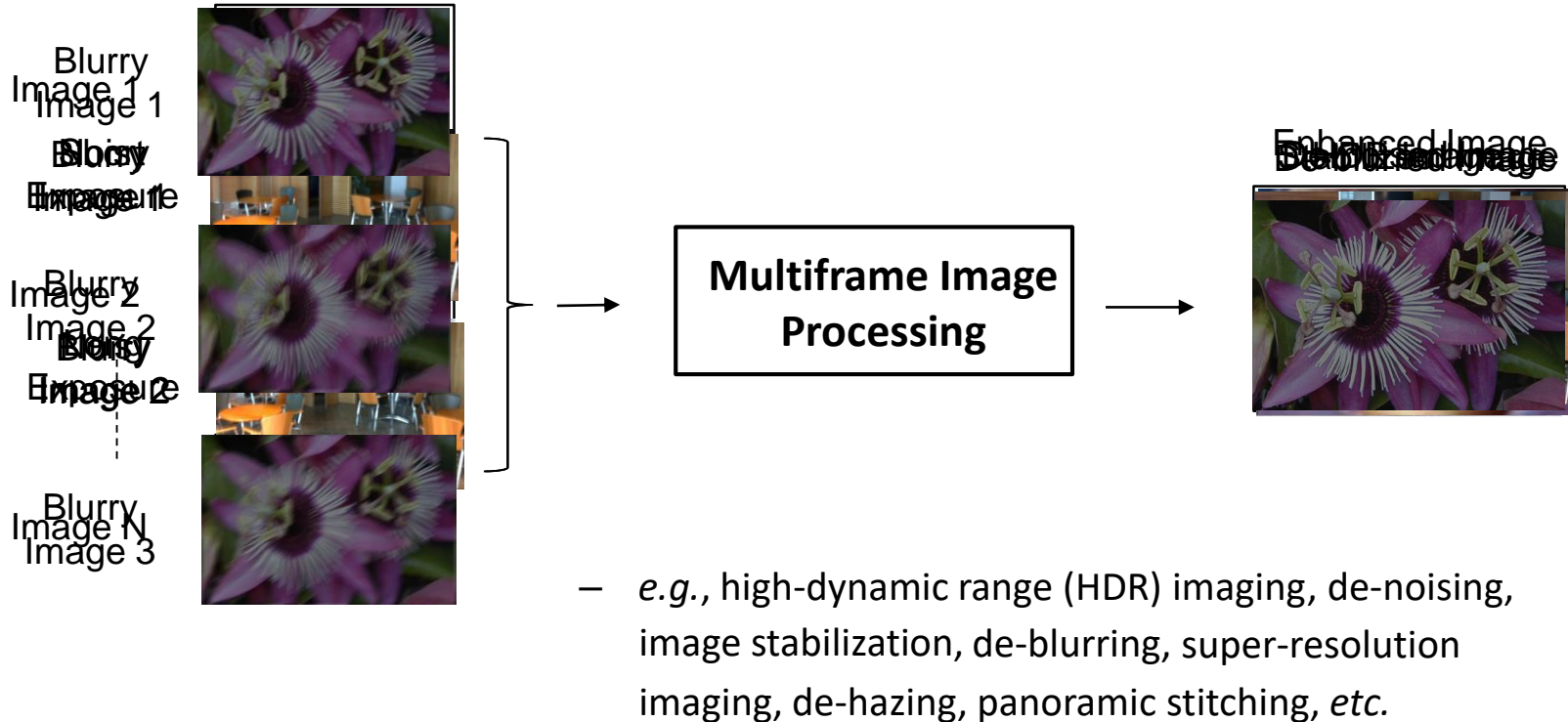# AMETHYST: Image Registration Engine for Multiframe Processing

