

Milestone 1: SPI-Transmitter

Lernziele

- Wiederholung Vivado-Projekte, VHDL und Synthese
- Inferenz und Initialisierung von On-Chip-Speichern
- RTL-Design am Beispiel einer synchronen Schnittstelle

Aufgabenstellung

Es soll ein SPI-Transmitter in VHDL entworfen werden. Der SPI-Transmitter soll ähnlich zu dem in der Vorlesung vorgestellten UART-Transmitter realisiert werden: Mit Hilfe eines Vorteilers kann die Datenrate (SCK-Frequenz) gewählt werden. Die ausgegebenen Daten werden einem ROM-Speicher entnommen.

Sie können gerne im Voraus die Planung der Aufgabe (insbesondere des Moduls „spi_transmitter“) mit Ihrem Versuchsbetreuer besprechen.

Durchführung

1. Erstellen Sie ein neues Vivado-Projekt und legen Sie hierbei die folgenden Dateien an ("create file"):
 spi2display.vhd (Toplevel des Moduls)
 spi_transmitter.vhd (Vorteiler und Endlicher Automat für die SPI-Übertragung)
2. Fügen Sie die vorgegebene Datei spi2display_rom.vhd mit „add or create **design** sources“ dem Projekt hinzu. Achten Sie darauf, dass der Haken bei „Copy Sources into project“ gesetzt ist.
 Dies gilt auch für alle weiteren vorgegebenen Dateien!
3. Fügen Sie die vorgegebene Testbench spi2display_tb.vhd mit „add or create **simulation** sources“ dem Projekt hinzu.
4. Fügen Sie die vorgegebene Constraints-Datei spi2display.xdc mit „add or create **constraints**“ dem Projekt hinzu.
5. Fügen Sie in den VHDL-Dateien spi2display.vhd und spi_transmitter.vhd die vorgegebenen Entities (s. Dateien im Versuchsverzeichnis) ein und schreiben Sie die VHDL-Implementierung der zugehörigen Architectures. Folgende Hinweise sollen Ihnen die Arbeit erleichtern:

spi2display

Dieses Modul instanziiert die beiden anderen Module. In diesem Modul wird keine Logik benötigt.

Achten Sie darauf, dass das Modul spi_transmitter einen Taktteilerwert (CLKDIV) erwartet. Diesen müssen Sie mit Hilfe der Generics CLOCK_FREQ und SCK_FREQ festlegen.

spi_transmitter

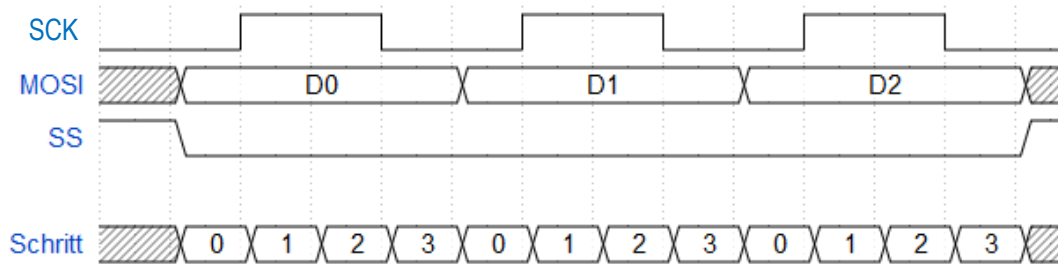
Für die Realisierung dieses Modul können Sie sich an den Konzepten des UART-Transmitters orientieren.

Bedenken Sie, dass Sie für jeden SCK-Taktzyklus zwei Schritte (SCK=0 und SCK=1) benötigen. Wenn Sie vermeiden möchten, dass sich die Signale MOSI und SCK gleichzeitig ändern, können Sie zum Beispiel eine Lösung mit 4 Schritten je SCK-Taktzyklus wählen. Ein entsprechendes Timing-Diagramm finden Sie unten.

