LYNXTechnik AG

Herausforderungen und Möglichkeiten von Künstlicher Intelligenz in TV und Broadcasting

Johanna Rohde – FPGA Firmware Engineer

About me

- **Johanna Rohde**

2016	Master of Science Electrical Engineering and Information Technology (TU Darmstadt)
2016	Máster Universitario en Ingeniería de Telecomunicación (Universitat Politècnica de València)
2016-2022	PhD Candidate at Rechnersysteme
Since 2022	FPGA Firmware Engineer at Lynx Technik AG

Overview

- Lynx Technik AG – Who we are
- Background
  - Video Streams
  - FPGAs
  - Yellobriks
- Implementing AI for TV and Broadcasting
  - Instant Dialogue Cleaner – The Software Solution
  - SDR to HDR Conversion
    - The DPU + Vitis AI Solution
    - The HLS or „AI is just an Algorithm“ Solution

Who We Are

Our Mission

The mission has been clear from the start: deliver cutting-edge and rock-solid products for Television Broadcast, Cinema Post-Production and Pro AV markets.

Proudly designed, engineered, and manufactured in Germany - we craft high-performance signal processing tools that professionals trust worldwide.

Founded in 2002 by 5 engineers

Now employing 32 employees directly.

Software R&D engineers, mechanical engineers, QA engineers, administration, assembly and sales.

Offices in Weiterstadt (HQ), Los Angeles, London and Singapore

Early days in 2002

The Products

Need to convert, synchronize, or distribute content? Our solutions handle it all—from analog to 8K over HDMI, Fiber, Coax or Ethernet.

- Robust card and rack frame

- Image processing
- Audio Embedder/De-Embedder
- Frame synchronizers
- Video/Audio conversion
- Fiber conversion
- Distribution Amplifiers
- Video Switching
- Fiber Splitters, Mux/Demux

- Multi-processing platform

- Up Down Cross conversions
- Video and Audio processing
- Static SDR <> HDR conversions
- Dynamic SDR <> HDR conversions
- Test Signals Generator
- Audio Video Sync Analyzer

- Modular brick solution

- Audio Video conversions
- Audio Embedder/De-Embedder
- Frame Synchronizers
- Fiber conversions
- Distribution Amplifier
- Fiber Splitters, Mux/DeMux
- Multiviewers
- HDMI conversion
- H265 Streamer
- Sync Generator
- AI Dialog Cleaner

- HD Test Pattern Generator

- 3G-SDI
- SMPTE 424M, SMPTE 292M, SMPTE 259M
- Audio Embedded
- 1x Reference Output

Real World Applications

LYNX Technik products are used daily in mission critical applications. They are trusted to perform and deliver with high uptimes. Some devices are used on World Cups, Olympic Productions, Rocket Launchpads, Submarines and Cruise ships!

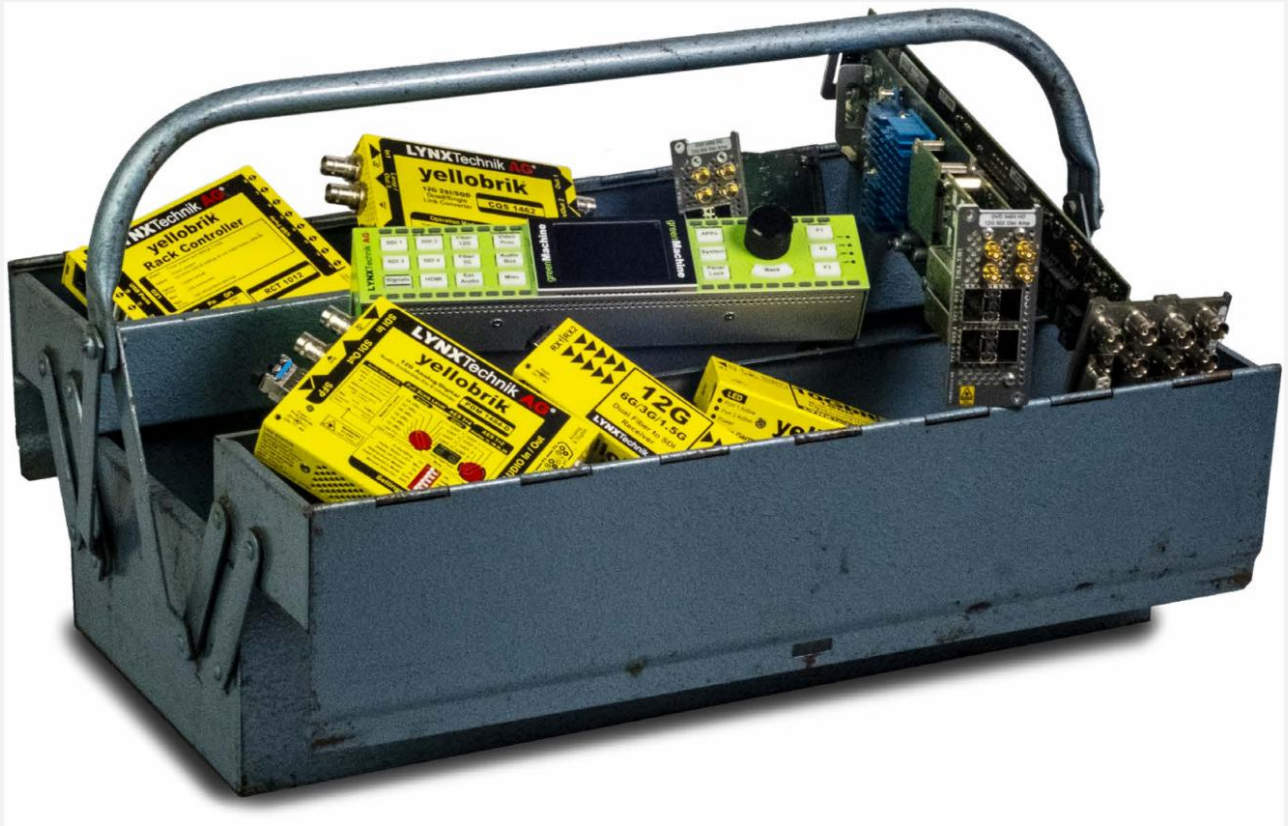
Television Broadcast
Traditional Broadcast, Remote Production, Outside Broadcast, Studio recording

Film & Media
On-set, post-production, Digital cinema, Preview

Government & Education
Court Houses, Aerospace, Transportation, College Campuses, Remote Learning

Cable, Satellite, Telco, Service Providers
MSO, DBS, Transportation Hubs, Contribution Content

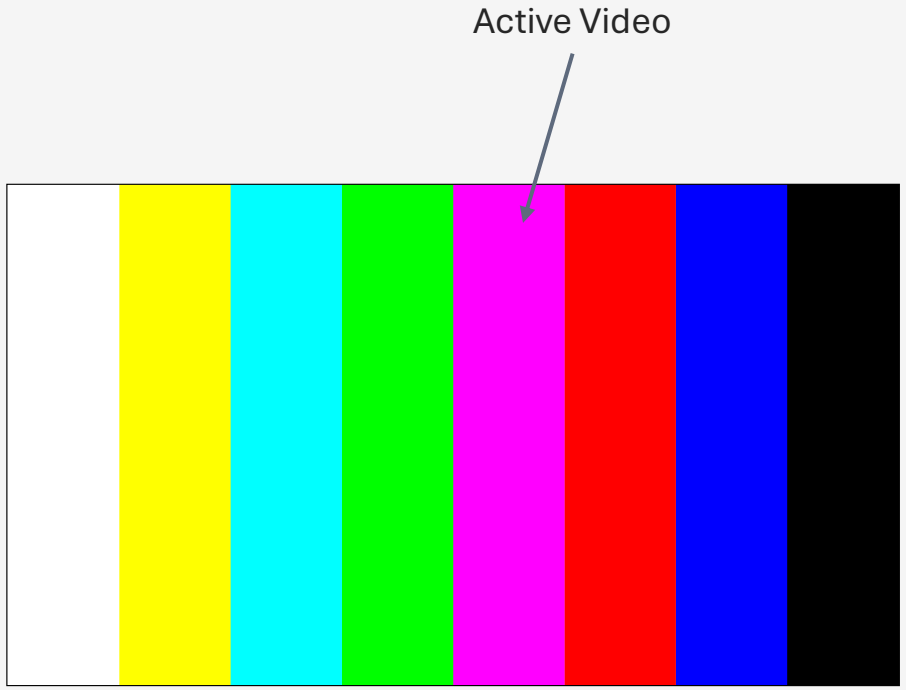
Industrial & ProAV
Theatres, Concert, Live Events, Stadiums, In-house productions, Online streaming, Monitoring, Surveillance, Fiber transport, Datacenters