Mohammed Shoaib, Rich Stoakley, Matt Uyttendayle, and Jie Liu

Sensing and Energy Research Group

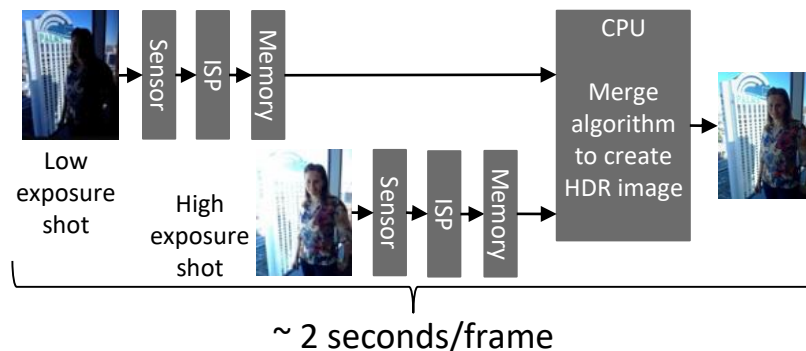**Microsoft Research**

MFP: Why should you care?

Blurry Image 1 / Blurry Image 1 / Exposure 1

Blurry Image 2 / Blurry Image 2 / Exposure 2

Blurry Image N / Blurry Image 3

Multiframe Image Processing

Enhanced Image / De-blurred Image

- *e.g.*, high-dynamic range (HDR) imaging, de-noising, image stabilization, de-blurring, super-resolution imaging, de-hazing, panoramic stitching, *etc.*

Multiframe processing (MFP) enables advanced algorithms for image analysis

Why is it hard?

_E.g._, HDR Photography

Typically, serial processing
→ frame delays cause issues:

Low exposure shot · High exposure shot · Sensor · ISP · Memory · CPU · Merge algorithm to create HDR image

~ 2 seconds/frame

1. Moving objects create artifacts

2. Moving camera also creates artifacts
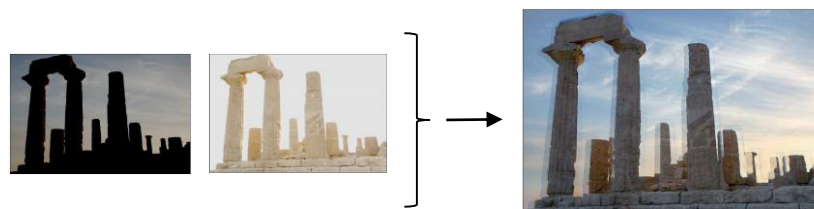
Frame misalignments lead to artifacts in fused image

IMG SOURCES: (1) L. Zhang, CVPR '10 (2) NVIDIA whitepaper, MWC '13

What are some existing solutions?

Solution 1: HDR Capture, *e.g.,* Toshiba T4K05
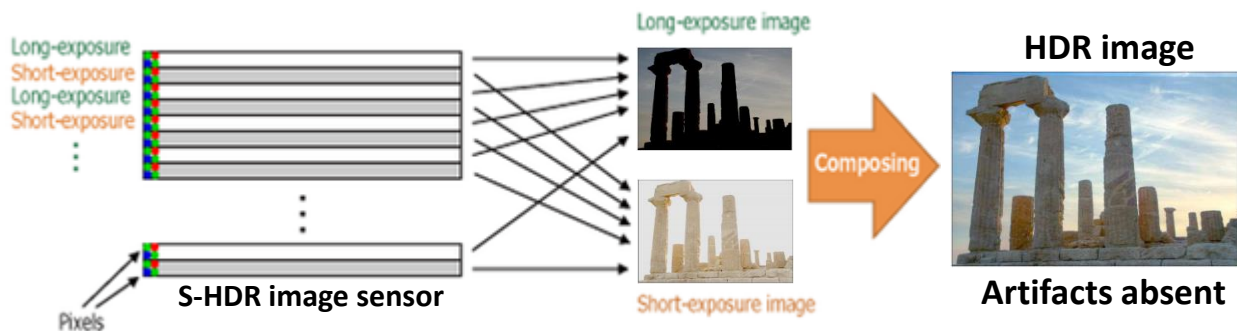
Solution 2: Algorithmic
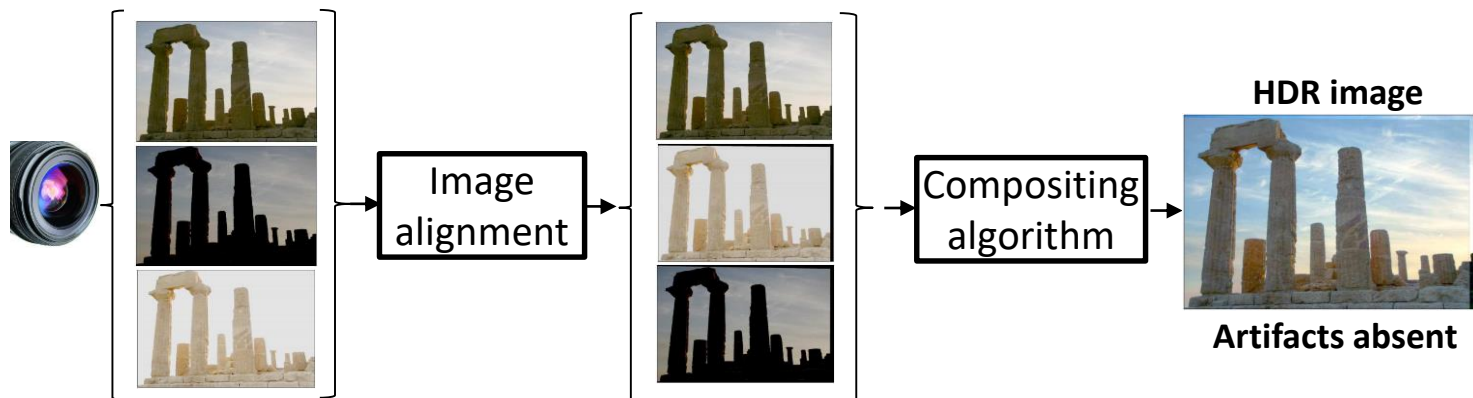
Algorithmic solution is more interesting → needs no hardware change and scales to other applications

