

Milestone 5: Audioplayer unter Linux

Lernziele

- Hardware-/Software-Kommunikation mit ACP-Port
- Inbetriebnahme eines Systems unter Linux
- Verwendung von Hardware-Interrupts unter Linux
- Linux-Softwareentwicklung für das ZyBo-Board

Aufgabenstellung

In diesem Milestone erstellen Sie einen Audioplayer, mit dem synthetisch generierte Töne und WAV-Dateien auf dem Headphone-Ausgang des ZyBo-Boards ausgegeben werden können.

Die unter Linux laufende Software soll die Ausgabedaten generieren (bzw. von der SD-Karte lesen) und in einem Pufferspeicher im SDRAM ablegen. Von dort werden die Daten von den Hardwarekomponenten gelesen und über eine I²S-Schnittstelle an den Audio-Codec-Baustein des ZyBo-Boards übergeben.

Da zeitgleich Daten generiert und ausgegeben werden müssen, werden zwei Pufferspeicher verwendet: Während die Daten eines Pufferspeichers (durch die Hardware) ausgegeben werden, kann die Software zeitgleich neue Ausgabedaten in einem zweiten Speicherbereich bereitstellen.

Der Wechsel der beiden Pufferspeicher erfolgt in der Hardware "automatisch" durch das IP *axi_2d_mm2vs*. Mit Hilfe eines Interrupts signalisiert das IP den Wechsel der beiden verwendeten Speicherbereiche.

Durchführung

Hardware-Design

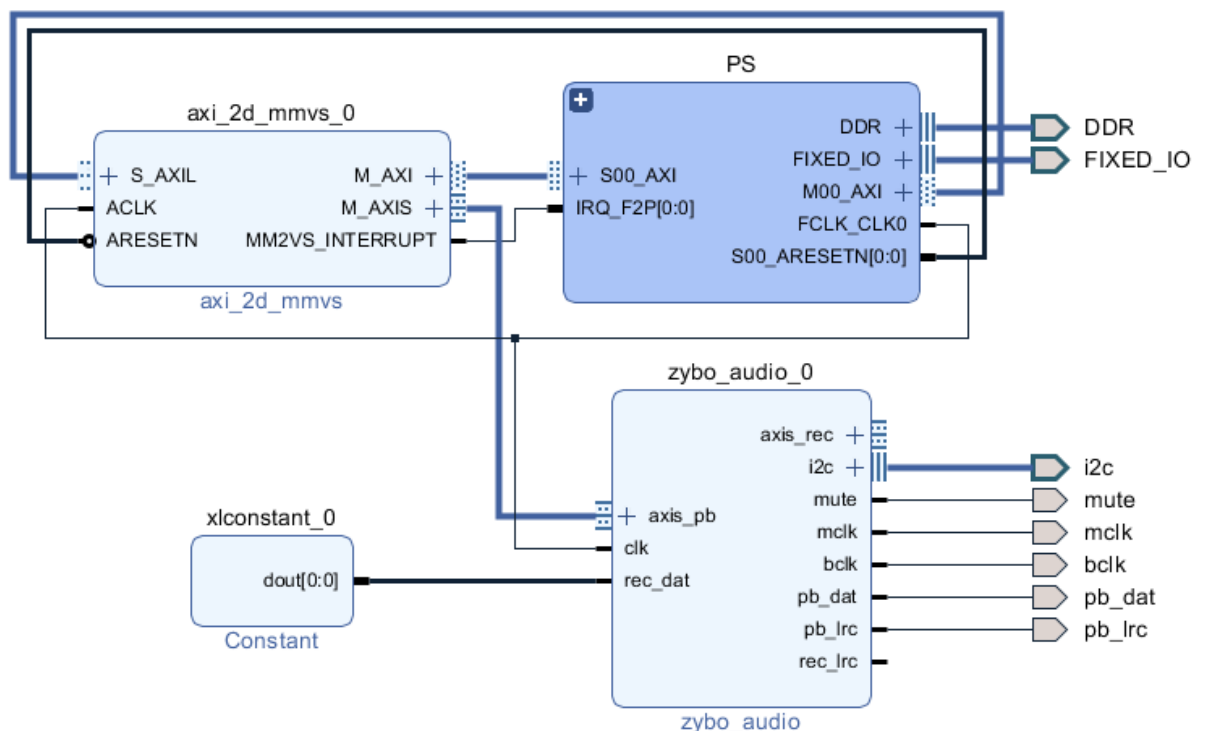
1. Erstellen Sie ein neues Vivado-Projekt *Milestone_5* und fügen die Constraints-Datei *milestone5.xdc* hinzu. Tragen Sie auch das Verzeichnis mit den vorgegebenen IPs unter den IP-Repositories ein.
2. Führen Sie das TCL-Skript *Zybo-Z7-20_Minimal_design_1.tcl* (zu finden im Ordner „Support / Startsysteme“) aus. Sie erhalten so ein Blockdesign, das ein für Linux konfiguriertes Processing-System enthält und auch zu den in Uboot enthaltenen Source-Dateien passt.
3. Aktivieren Sie im Processing-System den AXI-Master GP0 und den ACP-Port. Achten Sie darauf, dass Sie auch "Tie off AxUSER" auswählen.

