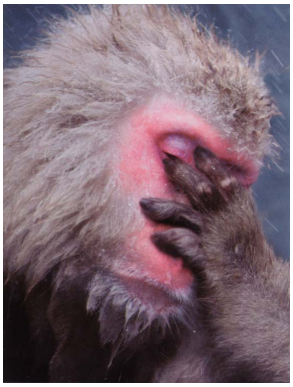
3. Übung: Codec: I^2S (24 P)



Why, oh why, didn't I keep it synchronous?

1. Aufgabe *ClkMaster*

In der letzten Übung war die Erzeugung der Taktsignale `Bclk`, `DAClrc` und `ADClrc` Sache des Codecs (*Master Mode*). Wie bereits angesprochen, würde dies in den späteren Ausbaustufen zu einem erhöhten Aufwand gegenüber der Takterzeugung im FPGA führen. Daher sollen diese Taktsignale nun von einer Unit namens `ClkMaster` erzeugt werden.

Extrem ***wichtig(!)*** ist natürlich, dass der Codec nun im *Slave Mode* betrieben wird (siehe Register *Digital Audio Interface Format*). Daher sollten Sie sich sicher sein, dass Ihre Beschreibung zur Konfiguration des Codec 100%ig funktioniert.

a☐ ***Test:*** Prüfen Sie zunächst, ob der Codec das Signal des Mikrofons im *Master Mode* korrekt über den ADC und den DAC führt. Stellen Sie nun den Parameter für den *Mode* auf *Slave Mode* um. Nun darf kein Signal mehr im Kopfhörer ankommen.

b☐$_6$ Beschreiben Sie als nächstes eine Unit `ClkMaster` (in der Group `StrobesClocks`), welche die Erzeugung der Takte `Mclk`, `Bclk` und `ADClrc` übernimmt (siehe Abb. 1). Das Signal `DAClrc` soll *nicht* von dieser Unit erzeugt werden.

Als Ausgangsbasis für die Unit `ClkMaster` könnten Sie beispielsweise die Unit `FreqDivider` heranziehen. Diese generiert bereits jetzt das Signal `Mclk`.

Die Ports und Generics der Unit `ClkMaster` sind:

```
1   generic (
2     gClkFrequency  : natural := 48E6;
3     gMclkFrequency : natural := 12E6;
4     gSampleRate    : natural := 44117);
5   port (
6     -- Sequential logic inside this unit
7     iClk         : in std_ulogic;
8     inResetAsync : in std_ulogic;
9     -- Main clock of the codec
10    oMclk : out std_ulogic;
11    -- Bit clock of the codec's digital audio interface
12    oBclk : out std_ulogic;
13    -- Left-right flag of the codec's digital audio interface for ADC
14    oADClrc : out std_ulogic);
```

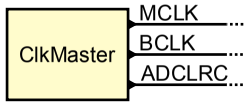


Abbildung 1: Unit `ClkMaster`: Es sind nur die wichtigsten Ports dargestellt.

