DISEÑO DE UN IP CORE PARA UNA INTERFAZ SPI

DESIGN OF AN IP CORE FOR A SPI INTERFACE

Jesús García González
Universidad Autónoma de Zacatecas, México
jesus.garcia.g92@hotmail.com

Remberto Sandoval Aréchiga
Universidad Autónoma de Zacatecas, México
rsandoval@uaz.edu.mx

Salvador Ibarra Delgado
Universidad Autónoma de Zacatecas, México
sibarra@uaz.edu.mx

Recepción: 29/abril/2020 **Aceptación:** 28/octubre/2020

Resumen

Este trabajo presenta el desarrollo de un IP CORE (*Intellectual Property Core*, núcleo de propiedad intelectual) con la funcionalidad de una interfaz SPI (*Serial Peripheral Interface,* interfaz periférica serial) de alta velocidad y con diferentes frecuencias de operación, cuyo objetivo es establecer comunicación (transmisión/recepción de datos) entre la tarjeta de desarrollo Zedboard a otros dispositivos. El IP CORE de la interfaz SPI está diseñado en base a una simple metodología, donde tiene por elementos principales un registro de desplazamiento PISO (*Parallel Input Serial Output,* entrada paralela salida serial) para la transmisión y un registro SIPO (*Serial Input Parallel Output,* entrada serial salida paralela) para la recepción de datos. Dicho módulo fue diseñado y verificado en el software Vivado, con el diseño de bancos de prueba (*test bench*), donde se obtuvieron resultados aceptables que prueban la funcionalidad y confiabilidad del módulo.

Palabras clave: Diseño, IP CORE, sistema embebido, SPI.

Abstract

This work presents the development of an IP CORE (Intellectual Property Core) with the funcionality of a high speed interface Serial Peripheral Interface (SPI) with

diferent operating frequencies, whose objective is to establish communication (data transmission / reception) between the Zedboard development card to other devices. The IP CORE of the SPI interface is designed based on a simple methodology, where its main elements are a PISO (Parallel Input Serial Output) shift register for transmission and SIPO (Serial Input Parallel Output) register for data reception. Said module was designed and verified in the Vivado software, with the design of test bench, where aceptable results were obtained that prove the functionality and reliability of the module.

Keywords: *Design, IP CORE, embedded system, SPI.*

1. Introducción

En los últimos años, el campo de las comunicaciones y la tecnología inalámbrica ha presentado enormes cambios en diversos dispositivos, protocolos y aplicaciones. Principalmente en dispositivos inalámbricos como: celulares, televisores, radio, dispositivos médicos y dispositivos militares, cada vez más pequeños y con mayor cantidad de funciones; todo ello gracias al diseño electrónico de sistemas embebidos o IP CORES, los cuales han presentado algunas ventajas como: bajo consumo de energía, menor tamaño, tecnología más barata, así como menor consumo de memoria. El presente trabajo describe el diseño de un IP CORE para una interfaz SPI, el cual ofrece las siguientes facilidades: capacidad de conexión con varios maestros o esclavos para establecer comunicación, con una capacidad de transmisión y recepción de datos de hasta 1024 bits en una sola transacción en modo full duplex (transmisión/recepción al mismo tiempo), además puede transmitir a 3 frecuencias diferentes: 1, 8, o 16 MHz, entre la tarjeta Zedboard y otros dispositivos [1].

El protocolo SPI es utilizado principalmente para comunicaciones entre circuitos integrados por su simplicidad y por su velocidad de transmisión. El SPI es un protocolo síncrono que trabaja en modo full duplex, esto quiere decir que puede recibir y transmitir información al mismo tiempo [2]. En la figura 1 se muestra la estructura general del protocolo. El protocolo SPI, básicamente se conforma de 2 bloques: el maestro y el esclavo (o pueden ser varios esclavos, figura 1). Donde el

maestro envía datos de manera serial al esclavo y también recibe datos de manera serial del esclavo.

Figura 1 Conexiones del Bus SPI.

Existen 4 modos en los que se puede transmitir y recibir información dependiendo de dos parámetros basados en la señal de reloj. El primero es la polaridad (CPOL) y el segundo la fase (CPHA).

