Diseño e implementación de módulo SPI

Trabajo práctico final

Circuitos Lógicos Programables

Ing. Guillermo Luis Castiglioni

(guillermo.castiglioni@gmail.com)

26/08/2022

Versión A

1. Introducción

1.1. Objetivo

El presente trabajo consiste en el diseño e implementación de un módulo SPI simplificado descripto en VHDL, simulado e implementado en un kit de FPGA.

1.2. Alcance

La versión del módulo desarrollada implementa únicamente el modo de operación 0 del protocolo SPI, con fines didácticos para evaluación de los contenidos dictados durante el curso de Circuitos Lógicos Programables.

2. Diseño

Para el desarrollo e implementación del módulo SPI propuesto se elaboró un diseño tipo estructural utilizando bloques con funcionalidades simples coordinados mediante una máquina de estados. Todos los bloques fueron descriptos en VHDL, simulados individualmente y luego integrados en el módulo completo.

2.1. Diagrama en bloques

En la figura 1 se muestra un diagrama en bloques del diseño planteados. Los objetos de almacenamiento, como registros y contadores, se implementaron de forma configurables en cantidad de elementos, brindando la posibilidad de modificar el ancho en bits de los datos a transmitir y recibir del módulo.

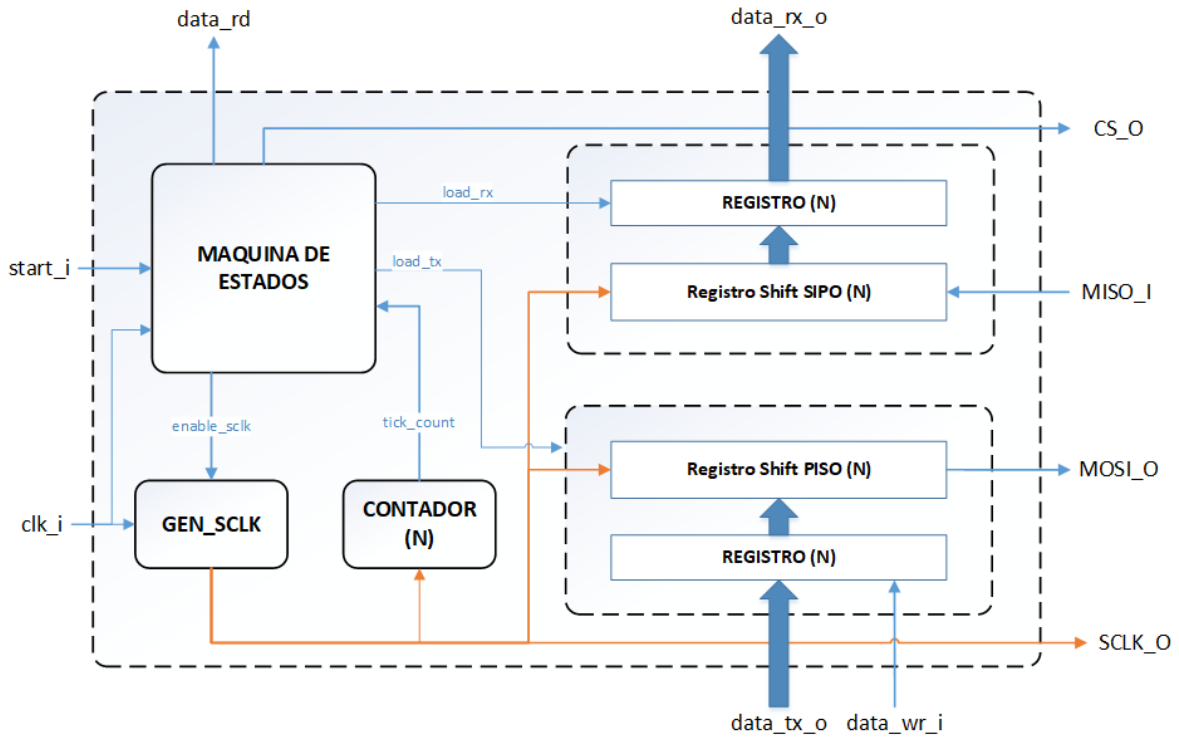


Figura 1: Diagrama de bloques

2.2. Interfaz de conexión

Nombre	Señal	Tipo	Descripción
Reloj de sincronismo	**clk_sys_i**	Lógica interna Entrada	Señal de reloj. Da el sincronismo de la lógica secuencial interna.
Reset sincrónico	**rst_sys_i**	Lógica interna Entrada sincrónica	Señal de reset activa en alto. Actúa sobre todos los registros y elementos de almacenamiento.
Reset asincrónico	**arst_sys_i**	Puerto I/O Entrada asincrónica	Señal de reset activa en alto. Actúa sobre todos los registros y elementos de almacenamiento.
Reloj SPI	**SCLK_O**	Puerto I/O Salida	Señal de reloj para transmisión y recepción de datos del protocolo SPI.
