Energy-Efficient Circuits and Systems for Computational Imaging and Vision on Mobile Devices

Priyanka Raina
Stanford University
praina@stanford.edu

SSCS Seminar
Oct 18, 2018

Domain Specific Architectures

Priyanka Raina
Stanford University
praina@stanford.edu

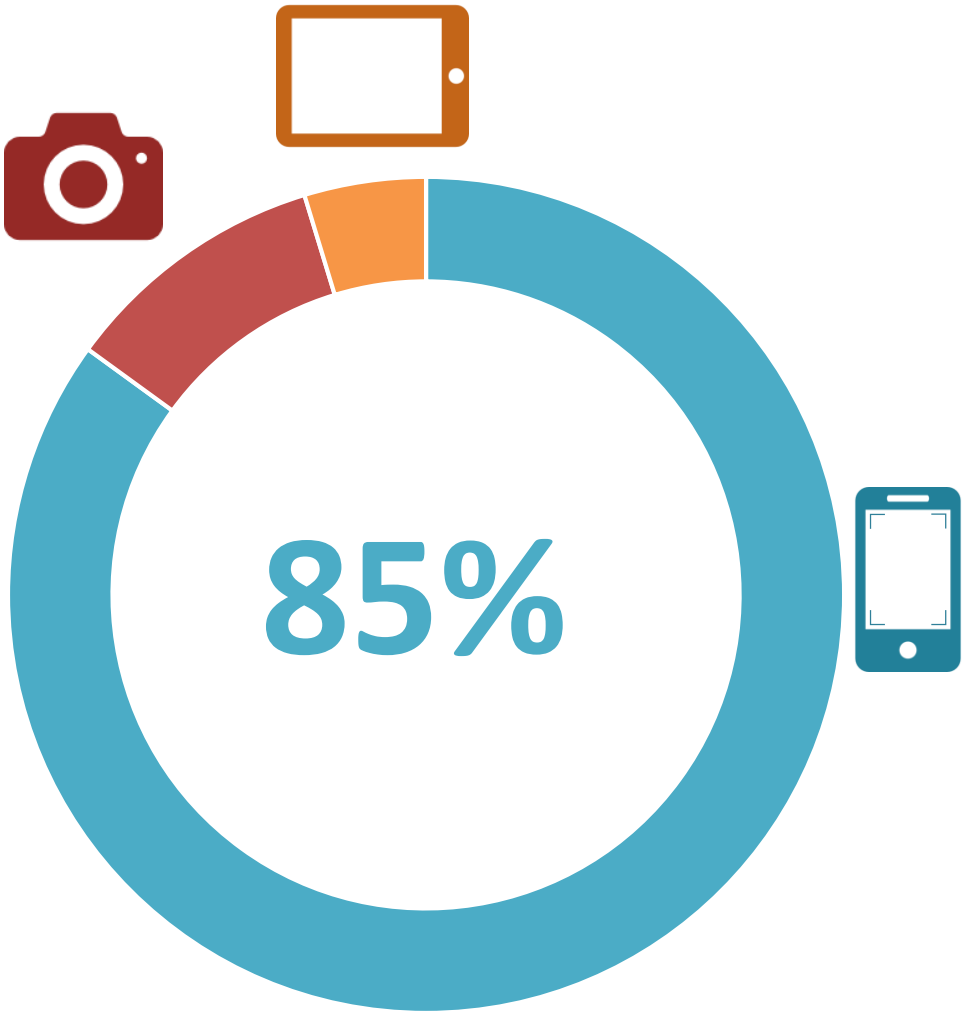
SSCS Seminar
Oct 18, 2018

whoami

- Past
  - Completed my PhD from MIT in January
    - With Prof. Anantha Chandrakasan
    - Thesis: Energy-efficient circuits and systems for computational imaging and vision on mobile devices
  - Was a visiting research scientist at Nvidia Research from Jan – Aug 2018
- Now
  - **Assistant Professor at Stanford since Sep 2018**

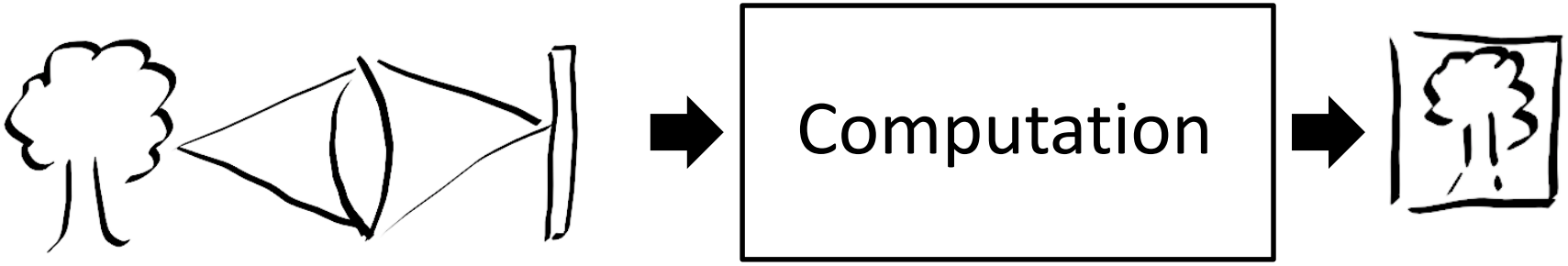
More than 1.2 trillion photos in 2018

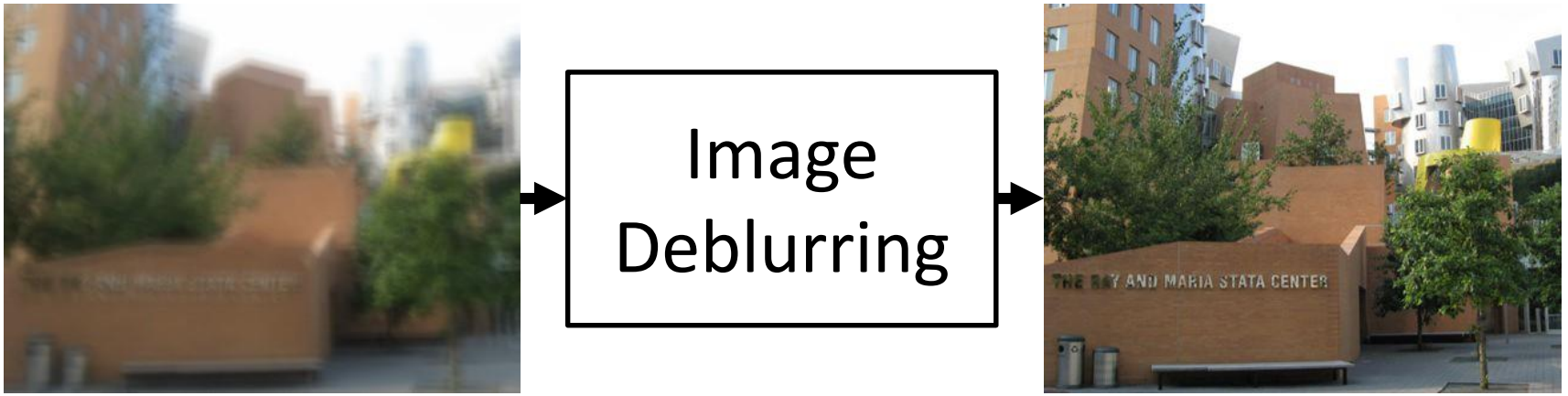
Cellphones are the main cameras



All imaging involves heavy computation

Image Capture





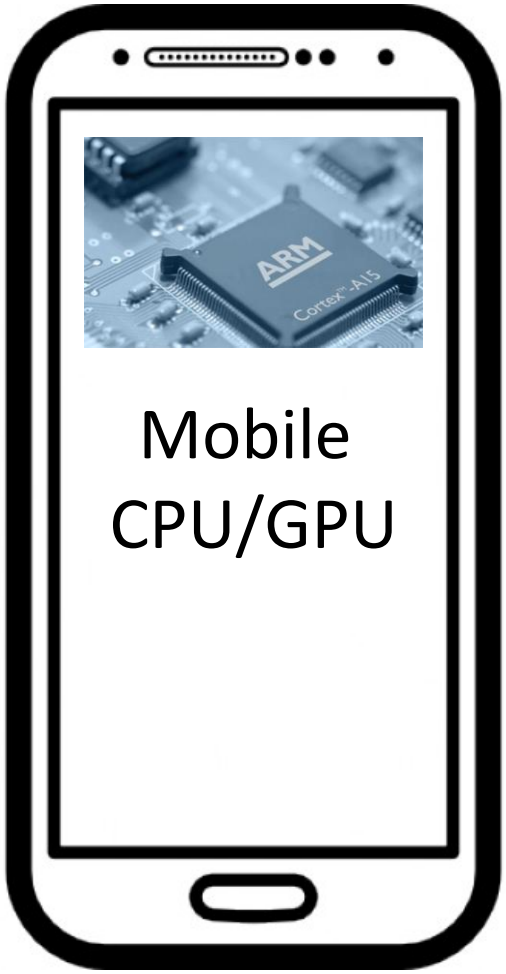
Mobile image processing is expensive

Image Capture

1810 mAH battery

Phone dead after deblurring only 10 photos!

Computation



13.6 minutes

2284 J per frame

Energy-Efficient Hardware Accelerators

Image Capture

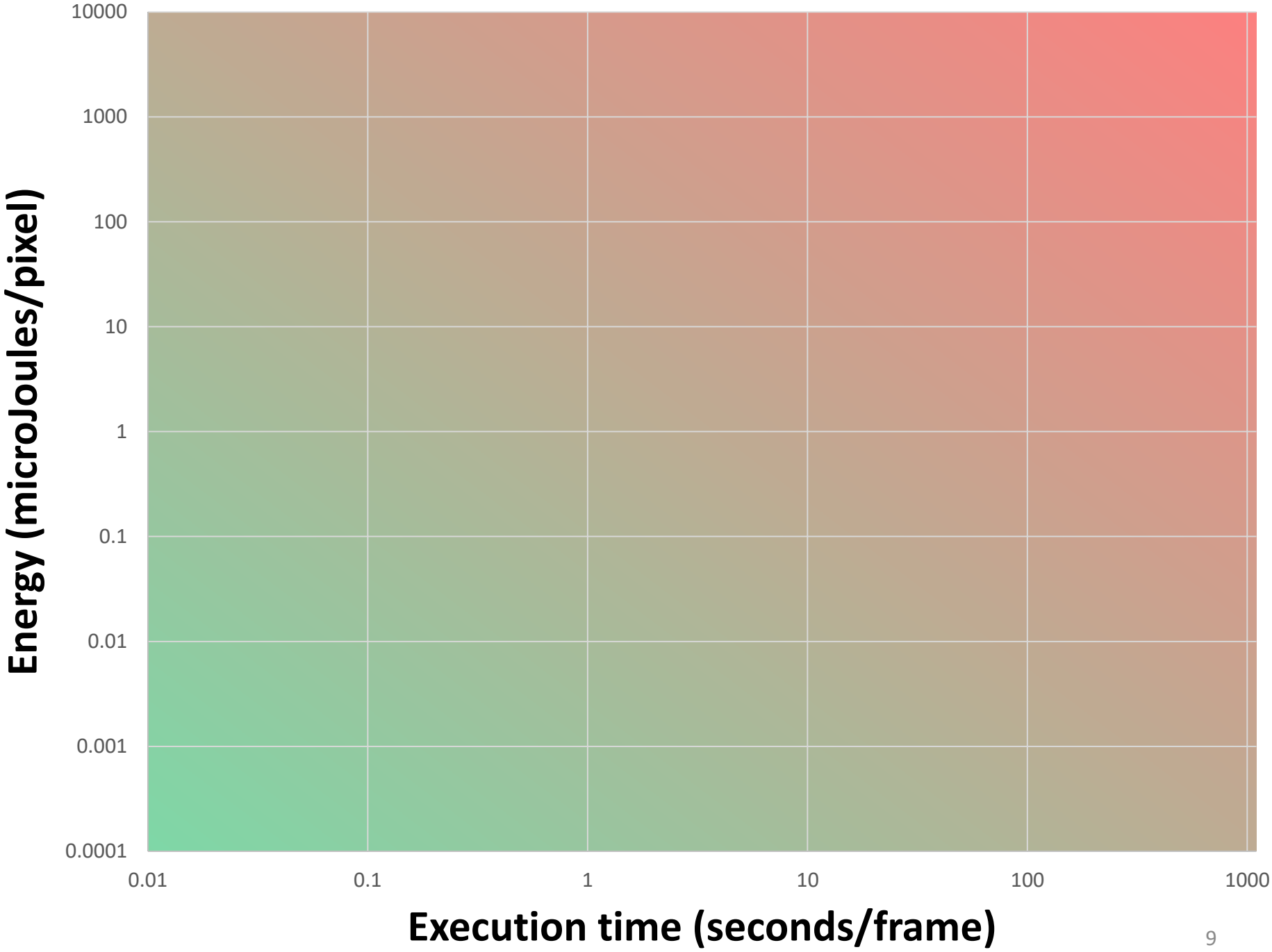
Computation

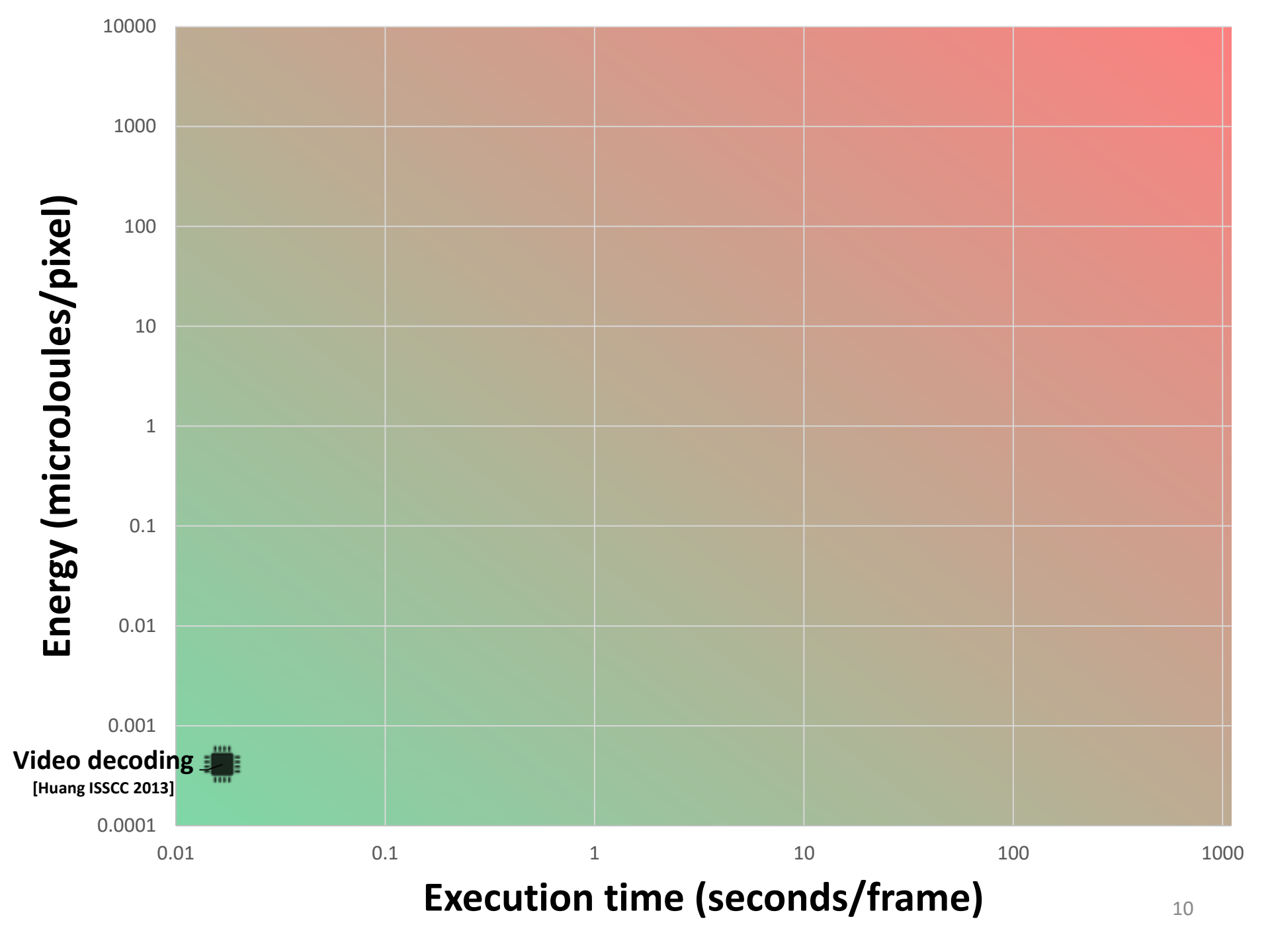
~~13.6 minutes~~

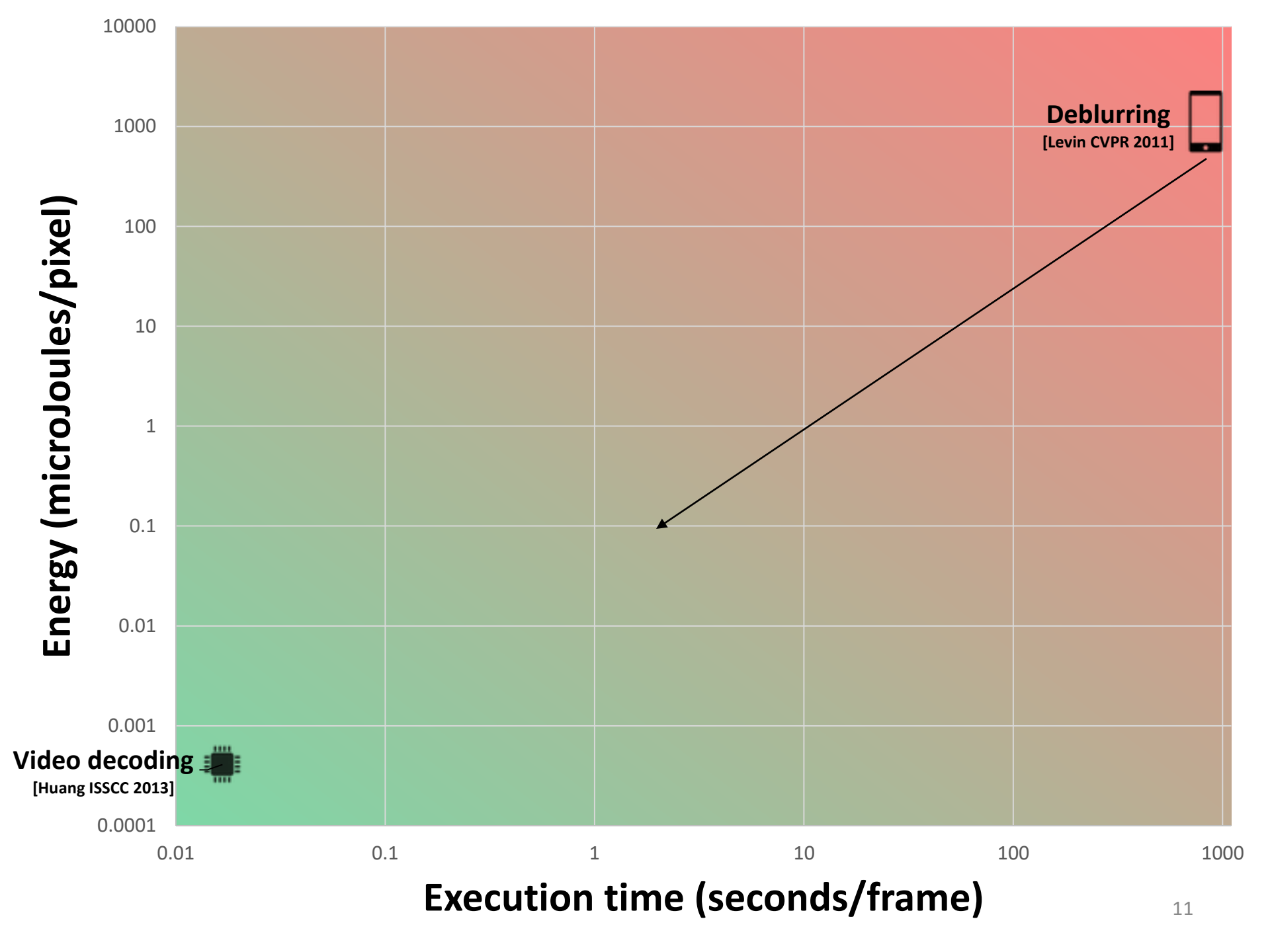
~~2284 J per frame~~

1.7 seconds

0.1 J per frame


10000
1000
100
10
1
0.1
0.01
0.001
0.0001
Energy (microJoules/pixel)
0.01
0.1
1
10
100
1000
Execution time (seconds/frame)


10000
1000
100
10
1
0.1
0.01
0.001
0.0001
Energy (microJoules/pixel)
Video decoding
[Huang ISSCC 2013]
0.01
0.1
1
10
100
1000
Execution time (seconds/frame)


10000
1000
100
10
1
0.1
0.01
0.001
0.0001
Energy (microJoules/pixel)
Deblurring
[Levin CVPR 2011]
Video decoding
[Huang ISSCC 2013]
0.01
0.1
1
10
100
1000
Execution time (seconds/frame)

Energy (microJoules)

10

1

0.1

0.01

0.001

0.0001

Video decoding

[Huang ISSCC 2013]

