Fachhochschule Nordwestschweiz
Hochschule für Technik

AOS Technologies AG

1.5 GPixel/s komprimieren

KTI-Projekt

Hochgeschwindigkeitskameras mit integrierter Videokompression

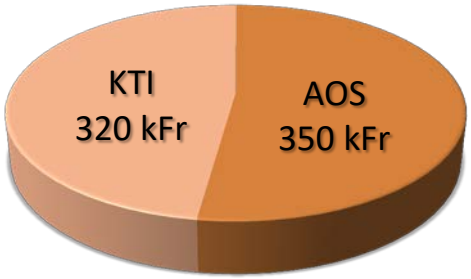
IMVS: Christoph Stamm

IME: Michael Pichler, Dino Zardet, Nils Frey, Marcel Baier

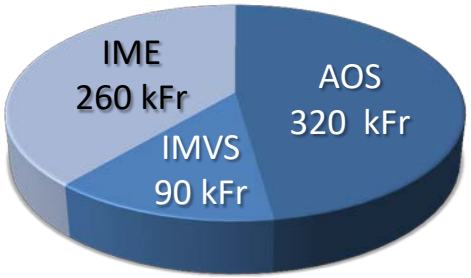
in Zahlen und Stichworten

- Geplante Laufzeit
  - 18 Monate, ab. 1.1.2016 (Verzögerung um 1 Jahr bei AOS)
- Projektkosten

wer den Kuchen bäckt

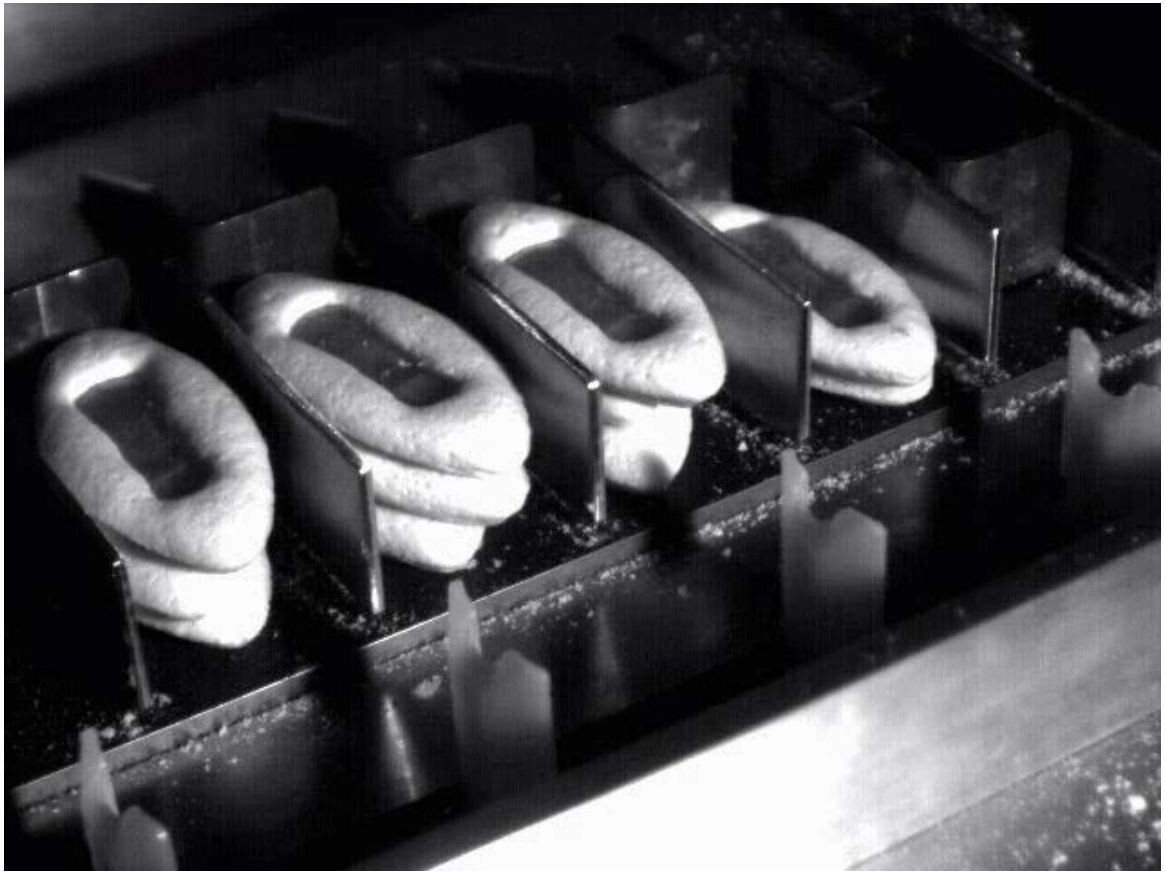


wer den Kuchen isst



- Projektablauf
  - AOS entwickelt
    - neue Kamera und neue Video-Verarbeitungssoftware
  - IMVS entwickelt
    - Videokompressionsverfahren (Codec) in C++
    - schnellen Decoder für die GPU in OpenCL
  - IME adaptiert das Codec und entwickelt
    - den Encoder des Videokompressionsverfahrens für ein FPGA

