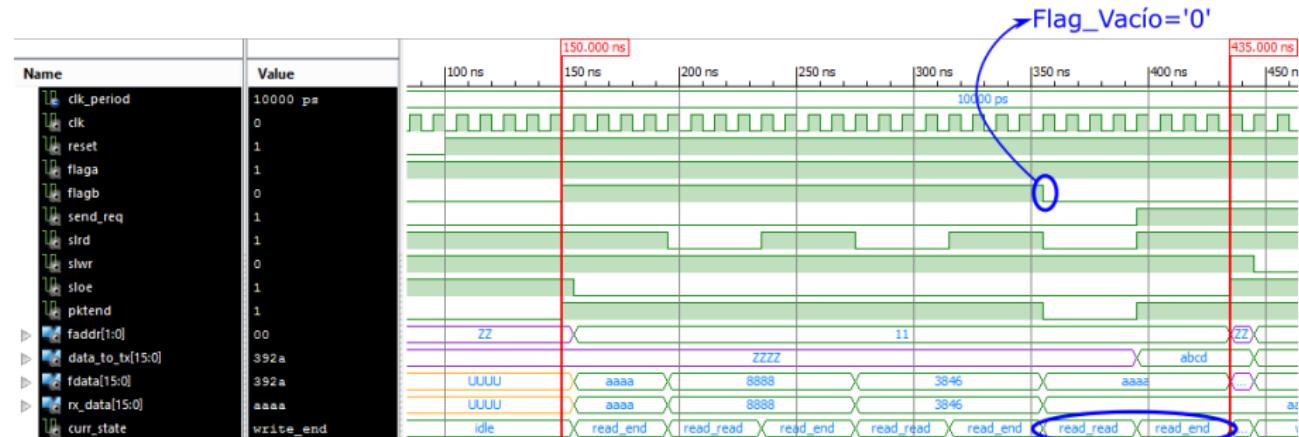
# Verificación Funcional

## Operación de lectura



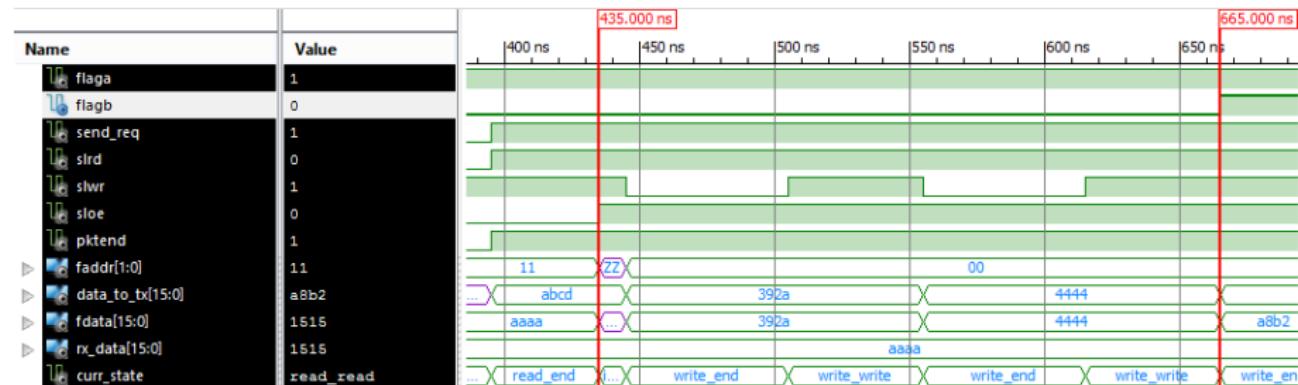
# Verificación Funcional

## Operación de lectura



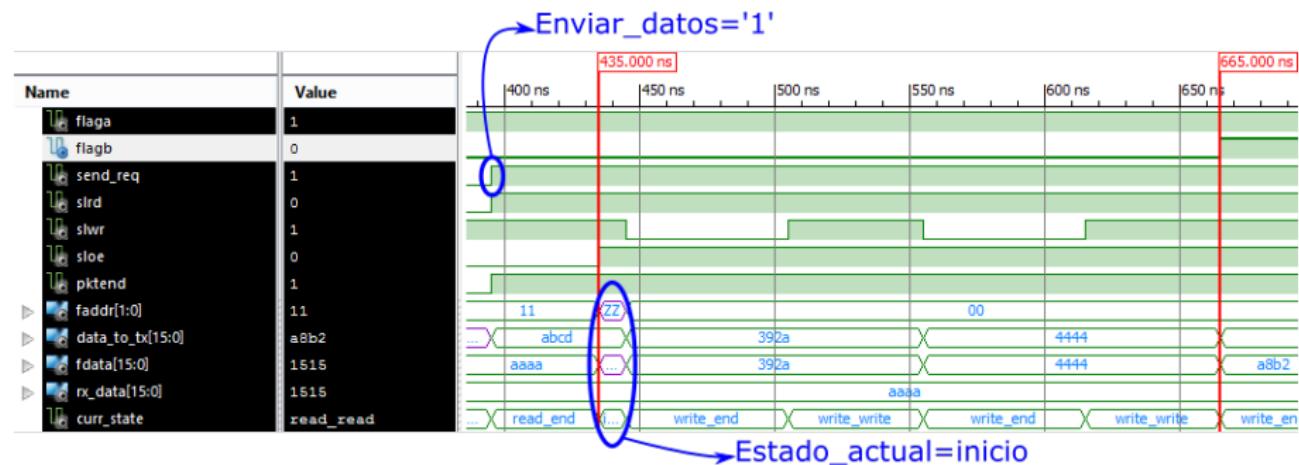
# Verificación Funcional

## Operación de escritura



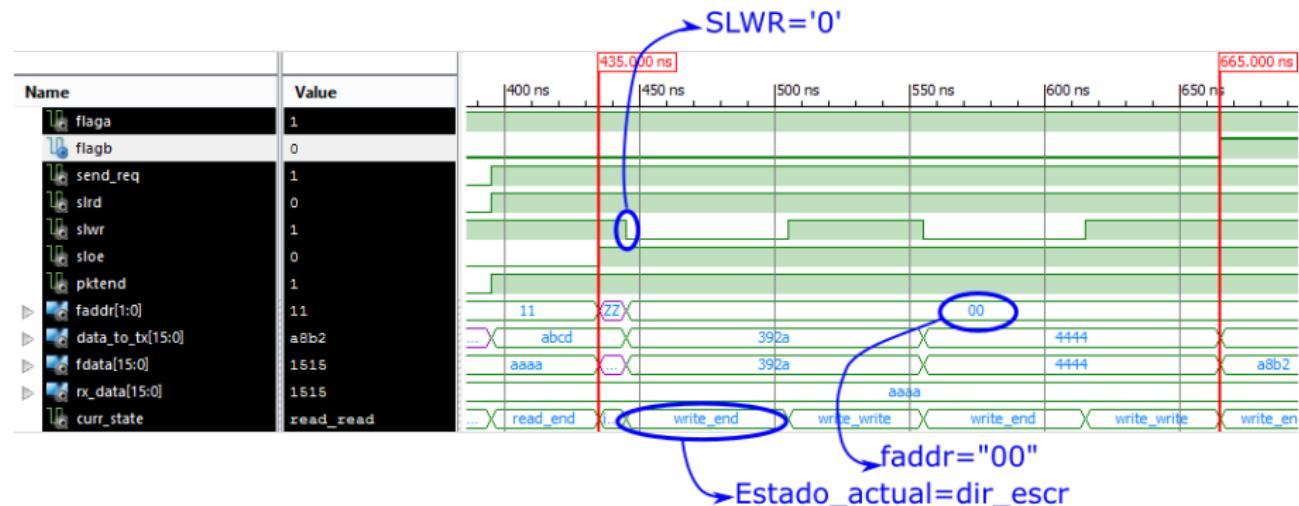
# Verificación Funcional

## Operación de escritura



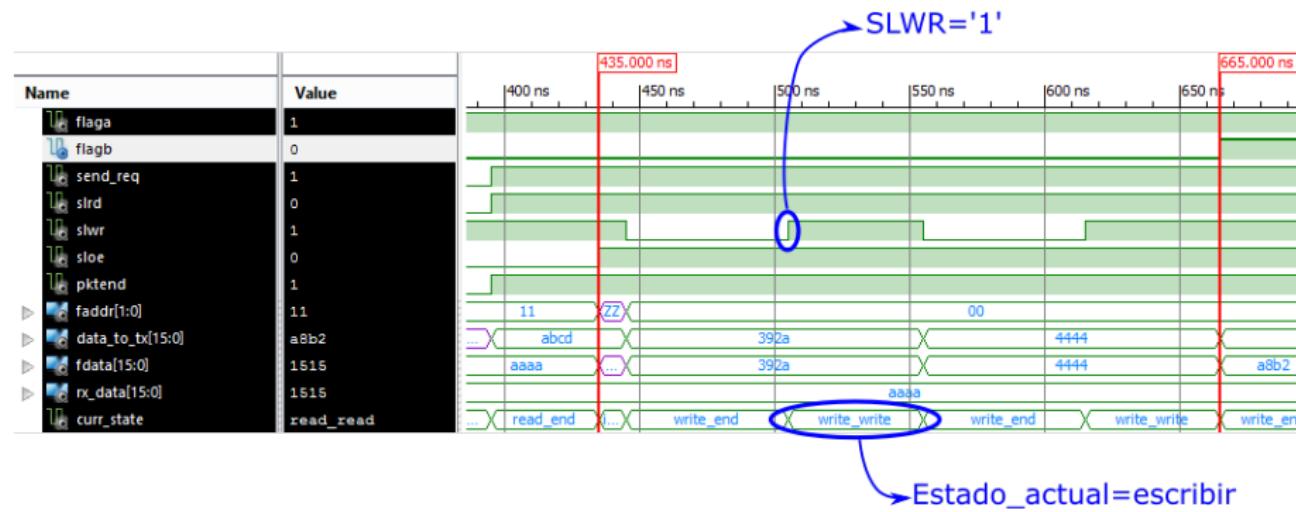
# Verificación Funcional

## Operación de escritura



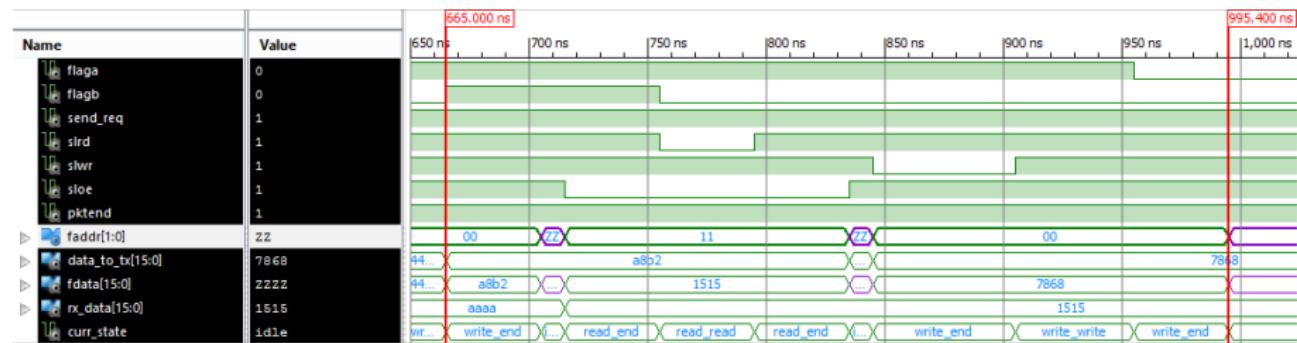
# Verificación Funcional

## Operación de escritura



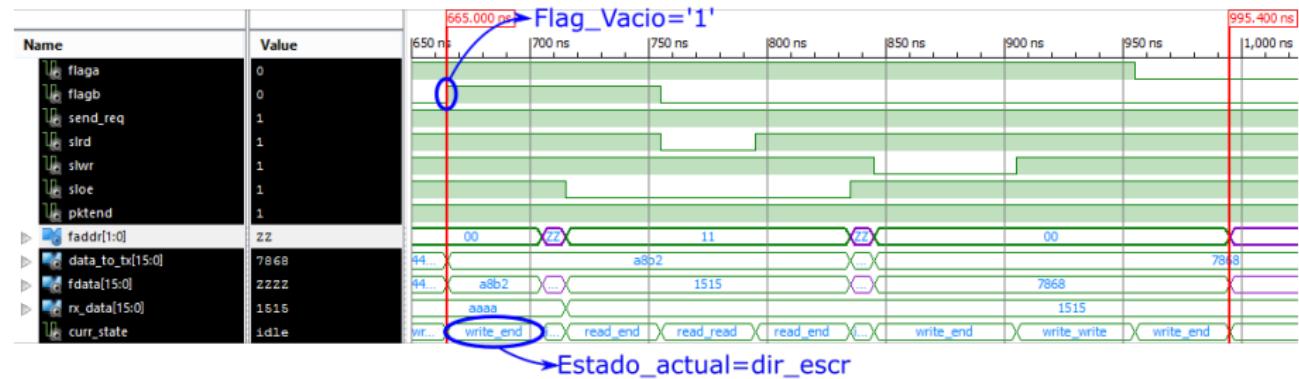
# Verificación Funcional

## Operación de escritura



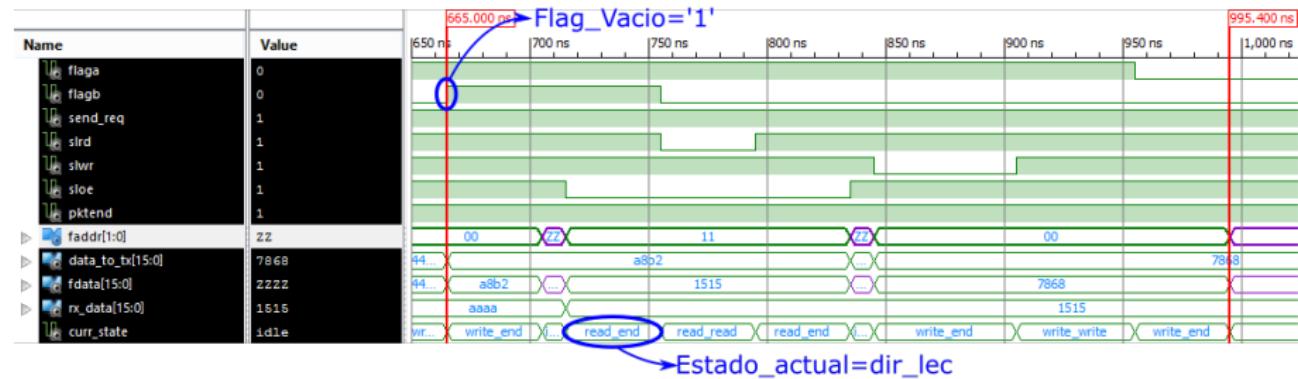
# Verificación Funcional

## Operación de escritura



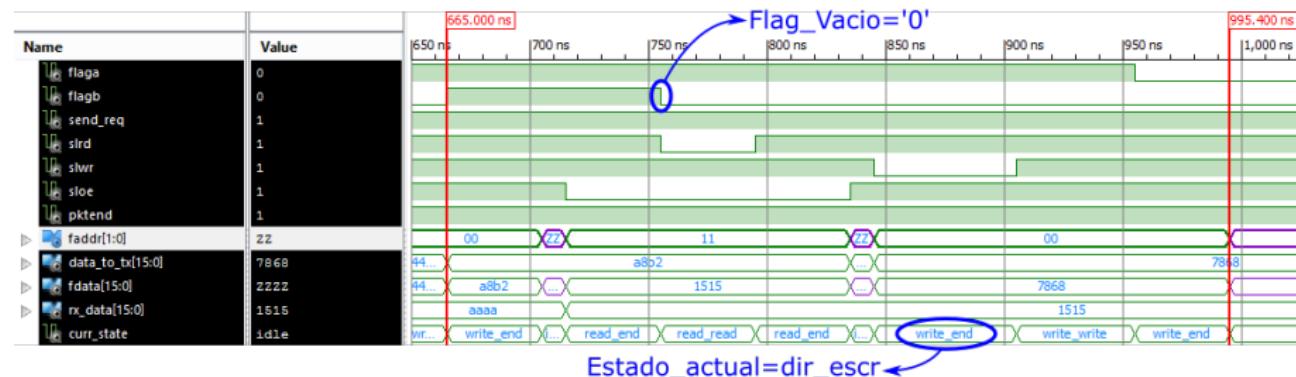
# Verificación Funcional

## Operación de escritura



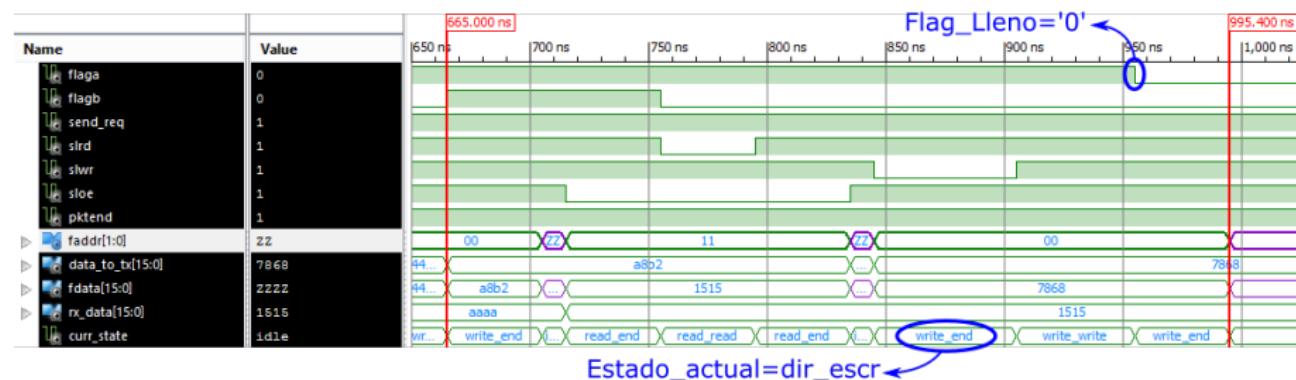
