

Milestone 6: IP für Videoverarbeitung

Lernziele

- Entwurf von Videostreaming-IP
- Simulation von Video-IP
- Verwendung von Zeilenspeichern
- Pipelining in Streaming-IP

Aufgabenstellung

In diesem Milestone erstellen Sie ein IP für die Filterung von Videosignalen, simulieren dieses IP und testen das IP unter Linux mit Hilfe von OpenCV.

Das Filter-IP soll für die 3x3-Filterung ausgelegt werden. Die Filterkoeffizienten und die Skalierung der Ausgangswerte des Filters sollen über ein AXIL-Interface wählbar sein.

Die Zieltaktfrequenz des IPs beträgt 100 MHz. Mit dieser Taktfrequenz ist es nicht möglich, die gesamte Filterfunktion in einem Taktzyklus zu realisieren. Da Videoverarbeitung in der Regel hohe Durchsatzraten erfordert, ist es sinnvoll, das Filter-IP so zu entwerfen, dass in jedem Taktzyklus ein neues Pixel verarbeitet werden kann. Hierzu wird die Filterfunktion in mehrere Teile aufgespalten, die von den Pixeln nacheinander durchlaufen werden (Fachbegriff: Pipelining).

Zur Realisierung des Systems werden neben dem zu entwerfenden Filter-IP auch vorgegebene IP-Module eingesetzt. Hierzu zählen Zeilenspeicher, IP-Blöcke zur Formatierung des Videostroms und Simulations-IP für den Test von Videostreaming-Komponenten.

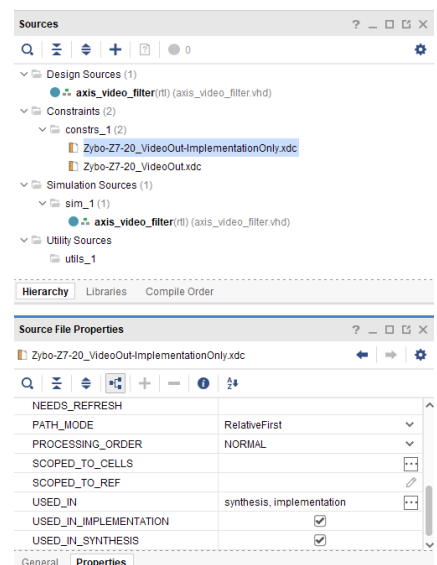
Durchführung

Realisierung des 3x3-Filter-IPs

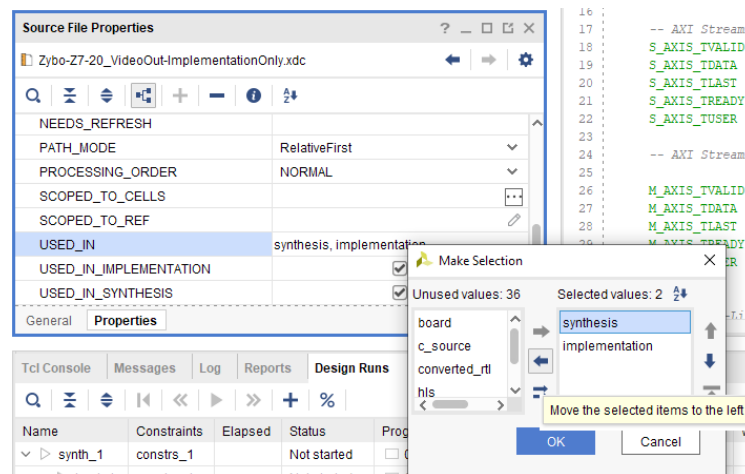
1. Erstellen Sie ein neues Vivado-Projekt *milestone6* und fügen Sie die vorgegebene VHDL-Datei *axis_video_filter.vhd* hinzu. Diese VHDL-Datei enthält den Startpunkt für die Erstellung des Filter-IPs.
2. Da später auch die Synthese eines Systems erfolgen soll fügen Sie bereits jetzt die beiden Constraints-Dateien *Zybo-Z7-20_VideoOut.xdc* und *Zybo-Z7-20_VideoOut-ImplementationOnly.xdc* zum Projekt hinzu.

Achten Sie darauf, dass „Copy constraints files into project“ ausgewählt ist.

