Convolution Engine: Balancing Efficiency & Flexibility in Specialized Computing

Did the heavy lifting but could not come today

Wajahat Qadeer, Rehan Hameed,

Ofer Shacham,

That's me ☺

Preethi Venkatesan, Christos Kozyrakis, Mark Horowitz

Stanford University

Smile, you're on camera

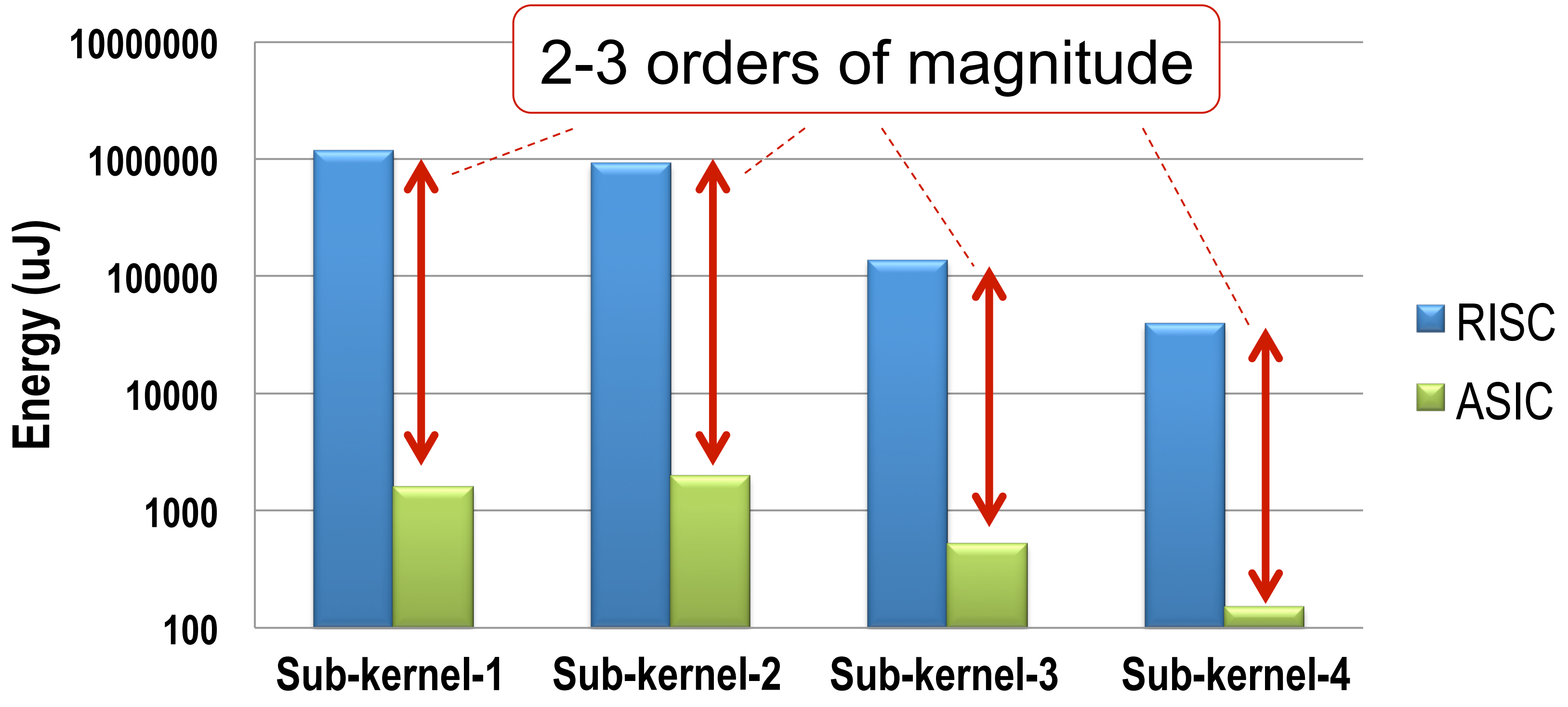
- By show of hands, who here has an (HD) camera on them?
- How many CPU's/GPU's in the room?
- How many of those xPU's are used for the image processing?

Imaging and video systems

- **High computational requirements, low power budget**
  - Stills: ~10M pixels x 10 frames per second
  - Video: ~2M pixels x 30 frames per second
  - ~400 math operations per pixel (just for the image acquisition)
- **On CPU… not enough horse power**
- **On GPU… too much power**
- **Typically use special purpose custom HW**
  - About 500X better performance, 500X lower energy than CPU

Example: H.264 encoder on RISC vs. ASIC

- **By coupling compute and storage closely together, ASIC's are orders of magnitude performance and energy more efficient**



* R. Hameed et. al., Understanding Sources of Inefficiency in General-Purpose Chips. ISCA '10

We are solving the wrong problem!

- Yes, ASIC is 1000X more efficient than general purpose
- Yes, general purpose is more programmable than ASIC
- Yes, we can make each one marginally better

- But those are good answers to all the wrong questions!

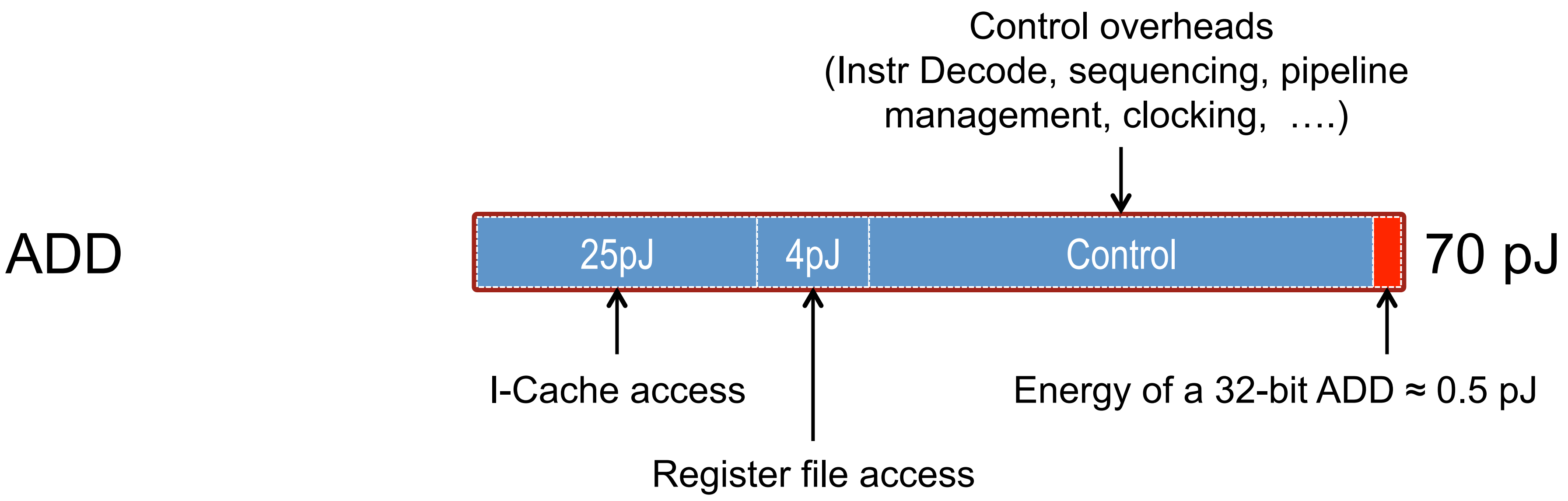
- The right questions:

  Why is the RISC energy so high?

  What type of computation can we make efficient?

  Can we make it just 100X better but keep it programmable?