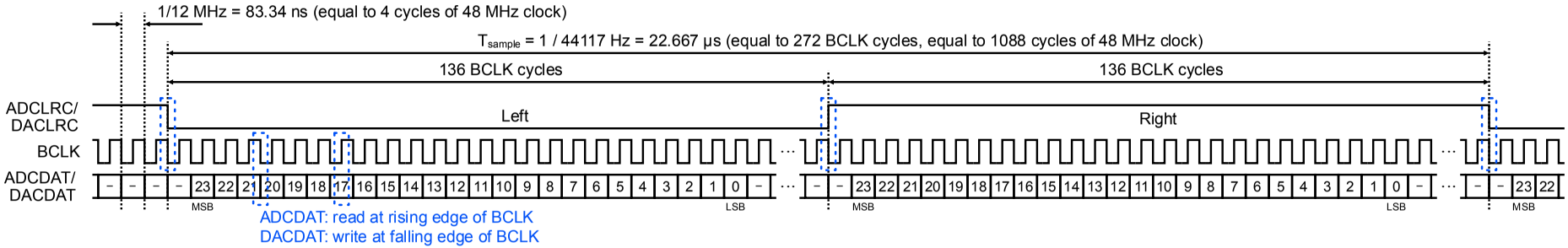


Abbildung 2: Signalverlauf an der digitalen Audioschnittstelle des Codec (I^2S-Format). Die dargestellte Situation bezieht sich auf eine Abtastrate von 44117 Hz. Der Bittakt `Bclk` hat die gleiche Frequenz wie der Systemtakt des Codec (`Mclk`), nämlich 12 MHz. Daher könnten bis zu 12 MHz / 44117 Hz = 272 Bits pro Stereosample übertragen werden. Pro Kanal wären also 272/2 Bits = 136 Bits möglich. Bei lediglich 24 tatsächlich zu übertragenden Nutzbits pro Kanal, besteht der größte Anteil an übertragenen Daten aus belanglosem Füllmaterial (mit „–“ bzw. „· · · “ kenntlich gemacht).

Das I²S-Protokoll ist in Abb. 2 für den Fall dargestellt, welcher später für uns von praktischer Bedeutung sein wird: $f_{\text{Sample}} = 44117\,\text{Hz}$, $f_{\texttt{Mclk}} = 12\,\text{MHz}$. Beachten Sie bitte, dass die Frequenz des Signales `Bclk` durchaus auch 12 MHz sein darf. Da der Takt `Mclk` diese Frequenz bereits zur Verfügung stellt, kann `Bclk` einfach identisch zu `Mclk` sein. Der Aufwand zu dessen Erzeugung ist somit gleich Null.

Es ist wichtig, dass die Phasenlage von `Bclk` und `ADClrc` eingehalten wird. Der Wechsel von `ADClrc` muss also exakt mit der fallenden Flanke von `Bclk` erfolgen.

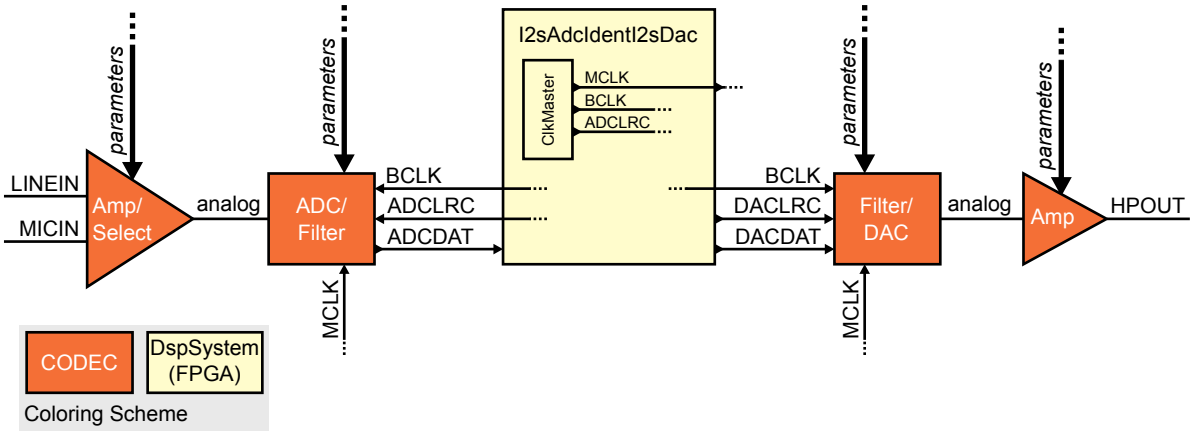


Abbildung 3: Test von ADC und DAC des Codec im *Slave Mode*. Der Codec muss selbstverständlich auch mit den richtigen Parameterwerten konfiguriert werden.

Das Signal `DAClrc` wird nicht von der Unit `ClkMaster`, sondern innerhalb des Testbeds erzeugt. Hierzu soll eine eigene Unit namens `I2sAdcIdentI2sDacSlaveMode` verwendet werden. Diese basiert auf dem bisher eingesetzten Testbed von `ConfigureCodecViaI2c`. Innerhalb der Unit `I2sAdcIdentI2sDacSlaveMode` wird `DAClrc` direkt an `ADClrc` angeschlossen. Die Daten werden wieder einfach vom ADC zum DAC weitergereicht. In Abb. 3 ist der prinzipielle Aufbau des kompletten Testsystems dargestellt.

c☐ Testen Sie Ihre Beschreibungen auf dem Board.