es läuft nicht immer rund



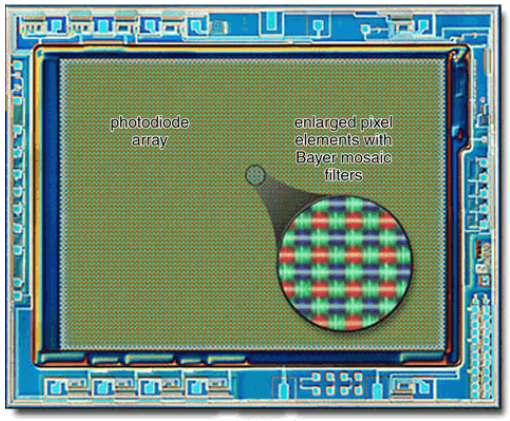
Hintergrund kurz erklärt

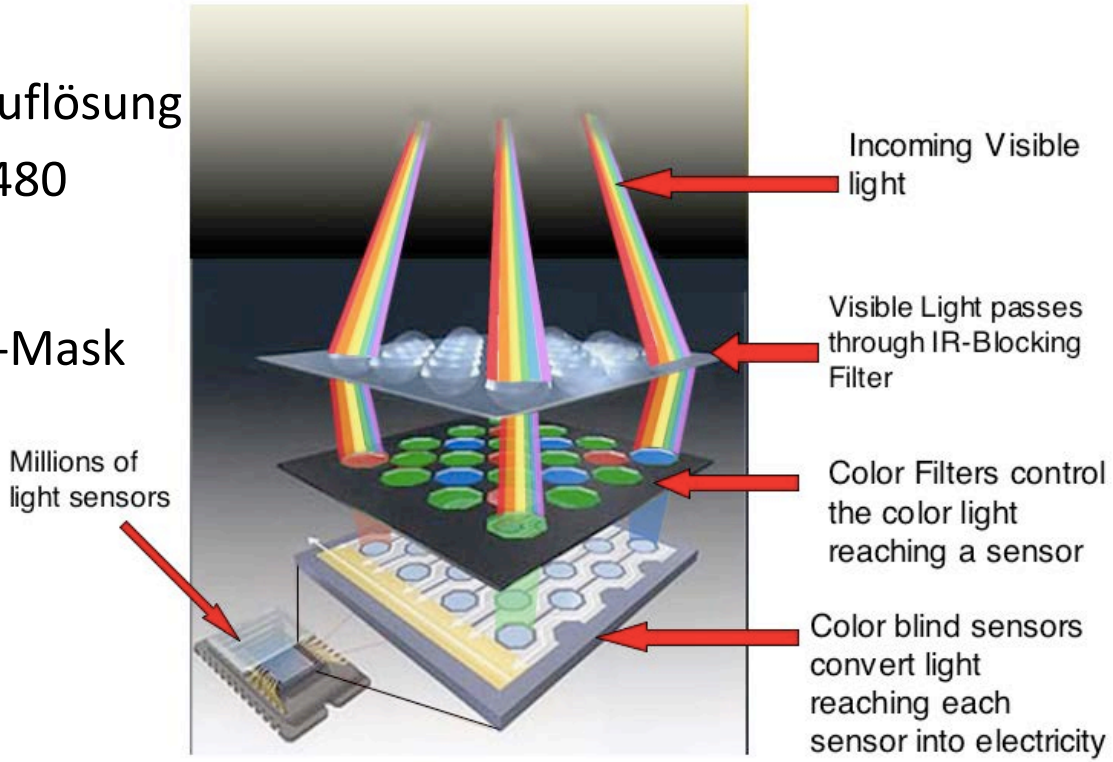
- Problemstellung
  - mit den sehr kompakten Hochgeschwindigkeitskameras der Firma AOS Technologies (Baden-Dättwil) lassen sich nur sehr kurze Aufnahmesequenzen (**maximal 8 Sekunden**) in der Kamera abspeichern
  - bei der Überwachung von sehr schnellen industriellen Prozessen braucht es öfters längere Aufnahmesequenzen (**z.B. 1 Minute**), um unvorhergesehene Abläufe in einem Video festhalten zu können

mögliche Lösungsansätze	Konsequenzen
– Video-Streaming an PC	verringerte Bildauflösung oder spezielle Hardware für Bildübertragung
– mehr Speicher in der Kamera	sehr schneller Speicher ist sehr teuer und Kamera wird grösser
– Videodaten komprimieren	Kamera benötigt mehr Rechenleistung

vom (Farb-)Bild zur Bewegung

- Maximale Bildauflösung
  - 1696 x 1710 2.9 MPixel (soll in Zukunft erhöht werden)
- Bildraten
  - 500 Bilder/s bei voller Auflösung
  - 4500 Bilder/s bei 640 x 480
- Sensortypen
  - monochrom oder Bayer-Mask

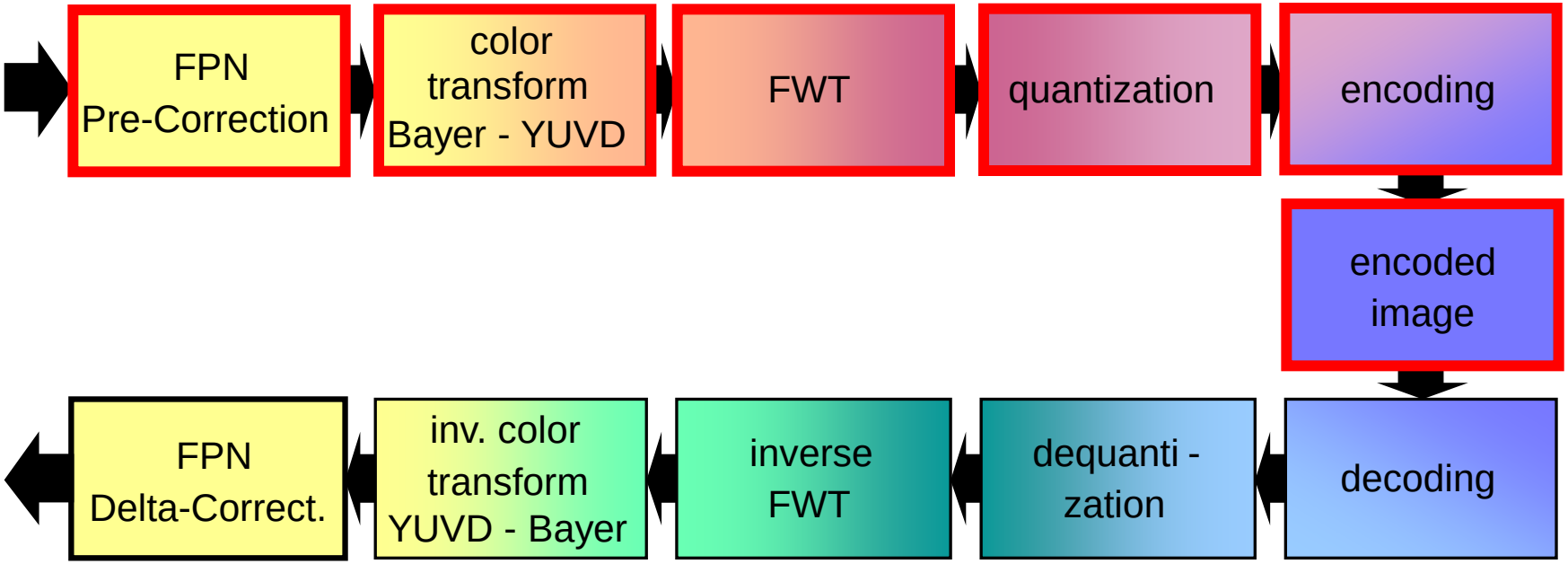




komprimieren ja, aber …

- Anforderungen an unseren Echtzeit-Encoder in der Kamera
  - Echtzeitanforderung erfüllen: 1.5 GPixel/s
  - minimaler Ressourcenverbrauch
  - verlustlose Kompression (möglichst hohe Kompressionsrate)
  - verlustbehaftete Kompression (möglichst hohe Videoqualität bei gegebener Kompressionsrate)
- abgeleitete Anforderungen
  - Datenfolge des Sensors berücksichtigen
  - minimaler Speicherbedarf (wenige Bildzeilen)
  - möglichst einfache Arithmetik (nur ganze Zahlen, Add/Sub/Shift)
  - möglichst gute Parallelisierbarkeit
- Konsequenzen
  - Standard Video-/Bild-Codecs eignen sich nicht