What are others doing about it?
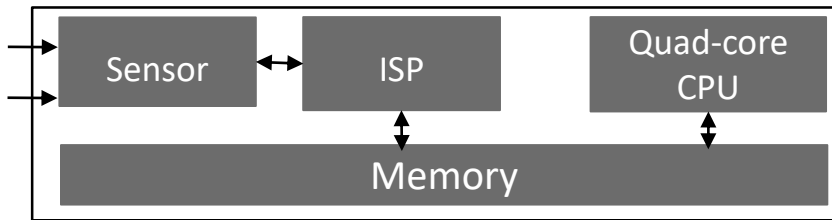
E.g., NVIDIA: Tegra 4 (2014)

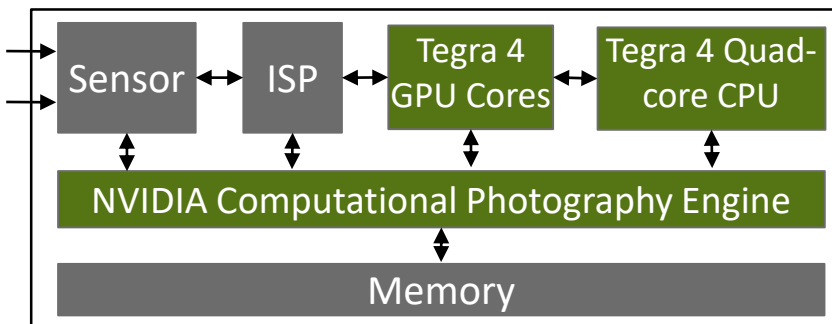**Fig: Camera architecture in current high-end mobile devices**