Bedenken Sie, dass das Select-Signal SS vor der Übertragung des ersten Bits aktiviert (low) und erst nach der Übertragung des letzten Bits deaktiviert werden sollte.

Wichtig: Anders als beim UART wird bei SPI das höchstwertige Bit (MSB, Bit 7) zuerst übertragen.

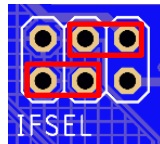
Anmerkung: Eine mögliche Lösung ist die Nutzung von 4 Schritten je SPI-Taktzyklus (s. Bild unten). Dann werden insgesamt 35 Schritte für die Übertragung eines Bytes benötigt. Auf diese Weise wird sichergestellt, dass die Datenleitung MOSI stabil ist, wenn sich der Pegel des Taktsignals SCK ändert (vgl. nachfolgendes Timingdiagramm). Es sind auch Lösungen mit weniger Schritten möglich. Wichtig ist in jedem Fall, dass das MOSI-Signal während der steigenden Flanke des SCK-Signals stabil ist.



Beispiel einer SPI-Übertragung. Dieses Beispiel verwendet 3 bit pro SPI-Frame. Das Display benötigt 8 bit pro Frame.

6. Simulieren Sie Ihr Design und überprüfen Sie es auf eventuelle Fehler.
7. Synthetisieren Sie das Projekt und testen Sie es auf dem Zybo-Board. Schließen Sie für den Test das Display an die Buchsenleiste **JE** (vorne links unter den Schiebeschaltern) an.

Achten Sie darauf, dass beim Display das SPI-Interface über die Jumper auf der Rückseite ausgewählt ist. Das nachfolgende Bild zeigt die korrekte Positionierung der Jumper (weitere Auswahlmöglichkeiten finden Sie in der Datei *DisplayInterfaceSelection.pdf*).



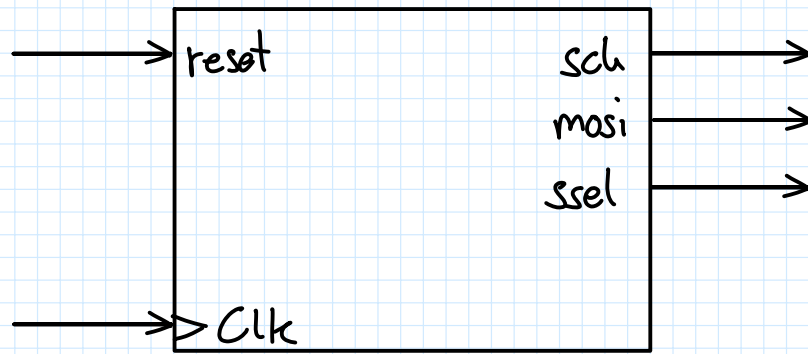
Abgabe

Um den Speicherbedarf von Vivado-IP-Projekten zu reduzieren, können Sie alle Verzeichnisse und Dateien mit Ausnahme der folgenden löschen: Verzeichnis <projectname>.srcs und Projektdatei <projectname>.xpr

