# Lehrdemonstrationen zur digitalen Filterung auf eingebetteten Systemen

Eduard Funkner Martin Schwarz

2025-06-19

# Inhaltsverzeichnis



## Einführung

Diese Bachlorthesis dient als eine Lerndemonstration zur Implentierung von digitalen biquadratischen-Filtern auf Mikrocontrollern sowie FPGAs. Im Umfang dieser Lerndemonstration werden digitale Biquad-Filter mithilfe von Matlab- und Pythontools entworfen und im Anschluss auf Hard und Software umgesetzt. Der Prozess der Umsetzung wird zum Verständis dokumentiert sowie genau erklärt, damit Leser dieser Lerndemonstration mithilfe ihrer Hardware für sich den Prozess reproduzieren können.

## IIR-Filter zweiter Ordnung

Infinite Impulse Response (IIR) Filter gehören in der Signalverarbeitung zu den grundlegenden Typen der digitalen Filter. Grundlegend zeichnen sich IIR-Filter durch ihre characteristik aus, bei der Berechnung des aktuellen Ausgangssingals durch die Verwendung des gegenwärtigen und vergangenen Eingabewertes als auch des vorherigen Ausgabewertes. Aufgrund dieser Rückkopplung kann es dazu führen, dass die Impulsantwort von IIR-Filtern theoretisch unendlich lang andauern kann, was sie grundsätzlich von Finite Impulse Response (FIR) Filtern unterscheidet.

Ein immenser Vorteil von IIR-Filtern liegt in der hohen Effizienz bei der Realisierung scharfter Frequenzgänge mit relativ wenigen Koeffizienten, während FIR-Filter für vergleichbare

Filtereigenschaften oft hunderte von Koeffizienten benötigen, können IIR-Filter ähnliche Ergebnisse mit deutlich geringerem Rechenaufwand erreichen. Aufgrund dieser Eigenschaft eignen sich IIR-Filter besonders gut im Einsatz von Systemen mit beschränkten Ressourcen oder in Echtzeitverarbeitung.

Biquadratische Filter, kurz als “Biquad” bezeichnet, sind eine spezielle Unterklasse von IIR-Filtern zweiter Ordnung. Durch das Kaskadieren solcher Biquad-Sektionen besteht die möglichkeit komplexe IIR-Filter höhrer Ordnung auf einfache Weise zu realsieren. Das Kaskadieren dieser Sektionen bringt Vorteile in Bezug auf numerische Stabilität, Flexibilät bei der Parameteranpassung und Modularität des Designs.

# Differenzengleichung eines IIR Filters zweiter Ordnung

Digitale Filter können mit einer linearen Differenzengleichung beschrieben werden. Differenzengleichung des Biquad-Filters:

$$y[n] = \frac{1}{a_0}(b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] - a_1 \cdot y[n-1] - a_2 \cdot y[n-2])$$

Der Wert $y[n]$ bezeichnet den Ausgabewert zum Sample $n$, während $x[n]$ den Eingangswert darstellt. Der aktuelle Ausgabewert $y[n]$ ergibt sich aus der gewichteten Summe der aktuellen und zwei vergangenen Eingabewerte, subtrahiert mit den zwei gewichteten vergangen Ausgabewerte. Die Koeffizienten $b_0$, $b_1$ und $b_2$ bestimmen den Einfluss des aktuellen und der vergangenen Eingabewerte (Feedforward), während $a_1$ und $a_2$ den Rückkopplungsanteil aus früheren Ausgabewerten (Feedback) modellieren. Im Allgemeinem wird der Koeffizient auf $a_0$ auf 1 normiert:

$$y[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2 \cdot x[n-2] - a_1 \cdot y[n-1] - a_2 \cdot y[n-2]$$

Zur Analyse des Frequenz- und Phasenverhaltens des Filters, wird die Differenzengleichung mittles der z-Transformation in den z-Bereich transformiert und daraus folgt die folgende Übertragunsfunktion.

# Übertragungsfunktion eines IIR-Filters zweiter Ordnung

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Bei dieser rationalen Funktion wird das Verhälnis zwischen Ausgang zu Eingang im Frequenzbereich beschrieben. Um die Stabilität des Biquad-Filters zu vergewissern, müssen alle Polstellen des Filters innerhalb des Einheitskreises der z-Ebene liegen.

## Implementierungsstrukturen biquadratischer Filter

# Differenzengleichung der Direktform 1:

$y[n] = w[n] - a_1 \cdot y[n-1] - a_2 \cdot y[n-2]$ $w[n] = b_0 \cdot x[n] + b_1 \cdot x[n-1] + b_2$

Differenzengleichung der Direktform 2:

$y[n] = b_0 \cdot w[n] + b_1 \cdot w[n-1] + b_2 \cdot w[n-2]$ $w[n] = x[n] - a_1 \cdot w[n-1] - a_2 \cdot w[n-2]$

Differenzengleichung der transponierten Direktform 2:

$y[n] = b_0 x[n] + s_1[n-1]$ $s_1[n] = s_2[n-1] + b_1 x[n] - a_1 y[n]$ $s_2[n] = b_2 x[n] - a_2 y[n]$

## Bilineartransformation

Mittels der Bilineartransformation ist es möglich einen bestehenden analogen Filter, anhand seiner Übertragungsfunktion zu digitalisieren.Die analogen Biquad Filter werden aus dem Expirement 4 des Analog Systems Lab Kit PRO entnommen.