in fünf Schritten zum Ziel



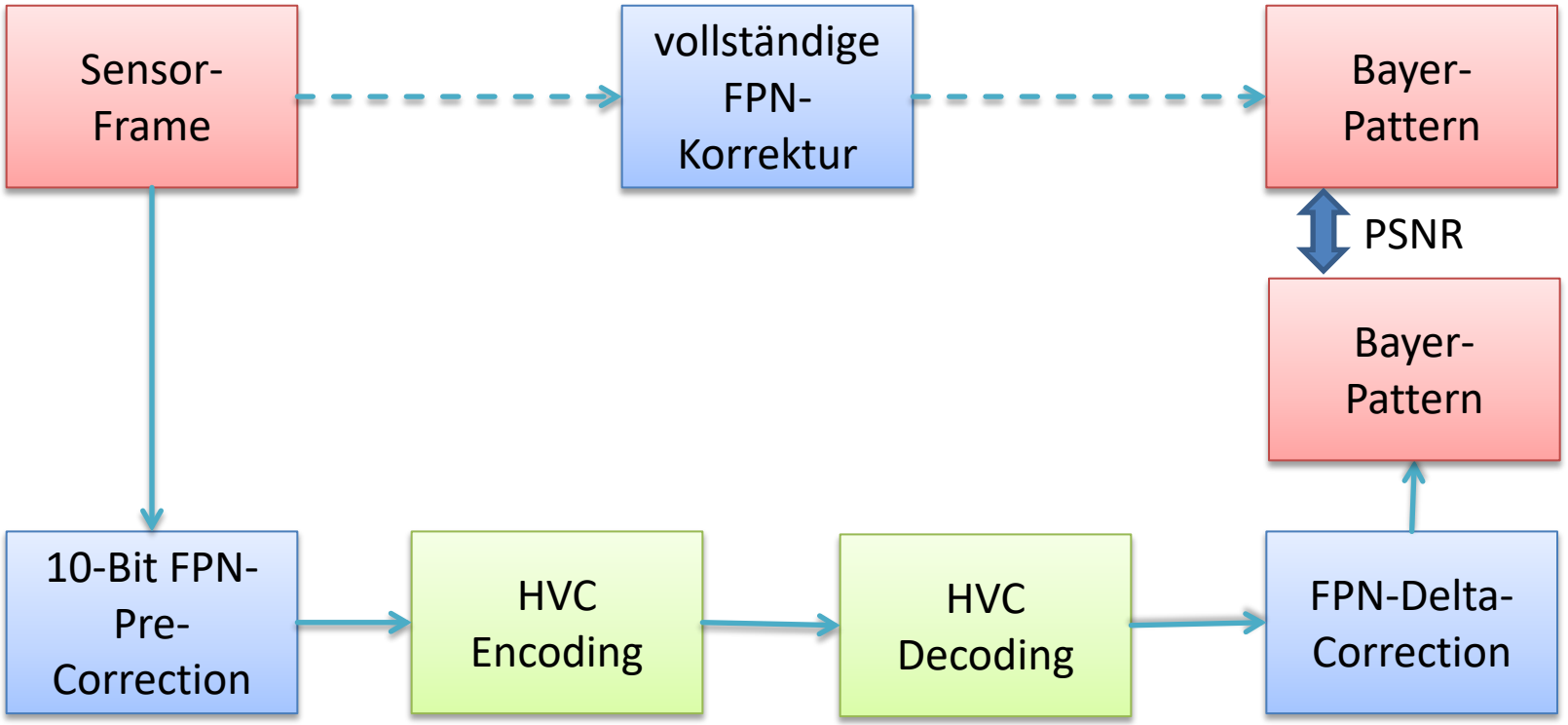
Originalbild

Bayer-Pattern RGGB oder Variante davon

«CMOS» rauscht ganz schön übel

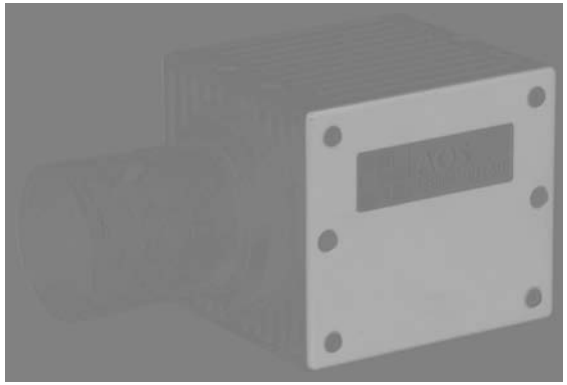
lästiges Rauschen unterdrücken

- Fixed Pattern Noise (FPN) Removal
  - bisher: quadratische Korrekturfunktion pro Pixel: $y = ax^2 + bx + c$
    (3 Korrekturparameter a, b, c mit insgesamt 48 Bit pro Pixel)
  - neu: Vorkorrektur mit 10 Bit pro Pixel