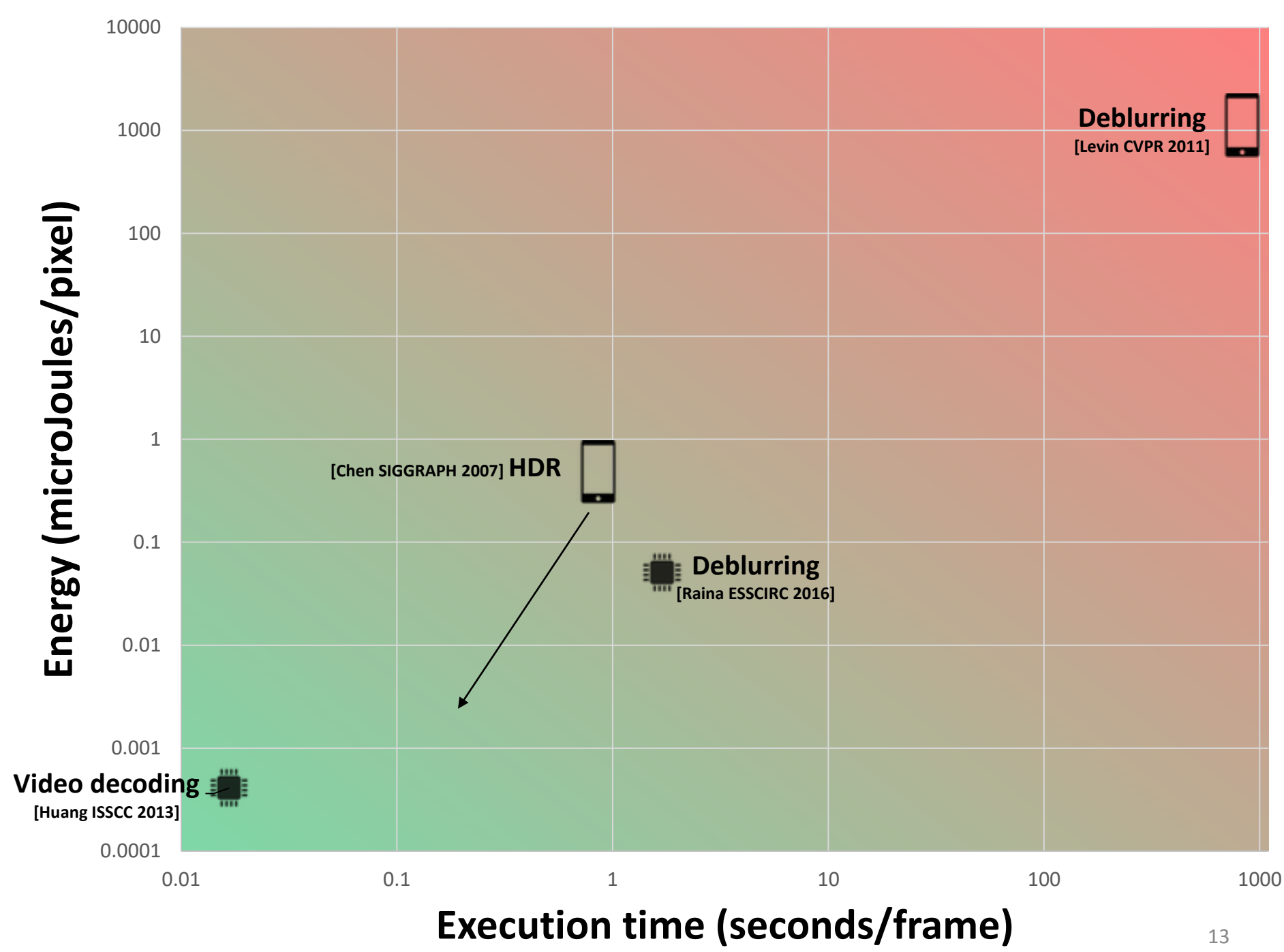
0.01

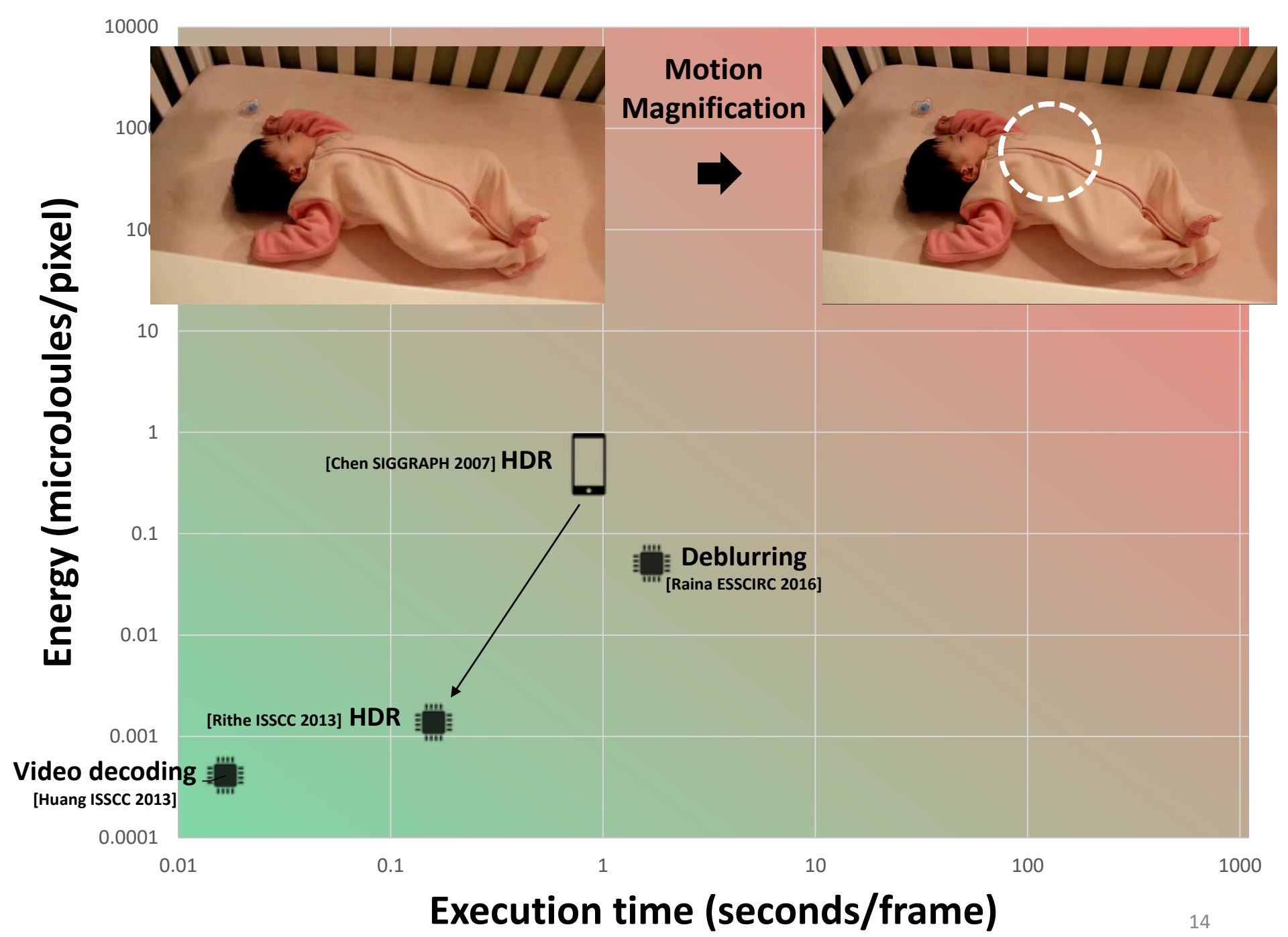
100

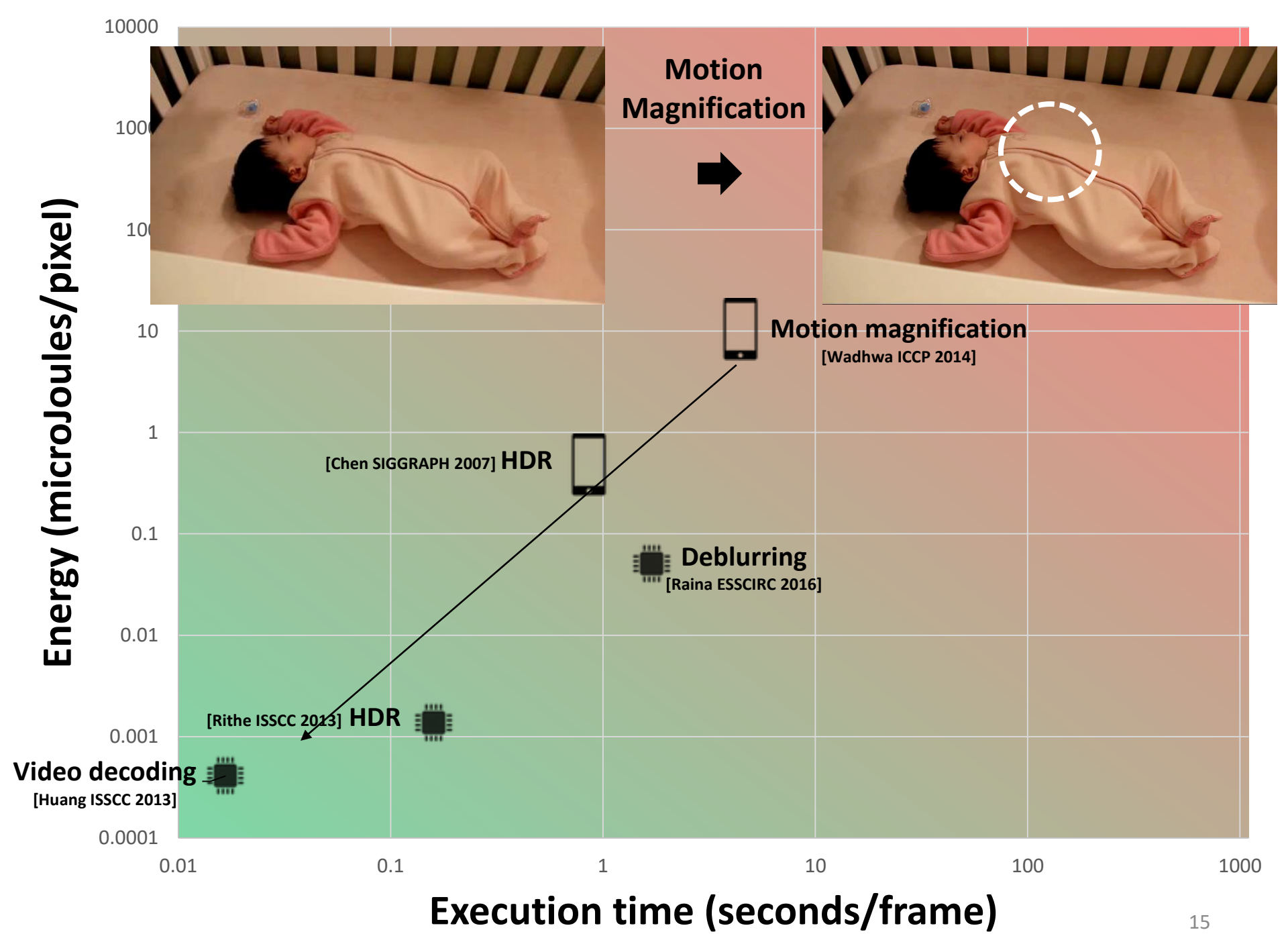
1000

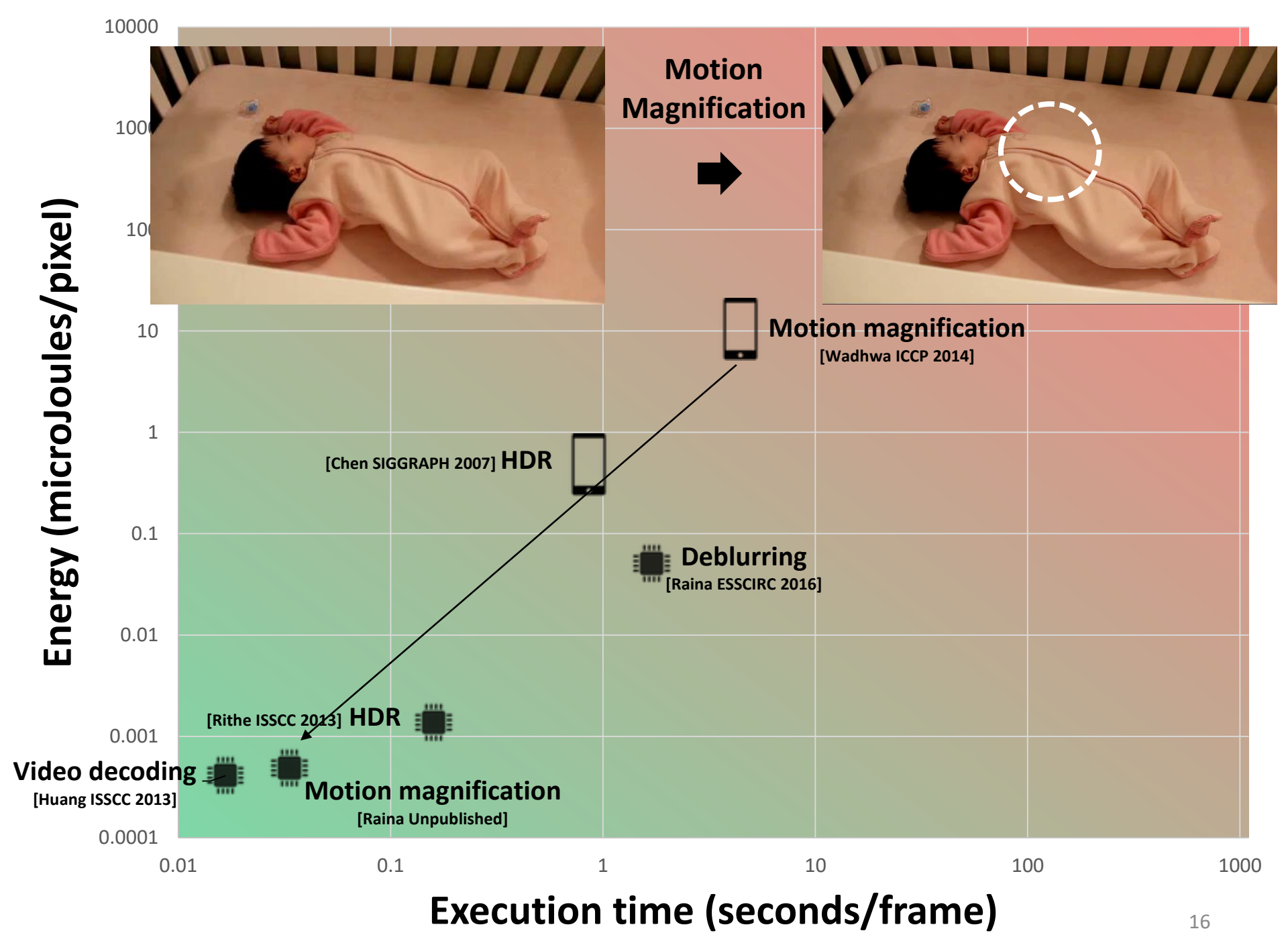
High Dynamic Range (HDR) Imaging

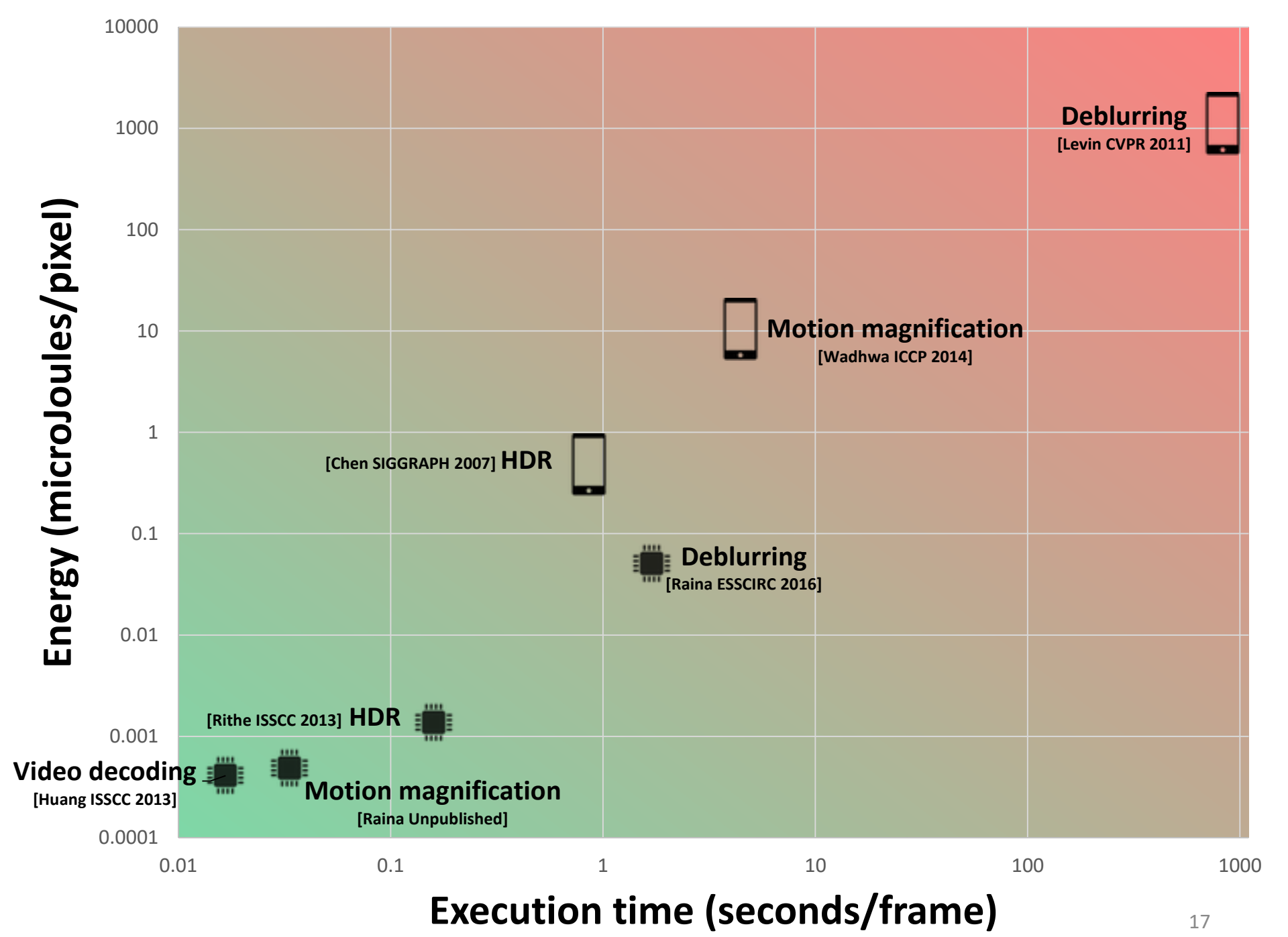
[Wikipedia]


10000
1000
100
10
1
0.1
0.01
0.001
0.0001
Energy (microJoules/pixel)
0.01
0.1
1
10
100
1000
Execution time (seconds/frame)
Deblurring
[Levin CVPR 2011]
[Chen SIGGRAPH 2007] HDR
Deblurring
[Raina ESSCIRC 2016]
Video decoding
[Huang ISSCC 2013]


Motion Magnification
[Chen SIGGRAPH 2007] HDR
Deblurring
[Raina ESSCIRC 2016]
[Rithe ISSCC 2013] HDR
Video decoding
[Huang ISSCC 2013]
Energy (microJoules/pixel)
Execution time (seconds/frame)
10000
100
10
1
0.1
0.01
0.001
0.0001
0.01
0.1
1
10
100
1000


Motion Magnification
Motion magnification
[Wadhwa ICCP 2014]
[Chen SIGGRAPH 2007] HDR
Deblurring
[Raina ESSCIRC 2016]
[Rithe ISSCC 2013] HDR
Video decoding
[Huang ISSCC 2013]
Energy (microJoules/pixel)
Execution time (seconds/frame)
10000
100
10
10
1
0.1
0.01
0.001
0.0001
0.01
0.1
1
10
100
1000


Motion Magnification
Motion magnification
[Wadhwa ICCP 2014]
[Chen SIGGRAPH 2007] HDR
Deblurring
[Raina ESSCIRC 2016]
[Rithe ISSCC 2013] HDR
Video decoding
[Huang ISSCC 2013]
Motion magnification
[Raina Unpublished]
Energy (microJoules/pixel)
Execution time (seconds/frame)
10000
100
10
10
1
0.1
0.01
0.001
0.0001
0.01
0.1
1
10
100
1000


10000
1000
100
10
1
0.1
0.01
0.001
0.0001
Energy (microJoules/pixel)
0.01
0.1
1
10
100
1000
Execution time (seconds/frame)
Deblurring
[Levin CVPR 2011]
Motion magnification
[Wadhwa ICCP 2014]
[Chen SIGGRAPH 2007] HDR
Deblurring
[Raina ESSCIRC 2016]
[Rithe ISSCC 2013] HDR
Video decoding
[Huang ISSCC 2013]
Motion magnification
[Raina Unpublished]

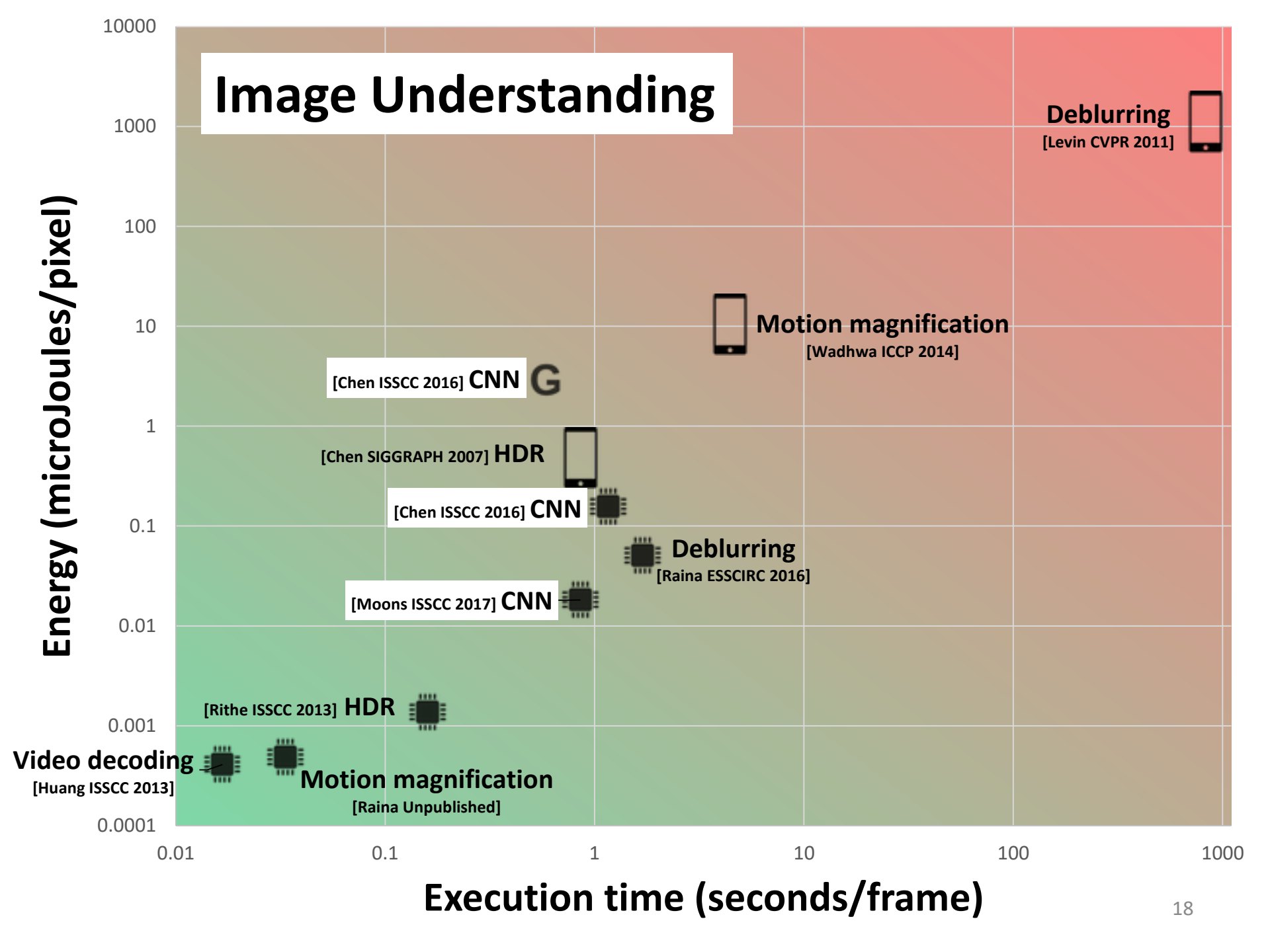
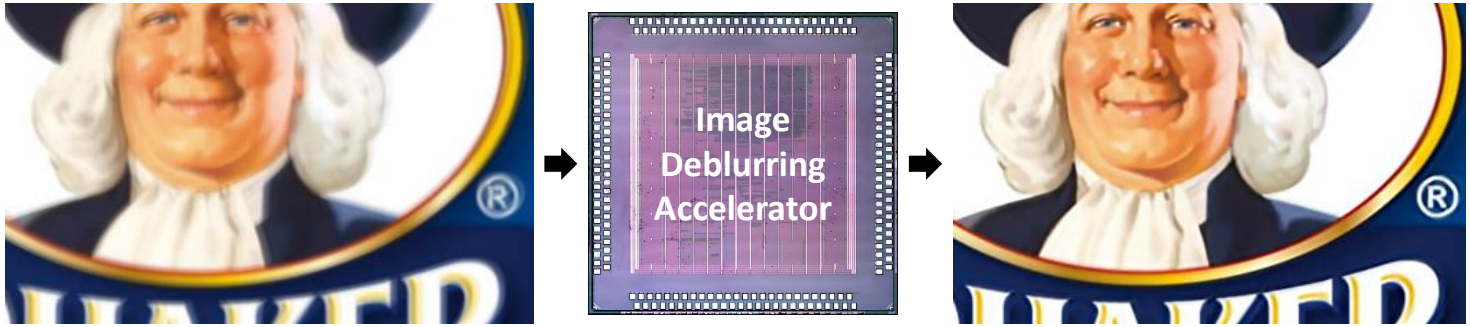