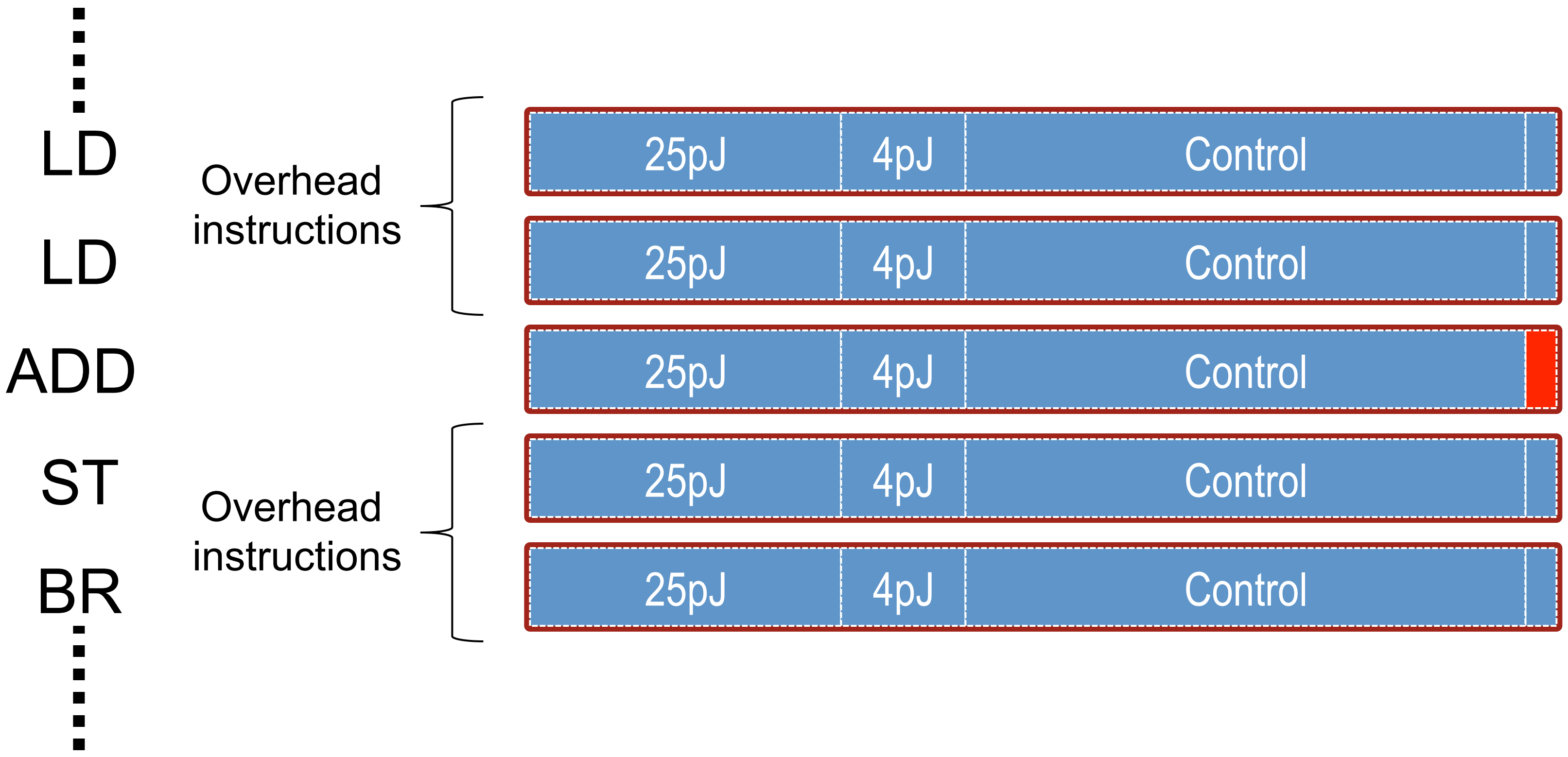
Anatomy of a RISC Instruction



Control overheads
(Instr Decode, sequencing, pipeline management, clocking, ….)
ADD
25pJ
4pJ
Control
70 pJ
I-Cache access
Energy of a 32-bit ADD ≈ 0.5 pJ
Register file access

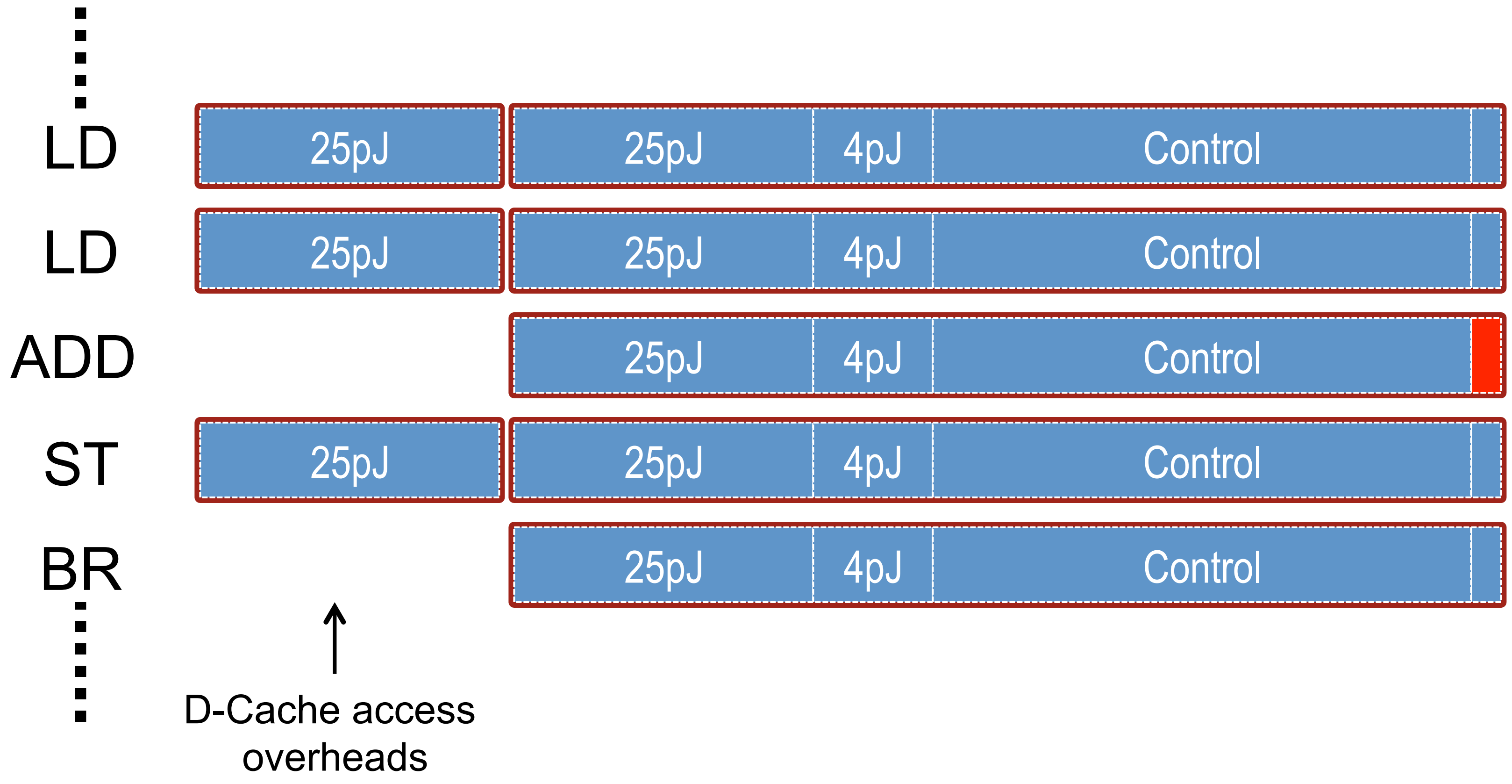
* Assuming a typical 32-bit embedded RISC in 45mn @ 0.9V technology

