# Accelerating compact digital image processing with real-time haze removal

ECE 8893 – Parallel Programming for FPGAs
Instructor: Prof. Callie Hao
Spring 2022
Final Presentation

ELECTRICAL [+] COMPUTER
ENGINEERING

Aloysius Leon Abreo
aabreo3@gatech.edu

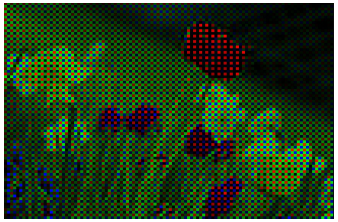
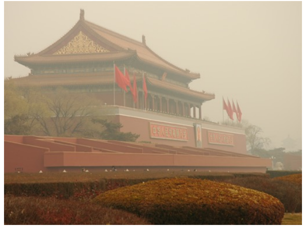
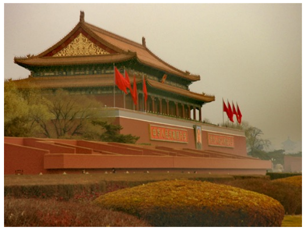
Ankur Bindal
abindal8@gatech.edu

Anirudh Gorantla Nagaraja
anagaraja7@gatech.edu

Georgia Tech School of Electrical and Computer Engineering

# Motivation

- Image processing systems are fundamental to aerial imagery, computer vision, video analysis and recognition.
- Fundamental components of an image pipeline:

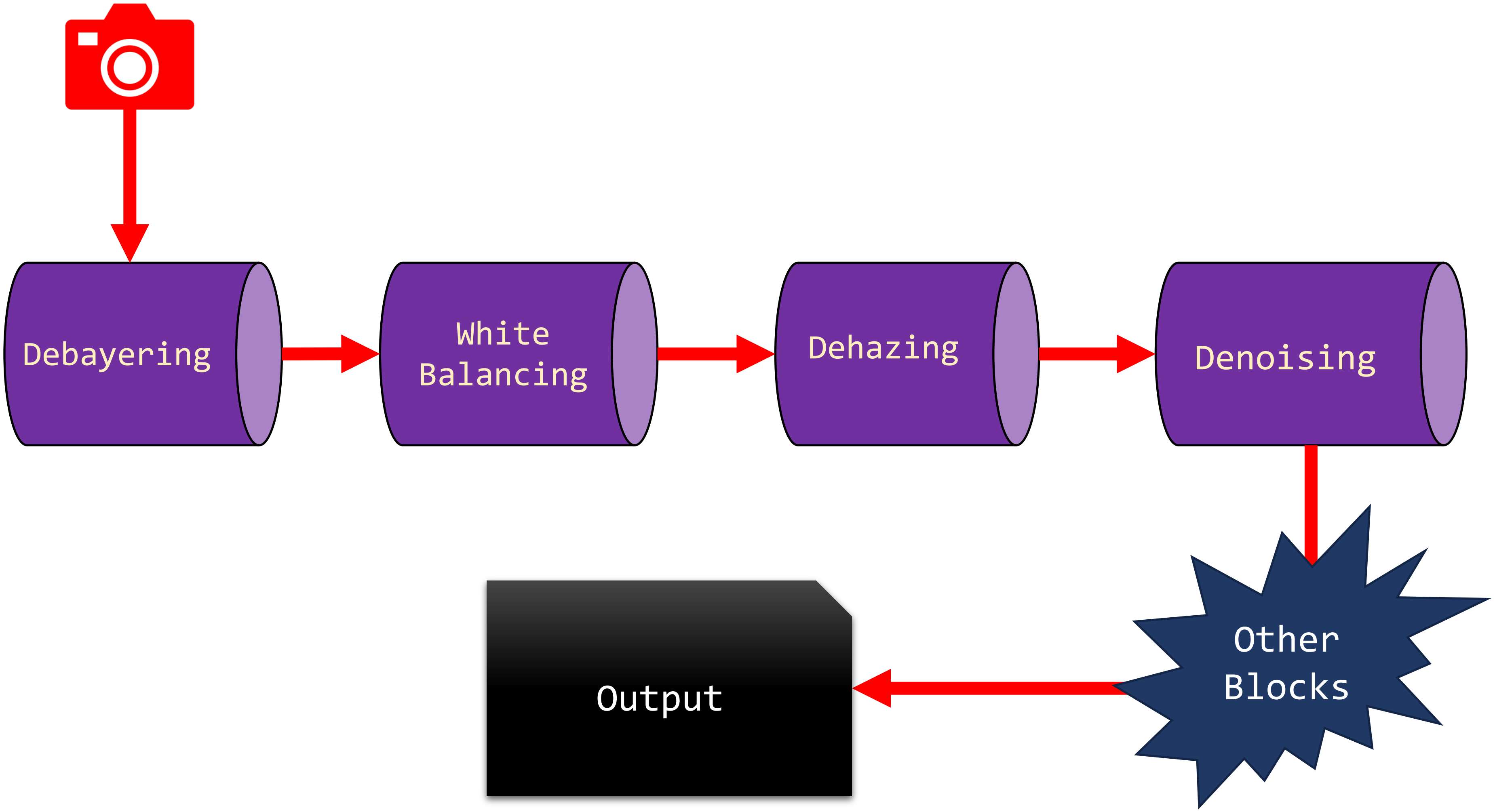
| | |
|---|---|
| **Debayering** |   |
| **White Balancing** |   |
| **Dehazing** |   |

