Audiofrequenzanalyse

Niklas Schelten, Jannes Potthoff, Steffen Büschking

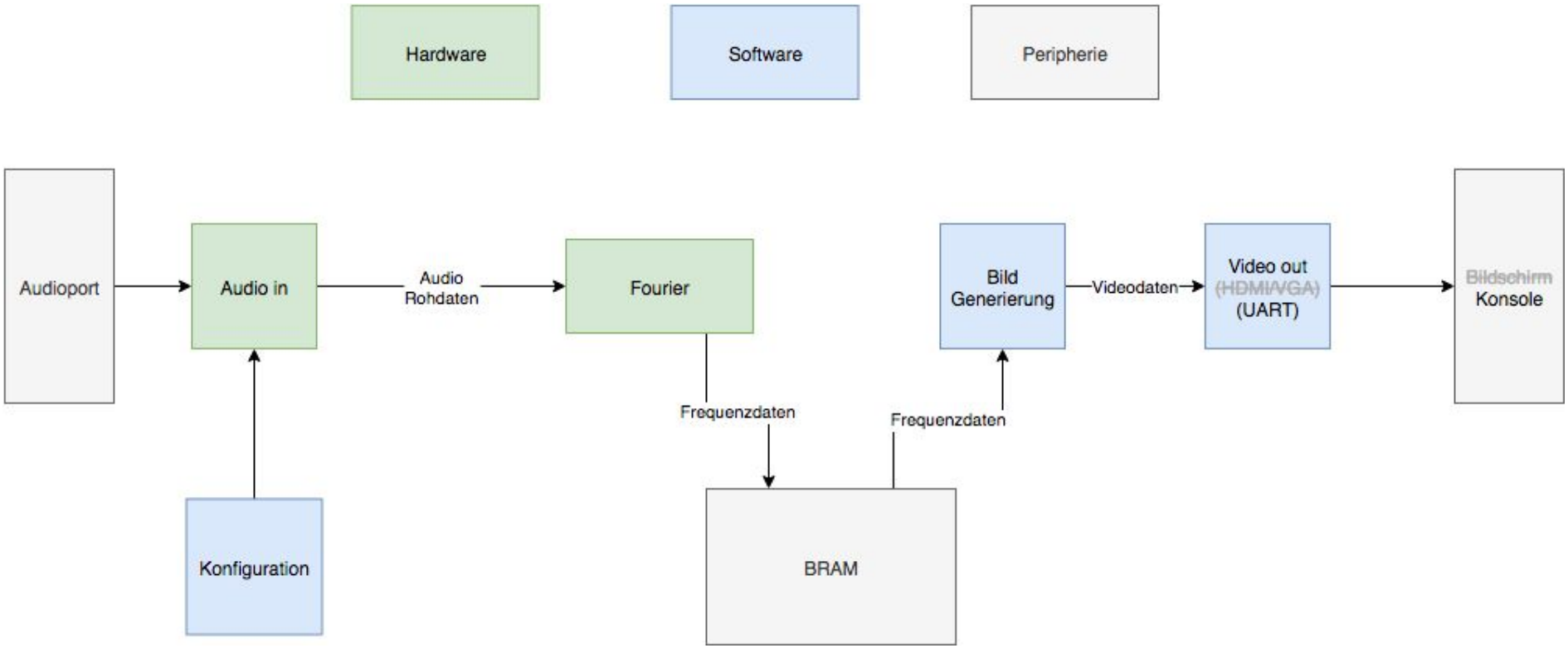
TU Berlin Fakultät IV - Embedded Systems Architecture

13. Februar 2019

Outline

1. Audio IN
2. Fouriertransformation
3. Video OUT
4. Bildgenerierung

Blockschaltbild



Outline

1. Audio IN
   - ADAU 1761
   - Kommunikation mit dem Chip
   - Empfangen der Audio Daten
2. Fouriertransformation
   - Theoretische Herangehensweise
   - FPGA Implementierung
   - Aktueller Stand
3. Video OUT
   - ADV7511 Chip
   - Kommunikation via I2C-Protokoll
   - Implementierung des HDMI-Treibers
4. Bildgenerierung

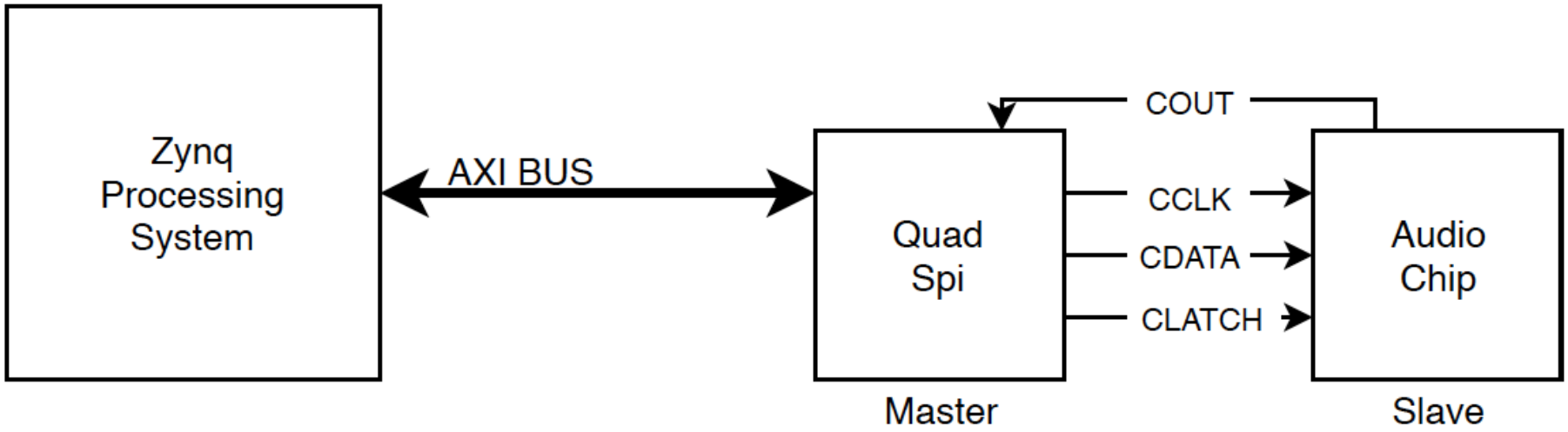
Allgemeiner Überblick

Technische Universität Berlin

- Stereo Audio Codec
- 24 bit AD-/DA-Umsetzer
- I2C und SPI Schnittstellen zur Konfiguration
- Digital audio serial data I/O
- Smartphones, Digitalkameras, Portable media players, AES Bachelor Projekt

SPI Protokoll

- Serial Peripheral Interface:
  - 4 - Wire Interface
  - synchrone, serielle Datenübertragung
  - Master Slave Prinzip
  - ADAU 1761 SPI PORT max. Frequenz: 10 MHz!!