Señal de datos	**MOSI_O**	Puerto I/O Salida	Señal de datos para salida serie del módulo SPI.
Señal de datos	**MISO_I**	Puerto I/O Entrada asincrónica	Señal de datos para entrada serie del módulo SPI.
Selección de dispositivo	**CS_O**	Puerto I/O Salida	Señal de selección de dispositivo. Activa en bajo. Se mantiene en bajo durante el proceso de transmisión y recepción de datos.
Inicio de transmisión	**start_i**	Lógica interna Entrada	Señal de tick sincrónica que inicia la transmisión del dato almacenado en el registro correspondiente y la recepción de un dato.
Dato recibido	**data_rd_o**	Lógica interna Salida sincrónica	Señal de tick activa en alto que indica la recepción de un dato.
Escritura de dato	**data_wr_i**	Lógica interna Entrada sincrónica	Señal de tick para almacenar el dato presente en el bus data_tx_i en el registro de transmisión.
Bus de datos	**data_tx_i**	Lógica interna Entrada	Bus de datos de entrada.
Bus de datos	**data_rx_o**	Lógica interna Salida	Bus de datos de salida para lectura del dato recibido por el módulo

2.3. Herramientas utilizadas

Para el desarrollo del proyecto se utilizó:

- **ISE 14.7 Design Suite + ISim Simulator:** entorno de desarrollo del fabricante XILINX, utilizado para desarrollo del código y simulación de cada bloque.
- **GHDL + GTKWake:** compilación y simulación de código de cada bloque.
- **Quartus Prime Lite Edition**: entorno de desarrollo del fabricante INTEL, utilizado para desarrollo e implementación del código en el kit de FPGA.
- **Placa de desarrollo DE1-SoC Board:** fabricante Terasic, FPGA Cyclone V SoC del fabricante Altera.

2.4. Código

A continuación se adjunta el enlace al repositorio donde encuentra el código completo elaborado.

https://github.com/gcasti/CESE_CLP

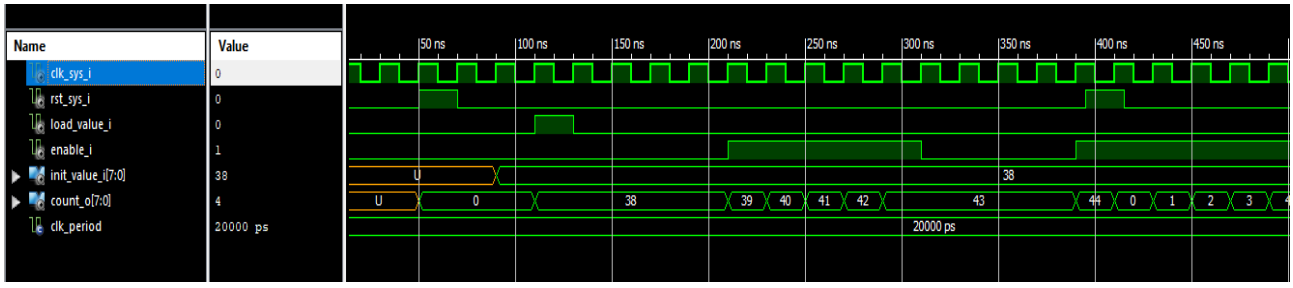
2.5. Simulaciones principales

Todas las simulaciones se realizaron utilizando ISim y *scripts* de bash para el uso de GHDL+GTKWake. A continuación se muestran las capturas de los principales módulos.