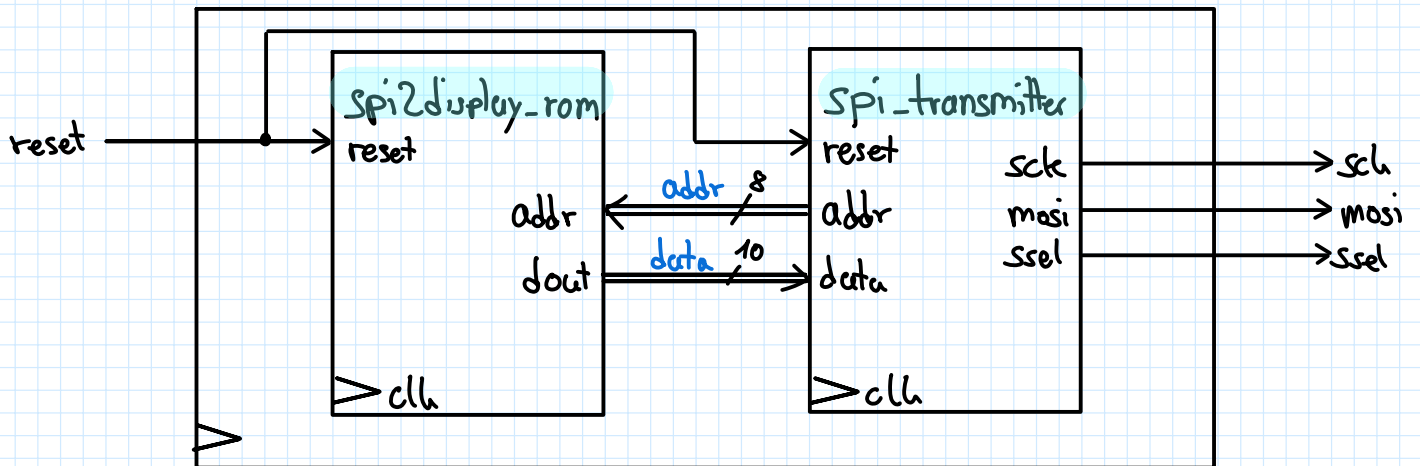
Für die Abgabe dieses Milestones archivieren Sie bitte beide erzeugte Projekte (nur die o.g. Dateien) in einer ZIP-Datei.

Fügen Sie auch einen Screenshot der Simulation und des FPGA-Ressourcenverbrauchs hinzu.