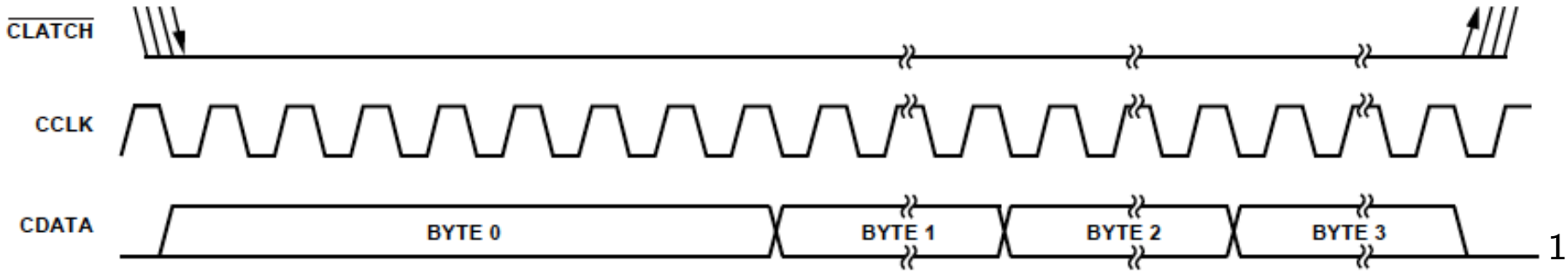


SPI Bit Timing

Technische Universität Berlin

Byte 0	chip_addr[6:0]	Befehl	Zugriffsart
	0b0000000	1	lesend
	0b0000000	0	schreibend
Byte 1	subaddr[15:8] = 0x40		
Byte 2	subaddr[7:0] = Registerabhängig		
Byte ...	Daten		

Achtung: Quad Spi empfängt nach jedem Schreiben Daten.



1

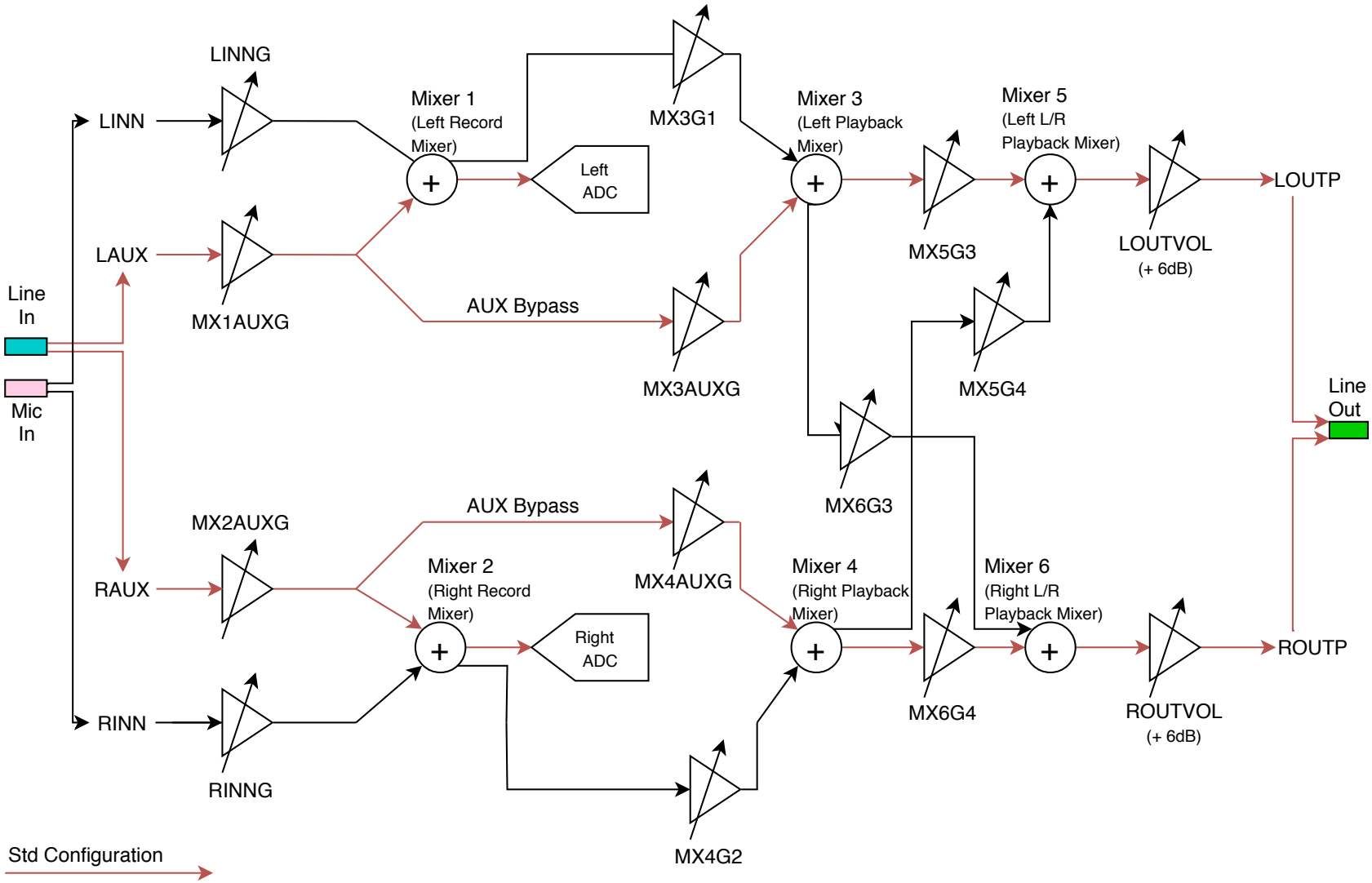
[1] https://www.analog.com/media/en/technical-documentation/data-sheets/adau1761.pdf