En todas las simulaciones se utilizaron 8 bits como cantidad de elementos de almacenamiento y de operación del módulo completo

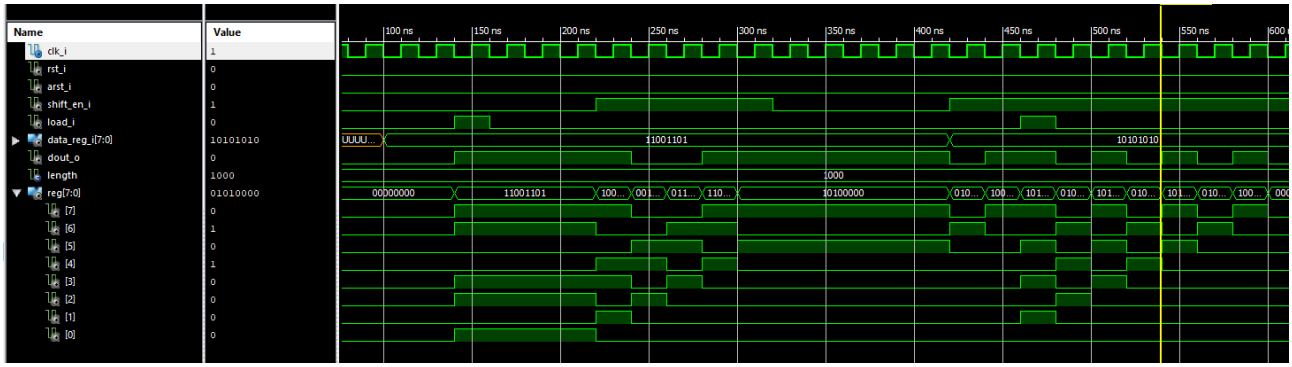
Contador configurable

Contador sincrónico ascendente configurable con carga de valor inicial.



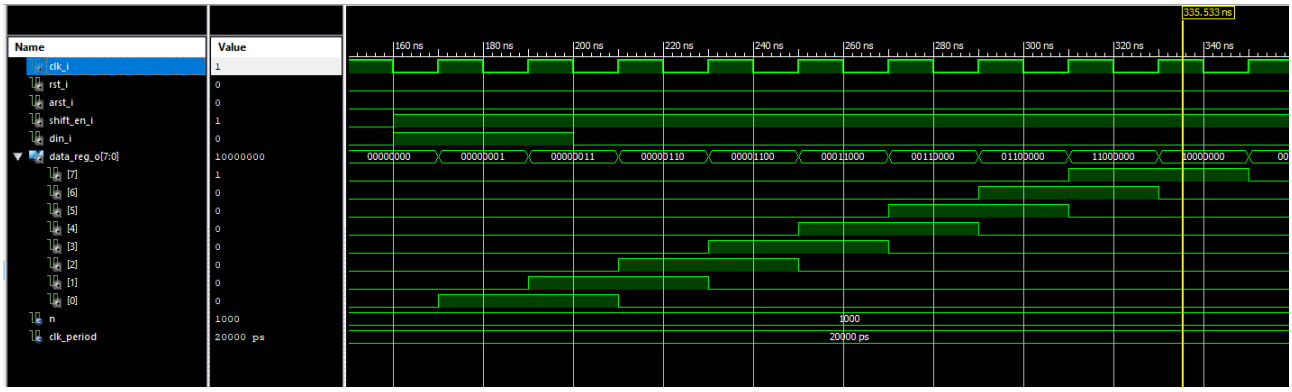
Registro de desplazamiento PISO

Registro de desplazamiento sincrónico con carga de datos paralela y salida serie.



