

## Módulo SPI Master

Microarquitecturas y Softcores

Carrera de Especialización en Sistemas Embebidos

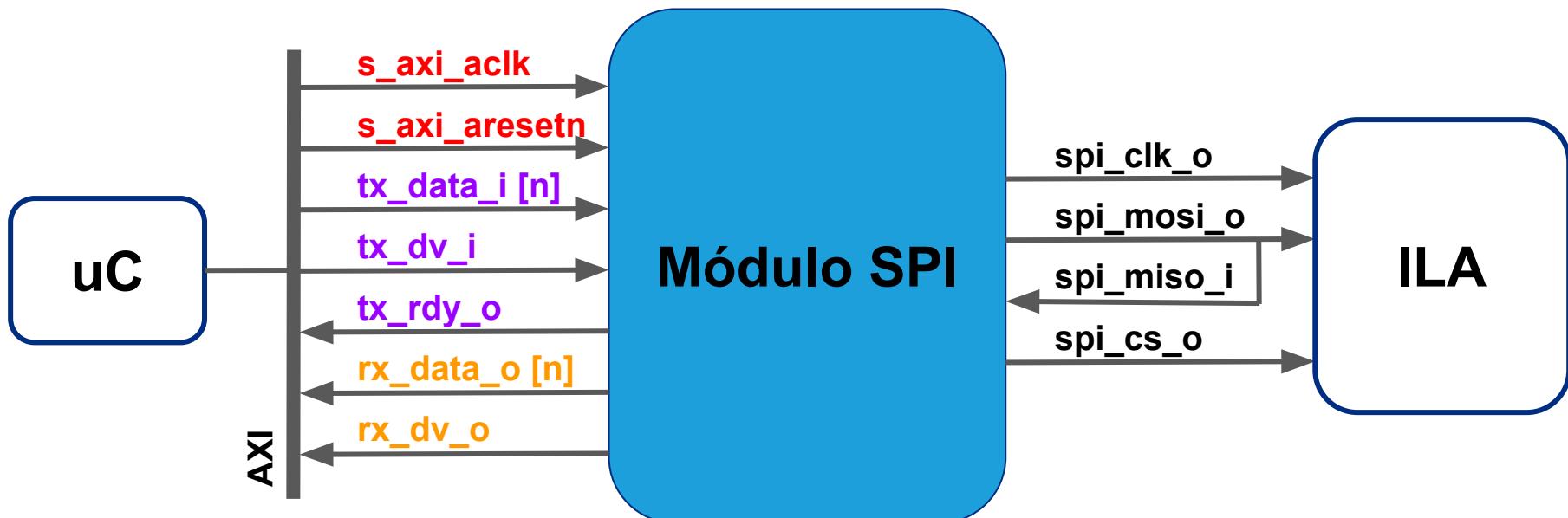
Alumno: Ing. Lucas Sebastián Kirschner

# Descripción del módulo

Módulo SPI Master con control automático de Chip Select (CS), soporte para los cuatro modos estándar de operación (CPOL/CPHA), ancho de palabra configurable (8 o 16 bits), selección del orden de bits (MSB/LSB first) y ajuste de frecuencia mediante parámetros de reloj de entrada y tasa de datos deseada



# Conexión lógica del módulo



# Parámetros del módulo

Package IP - spi\_master\_ip

Packaging Steps

- Identification
- Compatibility
- File Groups
- Customization Parameters
- Ports and Interfaces
- Addressing and Memory
- Customization GUI

Review and Package

Customization GUI

Layout

Show disabled ports

Component Name: spi\_master\_ip\_0

C S AXI BASEADDR: 0xFFFFFFFFFF

C S AXI HIGHADDR: 0x00000000

S\_AXI

- + spi\_miso\_i
- spi\_clk\_o
- s\_axi\_aclk
- spi\_mosi\_o
- s\_axi\_arstn
- spi\_cs\_o