Konfiguration

Technische Universität Berlin

- Startreihenfolge:
  1. Spannung einschalten
     => Nur die Register R0 und R1 sind zur Clk Konfiguration verfügbar
  2. PLL einschalten (interne Clk Generierung)
     => Warten auf PLL Lock Bit (polling)
  3. Core Clock einschalten
     => Ab jetzt sind alle Register verfügbar.
  4. Register setzen

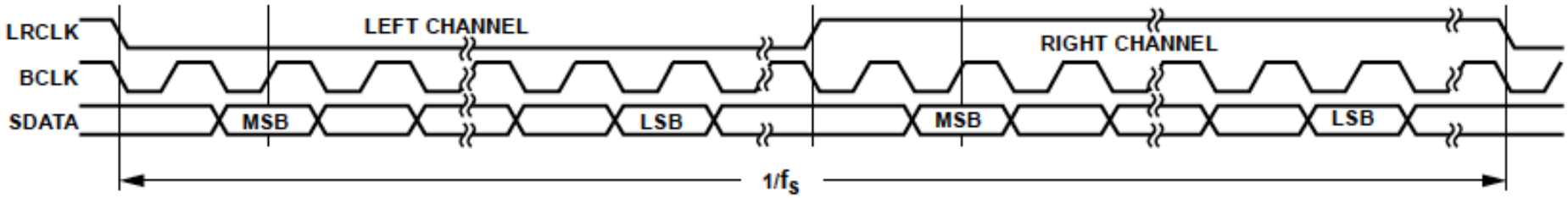
Konfiguration



I^2S Bit Timing

Technische Universität Berlin

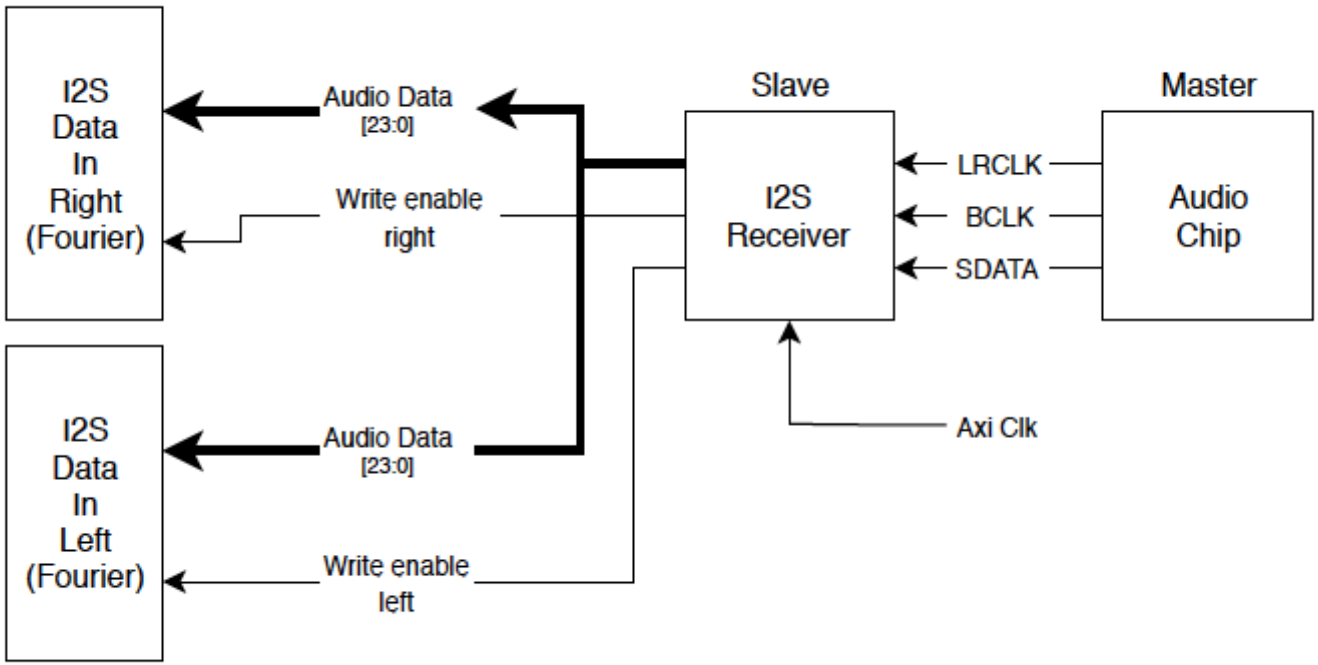
- 64 BCLK Cycles pro Audio frame
- 1 BCLK Cycle Data Delay
- 24 Bit Daten pro Channel



I^2S Protokoll

Technische Universität Berlin

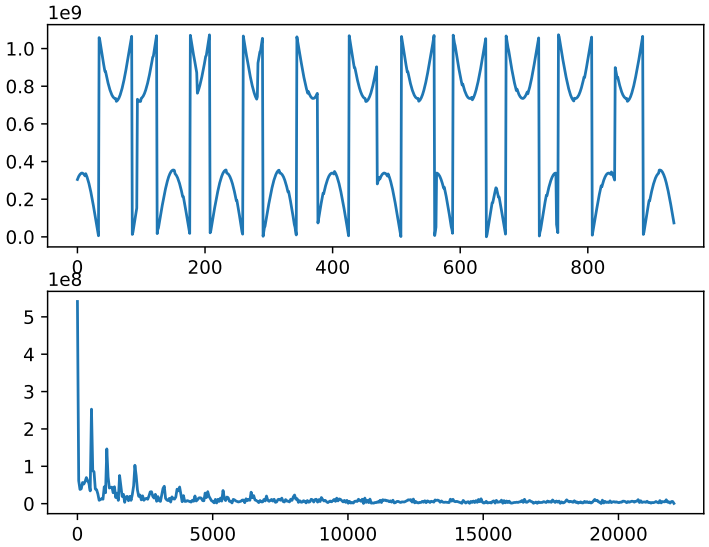
- Übertragung von Audiodaten
- Festes Übertragungsfenster => Bits per frame sind statisch
- Master Slave Prinzip
- Taktflankensteuerung durch BCLK nicht möglich!!