- Modo 0: CPOL=0 y CPHA=0. Modo en el cual el estado de reloj permanece en estado lógico bajo y la información se envía en cada transición de bajo a alto, es decir alto activo.
- Modo 1: CPOL=0 y CPHA=1. Modo en el cual el estado de reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.
- Modo 2: CPOL=1 y CPHA=0. Modo en el cual el estado de reloj permanece en estado lógico alto y la información se envía en cada transición de alto a bajo, es decir bajo activo.
- Modo 3: CPOL=1 y CPHA=1. Modo en el cual el estado de reloj permanece en estado lógico alto y la información se envía en cada transición de bajo a alto, es decir alto activo.

La figura 2, ilustra los modos de operación explicados en el párrafo anterior, con los que el SPI puede operar. Cabe mencionar que para el presente diseño se utiliza el modo 0, como modo de operación del SPI.Se eligió el protocolo de comunicación serial SPI por su simplicidad de uso y factibilidad en altas frecuencias de operación. El IP CORE SPI modelado, en un trabajo a futuro será implementado en un sistema

embebido de hardware controlado por software, para el control de la comunicación de datos entre la tarjeta Zedboard o cualquier otra tarjeta que integre una comunicación serial SPI, para altas o diferentes velocidades de transmisión y recepción de datos, como los son: Nexys4 DDR, FMCOMMS4-EBZ, arduino, otras.

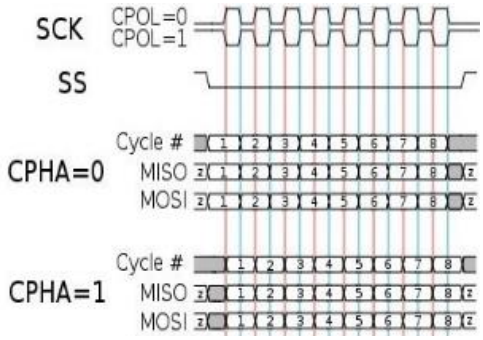

Figura 2 Modos de operación del SPI.

En el siguiente apartado se muestra la metodología usada para el desarrollo del diseño de un IP CORE para una interfaz SPI.

2. Métodos

El diseño de un IP CORE para una interfaz SPI se basa en un diseño Top-Down (de lo general a lo específico), y para ello se utiliza la siguiente metodología [3]:

- Diagrama de caja negra. El diagrama de caja negra es una representación gráfica del módulo, donde se definen las posibles señales de entrada y salida para el diseño de un IP CORE.
- Descripción funcional del sistema. En la descripción funcional se define el comportamiento en general del sistema embebido o IP CORE a través de pseudocódigo (descripción de alto nivel compacta e informal del principio operativo de un programa o algoritmo) o diagramas de flujo.
- Diagrama de caja blanca del sistema. El diagrama de caja blanca es una representación gráfica donde se muestra la arquitectura interna del sistema (o circuito), la relación que existe entre las entradas y salidas del mismo, es decir, los circuitos que conforman internamente el sistema, así como las interconexiones entre ellos.

- Se repiten los pasos anteriores (1 al 3) para cada componente del sistema. En este paso se repiten los 3 pasos anteriores (diagrama de caja negra, descripción funcional y diagrama de caja blanca) para cada componente interno del sistema.
- Codificación y sintaxis. En la codificación a través de lenguaje descriptor de hardware “HDL” (del inglés Hardware Description languaje) como vhdl, verilog, o system verilog se describe el funcionamiento del circuito, y con el uso del sintetizador se verifica la correcta escritura del código, así como convierte dicho código a su equivalente en hardware.
- Simulación e implementación del circuito. En la simulación se verifica el correcto funcionamiento del circuito o sistema digital, y en la implementación se refiere a las pruebas funcionales del sistema o IP CORE (en este caso se utiliza la tarjeta de desarrollo de sistemas digitales Zedboard para establecer comunicación vía SPI con otros dispositivos).

En figura 3 se muestra un diagrama general de la metodología empleada, la cual hace referencia a los puntos (1-6) explicados con anterioridad y que fue utilizada para el modelado del IP CORE SPI:

- **Desarrollo del diseño**. En este apartado se muestra el desarrollo del diseño de un IP CORE para una interfaz SPI en base a la metodología explicada en el tema 2.

