Ernst-Abbe-Fachhochschule Jena
Hochschule für angewandte Wissenschaften

Spezifikation

Implementierung eines SPI-Slaves mittels VHDL

Marc Ludwig

Jena, 12. Juli 2013

Betreuer: Prof. Dr. Ing. Jürgen Kampe
Dipl. Ing. Oliver Reimer

Inhaltsverzeichnis

Abstract

Das Serial Peripheral Interface (kurz SPI) ist ein von Motorola entwickeltes Bus-System mit einem sehr lockeren Standard für einen synchronen seriellen Datenbus (Synchronous Serial Port), mit dem digitale Schaltungen nach dem Master-Slave-Prinzip miteinander verbunden werden können. Ein ähnliches Bus-System existiert von National Semiconductor und nennt sich Microwire.
Es können theoretisch beliebig viele Teilnehmer an den Bus angeschlossen werden, wobei es immer genau einen Master geben muss. Er ist derjenige, der das Clock-Signal an SCK erzeugt und festlegt, mit welchem Slave er kommunizieren will. Das geschieht über die Leitung "Slave Select". Wird sie gegen Masse gezogen, wird der jeweilige Slave aktiv, "lauscht" an MOSI und legt seine Daten im Takt von SCK an MISO. Es wird somit ein Byte vom Master zum Slave und ein anderes Byte vom Slave zum Master transportiert.

1 Spezifikation des zu entwerfenden Systems in Text und Bild

1.1 Erläuterung der Aufgabe

Zur Implementierung des SPI-Slaves sind folgende Entwurfsschritte von nöten:

- algorithmische Ebene
  - Entwurf eines VHDL-Modells auf algorithmischer Ebene mit zugehöriger Testbench
  - Funktionale Simulation der algorithmischen Ebene
- Register Transfer Ebene
  - Entwurf eines VHDL-Modells auf Register Transfer Ebene mit zugehöriger Testbench
  - Funktionale Simulation der RT-Ebene vs. algorithmischer Ebene
  - Timing Simulation des Ergebnises
- Die Realisierung muss auf dem DE2-Evaluierungsboard lauffähig sein

1.2 Beschreibung der Ein- und Ausgänge

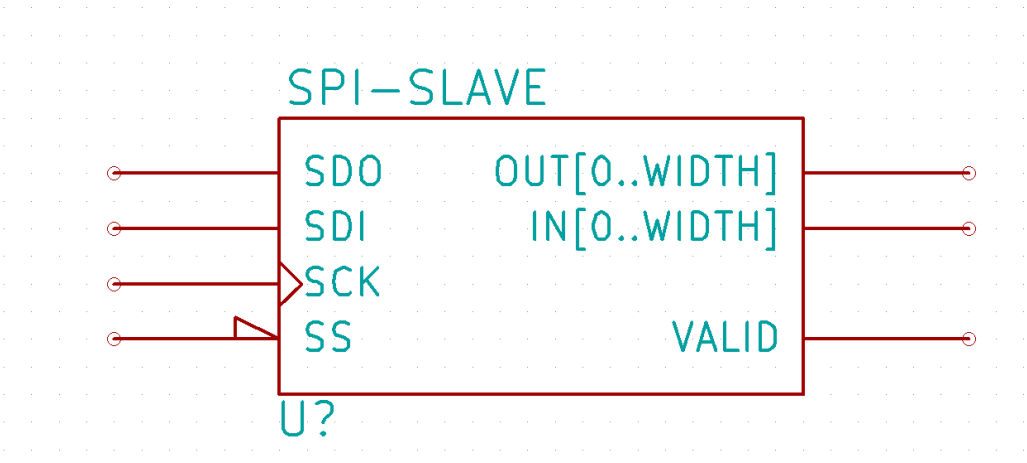

Abbildung 1.1: Blockschaltbild des Slaves (ohne generische Parameter)

Es existieren drei gemeinsame Leitungen, an denen jeder Teilnehmer angeschlossen ist:

SDO (englisch Serial Data Out) bzw. MISO oder SOMI (englisch Master in, Slave out)

SDI (englisch Serial Data In) bzw. MOSI oder SIMO (englisch Master out, Slave in)

SCK (englisch Serial Clock) bzw. SCLK, wird vom Master ausgegeben

OUT "Ausgaberegeister" Parallele Ausgänge des Ausgangswort, je nach Wortbreite über *generic* parametrisiert

IN "Eingangsregeister" Parallele Eingänge, je nach Wortbreite über *generic* parametrisiert

VALID Ausgang, welcher einen Impuls(halbe Periodenbreite) generiert wenn das Eingangswort gültig ist

Eine logisch-0 aktive Chip-Select-Leitungen, welche vom Master gesteuert wird. Diese Leitung wird je nach Anwendung unterschiedlich mit Bezeichnungen wie SS, CS oder STE für Slave Select, Chip Select bzw. Slave Transmit Enable bezeichnet, oft noch in Kombination mit einer Indexnummer zur Unterscheidung. Es gibt auch spezielle Anwendungen, bei denen sich mehrere Slaves diese Leitung teilen. Viele Einstellmöglichkeiten, beispielsweise mit welcher Taktflanke ausgegeben bzw. eingelesen wird, Wortlänge, Übertragung MSB oder LSB zuerst.

Viele Einstellungsmöglichkeiten sind unter anderem deshalb erforderlich, weil sich die Spezifikation für den SPI-Bus in vielen Dingen nicht konkret festlegt und deshalb verschiedene, zueinander inkompatible Geräte existieren. So ist häufig für jeden angeschlossenen Schaltkreis eine eigene Konfiguration des steuernden Mikrocontrollers (Master des SPI-Bus) erforderlich.