Setting The Standard Worldwide

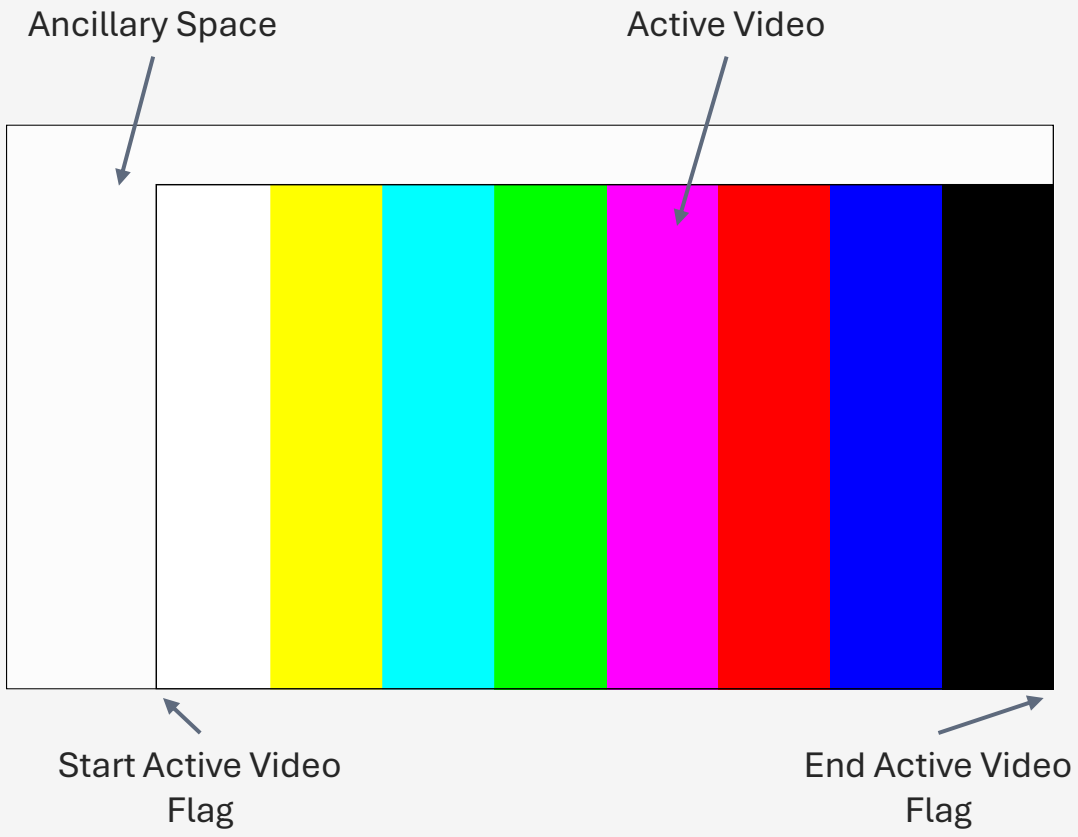
LYNX Technik is trusted by professionals around the world, backed by expert partners who ensure top-tier support. We don't just solve today's media challenges, we set new industry standards with award-winning, high-performance solutions that are built for the future.

Video Streams



- Society of Motion Picture and Television Engineers (SMPTE)
- Serial Digital Interface (SDI) Standard
- Active Video is the visible part of the video stream

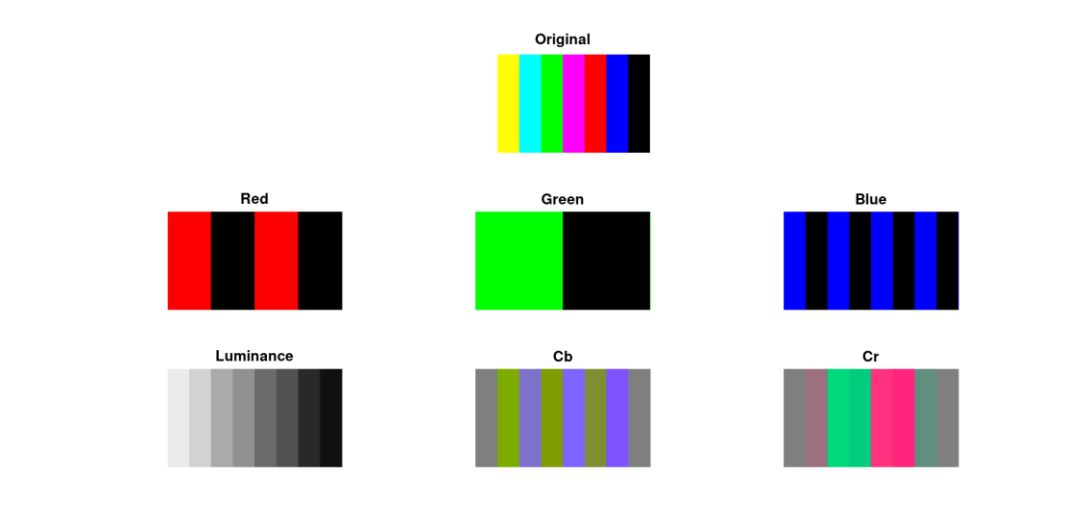
Video Streams



- Society of Motion Picture and Television Engineers (SMPTE)
- Serial Digital Interface (SDI) Standard
- Active Video is the visible part of the video stream
- Ancillary Space is for meta data
  - Video payload identifier (VPID) - describes the video format
  - Subtitles
  - Audio

Video Streams Characteristics

Video Streams Characteristics



- Resolution: 1280x720, 1920x1080, 3840x2160, ….
- Frames per Second: 60, 59.94, 50, 30, 29.97, ….
- Bits per Component: 8, 10, 12
- Color Space: YCbCr, RGB
- Color Encoding: BT.601, BT.709, BT.2020
- Subsampling: 4:4:4, 4:2:2, 4:2:0
- Scan Type: progressive, interlaced, psf
- Level A, Level B
- In total over 1000 different standards possible

Video Streams Characteristics

- Different Bitrates
  - Predefined bit rates for different groups of video standards
  - 1.485 Gbit/s (“1.5G”) – e.g 1080p30 YCbCr 422
  - 2.97 Gbit/s (“3G”) – e.g 1080p60 YCbCr 422
  - 12 Gbit/s (“12G”) – e.g 2060p60 YCbCr 422
- Whenever we develop something new, it must work for video standards up to 2060p60
- No microcontroller can handle this much data in real time

