ECE 4300 | Computer Architecture

Image Processing Performance | FPGA vs CPU

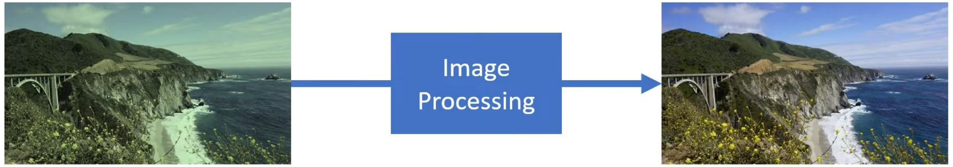
Jessalin Jiangkhov, Rick Unite, Izaiah Adams

Why we chose this?

- Combines current learning of Computer Architecture with previous experience of digital design
- Hardware and Software focused
- Easily doable
  - Components already in possession
- Real Life Relevance
  - FPGAs used for parallelism and low latency
  - CPUs used for flexibility, easy to program, not as efficient for data-parallel tasks
  - Understanding trade-off prepares for modern computing challenges in AI hardware, embedded systems and accelerators

What is Image Processing?

Also known as Digital Image Processing, it is a process of analyzing and manipulating images digitally.



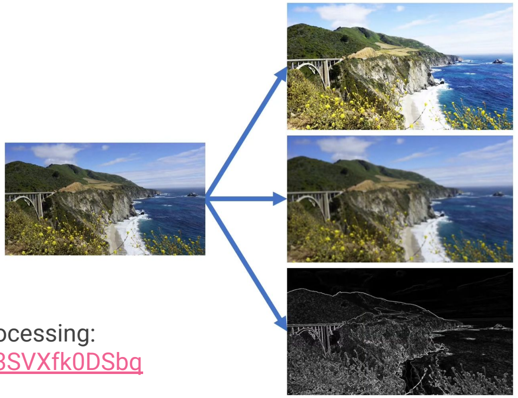


Image from Computer Vision vs Image Processing: https://youtu.be/9-8Js62wzQs?si=HMN_i3SVXfk0DSbq

Objective

Testing and comparing the speed, power consumption, resource utilization, and output quality of an FPGA producing an image vs a CPU producing an image

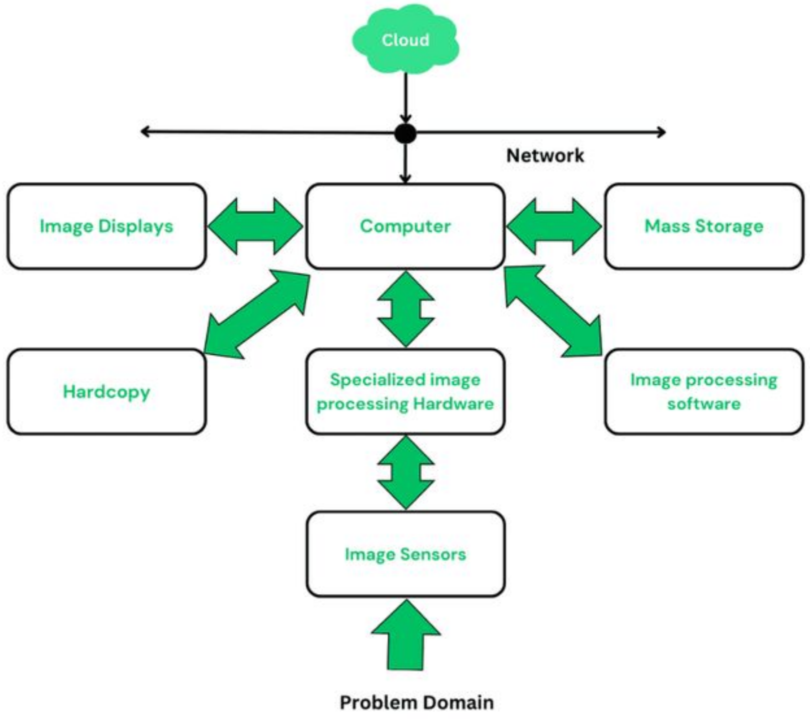


Image from geeksforgeeks: https://www.geeksforgeeks.org/electronics-engineering/what-is-image-processing/

Hardware Background | FPGA

Field-programmable gate arrays (FPGA)

- Integrated circuits
- Can be reconfigured to users' desires
- Contain programmable logic blocks
- Flexible usage
- Hardware-timed speed and reliability
- Parallelism
- Found in:
    - Data Centers, aerospace & defense systems, medical devices, etc

Hardware Background | CPU

Central Processing Unit (CPU)

- “Brain” of a computer
- Executes instructions
- Parallel Processing
    - Modern CPUs utilize multicore processing to execute tasks simultaneously
- Clock Signal used for operation synchronization
- Multithreading
    - A single core can handle multiple sequences of instructions (multitasking)
- Found in many common devices
    - Computers, smartphones, smart TVs, etc