~ 2 seconds/frame

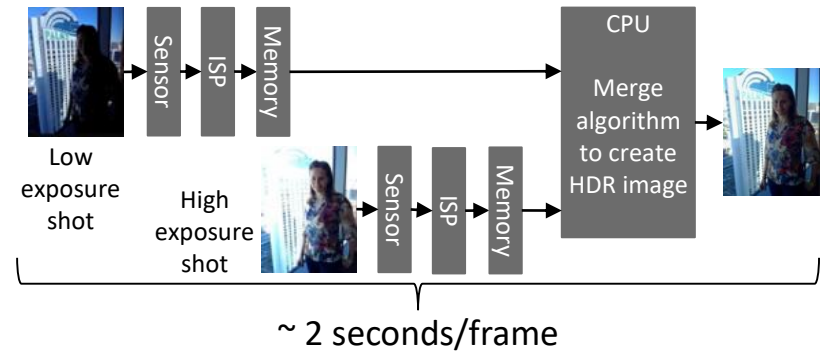**Fig: Chimera: The NVIDIA computational photography arch.**
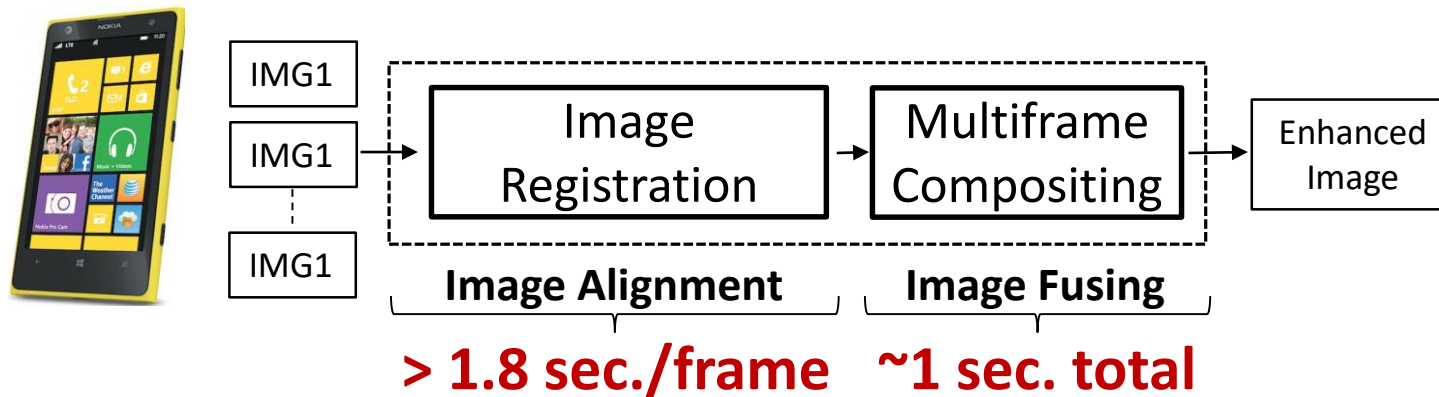

~ 0.2 seconds/frame
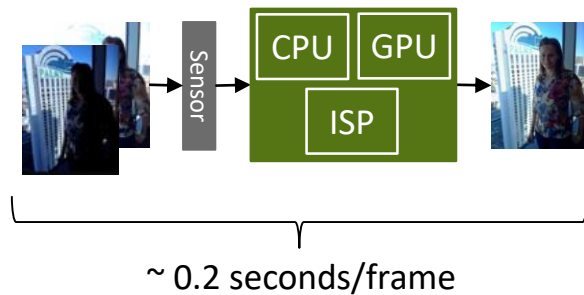
1st real-time HDR, 1st HDR panorama, 1st object tracking

Proprietary ISP-embedded algorithms use GPU for acceleration →~10x speedup and cost power

SOURCE: NVIDIA whitepaper, MWC '13

What are their limitations?

1. **Current solutions:** Modest speedups. Not generally applicable.

2. **Current algorithmic solutions:** slow on CPUs

IMG1		Image Registration	→	Multiframe Compositing	→ Enhanced Image
IMG1					
IMG1					