Other instructions overhead



* Assuming a typical 32-bit embedded RISC in 45mn @ 0.9V technology

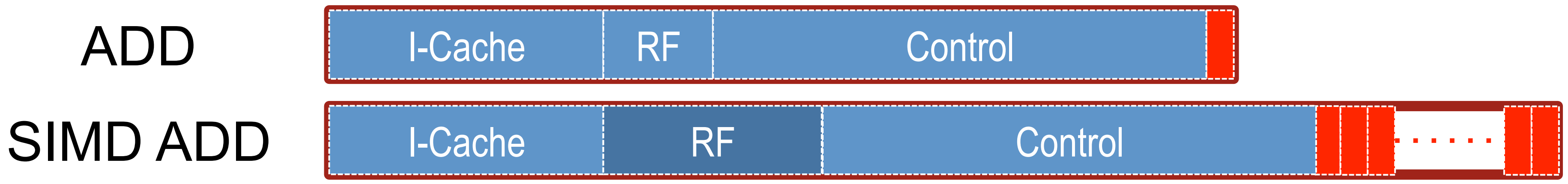
D-Cache accesses overhead



* Assuming a typical 32-bit embedded RISC in 45mn @ 0.9V technology

SIMD machines give some improvement

- **SIMD units amortize overhead <u>and</u> improve performance**



- **Achieves 10X better energy and performance AND is programmable**

- **Can we do 100X and keep it programmable?**

Energy efficiency in a programmable environment

Each memory and instruction fetch must be amortized by hundreds of operations

What we want to see

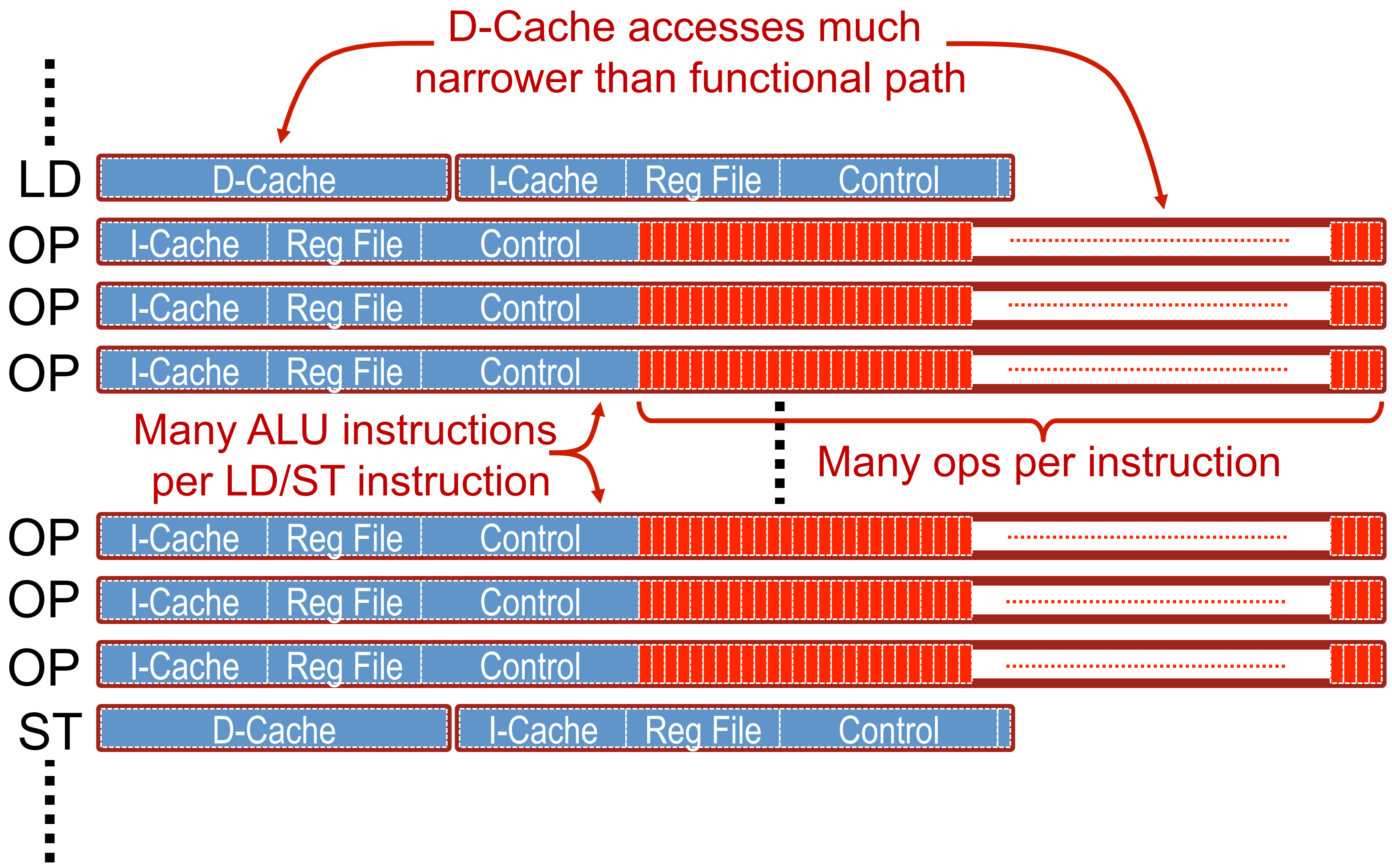


Image processing looks like convolution

- **Most of the computation is performed over (overlapping) stencils**
- **Looks like convolution:** $(Img \otimes f)_{[n,m]} = \sum_{l=-c}^{c} \sum_{k=-c}^{c} Img_{[k,l]} \cdot f_{[n-k,m-l]}$

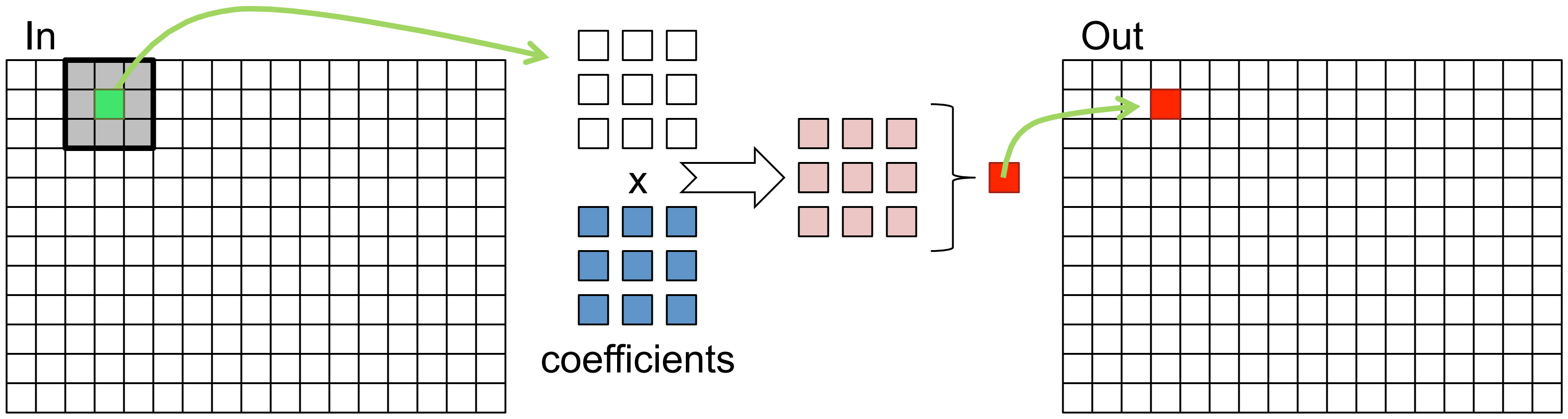


Image processing looks like convolution

- Most of the computation is performed over (overlapping) stencils
- Looks like convolution: $(Img \otimes f)_{[n,m]} = \sum_{l=-c}^{c} \sum_{k=-c}^{c} Img_{[k,l]} \cdot f_{[n-k,m-l]}$