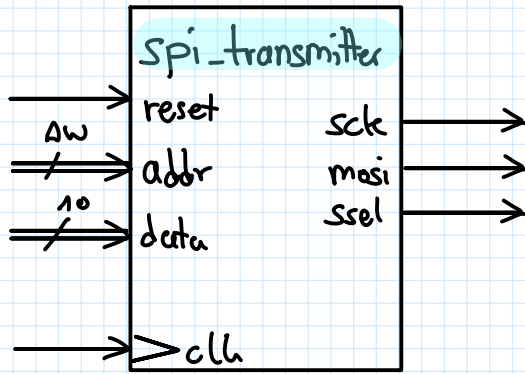
spi2display - Entity



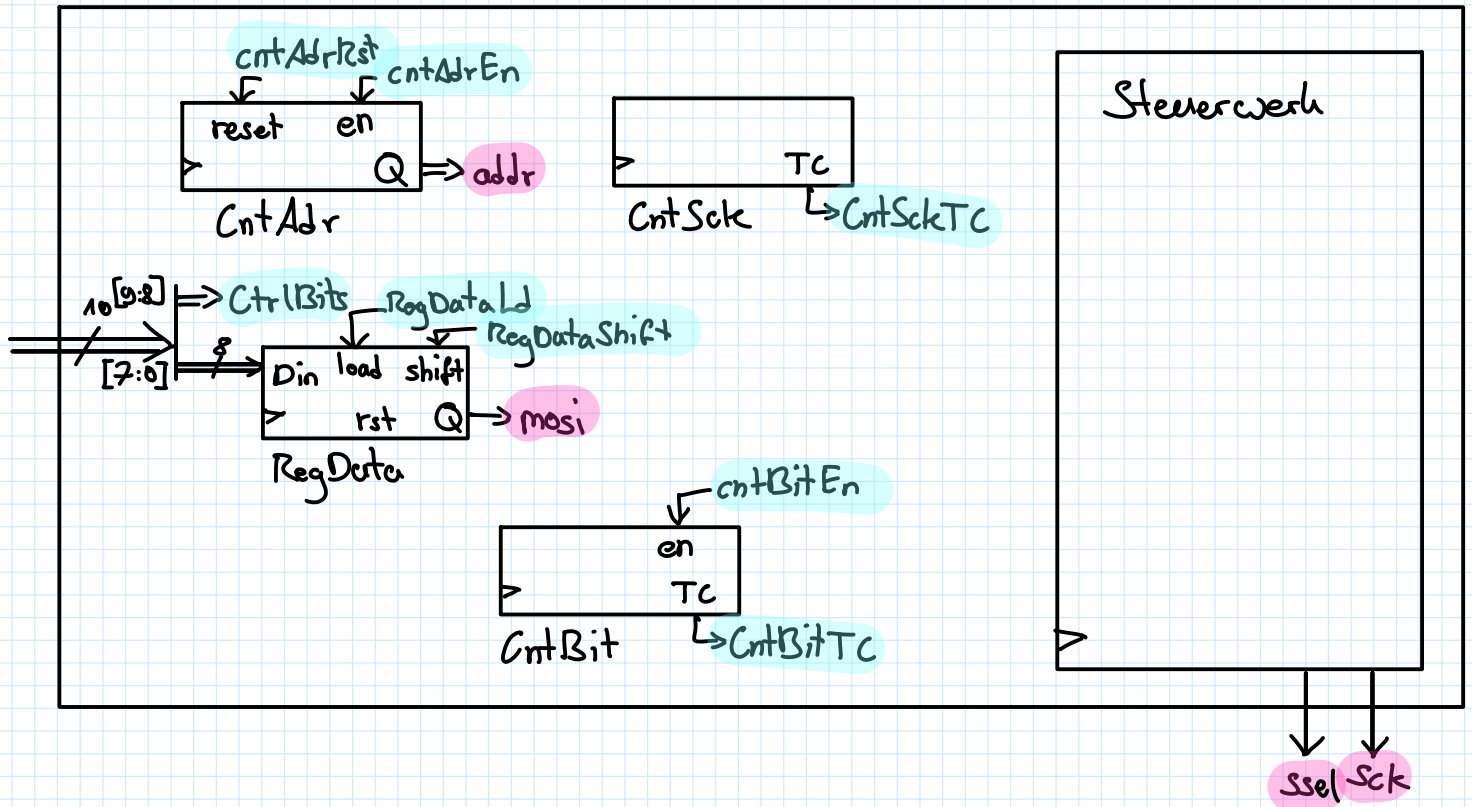
spi2display - Architecture

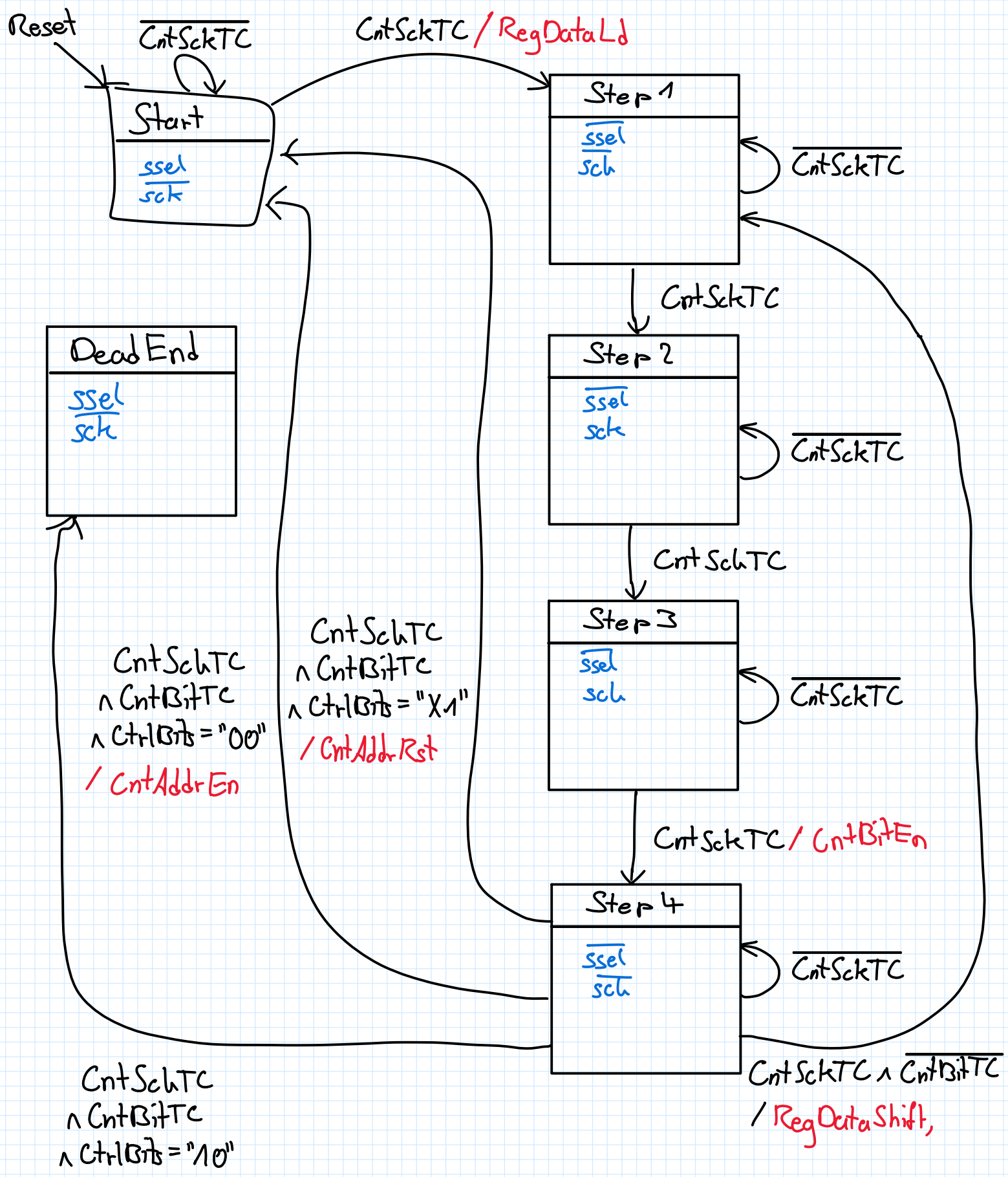


