

Milestone 2: Audio-Bitcrusher

Lernziele

- Wiederholung und Vertiefung: Vivado-Projekte, VHDL und Synthese
- Arbeiten mit IP-Integrator
- Anwenden des AXI-Stream-Protokolls
- Verwendung des ILA

Aufgabenstellung

Es soll ein „Bitcrusher“ für Mono-Audiosignale mit AXI-Stream-Schnittstellen entworfen werden. Der Versuch orientiert sich grob am Beispiel des Audio-Filters aus der Vorlesung.

Obwohl das IP mithilfe einer rein kombinatorischen Schaltung realisiert werden könnte, soll Ihre Lösung auf einem endlichen Automaten basieren. Auf diese Weise erwerben Sie Knowhow, dass Ihnen später, bei komplexeren Aufgaben behilflich sein wird.

Der endliche Automat soll mit drei Zuständen (**Eingabe**, **Berechnung**, **Ausgabe**) realisiert werden. Im Zustand **Eingabe** wird ein **Eingangswert** über das **AXIS-Slave-Interface eingelesen** und in einem **Register gespeichert**. Im nachfolgenden Zustand **Berechnung** erfolgt die Reduktion der Wortbreite. Das Ergebnis dieses zweiten Schritts wird ebenfalls in einem **Register abgelegt**¹. Im Zustand **Ausgabe** wird der zuvor berechnete Wert über das **AXIS-Master-Interface ausgegeben**.

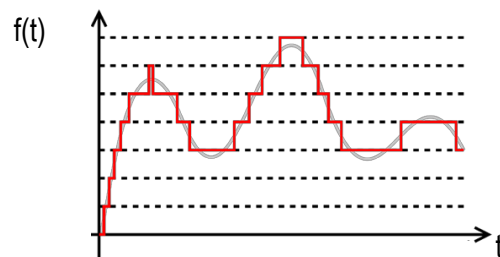
Mit diesem Ansatz benötigt das IP drei Taktzyklen zur Berechnung eines Ausgangswertes und es können nur alle drei Taktzyklen neue Werte an das IP übergeben werden. Im Hinblick auf den Datendurchsatz ist diese Lösung also schlechter als eine rein kombinatorische Lösung. Dies wird aus den oben genannten Gründen bewusst in Kauf genommen.

Hinweis: In den Zuständen ‚Eingabe‘ und ‚Ausgabe‘ muss jeweils das Valid/ Ready-Handshake des AXI-Protokolls beachtet werden. Dies bedeutet zum Beispiel, dass der Zustand ‚Eingabe‘ erst verlassen werden darf, nachdem ein gültiger Eingangswert (gekennzeichnet durch valid = 1) übertragen wurde.

Hintergrund

Ein Audio-Bitcrusher vermindert die Wortbreite der Abtastwerte. In diesem Milestone werden Mono-Signale verarbeitet, deren Abtastwerte eine Wortbreite von 16 bit (vorzeichenbehaftet) besitzen, was einem Wertebereich von -32768 bis 32767 entspricht.

Werden statt 16 weniger Bits verwendet, verringert sich die Auflösung²: Die Signalform des Audiosignals wird „eckiger“. Akustisch wird dies als Rauschen wahrgenommen. Die nachfolgende Abbildung zeigt ein hochaufgelöstes Signal (grau) und ein entsprechendes niedrigaufgelöstes Signal (rot).