FPGAs

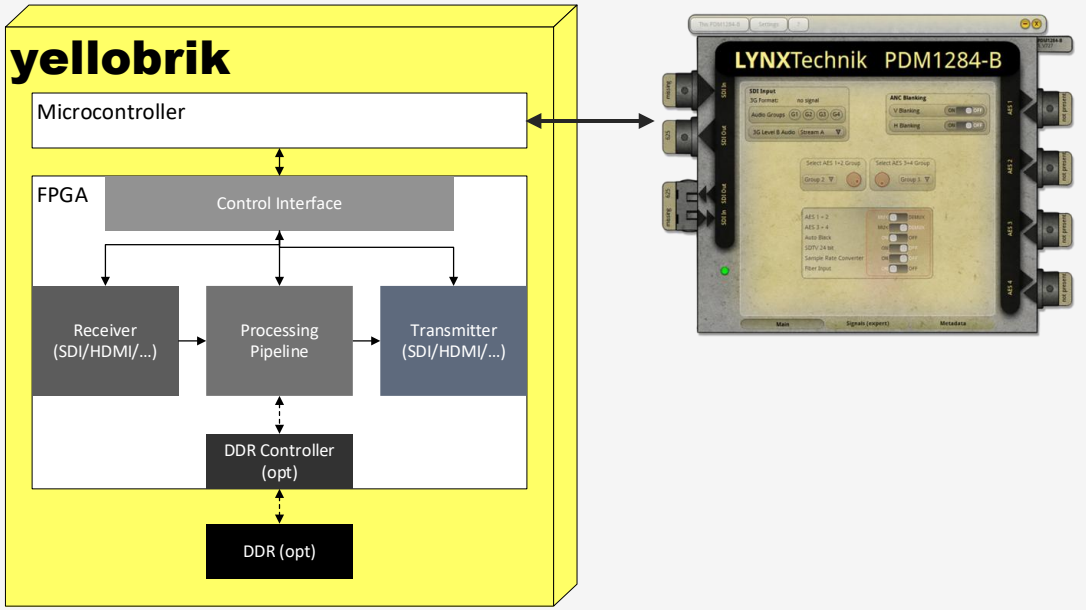
Microcontroller

- Fixed hardware architecture with a CPU, memory and peripherals
- Processes instructions sequentially
- Better suited for simple, low throughput, power sensitive applications
- Programmed in high level language such as C/C++, Python
- Fast turnaround time
- Many libraries available
- More variation, low costs

FPGA

- Programmable logic blocks and interconnects, allowing for highly customizable digital circuits
- Everything, everywhere, all at once
- Better suited for high throughput, parallel processing such as image processing
- Programmed in a hardware description language such as VHDL or Verilog
- Slow turnaround times due to synthesis
- IP blocks for basic functionalities
- Less vendors, less options, quite expensive

Basic Architecture of a Yellobrik



YelloGUI | Lynx Centraal

- The GUI is installed on the host computer. The Yellobrik is connected via USB
  - Access to internal yellobrik settings and controls.
  - Set video resolution, frequency
  - Adjust colour depth
  - Configure I/O
  - Transmission fallback test patterns
  - Update firmware
- The Microcontroller on the Yellobrik interacts with the GUI and the FPGA
  - Reports status to the GUI
  - Configures receiver, processing pipeline and transmitter via the control interface
- The FPGA receives, processes and transmits the video stream
- DDR Memory is optional, depending on the application

Implementing AI for TV and Broadcasting

Instant Dialogue Cleaner

-

The Software Solution

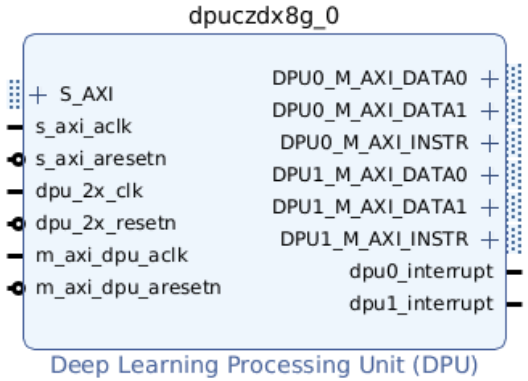


Image source: AMD

SDR to HDR Conversion

-

The DPU + Vitis AI Solution

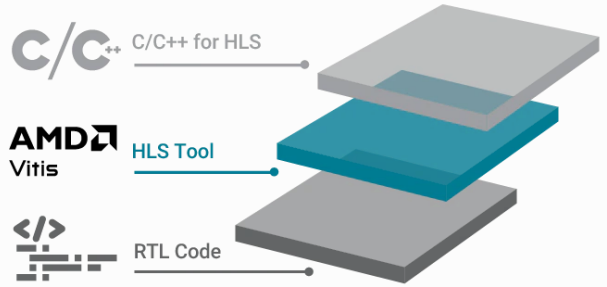


Image source: AMD

SDR to HDR Conversion

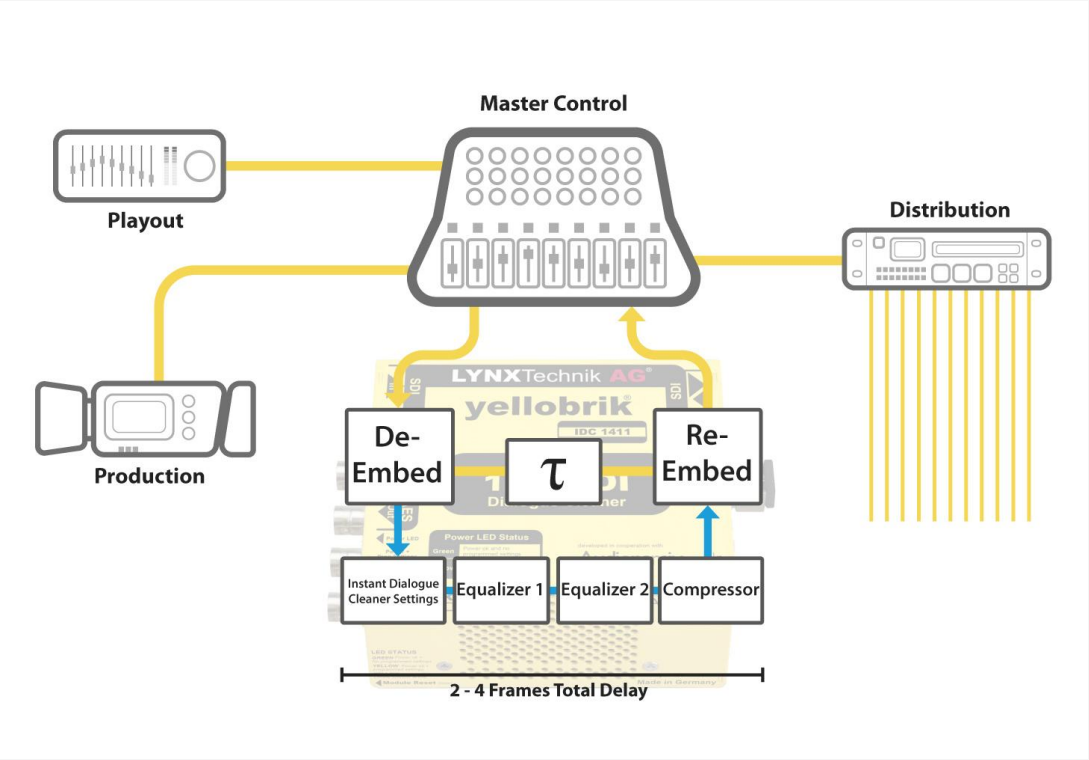
-

The HLS or „AI is just an Algorithm“ Solution

The Instant Dialogue Cleaner

- Enhances speech based on an AI dialogue cleaner in real time
- E.g., available for ARD and ZDF
  - “Klare Sprache”
  - HD Live TV
  - https://www.ard-digital.de/klaresprache-geraete
  - https://youtu.be/x8peF-j55LE

The Instant Dialogue Cleaner



- 2 Audio Inputs
  - Embedded in SDI video stream
  - AES input
  - User selectable
- Audio is deembedded from SDI video stream
- Audio is processed
- Audio is reembedded into SDI video stream
- 2 Audio Outputs
  - Embedded in SDI video stream
  - AES Output

The Instant Dialogue Cleaner

- A neural network separates speech and background
- User can apply gain separately
- Strength parameter makes filter more/less „aggressive“ – it will identify more/less content as noise
- Works in real time