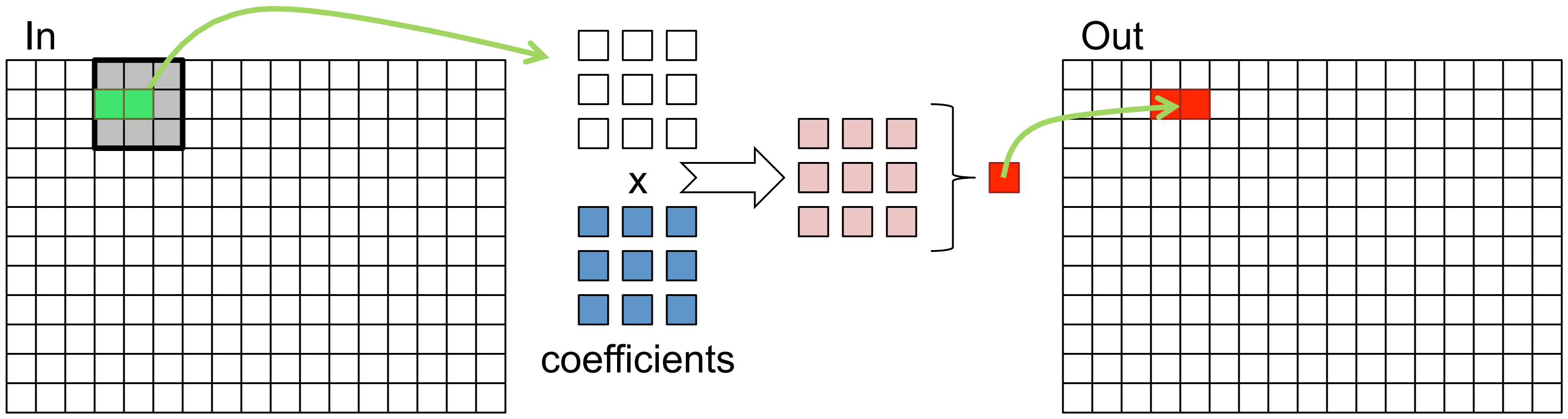
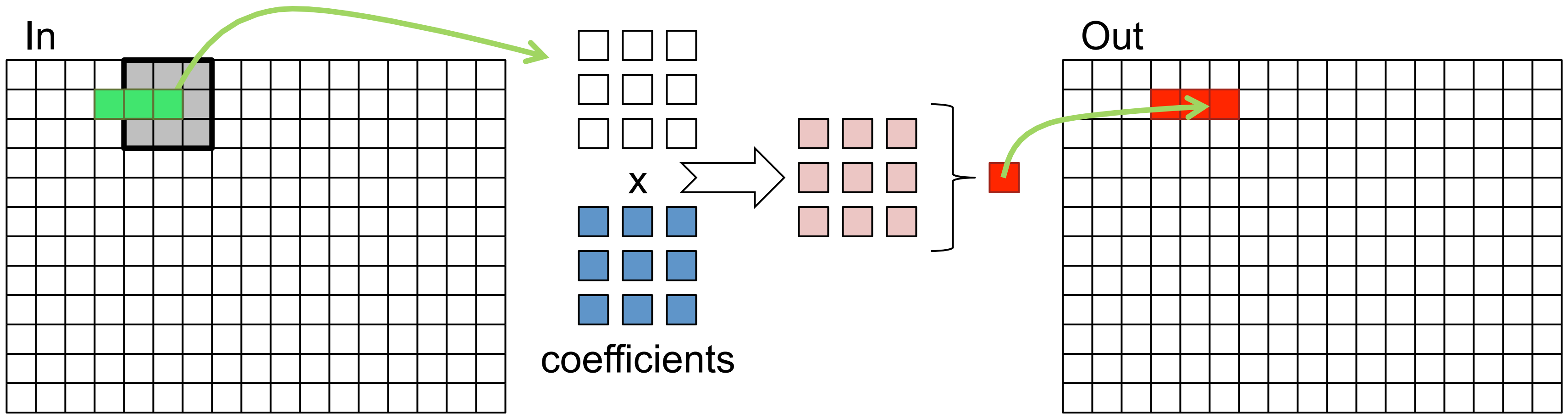


Image processing looks like convolution

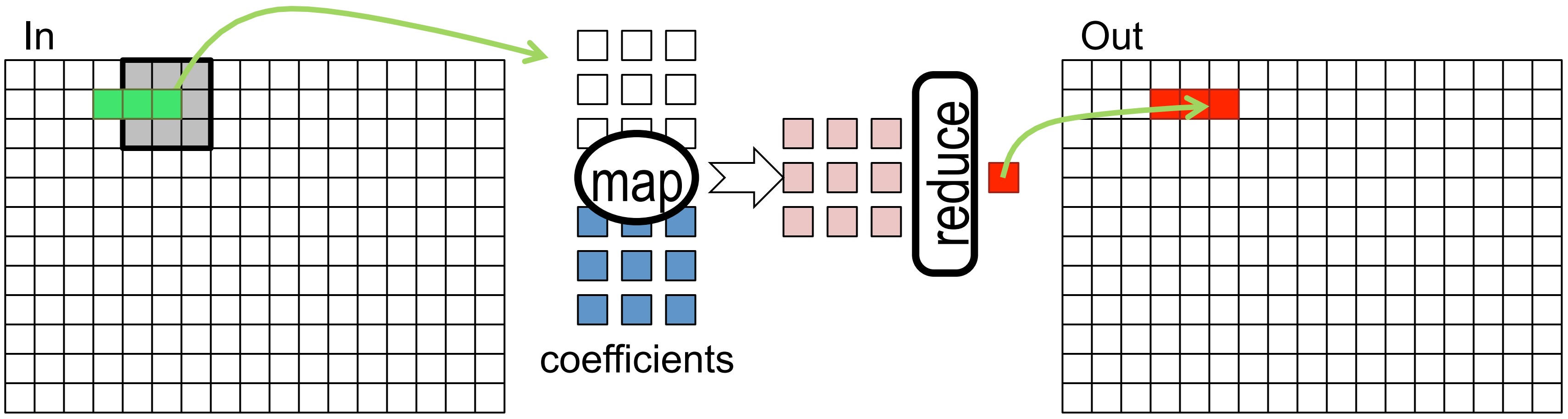
- Most of the computation is performed over (overlapping) stencils
- Looks like convolution: $(Img \otimes f)_{[n,m]} = \sum_{l=-c}^{c} \sum_{k=-c}^{c} Img_{[k,l]} \cdot f_{[n-k,m-l]}$