Data Size: 8 **8 BITS**

Mode: 0 **CPOL = 0 CPHA = 0**

First Bit: 0 **LSB FIRST**

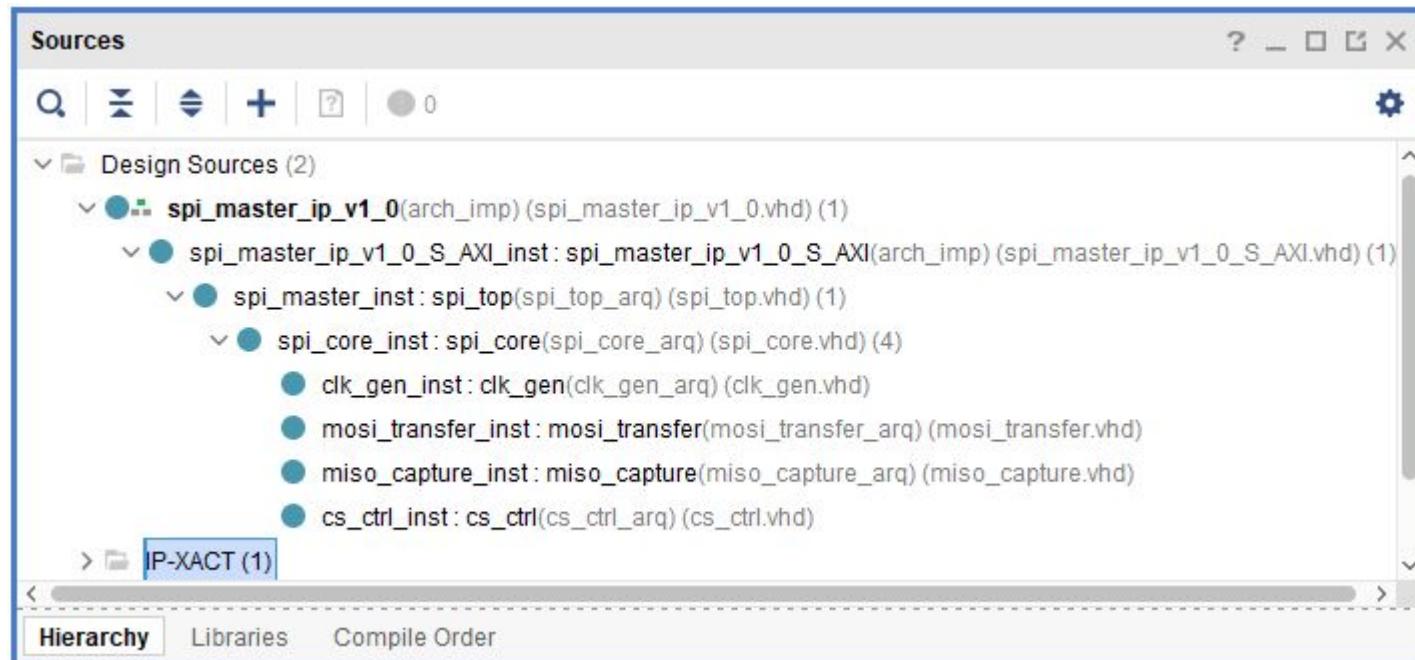
Clock Rate Hz: 125000000 **125MHz**

Sck Target Hz: 12500000 **12,5MHz**

C S Axi Data Width: 32

C S Axi Addr Width: 5

# Jerarquía del IP



# Mapeo de puertos y genéricos

spi\_master\_ip\_v1\_0\_S\_AXI.vhd

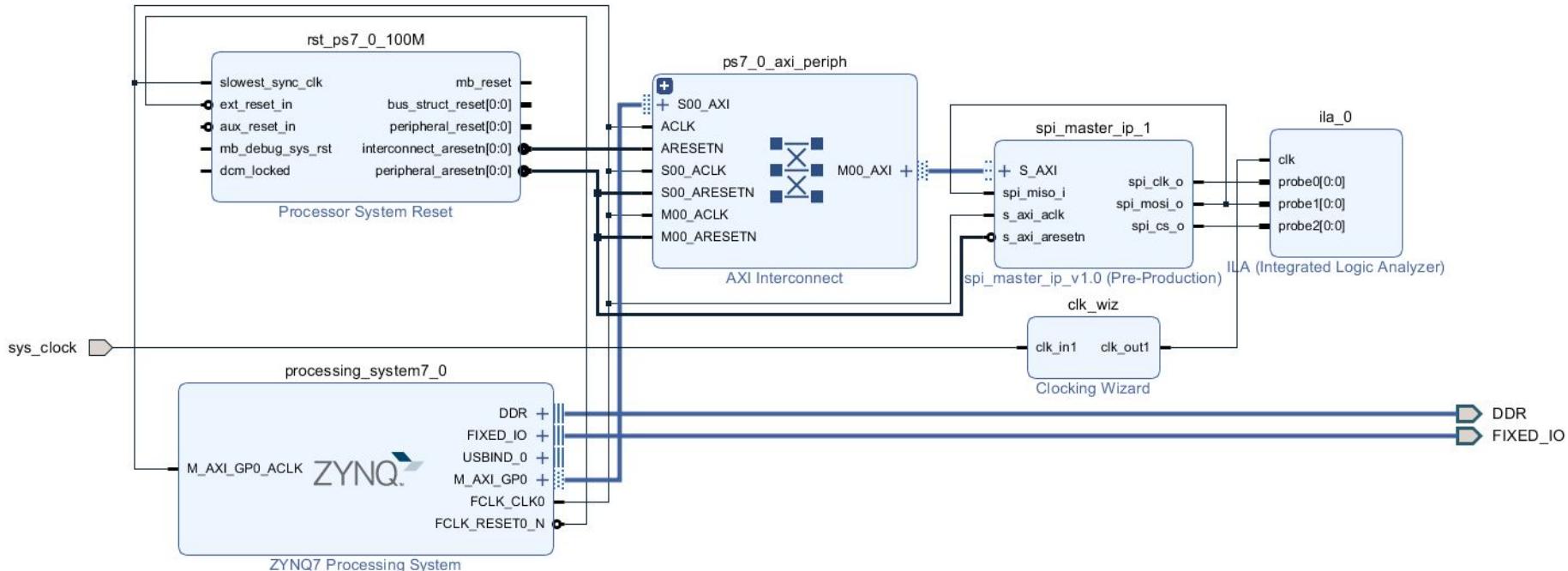
d:/CESE/5B2025/MyS/trabajo\_final/Repositorio/spi\_master\_ip/ip\_repo/spi\_master\_ip\_1.0/hdl/spi\_master\_ip\_v1\_0\_S\_AXI.vhd

```

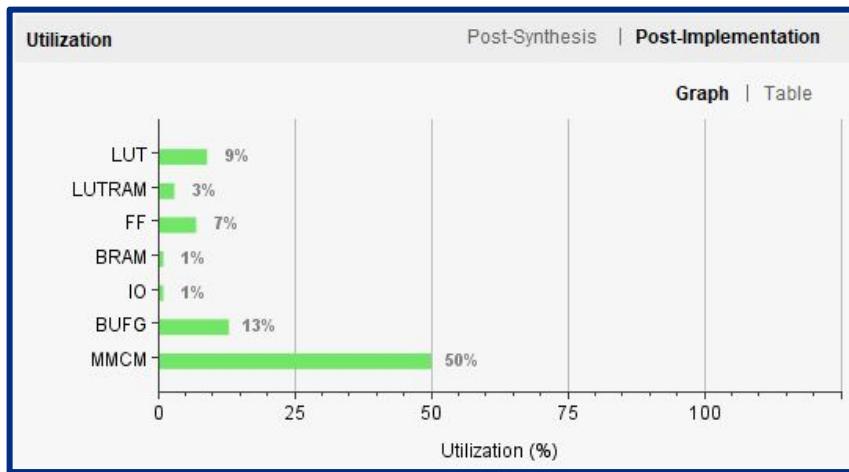
440      -- Add user logic here
441      spi_master_inst: spi_top
442      generic map(
443          DATA_SIZE      => DATA_SIZE,          -- largo de la trama
444          MODE          => MODE,              -- modo SPI
445          FIRST_BIT     => FIRST_BIT,        -- orden de transferencia
446          CLOCK_RATE_HZ => CLOCK_RATE_HZ,    -- clk_i => 125 MHz
447          SCK_TARGET_HZ => SCK_TARGET_HZ    -- spi_clk_o => 12,5 Mbps
448      )
449
450
451      port map(
452          -- Señales de control
453          rst_i        => S_AXI_ARESETN,       -- reset activo en bajo
454          clk_i        => S_AXI_ACLK,           -- reloj del sistema
455
456          -- Señales MOSI
457          tx_data_i   => slv_reg0(DATA_SIZE-1 downto 0),    -- dato de entrada a transmitir por spi_mosi_o
458          tx_dv_i     => slv_reg1(0),            -- pulso: indica dato valido en tx_data_i
459          tx_rdy_o    => slv_reg2_aux(0),         -- listo para aceptar un nuevo dato
460
461          -- Señales MISO
462          rx_data_o   => slv_reg3_aux(DATA_SIZE-1 downto 0),    -- dato de salida recibido por spi_miso_i
463          rx_dv_o     => slv_reg4_aux(0),            -- pulso: dato de entrada listo en rx_data_o
464
465          -- Interfaz SPI
466          spi_clk_o   => spi_clk_o,             -- linea SCK
467          spi_mosi_o  => spi_mosi_o,            -- linea MOSI (Master Out Slave In)
468          spi_miso_i  => spi_miso_i,            -- linea MISO (Master In Slave Out)
469          spi_cs_o    => spi_cs_o              -- linea CS (Chip Select)
470      );
471
472      -- User logic ends
473