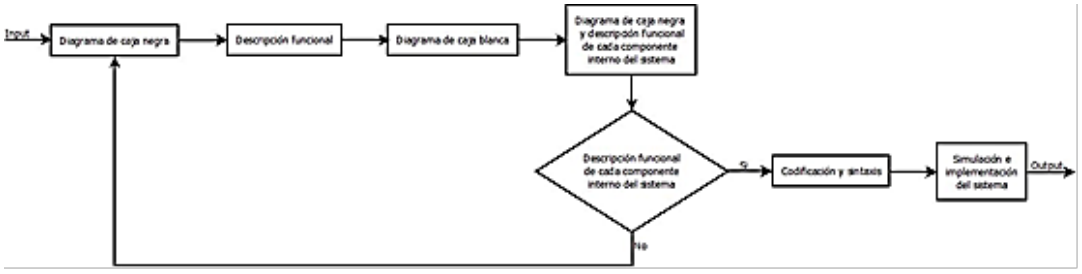

Figura 3 Metodología de diseño.

- **Diagrama de caja negra** . El diagrama de caja negra con sus entradas y salidas del diseño del CORE SPI se ilustra en la figura 4. Para transmitir datos con el IP CORE SPI se tiene que escribir el dato en la señal de entrada

Data_in, y este puede ser de n bits (8,12,16,32, 64...,1024), el cual es transmitido de manera serial a través de la señal MOSI (a una velocidad de 1, 8 o 16 MHz), pero también, al mismo tiempo puede recibir datos (n bits: 8,16,32,64…,1024) de manera serial a través de la señal MISO, los cuales se pueden ver en el registro de salida Data_out. La tabla 1 muestra el funcionamiento de cada señal, así como su tamaño y dirección.

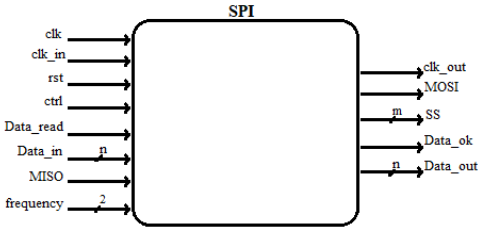

Figura 4 Diagrama de caja negra.

Tabla 1 SPI.

Bus name	Dirección	No. de bits	Descripción
Data_in	input	8	Señal de entrada de datos con tamaño de 1 byte “8 bits” (parametrizable n bits).
MISO	input	1	De sus siglas en inglés *Master In Slave Out*, señal del protocolo SPI encargada de recibir información del esclavo(s) al maestro de manera serial.
Data_read	input	1	Señal de control de un bit (1/0), la cual cuando su estado es alto indica al SPI que puede iniciar una transacción.
rst	input	1	Señal de reset utilizada para inicializar el sistema SPI con los datos de preferencia.
clk	input	1	Señal de reloj, encargada de sincronizar el IP CORE SPI.
clk_out	output	1	Señal de reloj, encargada de sincronizar a un esclavo (o varios esclavos).
MOSI	output	1	De sus siglas en inglés *Master Out Slave In*; señal del protocolo SPI encargada de transmitir datos del maestro hacia el esclavo(s) de manera serial.
Data_ok	output	1	Señal de control de 1 bit, la cual se pone en alto cuando el maestro ha recibido 8 bits a través de la señal MISO, de lo contrario se encuentra en estado bajo.
Data_out	output	8	Señal de salida de 1 byte (parametrizable n bits), encargada de mostrar los 8 bits recibidos por MISO, si Data_ok se encuentra en alto.
clk_in	input	1	Señal de reloj de entrada para la recepción de datos cuando el SPI se encuentra en modo esclavo.
ctrl	input	1	Señal de ctrl mediante la cual se define el comportamiento del SPI, como esclavo o como maestro.
SS	output	1	Señal de salida de un bit (m parametrizable) con la cual se elige al esclavo con el cual se realizará una transacción, así como su habilitación.
frequency	input	2	Señal de entrada de 2 bits con la cual se puede especificar a qué frecuencia se desea transmitir: a 1, 8 o 16 MHz.