Image Understanding
Deblurring
[Levin CVPR 2011]
Motion magnification
[Wadhwa ICCP 2014]
[Chen ISSCC 2016] CNN
[Chen SIGGRAPH 2007] HDR
[Chen ISSCC 2016] CNN
Deblurring
[Raina ESSCIRC 2016]
[Moons ISSCC 2017] CNN
[Rithe ISSCC 2013] HDR
Video decoding
[Huang ISSCC 2013]
Motion magnification
[Raina Unpublished]
Energy (microJoules/pixel)
10000
1000
100
10
1
0.1
0.01
0.001
0.0001
0.01
0.1
1
10
100
1000
Execution time (seconds/frame)

Energy-Efficient Imaging Accelerators

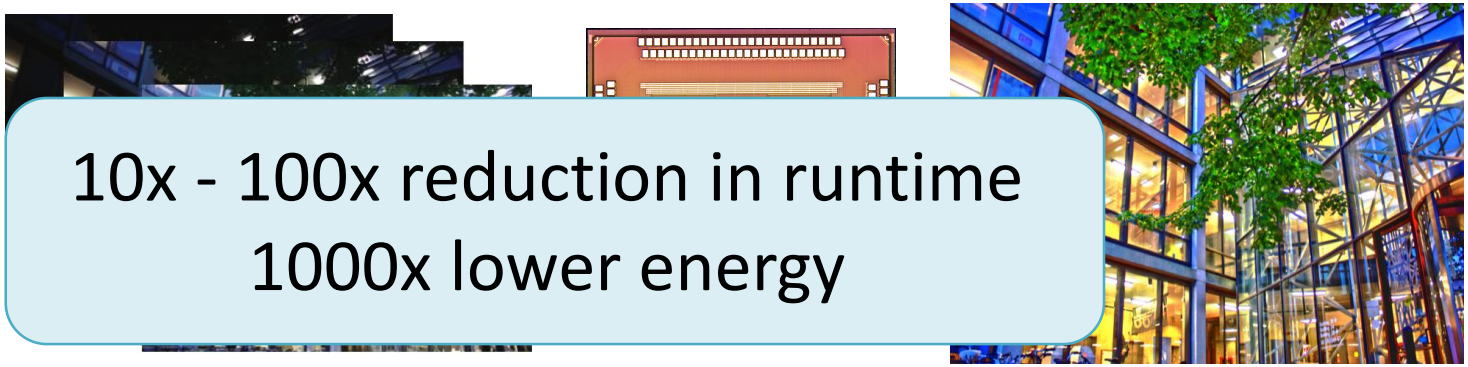
Image Deblurring

[**P. Raina**, M. Tikekar, A. P. Chandrakasan **ESSCIRC 2016, JSSC 2017]**



HDR & Low Light Imaging

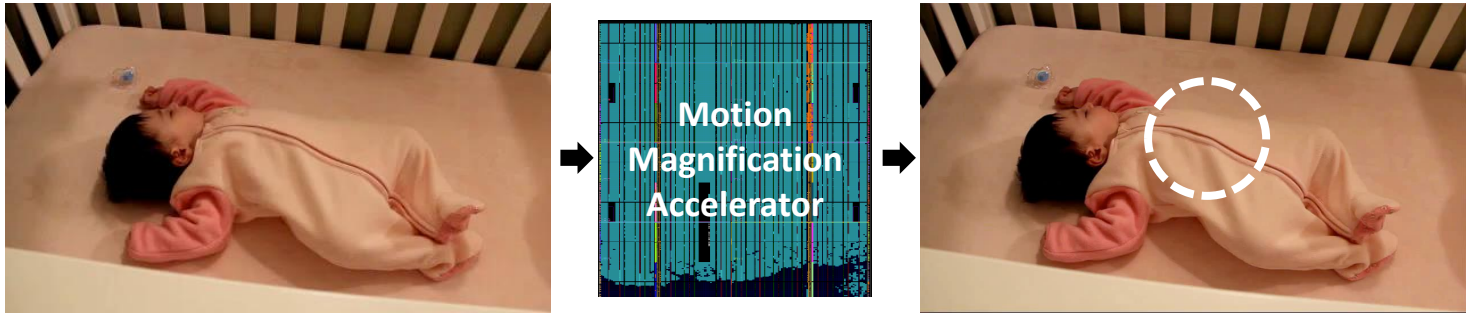
[R. Rithe, **P. Raina**, N. Ickes, S. Tenneti, A. P. Chandrakasan **ISSCC 2013, JSSC 2013]**



10x - 100x reduction in runtime
1000x lower energy

Motion Magnification

[**P. Raina**, D. Jeon, W. T. Freeman, F. Durand, A. P. Chandrakasan, In progress]



Camera Shake Blur

Image Deblurring

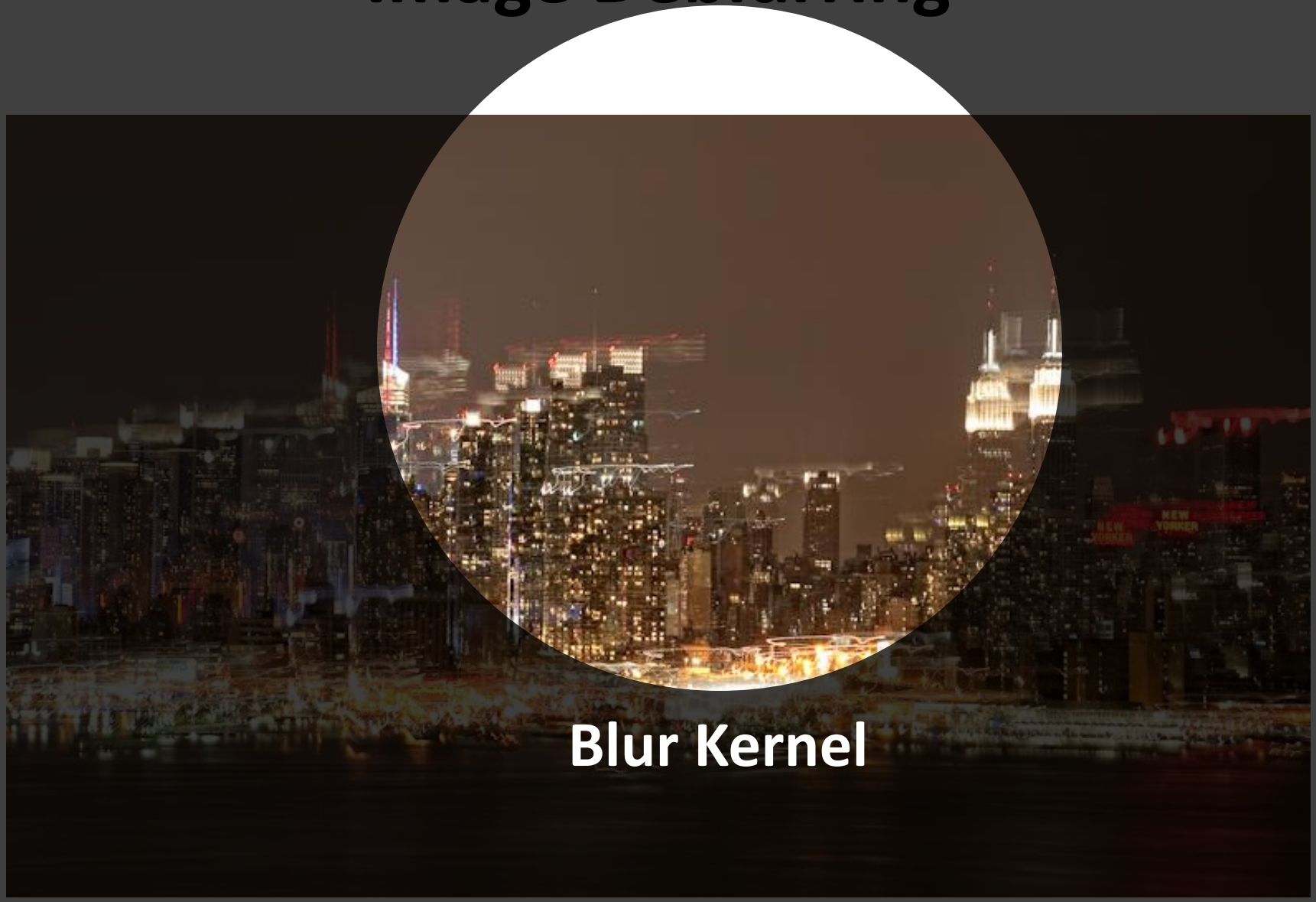
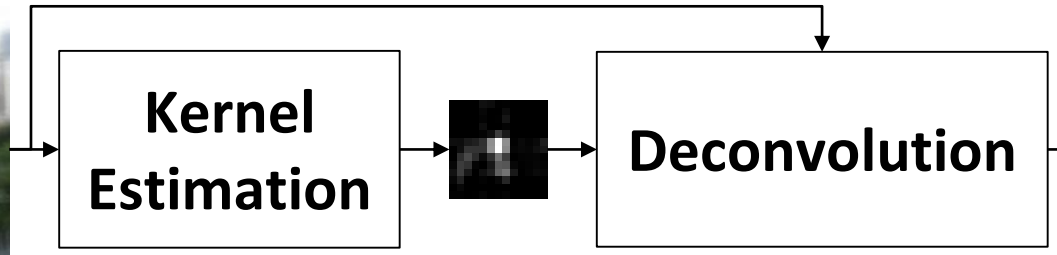


Image Deblurring

Input Blurred Image



Blur Kernel

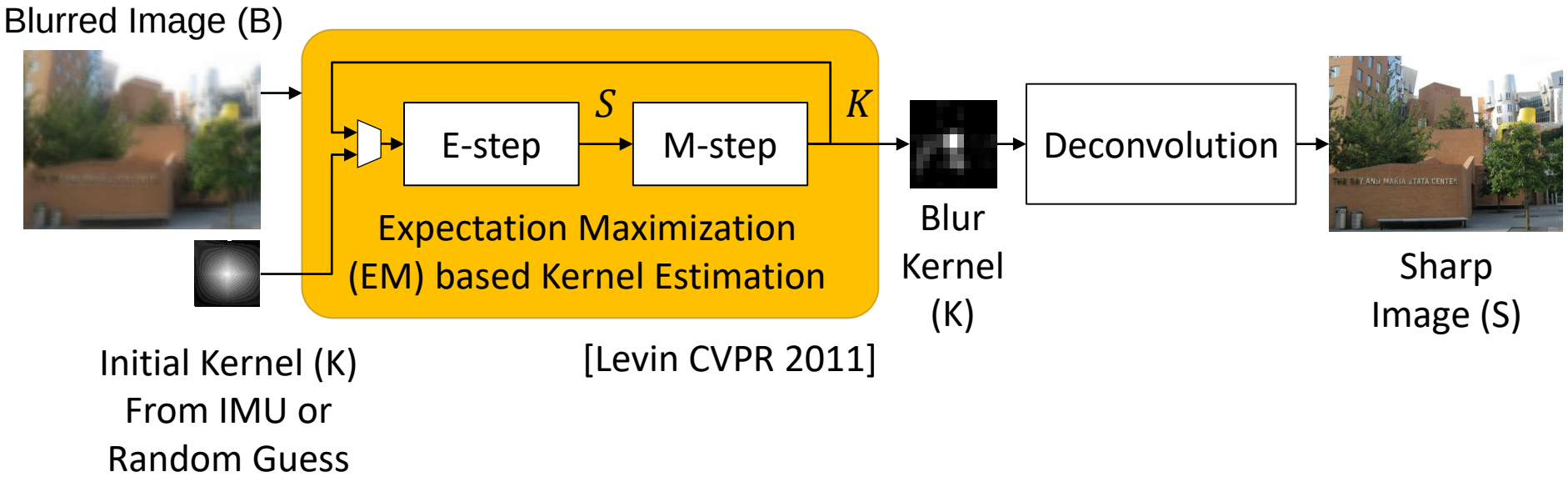
Output Sharp Image

Kernel Estimation	Deconvolution
134 s	.7 s

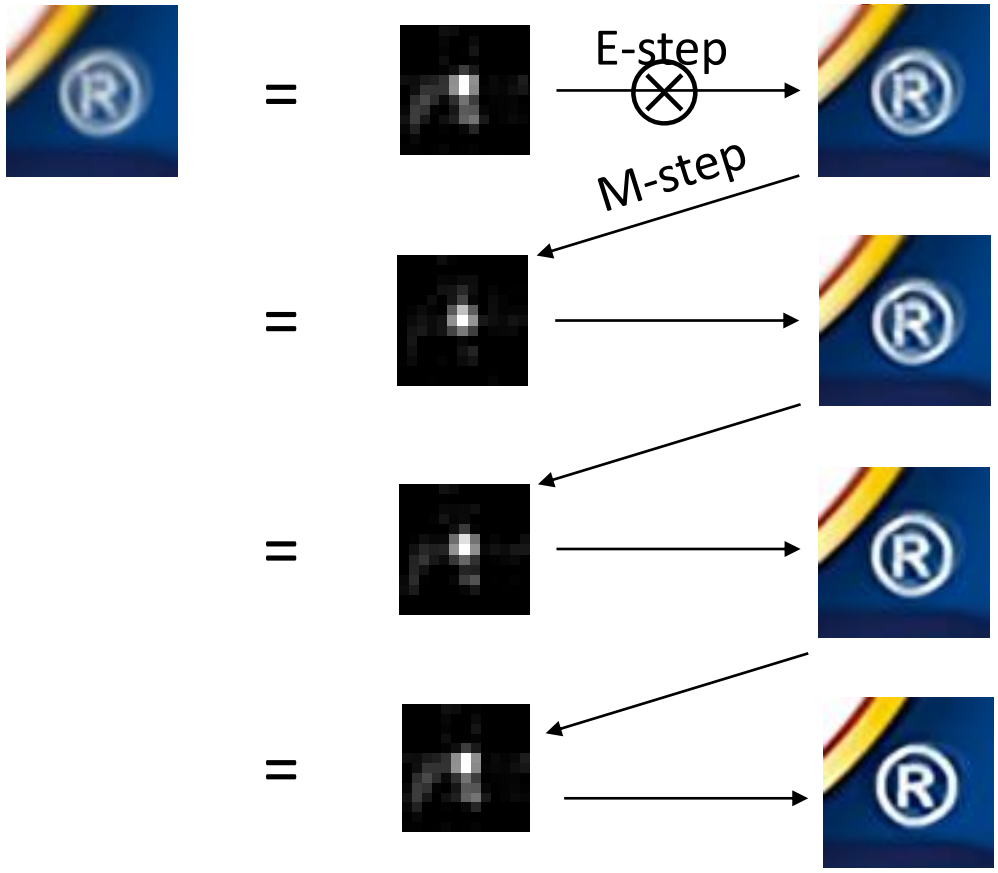
99% of deblurring time

Target for acceleration

EM-based Image Deblurring



EM-based Image Deblurring



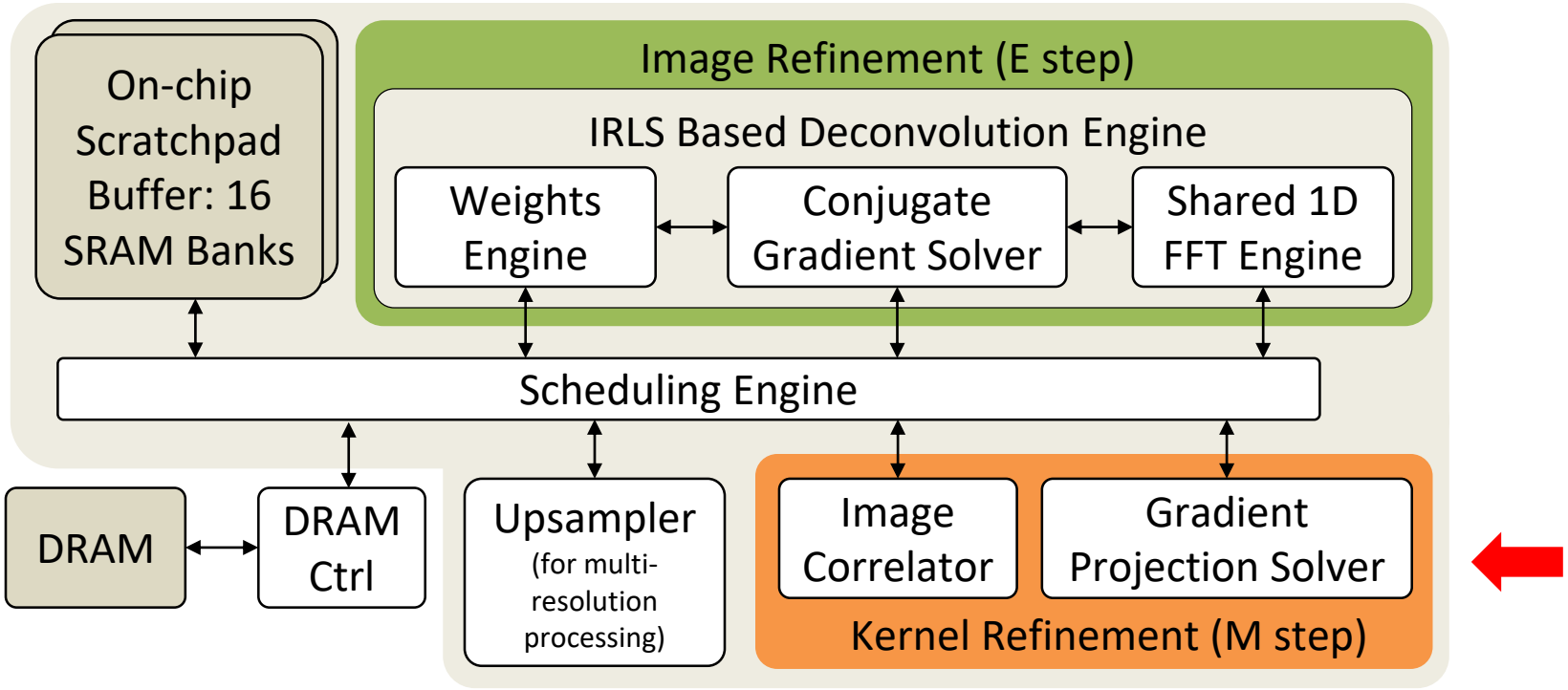
Regularization based on characteristics of naturally occurring sharp images and kernels is enforced

[Levin CVPR 2011]

Challenges and Techniques

- Highly sequential and iterative with nested loops
  - **Extracting parallelism from matrix operations for reducing execution time**
  - **Sharing arithmetic units and on-chip memory between non-concurrent stages to reduce area and leakage power**
- Very high memory bandwidth
  - **Exploiting spatial and temporal locality of memory accesses with on-chip scratchpad buffer to reduce memory bandwidth**

Deblurring Accelerator Architecture



[P. Raina, M.Tikekar, A. P. Chandrakasan, **ESSCIRC 2016, JSSC 2017, ISSCC SRP 2016]**