3. Nachdem das Projekt erfolgreich angelegt ist, müssen die Eigenschaften der Constraints-Datei *Zybo-Z7-20_VideoOut-ImplementationOnly.xdc* angepasst werden: Diese Datei (wie der Name bereits aussagt) wird nur für den Schritt „Implementation“ und nicht für den Schritt „Synthesis“ verwendet. Um Vivado dies mitzuteilen, wählen Sie die Constraints-Datei aus und scrollen im Source-File-Properties-Fenster bis zum Eintrag „USED IN“ (vgl. Abbildung rechts)
4. Durch Anklicken des Symbols mit drei Punkten am rechten Rand können Sie die Auswahl treffen, in welchen Übersetzungsschritten die Datei verwendet werden soll (vgl. nachfolgende Abbildung).



Wählen Sie *synthesis* aus und deaktivieren Sie diesen Eintrag durch einen Klick auf den Pfeil nach links. Unter *Selected values* sollte nun nur noch der Eintrag *implementation* stehen. Bestätigen Sie mit Klick auf Ok.



5. Nun können Sie den VHDL-Code für das Filter-IP erstellen. Das IP soll die folgenden Eigenschaften haben:

- Das IP soll eine zweidimensionale Filterung (vgl. Vorlesungsabschnitt "Videoverarbeitung") mit einer Fenstergröße von 3x3 Pixeln durchführen.
- Die Koeffizienten sollen über ein AXIL-Interface programmierbar sein.
- Eingangs- und Ausgangspixel sind vorzeichenlose Werte mit einer Wortbreite von 8 bit.
- Die benötigten Zeilenspeicher (engl. *line memories*) werden mit Hilfe des vorgegebenen IPs *axis_linemem_single_master* realisiert. Hiermit können am Eingang des Filters drei vertikal benachbarte Pixel als 24 bit breiter Vektor zur Verfügung gestellt werden.

Die im Bild am weitesten oben liegenden Pixel werden in den Bits [23:16] übergeben. Das Pixel aus der mittleren Zeile befindet sich in den Bits [15:8], während das Pixel aus der untersten Zeile in den Bits [7:0] übergeben wird.

- Die Filterkoeffizienten sollen eine (per Konfigurationsdialog) wählbare Wortbreite im Bereich von 3 bis 8 bit besitzen. In jedem Fall werden die Koeffizienten als vorzeichenbehaftet betrachtet.
- Die Filterberechnung soll ohne Genauigkeitsverlust durchgeführt werden. Hierfür muss die Wortbreite der VHDL-Signale ausreichend groß gewählt werden.
- Das Ergebnis der Filterfunktion besitzt eine Wortbreite, die größer als 8 bit ist. Um das Ergebnis der Filterfunktion auf die Ausgabewortbreite anzupassen, werden zwei Schritte durchgeführt:
 - Division durch eine Zweierpotenz. Diese Operation lässt sich durch Rechtsschieben einfach und aufwandsarm implementieren. Die Anzahl der Bits, um die geschoben wird, soll durch ein AXIL-Interface wählbar sein und im Bereich von 0 bis 15 einstellbar sein.
 - Anschließend Begrenzung (Clipping) des Wertes auf den Bereich [0,255]. Negative Werte werden auf null gesetzt, während positive Werte, die größer als 255 sind, auf 255 gesetzt werden sollen.
- Wählen Sie die Belegung der Register wie folgt (nicht angegebene Bits haben keine Funktion):

Reg.-Adresse	Bedeutung
0	[7:0] - Koeffizient (-1,-1)
4	[7:0] - Koeffizient (-1,0)
8	[7:0] - Koeffizient (-1,1)
12	[7:0] - Koeffizient (0,-1)
16	[7:0] - Koeffizient (0,0)
20	[7:0] - Koeffizient (0,1)
24	[7:0] - Koeffizient (1,-1)
28	[7:0] - Koeffizient (1,0)
32	[7:0] - Koeffizient (1,1)
36	[3:0] - Stellenzahl für Rechtsschieben