Es können theoretisch beliebig viele Teilnehmer an den Bus angeschlossen werden, wobei es immer genau einen Master geben muss. Er ist derjenige, der das Clock-Signal an SCK erzeugt und festlegt, mit welchem Slave er kommunizieren will. Das geschieht über die Leitung "Slave Select". Wird sie gegen Masse gezogen, wird der jeweilige Slave aktiv, "lauscht" an MOSI und legt seine Daten im Takt von SCK an MISO. Es wird somit ein Byte vom Master zum Slave und ein anderes Byte vom Slave zum Master transportiert.

Ein Protokoll für die Datenübertragung wurde von Motorola zwar nicht festgelegt, doch haben sich in der Praxis vier verschiedene Modi durchgesetzt. Diese werden durch die Parameter Clock Polarität (CPOL) und Clock Phase (CPHA) festgelegt. Bei CPOL=0 ist der Clock Idle Low, bei CPOL=1 ist der Clock Idle High. CPHA gibt nun an, bei der wievielten Flanke die Daten übernommen werden sollen. Bei CPHA=0 werden sie bei der ersten Flanke übernommen, nachdem SS auf Low gezogen wurde,

bei CPHA=1 bei der zweiten. Somit werden die Daten bei CPOL=0 und CPHA=0 mit der ersten Flanke übernommen, die nur eine High-Flanke sein kann. Bei CPHA=1 wäre es die zweite, also eine Low-Flanke. Bei CPOL=1 ist das ganze folglich genau andersherum, bei CPHA=0 Low-Flanke und bei CPHA=1 High-Flanke.

Zu beachten ist noch, dass der Slave bei CPHA=0 seine Daten schon beim Runterziehen von SS an MISO anlegt, damit der Master sie beim ersten Flankenwechsel übernehmen kann. Bei CPHA=1 werden die Daten vom Slave erst beim ersten Flankenwechsel an MISO gelegt, damit sie beim zweiten Flankenwechsel vom Master übernommen werden können. Der Master hingegen legt seine Daten immer zum gleichen Zeitpunkt an, meist kurz nach der Low-Flanke von SCK.

Mit jeder Taktperiode wird ein Bit übertragen. Beim üblichen Bytetransfer sind also 8 Taktperioden für eine vollständige Übertragung nötig. Es können auch mehrere Bytes hintereinander übertragen werden, wobei nicht festgelegt ist, ob zwischen jedem Byte das SS-Signal kurz wieder auf High gezogen werden muss. Eine Übertragung ist beendet, wenn das Slave-Select-Signal endgültig auf High gesetzt wird.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Tabelle 1.1: Übersicht der verschiedenen Modi

1.3 use-Cases

Da eine genaue Wortbreite in der Spezifikation nicht vorgegeben ist, wird im folgendem von einem 8Bit breitem Datenwort ausgegangen. Mögliche use-Cases sind:

- Ein Datenwort empfangen/senden
- n Datenworte empfangen/senden
- fehlerhaftes Datenwort empfangen/senden
- n fehlerhafte Datenworte empfangen/senden

Diese sollten in allen vier Modi abgedeckt werden (siehe Tabelle 1.1).

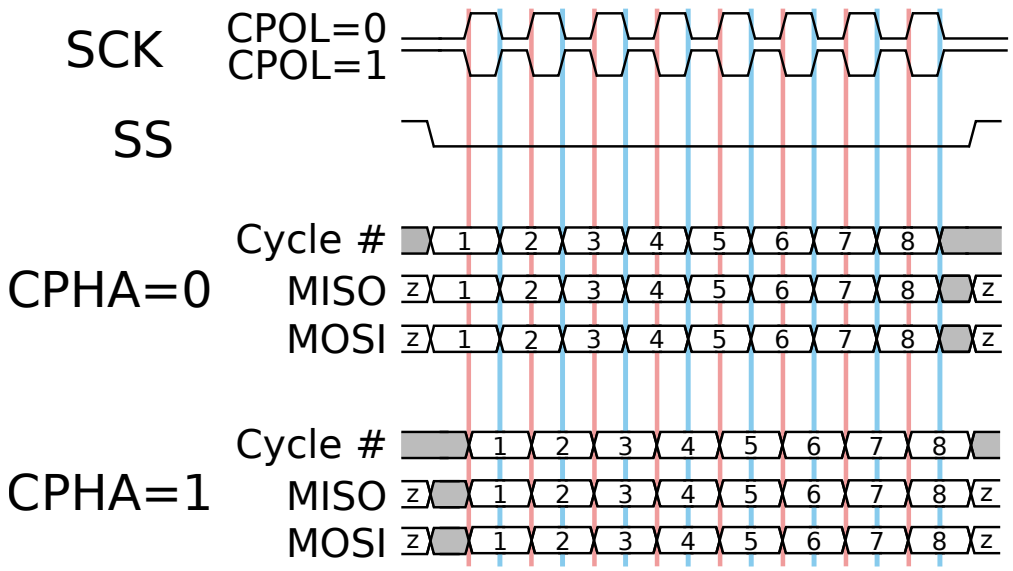

Abbildung 1.2: Timing Diagramm zum SPI Protokoll

1.4 Lösungsansatz

Nach dem Entwurf auf algorithmischer Ebene, wird das Modell mit der entsprechenden Testbench verifiziert. Ist dieder Test erfolgreich, so ist ein Signalübergangsgraph zu entwerfen und aus Ihm mit einer geeigneten Entwurfsmethode ein oder mehrere Automaten (Mealy/Moore/FSMD) abzuleiten. Diese Realisierung ist wieder mit einer Testbench zu verifizieren (cycle accurate) und auf dem DE2-Evaluationsboard zu implementieren.

Abbildungsverzeichnis

Quelltextverzeichnis