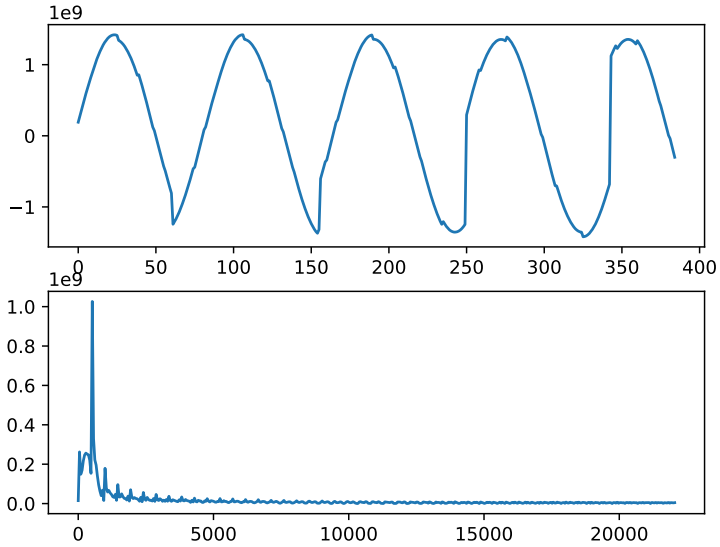
I2S Datenformat

Technische Universität Berlin

- 24 Bit 2-Komplement Zahlen!



Vorzeichenlose Interpretation



2-Komplement Interpretation

Erweiterungen

Technische Universität Berlin

- Entstörungsmöglichkeiten des Audiochips besser nutzen
- Darstellung des Eingangssigals der Transformation
- Möglichkeit zur Mikrofoneingabe schaffen

Outline

1. Audio IN
   - ADAU 1761
   - Kommunikation mit dem Chip
   - Empfangen der Audio Daten
2. Fouriertransformation
   - Theoretische Herangehensweise
   - FPGA Implementierung
   - Aktueller Stand
3. Video OUT
   - ADV7511 Chip
   - Kommunikation via I2C-Protokoll
   - Implementierung des HDMI-Treibers
4. Bildgenerierung

Fouriertransformation

Technische Universität Berlin

- Diskrete Fouriertransformation (DFT):

$$X_k = \sum_{t=0}^{N-1} x_t \cdot e^{-j \cdot \frac{2\pi k}{N} \cdot t}$$

$$= \sum_{t=0}^{N-1} x_t \cdot \left(\cos \frac{2\pi kt}{N} - j \cdot \sin \frac{2\pi kt}{N} \right)$$

Fouriertransformation

Technische Universität Berlin

- Diskrete Fouriertransformation (DFT):

$$X_k = \sum_{t=0}^{N-1} x_t \cdot e^{-j \cdot \frac{2\pi k}{N} \cdot t}$$
$$= \sum_{t=0}^{N-1} x_t \cdot \left(\cos \frac{2\pi kt}{N} - j \cdot \sin \frac{2\pi kt}{N} \right)$$

- Sliding DFT[2]:

$$X_{k,t} = (X_{k,t-1} + x_t - x_{t-N}) \cdot e^{-\frac{j2\pi k}{N}}$$

[2] https://www.comm.utoronto.ca/~dimitris/ece431/slidingdft.pdf

Rationale Zahlen

- Keine Floatingpoint-DSPs

$\rightarrow$ Fixedpoint-Arithmetik

Zweierkomplement			Positive Nachkommastellen		
$\ldots$	2^1	2^0	2^{-1}	2^{-2}	$\ldots$

Rationale Zahlen

- Keine Floatingpoint-DSPs

$\rightarrow$ Fixedpoint-Arithmetik

Zweierkomplement			Positive Nachkommastellen		
$\ldots$	2^1	2^0	2^{-1}	2^{-2}	$\ldots$

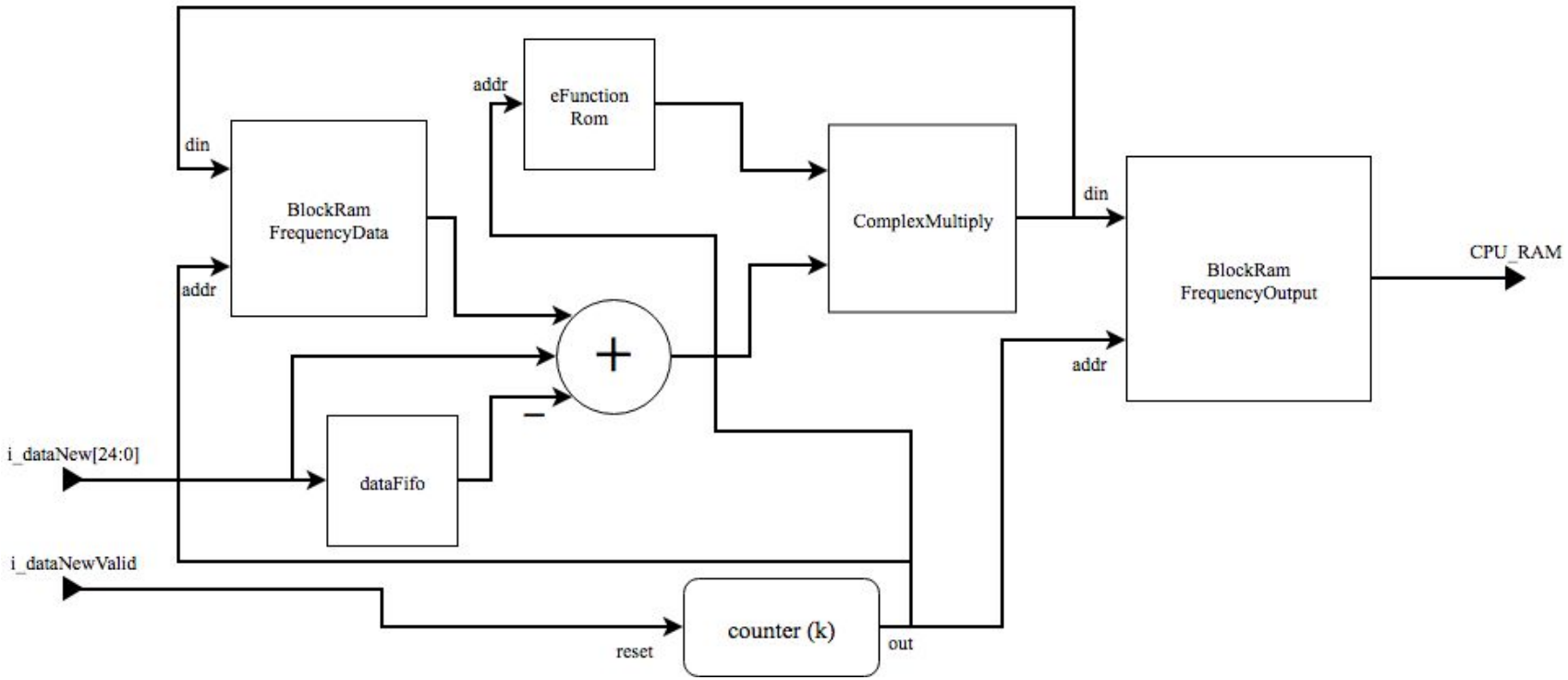
- Wie viele Vorkomma- und Nachkommastellen?
  - 25×18 DSP
  - Komplexe e-Funktion und Input $\in [-1; 1]$