High-Throughput Image Correlator

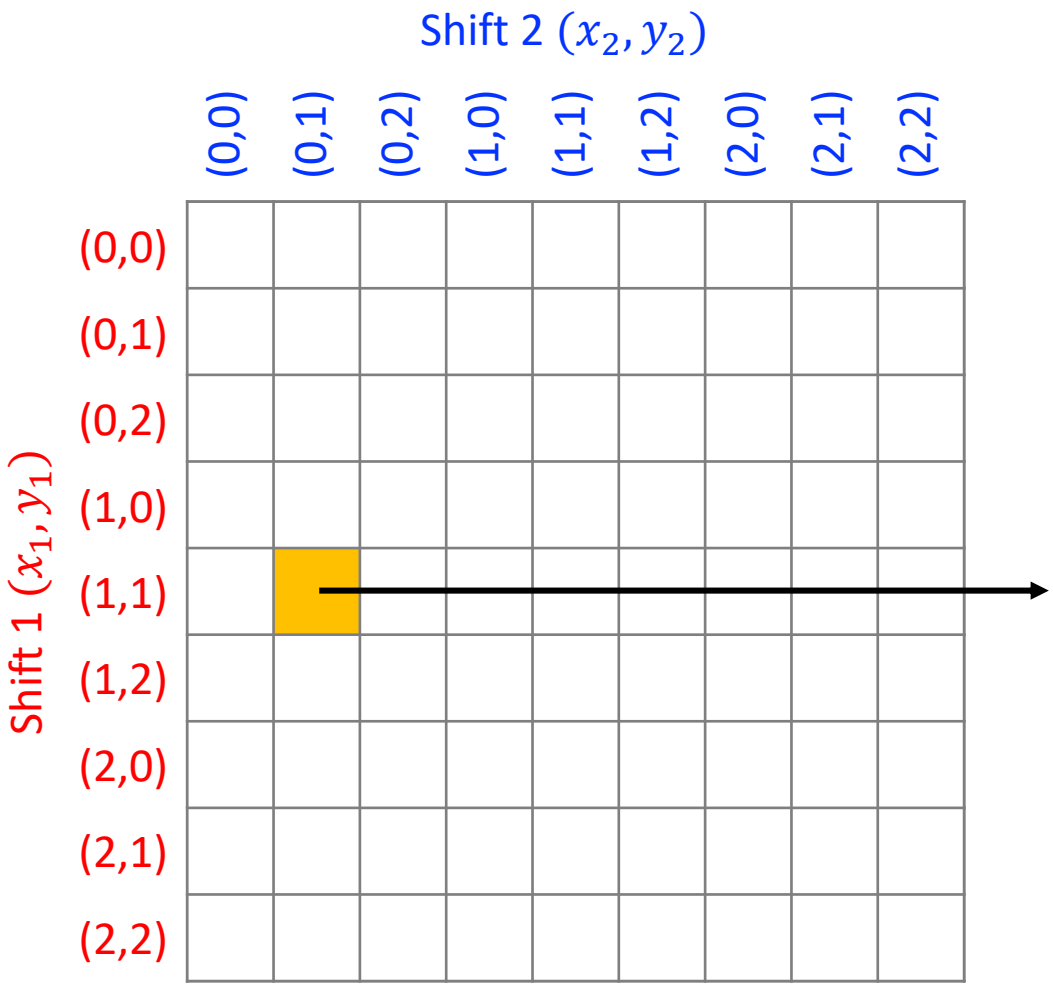
- **M-step:** $k = \arg\min \frac{1}{2} k^T \boldsymbol{A} k - b^T k \; ; k \geq 0$

- For an m × m kernel and n × n sharp image S, the $m^2 \times m^2$ correlation matrix is given by

$$\boldsymbol{A}(mx_1 + y_1, mx_2 + y_2) = \sum_{x=m-1}^{n-1} \sum_{y=m-1}^{n-1} S(x - x_1, y - y_1) * S(x - x_2, y - y_2)$$

- Shifts $(x_1, y_1), (x_2, y_2)$ vary from $(0,0)$ to $(m-1, m-1)$

Image Correlator

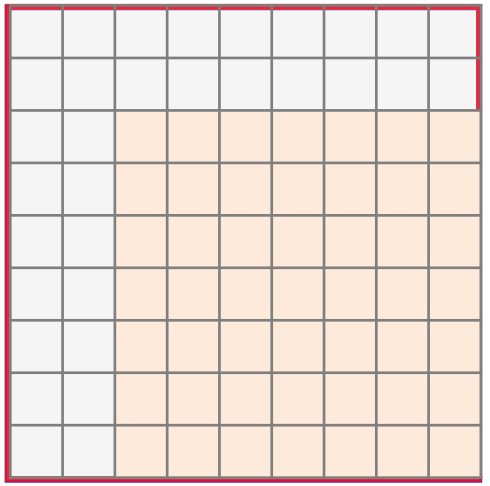


9 × 9 Correlation Matrix for a 3 × 3 kernel

$m^2 \times m^2$ $m \times m$

Sharp Image

n × n

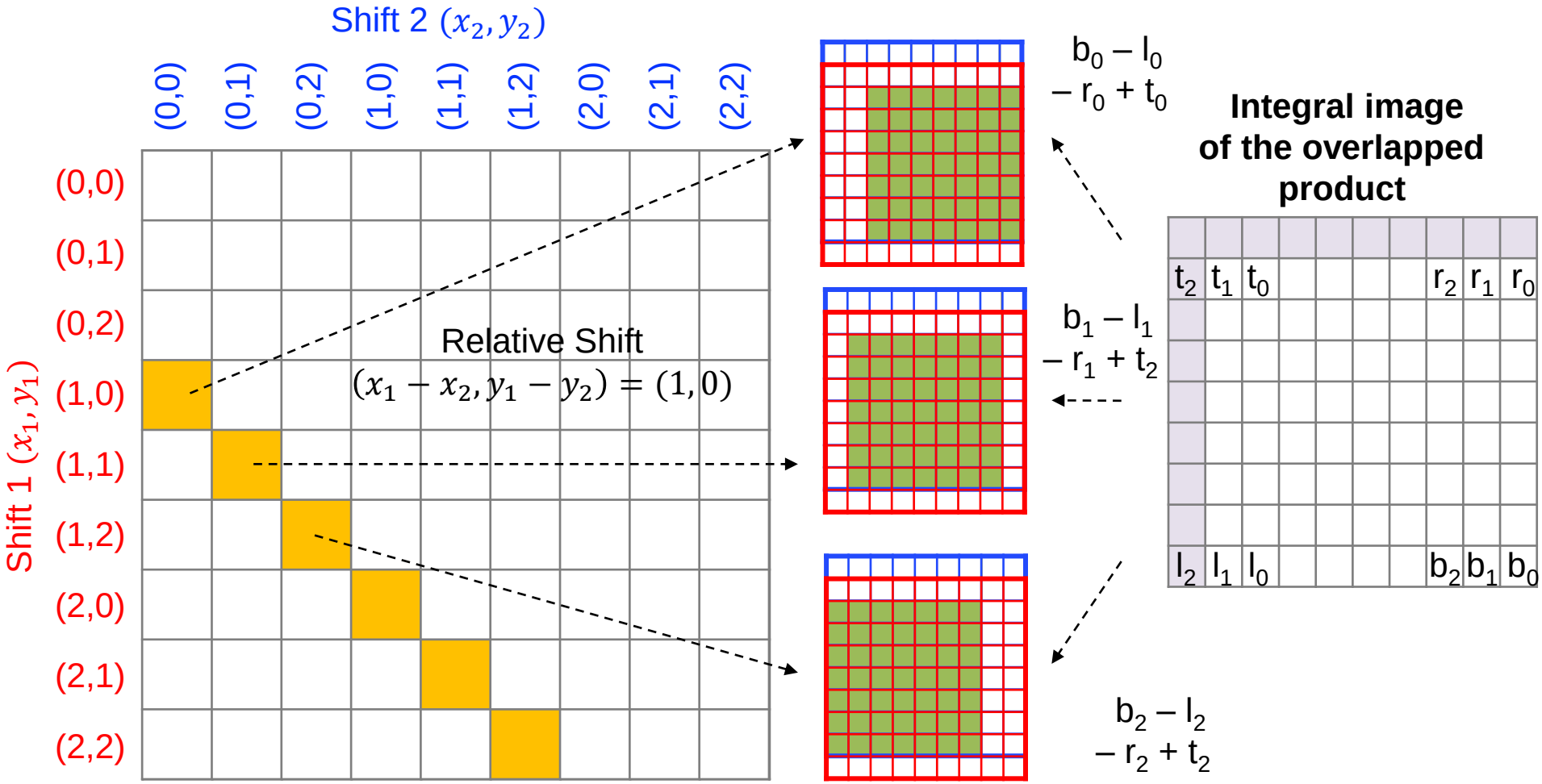


Multiply and accumulate in the orange area

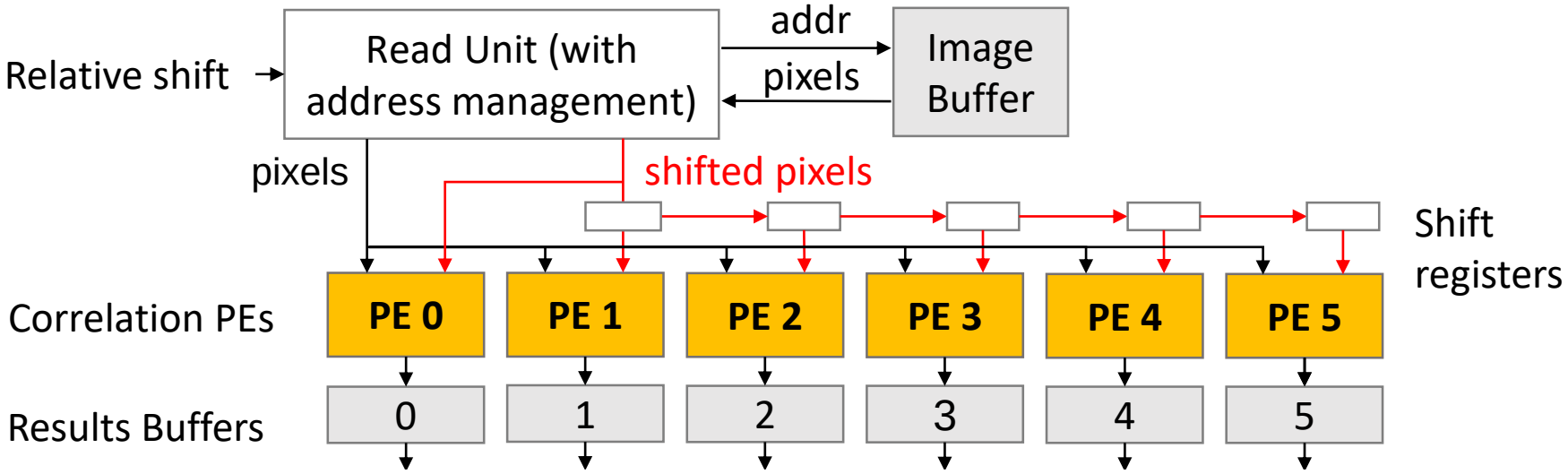
$O(m^4.n^2)$ Computation Time

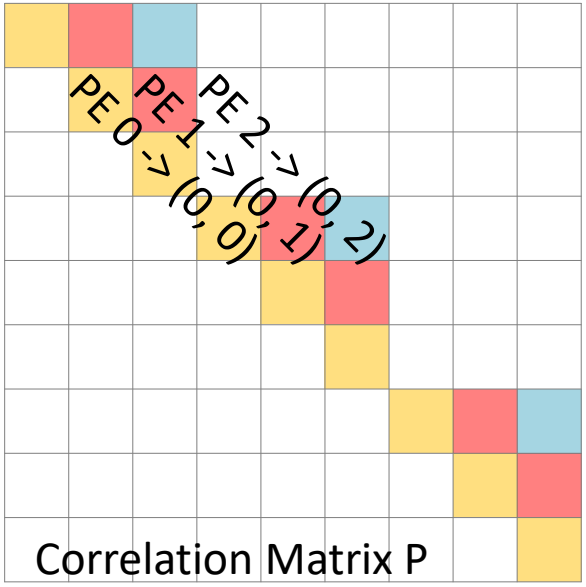
Diagonal Computation Reuse

Makes the computation of the correlation matrix $O(m^2n^2)$ rather than $O(m^4n^2)$, providing a speedup of 42× for a 13×13 kernel



6-Parallel Correlator with Image Tiling





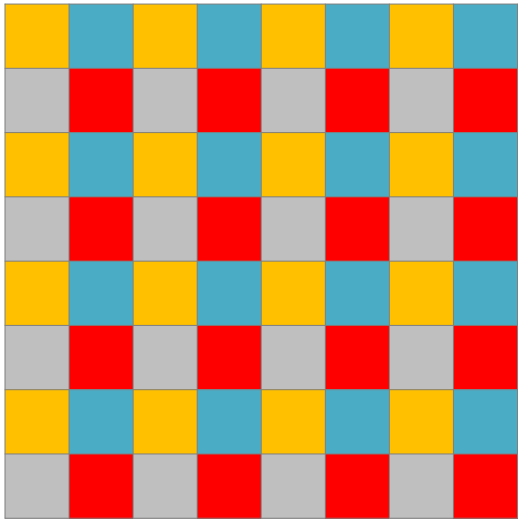
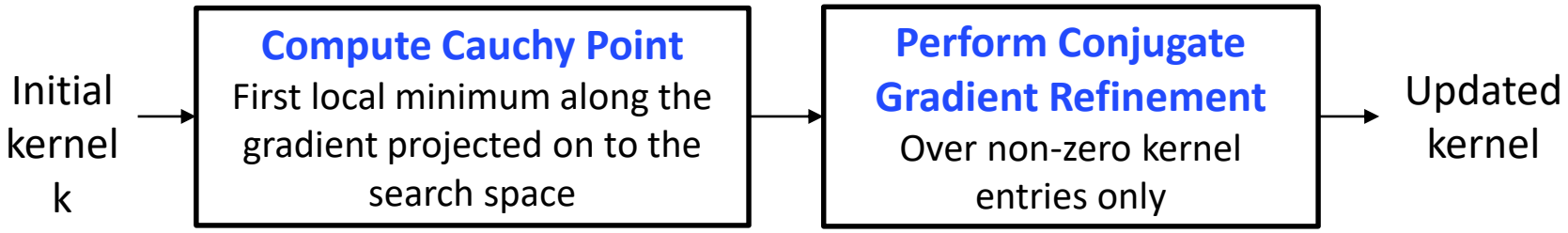


Image tiling for parallel access to any two shifted versions

Gradient Projection Solver

$$min_k \ \frac{1}{2} k^T A k - b^T k; k \geq 0$$

Image Correlator

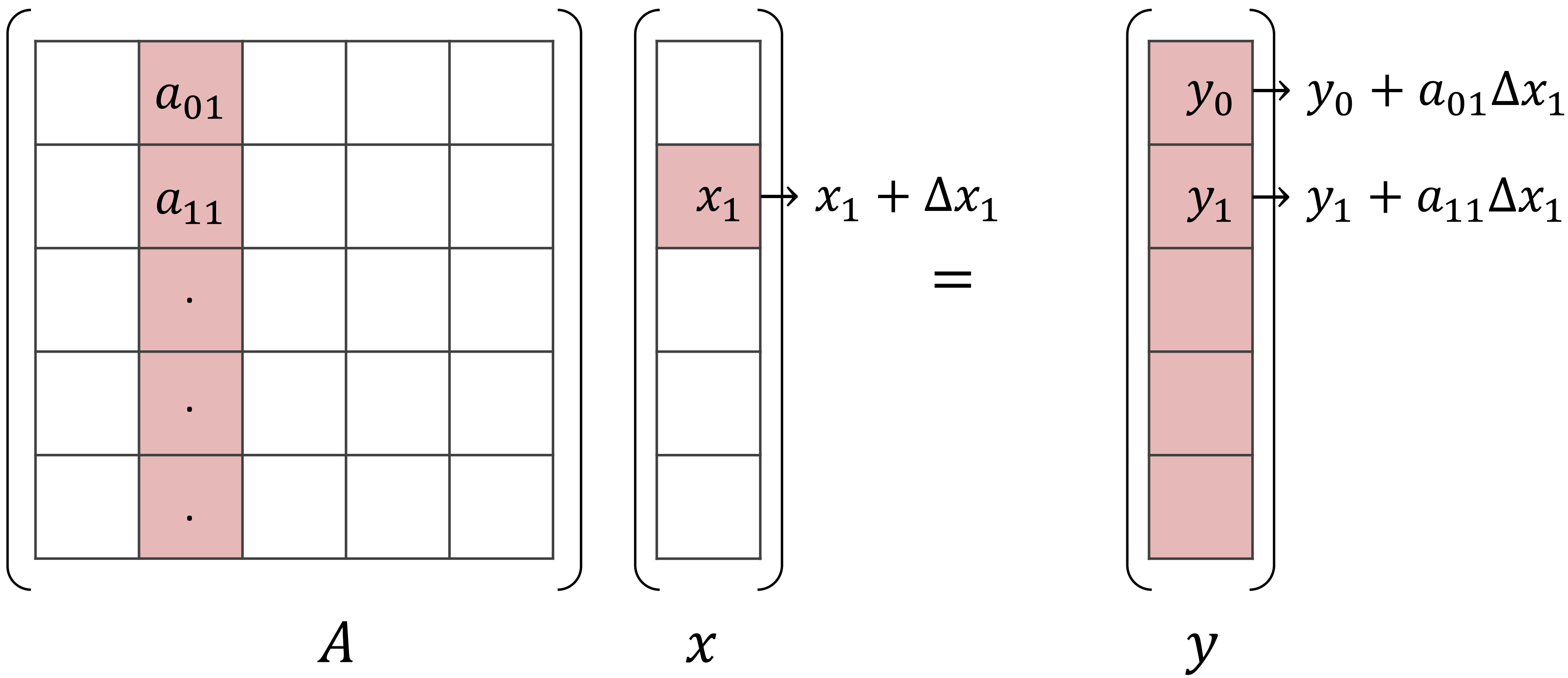


Matrix vector multiplications: expensive

Sparse updates save 11x compute and memory

Minimize $\frac{1}{2}k^T A k - b^T k$ to get the best k, where $k \geq 0$

Large (up to 841×841 pixels)



Hardware Sharing in Gradient Projection

Sharing of floating point units between the two steps results in 56% area savings in solver hardware

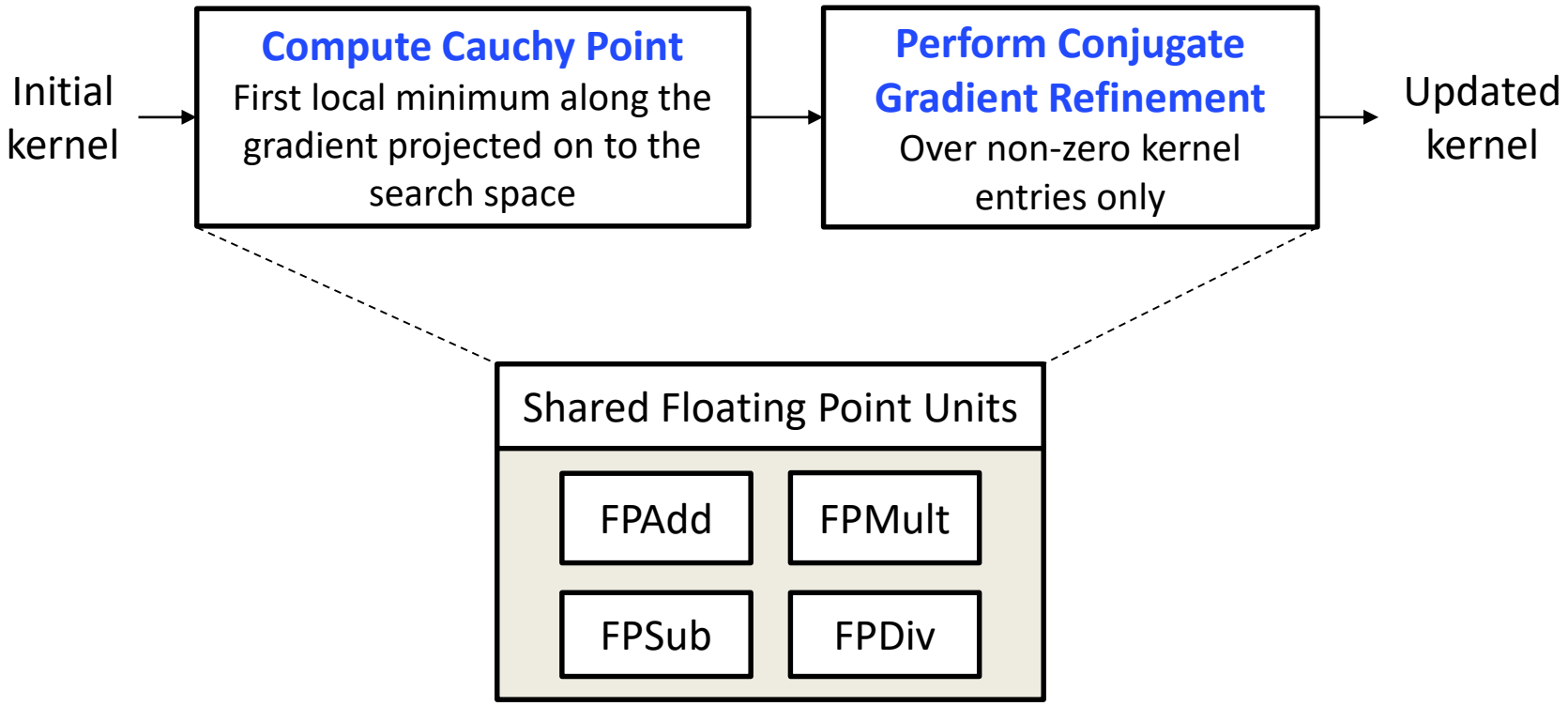
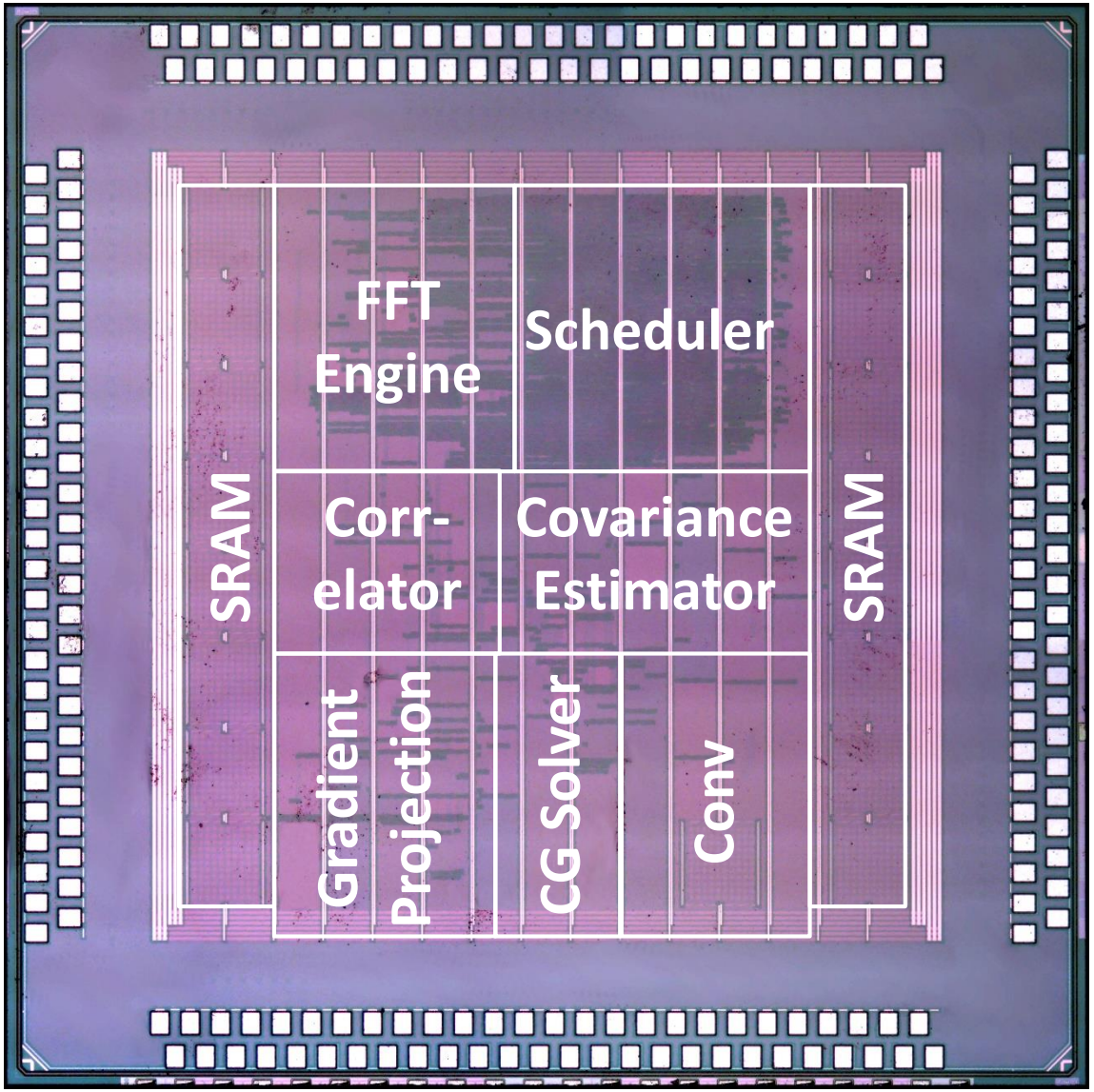
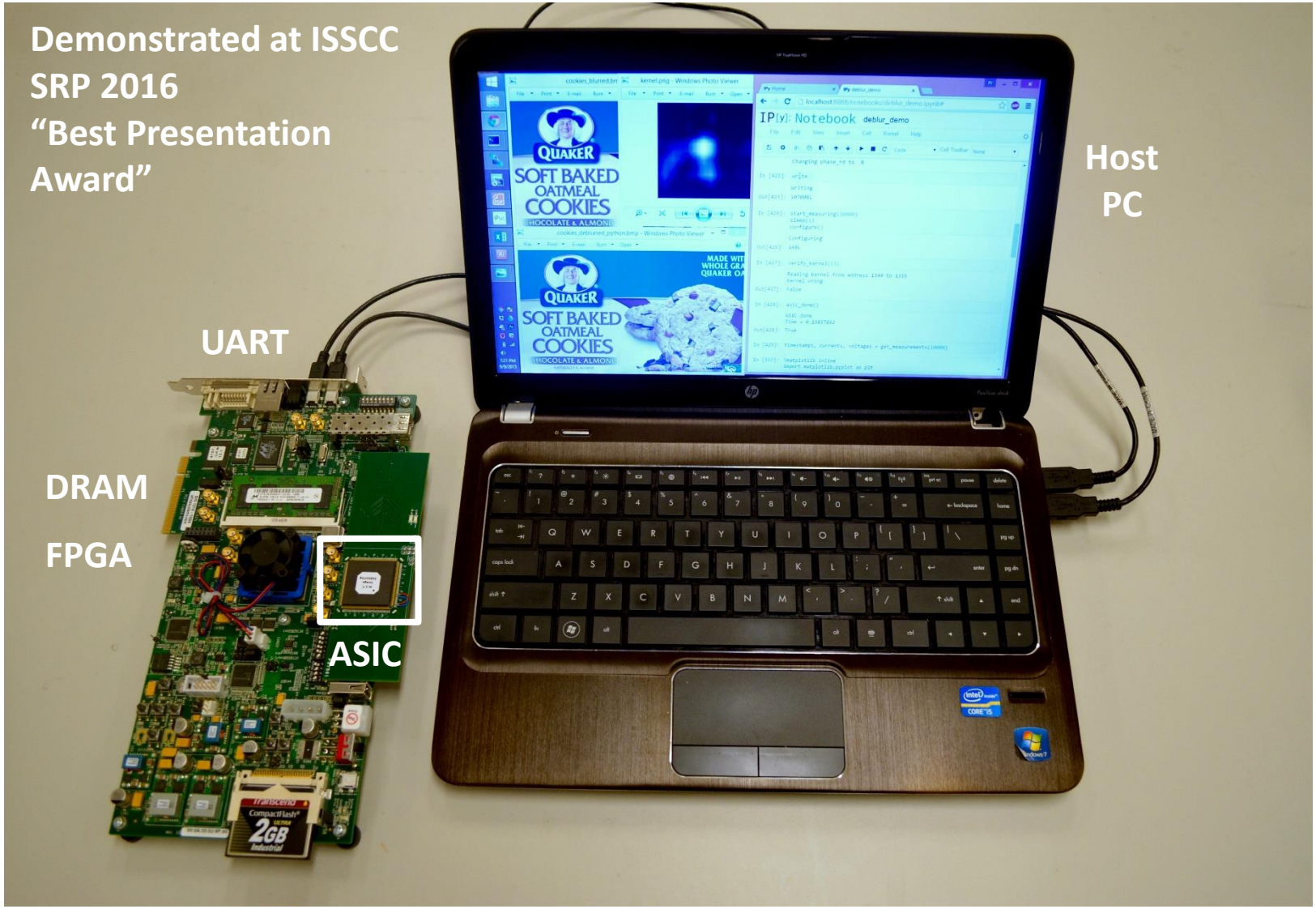