Spi-transmitter - Entity

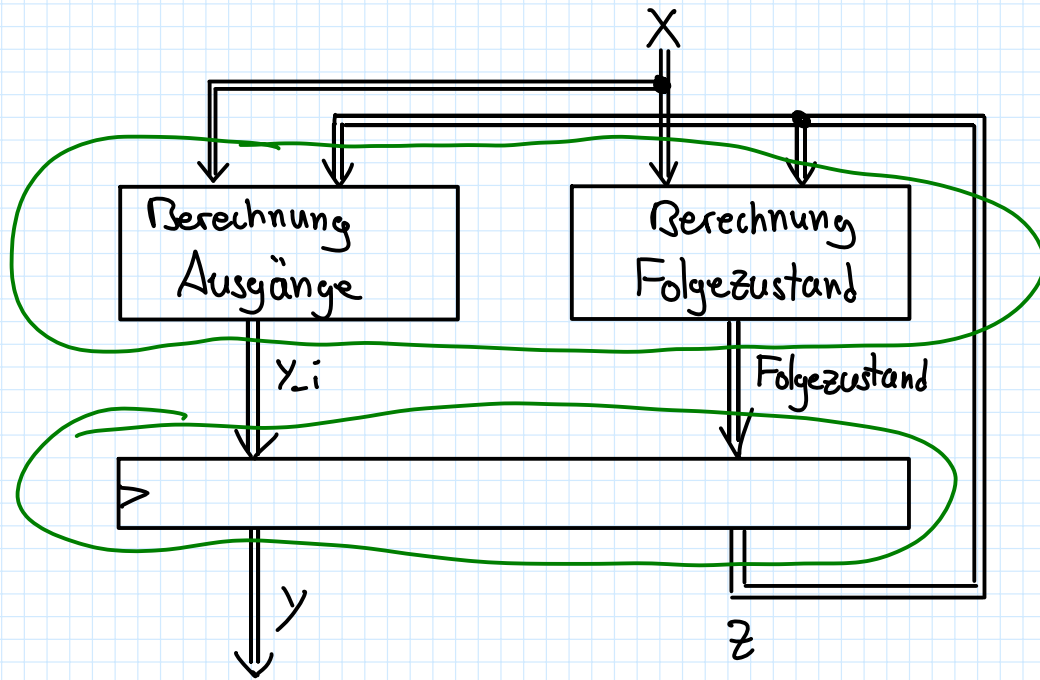


spi-transmitter



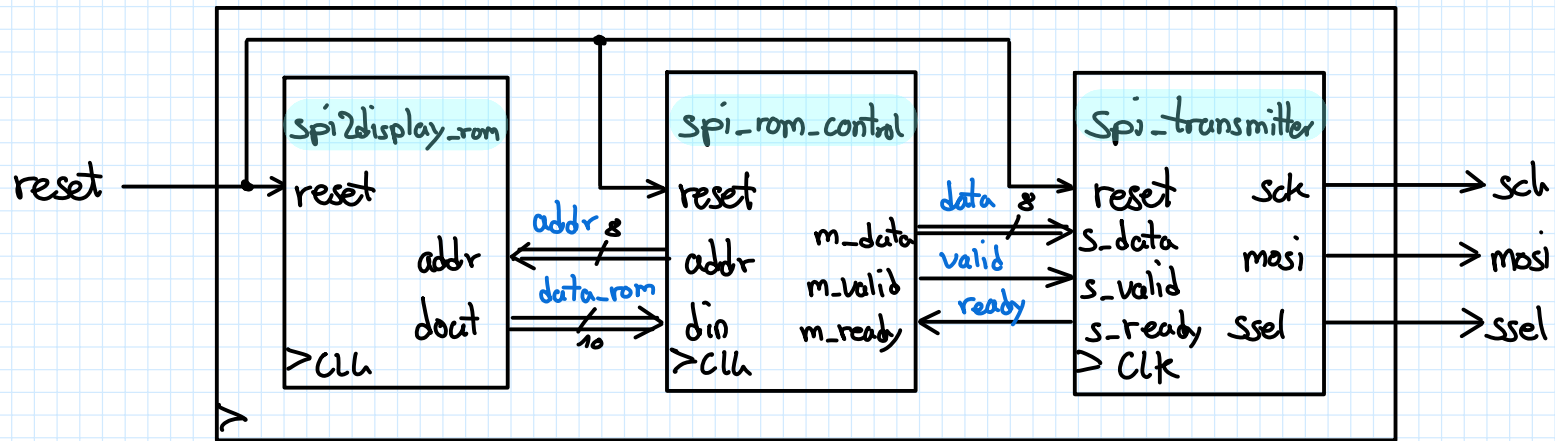


Endlicher Automat