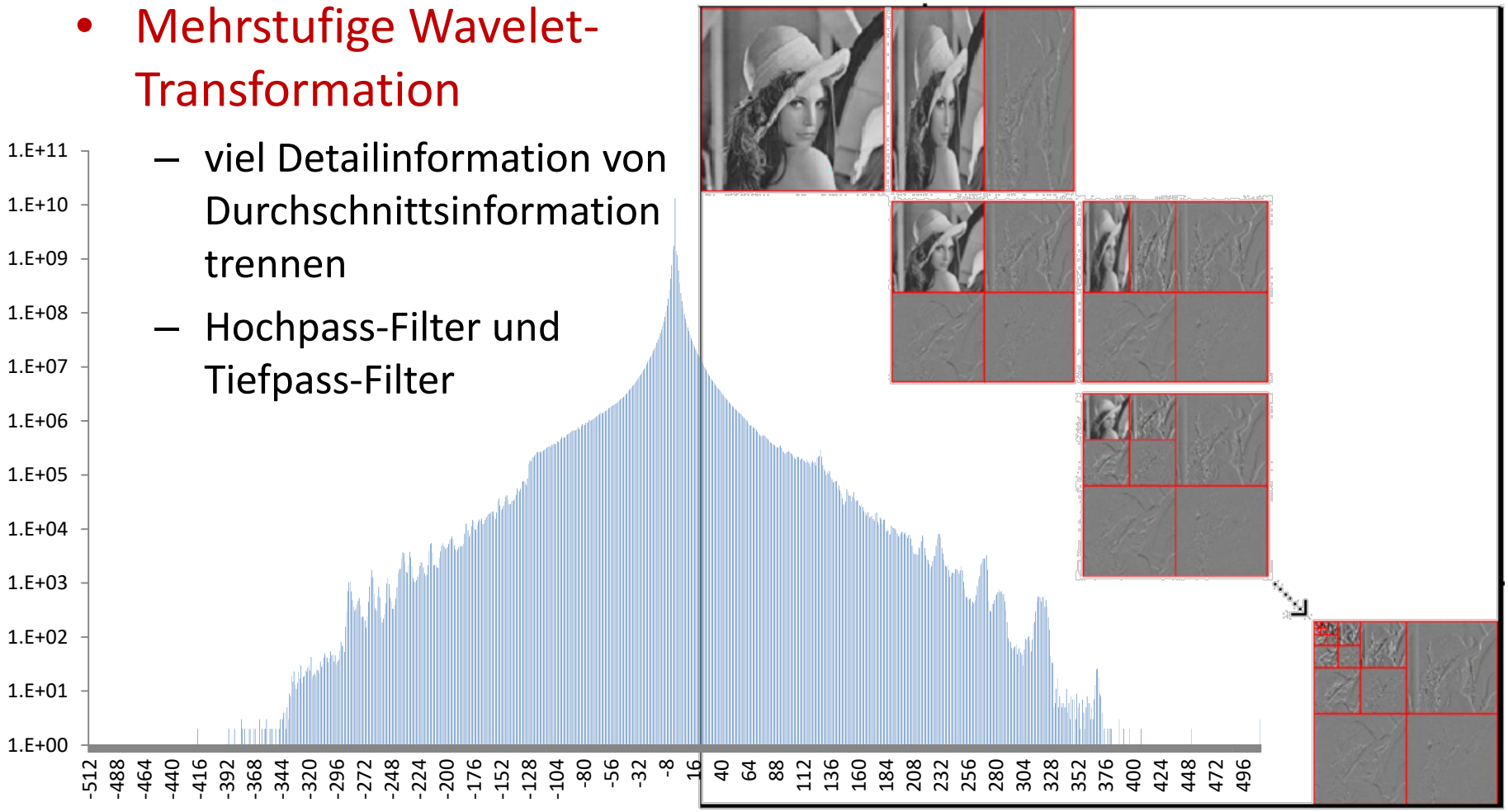
Energie bündeln lohnt sich

- Ziel der Farbtransformation
  - Bildenergie auf einen Kanal konzentrieren



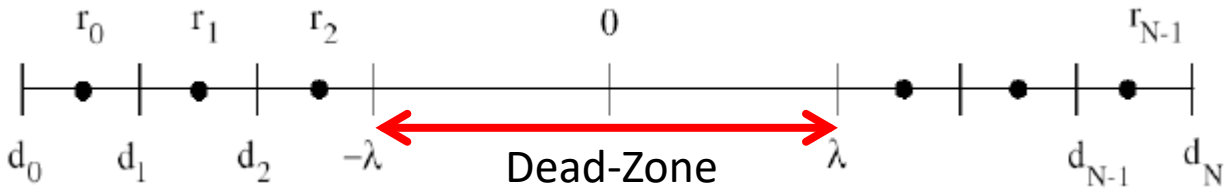
mit kleinen Wellen trennen

- Mehrstufige Wavelet-Transformation
  - viel Detailinformation von Durchschnittsinformation trennen
  - Hochpass-Filter und Tiefpass-Filter



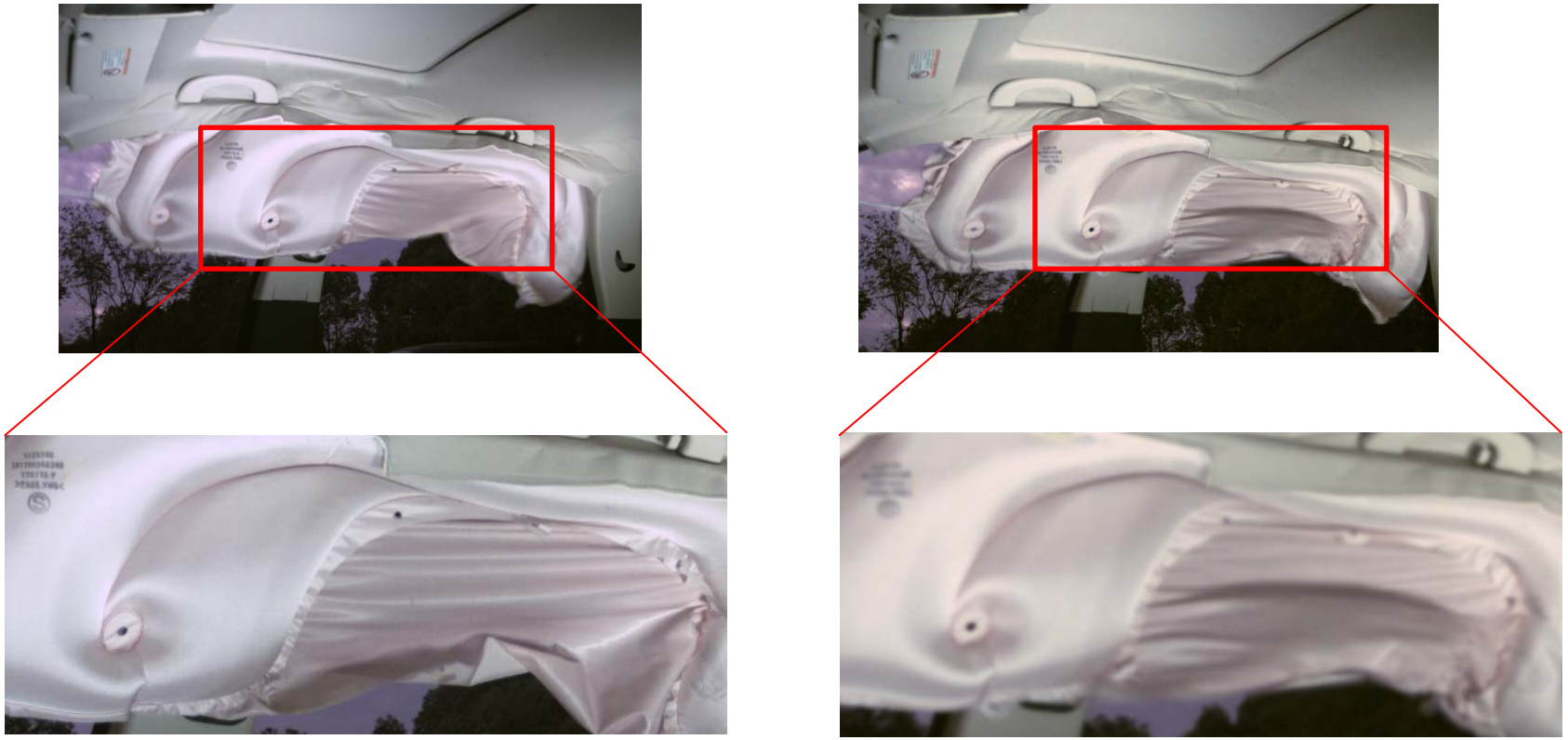
Details der Details entfernen

- Quantisierung der Wavelet-Koeffizienten
  - Quantisierungsparameter steuert die Stärke des Datenverlusts und somit die Kompressionsrate
  - uniforme, skalare Quantisierung abhängig von Subband und Level
  - Spannweiten sind Zweierpotenzen
- Neue Erkenntnis
  - durch Variieren der Breite der Dead-Zone kann eine zusätzliche Verfeinerung der Quantisierung erreicht werden, z.B. 5 oder 9 Unterstufen



Bildqualität kann gewählt werden

- Hochgeschwindigkeitskameras in Industrie und Forschung
  - Kameras sind Messmittel zur Erzeugung «technischer» Bilder



Daten weiter eindampfen

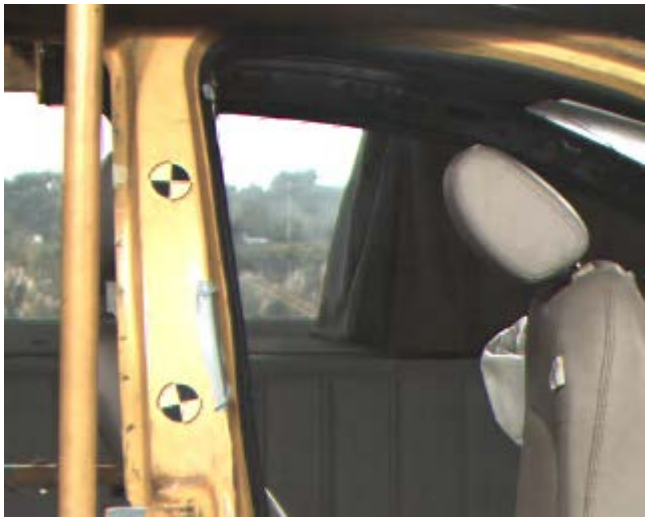
- Codierung der quantisierten Wavelet-Koeffizienten
  - häufige Wavelet-Koeffizienten (mit kleinem Absolutwert) sollen durch kurze Präfix-Codes repräsentiert werden
    - Codewörter variieren in der Länge zwischen 1 und ca. 50 Bit
  - Huffman-Codierung mit fixer Code-Tabelle lässt sich einfach parallelisieren
  - zusätzliches adaptives Run-Length-Encoding von Nullersequenzen kann die Kompressionsrate weiter erhöhen
    - erschwert aber die Parallelisierbarkeit und die Umsetzung in einem FPGA