https://en.wikipedia.org/wiki/Demosaicing
https://www.tuscanyphotographytours.com/articles/white-balance-in-photography/
http://ivc.uwaterloo.ca/database/Dehaze/Dehaze-Database.html

# Why FPGAs?

- Image processing algorithms are executed sequentially.
  - Scope for parallelization → **FPGAs!**
- Digital Image processing applications benefit from real-time processing.
  - Design an image processing accelerator → **FPGAs!**
- GPUs are traditionally used to accelerate these individual image processing algorithms with high power costs.
  - Low power and design cost → **FPGAs!**
- Newer image processing algorithms need to be validated.
  - Design complex algorithms easily using HLS → **FPGAs!**
- Modular IPs may be added to the processing pipeline for enhanced processing.
  - No struggling with complex RTL IP implementation → **FPGAs!**

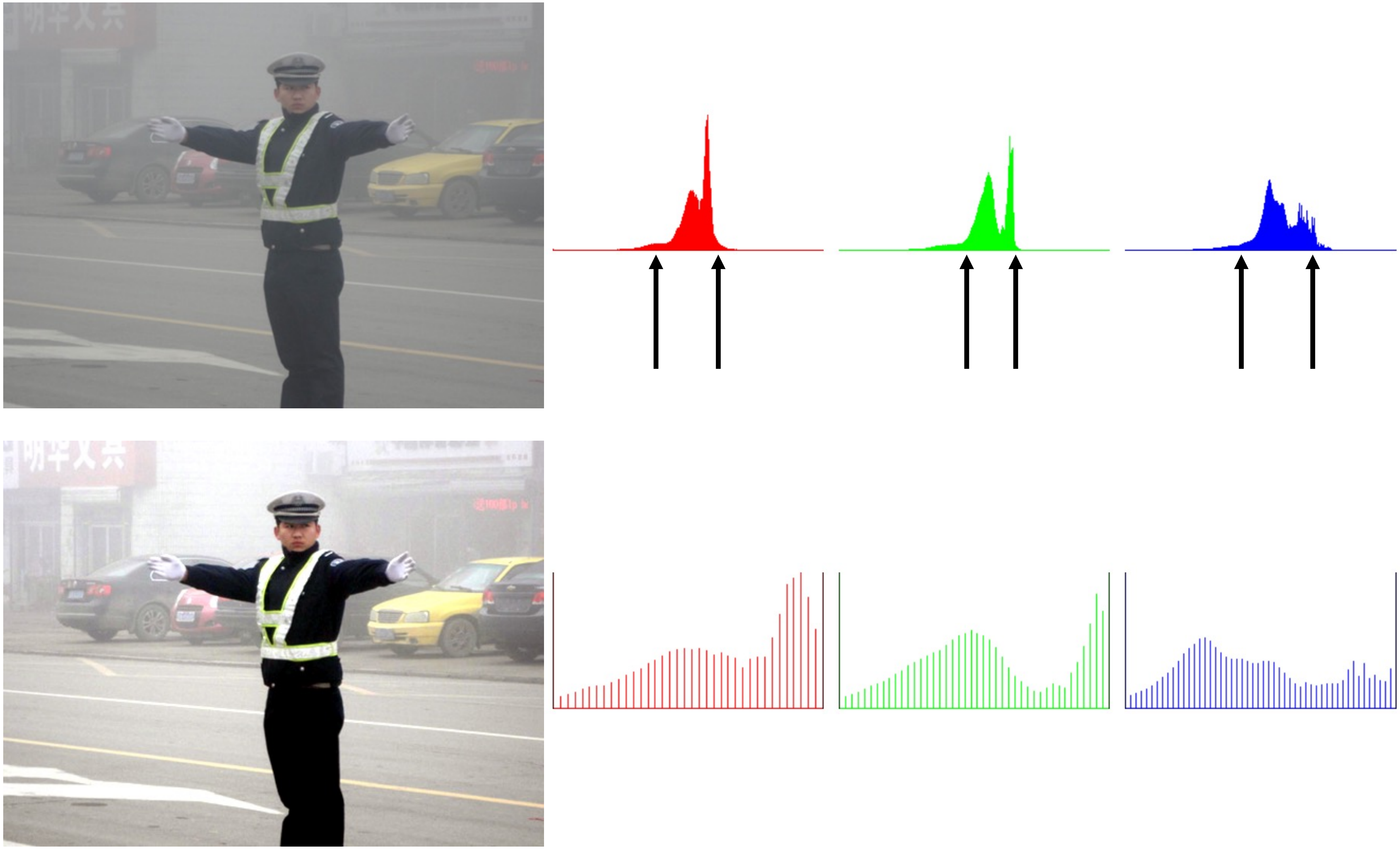
# Implementation: Image Pipeline



# Implementation: Debayering

- Different Kernels applied based on row/column index
- Initially:
  - Many if/else conditions prevented any optimizations
- Debayer Kernels are much smaller compared to CNN.
  - Combined kernels to one
  - Used row/column indexes to mask part of kernel