d☐₂ Um nun zu verifizieren, dass die Daten auch wirklich über die I²S-Schnittstelle laufen bietet sich eine Veränderung des Signals `DAClrc` an. Ein besonders einfacher Test besteht darin, das Signal `DAClrc` durch Invertieren des Signals `ADClrc` zu gewinnen. Welche Veränderung erwarten Sie sich unter diesen Umständen für das analoge Ausgangssignal gegenüber dem Eingangssignal?

e☐ Testen Sie, ob dieses Verhalten auf dem Board auch wirklich beobachtet werden kann.

f☐₂ Ein weiterer Test besteht darin, das Signal `DAClrc` gegenüber `ADClrc` um einige Zyklen von `Bclk` verfrüht zu erzeugen. Welche Veränderung am analogen Ausgangssignal erwarten Sie sich davon? Welchen Parameter des analogen Ausgangssignals steuert daher der Grad um den `DAClrc` verfrüht ist?

Dieser Test braucht nicht in der Realität durchgeführt zu werden. Wenn Sie dies dennoch probieren möchten, so kann die Verschiebung von `DAClrc` selbstverständlich ebensogut durch eine entsprechende Verspätung erzielt werden.

g☐₂ Der Codec hat eine digitale Auflösung von 24 Bit pro Kanal. Wie hoch könnte die Abtastfrequenz digitaler Audiodaten maximal sein, wenn diese noch vollständig über die I²S-Schnittstelle übertragen werden sollen? Die Frequenz von `Bclk` soll auch hier wieder 12 MHz betragen.

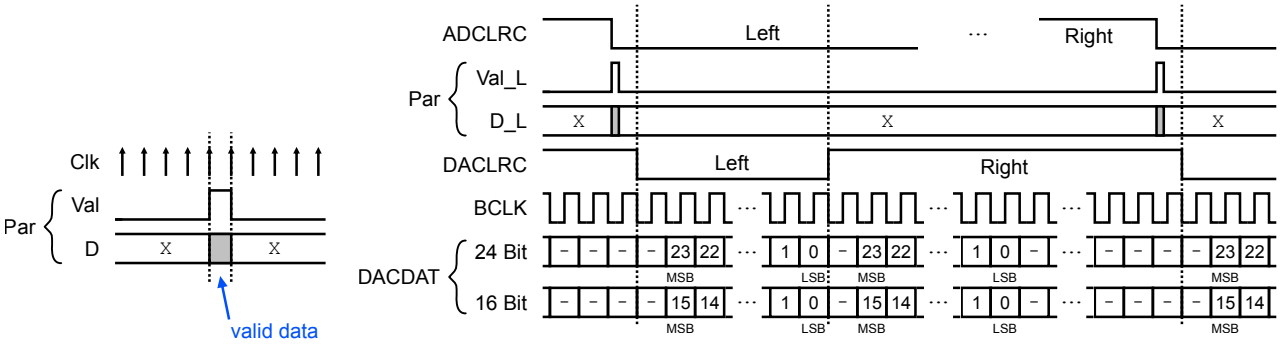
Tipp: Die Abtastfrequenz soll hierbei lediglich von der Datenübertragungsrate auf der I²S-Schnittstelle begrenzt sein (siehe Abb. 2), unabhängig von der tatsächlich maximal möglichen Abtastrate von ADC und DAC. Diese wäre laut Datenblatt mit 96 kHz begrenzt.

2. Aufgabe *ParToI2s*

Während der Codec, um die Anzahl der notwendigen I/O-Leitungen gering zu halten, die Audiosamples in Form eines seriellen Datenstroms (I^2S) sendet und empfängt, ist es für die Audiosignalverarbeitung im FPGA sinnvoll die Samples in Form von ganzen Datenwörtern zu verarbeiten. Jedes Audiosample wird dabei als Zahl dargestellt, deren Größe die Amplitude des Audiosignals angibt. Zu jedem Abtastzeitpunkt wird ein neuer Datenwert verarbeitet. Zwischen den Units unseres Audiosignalverarbeitungssystems sollen also stets zu jedem Abtastzeitpunkt ganze Audiodatenwörter (d.h. alle Datenbits eines Audiosamples *parallel*) übertragen werden. Diese Art der Übertragungsschnittstelle wird oft als Streaming-Interface bezeichnet. Sie sollten beispielsweise bereits das Avalon Streaming Interface (Avalon-ST) aus vorangegangenen Lehrveranstaltungen kennen. In dieser Übung wird jedoch eine einfachere Variante zum Einsatz kommen: das „*Par*"-Protokoll.