- **Descripción funcional**. En la descripción funcional, se describe el funcionamiento de manera general del circuito o sistema digital a través del diagrama de flujo de la figura 5.
- **Diagrama de caja blanca**. El Diagrama de caja blanca se ilustra en la figura 6, donde muestra la relación que existe entre las entradas y salidas del sistema descrito en el diagrama de flujo de la figura 5.

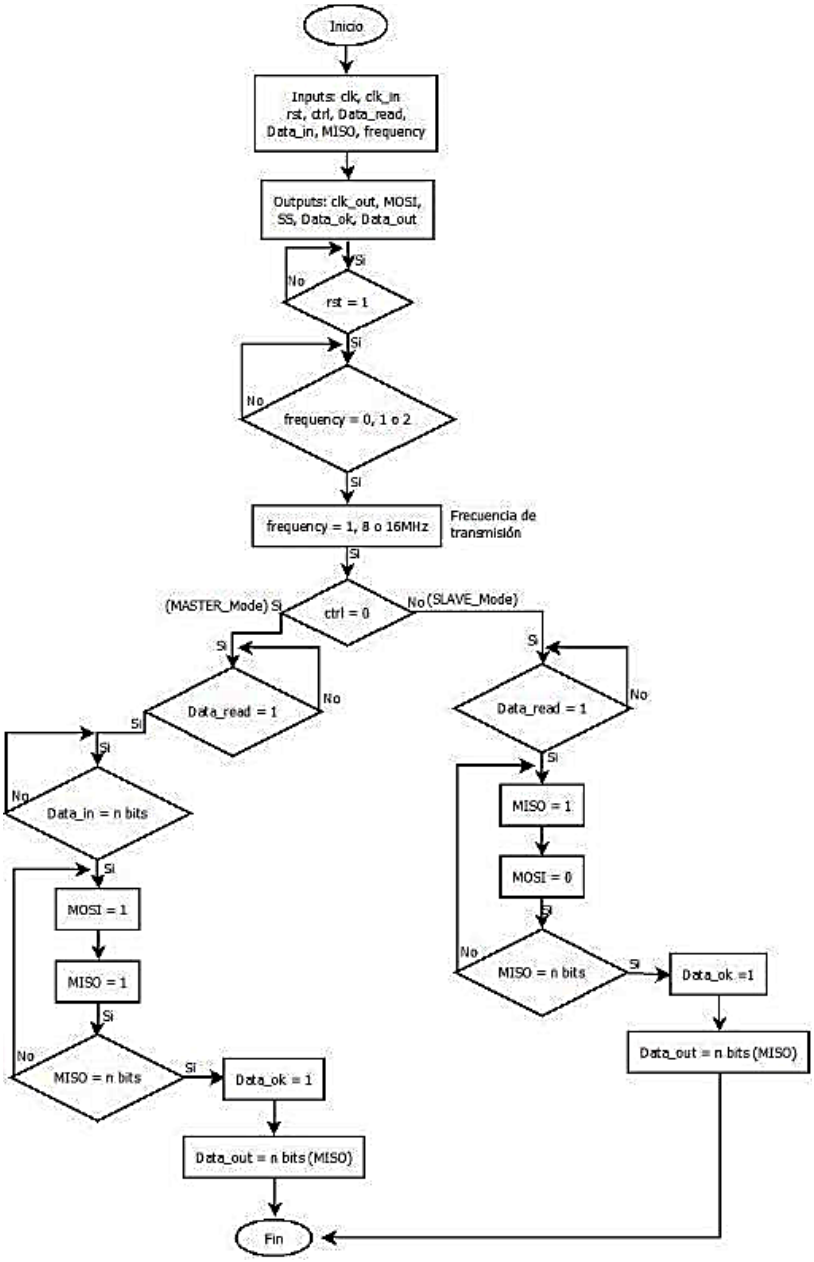

Figura 5 Diagrama de Flujo del SPI.

Este módulo SPI se conforma de 7 circuitos: una máquina de estados “FSM” (*Finite State Machine*) que controla las señales, un registro PISO (Parallel Input Serial Output), un preescalador de 1 MHz que sincroniza los circuitos, un registro SIPO (Serial Input Parallel Output), un registro de entrada/salida paralela PIPO (*Parallel Input Parallel Output*), y 2 multiplexores.