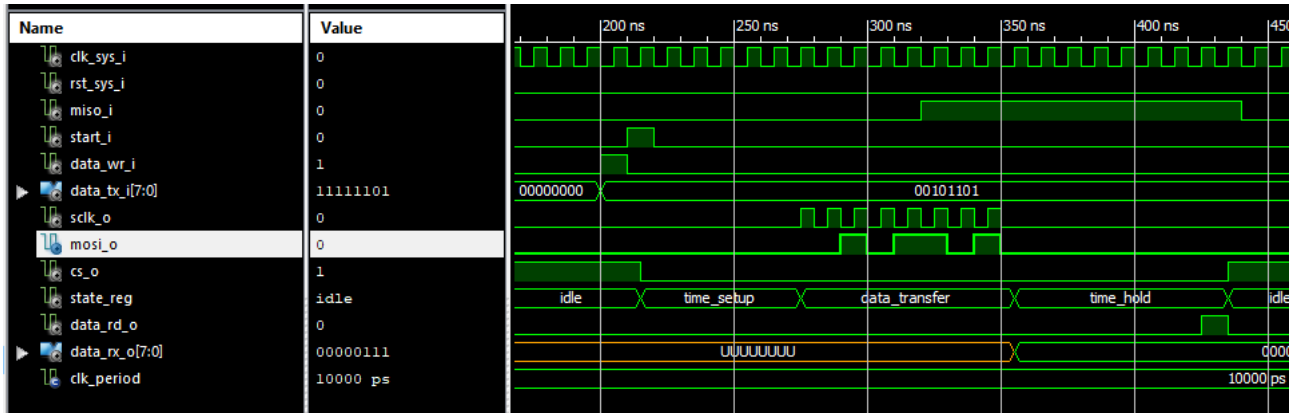
Registro de desplazamiento SIPO

Registro de desplazamiento sincrónico con entrada de datos serie y lectura de datos en paralelo.

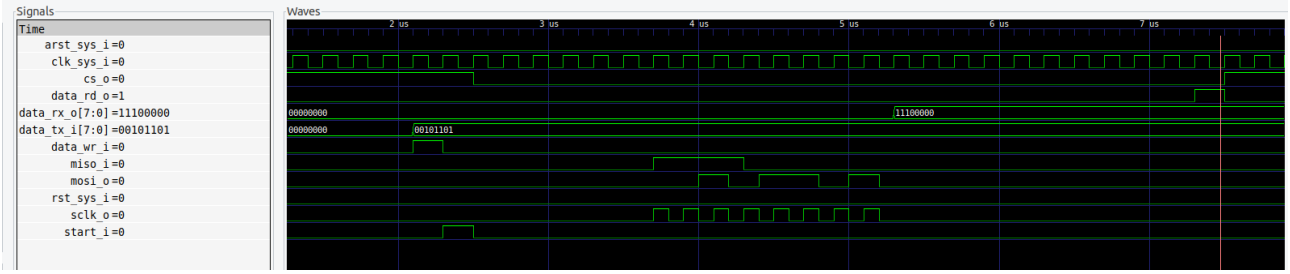


Módulo completo

Simulación funcional del módulo completo utilizando un periodo de reloj de 20 ns.



Simulación completa del módulo utilizando un periodo de reloj de 200 ns, correspondiente a una frecuencia de 5 Mhz utilizada para las pruebas en el kit de la FPGA.



2.6. Máquina de estado

Para coordinar la operación de todos los componentes se implementó una máquina de estados. En la figura 2 se muestra un diagrama generado con la herramienta integrada en la Quartus Prime Lite Edition y a continuación se describe brevemente el propósito de cada estado.

INIT: inicialización de elementos de almacenamiento.

IDLE: espera por señal de inicio de recepción y transmisión de datos.

TIME_SETUP: carga del dato a transmitir desde el registro correspondiente al registro de desplazamiento PISO. A su vez, se activa un temporizador y se espera el tiempo configurado como TIME SETUP del módulo SPI.

DATA_TRANSFER: se activa la generación de los pulsos de reloj de la señal SCLK y se permanece hasta que el contador alcance el valor de flancos configurados correspondientes a la cantidad de bits del módulo.

TIME_HOLD: se activa un temporizador y se espera el tiempo configurado como TIME HOLD del módulo SPI. Una vez alcanzado el tiempo correspondiente se transfiere el dato recibido a un registro de recepción y luego se regresa al estado IDLE.

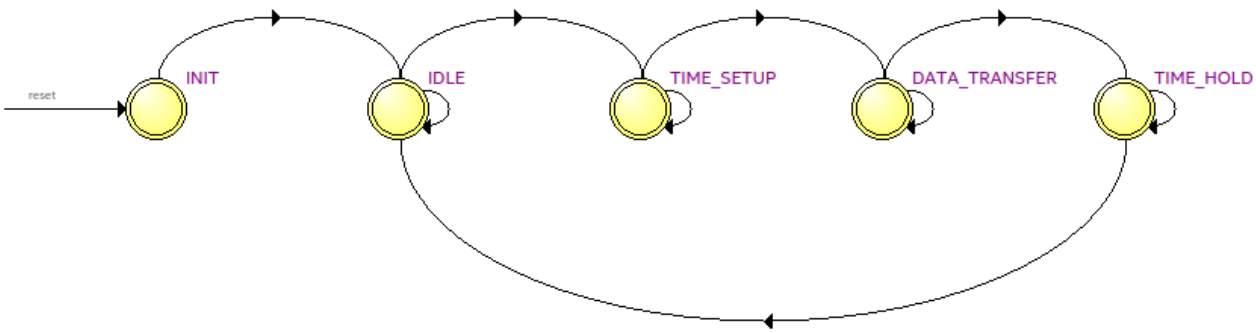


Figura 2: Diagrama de máquina de estados

2.7. Diagramas RTL

En la figura 3 se muestra un diagrama RTL generado con la herramienta integrada en Quartus Prime Lite Edition.