The Instant Dialogue Cleaner

Dialogue Cleaner

Receiver, Deembedder, Embedder, Transmitter

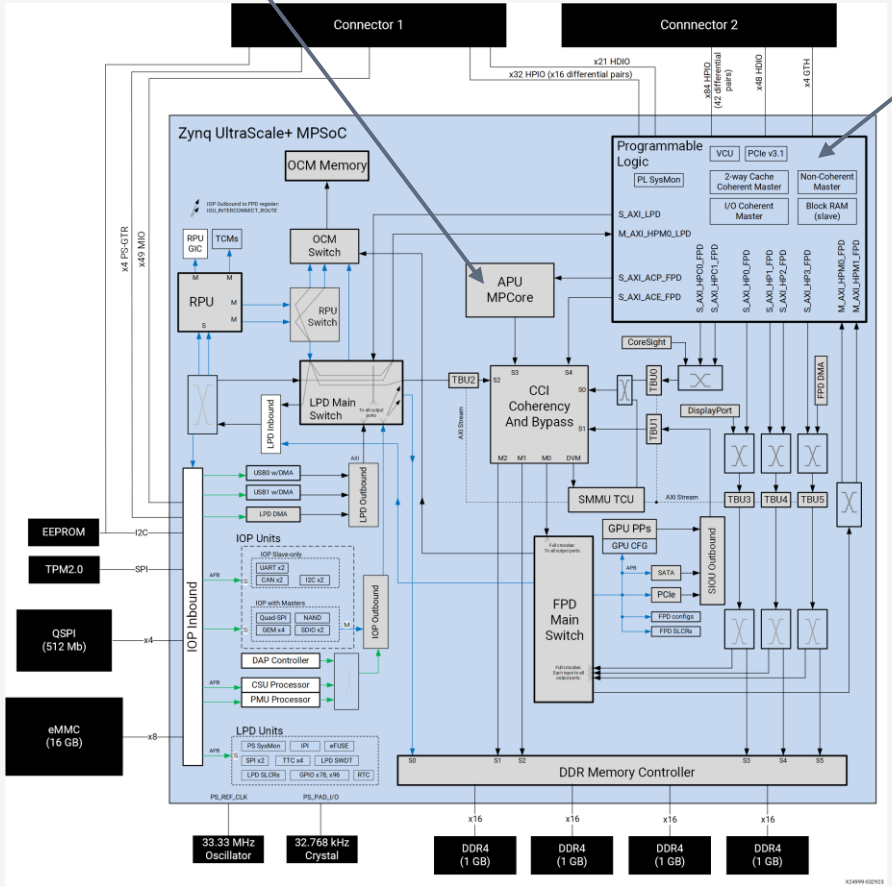
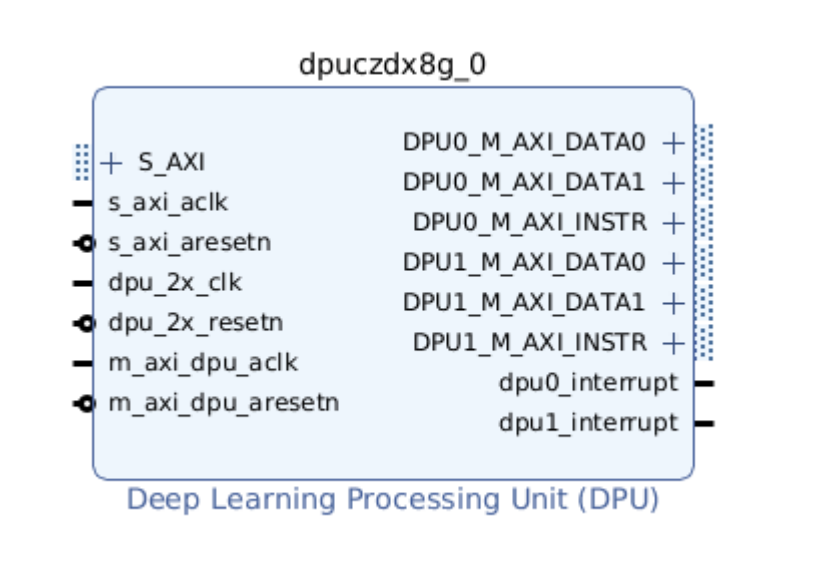
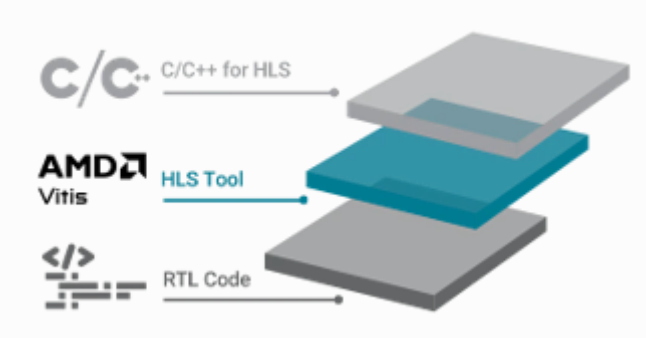


Image source: AMD

- Zynq UltraScale+ MPSoc
  - ARM Core and FPGA on one chip
- FPGA includes receiver, audio deembedder, audio embedder and transmitter
- Audio is written into a shared memory
- Audio processing (including dialogue cleaning) is done by the ARM Core of the Zynq
- Why process it in software?
  - The neural network is bought from Audionamix
  - Comes as software plugin
  - Audio is sampled with 48 kHz, so the data can easily be processed by the microcontroller

SDR to HDR Conversion





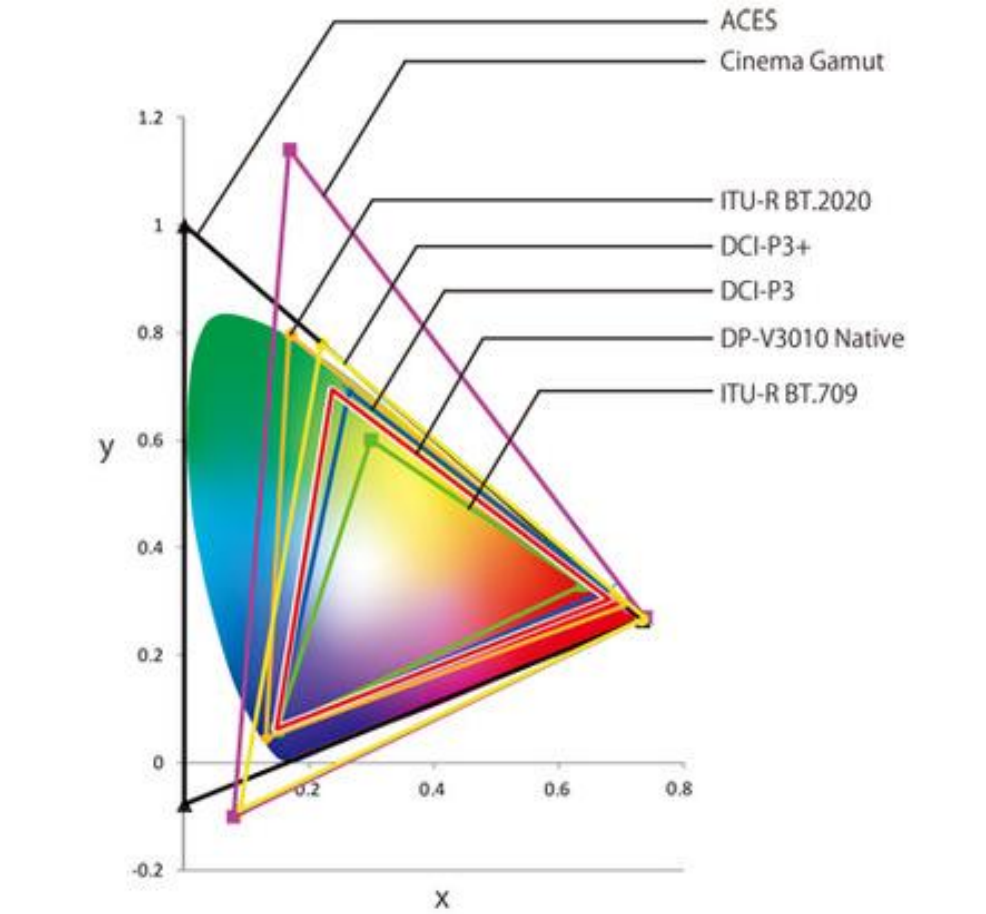
High Dynamic Range

HDR on HDR Display

HDR on SDR Display

- HDR stands for “High Dynamic Range”
- Dynamic range is the range between the maximum light intensity and the minimum light intensity
- HDR is the technology that increases a scene’s contrast range during recording, processing, distribution and playback