En figura 6 podemos ver lo siguiente:

- ✓ *Registro PISO “reg_piso”*. – Circuito encargado de recibir o almacenar datos de manera paralelo (datos de la señal “Data_in”) para su posterior transmisión de manera serial (“MOSI”).

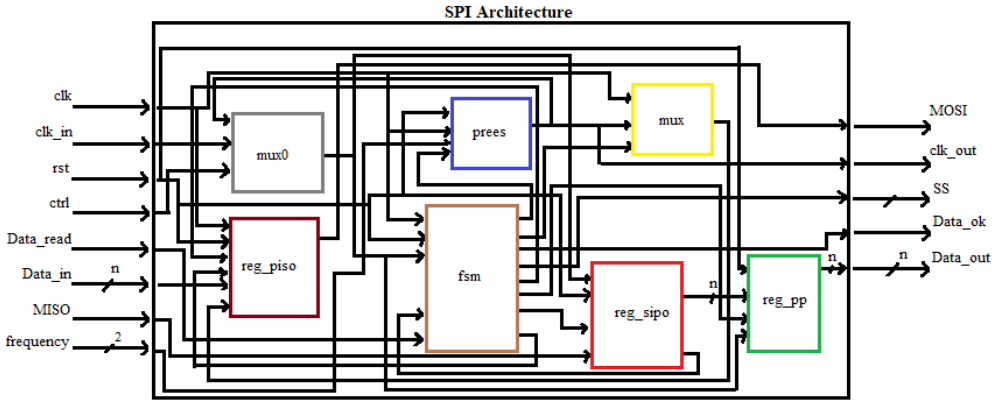

Figura 6 Diagrama de caja blanca.

- ✓ *Registro SIPO “reg_sipo”*. – Circuito encargado de transmitir o enviar de manera paralelo los datos recibidos de manera serial a través de la señal “MISO”, así mismo encargado de enviar una señal de habilitación a la máquina de estados, cuando se recibió la cantidad de bits (n bits) esperada a través de la señal “MISO”.
- ✓ *Multipexor “mux0”*. - Circuito encargado de definir el comportamiento del SPI “esclavo/maestro”. Si es esclavo el SPI trabajará internamente a la frecuencia que envíe el maestro, de lo contrario operará a la frecuencia preestablecida.
- ✓ *Multiplexor “mux”*. - Circuito de encargado de multiplexar la señal de reloj de entrada del registro piso, el cual recibe el dato de la señal

"Data_in" a una frecuencia y lo transmite de manera serial a otra frecuencia.

- ✓ *Preescalador "prees"*. – Preescalador, circuito encargado de dividir la frecuencia de 50 MHz a 1, 8 o 16 MHz, así como sincronizar los circuitos internos del sistema.
- ✓ *Registro PIPO "reg_pp"*. – Circuito encargado de recibir de manera paralelo (n bits) los datos enviados por el circuito SIPO y de transmitir los mismos datos de igual manera (paralelo) a través de la señal "Data_out". Este circuito fue requerido para la recuperación de un bit perdido por el retardo de un ciclo de reloj que cada registro provoca.
- ✓ *FSM*. - Máquina de estados, circuito encargado de controlar la señal de habilitación para la entrada de datos del registro PISO, la señal "Data_ok", así como otras señales.

3. Resultados

Una vez terminada la codificación del CORE SPI con HDL (*Hardware Description Languaje*, lenguaje descriptor de hardware) en el software Vivado, se realizó una simulación del sistema a través de bancos de prueba (*test bench*), con el objetivo de probar el funcionamiento del IP CORE SPI para cerciorar que se obtuvieran los datos esperados al transmitir y recibir información, figura 7.