▼ ACP Slave AXI Interface		
S AXI ACP interface	<input checked="" type="checkbox"/>	Enables AXI coherent 64-bit slave interface
Tie off AxUSER	<input checked="" type="checkbox"/>	Tie off AxUSER signals to high, enabling cohere

Der Speicherzugriff erfolgt in diesem System über den ACP-Port und nicht über einen der vier HP-Ports. Speicherzugriffe über den ACP-Port sind mit dem Cache der CPU synchronisiert, so dass Daten, die von der CPU erzeugt worden sind, aber noch nicht im SDRAM liegen, korrekt ausgelesen werden. Sollte dieser Aspekt für Sie unklar sein, fragen Sie im Rahmen einer Live-Veranstaltung nach.

4. Fügen Sie dem Blockdesign die IPs *axi_2d_mm2vs* und *zybo_audio* hinzu und lassen Sie die Verdrahtung der AXIL- und Full-AXI-Interfaces von Vivado vornehmen.
5. Nehmen Sie die Konfiguration des *zybo_audio*-IPs vor: Da der Systemtakt 100 MHz beträgt (FCLK_CLK0 des Processing Systems), stellen Sie für den I²S-Clock-Divider den Wert 3 ein. So können Audiosignale mit einer Abtastfrequenz von 48 kHz (annähernd) korrekt ausgegeben werden.

6. Konfigurieren Sie nun das Modul *axi_2d_mmvs*, indem Sie das Untermodul VS2MM deaktivieren und die Interrupt-Ausgänge aktivieren. Die anderen Einstellungen (Defaultwerte für die Register) können Sie unverändert lassen. Sie werden später durch Software gesetzt.
7. Verbinden Sie den AXIS-Ausgang des MMVS-IPs mit dem AXIS-Eingang des Audio-IPs.
8. Verbinden Sie den Interruptausgang des MMVS-IPs mit dem Interrupteingang des Processing-Systems (IRQ_F2P).
9. Erstellen Sie zur besseren Übersichtlichkeit eine Hierarchieebene (Create Hierarchy) und fassen Sie in dem neu erzeugten Block alle Blöcke – mit Ausnahme der IPs *axi_2d_mmvs* und *zybo_audio* – zusammen.
10. Führen Sie die Signale des Blocks *zybo_audio* nach außen. Achten Sie darauf, die gleichen Bezeichnungen wie in der nachfolgenden Abbildung zu verwenden. Die Ausgänge *rec_lrc* und *axis_rec* bleiben hierbei offen. Den Eingang *rec_dat* sollten Sie auf einen konstanten Wert legen. Hierfür bietet sich das Xilinx-IP *constant* an:



11. Überprüfen Sie im Address Editor, welche Adresse dem IP *axi_2d_mmvs* zugewiesen wurde. Dies sollte die Adresse 0x43C00000 sein. Ist dies nicht der Fall, ändern Sie die Adresse.
12. Lassen Sie das Blockdesign von Vivado überprüfen, indem Sie auf den Button "Validate Design" im oberen Bereich des Blockdesign-Editors klicken. Die vier eventuell auftretenden Warnungen bezüglich des SDRAM-Timings können Sie ignorieren. Alle anderen eventuell auftretenden Warnungen oder Fehlermeldungen sollten Sie entsprechend bearbeiten. Bei Unklarheiten wenden Sie sich im Rahmen der Support-Termine an Ihre Betreuer.
13. Erzeugen Sie den HDL-Wrapper für das Block Design.
14. Bevor Sie den Bitstream erzeugen, sollten Sie auswählen, dass auch eine Datei im .bin-Format erstellt wird. Diese Auswahl erfolgt in den Bitstream-Settings, die Sie durch Rechtsklick auf "Program and Debug" im Flow Navigator aufrufen können.
15. Erzeugen Sie nun den Bitstream und laden Sie die erzeugte Datei *design_1_wrapper.bin* (aus dem Verzeichnis <Projektname>.runs/impl1) auf die FAT-Partition der SD-Karte (vgl. Vorlesungs-Videos).
16. Benennen Sie die Datei *fpga.bin* in der FAT-Partition der SD-Karte um (so können Sie im Fall von Fehlern den ursprünglichen Zustand der SD-Karte leicht wiederherstellen). Anschließend benennen Sie die Datei *design_1_wrapper.bin* in *fpga.bin* um.
17. Booten Sie das Zybo-Board.