It does not have to be convolution

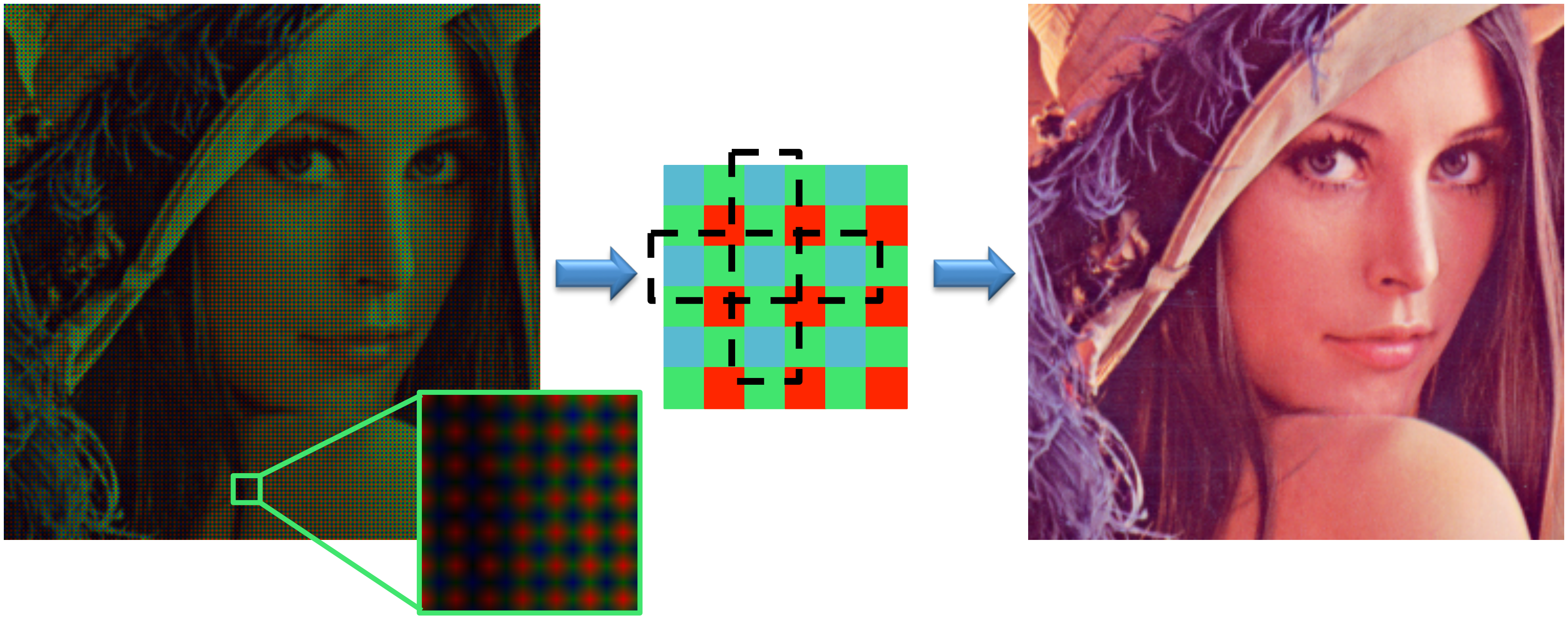
- **It only looks like convolution:**

$$\left(Img \overset{CE}{\otimes} f\right)_{[n,m]} = Reduce_{l=-c}^{c}\left[Reduce_{k=-c}^{c}\left[map\left(Img_{[k,l]}, f_{[n-k,m-l]}\right)\right]\right]$$



Let’s look at some convolution-like workloads

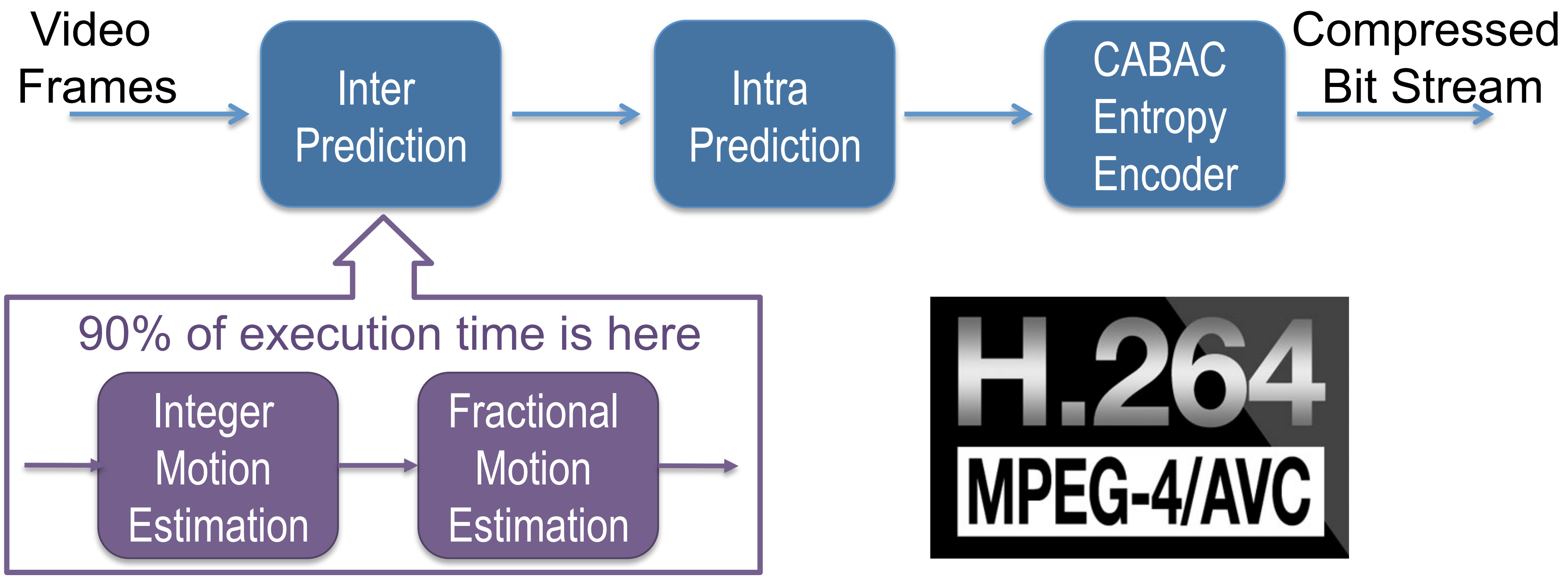
- **De-mosaic:**
  - Adaptive color plane interpolation (ACPI)*: image gradients followed by a three-tap filter in the direction of smallest gradient.



* Y. Cheng et. al. An adaptive color plane interpolation method based on edge detection. Journal of Electronics (China), 2007.