Image Alignment   **Image Fusing**

> 1.8 sec./frame   ~1 sec. total

Our target: ~100x speedup compared to software

**Image registration is a computational bottleneck
→ needs acceleration**

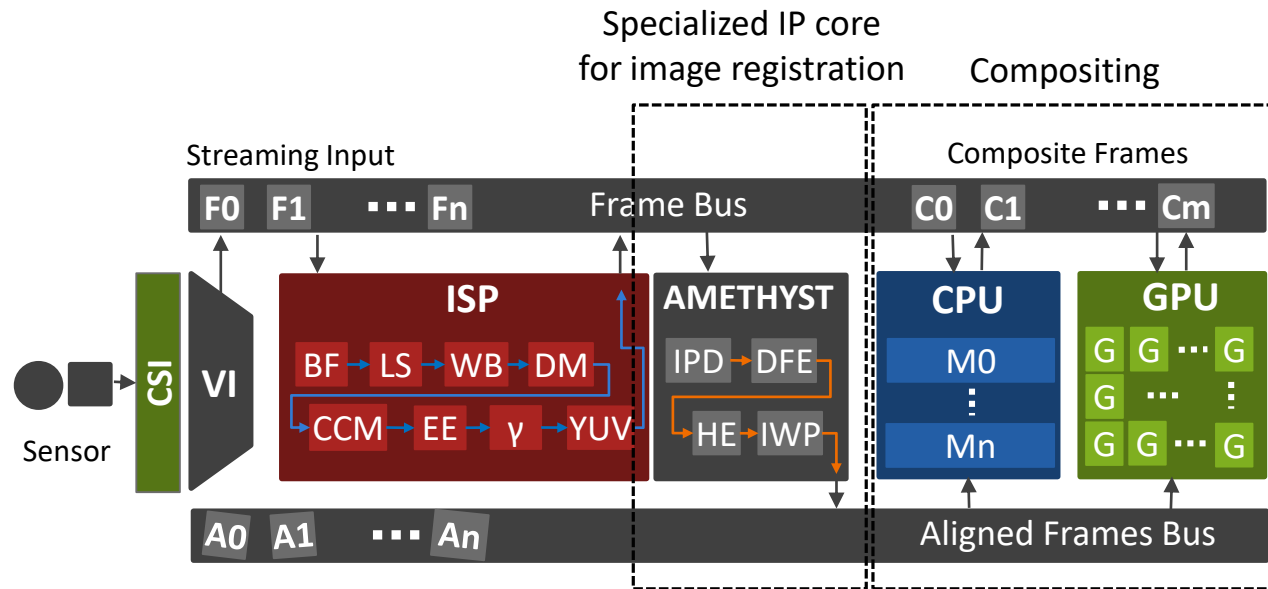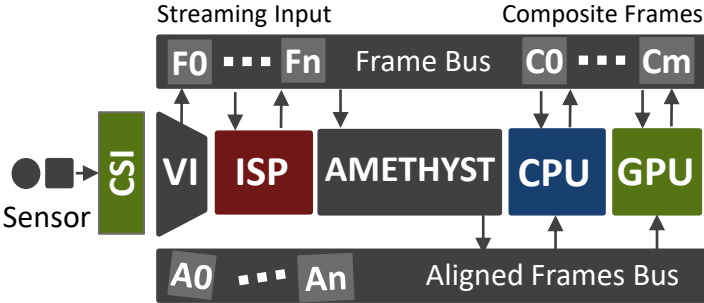# What have we done about it?

Fig: Proposed architecture for multi-frame image processing (MFP)

We propose an architecture for MFP that has a dedicated accelerator for image registration

What are our findings?

AMETHYST[+] Performance Summary

Technology	45 nm SOI
Area	0.15 mm^2 (30k gates)
Memory	< 2 MB
Frequency	1.1 GHz
Power	62.7 mW
Exec. Time	30ms/frame
Speed-up[$]	37x over CPU