  $\rightarrow$ Maximal erreichbarer Wert für X_k: N
  - $N = 512 \Rightarrow$ Q10.15 und Q2.16

VHDL Design

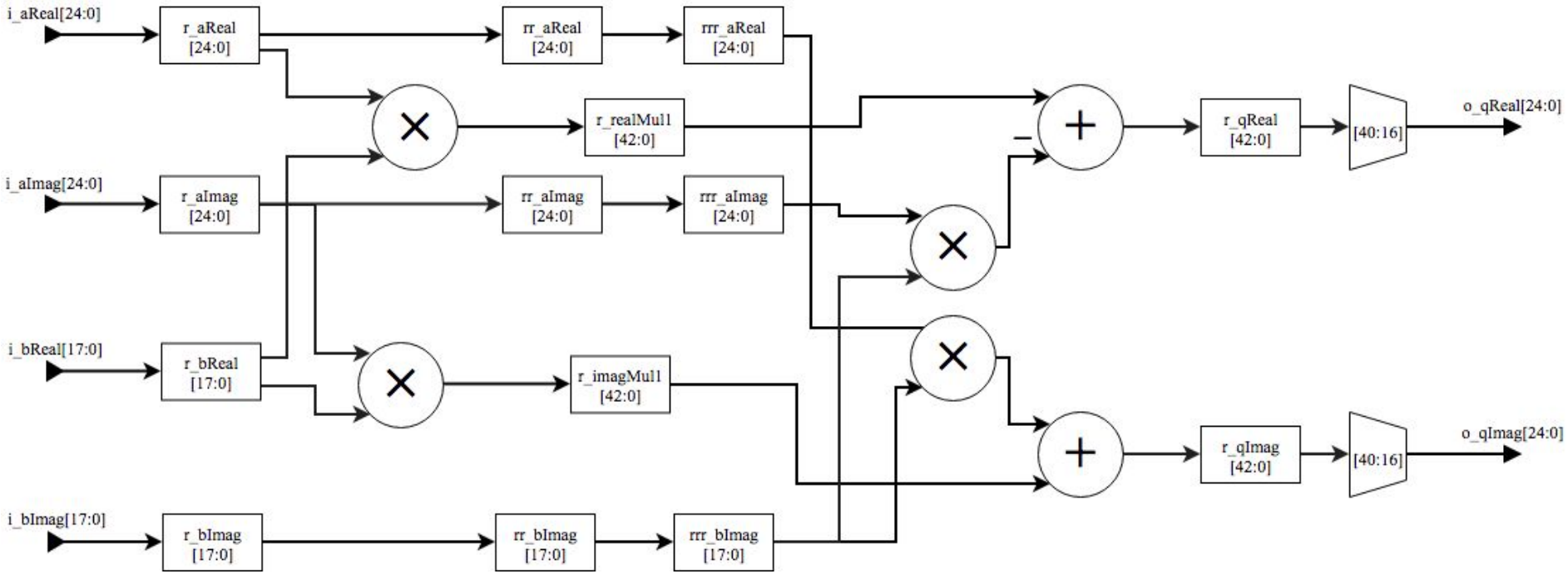
Technische Universität Berlin

$$X_{k,t} = e^{-\frac{j2\pi k}{N}}(X_{k,t-1} + x_t - x_{t-N})$$

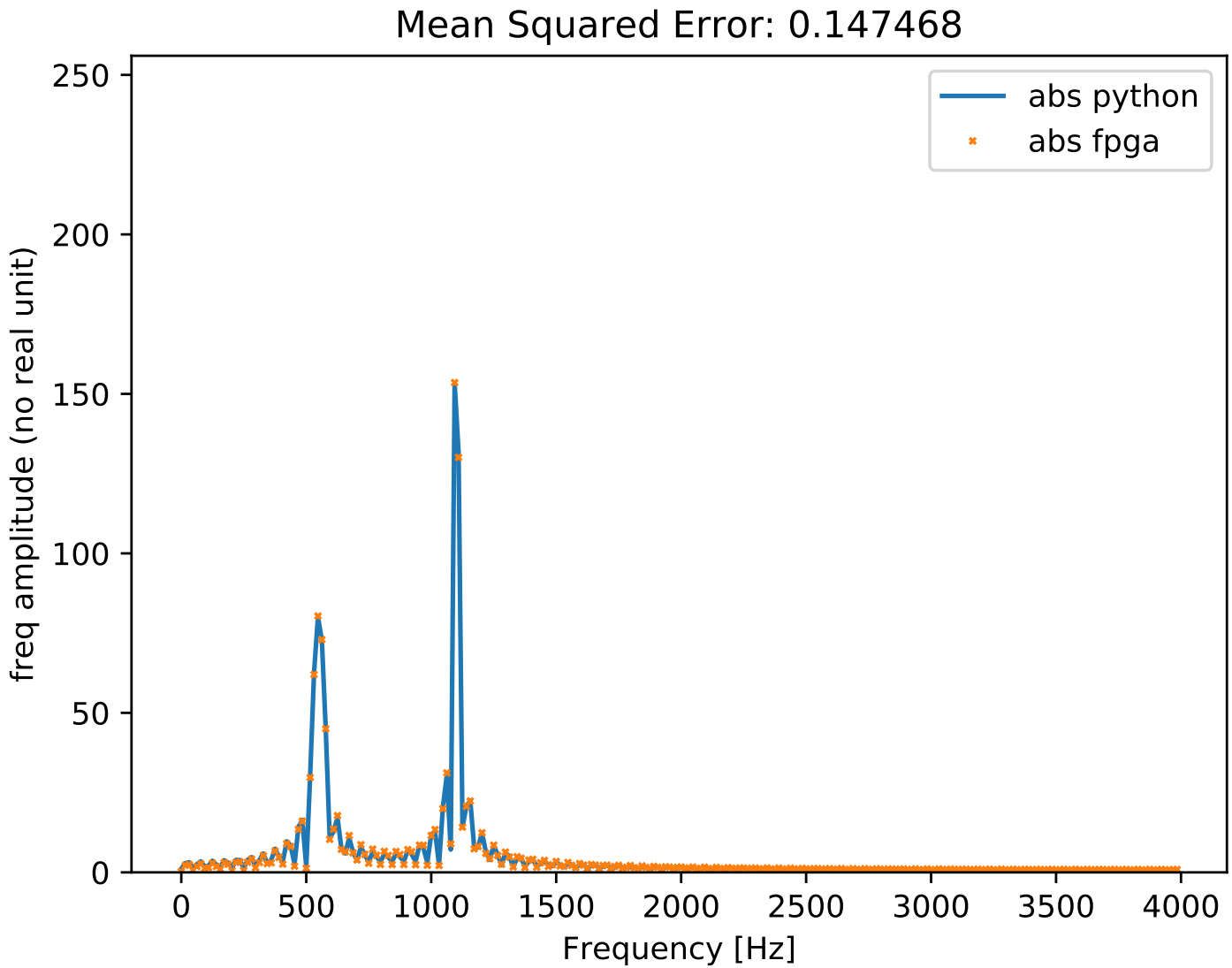


Komplexe Multiplikation

$$\mathrm{Re}\{q\} = \mathrm{Re}\{a\} \cdot \mathrm{Re}\{b\} - \mathrm{Im}\{a\} \cdot \mathrm{Im}\{b\}$$
$$\mathrm{Im}\{q\} = \mathrm{Im}\{a\} \cdot \mathrm{Re}\{b\} + \mathrm{Re}\{a\} \cdot \mathrm{Im}\{b\}$$



Vergleich Python vs. VHDL/SV I



Vergleich Python vs. VHDL/SV II

Technische Universität Berlin

- Audio 44 100 kHz $\rightarrow$ 22.68 µs pro Sample

Python

- 1 Sample in 1235 µs[3]
- 1 Sample in 713 µs[4]

[3] Intel i5-3570 @ 3.4GHz
[4] Intel Xeon E5-2687W v3 @ 3.1GHz

Vergleich Python vs. VHDL/SV II

- Audio 44 100 kHz $\rightarrow$ 22.68 µs pro Sample

Python

- 1 Sample in 1235 µs[3]
- 1 Sample in 713 µs[4]

VHDL

- 1 Sample in 267 Takte mit >100MHz

$\rightarrow$ 1 Sample in 2.67 µs