Dieses Protokoll, sowie dessen Umsetzung in das I^2S-Protokoll, sind in Abb. 4 dargestellt. Die Übersetzung vom parallelen Streaming-Protokoll Par in das serielle I^2S-Protokoll wird von der fertig zur Verfügung stehenden Unit `ParToI2s` übernommen.

a☐₂ Erstellen Sie zum Testen dieser Unit ein Testbed `TbdRectGen`, mit welchem Sie auf den beiden Audiokälen zwei unterschiedliche Rechtecksignale ausgegeben. Zur Erzeugung der beiden Rechtecksignale können Sie die fertige Unit `RectGen` zuhilfe nehmen. Diese gibt ein Rechtecksignal mit der für die gewählte Auflösung maximalen Amplitude und einer einstellbaren Frequenz aus. Die Samplerate, mit der die Unit die Abtastwerte des Rechtecksignals ausgibt, wird dabei von `ADClrc` gesteuert. In Abb. 5 ist ein Überblick über den Aufbau des zu erstellenden Testbeds dargestellt.



(a) Streaming-Protokoll Par (b) Par und I^2S

Abbildung 4: Signalverlauf von Par und Konvertierung zu I^2S.

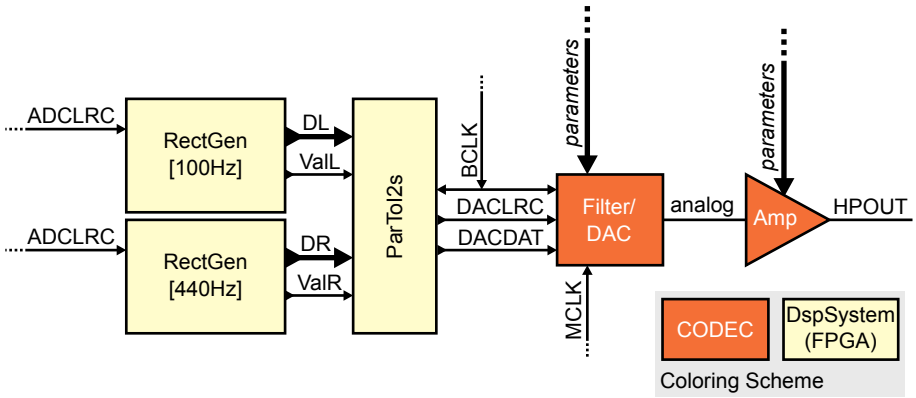


Abbildung 5: Testbed zur Überprüfung der Unit `ParToI2s`. Es wird lediglich der DAC-Teil des Codec verwendet. Er dient zur Ausgabe zweier unterschiedlicher Rechtecksignale auf den beiden Audiokanälen (links und rechts). Die Steuerung der Abtastzeitpunkte wird vom `ClkMaster` über das Signal `ADClrc` vorgenommen.