Legend:
- A4, PowerVR SGX535*
- Tegra3, 12C GeForce*
- Snapdragon, Adreno200*
- Xilinx V6, 240k LC
- Artix 7, 85k LC
- Smartfusion2, 56k LUT
- AMETHYST, 0.3 GHz
- AMETHYST, 0.8 GHz
- AMETHSYT, 1.1 GHz
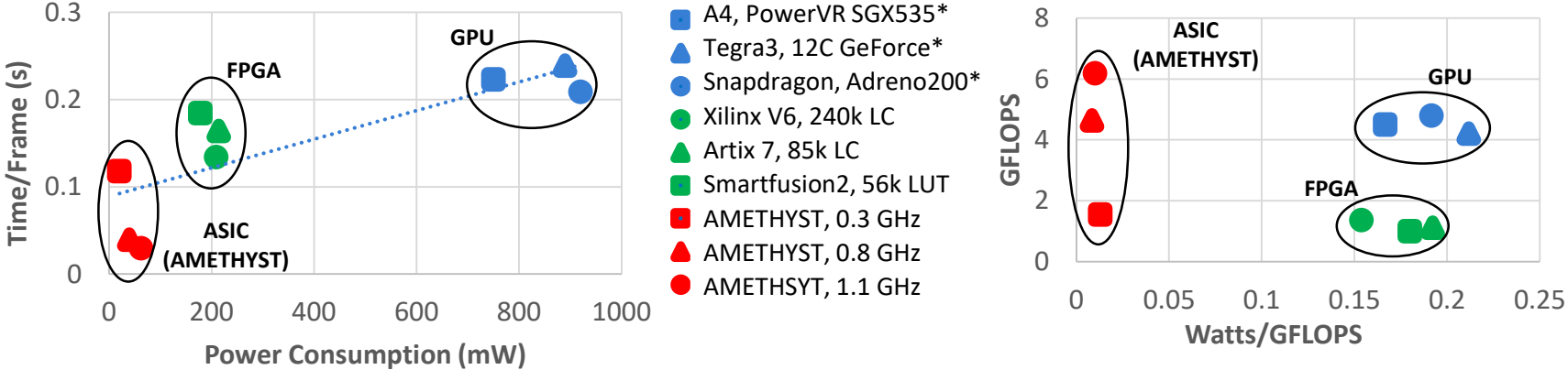
AMETHYST shows speed-up of : **8x over GPU and 5x over FPGA**
at a power lower by : **14x than GPU and 3x than FPGA**

What are our findings? Contd...

F0 **F1** **···** **Fn**   Frame Bus

Daisy (Sunflower) Feature Extraction   **Spatial Pooling**   **Homography Estimation** | **Image Warping**

Harris IPD   **T-block Processing**   RANSAC | Re-sample

| I_x | I_y |

Patch Buffer

G Block Smoothing $G(x,\sigma)$

gradient histogram (*e.g.*, SIFT, GLOH)
sine-weighted gradient quantization
quadrature steerable filters
Isotropic diff. of Gaussians (DoGs)

Covariance

NMS

Normalization

iterative

non-iterative

Feature Buffer

9x9 Jacobi SVD | Interpolate

P	P	···	P		Kernel 0
P	P	···	P		Kernel 1
⋮		⋱	⋮		⋮
P	P	···	P		Kernel n

Aligned Frames Bus   **A0** **A1** **···** **An**
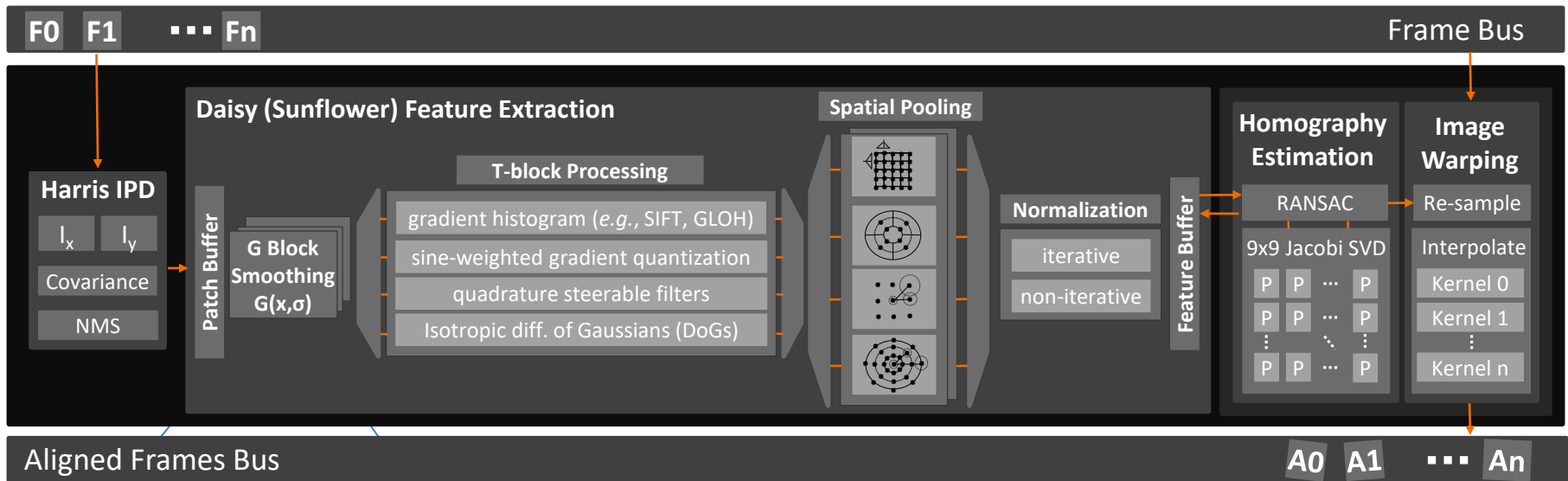
AMETHYST: Architecture Block Diagram