# Verificación Funcional

## Operación de escritura



# Verificación Funcional

## Operación de escritura



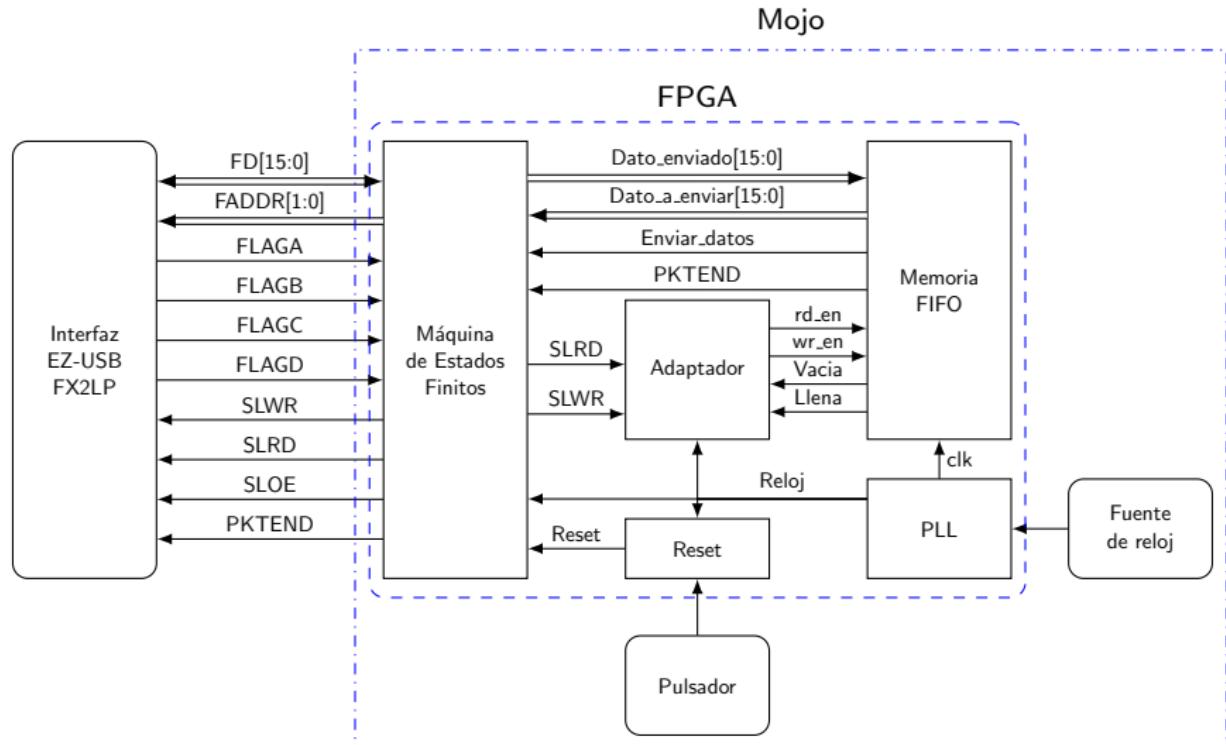
# Agenda

## 3 Evaluación y validación

- Desarrollo del sistema de pruebas
- Desarrollo de un programa de pruebas

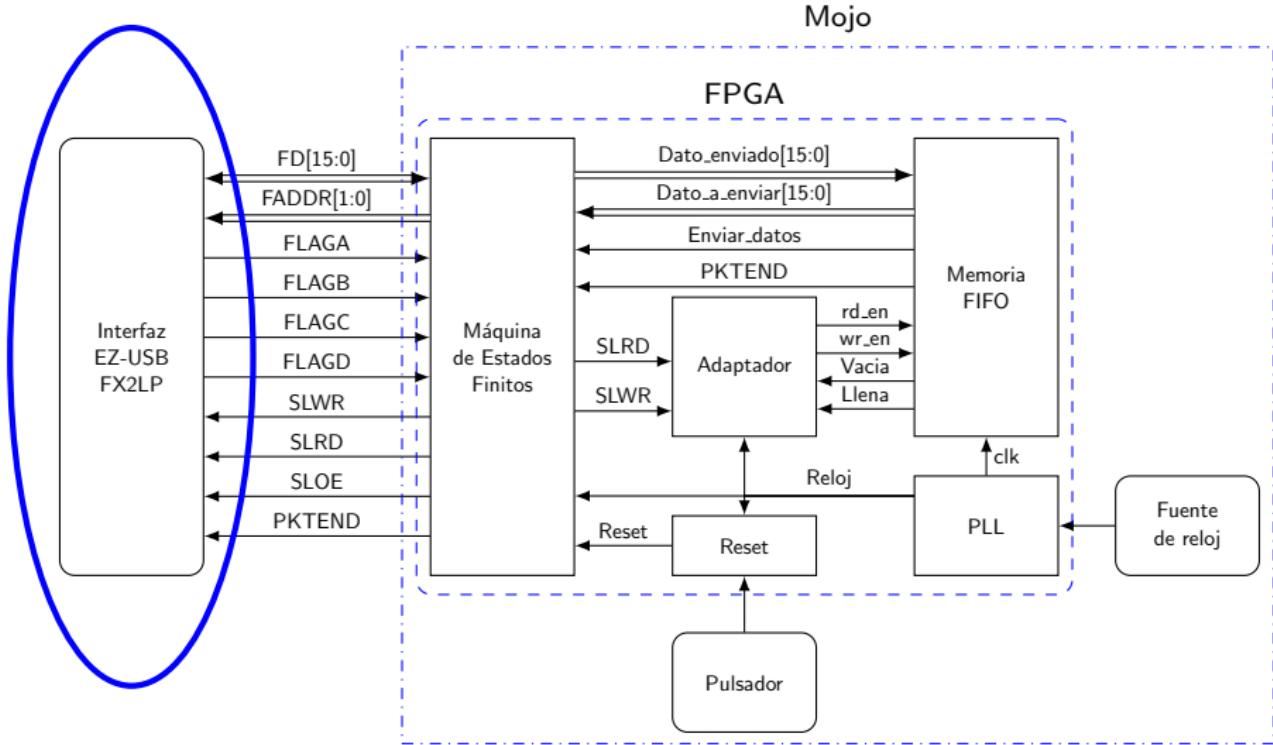
# Sistema de pruebas

## Arquitectura



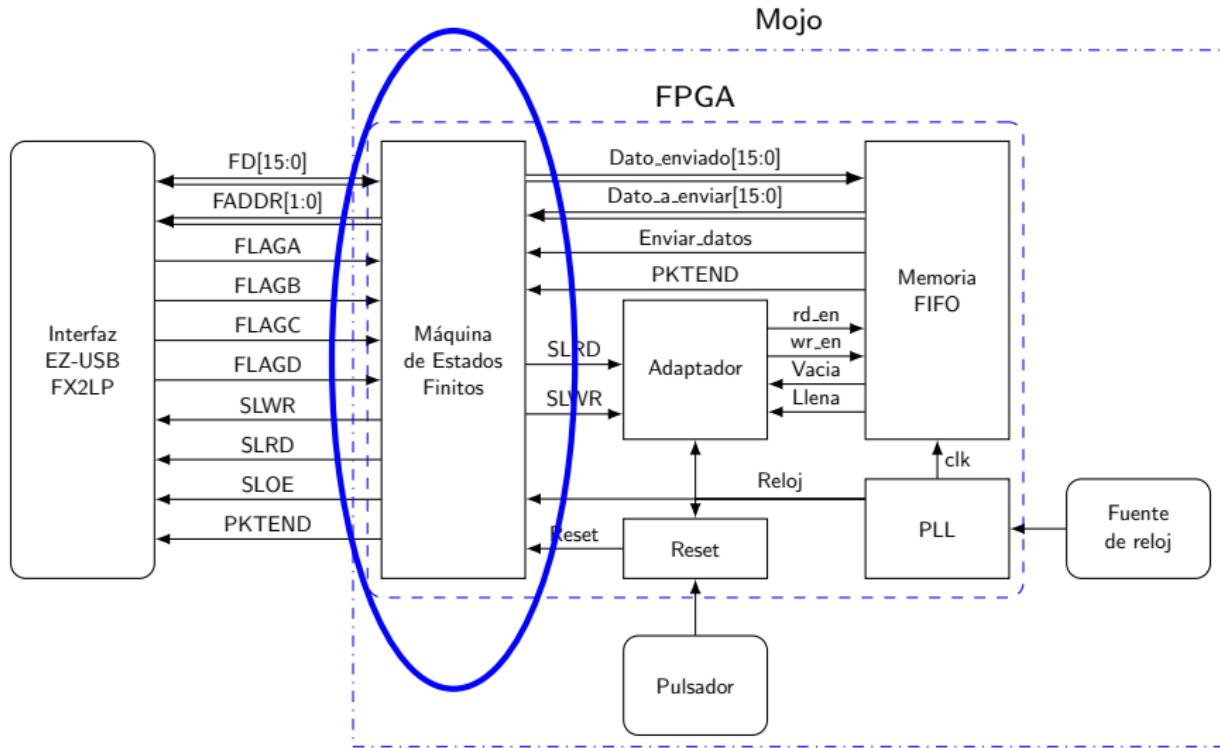
# Sistema de pruebas

## Arquitectura



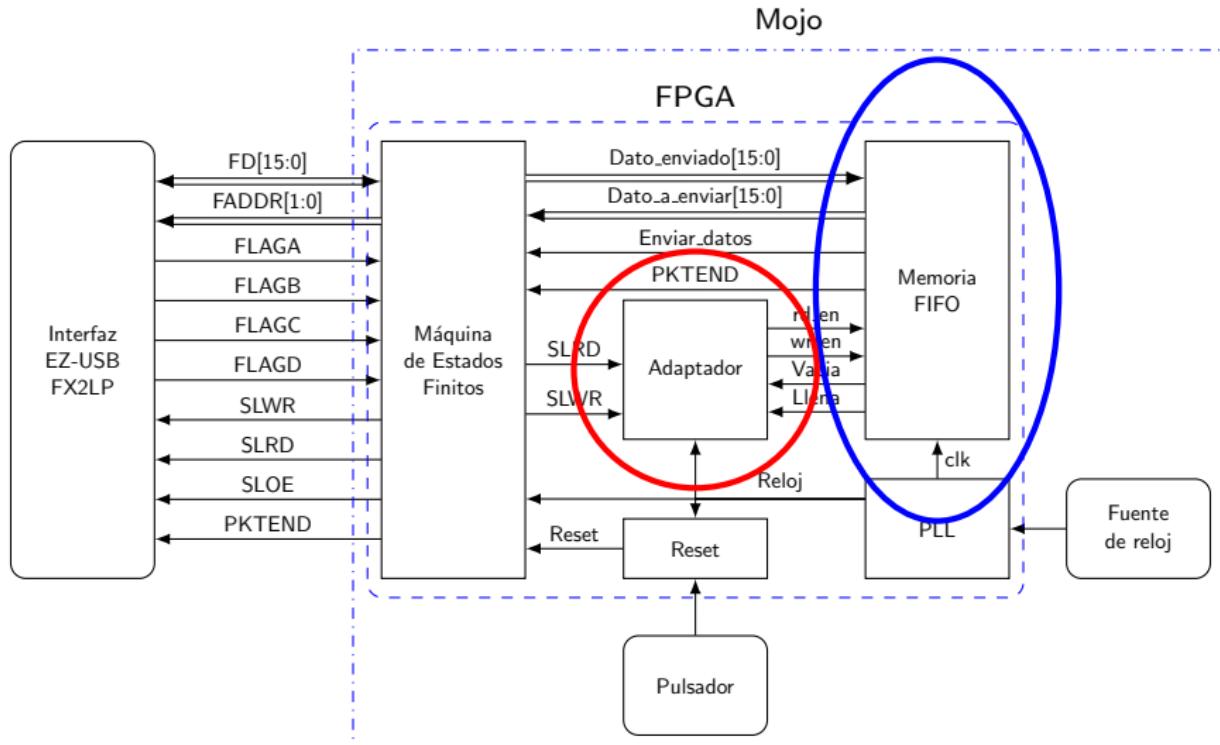
# Sistema de pruebas

## Arquitectura



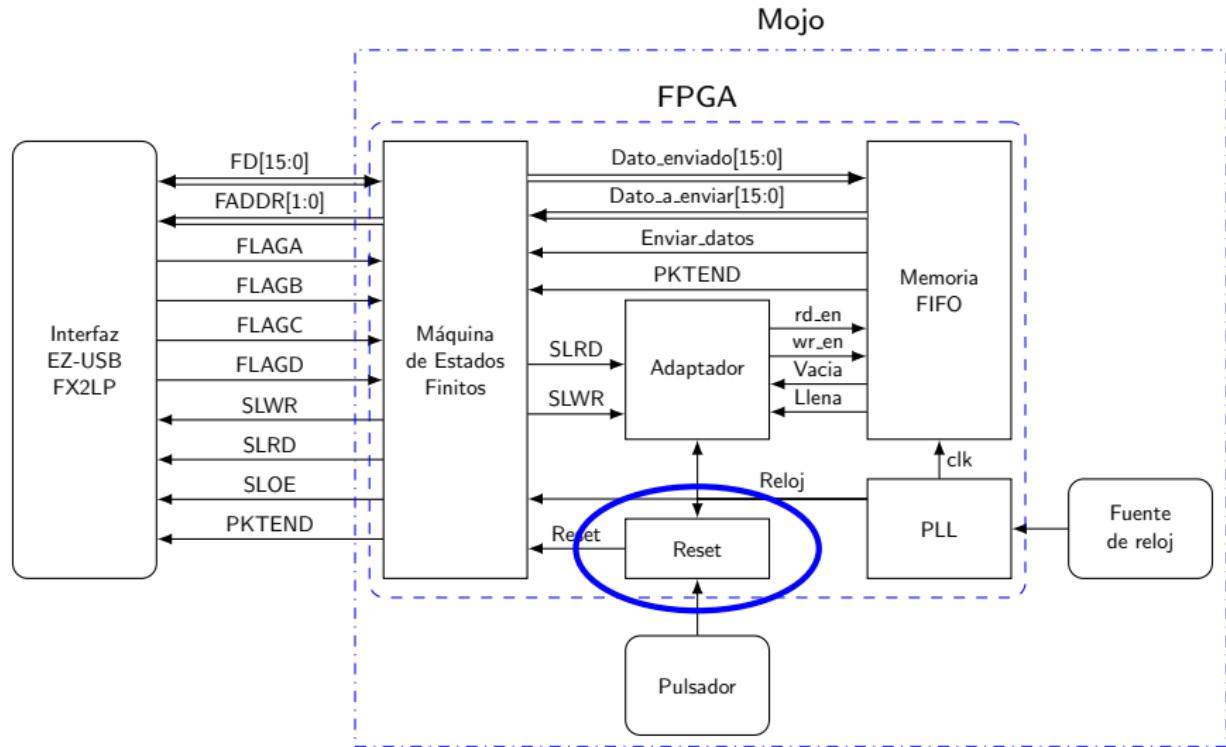
# Sistema de pruebas

## Arquitectura



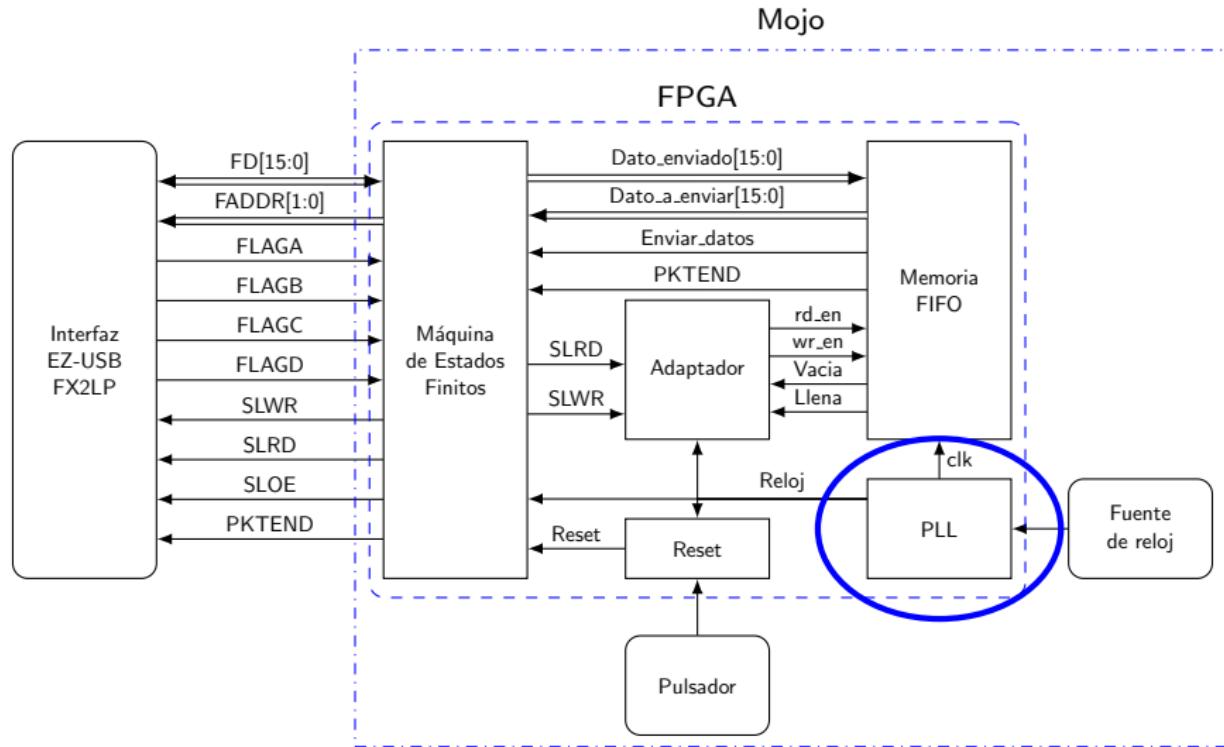
# Sistema de pruebas

## Arquitectura



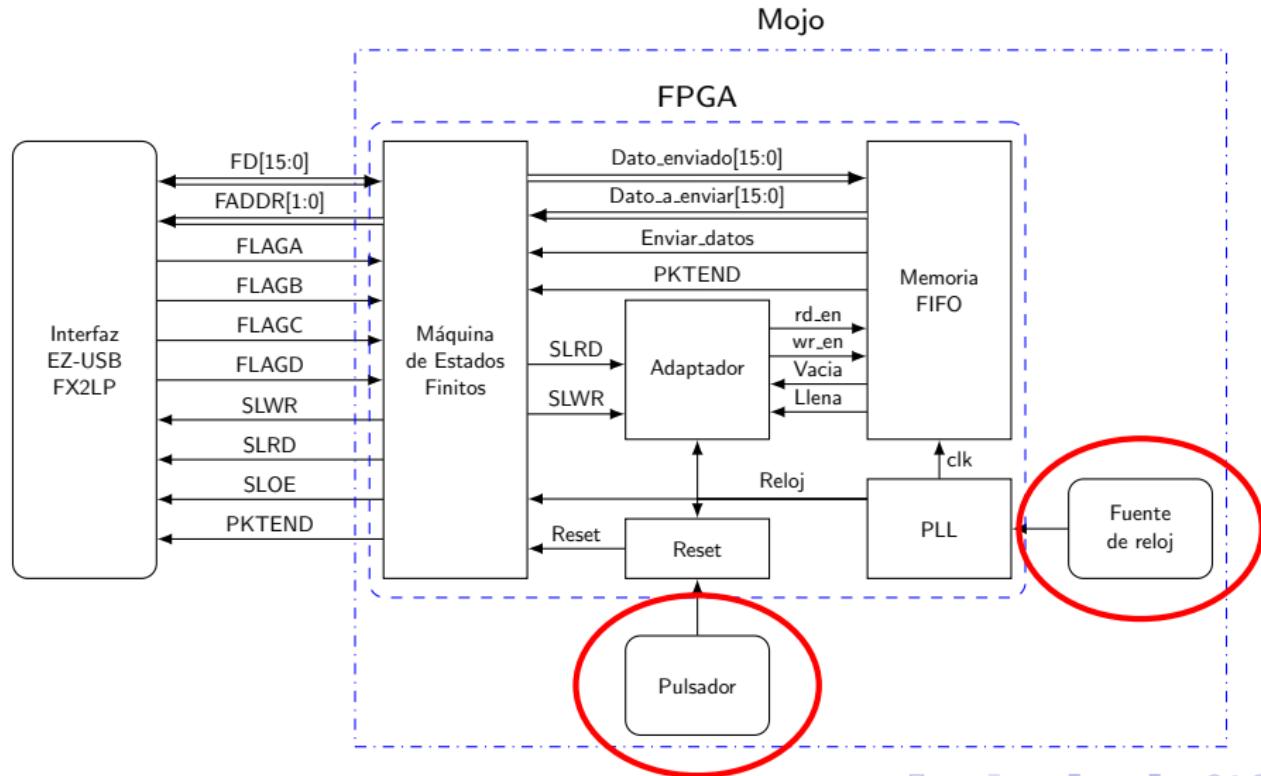
# Sistema de pruebas

## Arquitectura



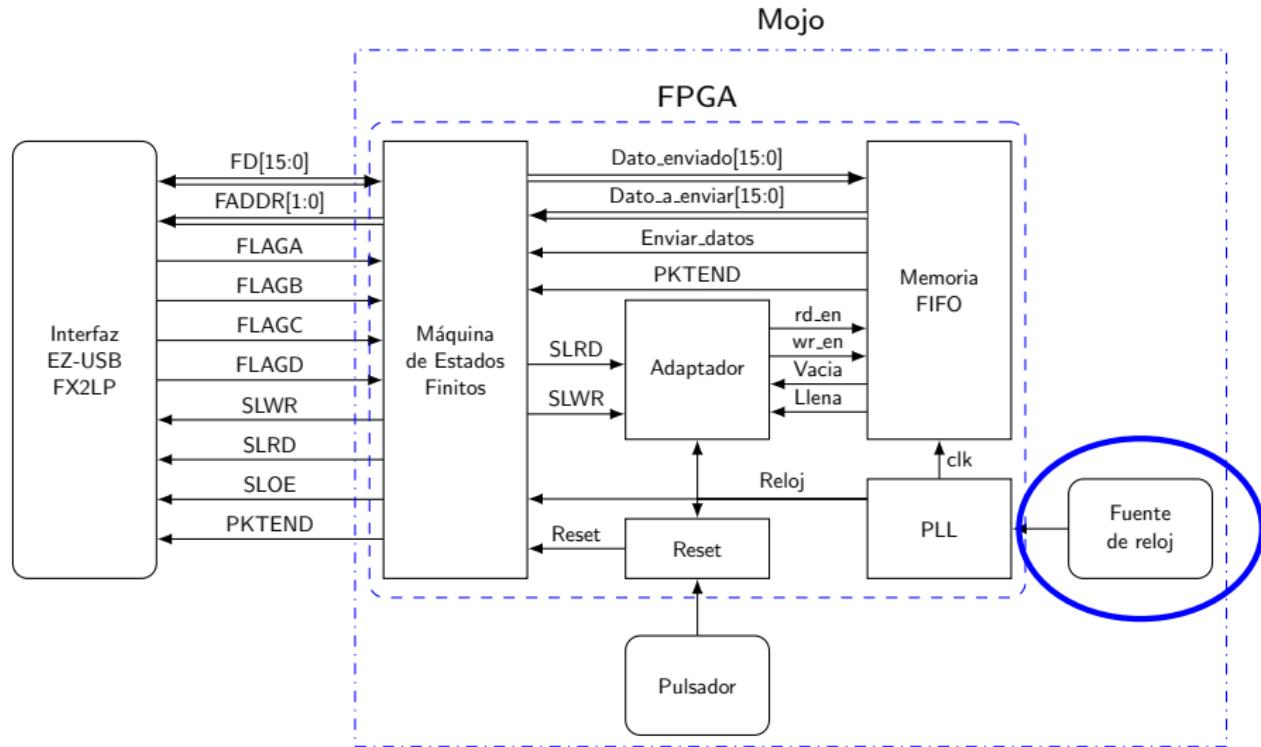
# Sistema de pruebas

## Arquitectura



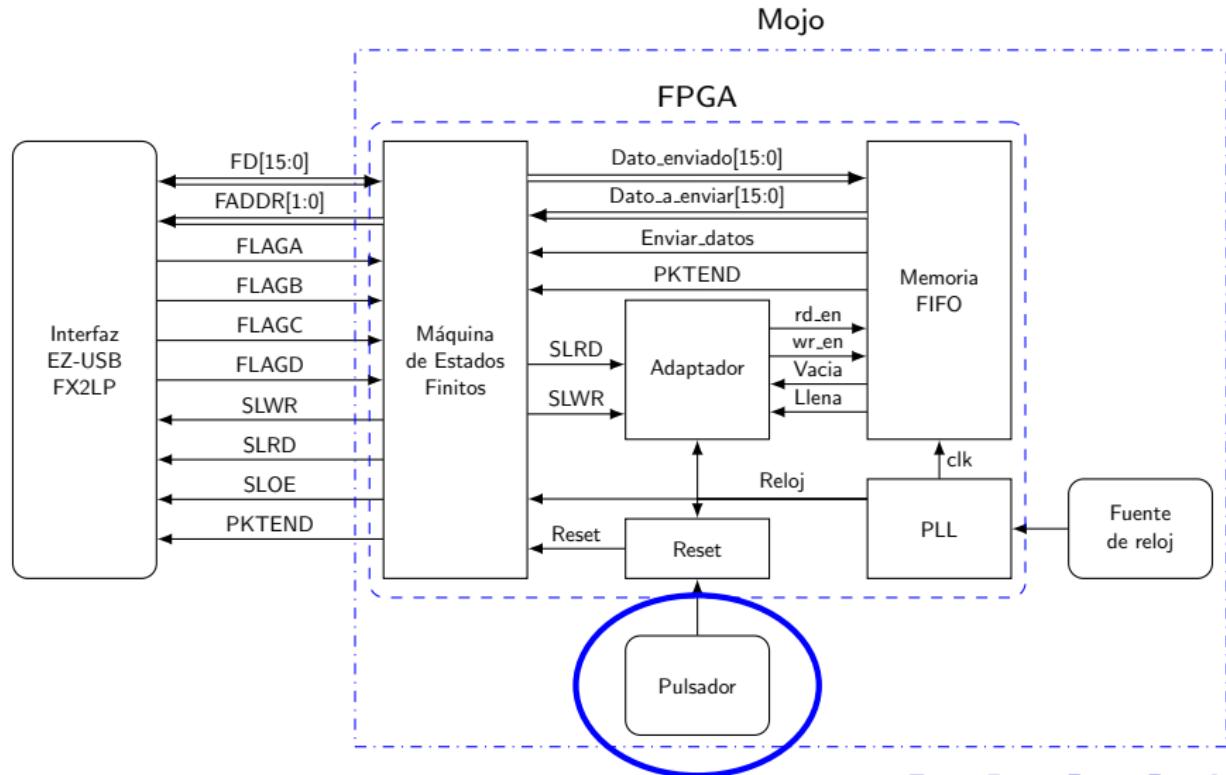
# Sistema de pruebas

## Arquitectura



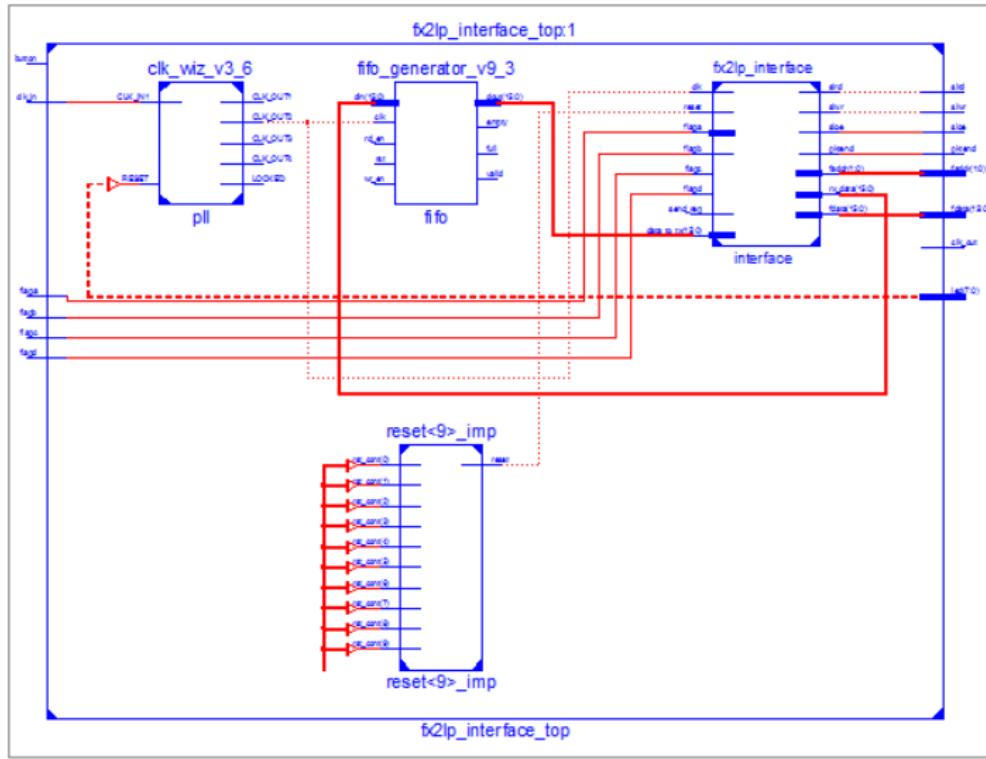
