Mobiles Hardware-Praktikum 2018

B. Völker, J. Burchard, T. Schubert, B. Becker
Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau

Übungsblatt 7 (26 Punkte)

Abgabe: 27.06.2018 bis 17:00 Uhr

Ziel:

Der Beschleunigungssensor auf dem DE0-Nano wird mittels SPI ausgelesen und die aktuelle Beschleunigung in alle 3 Achsen auf dem LCD angezeigt.

Prolog:

Auf dem DE0-Nano befindet sich ein 3-Achsen-Beschleunigungssensor (ADXL345). Dieser misst die aktuelle Beschleunigung in die X, Y und Z Richtung. Der Sensor ist über ein 3-wire SPI (Serial Peripheral Interface) mit dem FPGA verbunden. Über ein einziges SPI können viele Sensoren kommunizieren. Der aktive Sensor wird dabei über ein Chip Select Signal ausgewählt.
Zur Kommunikation werden drei Leitungen[1] benötigt:

Name	Funktion
CS_N	Chip Select (*active low*)
SCLK	Clock
SDIO	Bidirektionale Datenleitung

Auf diesem Blatt wird ein SPI-Master entwickelt. Dieser baut eine Verbindung zu einem SPI-Slave auf und fragt Daten von diesem ab. *CS_N*, *SCLK* und *SDIO* werden vom Master gesteuert. Nur, wenn der Slave eine Antwort überträgt, übernimmt er die Kontrolle über *SDIO*.

Laden Sie das Datenblatt für den ADXL345 Sensor herunter[2] und betrachten Sie das Timing-Diagramm für 3-wire SPI auf Seite 16.
Im folgenden wird diese Kommunikation schrittweise entwickelt. Importieren Sie das Quartus-Projekt "gSensorReader". Es enthält unter anderem die Datei *g_reader.vhd*, in welcher der SPI-Master implementiert werden soll. Wählen Sie diese Datei auch als Top-Level Entity für die Simulation.

Aufgabe 1 (*SCLK*) (3 Punkte):

Erstellen Sie zunächst einen neuen Prozess, um das Signal *SCLK* zu erzeugen. Führen Sie ein neues Signal *SCLK_int* ein, dass Sie (außerhalb des Prozesses) mit *SCLK* verbinden. *SCLK* soll eine Frequenz von 1 MHz haben. Simulieren Sie Ihr Design. Stellen Sie sicher, dass *clk_50* in der Simulation eine Frequenz von 50 MHz hat. Zum Simulieren der Clock können Sie einen eigenen Endlos-Prozess (ohne “wait;” und ohne Sensitivitätsliste) wie bei Blatt 6 verwenden.

Aufgabe 2 (State Machine) (6 Punkte):

Bevor Sie den Beschleunigungssensor auslesen können, muss dieser zunächst initialisiert werden. Nach der Initialisierung kann die Beschleunigung in X, Y und Z Richtung einzeln ausgelesen werden. Um dieses Verhalten zu verwirklichen, eignet sich ein Zustandsautomat.

[1] SPI kann sowohl mit 3 als auch mit 4 Leitungen benutzt werden, auf dem DE0-Nano sind aber nur 3 Leitungen verbunden.
[2] http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf

Führen Sie zunächst die Signale *transmission_count* (5 Bit) und *time_count* (16 Bit) ein. Mit *transmission_count* wird später die Anzahl der übertragenen Bit gezählt, mit *time_count* die Länge der Pause zwischen verschiedenen Abfragen.
Implementieren Sie dann einen Automaten mit den folgenden Zuständen: *init0, init1, init2, init3, init4, readX, readY, readZ, pause*. Zur Vereinfachung können Sie dazu in VHDL einen neuen Typen definieren:

```
-- define a new type (with the three possible values a, b, c)
type state_type is (a, b, c);
-- define a signal of type state_type with initial value a
signal state : state_type := a;
```

pause ist ein Wartezustand zwischen den Anfragen an den Beschleunigungssensor. Definieren Sie außerdem ein weiteres Signal *next_state*. Dieses speichert jeweils den nächsten Zustand nach *pause*. Initialisieren Sie *state* mit *pause* und *next_state* mit *init0*.
Implementieren Sie außerdem die folgenden Zustandsübergänge:

Ausgangszustand	Bedingung	Zielzustand	Auswirkung
init0	transmission_count = 16	pause	next_state := init1
init1	transmission_count = 16	pause	next_state := init2
init2	transmission_count = 16	pause	next_state := init3
init3	transmission_count = 16	pause	next_state := init4
init4	transmission_count = 16	pause	next_state := readX
pause	next_state = readX und time_count >= 50000 und SCLK_int = '1'	readX	
pause	next_state != readX time_count >= 500 und SCLK_int = '1'	next_state	
readX	transmission_count = 25	pause	next_state := readY
readY	transmission_count = 25	pause	next_state := readZ
readZ	transmission_count = 25	pause	next_state := readX

Nach einem Reset (*reset_n* = '0') soll der Automat immer im Zustand *pause* mit *next_state* = *init0* starten. Beachten Sie, dass alle Zustandsübergänge synchron mit *clk_50* erfolgen (also z.B. nur bei steigenden Flanken von *clk_50*). Durch die Bedingungen beim Verlassen des *pause*-Zustandes ist außerdem sichergestellt, dass Zustandsänderungen nur eintreten während *SCLK* = '1'.
Das Signal *time_count* soll im *pause* Zustand in jeder *clk_50* Periode erhöht werden und jeweils beim Betreten von *pause* auf 0 gesetzt werden. Das Signal *transmission_count* wird in den kommenden Aufgaben gesetzt.

Aufgabe 3 (*CS_N* und *SCLK*) (2 Punkte)

Das chip select Signal *CS_N* soll immer dann aktiv sein, wenn mit dem Beschleunigungssensor kommuniziert wird (also in allen Zuständen außer *pause*).
Des weiteren soll der Ausgang *SCLK* eine konstante '1' liefern, wenn nicht mit dem Sensor kommuniziert wird (also im Zustand *pause*).
Implementieren Sie diese Funktionalität am Besten außerhalb von Prozessen durch einfache *when*-Statements.

Aufgabe 4 (Daten schreiben) (6 Punkte)

Die Kommunikation mit dem Beschleunigungssensor erfolgt über die *SDIO* Leitung. Jede Übertragung wird vom Master initialisiert. *SDIO* wird bei fallenden Flanken von *SCLK* geschrieben (und in der nächsten Aufgabe bei steigenden Flanken gelesen).
Zur Initialisierung (*init0, ..., init4*) müssen jeweils 16 Bit übertragen werden (8 Bit Adresse + 8 Bit Daten, siehe Seite 23ff im Sensor-Datenblatt):

Zustand	Übertragung (von links nach rechts)
init0	00110001 01001000
init1	00100100 00000010
init2	00101100 00001001
init3	00101110 00010000
init4	00101101 00001000

Zum Auslesen der Sensoren werden erst 8 Bit gesendet (Adresse) und dann 16 Bit gelesen.

Zustand	Übertragung (von links nach rechts)
readX	11110010
readY	11110100
readZ	11110110

Verwenden Sie das Signal *transmission_count*, um das zu übertragende Bit zu bestimmen. Während Übertragungen soll *transmission_count* bei jeder steigenden Flanke an *SCLK* inkrementiert werden. Im *pause* Zustand wird *transmission_count* bei steigenden *SCLK* Flanken auf 0 gesetzt. Implementieren Sie diese Funktionalität am Besten in einem weiteren Prozess.
Wenn *SDIO* keinen definierten Wert hat (z.B. im *pause*-Zustand oder nach den ersten 8 Bit in den *read*-Zuständen), schreiben Sie ein 'Z'. Damit wird die Leitung auf hohe Impedanz gelegt und kann zum Beispiel vom SPI-Slave geschrieben werden.

Simulieren Sie ihr Design. Ihre Simulation sollte Abbildung 1 und Abbildung 2 gleichen.