# Implementation: White Balancing



# Implementation: White Balancing

- Need to calculate global data before applying local transformation.
- How do you get global data without stalling the pipeline?
- Use the previous frame’s global data to apply transformation on current frame.
- Assumption: the tone difference between adjacent frames isn’t too different.
- Collect global data while de-bayering and wait to accumulate all tile data.
- Apply local transformation on a tile basis.

# Implementation: Dehazing

- Dehazing a white balanced image involves:
  1. Calculating the **Min RGB value** for a given pixel in a tile.
  2. Calculating the **Median Value** for all the Min RGB values.
     - This is the expected value of the Haze = **Hazy Particle Map**
  3. Subtract the Hazy Particle Map from the image to get the recovered scene radiance (dehazed).
- Calculating Particle Map → needs 3 median calculations

| | | | | | | |
|---|---|---|---|---|---|---|
| 192 | 245 | 178 | 220 | 64 | 234 | 14 |
| 70 | 87 | 227 | 65 | 157 | 73 | 135 |
| 173 | 149 | 245 | 220 | 121 | 193 | 199 |
| 215 | 57 | 140 | 213 | 90 | 215 | 238 |
| 41 | 192 | 35 | 237 | 212 | 97 | 33 |
| 30 | 65 | 38 | 89 | 149 | 145 | 145 |
| 127 | 129 | 66 | 50 | 140 | 19 | 120 |