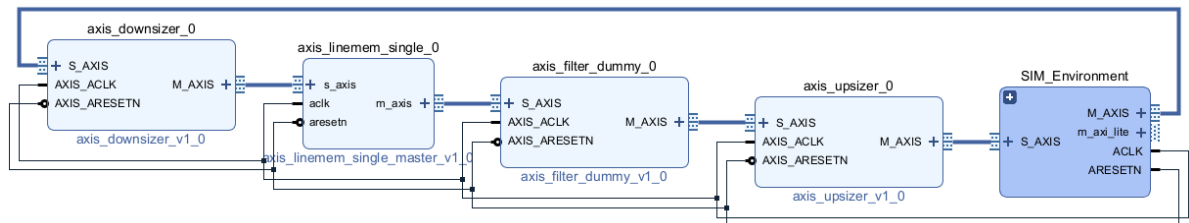
Für die Aufteilung der Filterfunktion in zwei Pipelinestufen ist es empfehlenswert, das Filter zeilen- und spaltenweise zu berechnen. So können zum Beispiel in der ersten Pipelinestufe zunächst die

(Zwischen-)Ergebnisse für die drei Filterzeilen bestimmt werden. In der zweiten Pipelinestufe erfolgt dann die Summation der Zwischenergebnisse und die Formatierung der Ausgangswerte.

Im Verzeichnis Codefragmente-3x3-Filter stehen weitere Codefragmente zur Verfügung, die Sie nur nutzen sollten, wenn Sie bei der Bearbeitung der Aufgabe auf Probleme stoßen. Versuchen Sie, ohne die Dateien in diesem Verzeichnis auszukommen. Beachten Sie bei Verwendung dieser Codeteile, dass die Dateien keine komplette Lösung der Aufgabe darstellen.

Simulation des Filters

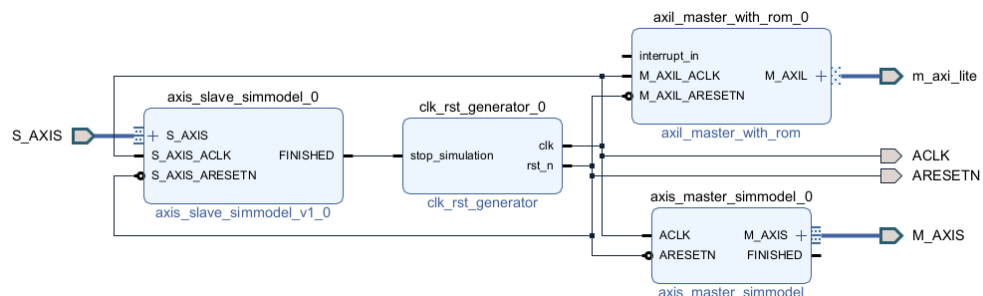
Das Block-Design für die Simulation für Ihr IP können Sie eigenständig aufsetzen oder Sie nutzen das vorgegebene TCL-Script *M6_SimulationSystem.tcl*, mit dem Sie das nachfolgend abgebildete System erstellen können. Denken Sie daran, vor dem Aufruf des Scripts die passenden IP-Repositories anzugeben.



Dieses System enthält eine "Dummy-Implementierung" des Filters (*axis_filter_dummy_0*), die keine Funktion besitzt. Die Eingangswerte werden unverändert weitergegeben. Neben dem Zeilenspeicher-IP befinden sich im System zwei weitere IPs. Das IP *axis_downsizer* reduziert die Wortbreite des Streaming-Kanals von 32 auf 8 bit, so dass nach diesem IP mit jedem "data beat" ein statt vier Pixel übergeben werden. Dies reduziert zwar den Datendurchsatz, vereinfacht aber die Implementierung des Filter-IPs. Das IP *axis_upsizer* generiert aus dem 8-Bit-Datenstrom des Filters einen Stream mit einer Wortbreite von 32 bit.