# Audio Reference Design

Hier wird erklärt wie das Audio Referenzdesign aus dem Base-Overlay entstanden ist. Prinzipiell wurde der Audioanteil aus dem Base-Overlay übernommen und die nicht benötigten Elemente entfernt. Die Einstellungen der Blöcke wurden ebenfalls übernommen. Um das Design in Vivado zu erhalten, wurde das Base-Overlay entsprechend der Anleitung auf dem Pynq-Repo unter Linux neu generiert. ### Einstellungen #### Processing System (PS): In der Clock Config muss eines der Clock Signale auf 100 MHz eingestellt werden, da der *audio_codec_controller* nur 100 MHz Clock Signale akzeptiert. Sollte ein anderer Tackt gegeben werden, gibt es eine Warnung in Vivado. Zusätzlich muss ein ein I2C Port am PS eingeschaltet werden. Im Base-Overlay wurde hierfür *I2C1*.

#### Constraints:

Die Constraints entstammen aus dem Pynq-Repo. Alles, was nicht benötigt wurde auskommentiert. Die Constraints weisen unter anderem den Benötigten internen Pis Namen zu, damit diese besser nachvollzogen werden können. Es ist notwendig, dass die Namen an dem Ein- und Ausgängen aus den Constraints übernommen werden, sonst greifen die Treiber in Pynq nicht.

#### Clocking Wizzard:

Die Einstellungen des Clocking Wizzard wurden aus dem Base-Overlay übernommen. Die Aufgabe des Clocking Wizzard ist es den 10 MHz Master Clock für den Audio Codec und damit der I2S Übertragung zu generieren. Hier wird dieser aus dem 100 MHz Clock generiert. Dies kann auch direkt über den PS mit den entsprechenden Einstellungen generiert werden. Das Wichtige ist, das dieses Tackt-Signal nach außen an den richtigen Pin geführt wird. **Zu den Einstellungen:** - Zur Sicherheit am Anfang die Board Parameter zurücksetzen. - *clk_in1* darf **nicht** auf *sys clock* eingestellt sein! - Die Einstellungen sollten so aussehen. - Wichtig ist, dass unten die *Input Frequency(MHz)* von *clk_in1* auf 100MHz eingestellt ist. - Wenn ein Slider vorhanden ist, sollte dieser auf Auto gestellt werden, dann wird die Input Frequenz übernommen. - Die restlichen Einstellungen sollten bereits so sein. - Der *clk_out1* soll auf 10MHz eingestellt werden. - Der *Reset Type* soll auf *Active Low*, da derselbe Reset vom *codec_controller* verwendet werden soll und dieser ebenfalls *Active Low*. - Bei synchronisierten Rest kann dieser auch *Active High* sein. - Diese letzten beiden Tabs sollten dienen als Überblick der vorherigen Einstellungen und sollten am Ende so aussehen. - Falls nicht: Die vorherigen Einstellungen überprüfen.

#### Audio Codec Controller:

Der *audio_codec_controller* ist eine eigene IP aus dem Pynq-Repo und generiert die vom Codec benötigten Signale sowie steuert den I2S Daten-Stream. Die Namen des Ein- und Ausgänge sowie der IP selbst müssen übereinstimmen, sonst greift der im Pynq enthaltene Audio-Treiber nicht bzw. es muss sonst der neue Name an den Treiber übergeben werden. Die *Codec Address (addr1,addr0)* sollten in den Blockeinstellungen auf *11* bereits voreingestellt sein.

#### Anmerkungen:

Vivado wird Warnungen bezüglich *clk_in1* und *clk_out1* geben, da diese Clocksignale oft durch das Führen nach außen von Vivado als neuer Clock-Tree missinterpretiert werden, obwohl diese an dem Haubt Clock-Tree verbunden sind. Diese Warnungen können entweder ignoriert werden oder es kann Vivado explizit die Zusammenhänge der Clocks mitgeteilt werden.

## Referenzdesign v2

Als Folge des Versuches Echtzeitfilterung umzusetzen entstand eine leicht veränderte Version des ursprügliches Referenzdesign. Dieses unterscheidet sich nur in der Art wie das 10MHz Clock Signal generiert wird. In diesem aktualisierten Design wird das 10 MHz-Taktsignal direkt im Processing System (PS) erzeugt, mit einem synchronisierten Reset-Signal. Der Vorteil dieses Ansatzes besteht darin, dass Vivado keine Probleme mit den Clock-Trees erkennt und entsprechend auch keine Warnungen ausgibt. Zusätzlich zum Taktsignal steht nun auch ein passendes, synchronisiertes Reset-Signal zur Verfügung.

Das aktualisierte Design wird aller Voraussicht nach die Grundlage für zukünftige Implementierungen bilden.