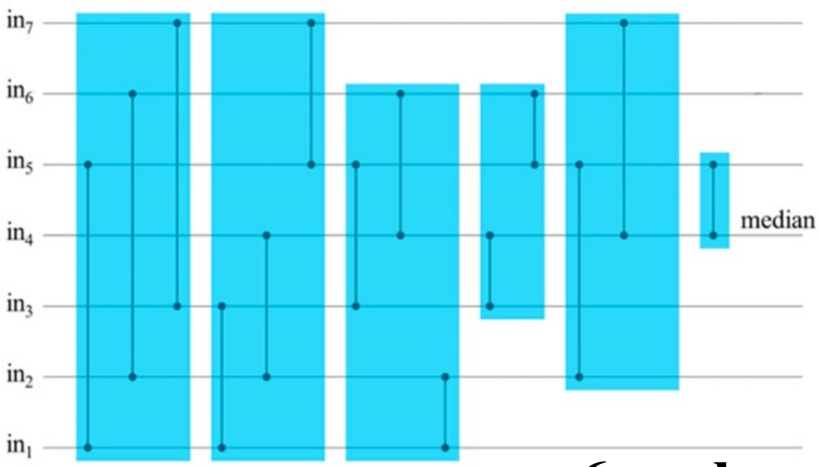
| | | | | | | |
|---|---|---|---|---|---|---|
| 192 | 245 | 178 | 220 | 64 | 234 | 14 |
| 70 | 87 | 227 | 65 | 157 | 73 | 135 |
| 173 | 149 | 245 | 220 | 121 | 193 | 199 |
| 215 | 57 | 140 | 213 | 90 | 215 | 238 |
| 41 | 192 | 35 | 237 | 212 | 97 | 33 |
| 30 | 65 | 38 | 89 | 149 | 145 | 145 |
| 127 | 129 | 66 | 50 | 140 | 19 | 120 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 192 | 245 | 178 | 220 | 64 | 234 | 14 |
| 70 | 87 | 227 | 65 | 157 | 73 | 135 |
| 173 | 149 | 245 | 220 | 121 | 193 | 199 |
| 215 | 57 | 140 | 213 | 90 | 215 | 238 |
| 41 | 192 | 35 | 237 | 212 | 97 | 33 |
| 30 | 65 | 38 | 89 | 149 | 145 | 145 |
| 127 | 129 | 66 | 50 | 140 | 19 | 120 |

**HLS Functions: mHMF_49 mHMF_13 mHMF_13 → All use mHMF_7!**

# Implementation: Dehazing (Odd-Even Sorting)

- Median Calculation – **mHMF_7** – based on Batcher's Sorting Network.




**6 cycles**

```
void compare_and_swap_unit(INT_TYPE &a, INT_TYPE &b)
{
    if (a < b)
    {
        a = a ^ b;
        b = b ^ a;
        a = a ^ b;
    }
}
```

```
void mHMF_7(INT_TYPE input[7], INT_TYPE &median)
{
    compare_and_swap_unit(input[4], input[0]); // Cycle 1
    compare_and_swap_unit(input[5], input[1]);
    compare_and_swap_unit(input[6], input[2]);

    compare_and_swap_unit(input[2], input[0]); // Cycle 2
    compare_and_swap_unit(input[3], input[1]);
    compare_and_swap_unit(input[6], input[4]);

    compare_and_swap_unit(input[4], input[2]); // Cycle 3
    compare_and_swap_unit(input[5], input[3]);
    compare_and_swap_unit(input[1], input[0]);

    compare_and_swap_unit(input[5], input[4]); // Cycle 4
    compare_and_swap_unit(input[3], input[2]);

    compare_and_swap_unit(input[6], input[3]); // Cycle 5
    compare_and_swap_unit(input[4], input[1]);

    compare_and_swap_unit(input[4], input[3]); // Cycle 6

    median = input[3]; // Median
}
```