Der Block *SIM_Environment* enthält neben den bereits bekannten IPs *clk_rst_generator* und *axi_master_with_rom* die beiden AXIS-IP-Blöcke *axis_master_simmodel* und *axis_slave_simmodel*. Beide letztgenannten Simulations-IPs sind speziell für die Simulation von Video-IP entwickelt worden. Der Master kann Bilddateien im unkomprimierten BMP- oder YUV-Format lesen und die Bilddaten als AXIS-Videostream ausgeben. Der Slave hat die Aufgabe die Daten eines Videostreams zu empfangen und in einer Bilddatei abzulegen. Die Konfiguration der IPs erfolgt über die zugehörigen Konfigurationsdialoge. Neben der Angabe von Dateinamen und der Größe der Bilder, ist der Eintrag *Pixel Format* von besonderer Bedeutung. Die hier eingetragenen Zahlenwerte codieren das verwendete Format des Videostreams. Die Codierung des Videoformats folgt den Vorgaben der Firma AMD (Tabelle 1-4 im Dokument "AXI4-Stream Video IP and System Design Guide"), wobei die von AMD definierten Formate 0, 1, 2, 5, 6 und 12 unterstützt werden. Mit Hilfe des von der Simulationsumgebung genutzten Formats 13 werden Graubilder mit 8 bpp, gepackt in 32-Bit-Worte, vom Master ausgegeben beziehungsweise vom Slave als Eingangsformat erwartet. Die Struktur des Block *SIM_Environment* ist in der nachfolgenden Abbildung dargestellt.

Als Hilfe bei der Konfiguration des IPs können Sie das Dokument *IP-Konfig-M6.pdf* heranziehen.

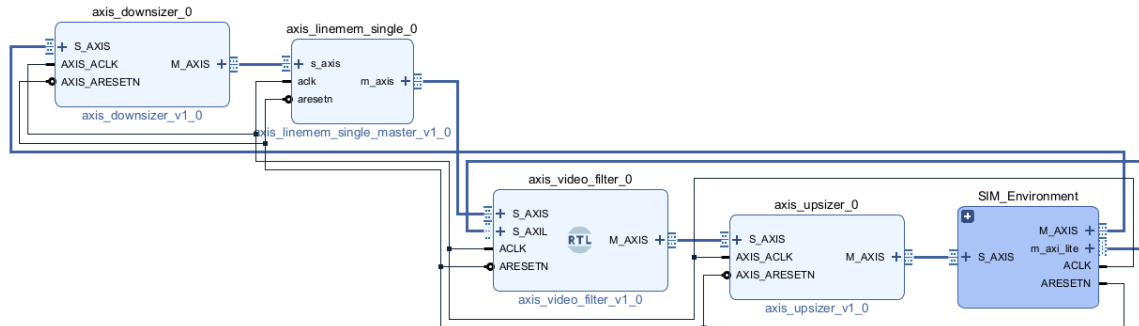


1. Generieren Sie den Design-Wrapper und kennzeichnen Sie ihn für die Simulation als Top-Level.
2. Stellen Sie sicher, dass sich die Dateien *Moewe-192x192.bmp* und *stimuli.mem* im Projektverzeichnis befinden.

- Führen Sie zunächst eine Simulation mit dem Dummy-Filter-IP durch. Bei Verwendung der Bilddatei *Moewe-192x192.bmp*¹ erhalten Sie nach ca. 3,2 ms Simulationszeit ein Ausgangsbild in der Datei *tst_out.bmp*. Überprüfen Sie, ob *tst_out.bmp* die monochrome Variante des Eingangsbildes ist. Sollte dies nicht der Fall sein, sollten Sie zunächst die Fehler in der Simulationsumgebung beheben.

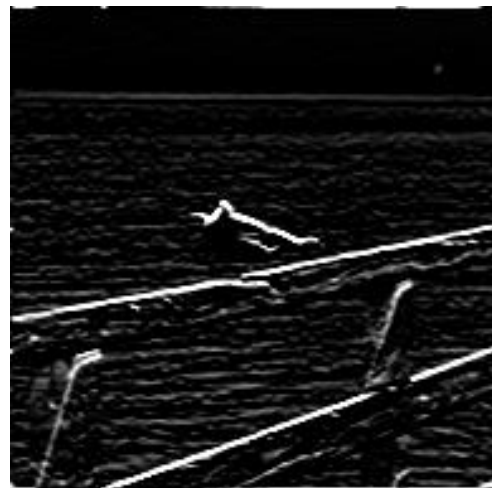
Hinweis: Bei der Erstellung der "Output Products" sollten Sie im Simulationsprojekt unter "Synthesis Options" die Variante "Global" wählen. Andernfalls versucht Vivado eine Out-Of-Context-Synthese für alle IP-Blöcke durchzuführen. Dies führt zu unnötigen Wartezeiten und misslingt für die Simulations-IPs, was störende Fehlermeldungen zur Folge hätte.