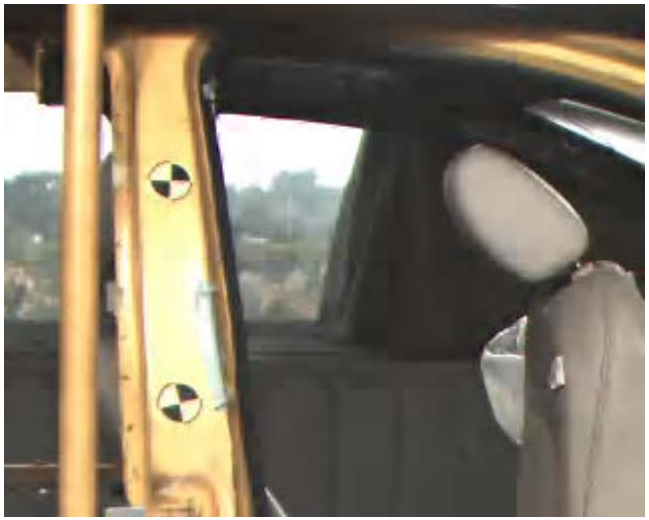
Bildqualität in Zahlen ausgedrückt

Original

40 dB

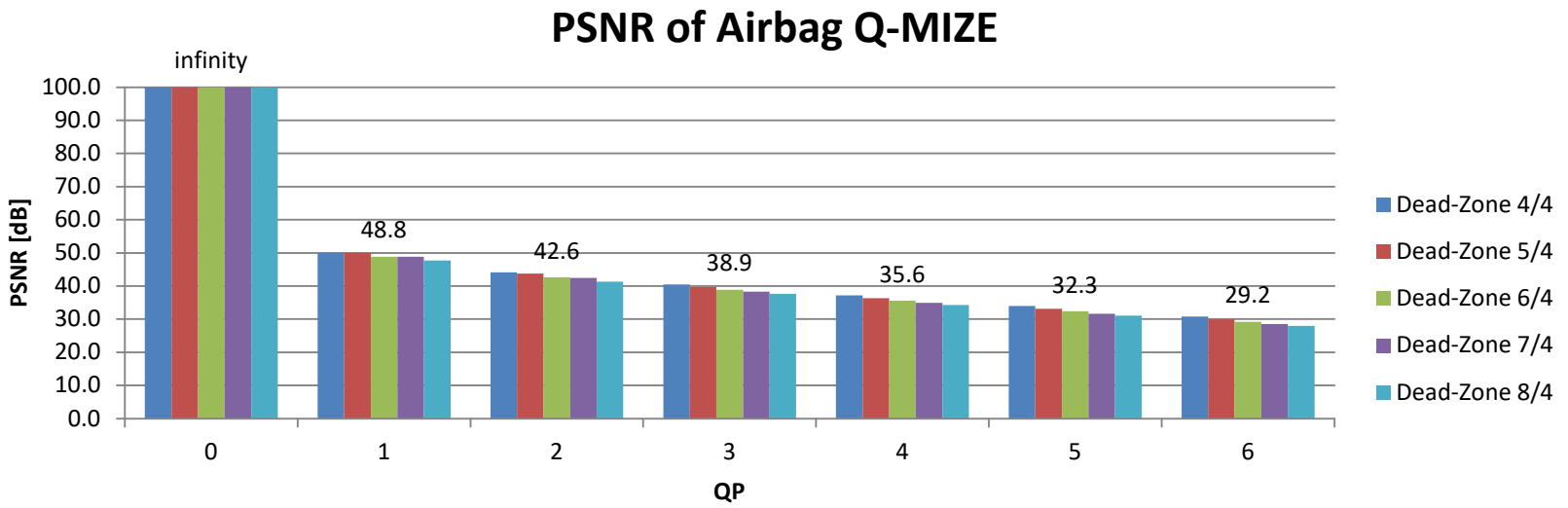
35 dB

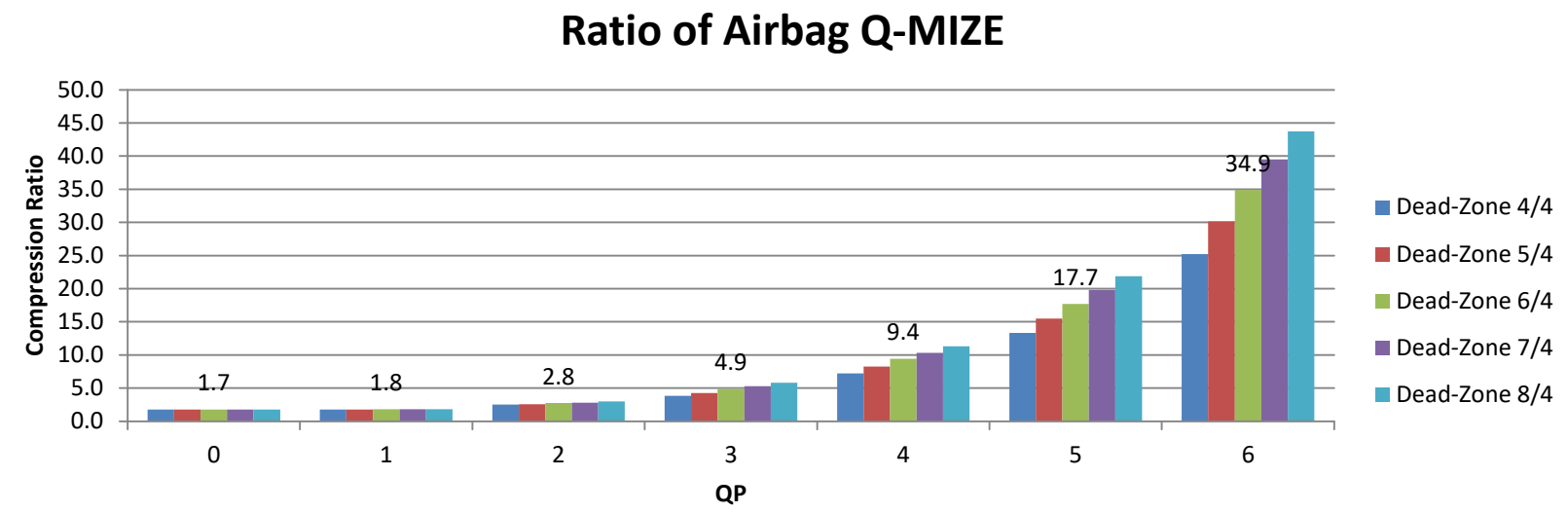




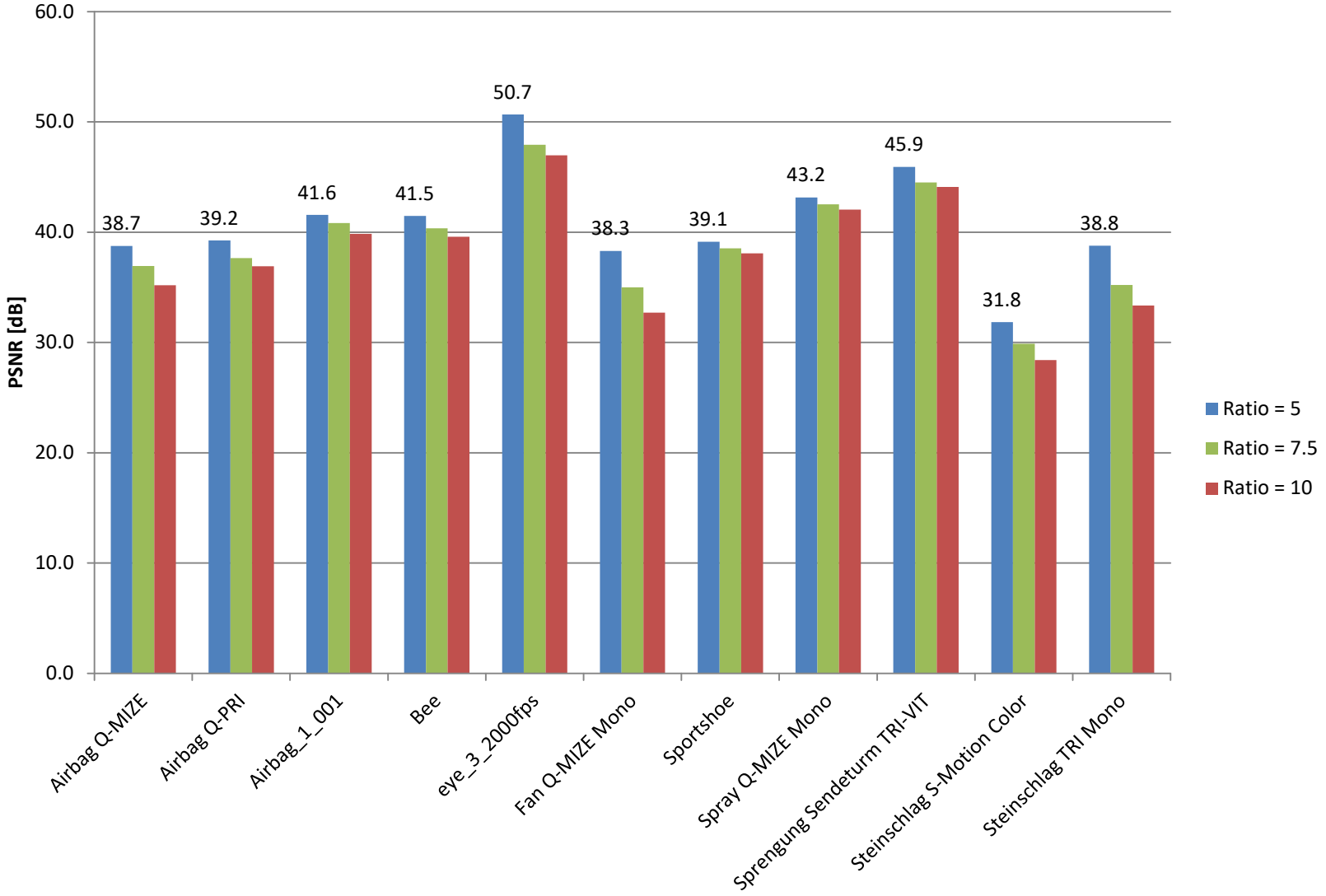
31 dB

der Einfluss der Todeszone





die zu erwartenden Resultate



