Hochschule Osnabrück
Fachbereich Elektrotechnik und Informatik
Labor für Digital- und Mikroprozessortechnik
Prof. Dr.-Ing. B. Lang, Dipl.-Inf. (FH) R. Höckmann

Praktikum Rechnerorganisation

Versuch 2: Serielle Kommunikation

Lernziele

- Realisierung digitaler Systeme in VHDL
- Vertiefung: Testbench und Simulation

Werkzeuge

- BASYS3-Board
- PmodUSBUART-Adapter
- ModelSim (Intel FPGA Starter Edition)
- Vivado (Version 2019.2)
- HTerm

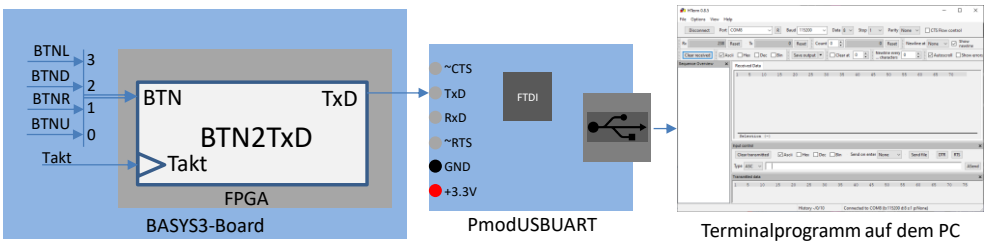
Aufgabenstellung

Die Kommunikation zwischen Rechnern und externen Geräten kann mittels asynchroner serieller Schnittstelle (UART) erfolgen. Dazu wurde in der Vorlesung ein serieller Empfänger vorgestellt.

Im vorliegenden Praktikumsversuch wird dazu das Gegenstück, ein serieller Sender, realisiert.

Die erstellte Komponente (UART_Sender.vhd) wird in eine Schaltung eingebaut (BTN2TxD.vhd), welche Änderungen der Taster detektiert und diese seriell als ASCII-Zeichen codiert an einen PC überträgt. Diese ASCII-Zeichen können dann beispielsweise mit einem Terminalprogramm auf dem PC dargestellt werden.

Folgende Übersicht zeigt den Aufbau:



Die Komponente "BTN2TxD" im FPGA überwacht die vier Taster "BTN(3 downto 0)". Wenn sich dort eine Änderung ergibt, wird der Binäre 4-Bit-Tasterwert in einem ASCII-Zeichen kodiert und über die serielle Leitung "TxD" ausgegeben.

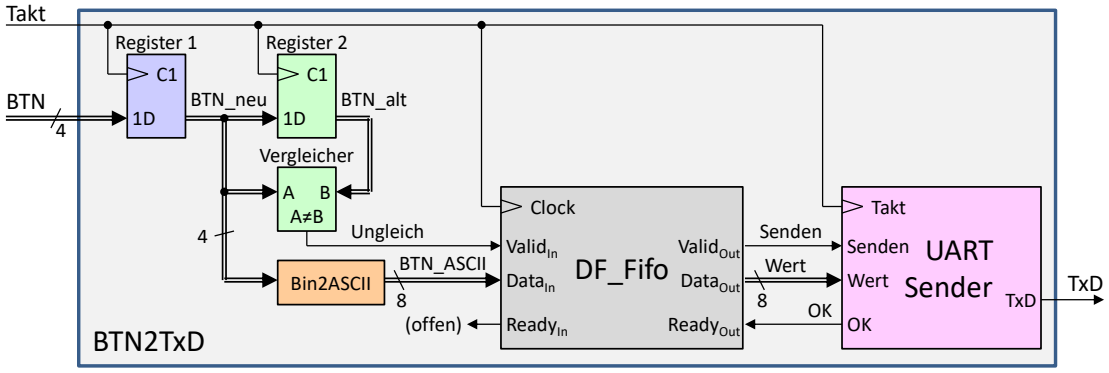
Über einen USB-UART-Adapter empfängt der PC die seriellen Daten, die dann in einem Terminal-Programm angezeigt werden können.

Vorbereitung (Vor dem Praktikumstermin)

- Bearbeiten Sie die in den einzelnen Aufgaben angegebenen Vorbereitungsaufgaben

Aufgabe 1: System mit Simulationsmodell simulieren

Das System ist wie folgt aufgebaut:



Die Tasterwerte "BTN" werden mit dem Register 1 auf den Systemtakt "Takt" synchronisiert.