4. Ersetzen Sie das Modul Filter-Dummy durch Ihr Filter IP.



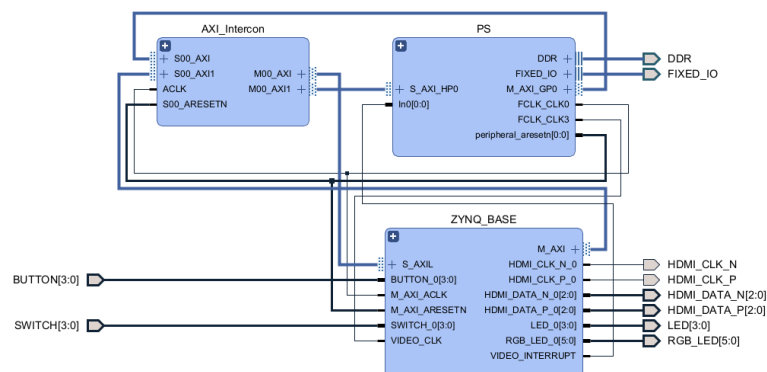
5. Mit der vorgegebenen Stimuli-Datei sollte Ihr IP einen Sobel-Filter zur Hervorhebung horizontaler Kanten implementieren. Simulieren Sie Ihr IP, überprüfen Sie das Ergebnis und beheben Sie eventuelle Fehler.

Das Eingangsbild und das erwartete Ausgangsbild sind nachfolgend dargestellt:



Hardware des Filtersystems

1. Erstellen Sie in dem existierenden Projekt ein neues Block-Design und führen Sie das TCL-Skript *Zybo-Z7-20_VideoOut.tcl* aus, um ein Basissystem zu erhalten.

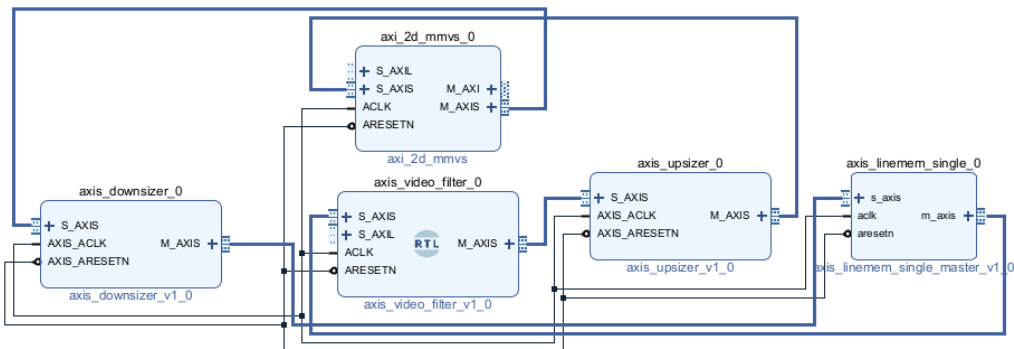


¹ In der vorgegebenen Simulationsumgebung ist die oberste Projektordner-Ebene für die Simulationsdateien „Moewe-192x192.bmp“, „tst_out.bmp“ und „stimuli.stm“ voreingestellt.

2. Ergänzen Sie das System mit den folgenden IPs:

- *axis_downsizer*
- *axis_linemem_single_master*
- *axis_video_filter*
- *axis_upsizer*
- *axi_2d_mm2vs*

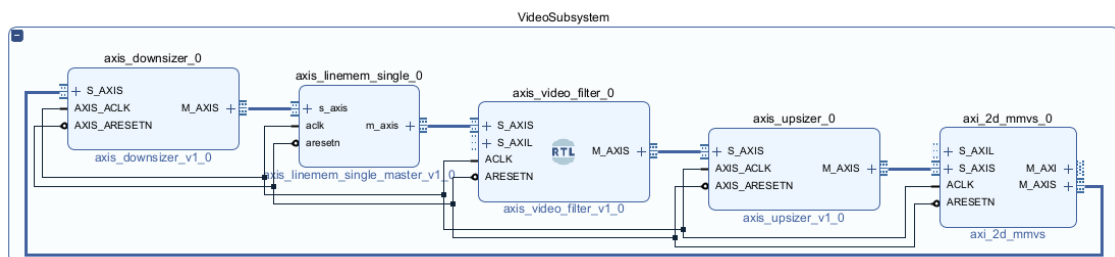
Einen möglichen Aufbau des Video-Subsystems zeigt die nachfolgende Abbildung:



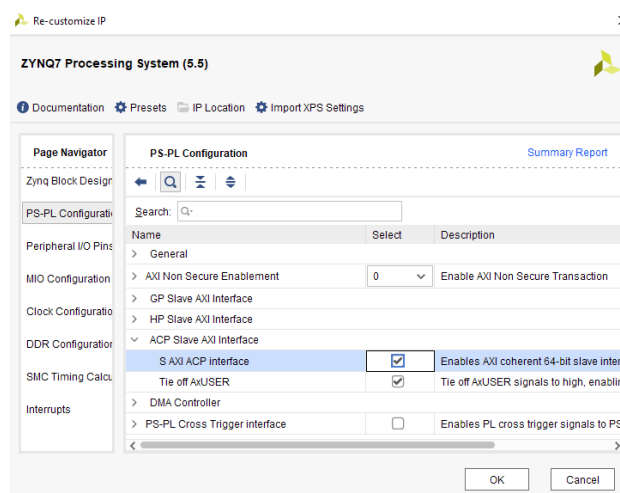
3. Konfigurieren Sie die IPs:

- Up- und Downsizer: Size Factor 4, Width Out 8
- Linememory: Data Width 8
- MMVS: Unter dem Reiter „General“ die Interruptausgänge aktivieren

4. Gruppieren Sie die neu hinzugefügten IPs und verbinden Sie die Takt- und Reset-Eingänge.

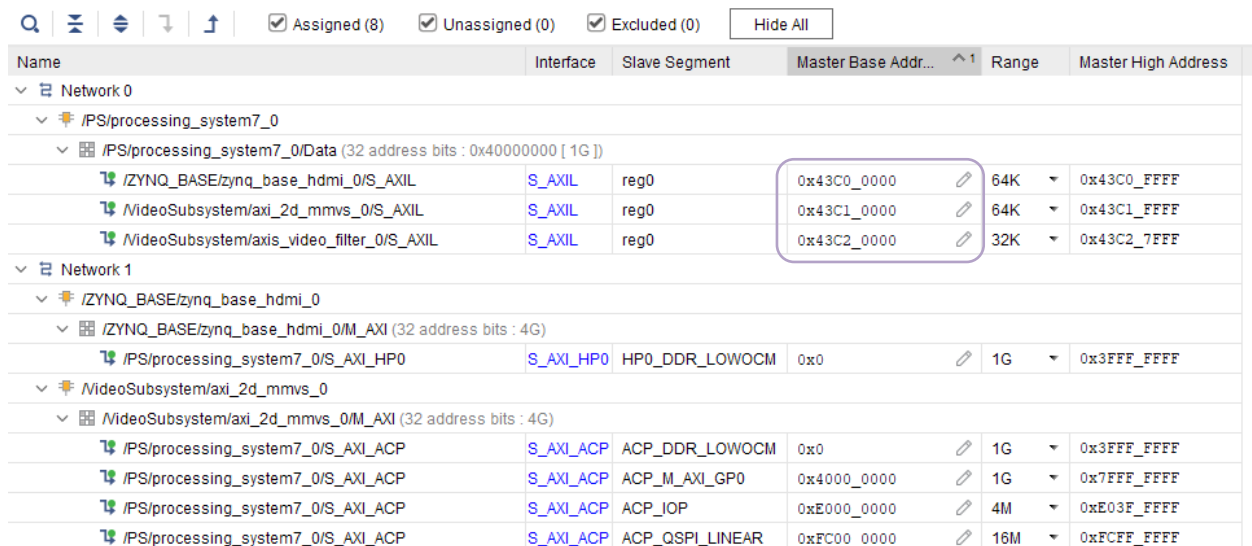


5. Aktivieren Sie den ACP-Port des Processing Systems (im Block PS). **Verzichten** Sie anschließend auf die von Vivado angebotene Connection Automation. Verbinden Sie den Takt-Eingang des ACP-Ports mit dem Taktsignal.