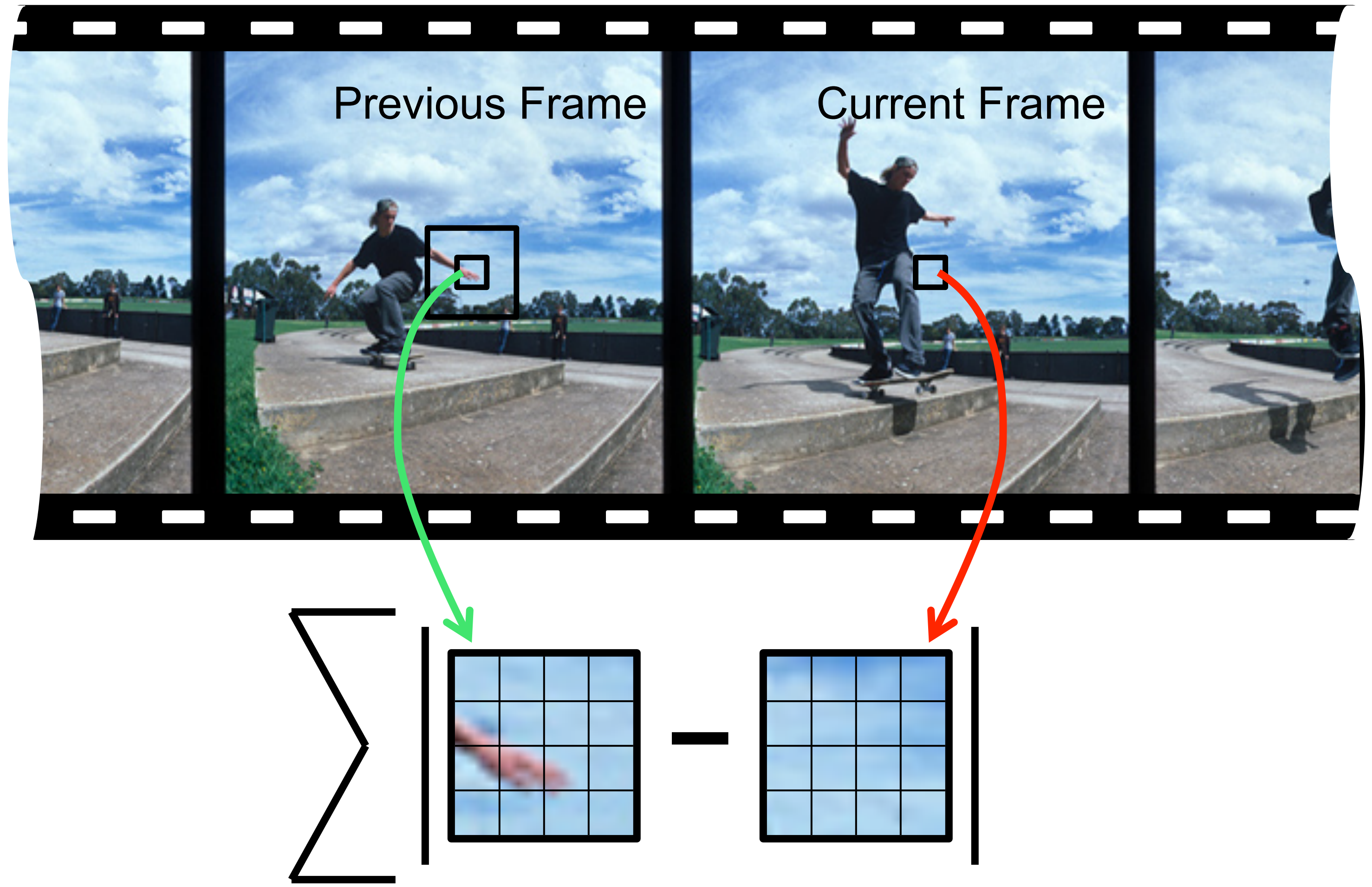
Let's look at more convolution-like workloads

- **H.264 (high definition) video encoder:**
  - IME: 2D-Sum of absolute differences
  - FME: Half pixel interpolation, quarter pixel interpolation, 2D SAD



The main computation behind H.264

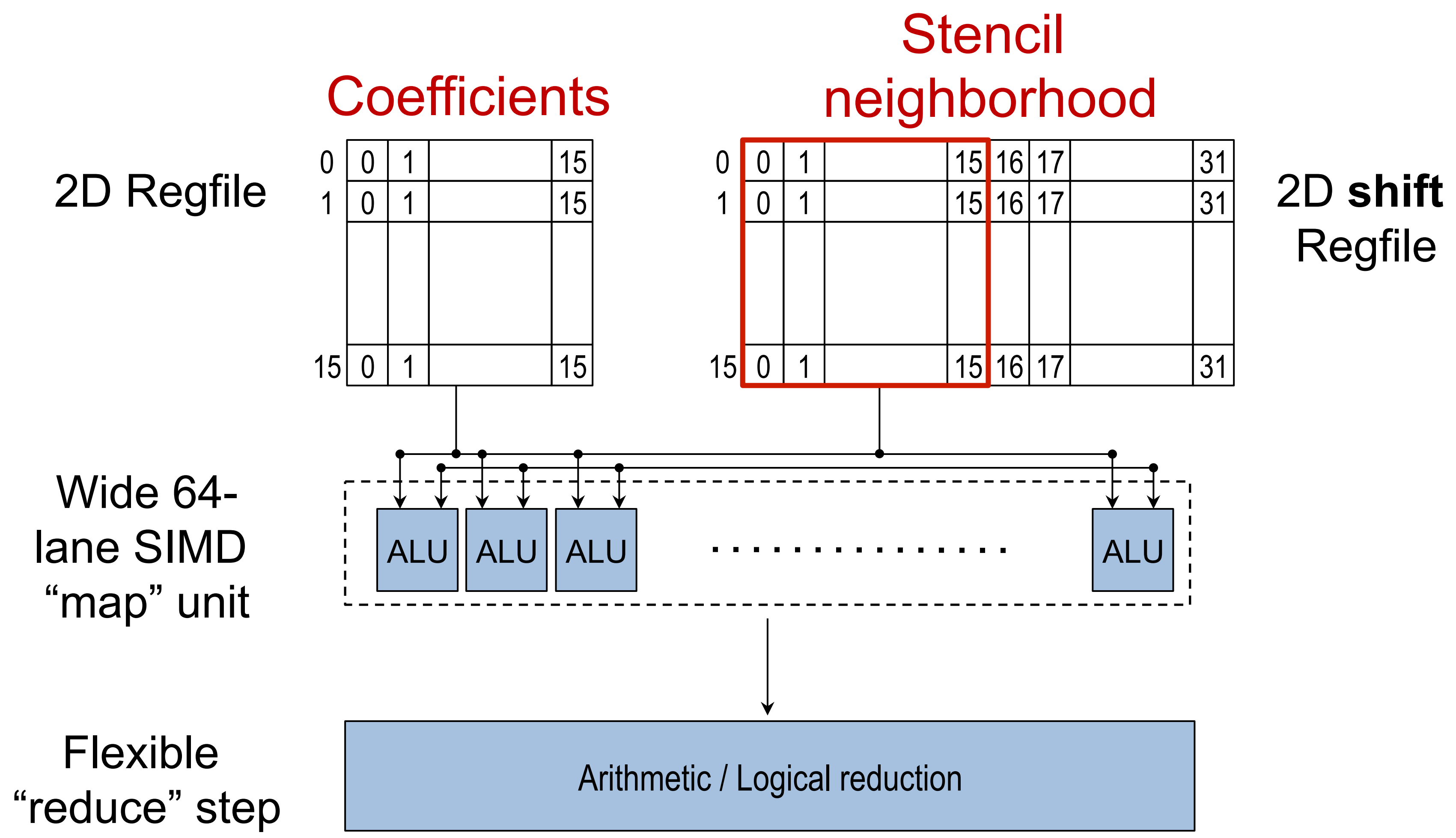
- Trying to find best match for a stencil within a small neighborhood