Wide Color Gamut (WCG)

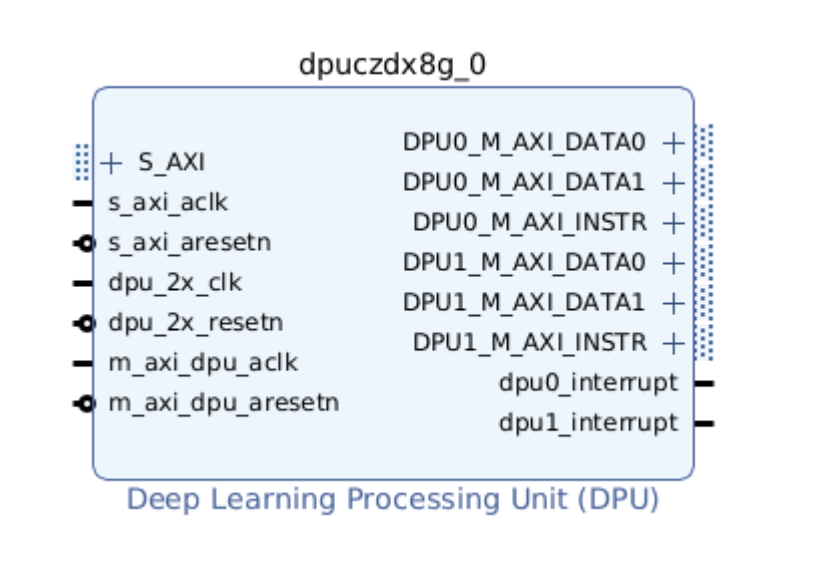


- A color gamut describes a range of color that can be captured/displayed by a given system.
- A wide color gamut makes the color palette bigger and allows more unique color available in an image.
- WCG can require higher bit depth.
- WCG is often found in conjunction with HDR but they are not intrinsically linked

SDR to HDR Conversion

- Most productions and consumer end devices already support HDR/WCG
- But sometimes it is necessary to include SDR/SCG sources
  - archive material
  - semi professional and old cameras that only produce SDR/SCG footage
- In that case, the SDR footage needs to be converted into HDR such that it integrates seamlessly
- Challenges
  - Details in the highlights and shadows must be reconstructed
  - Blocking instead of smooth color gradients
  - Real time
- Hochschule RheinMain Student Talk: AI-based HDR reconstruction and expansion for video applications (2023)
  - https://www.hdm-stuttgart.de/vmlab/conferences/vmlc2023

DPU and Vitis AI

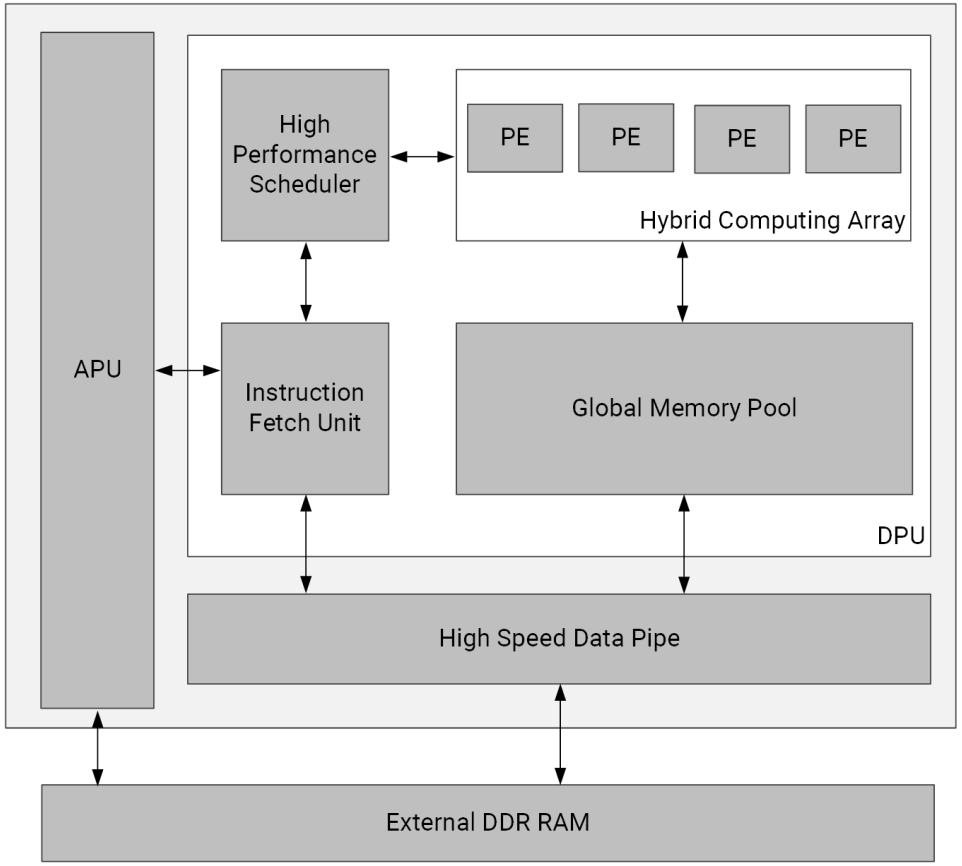


AMD

Vitis AI

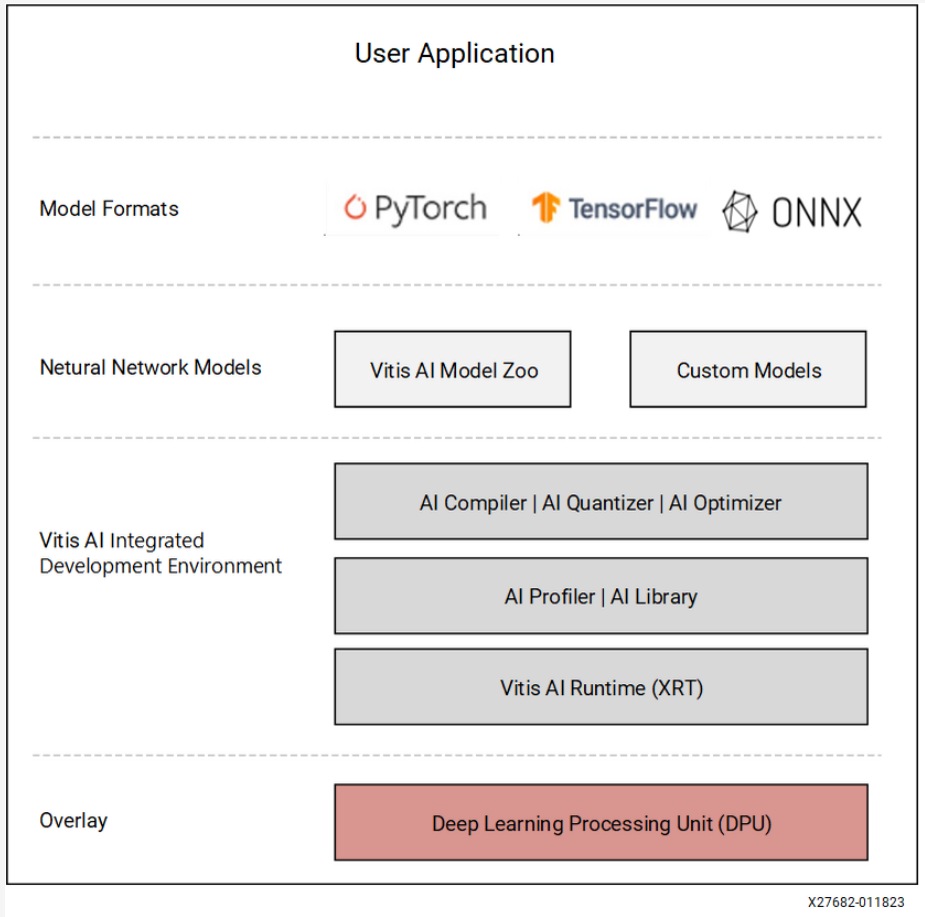
Deep Learning Processing Unit (DPU)