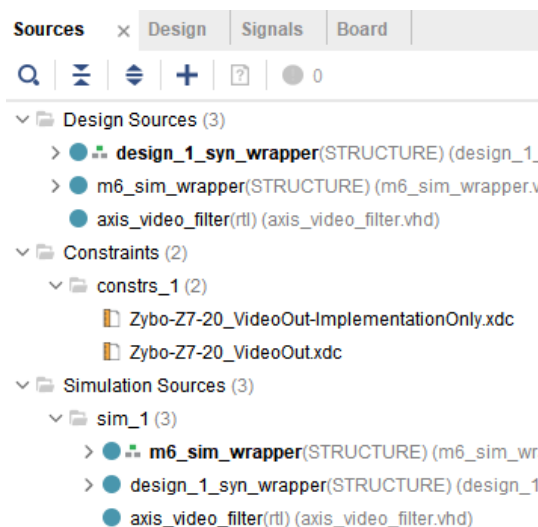
6. Konfigurieren Sie das AXI-Interconnect-IP für die AXIL-Verbindung im Block AXI_Intercon so, dass es drei Master-Ports besitzt. Verbinden Sie die neuen Master-Ports mit den AXIL-Slave-Ports des MMVS-IPs und des Filter-IPs. Verbinden Sie beim AXI-Interconnect-IP auch die neu eingefügten Takt- und Resetanschlüsse.

7. Das AXI-Interconnect-IP erhält weitere Takt- und Reset-Eingänge. Verbinden Sie diese mit dem Takt- und mit dem Resetsignal.
8. Fügen Sie im Block AXI_Intercon ein weiteres AXI-Interconnect-IP hinzu und konfigurieren Sie es so, dass es einen Master- und einen Slave-Port besitzt. Verbinden Sie die Takt- und Reset-Eingänge mit dem Takt- und mit dem Resetsignal.
9. Verbinden Sie den Slave-Port des neuen Interconnect-IPs mit dem Master-Port des MMVS-IPs. Den Master-Port des Interconnect-IPs verbinden Sie mit dem ACP-Port des Processing Systems.
10. Cross-Check: Sie sind (vermutlich) mit der Verdrahtung fertig, wenn Vivado keine „Connection Automation“ mehr anbietet.
11. Konfigurieren Sie die Adressen mit Hilfe des Address-Editors. Eine mögliche Konfiguration zeigt die nachfolgende Abbildung.



Name	Interface	Slave Segment	Master Base Addr...	Range	Master High Address
Network 0					
/PS/processing_system7_0					
/PS/processing_system7_0/Data (32 address bits : 0x40000000 [1G])					
/ZYNQ_BASE/zynq_base_hdmi_0/S_AXIL	S_AXIL	reg0	0x43C0_0000	64K	0x43C0_FFFF
/VideoSubsystem/axi_2d_mmvs_0/S_AXIL	S_AXIL	reg0	0x43C1_0000	64K	0x43C1_FFFF
/VideoSubsystem/axis_video_filter_0/S_AXIL	S_AXIL	reg0	0x43C2_0000	32K	0x43C2_7FFF
Network 1					
/ZYNQ_BASE/zynq_base_hdmi_0					
/ZYNQ_BASE/zynq_base_hdmi_0/M_AXI (32 address bits : 4G)					
/PS/processing_system7_0/S_AXI_HP0	S_AXI_HP0	HP0_DDR_LOWOCM	0x0	1G	0x3FFF_FFFF
/VideoSubsystem/axi_2d_mmvs_0					
/VideoSubsystem/axi_2d_mmvs_0/M_AXI (32 address bits : 4G)					
/PS/processing_system7_0/S_AXI_ACP	S_AXI_ACP	ACP_DDR_LOWOCM	0x0	1G	0x3FFF_FFFF
/PS/processing_system7_0/S_AXI_ACP	S_AXI_ACP	ACP_M_AXI_GP0	0x4000_0000	1G	0x7FFF_FFFF
/PS/processing_system7_0/S_AXI_ACP	S_AXI_ACP	ACP_IOP	0xE000_0000	4M	0xE03F_FFFF
/PS/processing_system7_0/S_AXI_ACP	S_AXI_ACP	ACP_QSPI_LINEAR	0xFC00_0000	16M	0xFCFF_FFFF