Die synchronisierten Tasterwerte "BTN_neu" werden im Register 2 gespeichert, damit stehen an seinem Ausgang "BTN_alt" die synchronisierten Tasterwerte des vorhergehenden Taktzyklus zur Verfügung. Sind die Tasterwerte "BTN_neu" und "BTN_alt" ungleich, wurde zwischen letztem und aktuellem Taktzyklus zumindest ein Taster verändert.

Aus den synchronisierten Tasterwerten wird mit einem kombinatorischen Prozess "Bin2ASCII" ein zugehöriger 8-Bit ASCII-Code "BTN_ASCII" erzeugt. Folgende Tabelle zeigt die Zuordnung der Tasterwerte zu ASCII-Codes:

Tasterwerte	ASCII-Zeichen	ASCI-Code (Hex)
0000	'0'	0x30
0001	'1'	0x31
...	...	...
1001	'9'	0x39
1010	'A'	0x41
1010	'B'	0x42
...	...	...
1111	'F'	0x46

Bei einer Änderung, (Ungleich='1'), wird der ASCII-Code des neuen Tasterwertes in einen FIFO-Speicher (DF_Fifo) eingeschrieben. Dieser kann 2^{11}=2048 Zeichen zwischenspeichern.

Die Zeichen werden vom Ausgang des FIFO-Speichers dann an den UART-Sender weitergereicht und damit aus dem Fifo gelöscht, wenn der Sender mit "OK='1'" signalisiert, dass er ein Wort liest. Der Sender serialisiert das Wort, ergänzt es um ein Start- und Stoppbit und gibt die seriellen Daten über seinen Ausgang "TxD" aus.

Die Komponente "BTN2TxD" wird Ihnen unvollständig bereitgestellt (BTN2TxD.vhd). In der Datei ist bereits die **entity** erstellt und in der **architecture** sind das Fifo und der UART-Sender instanziiert. Sie müssen die beiden Register, den Vergleicher und den Codewandler "Bin2ASCII" ergänzen.

Nach Fertigstellung wird die Komponente mit ModelSim simuliert. Dazu wird Ihnen eine geeignete Testbench (BTN2TxD_tb.vhd), die synthetisierbare Fifo-Komponente (DF_Fifo.vhd) und ein nicht synthetisierbares Modell der "UART_Sender"-Komponente (UART_Sender_nur_Simulation.vhd) zur Verfügung gestellt.

Vorbereitung (Vor dem Praktikumstermin)

- Ergänzen Sie die fehlenden Teile der "BTN2TxD"-Komponente in "BTN2TxD.vhd".
- Realisieren Sie alle fehlenden Teile mittels Prozessen und/oder bedingten Signalzuweisungen. Also **keine** zusätzlichen Komponenten erstellen und instanziieren.
- Hinweis: Der Codewandler kann einfach durch einen auf "BTN_neu" sensitiven Prozess mit case-Anweisungen realisiert werden.