Image Deblurring Accelerator



Technology	40 nm CMOS
Core Area	2.1 mm × 2.1 mm
Gate Count	2046 k
SRAM Size	272 kB
Core Supply Voltage	0.9 V
I/O Supply Voltage	2.5 V
Frequency	25 – 83 MHz
Core Power	59.5 mW (0.9 V, 83 MHz)
Kernel Size	7 × 7 to 29 × 29 pixels

Image Deblurring Demonstration



Deblurring Results on Real Images

Input Blurred Image (1920×1080 pixels) **Deblurred Image** **Estimated Kernel**

Deblurring Results on Real Images

Input Blurred Image (1920×1080 pixels)

Deblurred Image

Estimated Kernel

Energy and Runtime Reduction

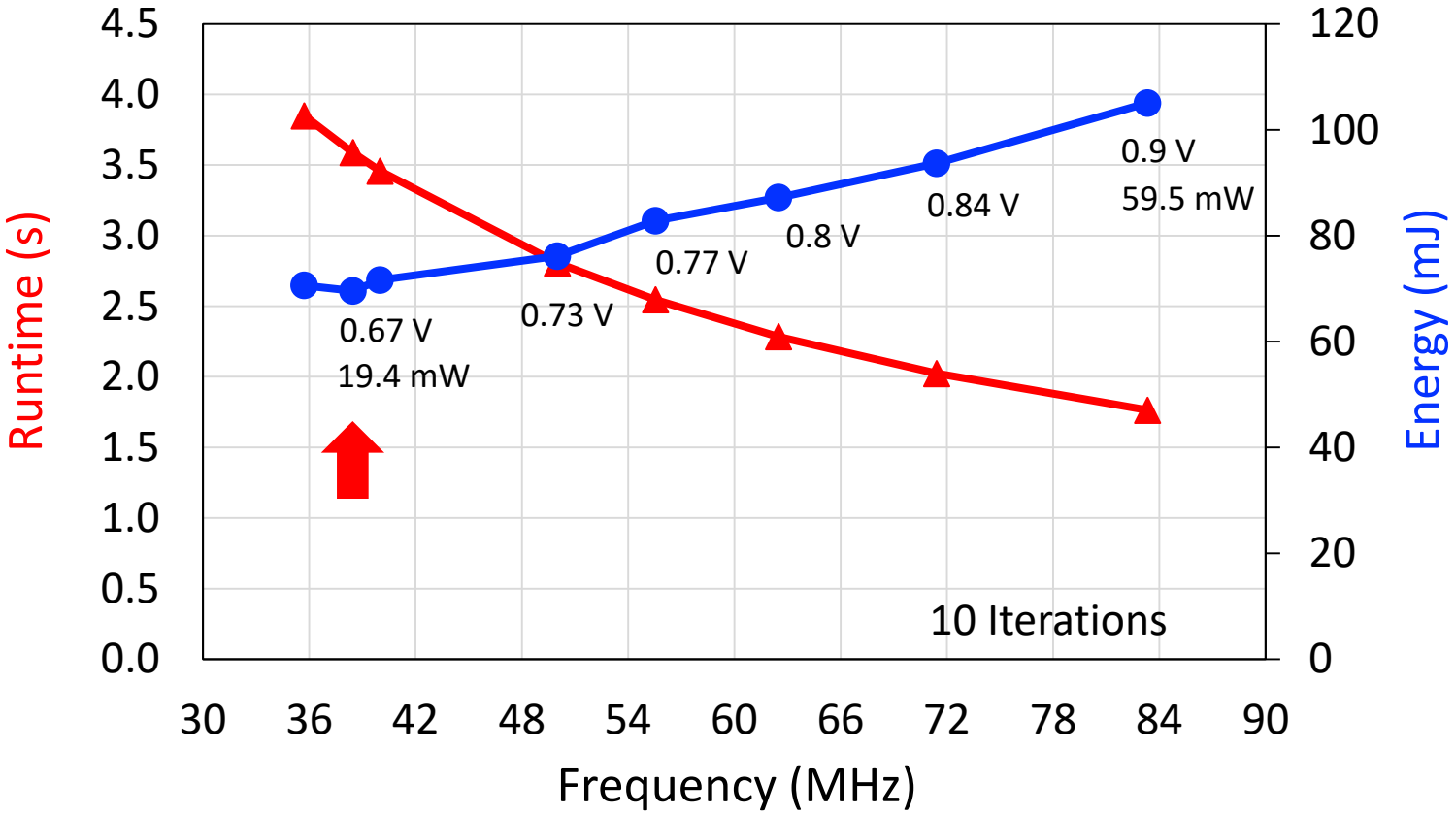
78× reduction in kernel estimation time, 56× reduction in total deblurring time for FullHD image and three orders of magnitude reduction in energy w.r.t. CPU

	Size			Time (s)		Energy (J)
Algorithm and Platform	Kernel	Patch	Image	Kernel Estimation	Full Deblurring	Kernel Estimation
This Work (EM with Accelerator + CPU)	**13×13**	**128× 128**	**1920× 1080**	**1.70**	**2.45**	**0.105**
EM on Intel Core i5	13×13	128× 128	1920× 1080	134.00	134.75	467.000
EM on Samsung Exynos 5422 Cortex-A15	13×13	128× 128	1920× 1080	816.00	-	2284.800
[1] on NVIDIA Tesla C2050 GPU	15×15	-	441× 611	169.70	170.50	-

[1] M. Hirsch, C. J. Schuler, S. Harmeling and B. Schölkopf, "Fast removal of non-uniform camera shake," ICCV, pp. 463-470, 2011.

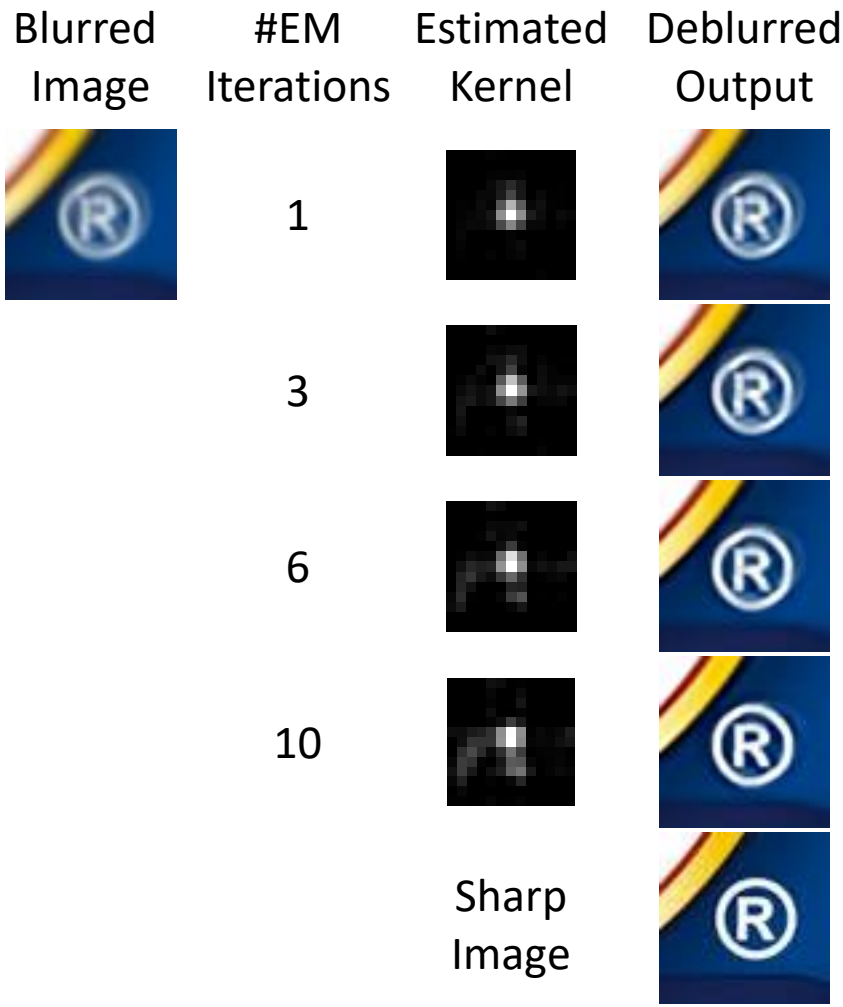
Energy Reduction with Voltage Scaling

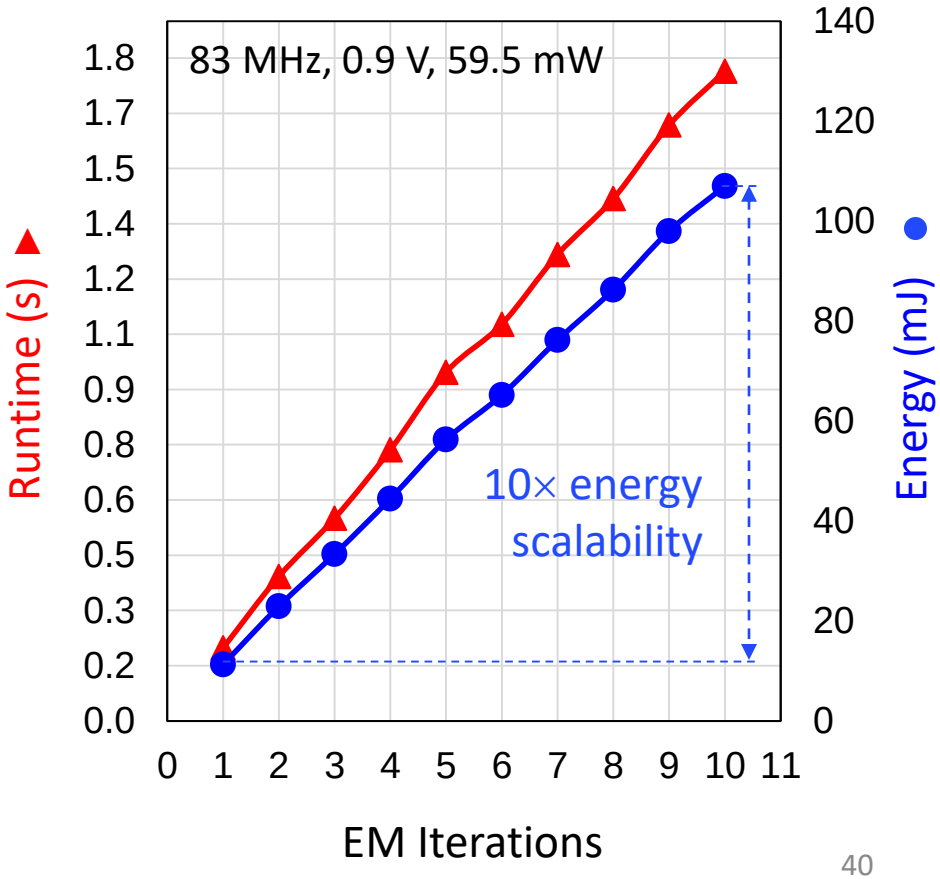
At the minimum energy point, the energy consumption is 33% lower than at nominal, and can be used for batch processing



Energy Scalability with Iterations

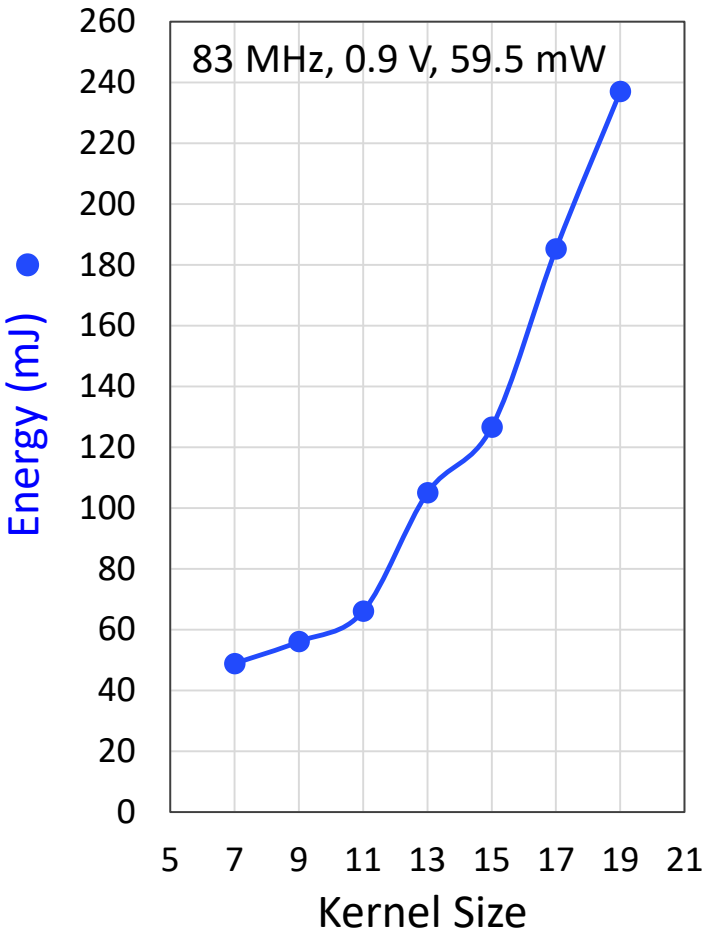
Number of EM iterations can be tuned to trade off image quality with runtime giving 10x energy scalability

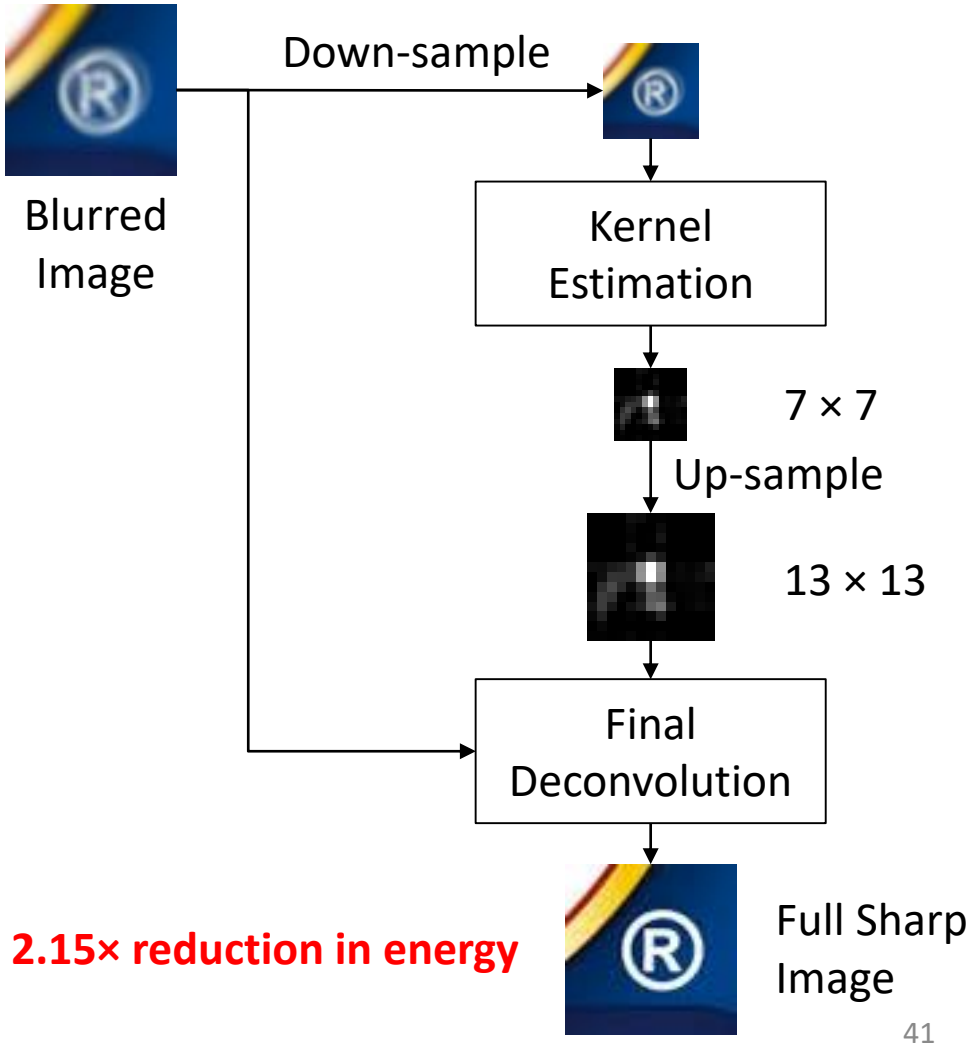




Energy Scalability with Kernel Size

Kernel size can be tuned to achieve energy scalability





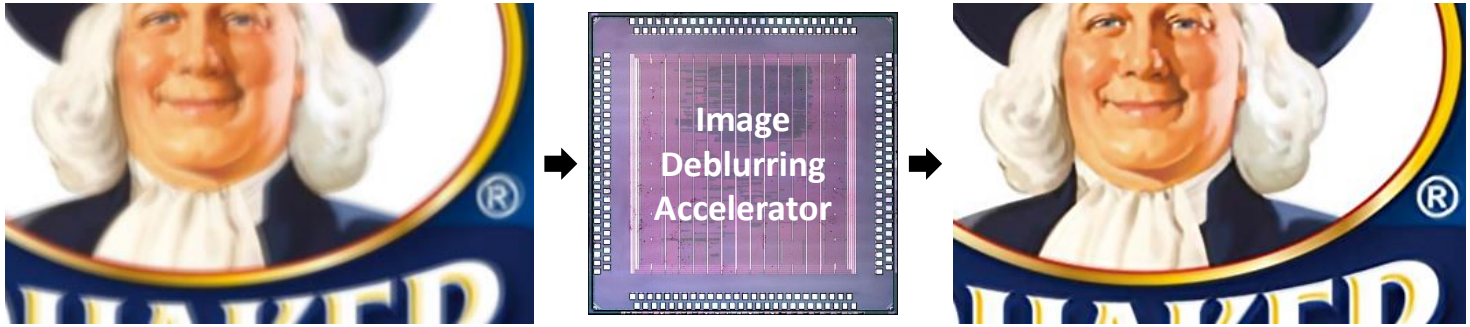
Summary: Image Deblurring Accelerator

- **First hardware accelerator** for image deblurring employing techniques such as
  - **Computation Reuse:** Statically based on shared computation and dynamically based on sparse data updates
  - **Hardware Sharing** between non-concurrent stages to reduce area and leakage power
  - **Memory management:** On-chip scratchpad buffer to reduce memory bandwidth
- **78× reduction** in kernel estimation time, and 56× reduction in total deblurring time of FullHD images
- **3 orders of magnitude reduction** in energy
- **10× energy scalability** allows trading off runtime with image quality in energy-constrained scenarios