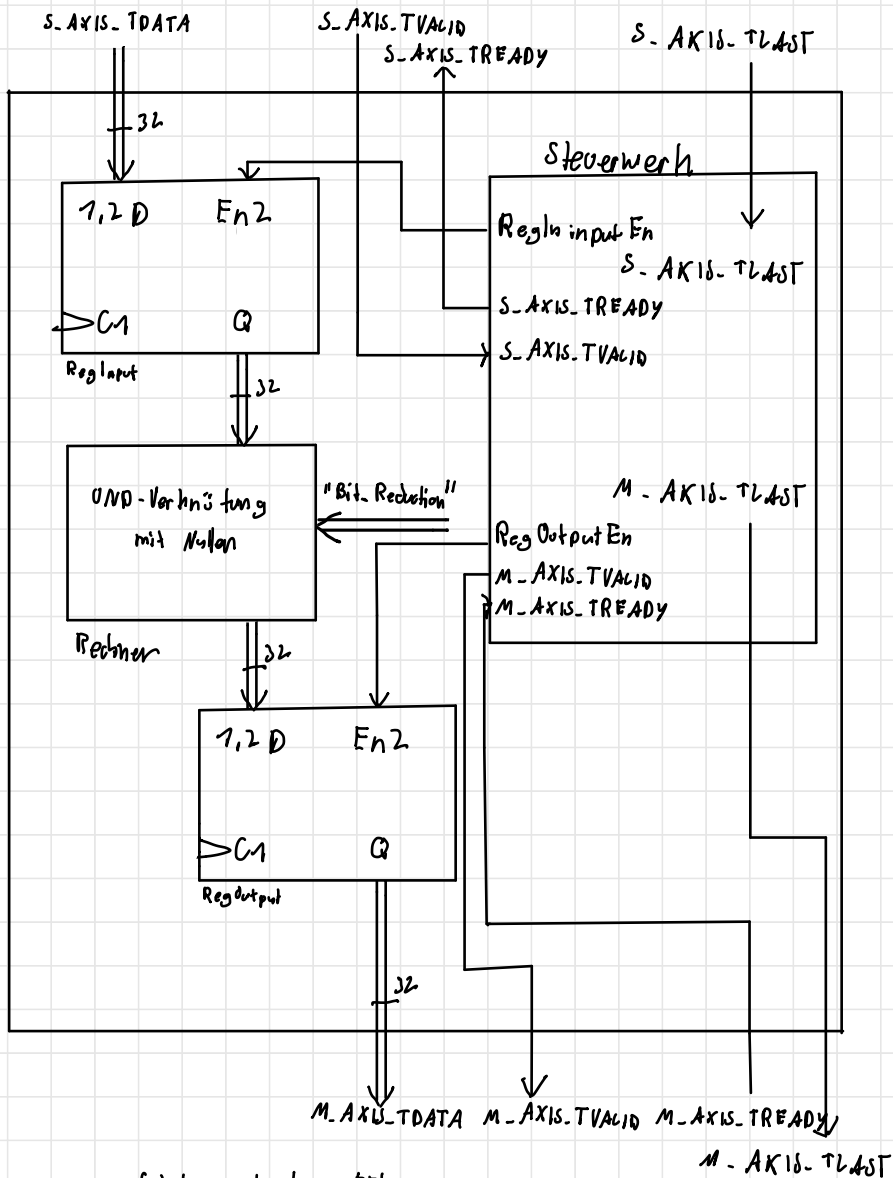


Bildquelle: [https://de.wikipedia.org/wiki/Quantisierung_\(Signalverarbeitung\)](https://de.wikipedia.org/wiki/Quantisierung_(Signalverarbeitung))

¹ Sie können für den ersten und den zweiten Verarbeitungsschritt das gleiche Register verwenden. Ebenso können Sie ein neues Register in der zweiten Stufe einsetzen. Falls Sie nicht sicher sind, wie Sie diese Varianten in VHDL beschreiben, wenden Sie sich an Ihren Versuchsbetreuer.

² Dieses Vorgehen wird häufig auch als *Quantisierung* bezeichnet.