Umsetzung Encoder in Hardware

Ziel: Verarbeitung von 500 Vollformat-Bilder pro Sekunde in Echtzeit = 1.5 GPixel/s = 11.6 Gbit/s

Realisierung des Encoders

1. **CPU** 15 MPixel/s (f = 3 GHz)

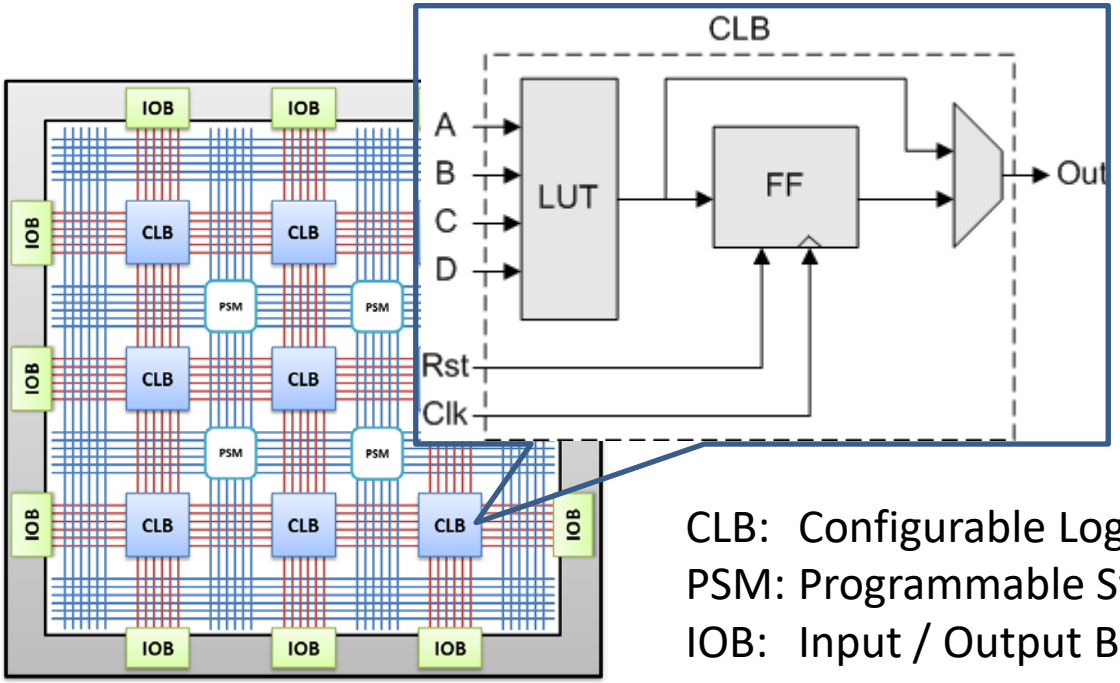
2. **GPU** 150 MPixel/s (f = 1.1 GHz)

3. **FPGA** 1500 MPixel/s ? (f = 103 MHz)