Energy-Efficient Imaging Accelerators

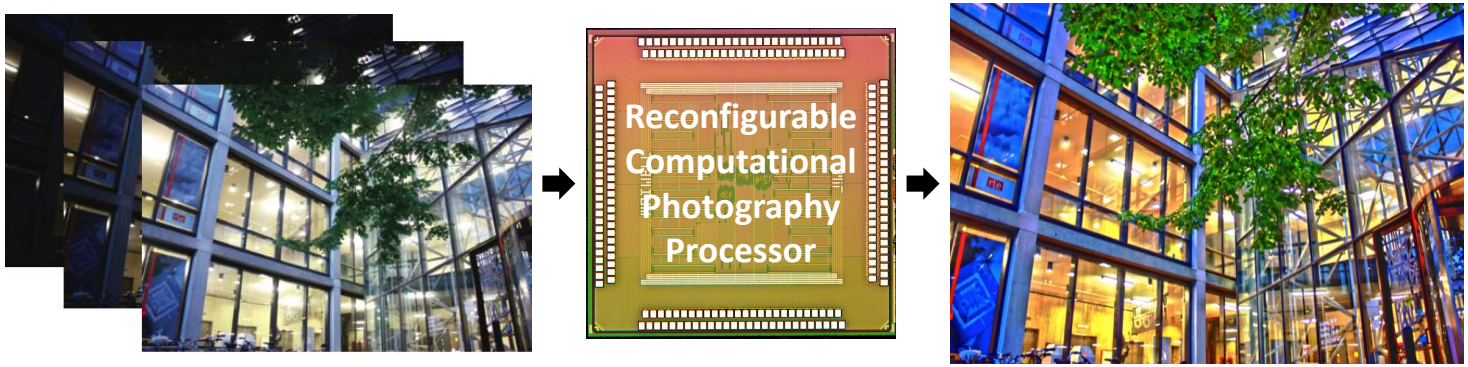
Image Deblurring

[**P. Raina**, M.Tikekar, A. P. Chandrakasan **ESSCIRC 2016, JSSC 2017]**



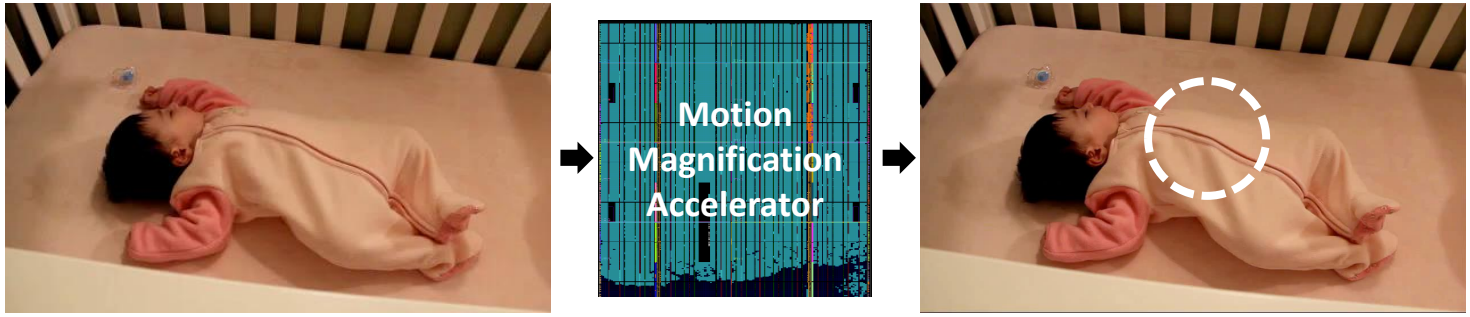
HDR & Low Light Imaging

[R. Rithe, **P. Raina**, N. Ickes, S. Tenneti, A. P. Chandrakasan **ISSCC 2013, JSSC 2013]**



Motion Magnification

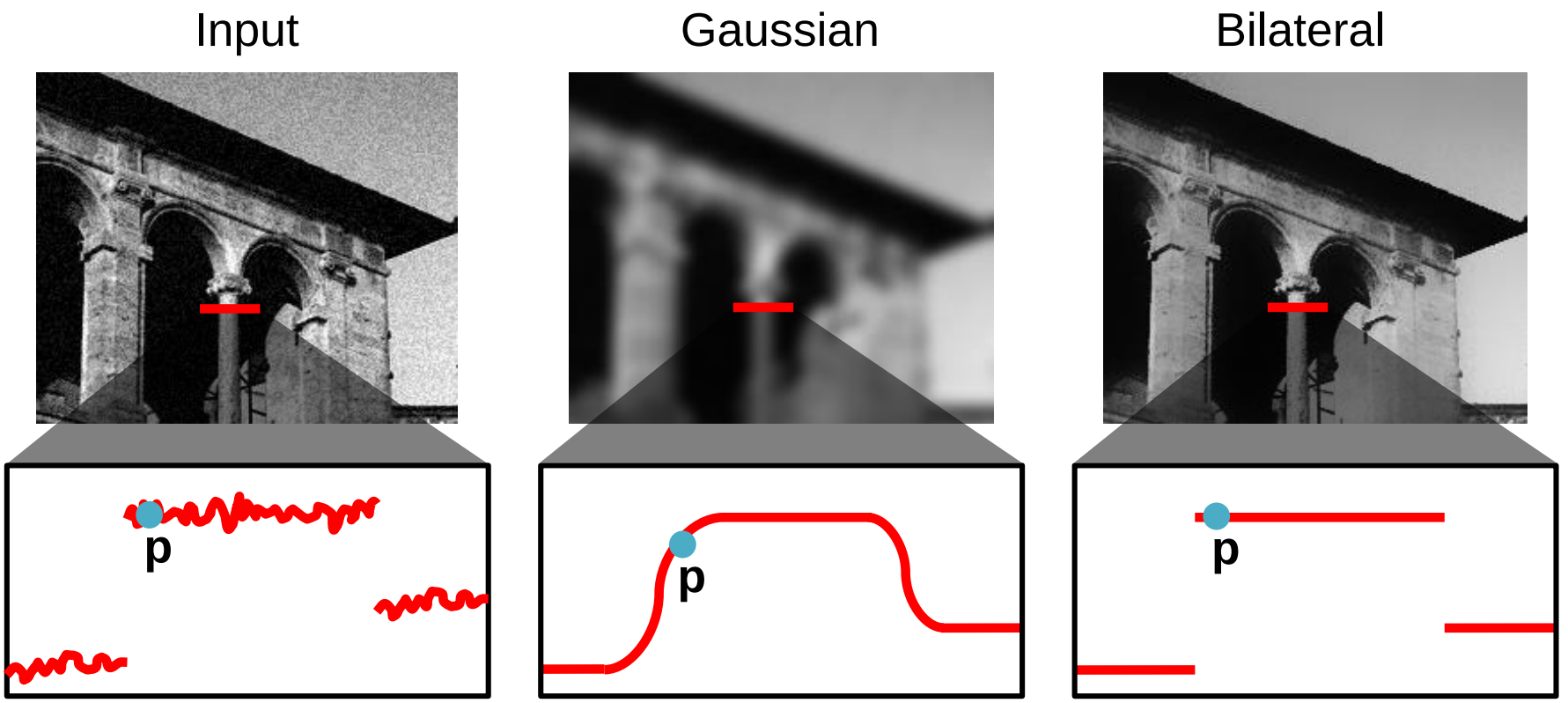
[**P. Raina**, D. Jeon, W. T. Freeman, F. Durand, A. P. Chandrakasan, In progress]



Gaussian vs Bilateral Filtering

$$I_p^G = \sum_{n=-N}^{N} G_S(n) \times I_{p-n}$$

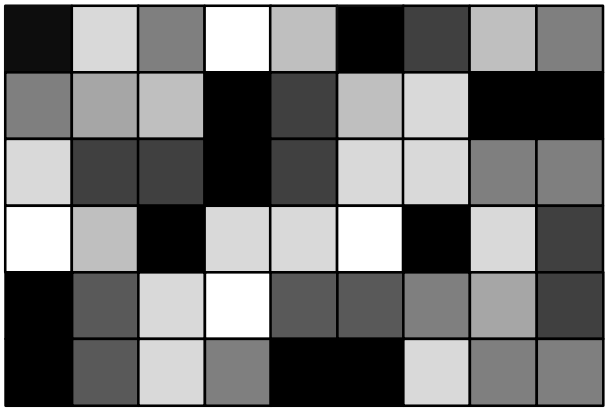
$$I_p^B = \sum_{n=-N}^{N} G_S(n) \times G_I(I_p - I_{p-n}) \times I_{p-n}$$



[Tomasi, ICCV 1998]

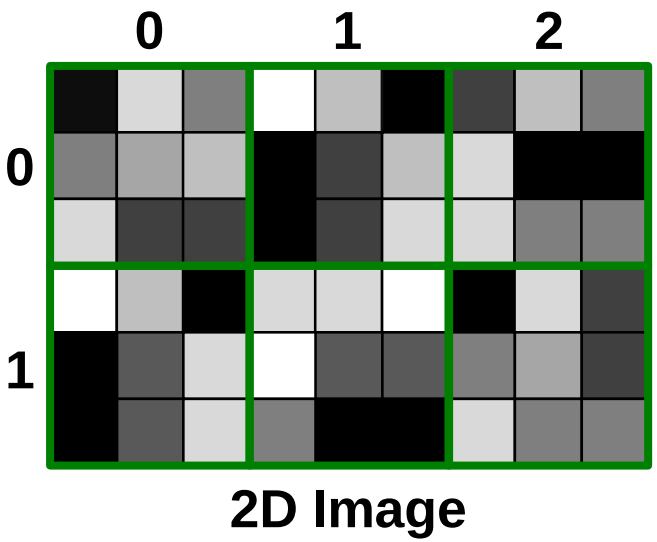
Bilateral Grid

[Chen, SIGGRAPH 2007]

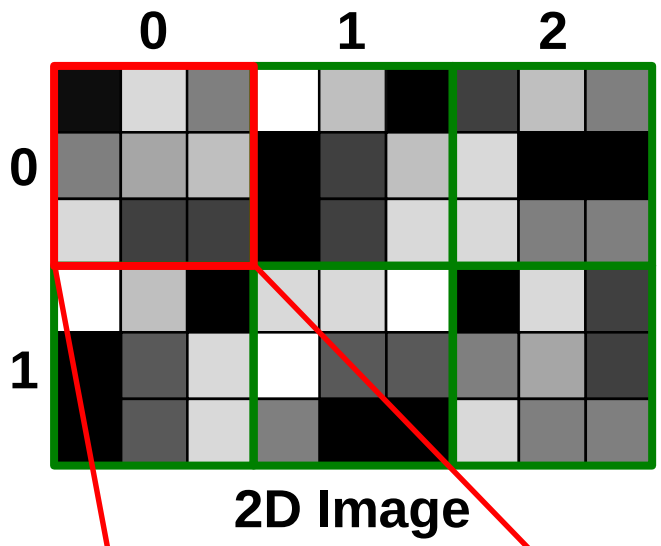


2D Image

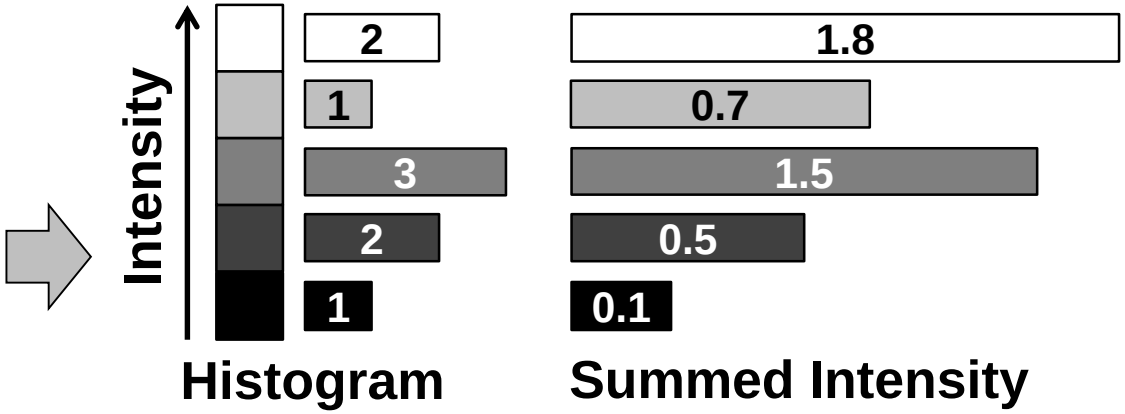
Bilateral Grid



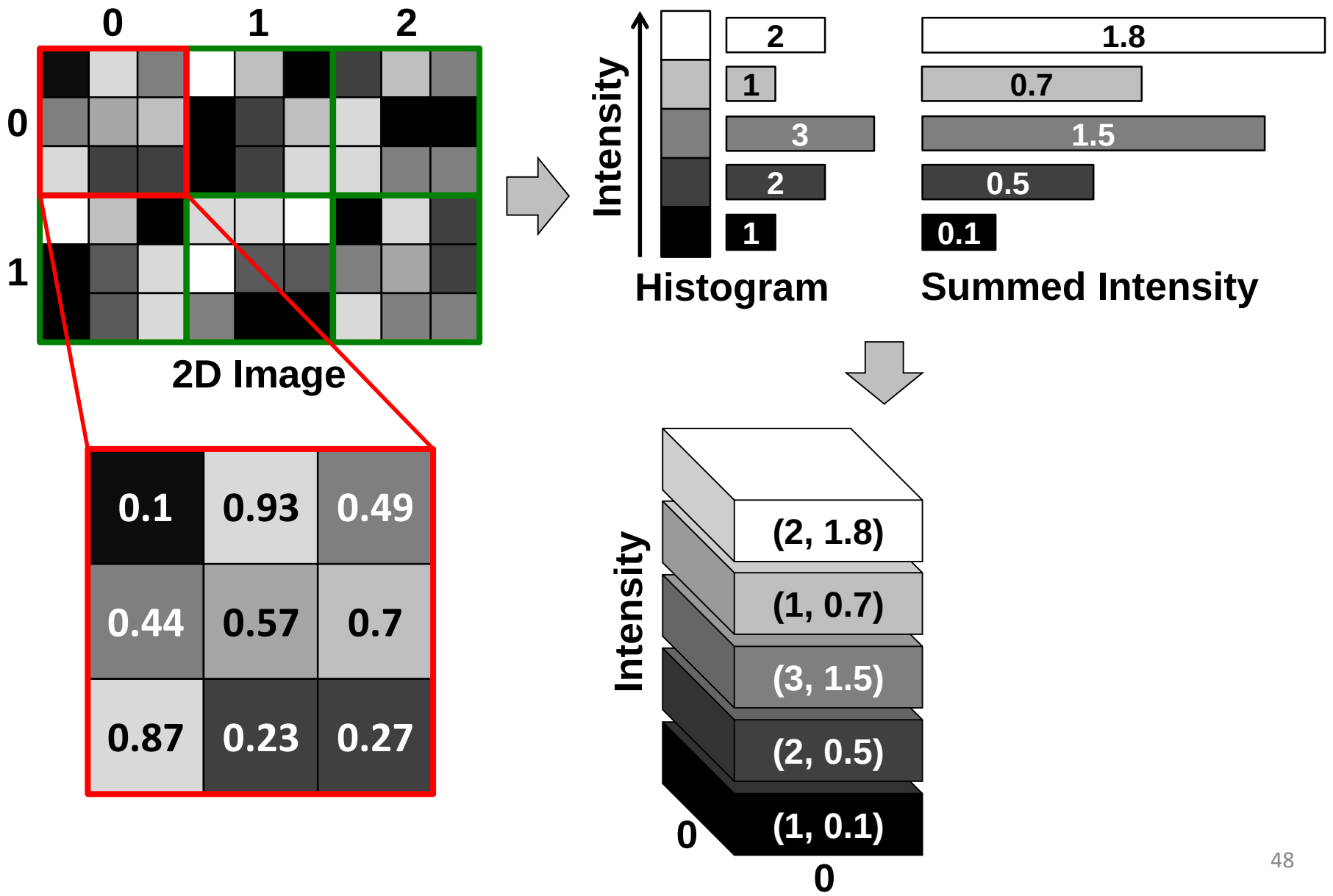
Bilateral Grid



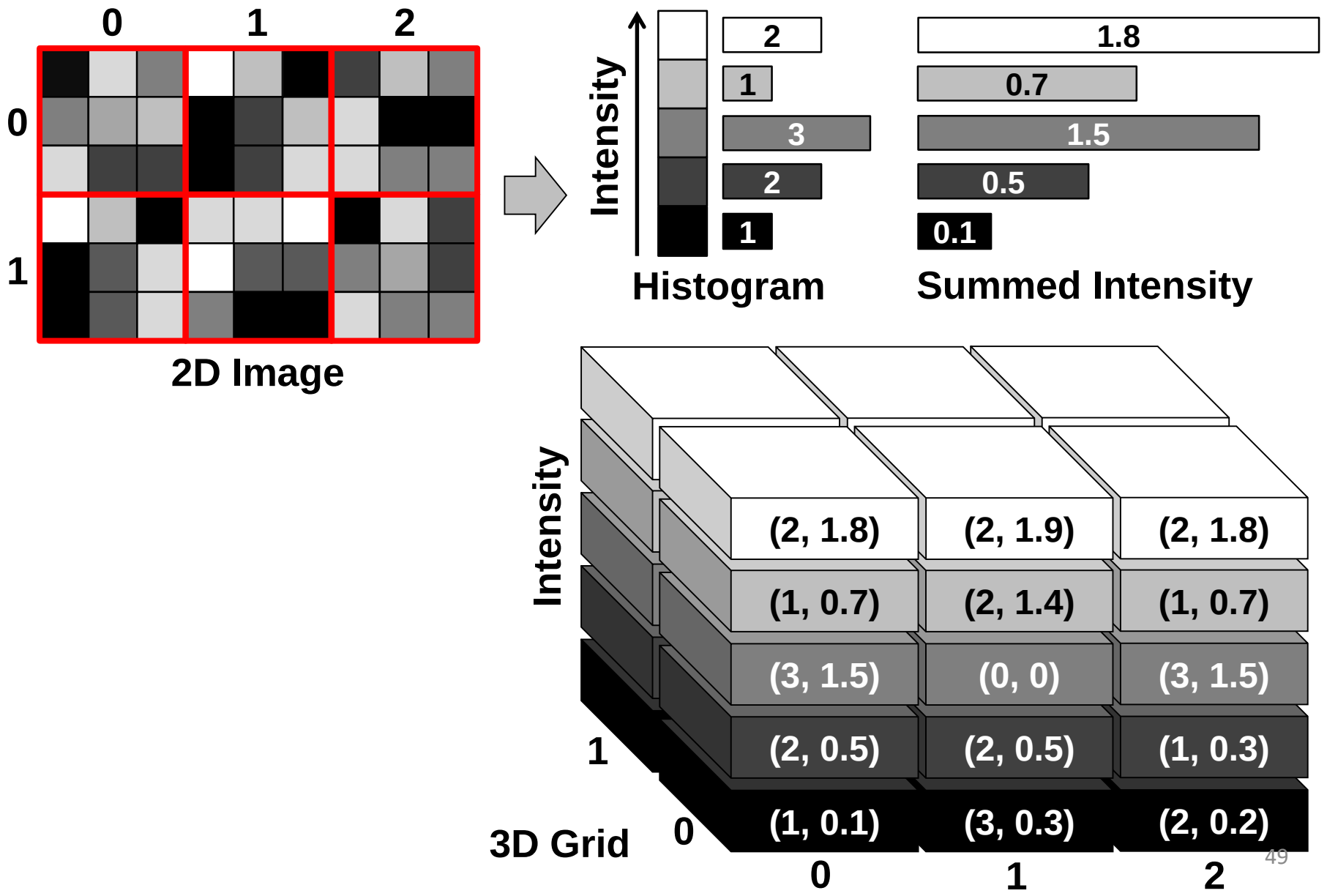
0.1	0.93	0.49
0.44	0.57	0.7
0.87	0.23	0.27



