Real Time Video Upscaling

Joshua Lau, James Garcia, Kunal Barde (B0)

Use Case/Application Area

Problem:

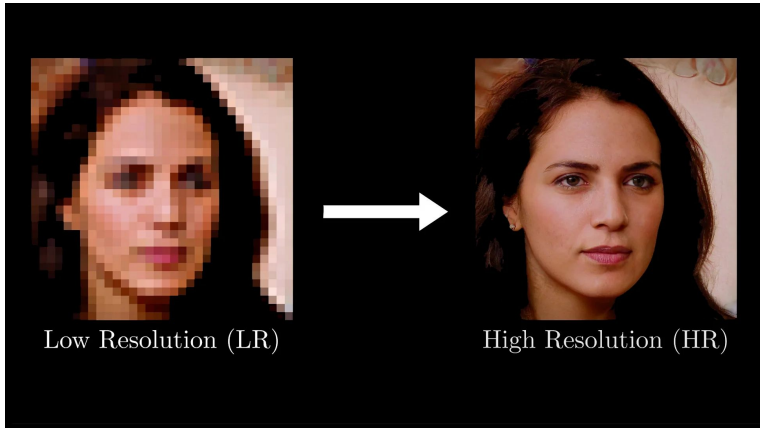
- People want to watch old home videos and movies
- Don't want to plan for upscaling videos ahead of time
- Don't want/know how to do it online or on a computer

Solution:

Plug-and-play, real-time, video super-resolution device - Enhancing 240p videos to 1080p.

Requirements:

- Scaling Factor of 4.5x
- SSIM > 0.66
- Latency < 60ms
- Throughput ≥ 30FPS



Solution Approach

- SRCNN-Ex
  - Software profiling
- Ultra96v2 FPGA
  - Hardware acceleration
- Ultra96v2 ARM Core
  - I/O and profiling
- Meet specifications

Pitfalls, aka Hardware can be Hard

- Model too large for optimal solution on Ultra96v2
  - Our ex-SRCNN -> *mathematically impossible* to implement, while meeting specifications.
- As Vitis giveth, Vitis taketh away
  - Chosen because a good solution is easy to make with Vitis
  - Tool is still rather opaque in its implementation, optimisation, etc.
  - Leads to longer iteration at tail-end; fine-tuning less obvious/methodical
- PetaLinux for Embedded ARM is fragile
  - Many one-off oddities
  - Video I/O requires manual patches of version-dependent known issues in the OS; documentation tends to be version-independent or less granular than is relevant

Real-Time I/O Functionality

- Iterative mechanism to stream frames into CNN kernel on FPGA
- Batching functionality present to consume multiple frames at a time
- Experimenting between interpolation methods on the host program vs. more expanded cnn architectures
- Critical part of our real-time system which works but needs finer tuning for batch based workloads