Kernel Matrix

ctrl regs
Idle

Out

Input Matrix

FIFO | FIFO | ·· | FIFO

FIFO
FIFO
⋮
FIFO

2d-PE 2d-PE ... 2d-PE   1d-PE
2d-PE   2d-PE
2d-PE 2d-PE ... 2d-PE   1d-PE

Local FIFO
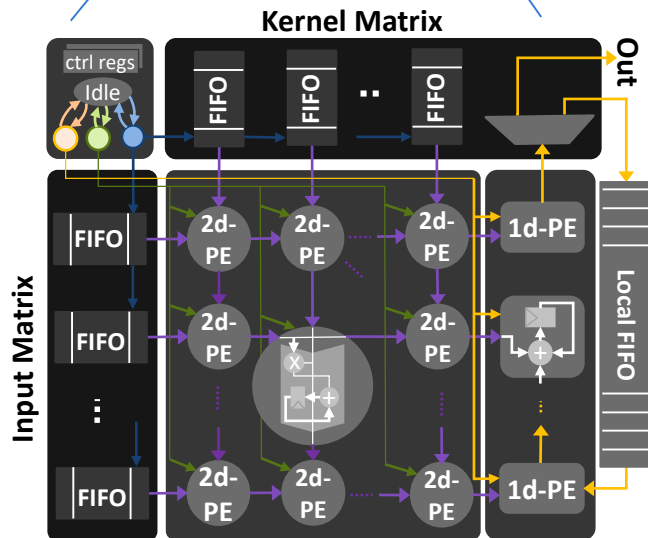
Highlights

- State-of-the-art algorithm[$] (from Photosynth)
- 1st MFP engine for re-targetable applications
- Extensively configurable parallelism
- Multi-level data pipelining and interleaving
- Systolic ops w/ 2-level vector reduction

[$]**M. Brown** *et al.*, **PAMI '10**
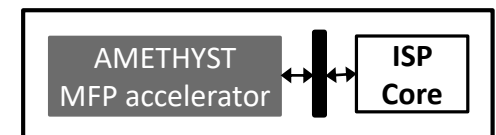
8

Next

Technical steps:

- Finish implementing RTL for HE and IWP modules
- Verify full-design on FPGA-based programmable SoCs (*e.g.*, Zynq)
- Develop HW-SW co-design with ARM core towards custom SoC
- Perform physical design and post-layout validation of SoC
- Integrate silicon-proven design IP with ISP core

Stage 1: FPGA validation
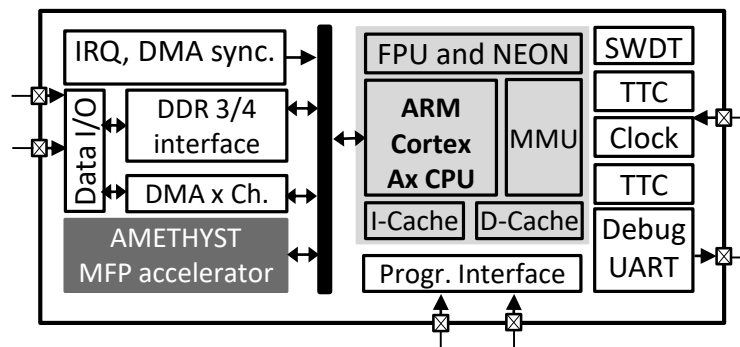
Stage 2: Silicon validation

Stage 3: ISP integration