En verde se distinguen los bloques instanciados como componentes internos, mientras que en amarillo se distingue la máquina de estados.

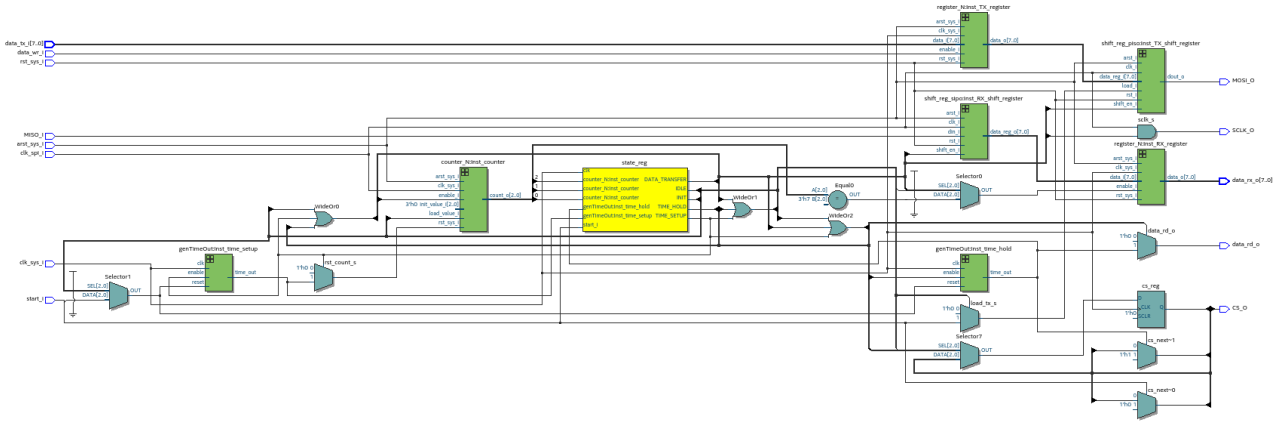


Figura 3: Diagrama RTL.

2.8. Reportes

El módulo completo fue implementado utilizando el kit de desarrollo mencionado en las herramientas utilizadas completando satisfactoriamente el proceso de análisis y síntesis, luego *Place and Route(Fitter)* hasta obtener el archivo binario para grabado de la FPGA.

Analysis & Synthesis Summary

<<Filter>>

Analysis & Synthesis Status	Successful - Thu Sep 1 13:49:11 2022
Quartus Prime Version	21.1.1 Build 850 06/23/2022 SJ Lite Edition
Revision Name	MasterSPI
Top-level Entity Name	MasterSPI
Family	Cyclone V
Logic utilization (in ALMs)	N/A
Total registers	48
Total pins	27
Total virtual pins	0
Total block memory bits	0
Total DSP Blocks	0
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0
Total DLLs	0

Fitter Summary

<<Filter>>

Fitter Status	Successful - Thu Sep 1 13:49:39 2022
Quartus Prime Version	21.1.1 Build 850 06/23/2022 SJ Lite Edition
Revision Name	MasterSPI
Top-level Entity Name	MasterSPI
Family	Cyclone V
Device	5CSEMA5F31C6
Timing Models	Final
Logic utilization (in ALMs)	27 / 32,070 (< 1 %)
Total registers	49
Total pins	27 / 457 (6 %)
Total virtual pins	0
Total block memory bits	0 / 4,065,280 (0 %)
Total RAM Blocks	0 / 397 (0 %)
Total DSP Blocks	0 / 87 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total PLLs	0 / 6 (0 %)
Total DLLs	0 / 4 (0 %)

3. Conclusiones

El trabajo permitió completar todo el ciclo de desarrollo e implementación de un módulo digital en una FPGA aplicando los conceptos vistos a lo largo del curso en una plataforma disponible y utilizando múltiples herramientas. Durante la presentación del trabajo se mostraron las herramientas anexas para probar el funcionamiento del módulo sobre el kit de desarrollo y se mencionaron las mejoras posibles a implementar ampliando las capacidades del módulo.