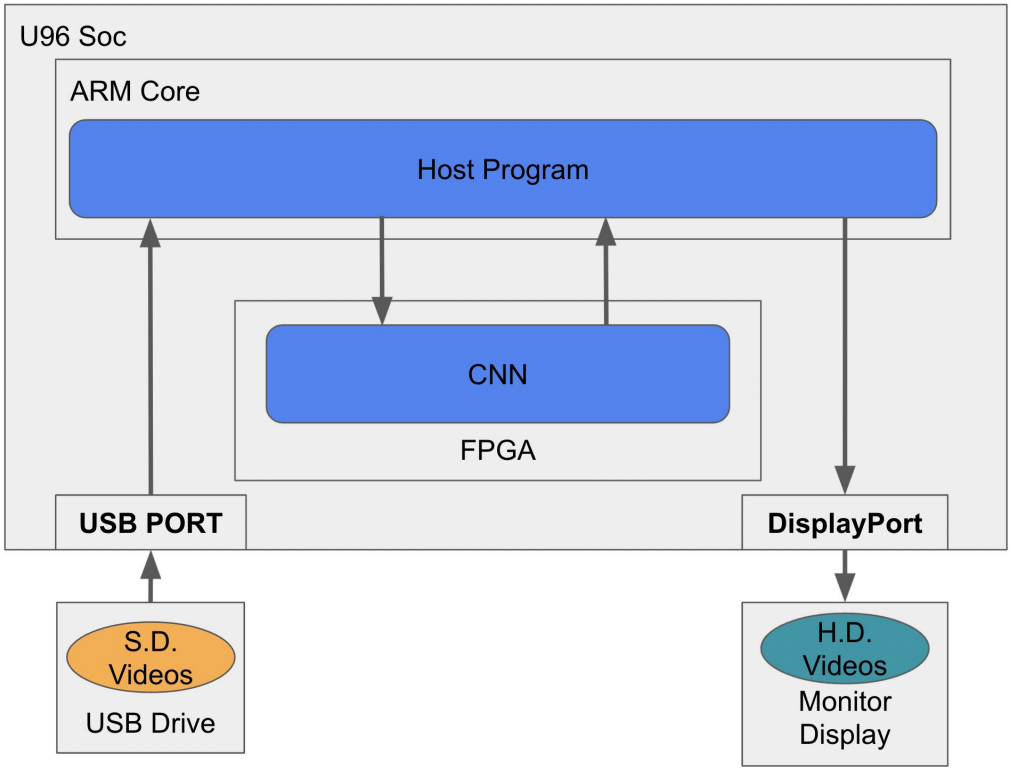
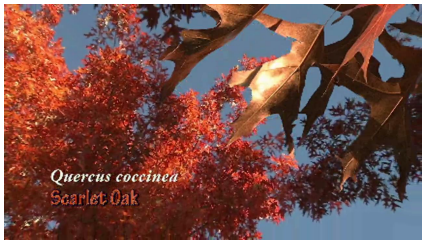


Image Quality

- SSIM requirement met using SRCNN-Ex implementation
- Testing on GPU seemed to show CNN would be fast enough, however, timings did not work on FPGA.



Timing

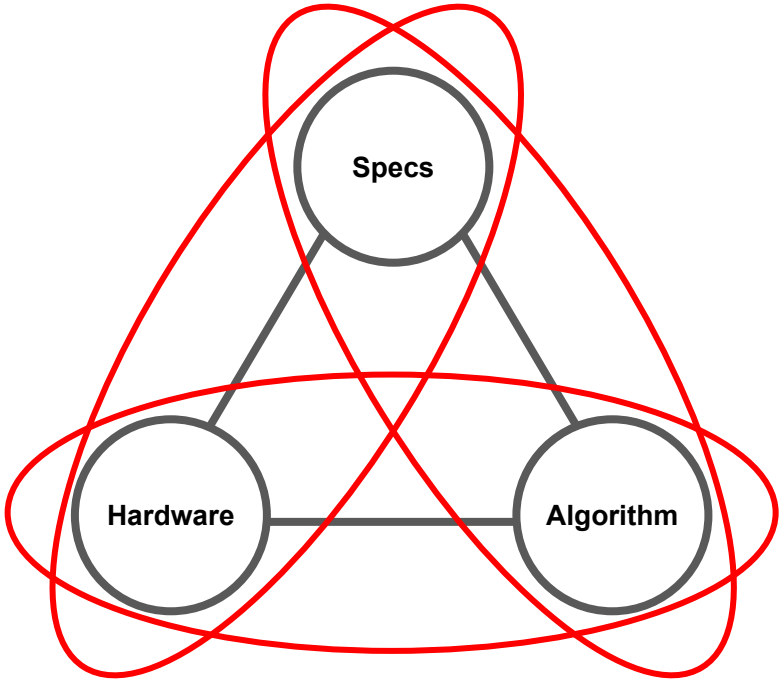
- Much more difficult to meet
  - Impossible in some cases
- Iterative host-side frame processing causes latency-bound throughput
- Forced us to do overarching design trade-offs

Model	**Single-Frame End-to-End Latency**
SRCNN-Ex (U96)	115.401s (5.92s theoretical bound)
FSRCNN (U96)	2.989s (64ms theoretical bound)
SRCNN-Ex (embedded device avg.)	~200s

Tradeoff

- Non-tight coupling of HW and Algo design led to a “pick-two-triangle” trade-off

1. Fixing HW and an algorithm leads to missing specs
2. Fixing an algorithm and the specs requires different HW
3. Fixing HW and specs requires a different algorithm



Change in Approach

Main change: Going from model based on SRCNN-Ex to FSRCNN-s.