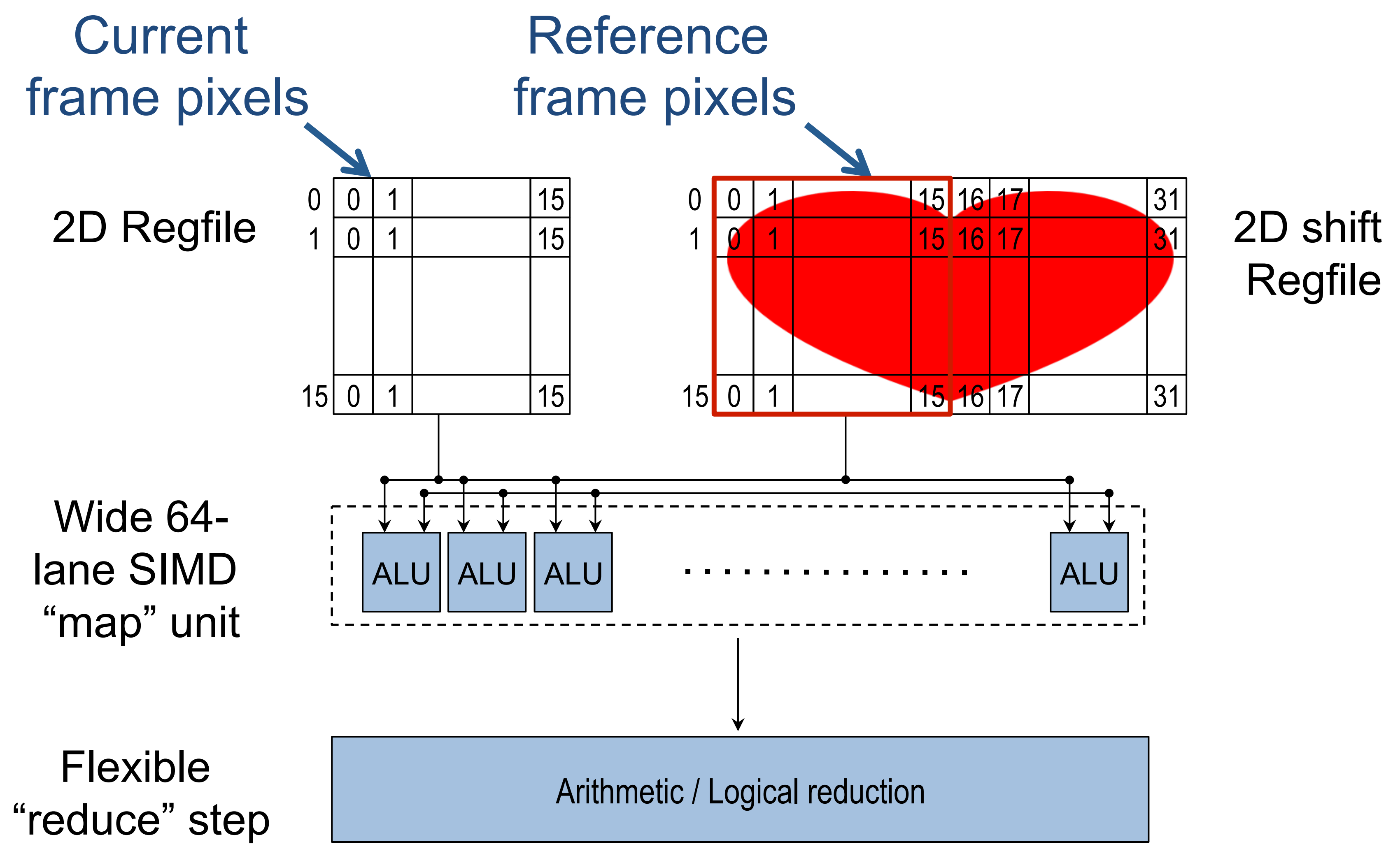
The convolution engine must support different ops

	Map	Reduce	Stencil Size	Data Flow
IME SAD	Abs Diff	Add	4x4	2D Convolution
FMW ½ pixel up-sample	Multiply	Add	6	1D Horizontal & vertical conv.
FME ¼ pixel up-sample	Average	None	--	2D Matrix operation
SIFT Gaussian blur	Multiply	Add	9, 13, 15	1D Horizontal & vertical conv.
SIFT DoG	Subtract	None	--	2D Matrix operation
SIFT Extreme	Compare	Logic AND	3	1D Horizontal & vertical conv.
Demosaic interpolation	Multiply	Complex	3	1D Horizontal & vertical conv.

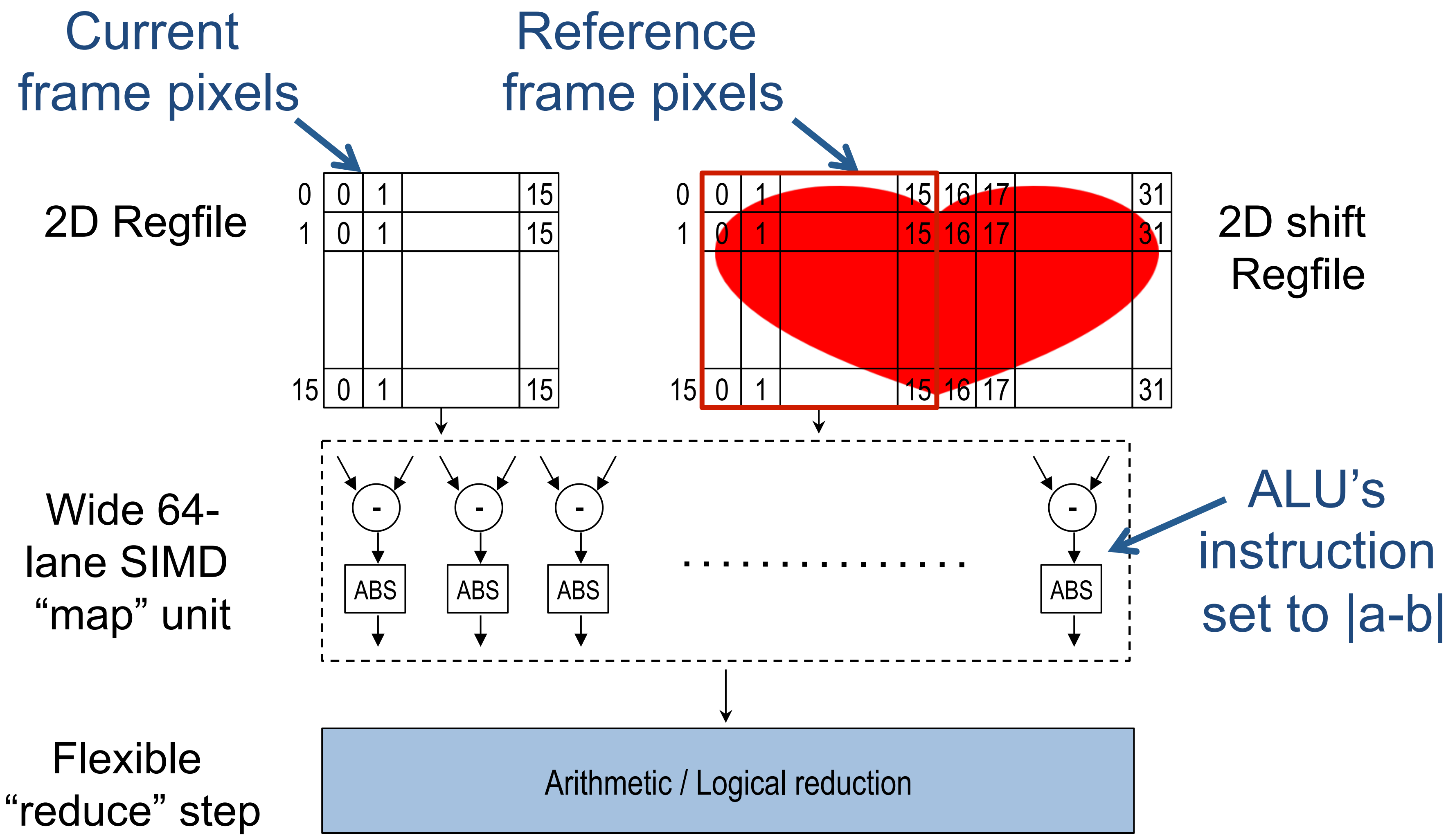
Convolution Engine: An architecture for convolution-like kernels