- Configurable computation engine optimized for convolutional neural networks
- IP core designed for the programmable logic of the Zynq Ultrascale+ MPSoC
- Executes microcode generated from a neural network graph.
- Requires memory access for input, output and temporary data
- Support a predefined list of operators commonly used in CNNs
  - https://docs.amd.com/r/en-US/ug1414-vitis-ai/Currently-Supported-Operators



Vitis AI

- Vitis AI is not a single software but a whole software ecosystem
- The goal is to take a CNN (either from the Vitis AI model zoo or a custom model) and to accelerate it using DPUs
- In our case: A PyTorch Custom Model created by the Hochschule Rhein-Main
- It's a four-step process: Quantize, export, compile, deploy
- Quantization
  - CNN usually operate on Float32
  - DPUs operate on Int8 fix point
  - The process of converting the input and the weights from Float32 to Int8 is called quantization
  - Advantages:
    - Integer value can be processed faster
    - Less memory needed for weights, input and temporary data
    - Less memory accesses speed up the calculation
  - Disadvantages:
    - Reduces the precision of the CNN



Vitis AI

- Vitis AI is not a single software but a whole software ecosystem
- The goal is to take a CNN (either from the Vitis AI model zoo or a custom model) and to accelerate it using DPUs
- In our case: A PyTorch Custom Model created by the Hochschule Rhein-Main
- It's a four-step process: Quantize, export, compile, deploy
- Quantization
  - CNN usually operate on Float32
  - DPUs operate on Int8 fix point
  - The process of converting the input and the weights from Float32 to Int8 is called quantization
  - Advantages:
    - Integer value can be processed faster
    - Less memory needed for weights, input and temporary data
    - Less memory accesses speed up the calculation
  - Disadvantages:
    - Reduces the precision of the CNN

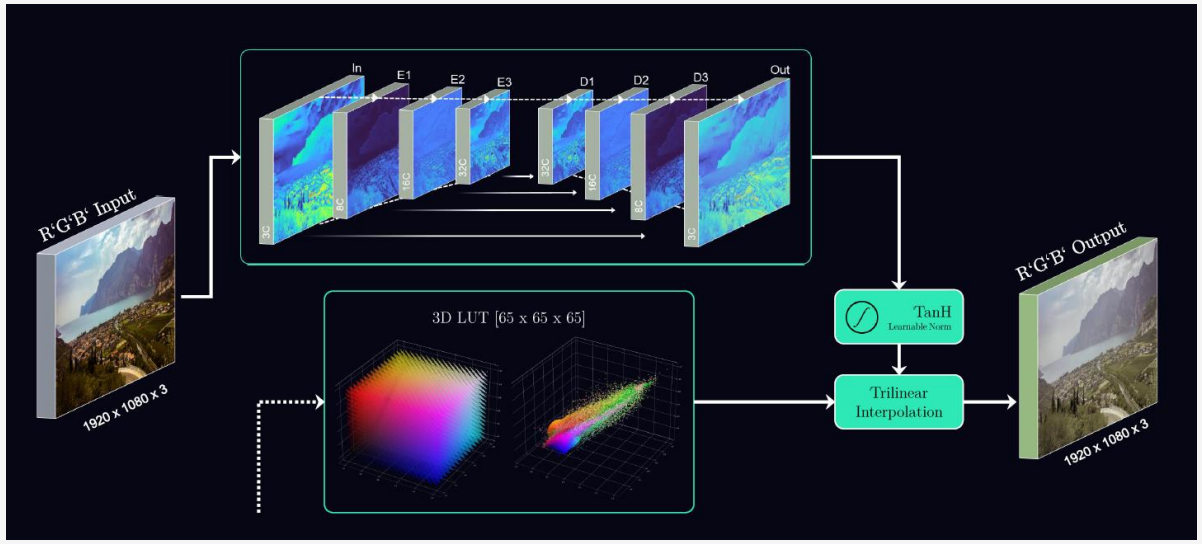
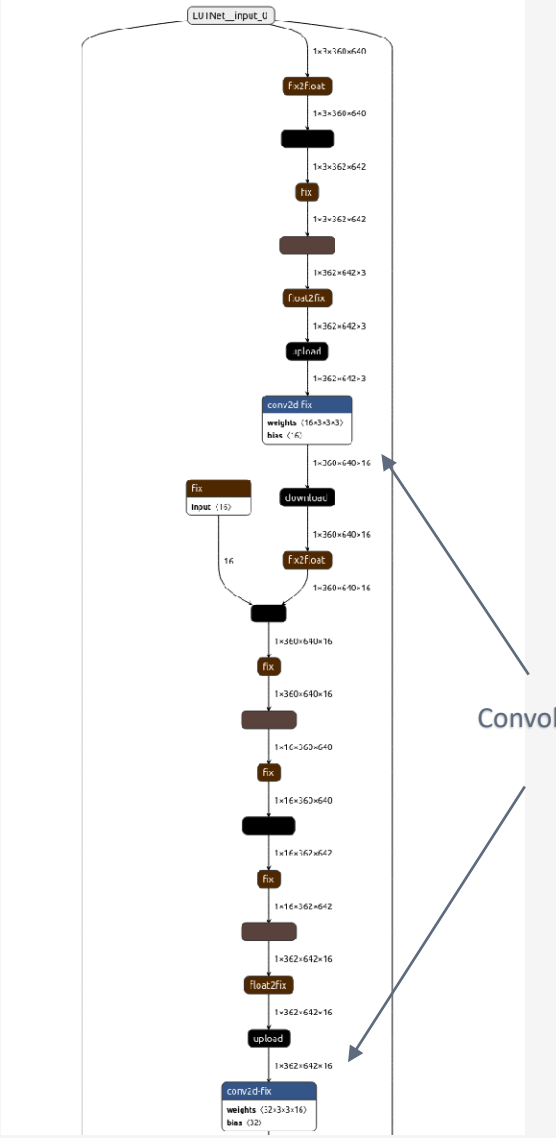


Image source: Pascal Kutschbach
Hochschule Rhein-Main

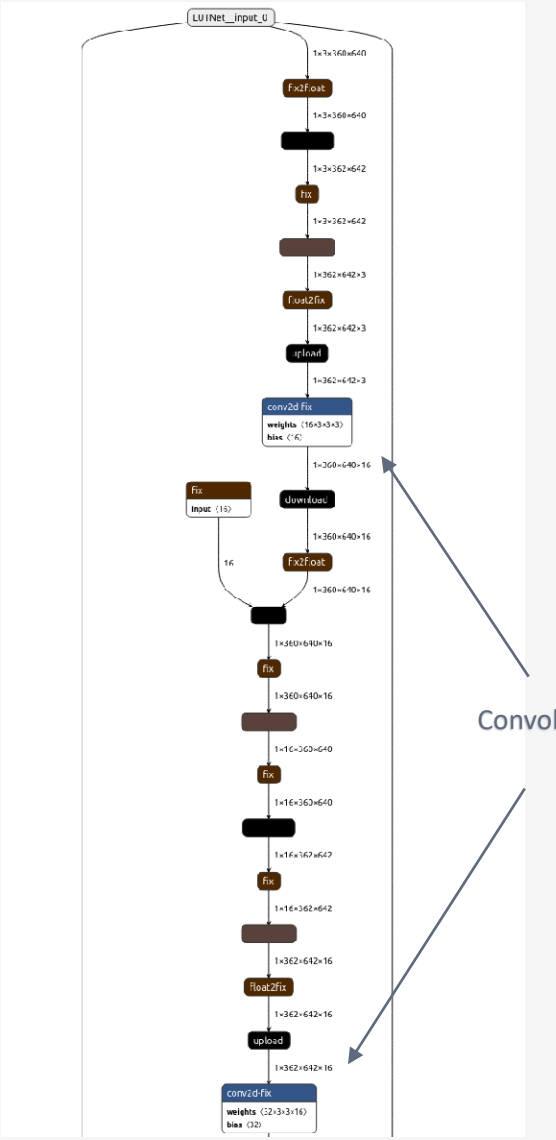
Vitis AI

- Export
  - The quantized model is compiled into a xmodel file
- Compile
  - The Vitis AI Compiler generates a „computation graph“ where nodes are operations performed on the CNN (e.g., convolution, matrix multiplication, …)
  - Each node is assigned a compute unit. This is either the CPU or the DPU.
  - To achieve best performance, it is desirable that as many nodes as possible are mapped on the DPUs
- Deploy
  - Run the CNN on the Zynq Board
  - Vitis AI Library provides preimplemented operations
  - Vitis AI Runtime (XRT) takes care of executing the operations and moving data to and from the DPU