# Sistema de pruebas

## Arquitectura



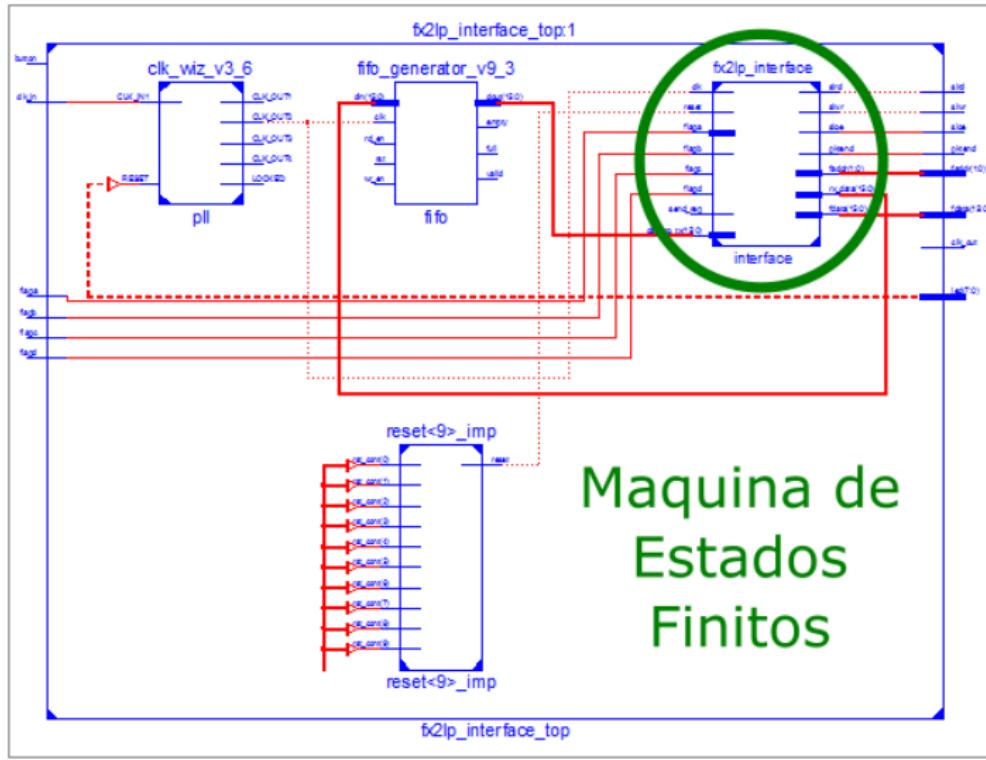
# Sistema de Pruebas

## Implementación en VHDL



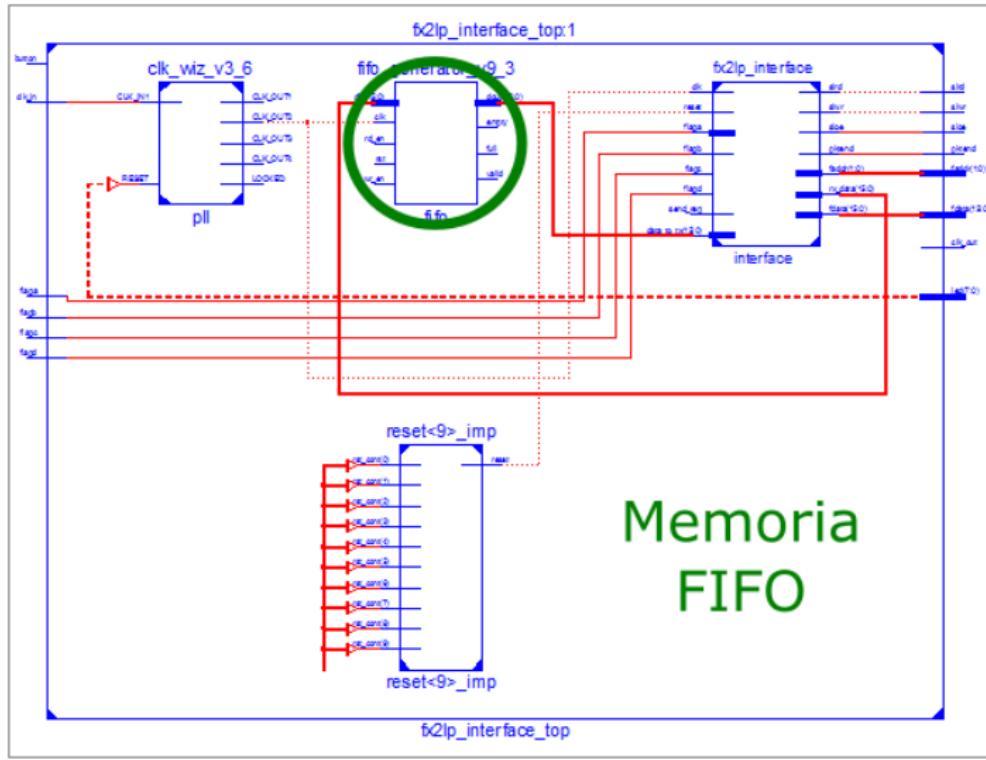
# Sistema de Pruebas

## Implementación en VHDL



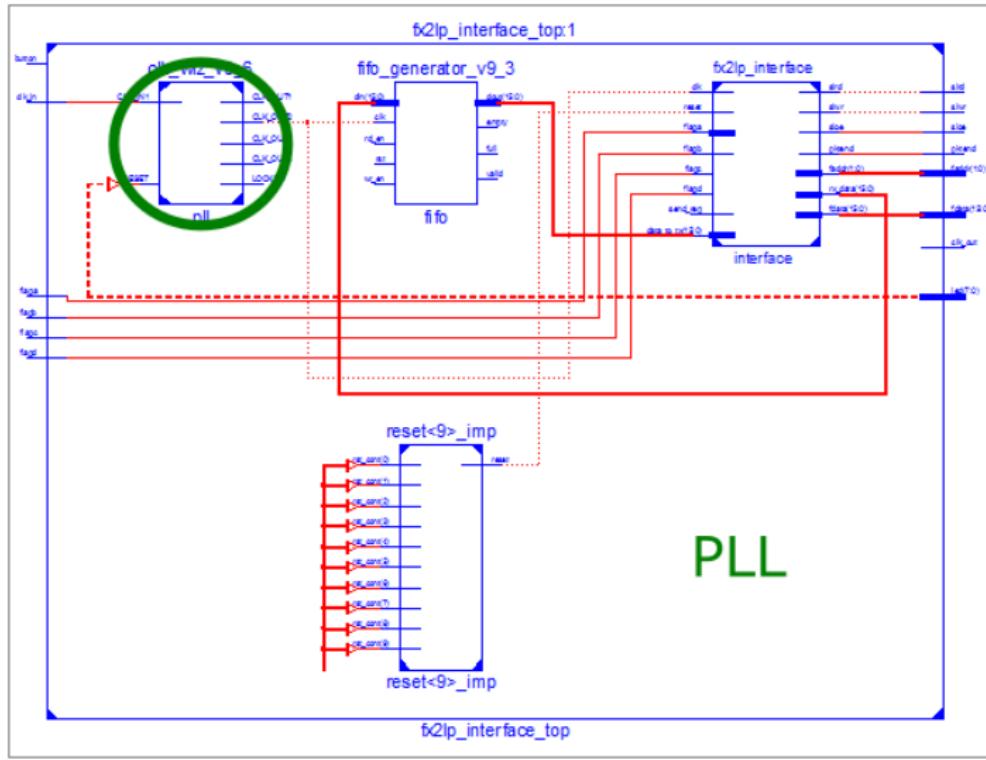
# Sistema de Pruebas

## Implementación en VHDL



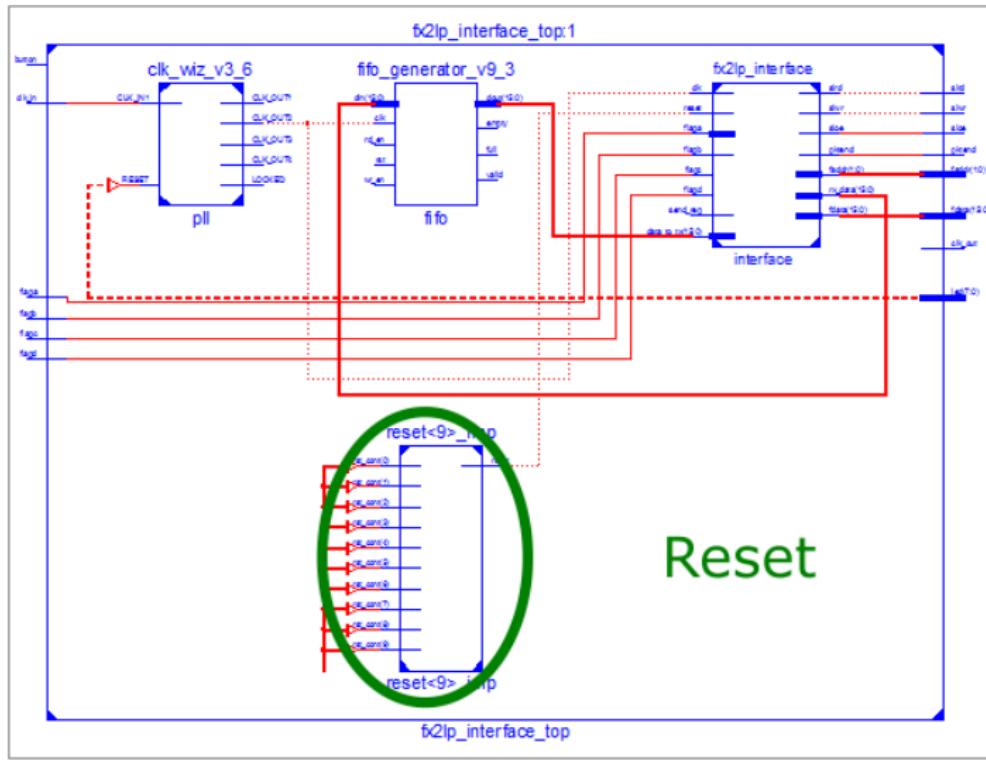
# Sistema de Pruebas

## Implementación en VHDL



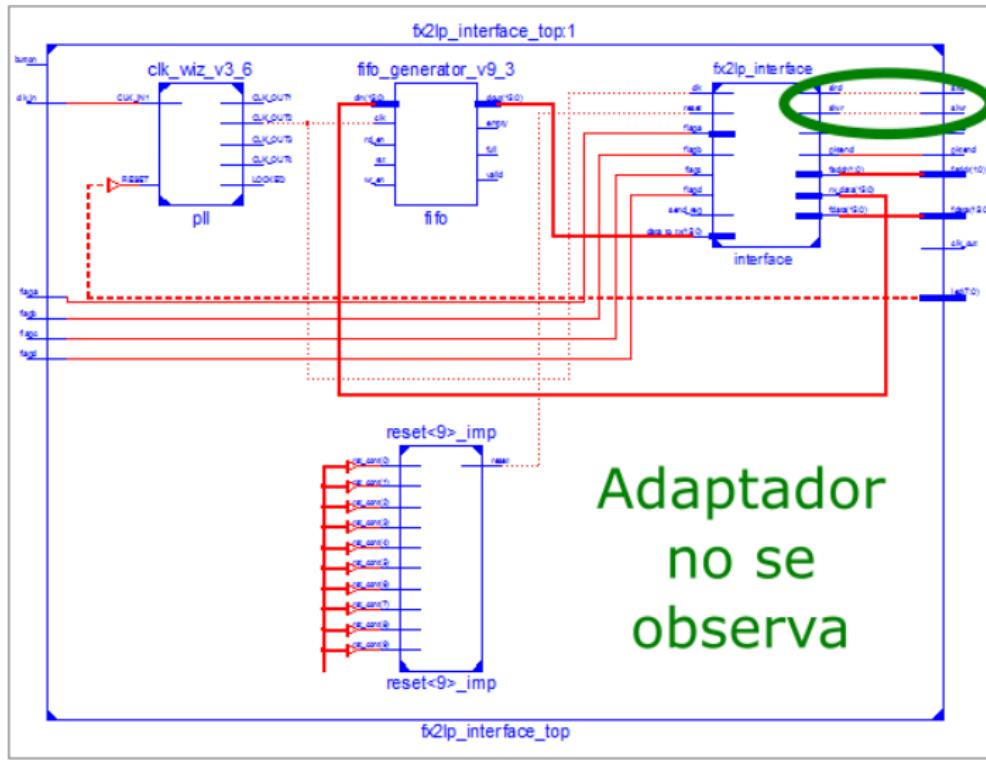
# Sistema de Pruebas

## Implementación en VHDL



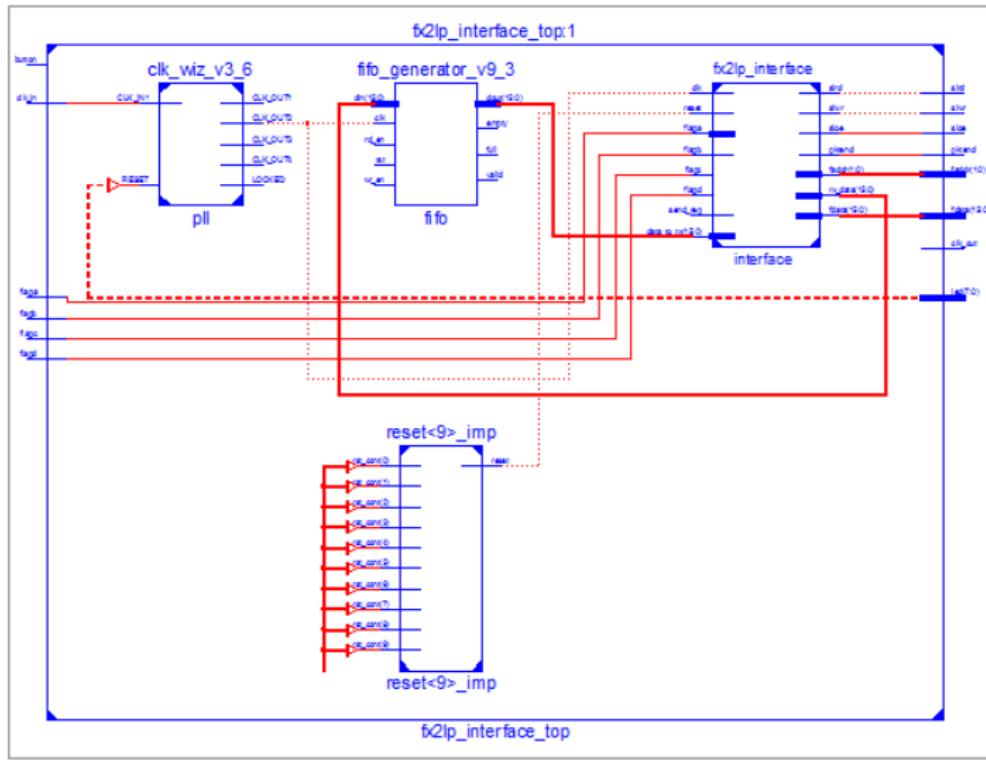
# Sistema de Pruebas

## Implementación en VHDL



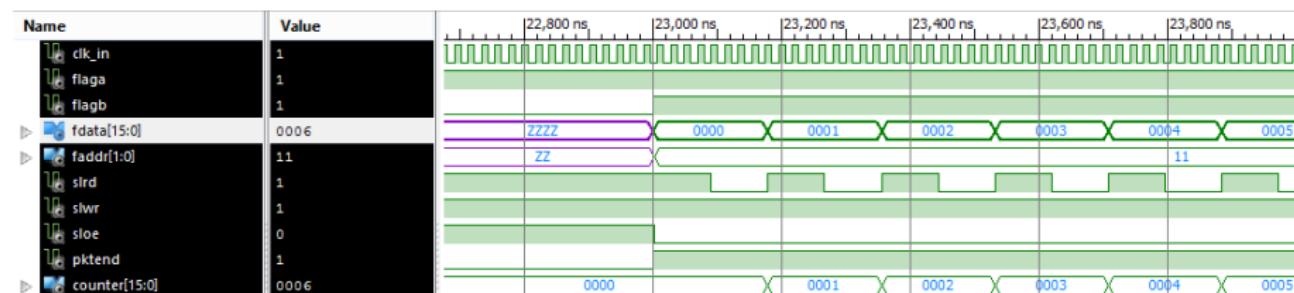
# Sistema de Pruebas

## Implementación en VHDL



# Verificación Funcional

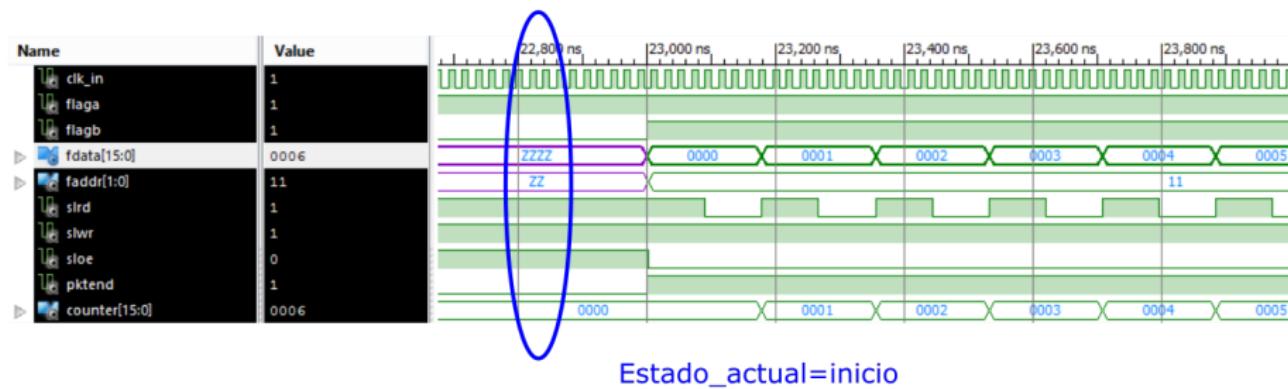
## Recepción de datos



# Verificación Funcional

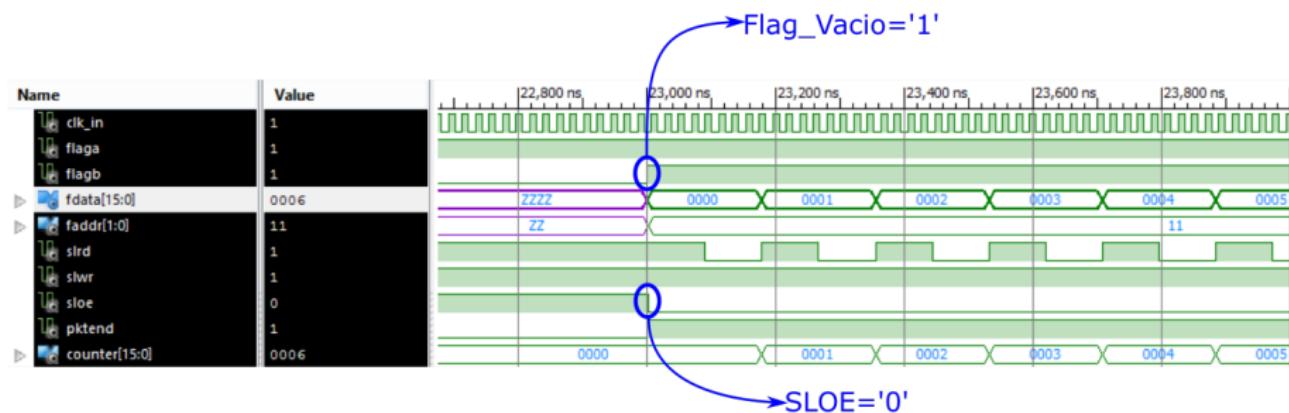
## Recepción de datos

Sistema sin estímulos