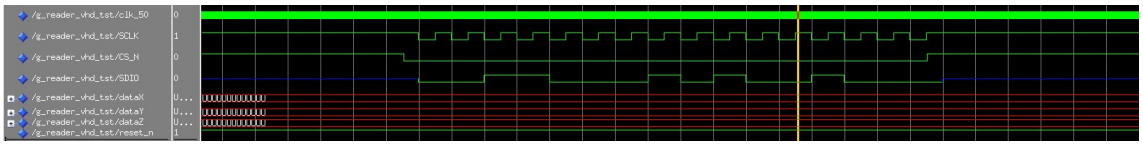

Abbildung 1: Simulation während der Übertragung von *init0*. Beachten Sie: Zunächst wird CS_N aktiv. Dann folgt die erste fallende Flanke an $SCLK$. Der Wert von $SDIO$ wird jeweils nur bei fallenden Flanken geändert. Der SPI-Slave liest den Wert an *SDIO* bei jeder steigenden Flanke von *SCLK*.

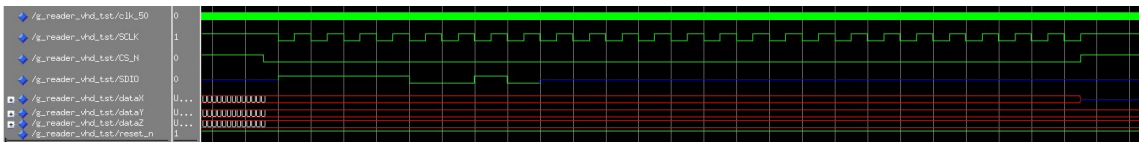

Abbildung 2: Simulation während der ersten Übertragung von *readX*. Nach der Übertragung von 8 Bit setzt der Master $SDIO$ auf 'Z' und erlaubt damit dem Slave die Kontrolle über die Leitung. Da es in der Simulation keinen Slave gibt verbleibt $SDIO$ auf 'Z'.

Aufgabe 5 (Verständnis) (5 Punkte)

Verwenden Sie das Datenblatt um die genaue Bedeutung (welche Einstellung wird mit jedem Bit in welchem Register gesetzt) der Befehle init0 - init4 herauszufinden. Fügen Sie in Ihrem Code einen Kommentar mit einer Beschreibung für jeden Befehl hinzu.

Aufgabe 6 (Daten lesen) (4 Punkt)

Abschließend müssen noch die Antworten des Beschleunigungssensors auf die Anfragen *readX*, *readY* und *readZ* gelesen werden. Beachten Sie: der Sensor antwortet mit zwei Byte $\langle b_h, b_l \rangle$ auf Anfragen nach der aktuellen Beschleunigung. Zunächst wird b_l übertragen (*transmission_count* 8 bis 15), dann b_h (*transmission_count* 16 bis 23). Die einzelnen Byte werden jeweils von links nach rechts ("most significant bit first") übertragen. Die höchsten 3 Bit in b_h haben immer den Wert

'0' und können daher ignoriert werden. Eine Übertragung könnte zum Beispiel folgendermaßen aussehen:

$$\overbrace{10010010}^{b_l}\ \overbrace{\underbrace{000}_{immer\ '0'}\ 11100}^{b_h}$$

Speichern Sie den relevanten Teil der Antwort in einem 13 Bit Vektor *read_temp* zwischen (in obigem Beispiel sollte also der Wert "11100 10010010"gespeichert werden).
Lesen Sie dazu bei steigenden Flanken von *SCLK* den aktuellen Wert an *SDIO*. Beginnen Sie mit dem Lesen des ersten Bits bei *transmission_count* = 8. Erreicht *transmission_count* den Wert 24, schreiben Sie den Wert von *read_temp* an den entsprechenden Ausgang (also z.B. *dataX* wenn der aktuelle Zustand *readX* ist).

Aufgabe 7 (Testen)

Wählen Sie die Datei *gSensorToLCD.bdf* als Toplevel-Entity aus und kompilieren Sie das Projekt neu. Verbinden Sie außerdem das LCD wie auf Übungsblatt 6 mit dem FPGA. Übertragen Sie das Projekt auf das FPGA. Sie sollten jetzt die aktuelle Beschleunigung in allen drei Achsen auf dem LCD angezeigt bekommen.

Hinweis: Der Beschleunigungssensor misst auch die Erdbeschleunigung. Je nachdem wie Sie den Roboter halten sollten Sie unterschiedliche Beschleunigungen in Richtung der drei Achsen messen.

Abgabe:

Archivieren Sie das Quartus II-Projekt und laden Sie es im Übungsportal hoch. Überprüfen Sie die Archiv-Datei auf Vollständigkeit.