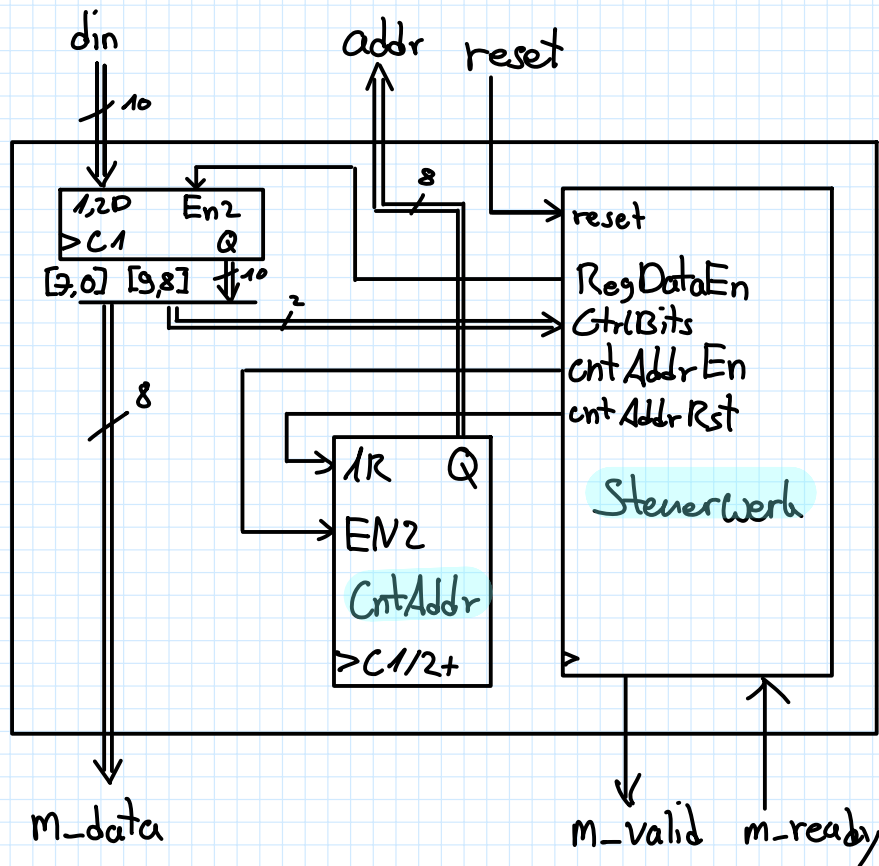


Überarbeitetes Design - mit Control-Modul

Spi2display - Architecture



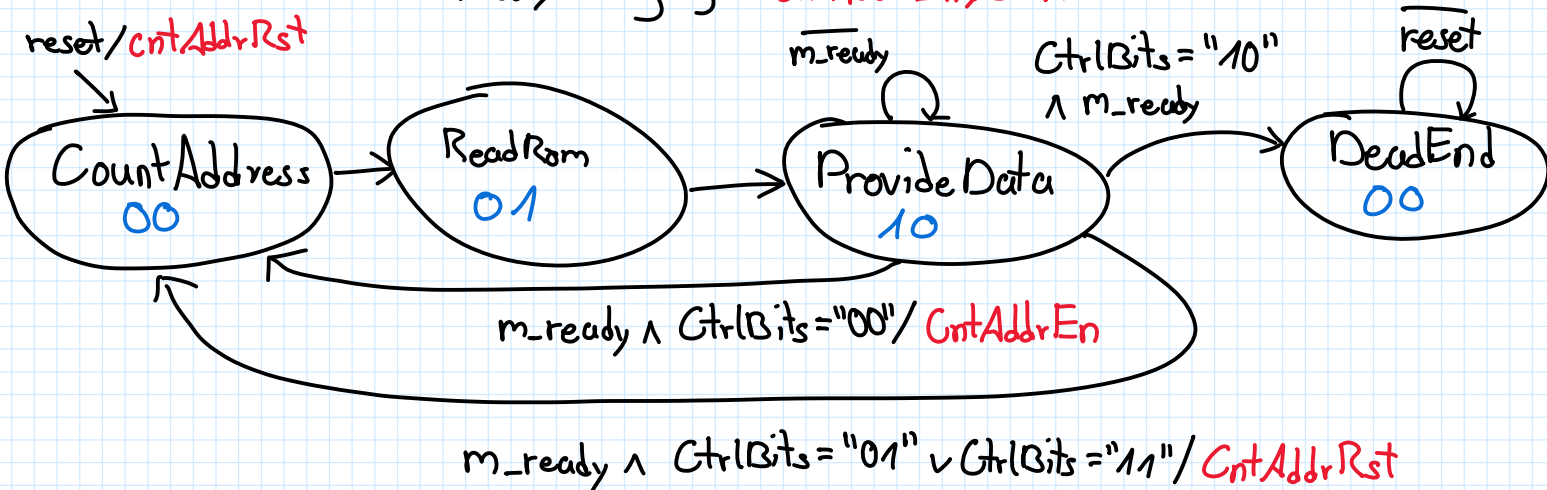