[3] Intel i5-3570 @ 3.4GHz
[4] Intel Xeon E5-2687W v3 @ 3.1GHz

Erweiterungsmöglichkeiten

Technische Universität Berlin

- Größeres $N \rightarrow$ mehr Bins und höhere Frequenzauflösung
  - Takte pro Sample: $N + 11 \rightarrow$ 8-fache Bin Anzahl kein Problem

Erweiterungsmöglichkeiten

Technische Universität Berlin

- Größeres $N \rightarrow$ mehr Bins und höhere Frequenzauflösung
  - Takte pro Sample: $N + 11 \rightarrow$ 8-fache Bin Anzahl kein Problem
- Höher aufgelöster Input, aktuell 16bit, ADC liefert 24bit
  - Schwieriger, weil 25×18 DSPs
  - Mehrere DSPs für eine Multiplikation, mit IP Generator möglich
  - Eventuell Performanceprobleme

Outline

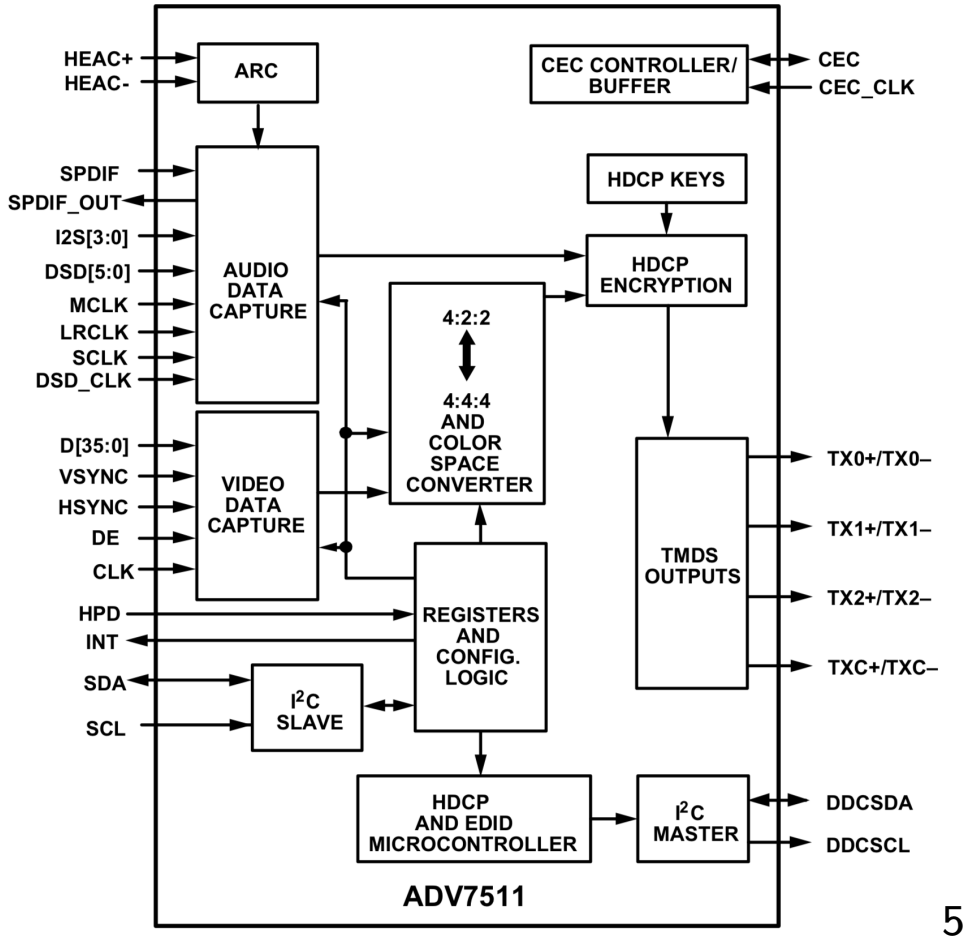
1. Audio IN
   - ADAU 1761
   - Kommunikation mit dem Chip
   - Empfangen der Audio Daten
2. Fouriertransformation
   - Theoretische Herangehensweise
   - FPGA Implementierung
   - Aktueller Stand
3. Video OUT
   - ADV7511 Chip
   - Kommunikation via I2C-Protokoll
   - Implementierung des HDMI-Treibers
4. Bildgenerierung