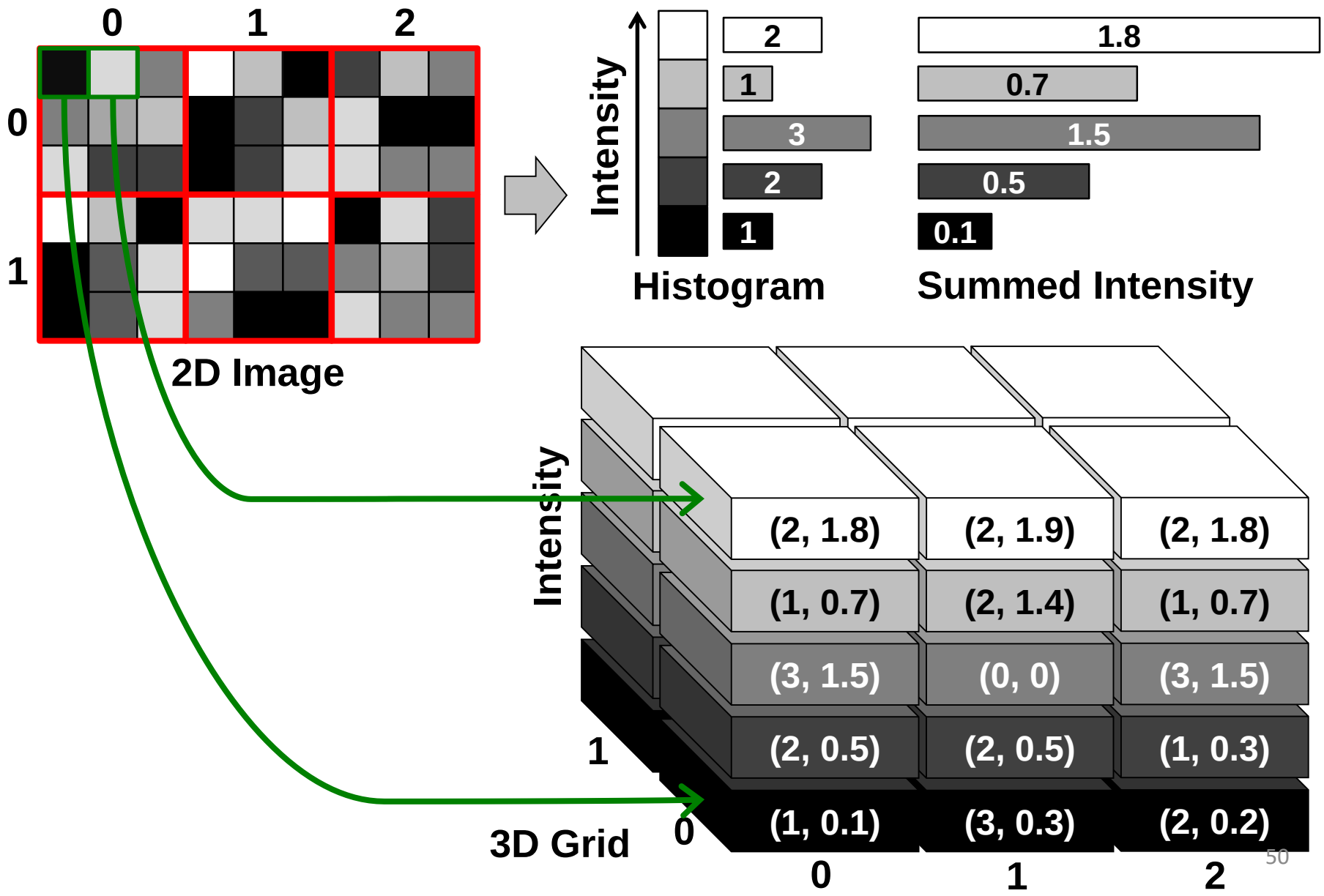
Bilateral Grid



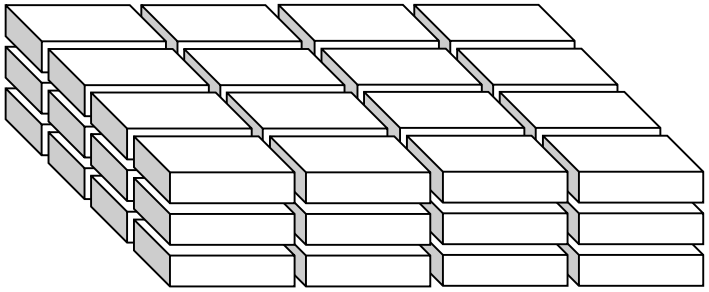
Bilateral Grid



Bilateral Grid

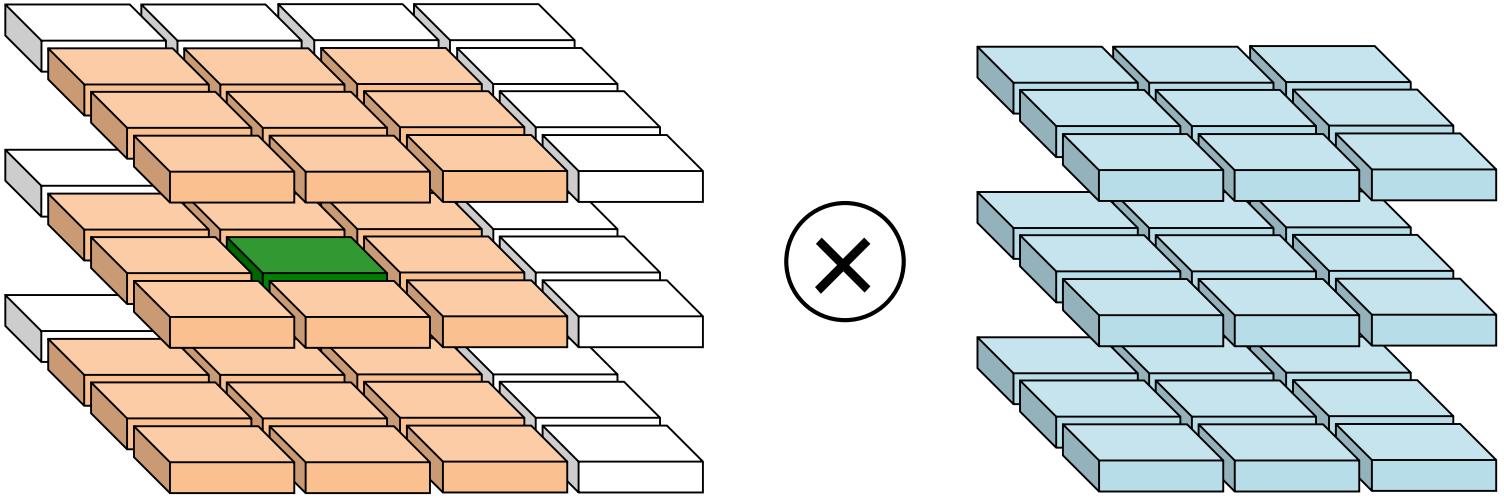


Grid Filtering (Convolution)



Grid

Grid Filtering (Convolution)



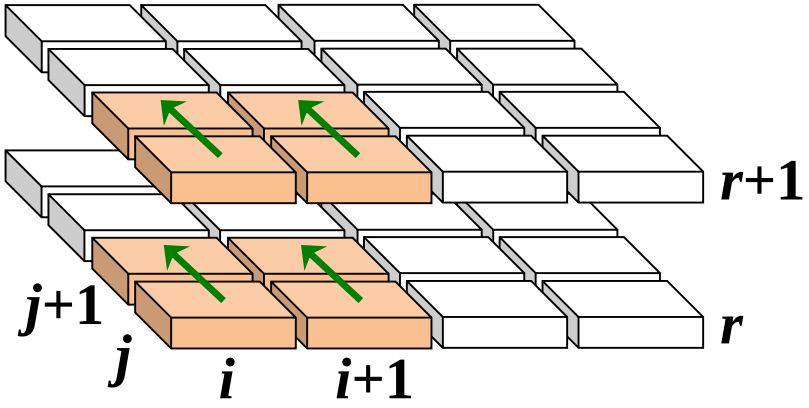
Grid

Gaussian Kernel

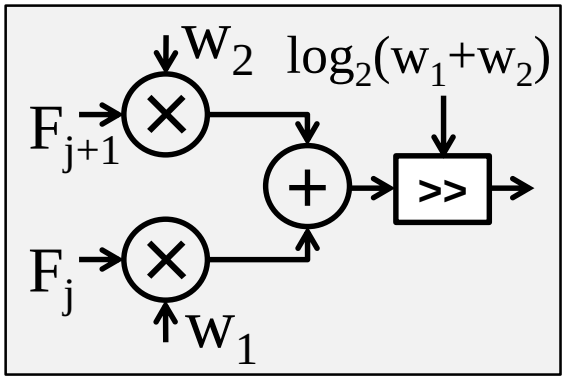
The grid intensities and weights are convolved with a 3×3×3 Gaussian kernel – equivalent to Bilateral filtering in the 2D image domain

Grid Interpolation

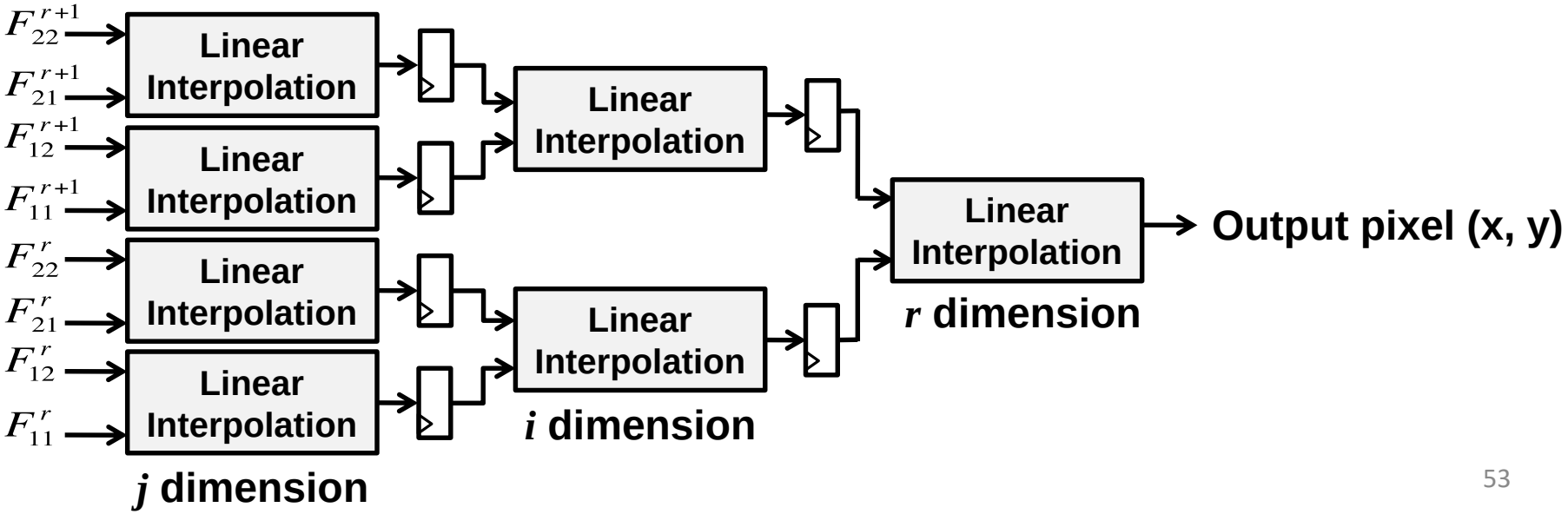
- **Output pixel at location (x, y) is obtained by tri-linear interpolation of 2×2×2 filtered grid**



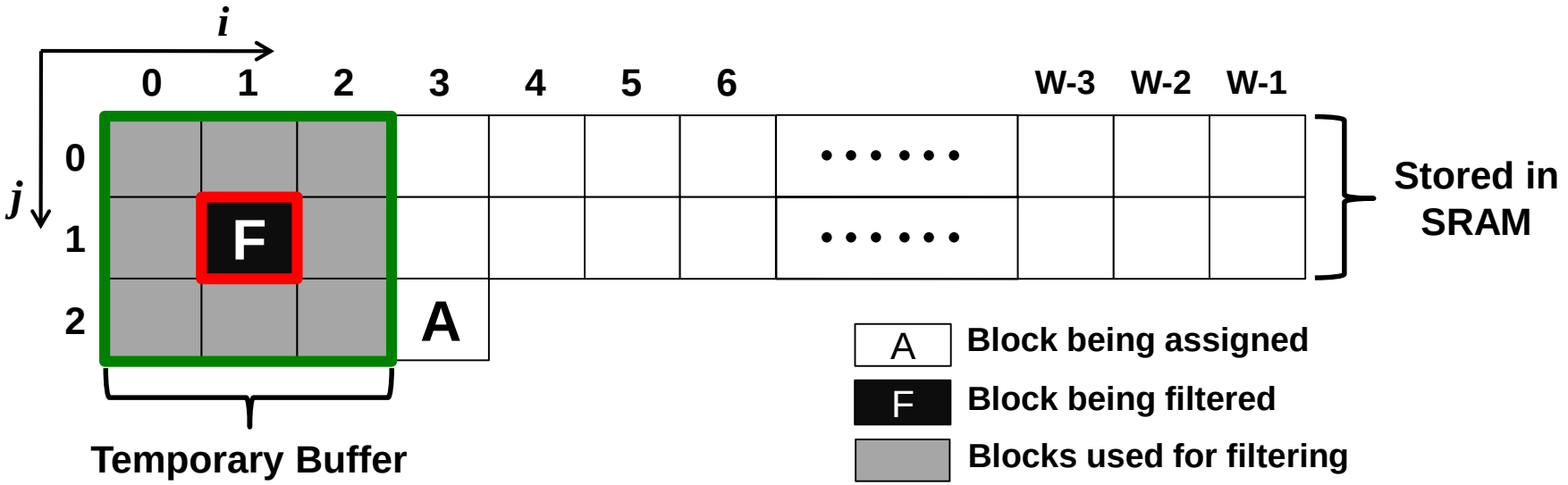
$$w_1 = y/\sigma_s - j$$
$$w_2 = j+1 - y/\sigma_s$$



Linear Interpolation

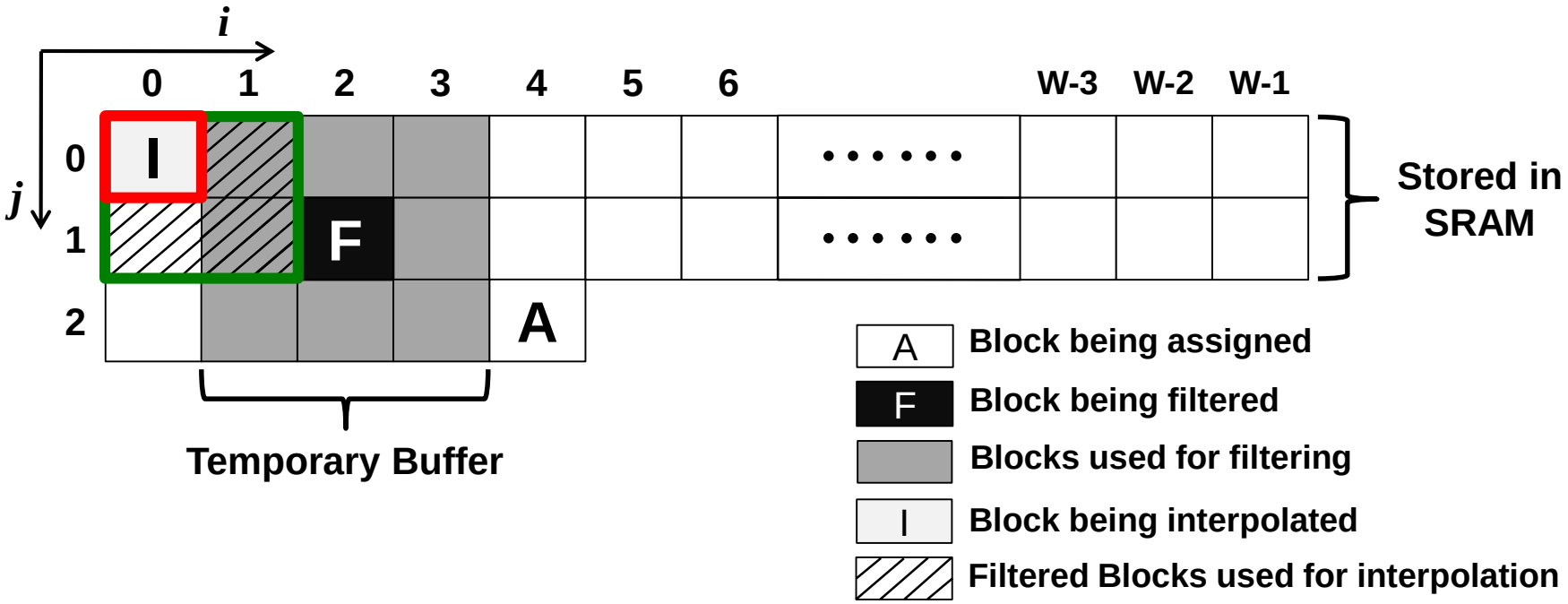


Bilateral Filter Engine: Line Buffering



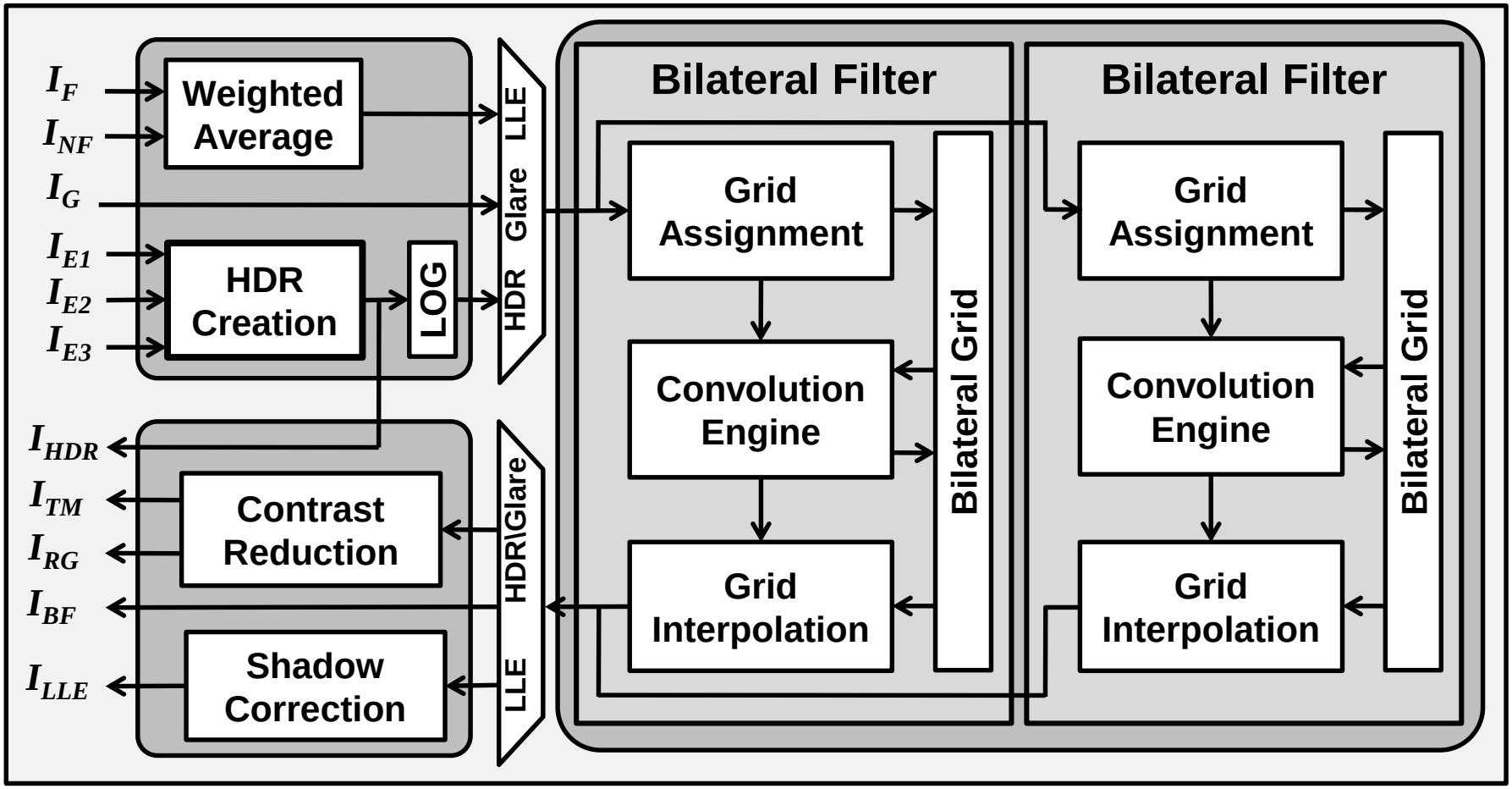
- **Grid Filtering**
  - **Convolution kernel: 3×3×3**
  - **Convolution engine begins filtering block (i, j) when block (i+2, j+1) is being assigned**

Bilateral Filter Engine: Line Buffering



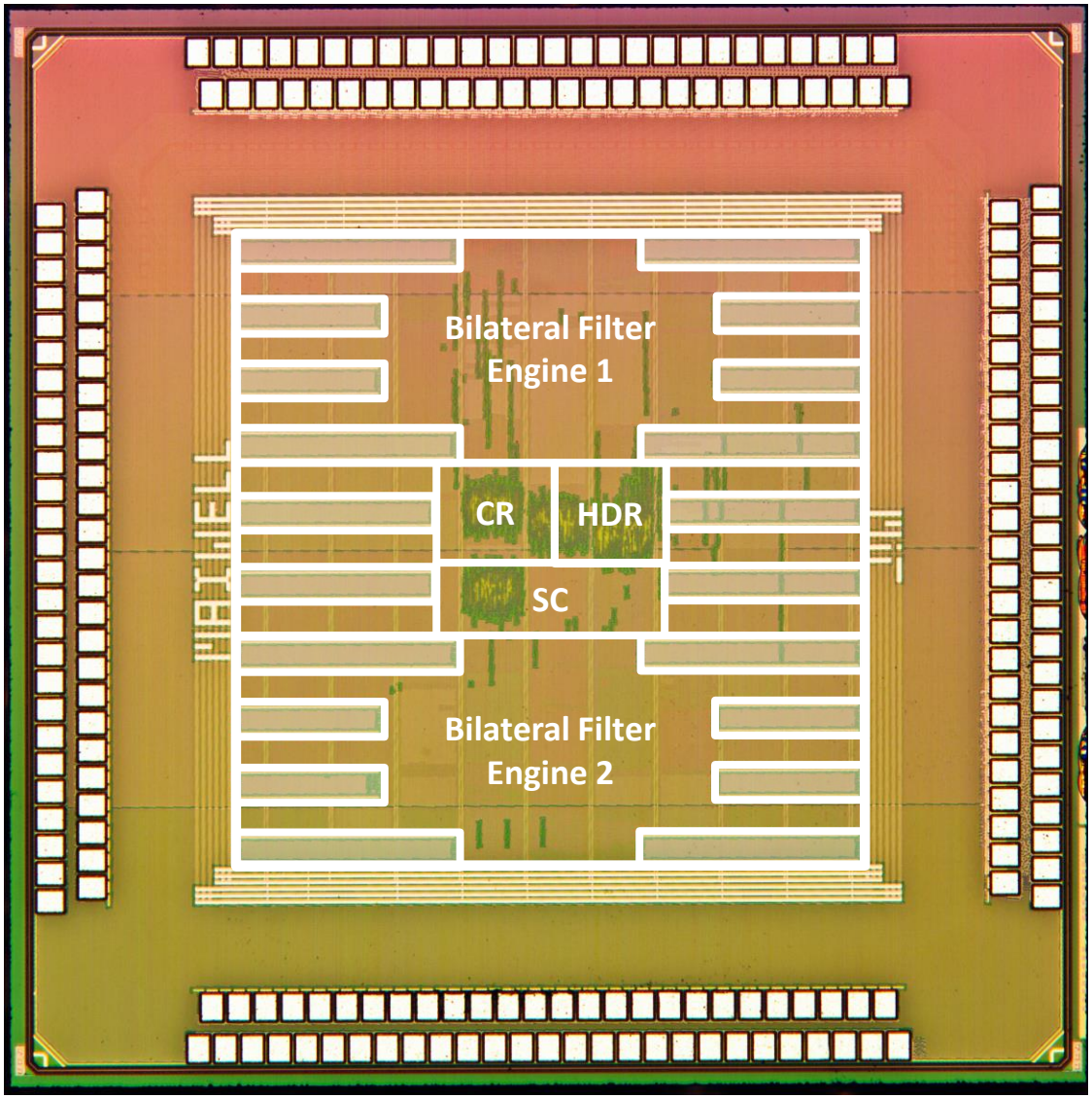
- **Grid Interpolation**
  - **Tri-linear interpolation: 2×2×2**
  - **Interpolation engine begins interpolating block (i, j) when block (i+2, j+1) is being filtered and block (i+4, j+2) is being assigned**