Spi_rom_control - Architecture



Zustandsdiagramm

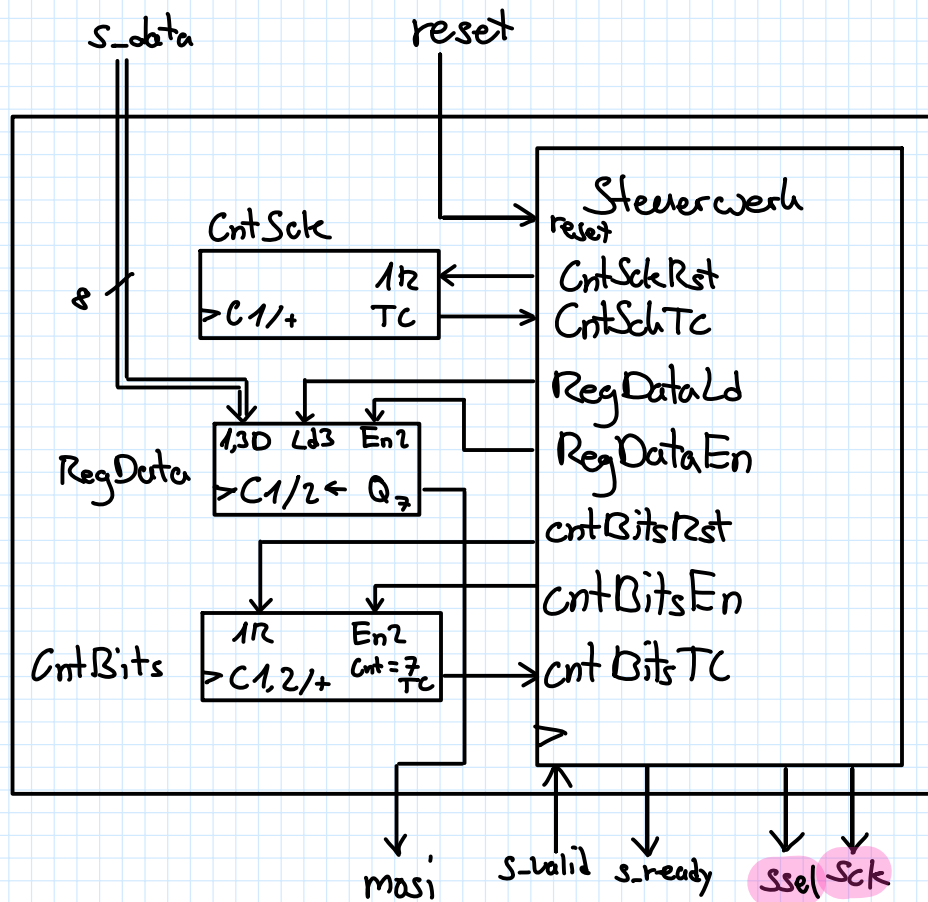
Moore-Ausgänge: m_valid , $RegDataEn$

Mealy-Ausgänge: $cntAddrEn$, $cntAddrRst$



spi_transmitter

s_data



Moore-Ausgänge: ssel, scl, s_ready

Mealy-Ausgänge: RegDataLd, RegDataEn, CntBitsEn, CntBitsRst, CntSchRst