```

# Block Design



# Utilización de recursos



**Utilization**

Post-Synthesis | Post-Implementation

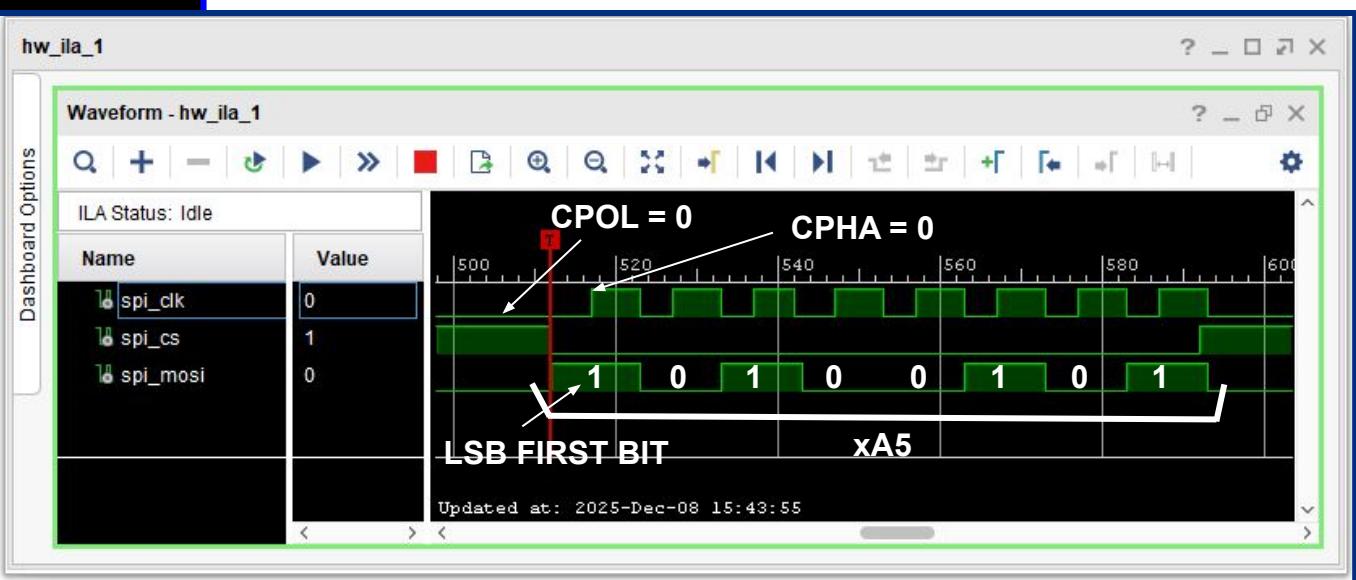
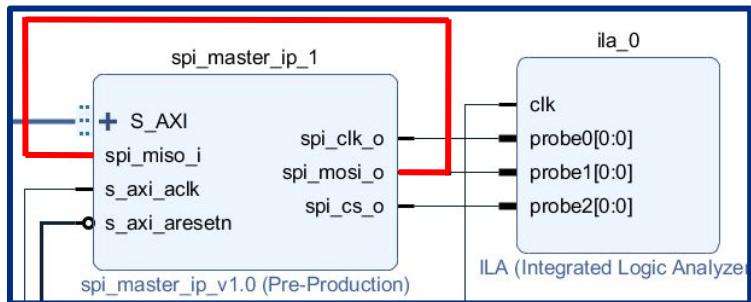
Graph | Table

Resource	Utilization	Available	Utilization %
LUT	1596	17600	9.07
LUTRAM	157	6000	2.62
FF	2456	35200	6.98
BRAM	0.50	60	0.83
IO	1	100	1.00
BUFG	4	32	12.50
MMCM	1	2	50.00

# Software de Prueba

```
86 int main(void)
87 {
88     uint8_t tx_byte = 0x00;
89     uint8_t rx_byte;
90
91     xil_printf("--- Prueba transmit/receive SPI Master IP ---\r\n");
92
93     while (1)
94     {
95         /* 1) Esperar disponibilidad del transmisor */
96         spi_wait_tx_ready();
97
98         /* 2) Iniciar transmisión del byte actual */
99         spi_start_transfer(tx_byte);
100
101        /* 3) Esperar y leer el byte recibido */
102        rx_byte = spi_wait_and_read_rx();
103
104        /* 4) Mostrar resultado */
105        xil_printf("tx_byte = 0x%02X, rx_byte = 0x%02X\r\n", tx_byte, rx_byte);
106
107        /* 5) Pausar entre transferencias */
108        sleep(5);
109
110        /* 6) Incrementar el dato a transmitir */
111        tx_byte += 1;
112    }
113
114    return 0;
115 }
```

# Resultados



**Fin de la presentación**

**Muchas gracias**