axis-audio-bitcrusher - Architecture



Beispiel

Setzt untersten Bits auf 0

g:8 → 0x1200 ✓

g:12 → 0x1000

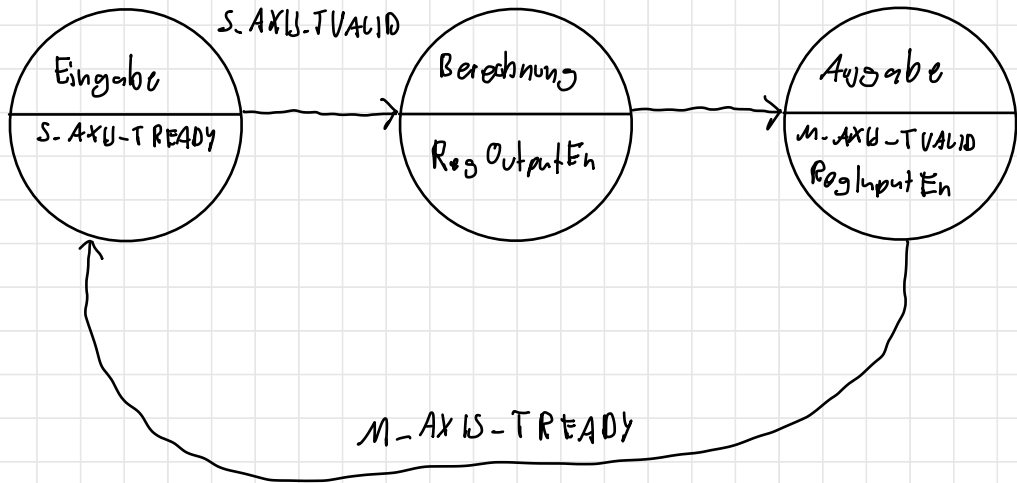
Wert: 0x1234

soll beim letzten Wert TLAST gesetzt werden?

axis_audio_bitcrusher ~ andler best-and-joint-met

Input : M-AXIS-TREADY
S-AXIS-TVALID

Moat : M-AXIS-TVALID RegInputEn
S-AXIS-TREADY RegOutputEn



Besonderheiten

Da die Audiohardware des ZyBo-Boards Abtastwerte mit einer Wortbreite von 16 bit erwartet, kann also keine Wortbreitenreduktion verwendet werden. Um dennoch einen Effekt zu erreichen, der einer Wortbreitenreduktion entspricht, soll das IP die unteren Bits der Abtastwerte auf null setzen. Die Anzahl der auf null gesetzten Bits soll konfigurierbar sein.

Zur Verdeutlichung ein Beispiel in hexadezimaler Darstellung: Der zu verarbeitende Abtastwert besitzt den Wert 0x1234. Werden die untersten 8 bit auf null gesetzt, ist der Ausgangswert 0x1200. Werden dagegen die 12 niederwertigsten Bit auf null gesetzt, würde das IP den Wert 0x1000 ausgeben.

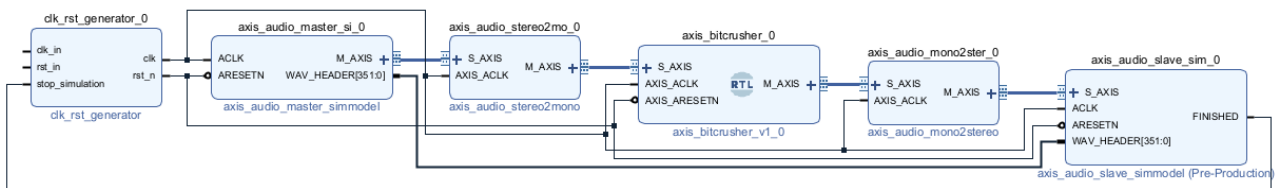
Durchführung

Entwurf des „axis_audio_bitcrusher“

1. Erstellen Sie das Vivado-Projekt „axis_audio_bitcrusher“ und fügen Sie die VHDL-Datei „axis_audio_bitcrusher.vhd“ hinzu.
2. Fügen Sie dem Projekt die vorgegebene Constraints-Datei „milestone2.xdc“ hinzu.
3. Die vorgegebene VHDL-Datei „axis_audio_bitcrusher.vhd“ enthält nur die Entity. Ergänzen Sie die Datei und realisieren Sie den Audio-Bitcrusher mit AXI-Stream-Schnittstelle.

Simulation

1. Fügen Sie das IP-Verzeichnis „IP“ (aus den Vorlesungsdaten) dem Projekt hinzu.
2. Erstellen Sie ein Block-Design *design_1_sim* und fügen Sie den Bitcrusher hinzu und bauen Sie eine Simulationsumgebung entsprechend des nachfolgenden Diagramms auf.

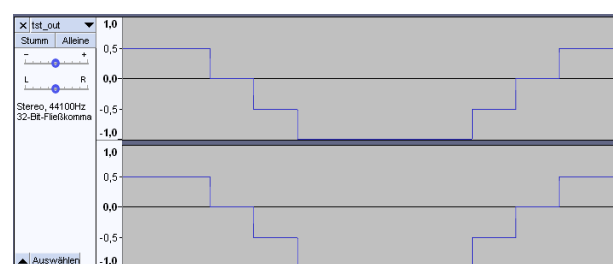
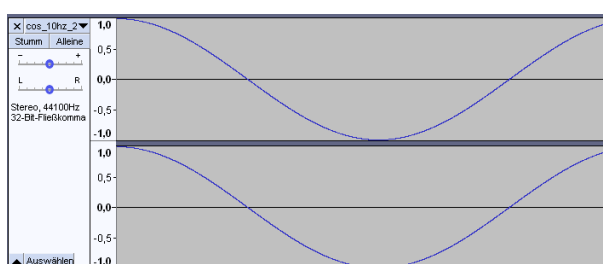