Reconfigurable Processor



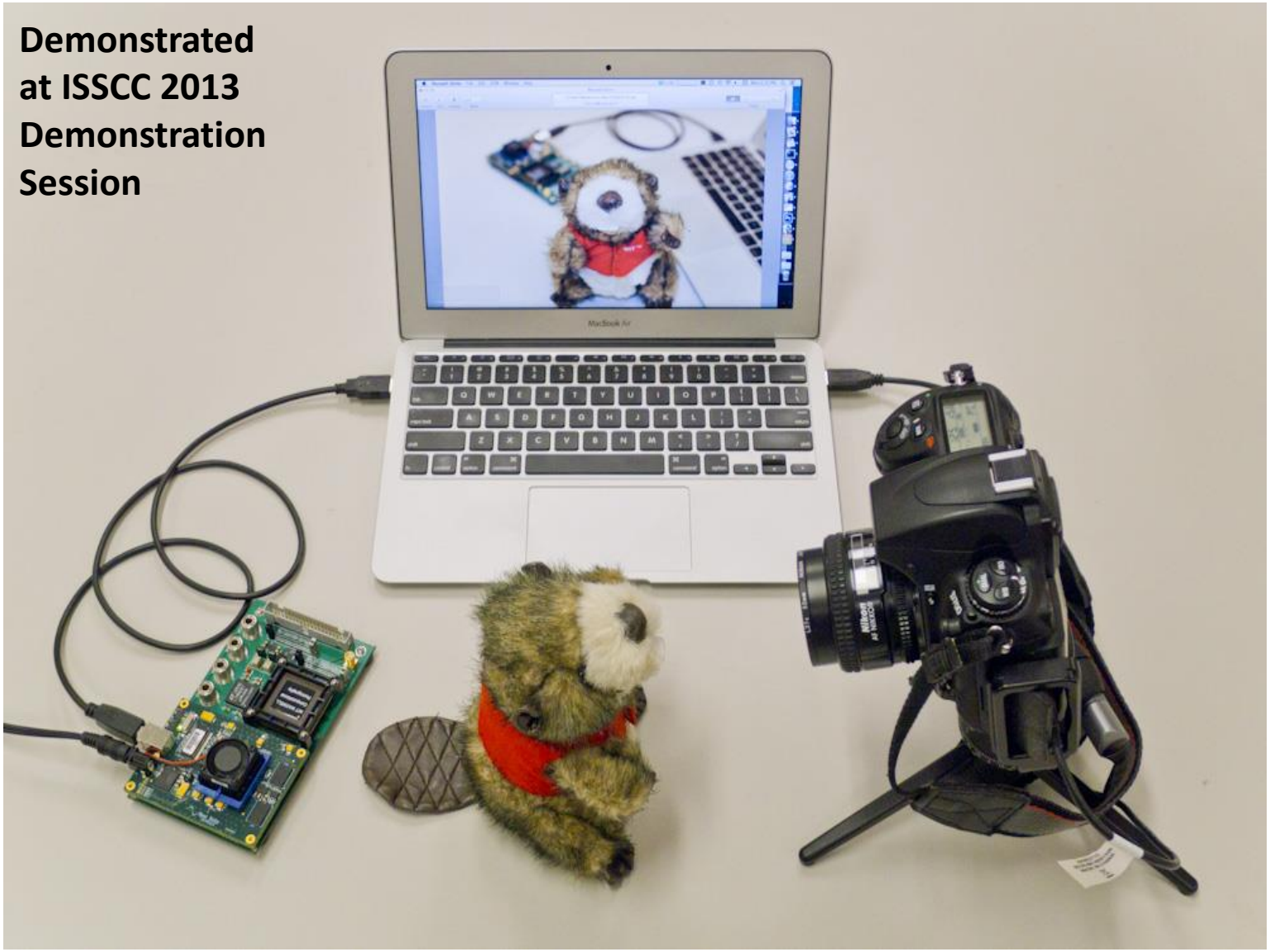
[R. Rithe, **P. Raina**, N. Ickes, S. Tenneti, A. P. Chandrakasan **ISSCC 2013, JSSC 2013**]

Processor Implementation



Technology	40nm CMOS
Core Area	1.1mm × 1.1mm
Transistor Count	1.94 million
SRAM	21.5 kB
Core Supply Voltage	0.5V to 0.9V
I/O Supply Voltage	1.8V to 2.5V
Frequency	25 – 98 MHz
Core Power	17.8mW (0.9V)

Real-Time Demonstration System



HDR Imaging Results

Low-light Imaging Results

Processor Performance

Processor	Technology (nm)	Frequency (MHz)	Power (mW)	Runtime (s)	Energy (mJ)
Intel Atom	32	1800	870	4.96	4315
Qualcomm Snapdragon	28	1500	760	5.19	3944
Samsung Exynos	32	1700	1180	4.05	4779
TI OMAP	45	1000	770	6.47	4981
This Work	**40**	**98**	**17.8**	**0.77**	**13.7**

5.2x

287x

For 10 Mpixel image size

Summary: Reconfigurable Computational Photography Processor

- **Multi-application processor** for computational photography – implements HDR, low-light imaging and glare reduction
- **On-chip cache** (21.5kB) reduces **memory bandwidth** from 1.6 GB/s to 356 MB/s (**78% lower**) and **memory power** form 175 mW to 108 mW (**38% lower**)
- **Reconfigurable bilateral grid** for **7.2x energy-scalablity** from 1.37 mJ/Mpixel at (16 levels, 16x16 blocks) to 0.19 mJ/Mpixel at (4 levels, 128x128 blocks)
- **Real-time processing of HD images** with 17.8mW power consumption at 0.9V, **280x more energy-efficient** compared to mobile CPU
- Energy scalable implementation enables efficient integration into mobile devices

Designing Domain Specific Architectures

- Future is about ASICs but with some programmability
- Specialization for
  - For a class of problems
  - For a platform (mobile, cloud, automotive, IoT) with different design targets (energy, throughput, latency, accuracy)

Designing Domain Specific Architectures

- Efficiency comes from
  - Modifying the algorithm (at different levels) such that it becomes well suited for hardware implementation
  - Reduced or amortized instruction overhead by doing a lot of work per instruction by exploiting
    - Data parallelism
    - Complex operations
  - Reducing precision and using the right data encoding (affects both compute and memory)
  - Designing a memory hierarchy that exploits locality

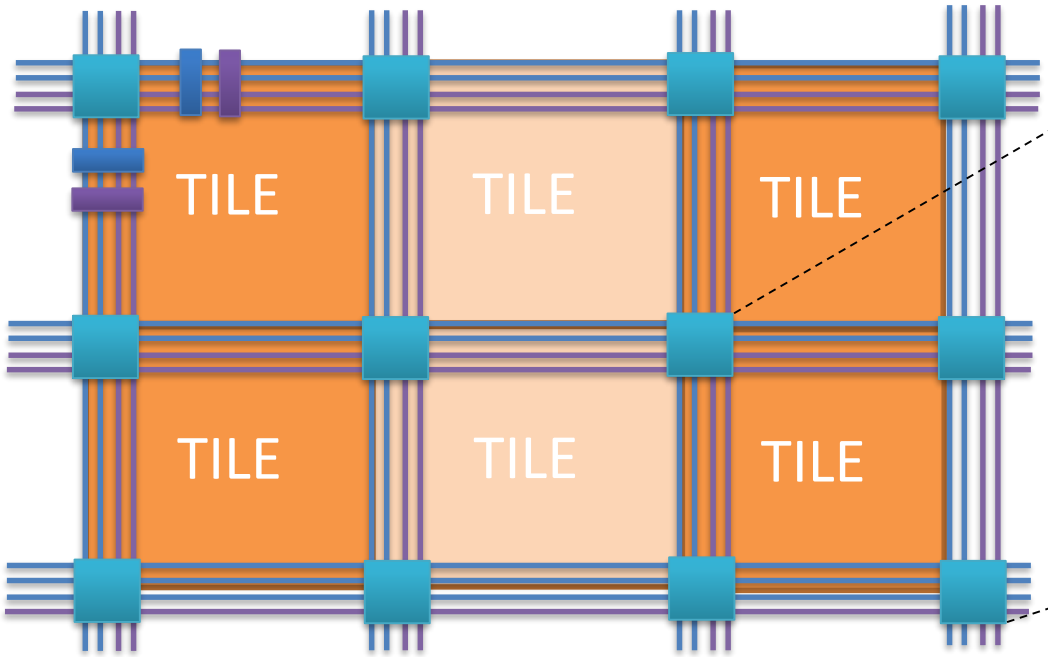
- Key challenges
  - Algorithm-architecture co-design
  - Designing a domain specific programming model
  - Lowering design costs

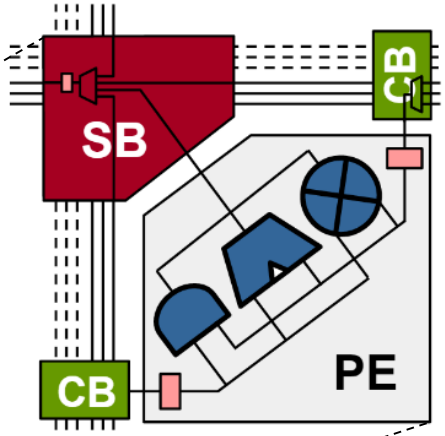
WHAT AM I WORKING ON RIGHT NOW?

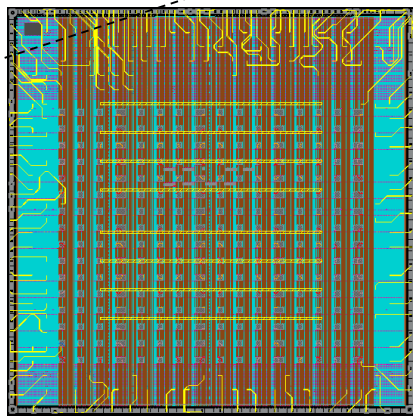
Stanford Agile Hardware (AHA) Project

With Mark Horowitz, Pat Hanrahan, Clark Barrett, Kayvon Fatahalian and all AHA Group Students

Coarse Grained Reconfigurable Array (CGRA)







16 bit bus

1 bit bus

Switch box

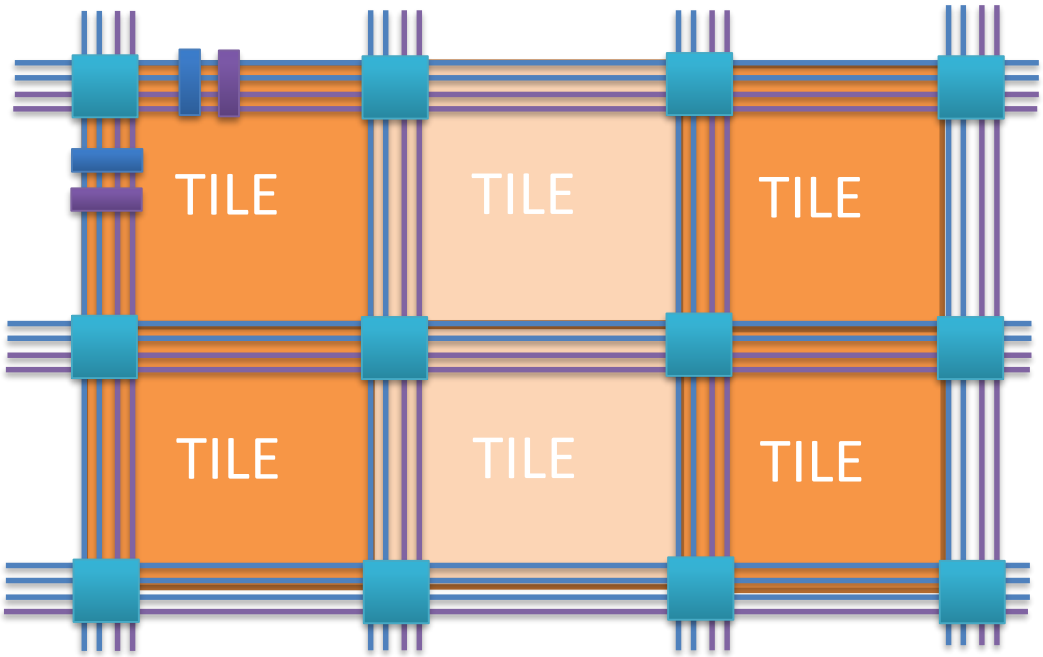
16 bit Connection box

1 bit Connection box

Stanford Agile Hardware (AHA) Project

With Mark Horowitz, Pat Hanrahan, Clark Barrett , Kayvon Fatahalian, and all AHA Group Students

Coarse Grained Reconfigurable Array (CGRA)



16 bit bus

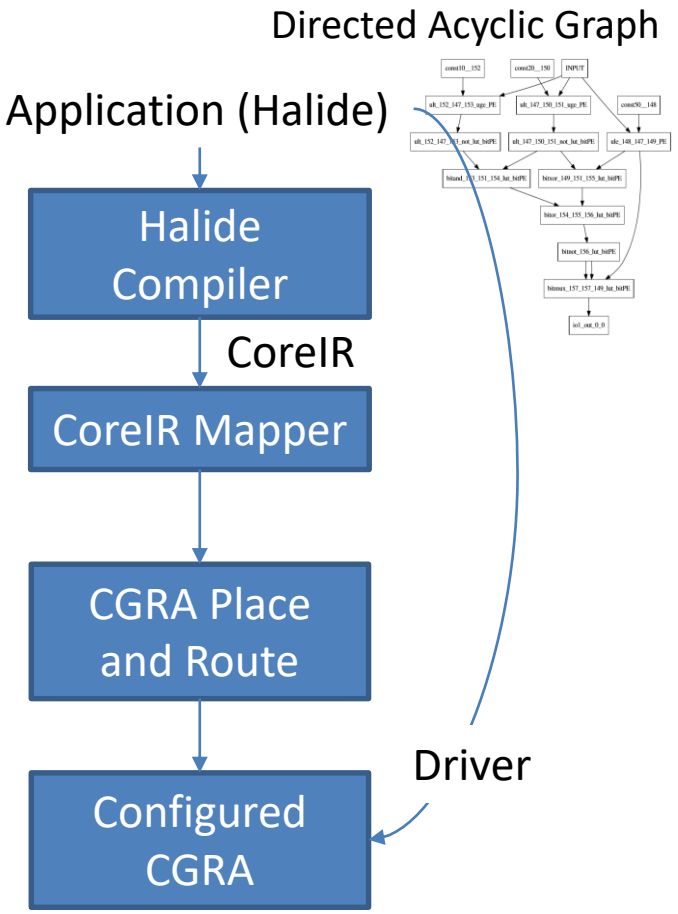
1 bit bus

Switch box

16 bit Connection box

1 bit Connection box

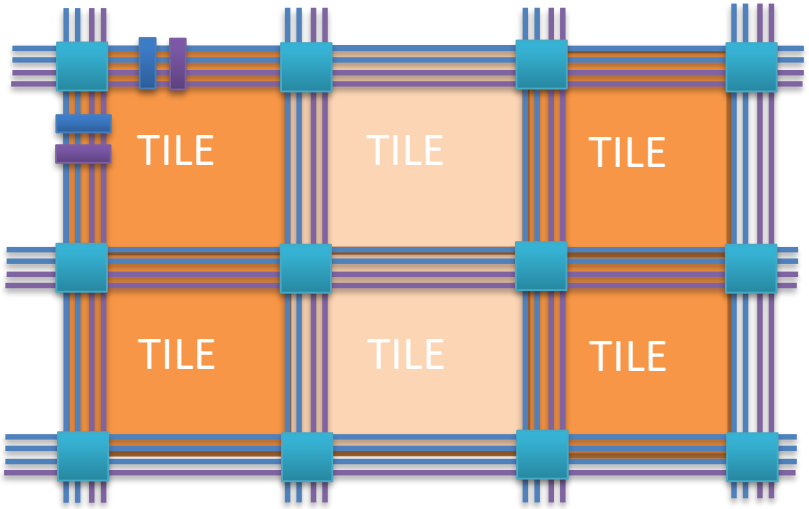
Programming Toolchain



Stanford Agile Hardware (AHA) Project

With Mark Horowitz, Pat Hanrahan, Clark Barrett , Kayvon Fatahalian, and all AHA Group Students

Coarse Grained Reconfigurable Array (CGRA)



16 bit bus

1 bit bus

Switch box

16 bit Connection box

1 bit Connection box

Application (Halide)

Logical Description (Verilog – like)

Logical Description' (Verilog-like)

Floorplan

Logical Description'' (Verilog-like)

Testing Collateral

Logical Description''' (Verilog-like)

Verilog

Physical Design Collateral

Documentation

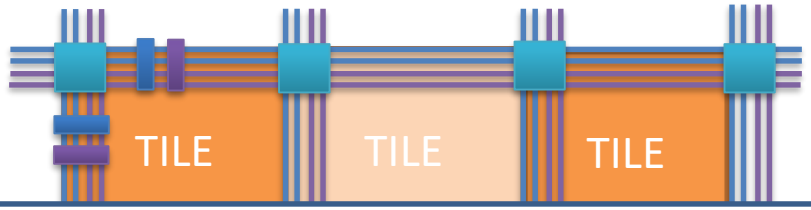
Staged generator that captures

- Logical design
- Physical design
- Testing/verification
- Software API

Stanford Agile Hardware (AHA) Project

With Mark Horowitz, Pat Hanrahan, Clark Barrett , Kayvon Fatahalian, and all AHA Group Students

Coarse Grained Reconfigurable Array (CGRA)



Application (Halide)

Logical Description (Verilog – like)

Staged generator that captures
- Logical design
- Physical design
- Testing/verification

Software first approach combined with an end-to-end hardware generator enables rapid design space exploration over a large set of applications with increasing levels of fidelity

1 bit Connection box

Logical Description''' (Verilog-like)

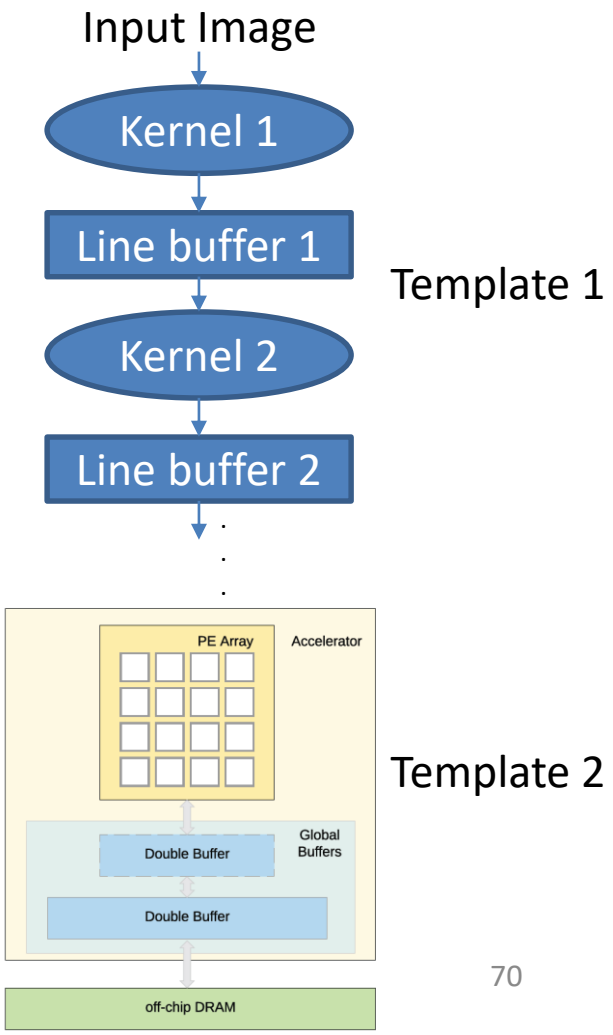
Verilog | Physical Design Collateral | Documentation

Halide to ASIC

With Xuan Yang, Kartik Prabhu, Mark Horowitz and Kayvon Fatahalian

- Extension of Halide to CGRA/FPGA
- Hardware generation using **templates**
- Questions that we want to address
  - Given a set of templates and and an application (graph of kernels), how do you automatically decide which template to use for which sub-graph?
  - How do you optimally partition resources (figure out parameters for the template instantiations)?
  - How do you virtualize the templates when you are resource constrained?
  - How do we find the templates?

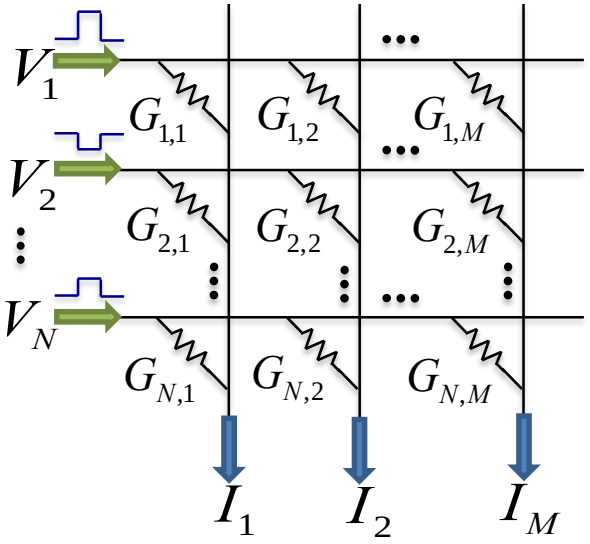
Example: Image processing + DNN kernels



Leveraging New Memory Technologies

With Haitong Li, Weier Wan, Philip Wong and Subhasish Mitra

- Many applications are memory bound
  - Avoid reading out - leverage new memory technologies for performing processing in memory
  - Leverage new memory technologies that offer high bandwidth via 3D integration



Designing Domain Specific Architectures

- Efficiency comes from
  - Modifying the algorithm (at different levels) such that it becomes well suited for hardware implementation
  - Reduced or amortized instruction overhead by doing a lot of work per instruction by exploiting
    - Data parallelism
    - Complex operations
  - Reducing precision and using the right data encoding (affects both compute and memory)
  - Designing a memory hierarchy that exploits locality

- Key challenges
  - Algorithm-architecture co-design
  - Designing a domain specific programming model
  - Lowering design costs