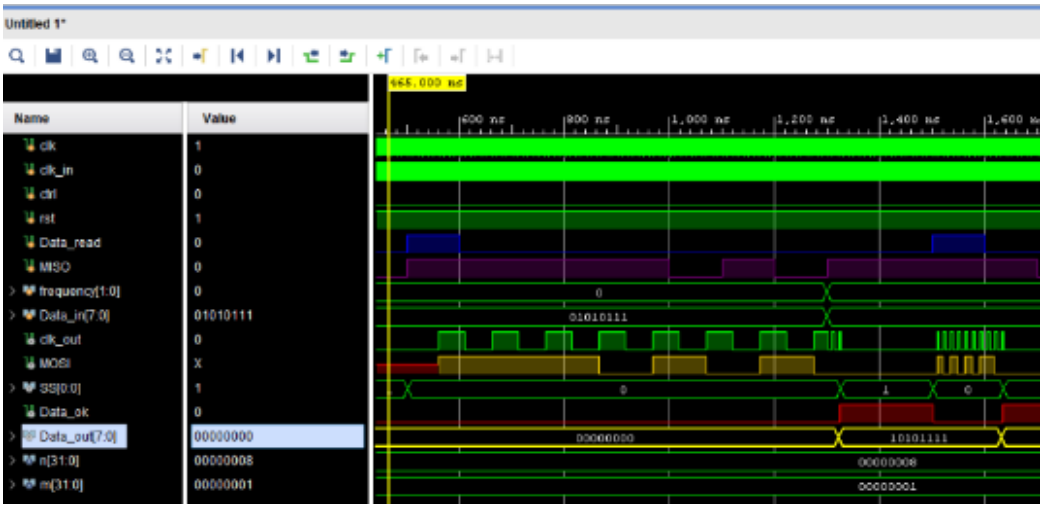

Figura 7 Simulación del SPI.

Como puede observarse, de acuerdo con cada transición positiva del reloj, mientras la señal de control Data_read (señal de color azul) se encuentre en alto o recibe un pulso positivo, si se tiene un byte, es decir 8 bits (“n bits”: 0-1023) en la entrada Data_in (señal de color verde), la señal MOSI (señal de color dorado) comienza a transmitir de manera serial dicho dato a la frecuencia especificada.

Por otra parte, la señal de salida Data_ok (señal de color rojo), se pone en alto, cuando se han recibido 8 bits (n bits dependiendo el tamaño de operación del SPI) a través de la señal MISO (señal de color purpura), entonces la señal Data_out (señal de color amarillo) transmite de manera paralelo esos 8 bits.

4. Conclusiones

Una diferencia importante sobre otros IP CORES SPI como el de Xilinx, es que el número de datos a operar es parametrizable, puede trabajar con 8, 12, 16, 32 e inclusive hasta 1024 bits para la transmisión y recepción de datos en una sola operación; otra cualidad del SPI modelado es que el usuario puede elegir la frecuencia para la transmisión de datos entre 1, 8, o 16 MHZ.

Para la recepción de datos, si se desea recibir un dato en especial, es recomendable tener cuidado con la señal de habilitación (Data_read) del módulo SPI; basta habilitar (poner en alto 1) dicha señal en un ciclo de reloj para ejecutar solo una operación, de lo contrario mientras la señal de habilitación esté activa el SPI seguirá recibiendo datos.

Se pretende realizar un controlador estándar con el SPI modelado.

Un trabajo a futuro es realizar un sistema embebido de hardware controlado por software, el sistema en hardware será diseñado en el software Vivado, y dicho sistema con la herramienta SDK (*Software Development Kit,* equipo para desarrollo de software) de Vivado será programado en el lenguaje de programación C, para controlar la transmisión y recepción de datos entre la tarjeta de desarrollo Zedboard o cualquier otra tarjeta que integre el sistema en chip o SoC (System on Chip por sus siglas en inglés) Zynq 7000, hacia otros dispositivos que utilicen el protocolo de comunicación serial SPI.

5. Agradecimientos

Se agradece al Centro de Investigación e Innovación en Desarrollo para Telecomunicaciones “CIIDTE” por las facilidades otorgadas a través de su equipo de instrumentación para el desarrollo del presente trabajo, así como la atención y asesorías de los investigadores y colaboradores del CIIDTE.

6. Bibliografía y Referencias

[1] Li-li Li, J.-y. H.-p.-h. (2014). Design of Microcontroller Standard SPI Interface. Applied Mechanics and Materials, 7.

[2] Navarro, K. (2014). ¿Cómo funciona el rpotocolo SPI? panamahitek, 4.

[3] Schweers, R. J. (2002). Descripción en VHDL de arquitecturas para implementar el algoritmo CORDIC. Buenos Aires, Argentina: Universidad Nacional de la Plata.