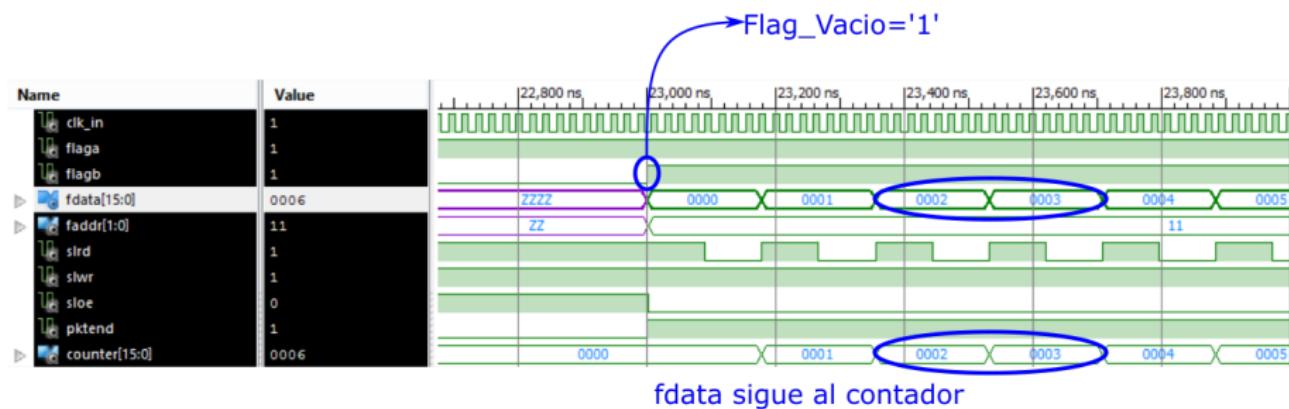
# Verificación Funcional

## Recepción de datos



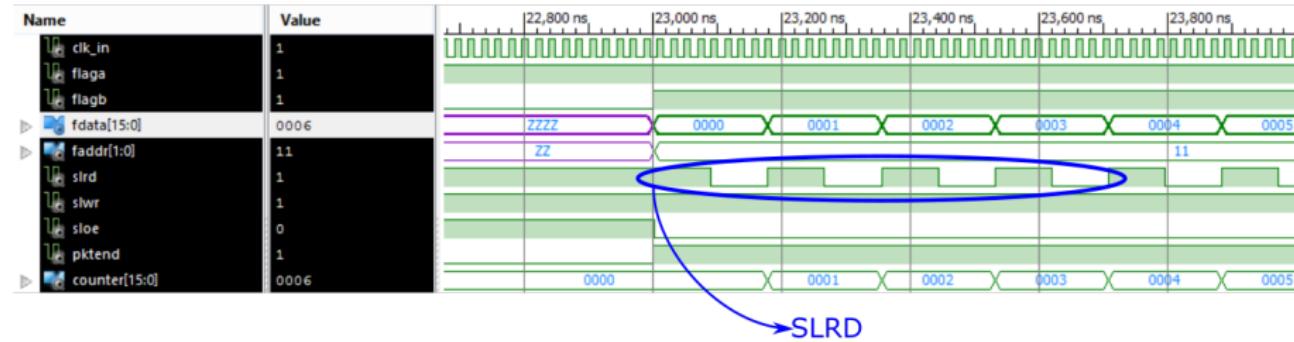
# Verificación Funcional

## Recepción de datos



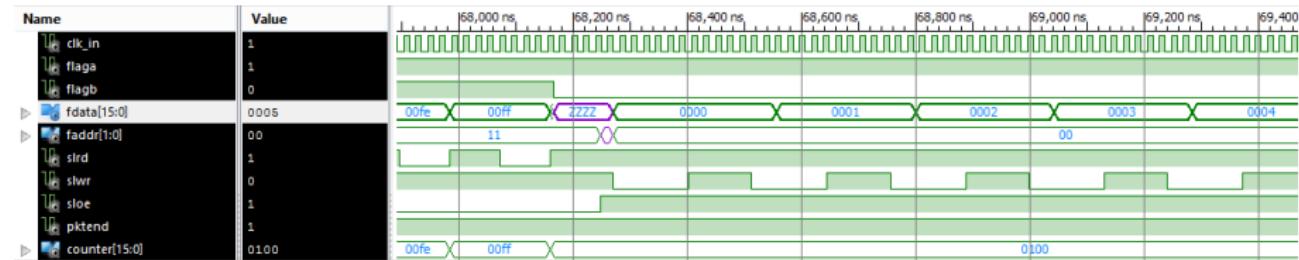
# Verificación Funcional

## Recepción de datos



# Verificación Funcional

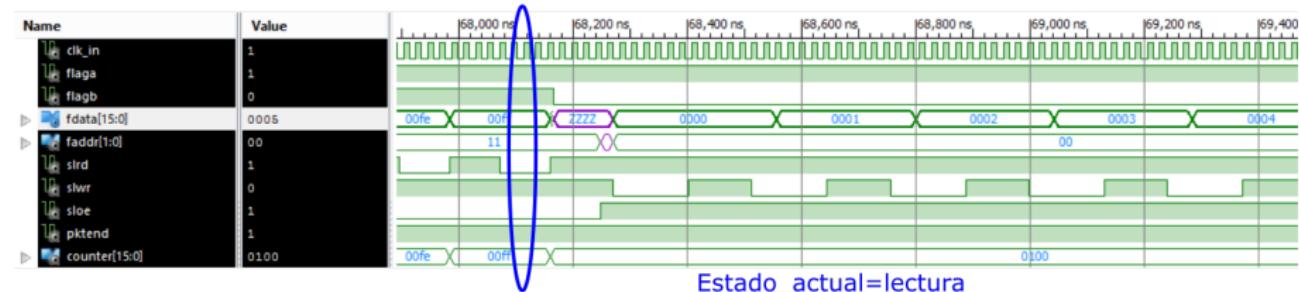
## Transmisión de datos



# Verificación Funcional

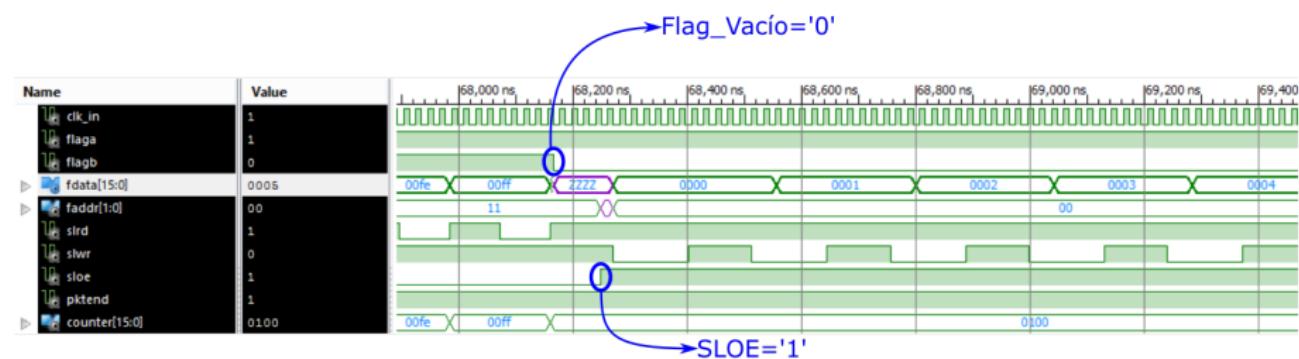
## Transmisión de datos

Sistema leyendo la Interfaz USB



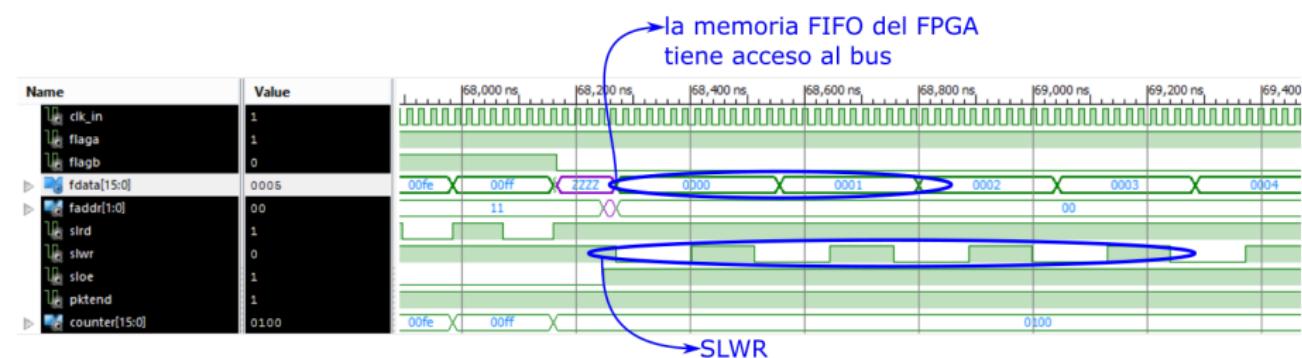
# Verificación Funcional

## Transmisión de datos



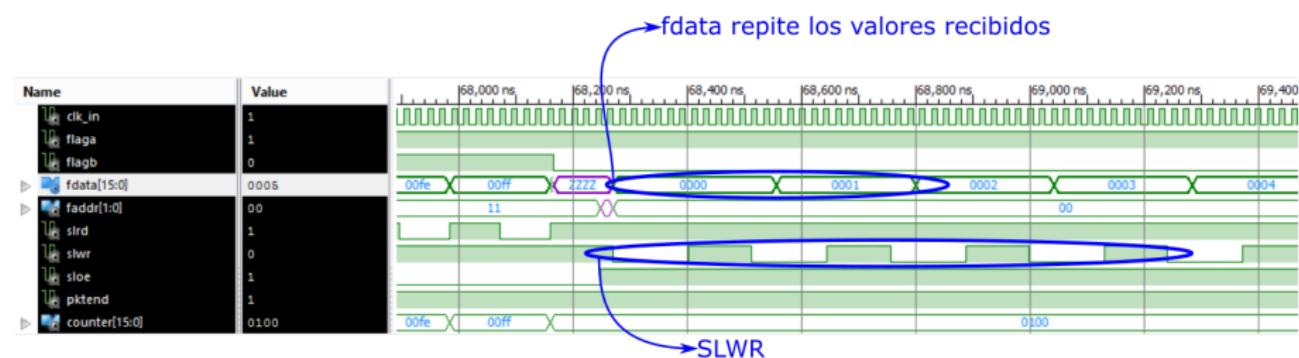
# Verificación Funcional

## Transmisión de datos



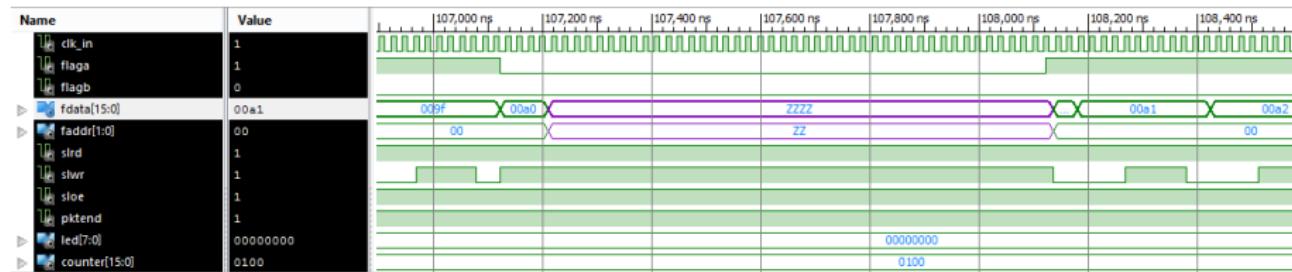
# Verificación Funcional

## Transmisión de datos



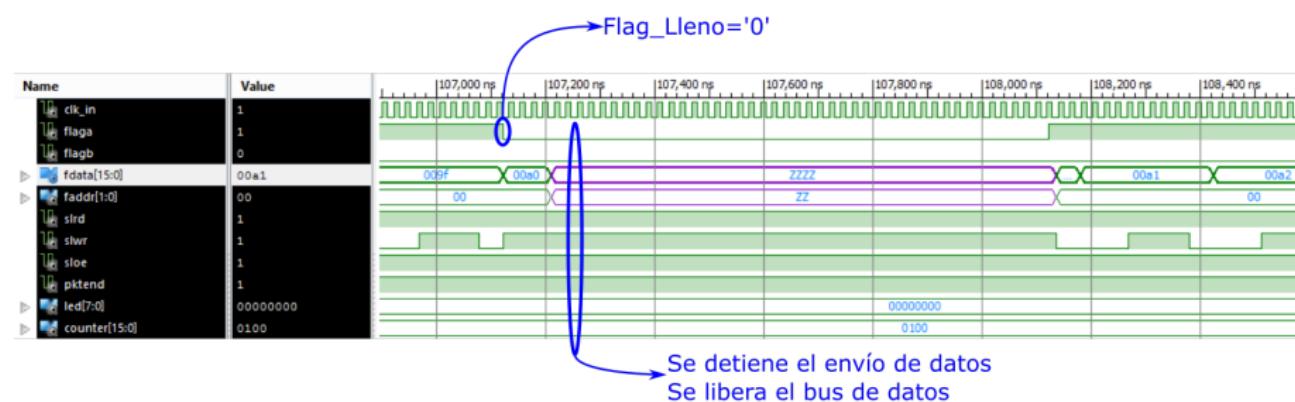
# Verificación Funcional

## Transmisión de datos



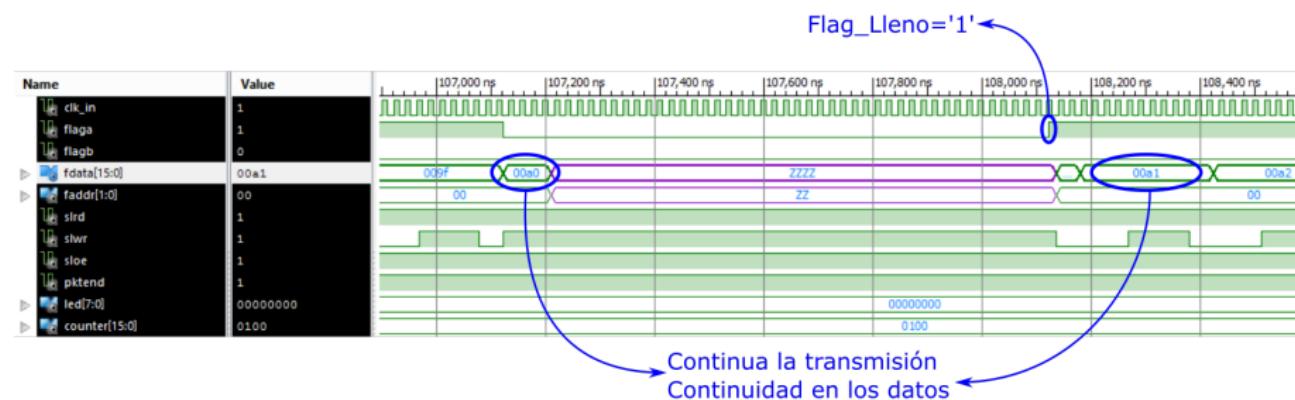