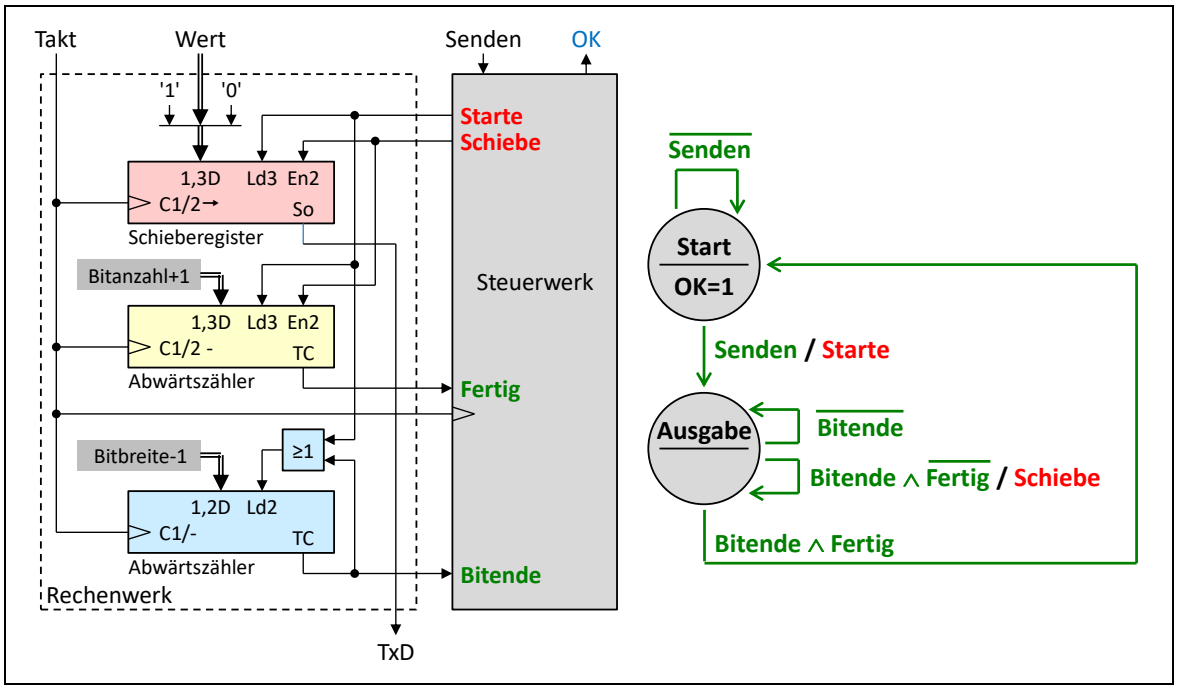
Durchführung

1. Legen Sie die vorgegebenen und die von Ihnen vervollständigten Dateien im Arbeitsverzeichnis ab.
2. Simulieren Sie das Design in ModelSim mit der vorgegebenen Testbench (BTN2TxD_tb.vhd). Nutzen Sie dabei das vorgegebene Simulationsskript (test_BTN2TxD_nur_Simulation.do). Überprüfen Sie die Ausgaben in der Simulations-Konsole und die Ausgänge im Wave-Fenster. Korrigieren Sie möglicherweise noch vorhandene Fehler.

Hinweis: Eine Synthese ist noch nicht möglich, dazu muss erst das nicht synthetisierbare Modell der "UART_Sender"-Komponente durch ein synthetisierbares Modell ersetzt werden. Dies erfolgt nun in Aufgabe 2.

Aufgabe 2: Synthetisierbare Komponente erstellen, verifizieren und in Hardware testen

Die synthetisierbare Komponente "UART_Sender" soll, entsprechend wie der in der Vorlesung vorgestellte UART-Empfänger, aus Rechen- und Steuerwerk aufgebaut werden:



Links ist das Rechenwerk gezeigt:

- Das Rechtsschieberegister nimmt das zu sendende Datenwort zusammen mit vorangestelltem Startbit ('0') und nachfolgendem Stoppbit ('1') auf. Beim Schieben werden die Bits beginnend mit dem Startbit serialisiert und über "TxD" ausgegeben. Dabei werden an höchster Stelle '1'-Bits eingefügt.
- Der in der Mitte gezeigte Abwärtszähler zählt die noch zu sendenden Bits im Schieberegister. Er muss die Bits des "Wert"-Signals (Bitanzahl) plus 1 Startbit plus 1 Stoppbit zählen. Da er bis 0 zählt, wird er mit (Bitanzahl+2)-1 = Bitanzahl+1 initialisiert.
- Der untere Abwärtszähler zählt die Breite jedes einzelnen Bits. Der Wert Bitbreite gibt dabei die zeitliche Breite jedes Bits auf dem "TxD"-Ausgang in Anzahl von Systemtaktzyklen an. Der Zähler wird geladen wenn das "Starte"-Signal anliegt oder wenn er sein Zählende erreicht hat. Da der Zähler ebenfalls abwärts bis 0 zählt, wird er mit dem Wert Bitbreite-1 geladen.
- Rechts ist das Zustandsdiagramm des Steuerwerks gezeigt:
- Der Automat wartet im Start-Zustand, bis am Eingang das "Senden"-Signal aktiviert wird. Dabei muss das zu sendende Datenwort am Eingang "Wert" anliegen. Ist "Senden" aktiv, wird beim Übergang zum Ausgabe-Zustand das Mealy-Signal "Starte" aktiviert. Dies bewirkt, dass mit der Taktflanke das "Wert"-Signal ergänzt um Start- und Stoppbit in das Schieberegister, Bitanzahl+1 in den Zähler in der Mitte und Bitbreite-1 in den unteren Zähler geladen werden. Damit beginnt sofort die Ausgabe des Startbits.
- Im Zustand Ausgabe wartet der Automat auf das Signal "Bitende". Wenn es aktiv ist, kann das nächste Bit ausgegeben werden. Sind noch Bits im Schieberegister vorhanden ("Fertig" nicht aktiv), wird das Mealy-Signal "Schieben" aktiviert und wieder in den Ausgabe-Zustand verzweigt. Ist die Übertragung abgeschlossen ("Fertig" aktiv) wird zurück in den Start-Zustand verzweigt und auf das nächte Datenwort gewartet.