Was ist ein FPGA

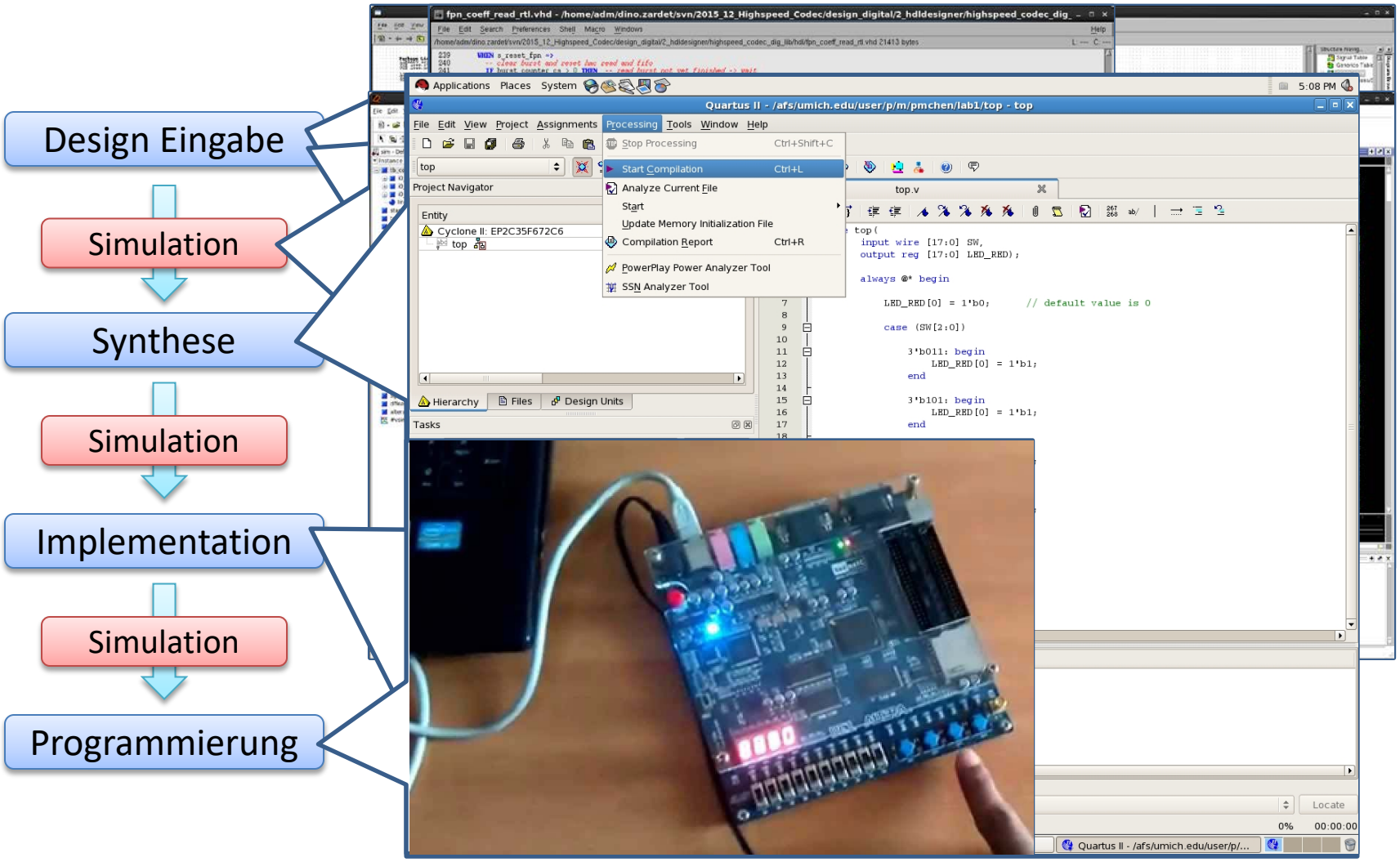
- FPGA: Field Programmable Gate Array

  Im Feld programmierbare (Logik-)Gatter-Anordnung
  - digitale Logikblöcke in einer regelmässigen Struktur angeordnet
  - Funktion der einzelnen Logikblöcke ist vom Anwender programmierbar



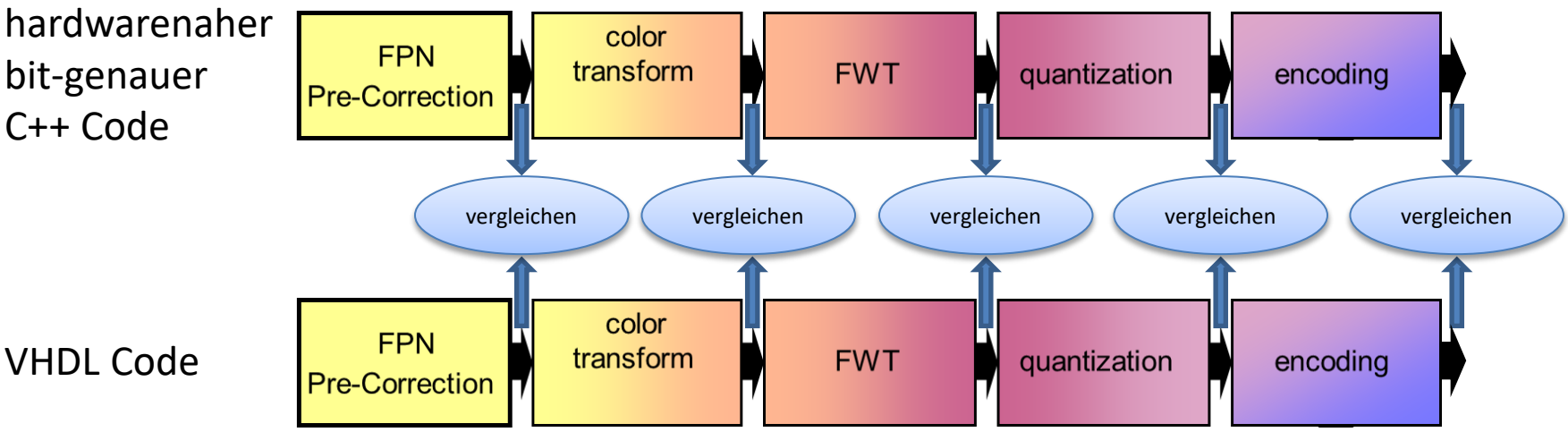
CLB: Configurable Logic Block
PSM: Programmable Switch Matrix
IOB: Input / Output Block