# Verificación Funcional

## Transmisión de datos



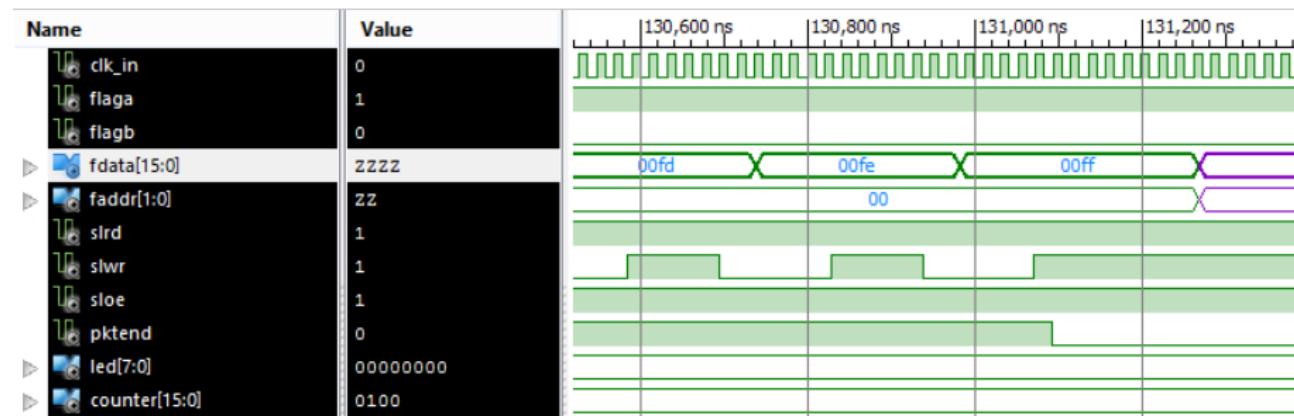
# Verificación Funcional

## Transmisión de datos



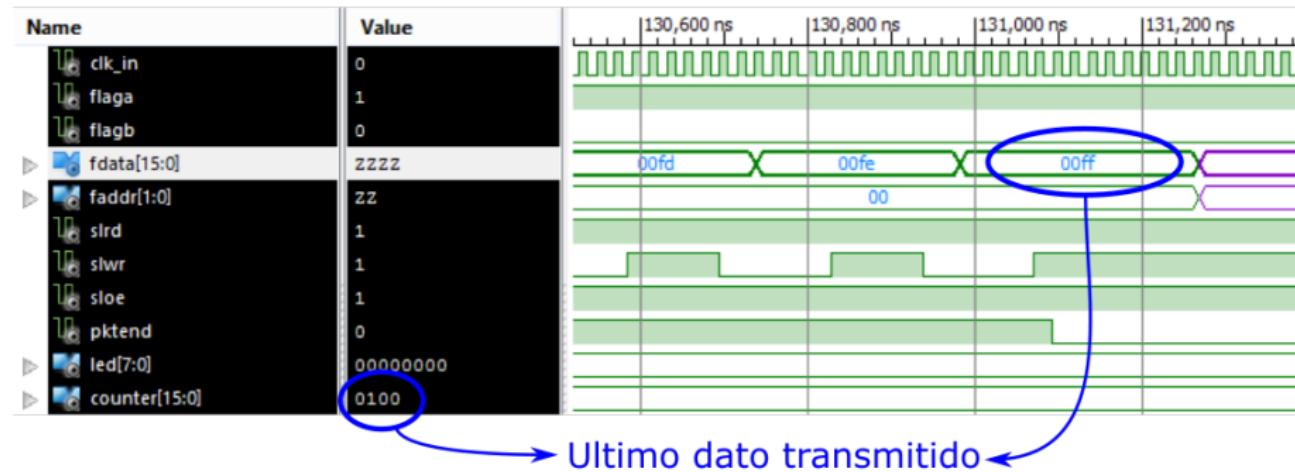
# Verificación Funcional

## Transmisión de datos



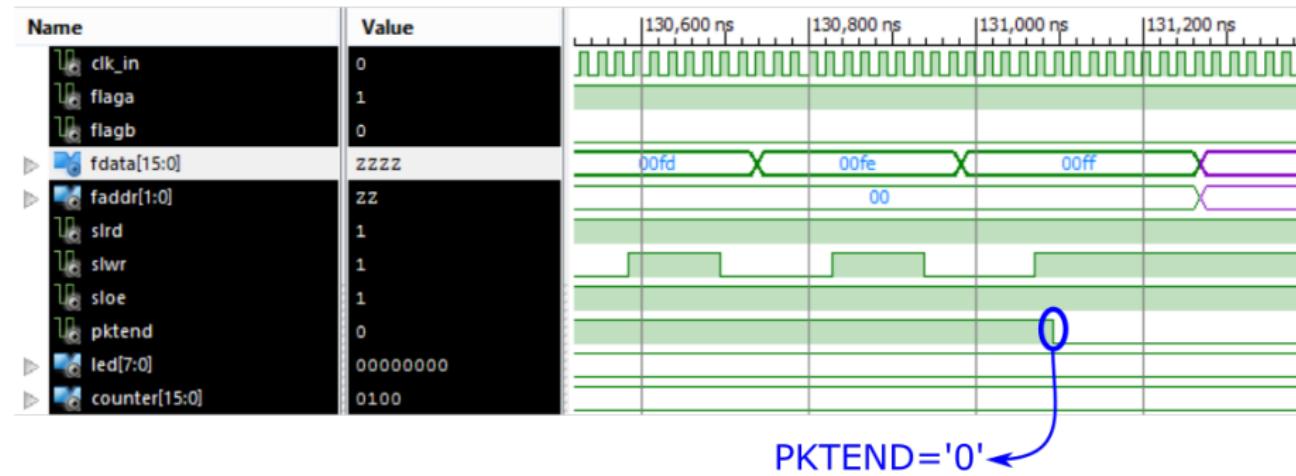
# Verificación Funcional

## Transmisión de datos



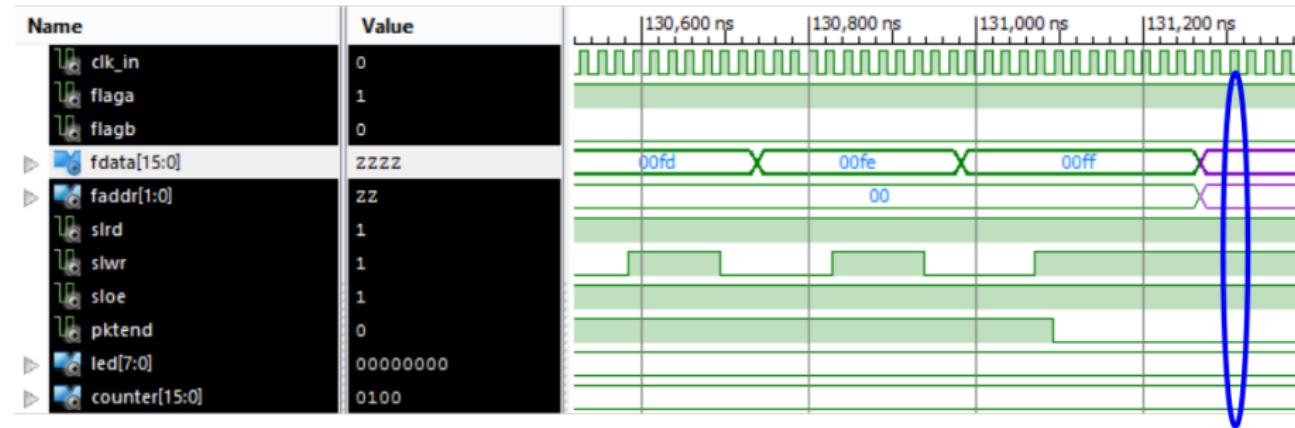
# Verificación Funcional

## Transmisión de datos



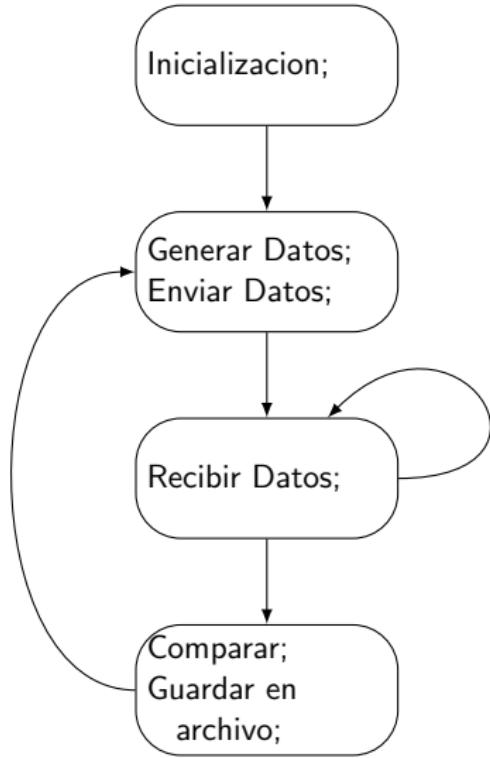
# Verificación Funcional

## Transmisión de datos



Estado\_actual=inicio

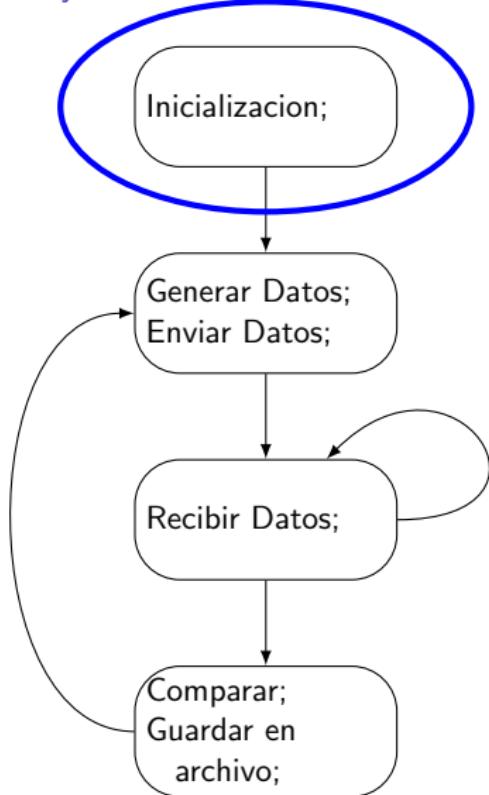
# Programa de PC para la transmisión de datos hacia y desde el FPGA



Se realizó un programa de computadoras, escrito en lenguaje C, para el cual se utilizó la biblioteca libusb-1.0, que permite obtener acceso al puerto USB como así también transmitir y recibir datos a través de él.

# Programa de PC para la transmisión de datos

hacia y desde el FPGA