Vorbereitung (Vor dem Praktikumstermin):

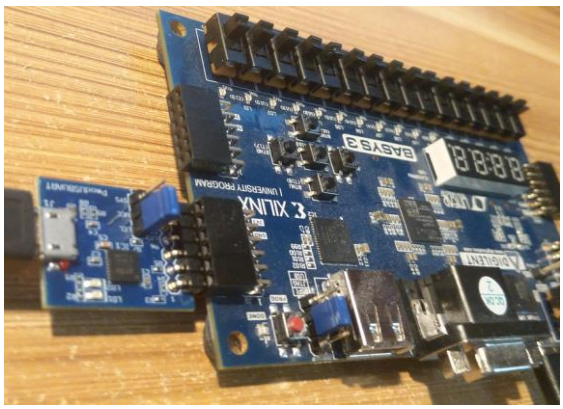
Vervollständigen Sie die VHDL-Beschreibung des UART-Senders (UART_Sender.vhd). Dort sind bereits die **entity** und der Rahmen für die **architecture** erstellt. Die Beschreibungen von Rechenwerk und Steuerwerk müssen Sie in den zugehörigen Blöcken ergänzen. Achten Sie auf eine generische Beschreibung.

Es ist selbstverständlich erlaubt, Code-Fragmente des in der Vorlesung vorgestellten UART-Empfängers zu verwenden und anzupassen.

Durchführung

Verifizieren der neu erstellten synthetisierbaren Sender-Komponente:

1. Simulieren Sie das Design in ModelSim mit der vorgegebenen Testbench (UART_Sender_tb.vhd). Nutzen Sie dabei das vorgegebene Simulationsskript (test_UART_Sender.do). Überprüfen Sie die Ausgaben in der Simulations-Konsole und die Ausgänge im Wave-Fenster. Korrigieren Sie möglicherweise noch vorhandene Fehler in der Datei "UART_Sender.vhd".
2. Simulieren Sie nun das Gesamtsystem mit dem Simulationsskript test_BTN2TxD.do. Überprüfen Sie die Ausgaben in der Simulations-Konsole und die Ausgänge im Wave-Fenster. Korrigieren Sie eventuelle Fehler.
3. Starten Sie Vivado und erstellen Sie ein neues Projekt BTN2TxDProject.
4. Fügen Sie dem Projekt die folgenden VHDL-Dateien hinzu:
   - BTN2TXD.vhd
   - UART_Sender.vhd
   - DF_Fifo.vhd
5. Fügen Sie die folgende Constraints-Datei hinzu:
   - BTN2TxD.xdc
6. Wählen Sie als Bauteil/Part wieder **xc7a35tcpg236-1** aus.
7. Lassen Sie Vivado eine Bitstream-Datei erzeugen (Generate Bitstream).
8. **Schalten Sie (falls eingeschaltet) das BASYS3-Board aus.** Stecken Sie das PmodUSBUART-Modul in die Erweiterungsbuchse JB des Basys3-Boards (untere Reihe, siehe Foto). Verbinden Sie das Basys3-Board und den PmodUSBUART-Adapter mit jeweils einem USB-Kabel mit dem PC. Schalten Sie das Basys3-Board ein.
9. Im Windows-Gerätemanager können Sie die Bezeichnung des COM-Ports herausfinden. Wenn dort mehrere „USB Serial Port“ angezeigt werden, müssen Sie ausprobieren, welcher zum PmodUSBUART-Modul gehört, indem Sie dessen USB-Verbindung kurzeitig trennen und wiederherstellen.
10. Starten Sie das Terminalprogramm HTerm (http://der-hammer.info/pages/terminal.html) auf dem PC. Verbinden Sie es mit dem virtuellen COM-Port des PmodUSBUART-Moduls und stellen Sie im Terminalprogramm die Kommunikationsparameter ein (115200 Baud, keine Parität, 1 Stoppbit).
11. Übertragen Sie die Bitstream-Datei auf das FPGA und testen Sie die Funktion durch Drücken verschiedener Taster auf dem Board.



Bitte legen Sie nach Abschluss des Versuchs folgende Dateien im OSCA-Dateibereich Ihrer Arbeitsgruppe in einem neuen Ordner „V2“ ab:

- BTN2TxD.vhd
- UART_Sender.vhd