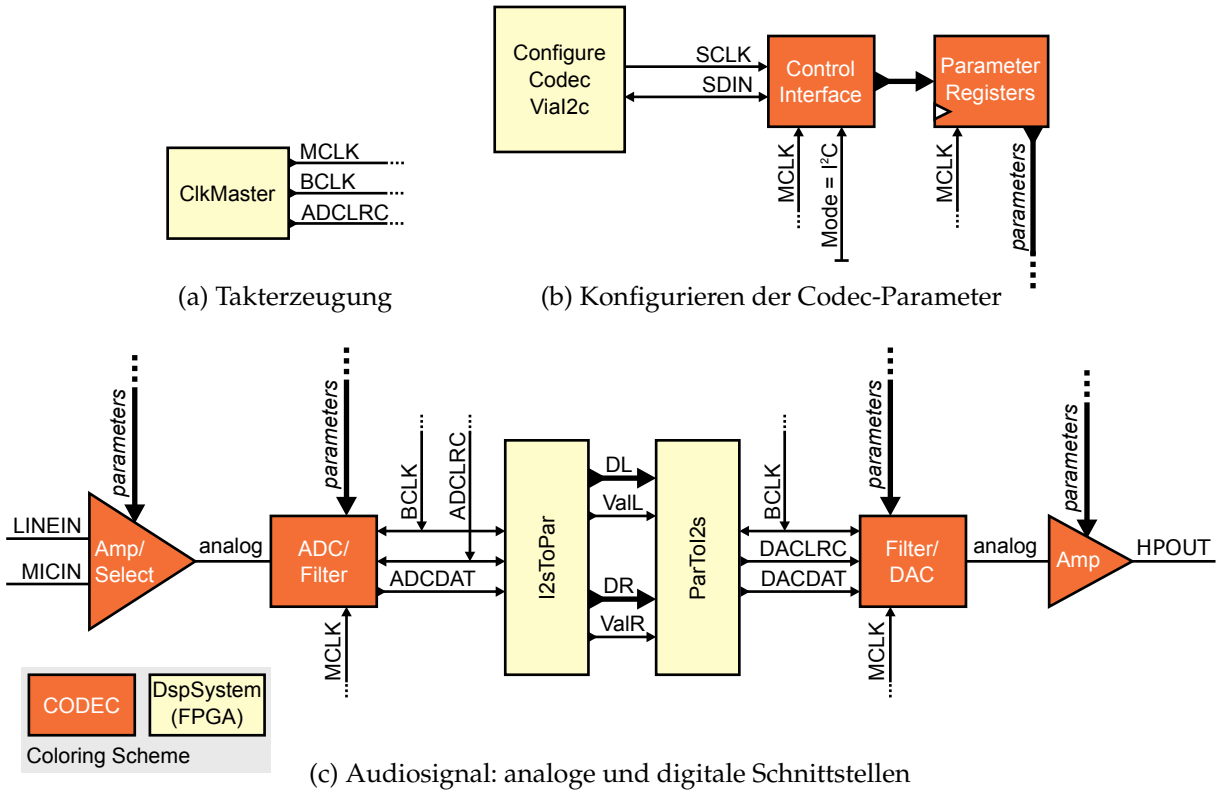
b☐₂ Das Testbed soll zunächst in der Simulation geprüft werden. Erstellen Sie eine entsprechende Testbench und automatisieren Sie den Ablauf der Simulation mittels Scriptsteuerung.

c☐₁ Was fällt am Signal `oLrc` der Unit `ParToI2s` auf? Ist der Verlauf konform zum Datenblatt?

d☐₁ Realisieren Sie das Testbed auf dem Board und messen Sie das Signal mit dem AudioTester (***Screenshot!***).

3. Aufgabe *I2sToPar*

Auch für die Übersetzung vom seriellen I²S-Protokoll in das parallele, taktsynchrone Streaming-Protokoll Par steht eine fertige Unit `I2sToPar` zur Verfügung. In dieser Aufgabe sollen Sie diese innerhalb des Testbeds `TbdI2sParI2s` (Abb. 6) testen. Die Signalverläufe der beiden Protokolle sind in Abb. 7 (vereinfacht) dargestellt.



(a) Takterzeugung

(b) Konfigurieren der Codec-Parameter

(c) Audiosignal: analoge und digitale Schnittstellen

Abbildung 6: Übersicht über das Testbed `TbdI2sParI2s`.