Allgemeine Spezifikationen

Technische Universität Berlin

- kompatibel mit HDMI 1.4, DVI 1.0
- Farbtiefe: 8, 10 oder 12 Bit
- Farbkodierung: RGB oder YCrCb (4:4:4 und 4:2:2)
- HSYNC und VSYNC (separat oder eingebettet)
- kompatibel mit 1080p 60Hz
- Ausnahmen auf Zedboard:
  - 4:2:2 YCrCb mit 8 Bit Farbtiefe
  - Begründung: 20 Pins an Masse angeschlossen
- Audioeingabe und Audio Return Channel
- Konfiguration durch I2C Schnittstelle

Blockschaltbild ADV7511



[5]

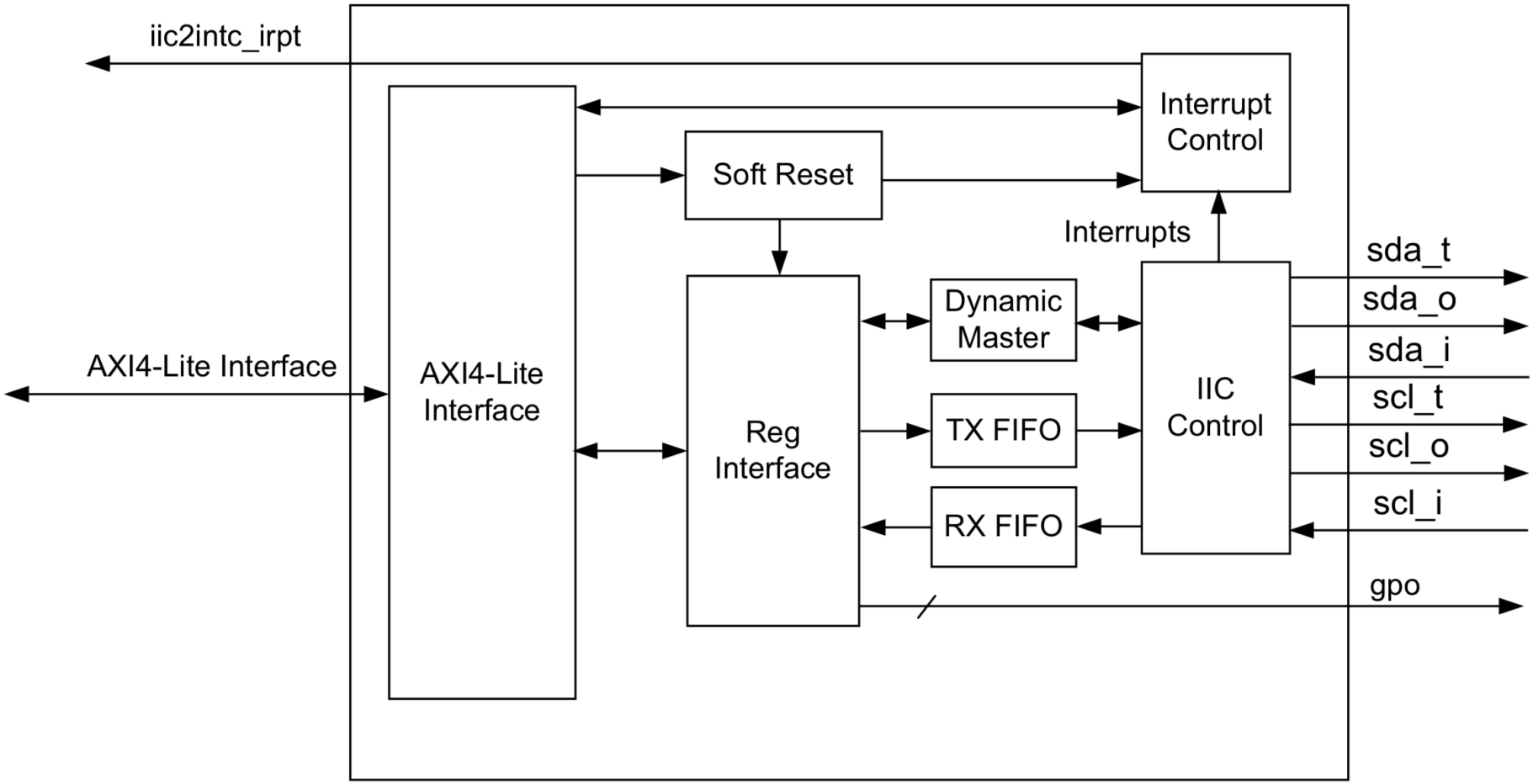
[5] https://www.analog.com/media/en/technical-documentation/user-guides/ADV7511_Hardware_Users_Guide.pdf

Versuch 1: AXI IIC

Technische Universität Berlin

- Verwendung vom AXI IIC LogicCORE IP
- Abstraktion des I2C-Protokolls durch Registerzugriffe
- Registerzugriff mittels AXI4-Lite Interface (Memory Mapped I/O)
- Probleme während des Debuggen:
  - MSMS Bit wird gesetzt -> Indikator für erfolgreiches Start Signal
  - Bus bleibt trotzdem idle (BB Bit auf 0)
  - TX_FIFO wird nicht leer -> Dynamische Logik blockiert weitere Eingaben in FIFO

Versuch 1: AXI IIC



6

[6] https://japan.xilinx.com/support/documentation/ip_documentation/axi_iic/v2_0/pg090-axi-iic.pdf

Versuch 2: Eigener I2C-Master

Technische Universität Berlin

- Entwicklung eines I2C-Master in VHDL
- Abstraktion der Schreibe- und Leseoperationen auf dem FPGA
- Steuerung der Operationen ebenfalls durch Memory Mapped I/O
- Simulation funktioniert nach dem I2C-Protokoll
- Nicht synthetisierbar wegen Probleme mit bidirektionalen Ports (INOUT)