12. Die Größe der FIFOs des MMVS-IPs auf 2^9 Einträge voreingestellt, Konfigurationsparameter *FIFO Address Width* = 9. Erhöhen Sie diesen Parameter bei beiden Subblöcken (MM2VS und VS2MM) auf 12. Auf diese Weise können auch größere Latenzzeiten während des Speicherzugriffs überbrückt werden, ohne dass die zugehörigen AXI-Streams angehalten werden müssen.
13. Verbinden Sie die Interruptleitung **VS2MM_INTERRUPT** des IPs **AXI_2D_MMVS** mit einem weiteren Interrupteingang des Processing Systems. Bedenken Sie, dass im Devicetree eine Zuordnung zwischen IP-Basisadresse und der Interruptleitung vorgenommen wurde (vgl. Vorlesung). Wenn Sie diese Zuordnung berücksichtigen, ist die Modifikation des Devicetrees nicht erforderlich.
14. Wählen Sie in den Bitstream-Settings aus, dass auch eine Datei im bin-Format erzeugt wird.
15. Generieren Sie den Design-Wrapper und kennzeichnen Sie ihn für die Synthese (also unter Design Sources) als Top-Level.



Sources	Design	Signals	Board
Design Sources (3)			
design_1_syn_wrapper(STRUCTURE) (design_1_			
m6_sim_wrapper(STRUCTURE) (m6_sim_wrapper.v			
axis_video_filter(rtl) (axis_video_filter.vhd)			
Constraints (2)			
constrs_1 (2)			
Zybo-Z7-20_VideoOut-ImplementationOnly.xdc			
Zybo-Z7-20_VideoOut.xdc			
Simulation Sources (3)			
sim_1 (3)			
m6_sim_wrapper(STRUCTURE) (m6_sim_wr			
design_1_syn_wrapper(STRUCTURE) (design_1			
axis_video_filter(rtl) (axis_video_filter.vhd)			

16. Nachdem Sie die Validierung des Block Designs ("Validate Design") fehlerfrei durchgeführt haben, können Sie die Synthese durchführen. Die vier Fehlermeldungen bezüglich des SDRAM-Timings können Sie ignorieren.
17. Schauen Sie sich nach dem Erzeugen des Bitstreams die Project Summary an. Sollten dort kritische Fehler bezüglich des Timings Ihres Systems angezeigt werden, liegt dies vermutlich an Ihrem Filter-IP. Mit Unterstützung durch Ihre Betreuer können Sie die Ursache eingekreisen und den Fehler beheben.
Timingfehler im Bereich von wenigen Picosekunden (< 100 ps) sollte Ihnen kein Kopfzerbrechen bereiten. Die Synthese ist meist pessimistischer als das Silizium. 😊

Sollten Sie bei der Erstellung des Systems auf unüberwindliche Probleme stoßen, können Sie von Ihren Betreuern ein TCL-Skript anfordern, welches das System automatisch erstellt und korrekt konfiguriert.

Test des Systems unter Linux

Testen Sie das System unter Linux: Einen Vorschlag für eine Implementierung der Software finden Sie unter *milestone_6_template.cpp*. Dieser Code steht auch auf der virtuellen Maschine als Eclipse-Projekt *Milestone_6* zur Verfügung.

Beachten Sie, dass der Code gegebenenfalls z.B. im Hinblick auf die verwendeten UIO-Geräte angepasst werden muss. Bei der Verwendung der Vorlage sollten Sie die Funktion des Codes nachvollziehen und eventuelle Unklarheiten mit Ihren Betreuern klären.

Testen Sie auch andere Filterkoeffizienten.

Einige Beispiele finden Sie unter <https://de.wikipedia.org/wiki/Faltungsmatrix> sowie den in diesem Artikel angegebenen weiterführenden Links.

Abgabe

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

Fügen Sie diesem Dokument auch einen Screenshot Ihrer Linux-Umgebung hinzu, in dem der Test Ihres IPs erkennbar ist.

Erstellen Sie eine ZIP-Datei, in der Sie das o.g. PDF-Dokument, den VHDL-Code Ihres Filter-IPs und Ihren finalen C++-Code einfügen.