# Denoising

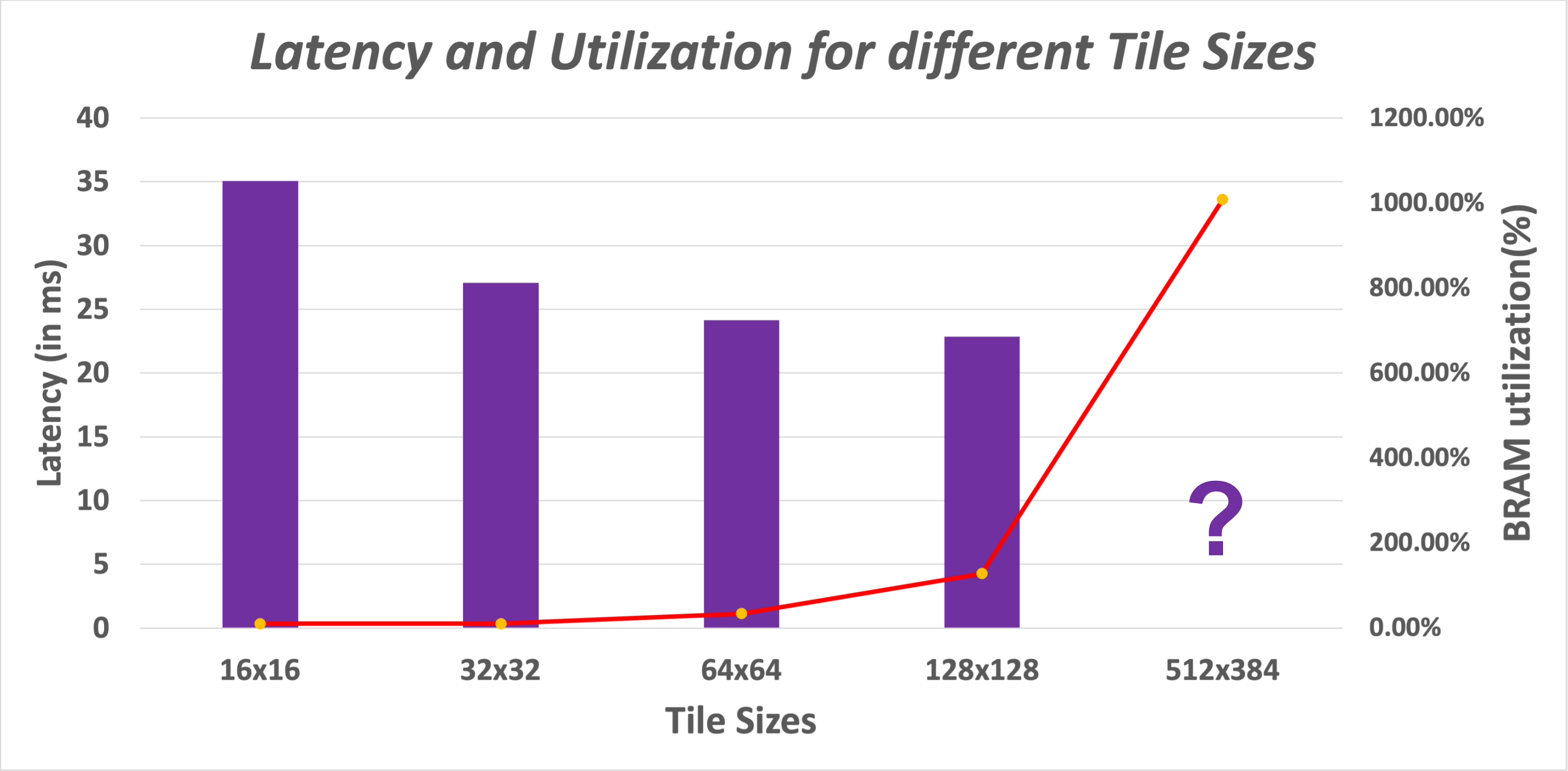
- Dehazing causes the image to lose some scene radiance information.
- Need to add it back using denoising and radiance recovery.
- Denoising smoothens out the recovered scene radiance for further use.