3. Erstellen Sie den HDL-Wrapper und achten darauf dass der Wrapper unter *Simulation Sources* als Top-Level ausgewählt ist.
4. Tragen Sie in den IPs *axis_audio_master_simmodel* und *axis_audio_slave_simmodel* Audiodateinamen ein. Stellen Sie sicher, dass die Eingabedatei unter dem angegebenen Pfad verfügbar ist.
5. Simulieren Sie Ihr IP.

Bitte beachten: Mit den Default-Einstellungen des Audio-Master-IPs wird nur alle 2083 Taktzyklen ein neuer Wert ausgegeben. Dies entspricht zwar der Realität in der Hardware, führt aber zu langen Simulationszeiten. Alternativ können Sie die Zeit zwischen zwei Audiosamples in der Konfiguration des Master-IPs reduzieren (z.B. auf 3).

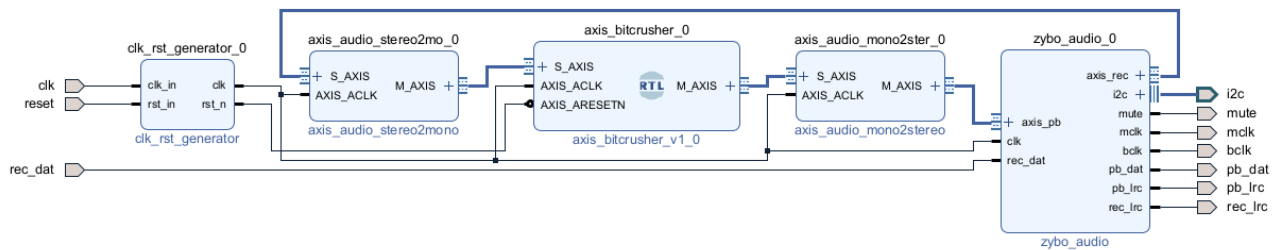
Nachfolgend sehen Sie die Darstellung der Ergebnisse mit Hilfe des Audioprogramms Audacity (<https://www.audacityteam.org/>). Hierbei wurde die Audiodatei „cos_10hz_25ms.wav“ für die Simulation mit der Parametrierung BIT_REDUCTION = 14 verwendet.

Es werden jeweils die unteren 14 Bit auf Null gesetzt. Damit kann das Ausgangssignal nur noch 4 unterschiedliche Werte annehmen: $2^{14} = 16384$, 0, $-2^{14} = -16384$ und $-2^{15} = -32768$.



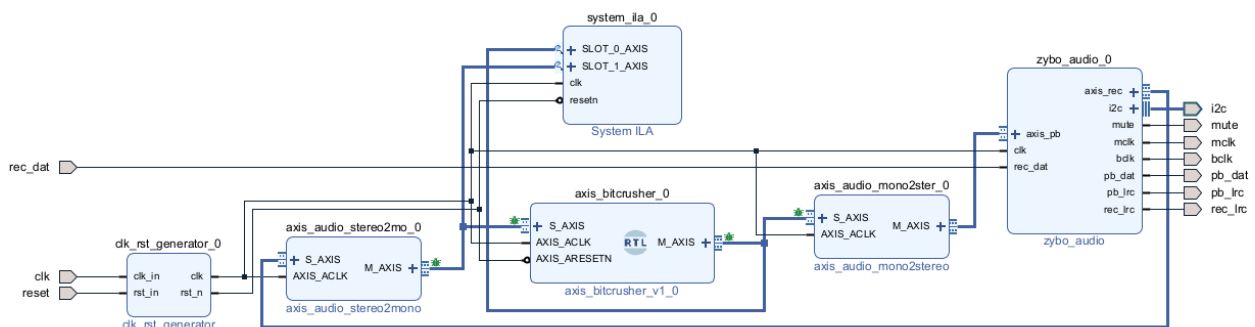
Synthese & Test

- Erstellen Sie ein Block-Design *design_1_syn* und fügen Sie den Bitcrusher hinzu. Bauen Sie ein Block-Design entsprechend des nachfolgenden Diagramms auf.