SRCNN-Ex

- Shallow, but operates on the pre-upscaled frame
- More robust - less variance in SSIM
- Highest SSIM out of implementations considered

FSRCNN-s

- Deeper, but operates on the native frame. Less data processed, increase in throughput
- 2 orders of magnitude fewer computations than ex-SRCNN
- Decreased SSIM compared to other implementations

Complete Solution - Specifications Met

- Scaling Factor and SSIM have never been the problem for our implementations
- Failed to meet either timing requirement in both FPGA implementations

	SRCNN-Ex	FSRCNN	FSRCNN-s
Scaling Factor = 4.5	4.5	4	4
SSIM > 0.66	~0.751	~0.734	~0.715
Latency < 60ms	115401ms	2989ms	20ms (ideal)
Throughput ≥ 30FPS	~8.7mFPS	~0.33FPS	49.7FPS (ideal)

Complete Solution - User Experience

Initialisation:

- Plug in the Ultra-96
- Power up the Ultra-96
- Connect Ultra-96 to compatible monitor / display

User Flow:

- Plug USB into Ultra-96, containing single video file
- Launch upscaling program
- Upscaled video is displayed on external monitor / display

Revised Schedule

Hardware											
Acquire Ultra96	~~JSG~~										
Acquire Peripherals	~~JSG~~										
Research I/O	~~JSG~~	~~JSG~~	~~JSG + KB~~								
Implement I/O			KB + JSG	KB	KB	KB	KB	KB			
Test I/O			JSG + KB	JSG	JSG	JSG	JSG	JSG			
Get Comms between ARM Core and FPGA		~~JSG~~									
Write Math Functions for CNN in Vitis HLS				~~JSG~~	~~JSG~~	~~JSG~~	~~JSG~~	~~JSG~~			
Get full-sized model to synth on the board properly								JSG	JSG	JSG	JSG
Optimize tile sizes								JSG	JSG	JSG	JSG
Speed up w/ HLS pragmas								JSG	JSG	JSG	JSG
Validate HW				KB	KB	KB	KB	JSG+KB	JSG+KB		
Port SW model onto FPGA							~~JSG~~				
Validating FPGA model against SW model							KB	KB	KB		
Time Benchmarking								KB	KB		
LV								KB	KB	KB	KB
screen display								KB +JSG	KB +JSG	KB +JSG	KB +JSG
video routing								KB	KB	KB	KB
Software											
Research DSP vs CNN models	~~ALL~~										
Acquire AWS Credits		~~KB~~									
Setup AWS		~~KB~~	KB + JL								
Acquire Dataset	~~JL~~										
Familiarize VMAF Documentation	~~JL~~	~~JL~~									
Research specific CNN models		~~KB + JL~~	KB + JL								
Benchmark VMAF		~~JSG + JL~~									
Research SSIM	~~KB~~										
Benchmark SSIM		~~KB + JL~~									
Benchmark CNN Models		~~JL + JSG~~	~~JL~~								
Develop Python Code for Training			~~JL + KB~~	~~JL~~	~~JL~~						
Train Model				~~JL~~	~~JL~~	~~JL~~	~~JL~~				
Test/Evaluate Model				JL	JL	JL	JL				
Further optimizing weights (keeping hyperparams const so HW model stays const)							JL	JL	JL		
Misc											
Slack/Slop										ALL	
Milestones											
Proposal Presentation	~~JSG~~										
Design Presentation			~~KB~~								
Design Review Report			~~ALL~~	~~ALL~~							
Interim Demo							~~ALL~~	~~ALL~~			
Final Presentation											JL

JSG
JL
KB

James
Joshua
Kunal