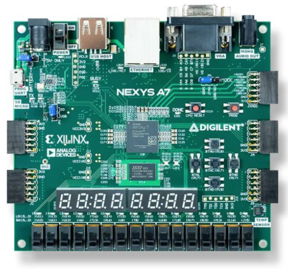
Hardware Specs

FPGA - NEXYS A7 100T

- 128MiB DDR2
- Serial Flash
- 4860 Kbits of fast block RAM
- Six clock management tiles
- Internal clock speed >450MHz
- 12-Bit VGA Output

CPU - Ryzen 9 7600x

- Zen 4 Processor Architecture
- 6 CPU Cores
- 12 Threads
- Max Boost Clock of 5.3GHz
- Base clock of 4.7GHz
- Graphics Model: AMD Radeon Graphics
- Graphics Core Count: 2
- Graphics Freq.: 2200MHz



Other Necessary Components

- Monitor capable of VGA (Video Graphics Array) Display for the FPGA output
- Working computer capable of using the desired CPU
- OV7670 camera module for FPGA Camera capabilities
- Vivado 2025.1 will be used to program FPGA

OV7670 Camera Module

Testing specifics: Image processing algorithms

- Simple, compute-heavy kernels - ideal for both CPU and FPGA
  - Grayscale conversion
    - RGB -> grayscale using weighted sum (simple, tests memory throughput)
  - Image convolution
    - Apply filters like blur, sharpen, etc. (great for pipeline parallelism)
  - Sobel edge detection
    - Gradient computation (x, y directions) (common benchmark for edge detection)
  - Gaussian blur
    - Smooth image; uses 2D convolution (compute-intensive, good for throughput test)
  - Thresholding / binarization
    - Convert image to black/white by pixel intensity (easy control and branching test)
  - Matrix multiplication (for comparison)
    - Core operation for CNN-like workloads (useful for performance scaling)

Testing specifics: Performance metrics

- Measure quantitatively comparable results for both platforms:
  - Execution Time / Latency
    - Wall-clock timing (time.perf_counter() (CPU) / on-chip timer (FPGA))
  - Throughput (FPS)
    - 1 / frame time (Manual calculation)
  - Resource Utilization
    - LUTs, FFs, BRAM, DSP usage (Vivado/Quartus synthesis report)
  - Power consumption
    - Estimated power (Xilinx Power Analyzer / Quartus Power tool)
  - Energy per frame
    - Power x time (Derived metric)
  - Clock frequency
    - Achieved timing (MHz) (Synthesis report)

Testing specifics: Datasets

- Open source dataset:
  - Kodak dataset (24 uncompressed images)
  - Berkeley Segmentation Dataset (BSDS500)
  - COCO / ImageNet samples (downscaled)
  - Or capture our own test images
  - (MUST USE SAME IMAGES FOR BOTH CPU AND FPGA)

Testing Process

1.) Select algorithm
2.) Implement on CPU using OpenCV
3.) Implement on FPGA using HLS (C code -> hardware)
4.) Run both on same dataset (same image size, e.g 512 x 512 or 1080p)
5.) Record results (time, fps, LUTs, power)
6.) Plot comparison graphs: latency vs throughput, resource vs performance

Testing Process | FPGA

1. Testing the FPGAs capabilities using an algorithm to produce an image to a VGA Monitor
   a. Will reference this tutorial: https://www.instructables.com/Image-from-FPGA-to-VGA/
2. Testing the FPGAs capabilities of outputting the live feed of a camera
   a. Will reference this GitHub: https://github.com/amsacks/OV7670-camera
      i. User's Demo Video: https://youtu.be/PbZsmB2INCU?si=O62HTviTFL-5ToOY
3. Record Data and Outputs
   a. Take pictures of Image / Video Outputs
   b. Note device temperature

Current Status

- Most components obtained
  - Further research was necessary
  - Must order camera for FPGA
- Testing will be conducted soon
  - Availability Issues
- Benchmarking and other tests will be conducted on separate devices
  - Results will later be compared
- Recovered from a hardware failure

Data and Results

- Typical Trend (Hypothetical)
    - FPGA: 3–10× faster for small kernels when pipelined properly.
    - CPU: Easier to implement, but slower and more power-hungry.
    - FPGA: Requires more setup but demonstrates parallelism.

Issues and Other Possible Variables

- Hardware-related issues
  - FPGA resource limitations: not enough LUTs, BRAM, or DSP slices for large images or complex filters
  - Power and Heat Constraints: Running the FPGA at high frequency or full utilization may cause overheating
  - Hardware interface or Driver Issues: Data transfer between CPU and FPGA can be buggy or slow
- Software and Toolchain problems
  - Toolchain complexity: Steep learning curves
  - Simulation vs Real Hardware Mismatch: Design works in simulation but fails on the actual board
  - Timing and Optimization: FPGA synthesis succeeds but achieved frequency is lower than expected
  - CPU Version Optimization Bias: CPU version might run slower or faster depending on how we implement it
- Benchmarking & Measurement Pitfalls
  - Unfair Comparison: (different resolutions, data movement overhead not included)
  - Inaccurate Timing: Measuring performance incorrectly (I/O or OS overhead)
  - Power Estimation Uncertainty: Estimating FPGA power = hard without a physical meter
- Algorithmic and Data Issues
  - Large Image Sizes: might exceed FPGA memory
  - Fixed-Point Precision Errors: converting floating point to fixed point in FPGA = output accuracy may degrade