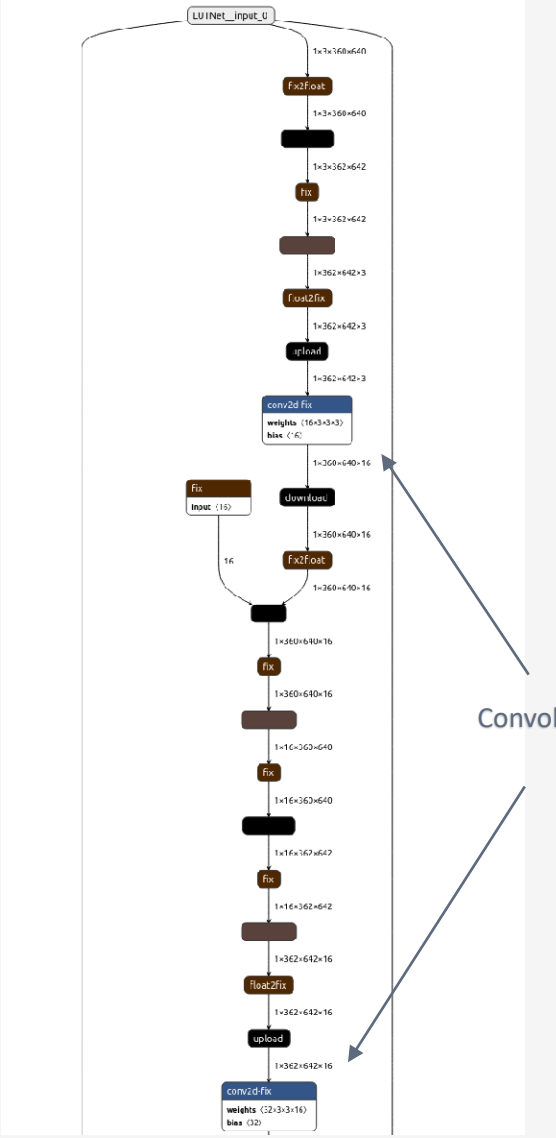
Results

- “Oh, that sounds easy”– No!
  - It took us 9 months to get to a deployed CNN
  - Four (unanswered) bug reports to Xilinx
  - Meaningless error messages
  - Countless workarounds
- Input: 360x640 Pixel
  - Runtime CPU: 0,1s



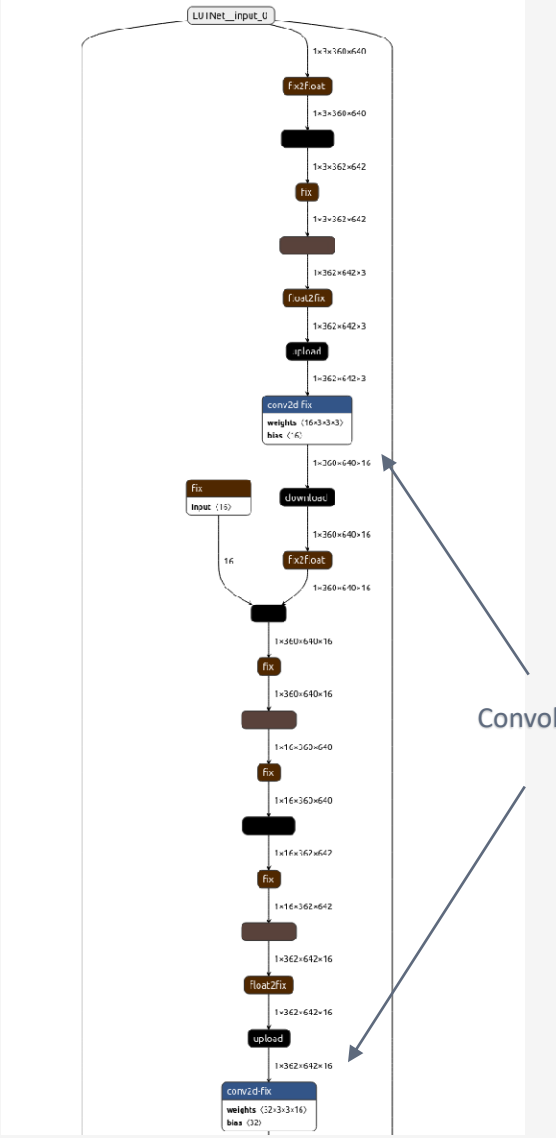
Results

- “Oh, that sounds easy”– No!
  - It took us 9 months to get to a deployed CNN
  - Four (unanswered) bug reports to Xilinx
  - Meaningless error messages
  - Countless workarounds
- Input: 360x640 Pixel
  - Runtime CPU: 0,1s
  - Runtime DPU: 5s
  - Runtime for 60 fps: 0,017s