FPGA Design Flow

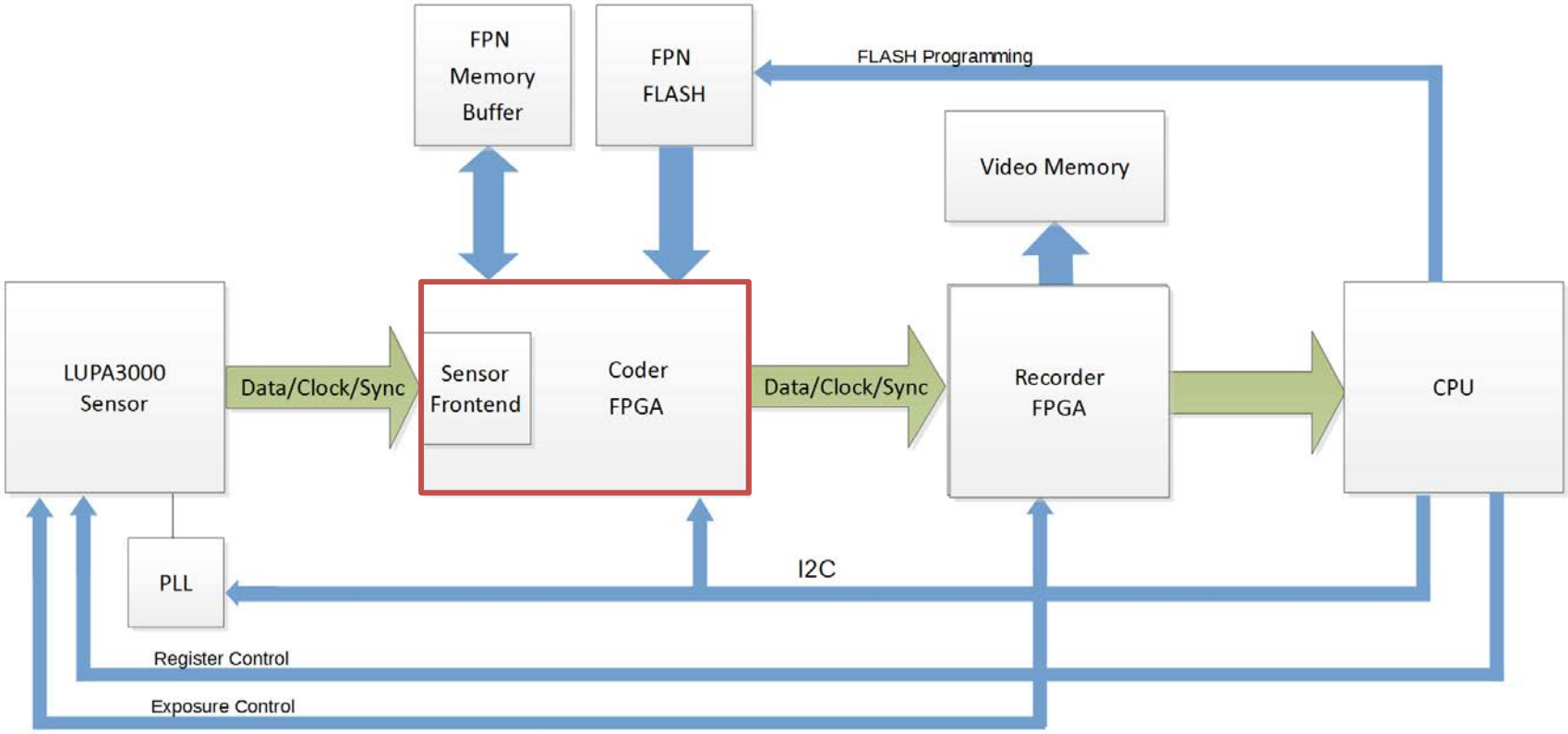


FPGA Entwicklung

- Keine Knopfdrucklösung (C++ -> FPGA)
  - z.B. Adaptives RLE: 30 Zeilen C++ Code, 1 Monat Arbeit für VHDL Code
- C++ Code -> hardwarenahes bit-genaues C++ Modell
- VHDL codieren
- Vergleich der Ausgangsdaten (VHDL – C++)



Ein Blick ins Innere der Kamera

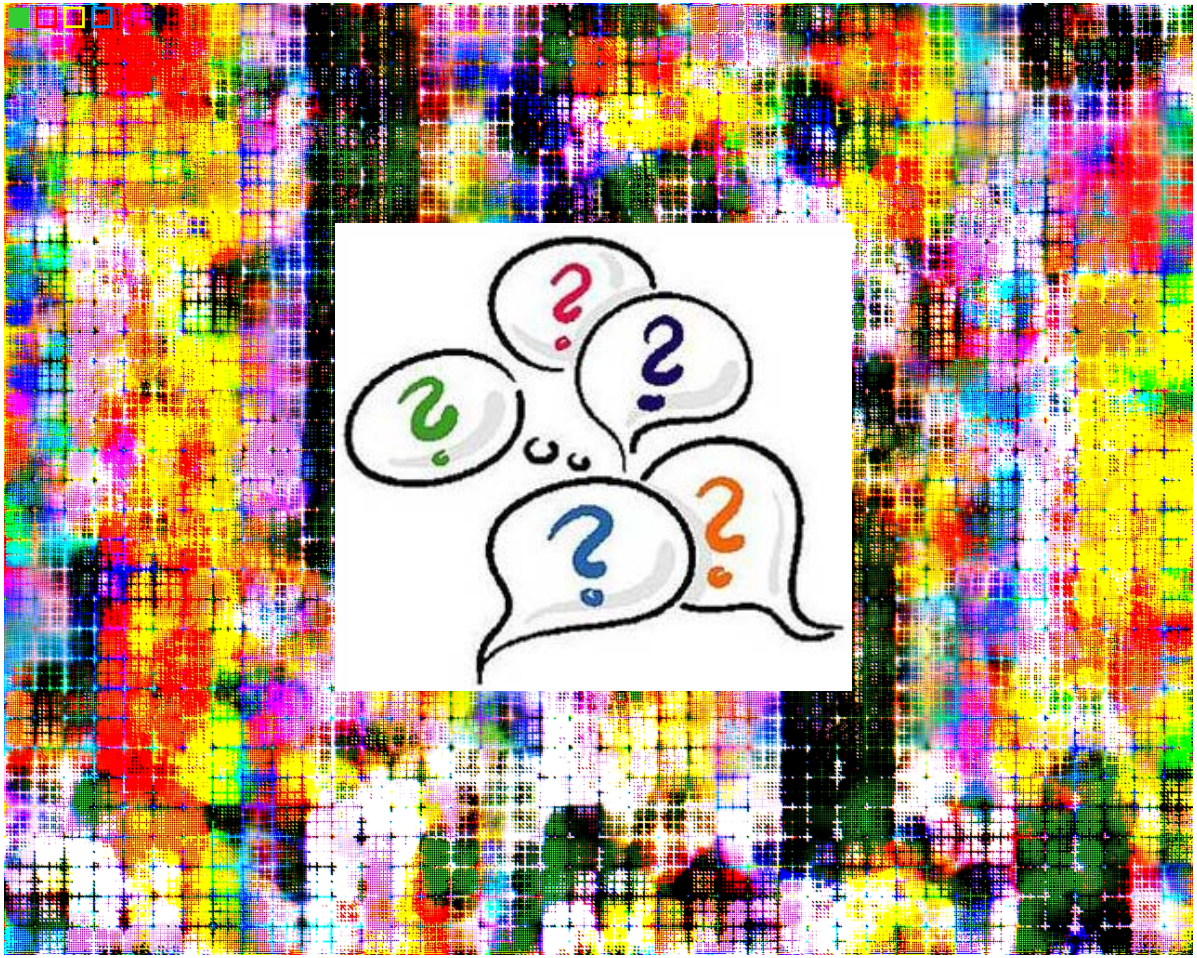


Anpassung Kamera-Hardware

Sensor → Recorder FPGA/ Memory → CPU

Sensor → Coder FPGA → Recorder FPGA/ Memory → CPU

Danke für Ihre Aufmerksamkeit



CodecArt http://web.fhnw.ch/technik/projekte/computervision/codecart