HDMI Treiber

Technische Universität Berlin

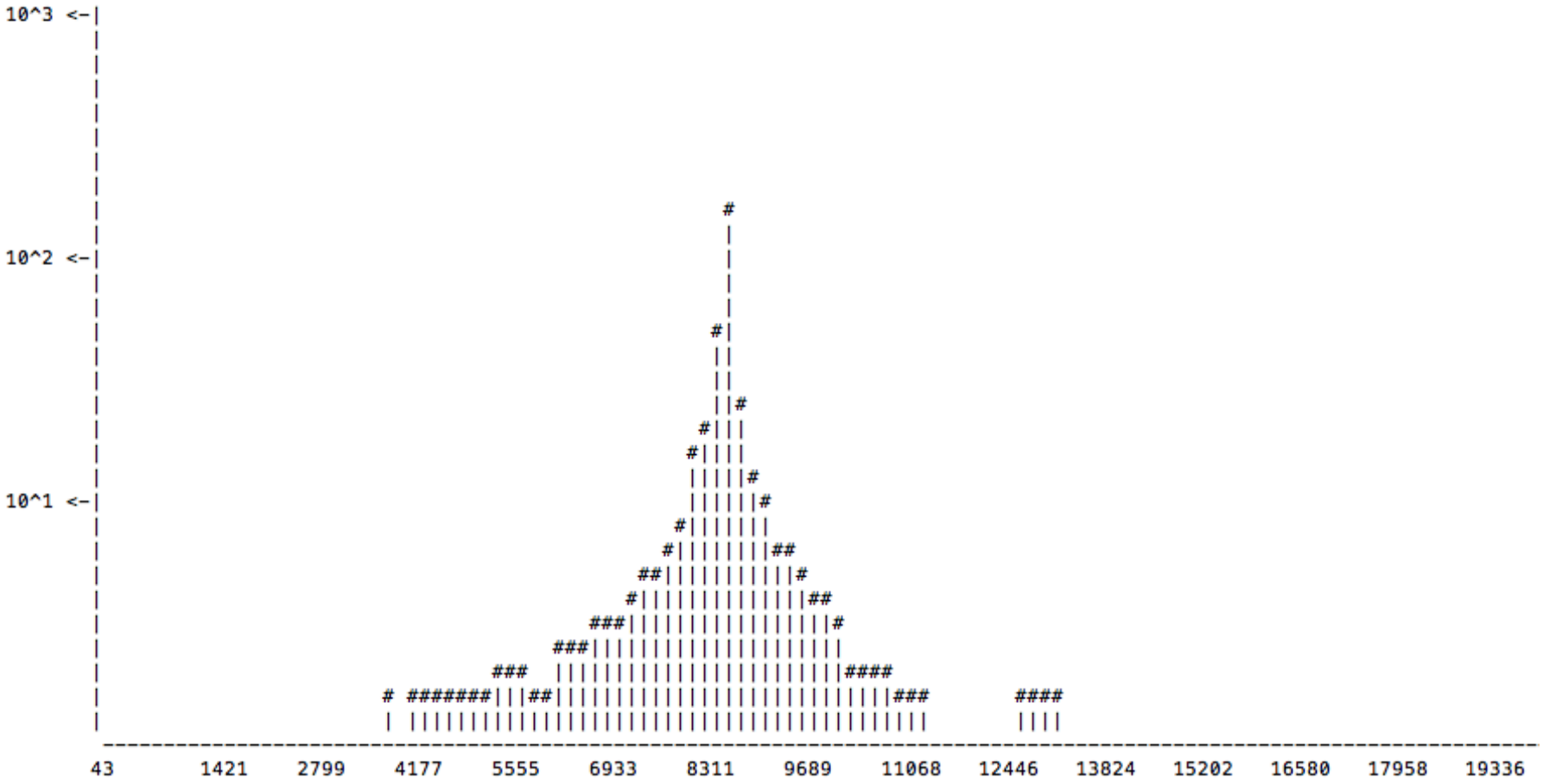
- Implementierung auf Softwareebene mittels C
- Konfigurationen für:
  - Video Input Format (4:2:2 YCrCb mit 8 Bit Farbtiefe)
  - Video Output Format (4:4:4 RGB mit 8 Bit Farbtiefe)
  - Eingebettete HSYNC, VSYNC und DE Signale für 1080p 60Hz
  - Ausschalten der Audiofunktion => keine Störgeräusche
- Nutzung des I2C-Protokolls zur Kommunikation auf FPGA
- wenig Abstraktion vorhanden -> Fokus lag auf Debugging beim I2C-Protokoll

Outline

1. Audio IN
   - ADAU 1761
   - Kommunikation mit dem Chip
   - Empfangen der Audio Daten
2. Fouriertransformation
   - Theoretische Herangehensweise
   - FPGA Implementierung
   - Aktueller Stand
3. Video OUT
   - ADV7511 Chip
   - Kommunikation via I2C-Protokoll
   - Implementierung des HDMI-Treibers
4. Bildgenerierung

The BS-Praktikum way

- Uart und Ascii art



Warum Messtechnik Labor doch nützlich ist

Technische Universität Berlin

- Python liveplot
  - Übertragung von Audio Menü und Transformationsdaten über Uart
  - Nur Menü anzeigen
  - Transformationsdaten in eine Textdatei schreiben

Anhang I - System Verilog Testbench

Technische Universität Berlin

```
1   module testbench_complex_mul;
2     // clock gen
3     always #(10/2) clk = ~clk;
4     initial begin
5     clk = 1;
6     end
7     // verification
8     initial begin
9       reset = 1;
10      repeat(5) begin
11        @(negedge clk);
12      end
13      reset = 0;
14      @(negedge clk);
15      $display("[CM Testbench] Starting ComplexMultiply tests. @ %01t", $time);
16      for (i = 0; i < MAX_TESTS + reg_stages; i++) begin
17        if (dut_cm_q_ref[i - reg_stages] != dut_cm_q) begin
18          $fatal(1, [... printing false bits ...]);
19        end else begin
20          $display("calculation correct @ %01t", $time);
21        end
22      end
23      $display("[CM Testbench] ComplexMultiply tests successful. @ %01t", $time);
24      $stop;
25    end
26  endmodule
```