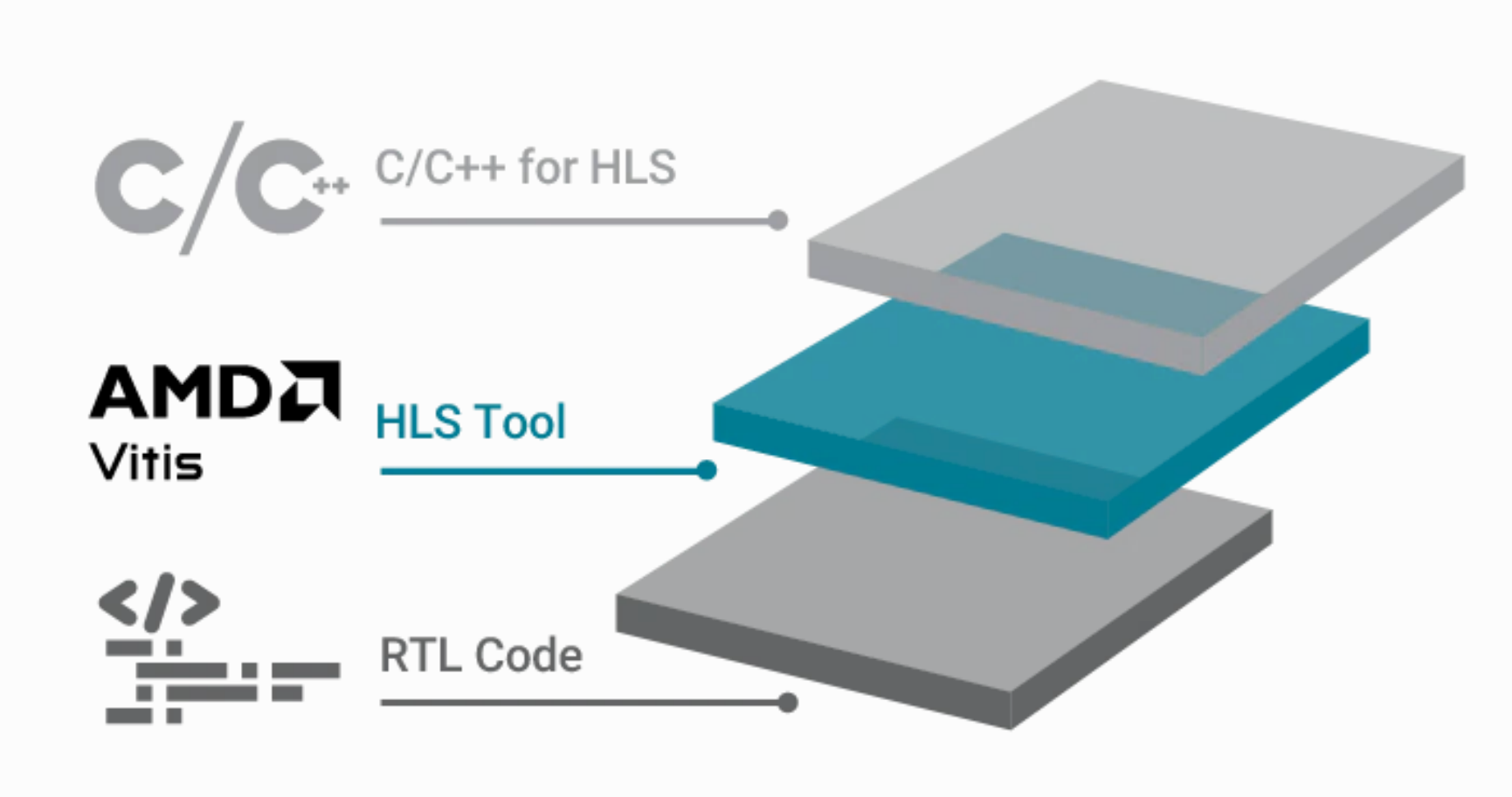


Results

- “Oh, that sounds easy”– No!
  - It took us 9 months to get to a deployed CNN
  - Four (unanswered) bug reports to Xilinx
  - Meaningless error messages
  - Countless workarounds
- Input: 360x640 Pixel
  - Runtime CPU: 0,1s
  - Runtime DPU: 5s
  - Runtime for 60 fps: 0,017s
- Too many operators of the CNN were not supported by the DPU
- In between the data is …
  - converted from float to fix point (and vice versa)
  - moved to and from the shared memory space of the DPU



The HLS or „AI is just an Algorithm“ Solution



High-Level Synthesis (HLS)

- "HLS": Development tool for converting high-level language code into RTL code
- Variants:
  - Vitis HLS (successor to Vivado HLS)
  - Intel HLS
  - HDL Coder
  - Many other tools, some of which are actively developed by universities
- HLS tools are designed to enable the development of **complex hardware modules** at a clear, **component-based level** without the need to immediately focus on the details of low-level implementation

- Component-based Level:

```
void top (a,b,c,d) {
 ...
 func_A(a,b,i1);
 func_B(c,i1,i2);
 func_C(i2,d)

 return d;
}
```

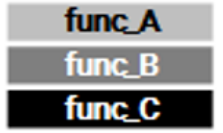


Image source: AMD

- Concurrently operating hardware modules:

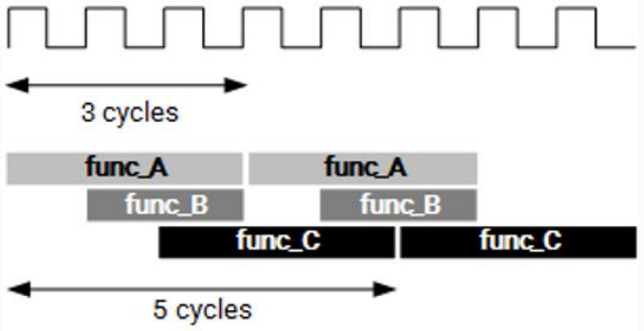


Image source: AMD

The Pros and Cons of HLS

- Pros
  - Lower entry barrier
  - Shorter implementation time
  - Automatic adaptation of the interfaces between function modules (bit width, use of FIFOs or BRAM)
  - Shorter test times (HLS tools often contain a testbench for co-simulation of the generated HDL code, which uses C/C++ code)
  - Test results include timing, resource usage and the precision of fixed-point approximations
- Cons
  - To write powerful code, you need to have hardware knowledge and consider the same things as in HDL (example: although C/C++ is used, image rotation is not a trivial task to implement)

- Component-based Level:

```
void top (a,b,c,d) {
  ...
  func_A(a,b,i1);
  func_B(c,i1,i2);
  func_C(i2,d)

  return d;
}
```

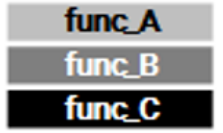


Image source: AMD

- Concurrently operating hardware modules:

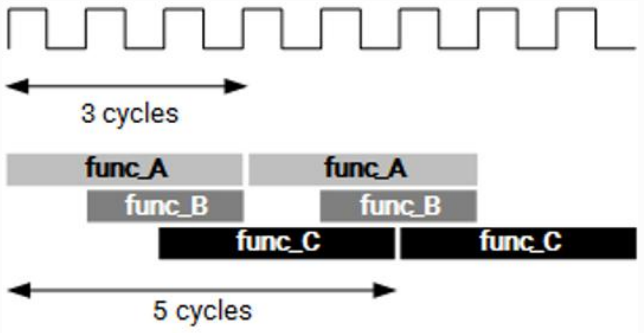


Image source: AMD

Application example: CNN for FPGA

- Customized functions can be implemented using HLS. However, additional challenges arise in this development pipeline:
  - It must be considered that the CNN software model usually uses **floating point** numbers, whereas the CNN hardware model uses **fixed-point** numbers.
  - The **interface** for transferring the parameters of the CNN software model to the mirrored CNN hardware model must be harmonized.
  - **Quantization-aware training or Post-Training Quantization** should be implemented for optimal results
- It should be noted that there are already tools for this, such as Brevitas and FINN, which also use a common exchange standard (ONNX).
- However, these tools are still under development, and they also have limitations.

- CNN example:

Application example: CNN for FPGA

- Tasks on software side:
  - Development of a CNN model to fulfil an intended task
  - Although any function can in principle be programmed using HLS, not every function can be implemented on hardware with limited resources.

    → CNN functions must be adapted or selected appropriately
  - Quantization-aware training can require the use of extended functionalities of known tools such as Tensor Flow and Pytorch or the use of specialized tools such as Brevitas.

- CNN example:

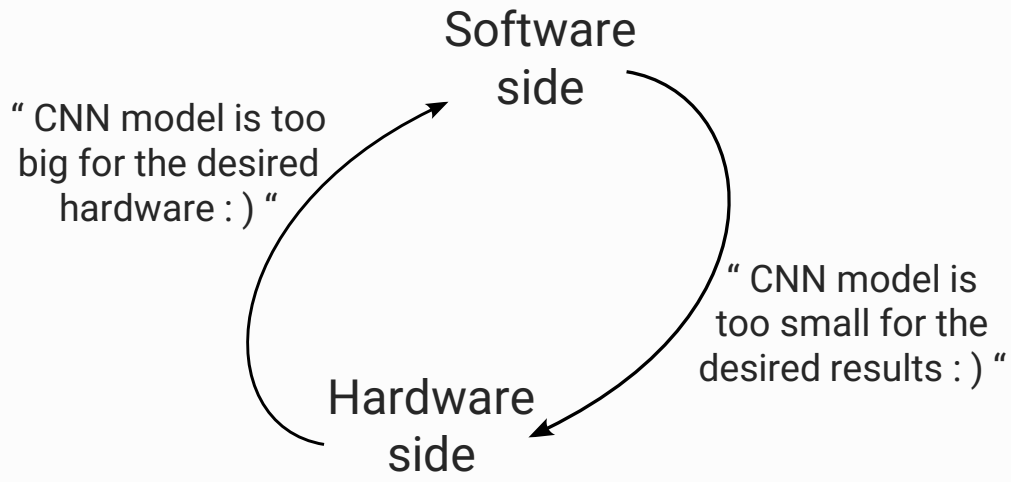
Application example: CNN for FPGA

- Tasks on hardware side:
  - Programming the selected CNN modules with regard to resources, timing and accuracy
  - Implementing and testing of the CNN modules on the hardware
  - As the software side may generate modules faster, clear communication between the hardware and software sides is essential

- CNN example:

Application example: CNN for FPGA

- The development process is organized as a loop:
  - Software side informs about:
    - Changes to the CNN
    - Achievable results
  - Hardware side informs about:
    - Resource utilization of the FPGA (LUTs, DSPs)
    - Runtime constraints (e.g., 20ms for 50fps)
    - Feasible modules
    - Achieved accuracy of the fixed-point numbers



LYNXTechnik AG

Thank You

www.lynx-technik.com