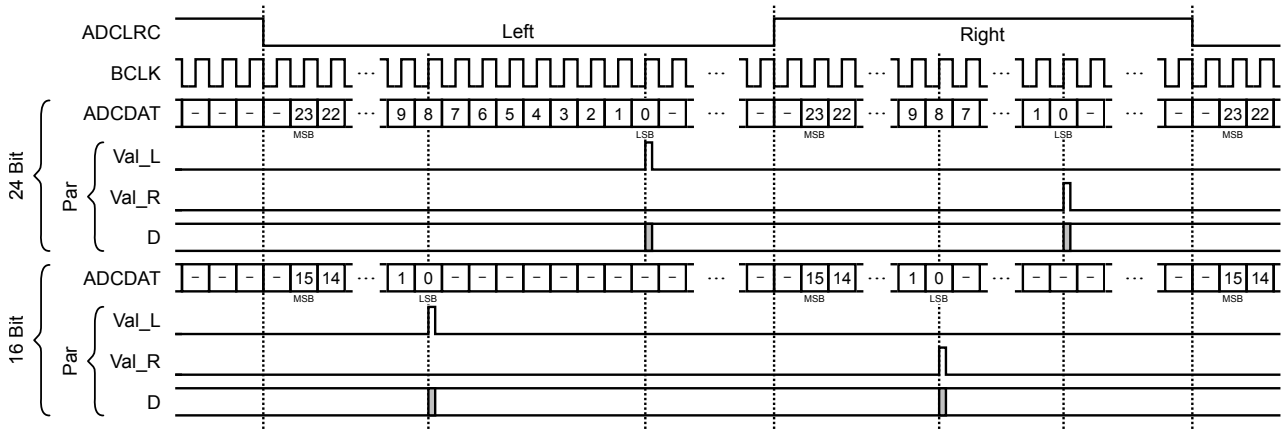


Abbildung 7: Signalverlauf an der Unit `I2sToPar` für die Wortlängen 16 und 24 Bit.

a☐₂ Erstellen Sie ein entsprechendes Testbed und weisen Sie die korrekte Funktion mittels Simulation des Testbed in einer Testbench nach. Außer den seriellen Daten der ADC-seitigen I²S-Schnittstelle des Codec werden alle Signale durch die Unit `ClkMaster` bereitgestellt. In der Testbench wird ein sogenanntes LFSR (***L**inear **F**eedback **S**hift **R**egister*) zur Erzeugung zufälliger Daten verwendet. Der hierfür zuständige Teil der VHDL-Beschreibung:

```
-- Generate pseudo random data with an LFSR. Description is optimized for
-- simulation performance. Code is not synthesizable.
RandomBit : process is
  -- Define a data structure for an LFSR.
  variable vLfsr : std_ulogic_vector(31 downto 0) := (others => '1');
begin
  -- With every falling edge of Bclk.
  wait until (Bclk'event) and (Bclk = '0');
  -- Shift the LFSR once. Shift in a new bit based on linear automata theory.
  vLfsr := vLfsr(vLfsr'left-1 downto 0) & (vLfsr(vLfsr'left) xor vLfsr(11));
  -- Take any of the bits in the LFSR as source for the output.
  ADCdat <= vLfsr(0);
end process;
```

b☐₂ Untersuchen Sie die Abfolge, in der die einzelnen Abtastwerte für den linken und rechten Kanal ausgegeben werden. Interessant sind insbesondere die ersten 50 μs der Simulation. Wie groß ist der Unterschied der Verzögerungen des linken im Vergleich zum rechten Kanal? Welcher von beiden Kanälen wird früher ausgegeben?

Tipp: Sie können annehmen, dass der Codec für beide Kanäle eine identische Verzögerung zwischen der jeweiligen Flanke von `DAClrc` und der hörbaren Ausgabe des Signals aufweist. Betrachten Sie daher die Verzögerung zwischen dem parallelen Datenwort (gültig bei `ValX = '1'`) und der Ausgabe ans Codec (d.h. Flanke von `DAClrc`) für jeden der beiden Kanäle. Achten Sie dabei darauf, dass Sie für beide Kanäle ein Datenwort aus *demselben* Abtastzeitpunkt betrachten.

c☐₂ Die gehörmäßige Richtungswahrnehmung orientiert sich an mehreren Faktoren, von denen einer der Laufzeitunterschied eines Geräuschs bei der Ankunft an linkem und rechtem Ohr ist. Nehmen Sie an, dass der linke Audiokanal ausschließlich das linke Ohr und der rechte Kanal ausschliesslich das rechte erreicht (z.B. bei Verwendung eines Kopfhörers). Wird die Signalquelle in unserem Fall nach links oder nach rechts verschoben? Um welche Distanz?

d☐ Prüfen Sie das Testbed auf dem Board. Schalten Sie versuchsweise den linken Kanal auf der Ebene der parallelen Datenweitergabe (`DL`, `ValL`) ab, um zu prüfen, ob die Audiosignale beide Schnittstellenumsetzungen korrekt durchlaufen.

Abgabe des Sourcecodes

Neben den regulären ***Commits*** im Laufe der Arbeit an jedem Übungszettel ist der Endstand jedes Übungszettels als Tag im Repository abzulegen. Jeder Tag hat dabei einen Namen der Form „ue-<Nr>“ zu tragen, wobei <Nr> die Nummer des jeweiligen Übungszettels ist.

a☐ Erstellen Sie bis spätestens ***23:59 Uhr am Vortag des Abgabetags*** (siehe *Semesterübersicht*) einen Tag `/tags/ue-03` mit dem Endstand Ihrer Übungsausarbeitung.