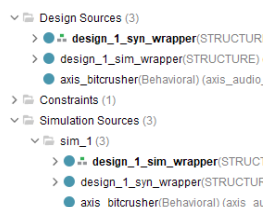


- Wählen Sie für die AXIS-Anschlüsse des Bitcrushers die Debug-Funktion hinzu. Vivado bietet Ihnen eine *Connection Automation* an. Nehmen Sie dieses Angebot an in dem Sie auf *Run Connection Automation* klicken und anschließend *All Automations* anwählen. (Eventuell müssen Sie einige Verbindungen selbst herstellen.)

Anschließend sollte das Blockdesign etwa so aussehen:



- Erstellen Sie den HDL-Wrapper und achten darauf dass der Wrapper unter *Design Sources* als Top-Level ausgewählt ist.



- Verbinden Sie den Audioausgang Ihres PCs mit dem Line-In-Eingang des ZyBo-Boards und schließen Sie den Kopfhörerausgang (HPH OUT) an Ihre Boxen bzw. Ihr Headset (mit einem 3-poligem Klinkenstecker) an.
- Synthetisieren Sie das Design und laden Sie die Bitstream-Datei auf das ZyBo-Board.
- Testen Sie die Funktionsweise sowohl akustisch als auch mit dem integrierten Logicanalyzer (ILA). Sie können die vorgegebenen Audiodateien (Netcase-Ordner *Support/Audio/AudioSnippets*) verwenden. Verwenden Sie unterschiedliche Wortbreiten-Konfigurationen des Bitcrushers.

Tipp: Verwenden Sie eine sinnvolle Triggerbedingung für den ILA, zum Beispiel:

Trigger Setup - hw_ila_1			
Capture Setup - hw_ila_1			
Name	Operator	Radix	Value
slot_0: axis_bitcrusher_0_M_AXIS: TVALID	==	[B]	1

Abgabe

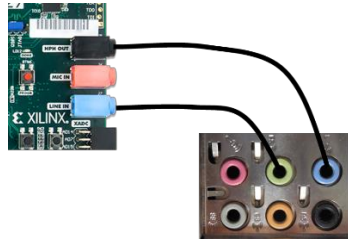
Um den Speicherbedarf von Vivado-IP-Projekten zu reduzieren, können Sie alle Verzeichnisse und Dateien mit Ausnahme der folgenden löschen: Verzeichnis <projectname>.srcs und Projektdatei <projectname>.xpr

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

Fügen Sie auch aussagekräftige Screenshots der Messungen mit dem ILA hinzu.

Weitere Experimentiervorschläge für Interessierte

1. Verbinden Sie nun den HPH-OUT-Ausgang des Zybo-Boards mit dem Line-In-Eingang der Soundkarte Ihres PCs. So können Sie mit einer Audiosoftware (zum Beispiel mit dem Open-Source-Programm *Audacity*) das veränderte Audiosignal mit Ihrem PC aufnehmen. Vergleichen Sie das aufgenommene Audiosignal mit der abgespielten Audiodatei.



2. Nehmen Sie den Audio-Filter mit in Ihr System und versuchen Sie unterschiedliche Einstellungen. Sie können auch, wenn Sie ein Mikrofon besitzen, dieses anstelle des PCs an das Zybo-Board anschließen. Denken Sie daran, im IP *zybo_audio* den Mikrofon-Eingang auszuwählen.
3. Falls Sie eine besondere Herausforderung suchen, können Sie auch ein IP realisieren, das von 8 eingelesenen Abtastwerten nur einen (den aber 8-mal) weitergibt. Die anderen 7 eingelesenen Abtastwerte werden verworfen. Mit diesem Effekt wird de facto die Abtastfrequenz reduziert. Dies führt (genauso wie die Reduktion der Wortbreite) zu "Fehlern" im Audiosignal. Als Testsignal können Sie Musik verwenden, die Sie mit einem Audioprogramm (z.B. Audacity) in das WAV-Format umwandeln. Unabhängig davon, ob Sie dieses IP realisieren: Wenn Sie wissen wollen, was bei Verwendung dieses IPs passiert, erklären wir Ihnen das gerne. Oder Sie googlen eigenständig zum Stichwort "Aliasing". *Warnung: Wir haben aktuell keine Musterlösung für dieses IP ;-)*