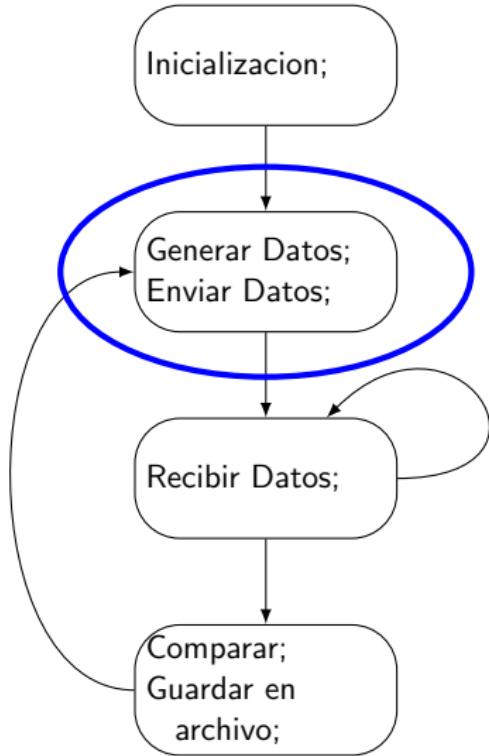


## Inicialización:

- Se listan los dispositivos
- Se busca y selecciona el adecuado
- Se configura la interfaz

# Programa de PC para la transmisión de datos

## hacia y desde el FPGA

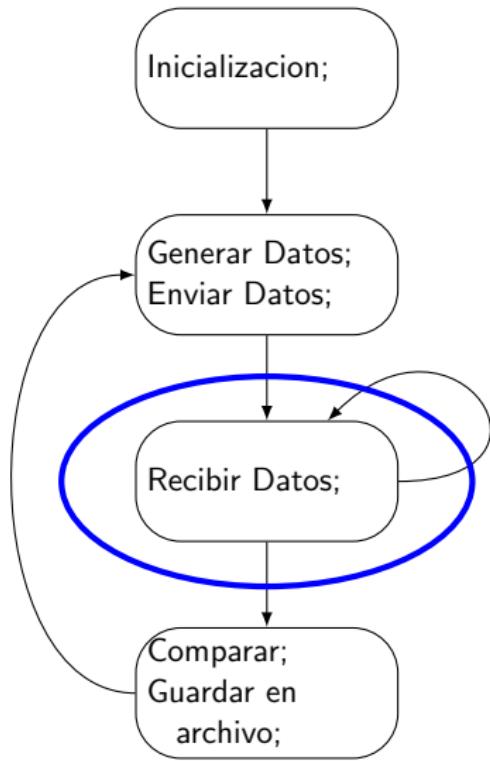


### Generar Datos:

	Paridad columna		
Fila 1	0	0	1
Fila 2	1	1	0
Fila 3	0	1	0
Paridad fila	1	0	1

- Se generan datos aleatorios de 7 bits
- Se agrega un bit de paridad.
- Cada 7 bytes generados, se agrega 1 Byte con las paridades verticales.
- Se generan 16 tramas de 8 bytes, para alcanzar un paquete de 128 Bytes de datos.

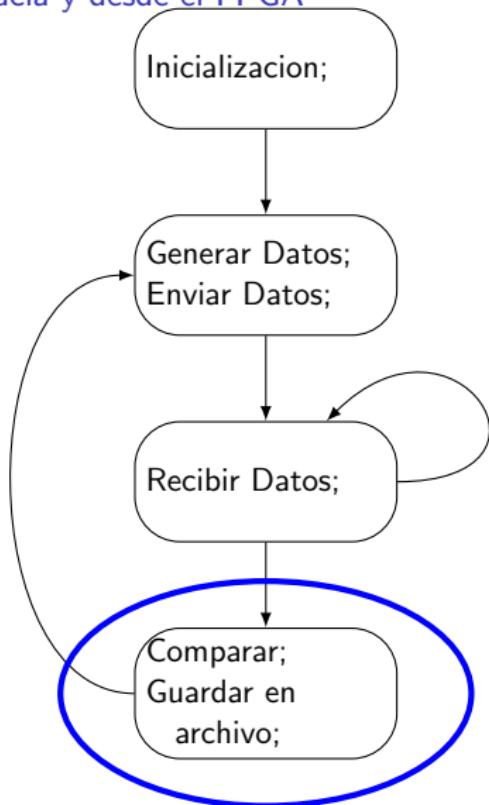
# Programa de PC para la transmisión de datos hacia y desde el FPGA



Recibir datos:  
Se solicita la recepción de datos y se espera  
hasta que estos lleguen

# Programa de PC para la transmisión de datos

hacia y desde el FPGA

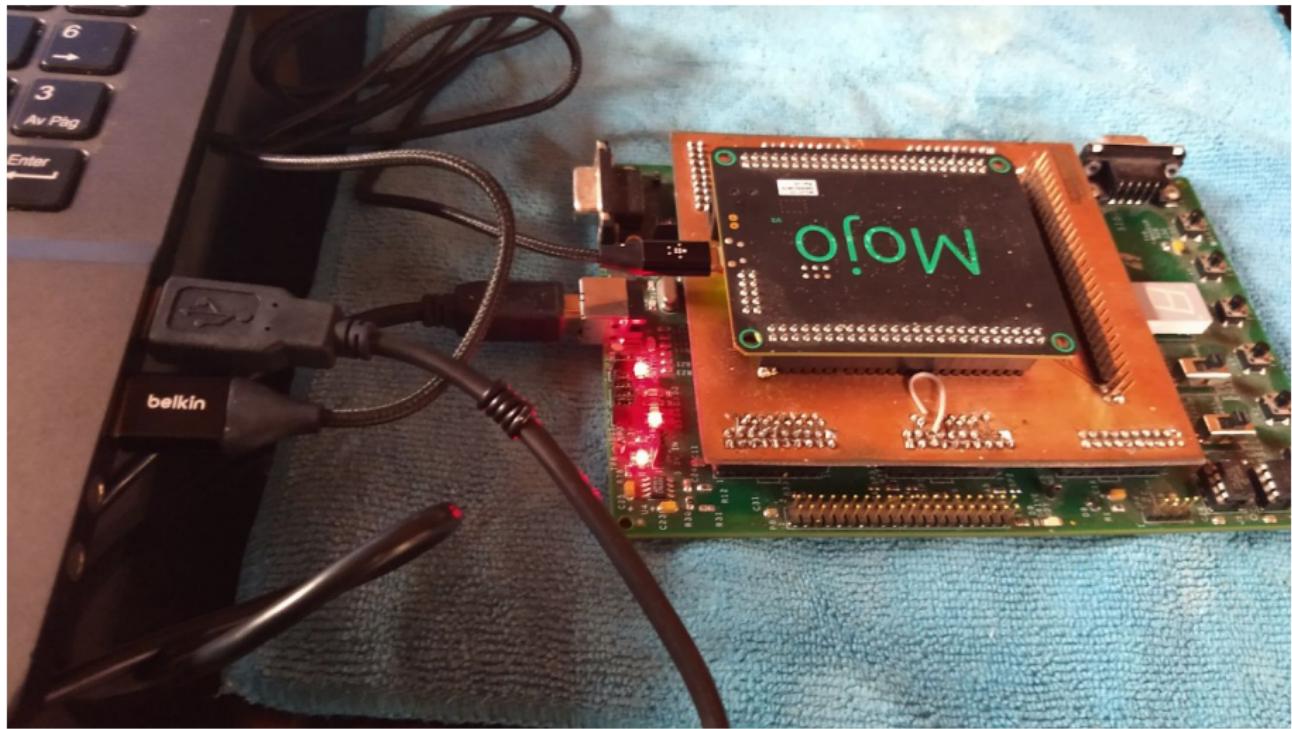


Comparar:

- Se chequea que los datos recibidos no tuvieron errores.
- Se comparan los datos recibidos con los enviados.

Guardar en archivo;

# Sistema en funcionamiento



# Agenda

## 4 Resultados y conclusiones

- Pruebas y resultados
- Conclusiones
- Trabajo futuro

# Resultados de la prueba de comunicación

- El sistema envió y recibió paquetes durante 24 horas, logrando la correcta transmisión y recepción de 388.191.289 paquetes de 128 bytes cada uno.
- El sistema desarrollado estableció una comunicación USB de alta velocidad.
- La tasa de transferencia de datos es de 12,4 Mbps.
- No hubo perdida ni errores en los datos transmitidos.
- La tasa efectiva de transmisión de datos útiles fue de 9,12 Mbps.

# Conclusiones

- Se desarrollo un sistema de comunicación USB 2.0 que permite conectar una FPGA con una PC. El enlace pudo ser operado a 480 Mbps.
- El sistema elaborado permite leer y escribir datos en forma robusta, es decir, sin perder datos.
- Se adquirieron y entendieron conceptos importantes de la norma USB 2.0.
- Se seleccionó y se configuró el controlador FX2LP como interfaz USB.
- Se seleccionó el FPGA Spartan 6 de Xilinx Inc, incorporado en la placa de desarrollo Mojo v3.

# Conclusiones

- Se desarrolló una máquina de estados finitos implementada con el FPGA que comanda la comunicación entre este dispositivo y la interfaz USB.
- Se desarrolló un programa de computadoras que permite enviar y recibir datos a través del puerto USB.
- Se logró transmitir información desde y hacia la PC a una tasa efectiva de 9,12 Mbps.

# Trabajos futuros

- Realizar una prueba de máxima transferencia de datos a través del envío ininterrumpido de datos desde la FPGA hacia la PC.
- Implementar una comunicación síncrona entre ambos sistemas.
- Estudiar la migración del sistema desarrollado hacia USB 3.1

Muchas gracias.

# Consultas

Consultas.