Software des Audioplayers

1. Starten Sie die virtuelle Linux-Maschine. Nachdem Booten starten Sie die Entwicklungsumgebung Eclipse. Im Workspace es1 finden Sie ein vorbereitetes Projekt mit dem Namen *Milestone_5*, welches die Sourcedateien *gip.h* und *Milestone5.cpp* enthält.
2. Erstellen Sie ein Programm, das kontinuierlich einen sinusförmigen Ton ausgibt. Beachten Sie die folgenden Hinweise:
 - Kommentare zur Bedeutung der Register des IPs *axi_2d_mmvs* finden Sie in der Header-Datei *gip.h*. Weitere Informationen zu dem IP können Sie auch dem IP-Datenblatt *axi_2d_mm2vs-IP-Datenblatt.pdf* entnehmen.
 - Sie sollten das Register *MM2VS_InterruptLine* auf die Anzahl der verwendeten Zeilen (*MM2VS_VerticalLines*) minus 1 setzen. So erhalten Sie den Interrupt frühzeitig nachdem die Pufferspeicher von der Hardware gewechselt worden sind. Mit Hilfe des Registers *MM2VS_LastFrameAddress* erhalten Sie die (physikalische) Startadresse des Pufferspeichers, der gerade nicht von der Hardware benutzt wird und somit mit neuen Daten überschrieben werden kann.

Achtung, Sonderfall beachten! Beim ersten Interrupt nach Start des IPs nach einem Reset ist *MM2VS_LastFrameAddress* gleich Null und zeigt somit nicht auf eine gültige Adresse. Diesen Fall sollten Sie abfangen. Eine Möglichkeit hierzu ist, dass Sie *MM2VS_LastFrameAddress* auf Null testen. Die Daten müssten dann zu Beginn des Pufferspeichers abgelegt werden.

- Die Startadresse für die beiden Pufferspeicher muss größer oder gleich 0x3000 0000 sein, da der darunter liegende Speicherbereich von Linux genutzt wird. Der Democode verwendet die Adresse 0x3800 0000.
- Das Mapping des Speichers kann mit Hilfe von UIO16 erfolgen. Über dieses Gerät stehen Ihnen 0x2000 0000 Bytes ab Adresse 0x3000 0000 zur Verfügung.
- Beachten Sie, dass nach jedem MM2VS-Interrupt das Zurücksetzen des Interrupt-Status durch Löschen des *InterruptStatus*-Registers erfolgen muss.

Eine mögliche Lösung steht Verfügung: Die Lösung wird in den Dateien *democode_stufe[1...5].cpp* schrittweise verfeinert. Versuchen Sie ohne diese Lösung auszukommen bzw. möglichst wenige der vorgegebenen Schritte zu verwenden.

3. Übersetzen Sie das Programm durch Anklicken des Hammer-Symbols.
4. Legen Sie für das Projekt eine „Debug-Konfiguration“ an. Das Vorgehen ist in den Vorlesungs-Videos beschrieben.
5. Starten Sie das Debugging auf dem ZyBo-Board und überprüfen Sie die Ausgabe mit Hilfe Ihres Kopfhörers bzw. Headsets. **Achtung:** Die ausgegebenen Töne können laut sein.

Es sollte ein kontinuierlicher Ton zu hören sein, der nicht durch gelegentliches leichtes Knacken unterbrochen wird.

6. Erweitern Sie Ihren Code: Nun soll eine WAV-Datei ausgegeben werden. Beispiele für WAV-Dateien finden Sie auf dem Zybo-Board im Ordner */home/user/Music*.

Führen Sie die Erweiterung des Codes so durch, dass beide Funktionen (Sinuston, Datei abspielen) über ein *#define* auswählbar sind.

Beachten Sie, dass die ersten 44 Bytes einer WAV-Datei den Header enthalten. Diese Daten sollten nicht an den Audio-Codec ausgegeben werden. Die Daten nach dem Header können Sie unverändert in den Pufferspeicher kopieren.

Ihnen wird auffallen, dass die vorgegebenen Dateien nicht ganz korrekt klingen: Die Dateien sind mit einer Abtastfrequenz von 44,1 kHz aufgezeichnet worden und werden nun mit 48 kHz abgespielt. Dateien, die korrekt mit einer Abtastfrequenz von 48 kHz abgespielt werden können, finden Sie im Netcase-Verzeichnis *Support\Audio\AudioSnippets*.

Abgabe

Erstellen Sie einen Screenshot vom Ressourcenverbrauch der Programmable Logic, fügen Sie diesen in ein Dokument ein, aus dem sie bitte eine PDF-Datei erzeugen.

Erstellen Sie eine ZIP-Datei, in der Sie das o.g. PDF-Dokument und Ihren finalen C-Code einfügen.