# Optimizations – Input Tiling

- Need Extra Input to process each output tile.
- Overlap Input tile from DRAM
- Process fixed size Output Tile
- Bigger Tile size
  - Faster processing
  - High Utilization
- DSA to get an optimal tile size. We found 64x64 to be best.
- **Other Optimizations:**
  - Partitioned all small arrays.
  - Avoided float division – modified equations accordingly.

# Results

- Real-time latency achieved – 24.156ms. (~**41 frames/sec**)
- Low utilization (**34%**, 6%, 12.5%, 36%)
- Implemented White balancing on the FPGA (Lab 3B).



# Results – Before - After

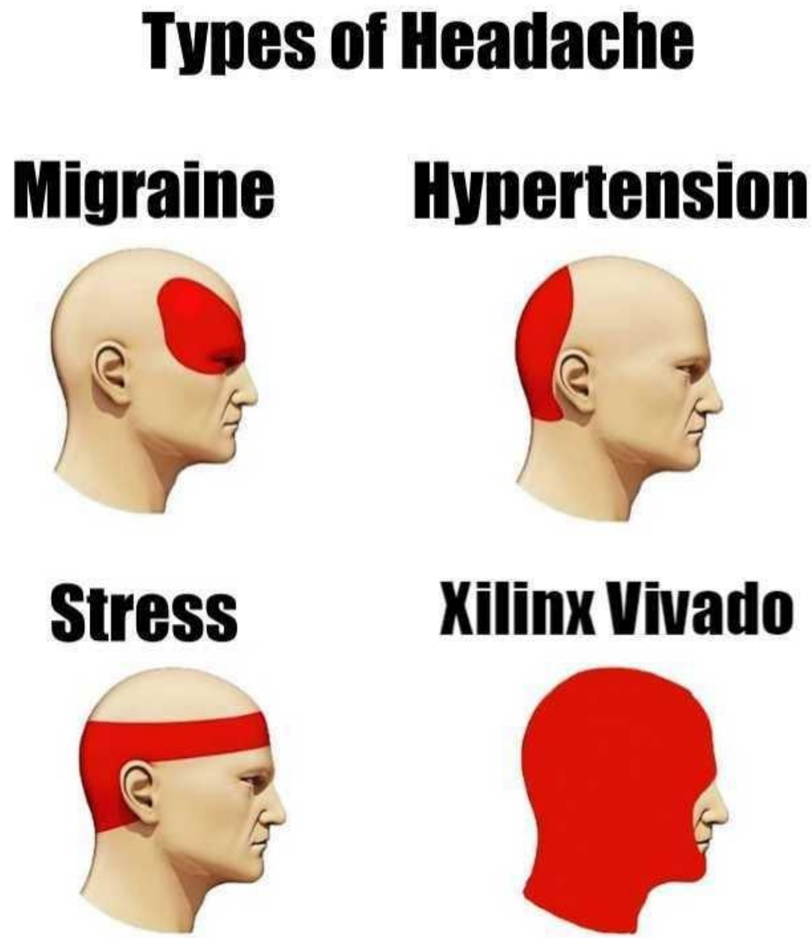
# Results – Before - After

# Scope for improvement & Future work

- Frequent DRAM access is the current bottle neck
  - We can implement a partial buffer to stream read/writes with the DRAM.
- Calculation of global state for White balancing
  - Prevents parallelization.
- Faster median calculation
  - Currently, the code implements median calculation described in the paper (for 7 elements). We can expand the input size to the function to cover larger tile sizes.
- Radiance recovery
  - Radiance recovery is mentioned in the paper as the next step to the dehazed and denoised image. This processing block can be implemented.

# Q & A?


When implementation is almost done and you realize there is a typo


*Thanks for the semester, Professor!*