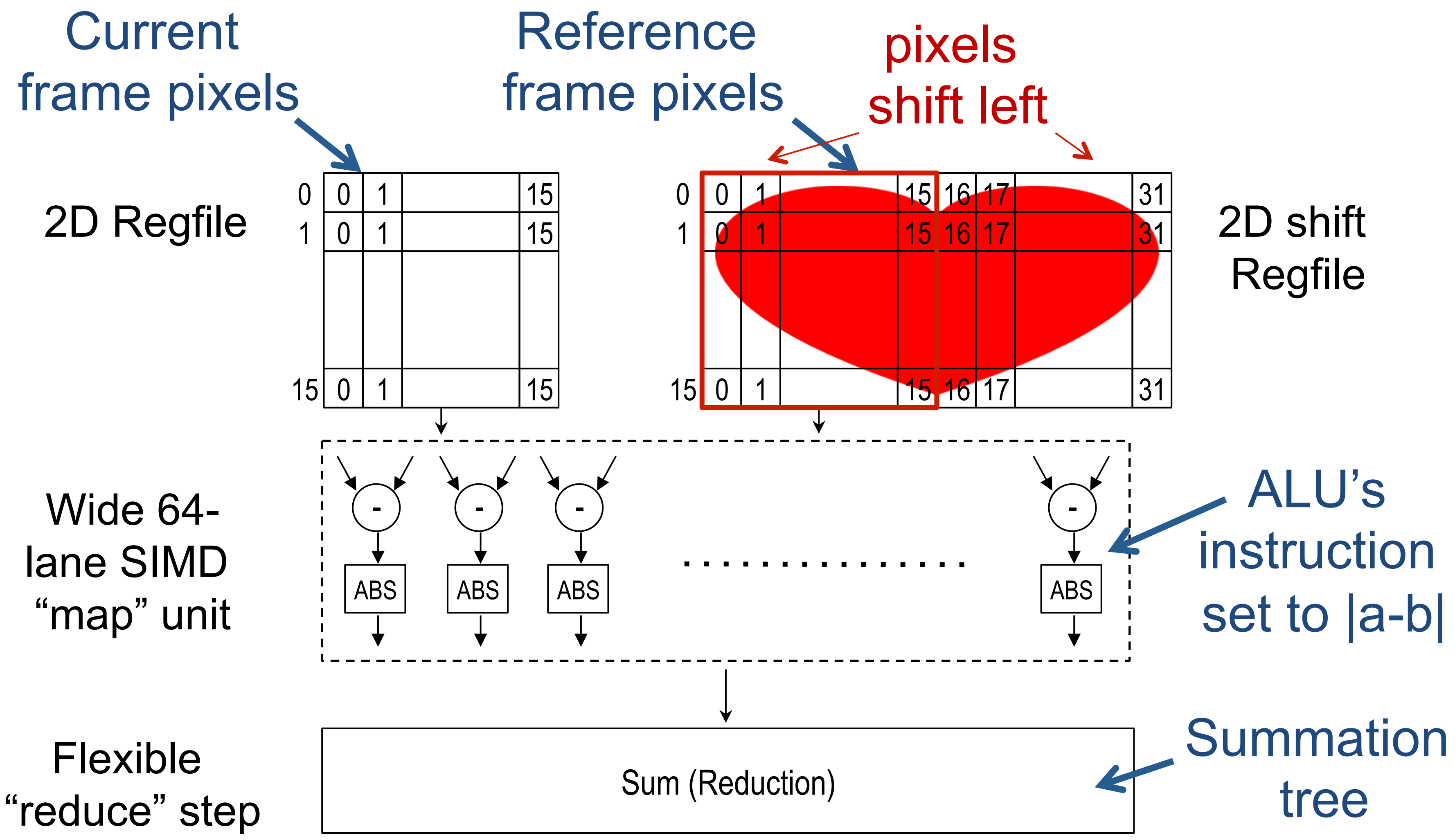
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



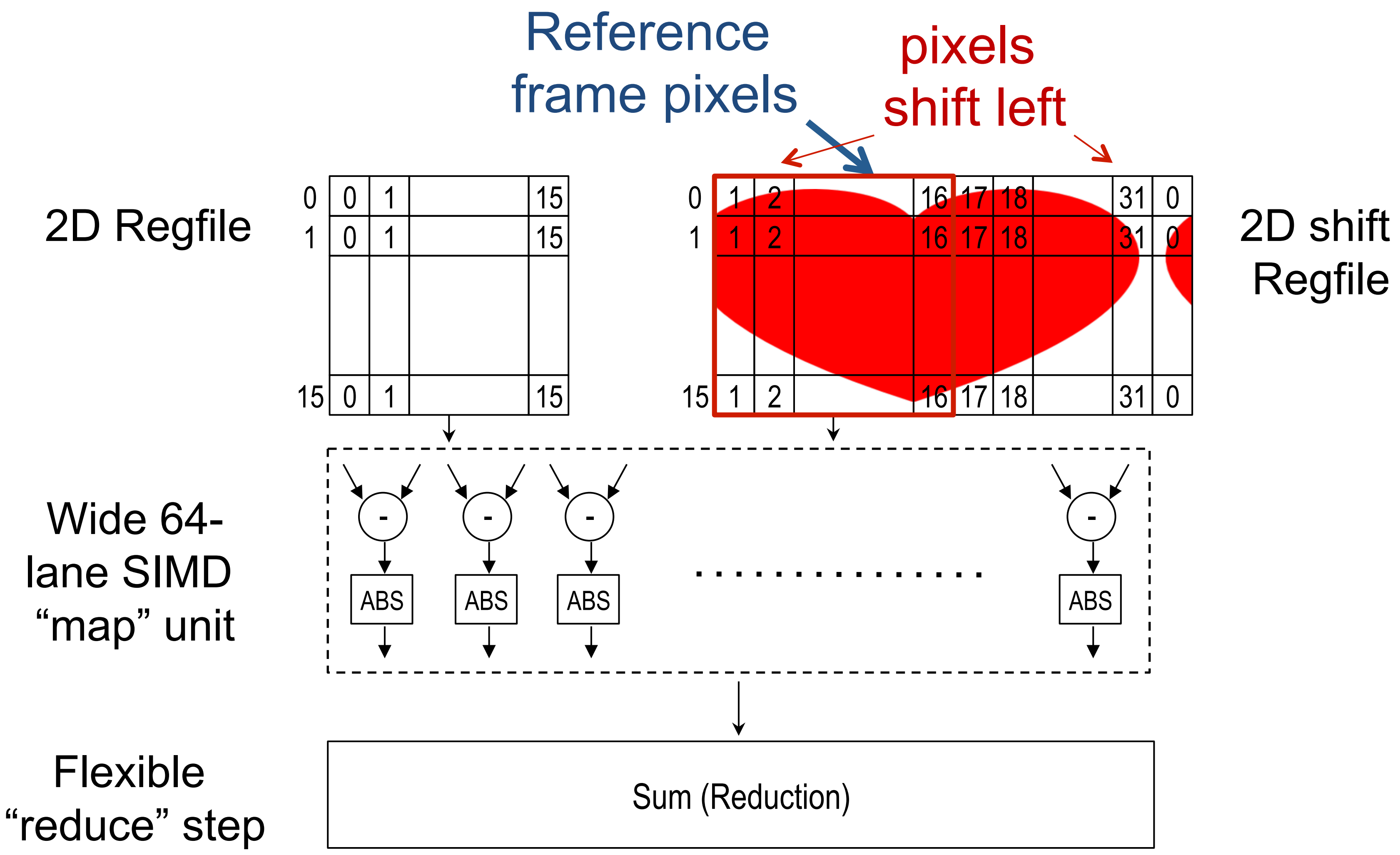
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



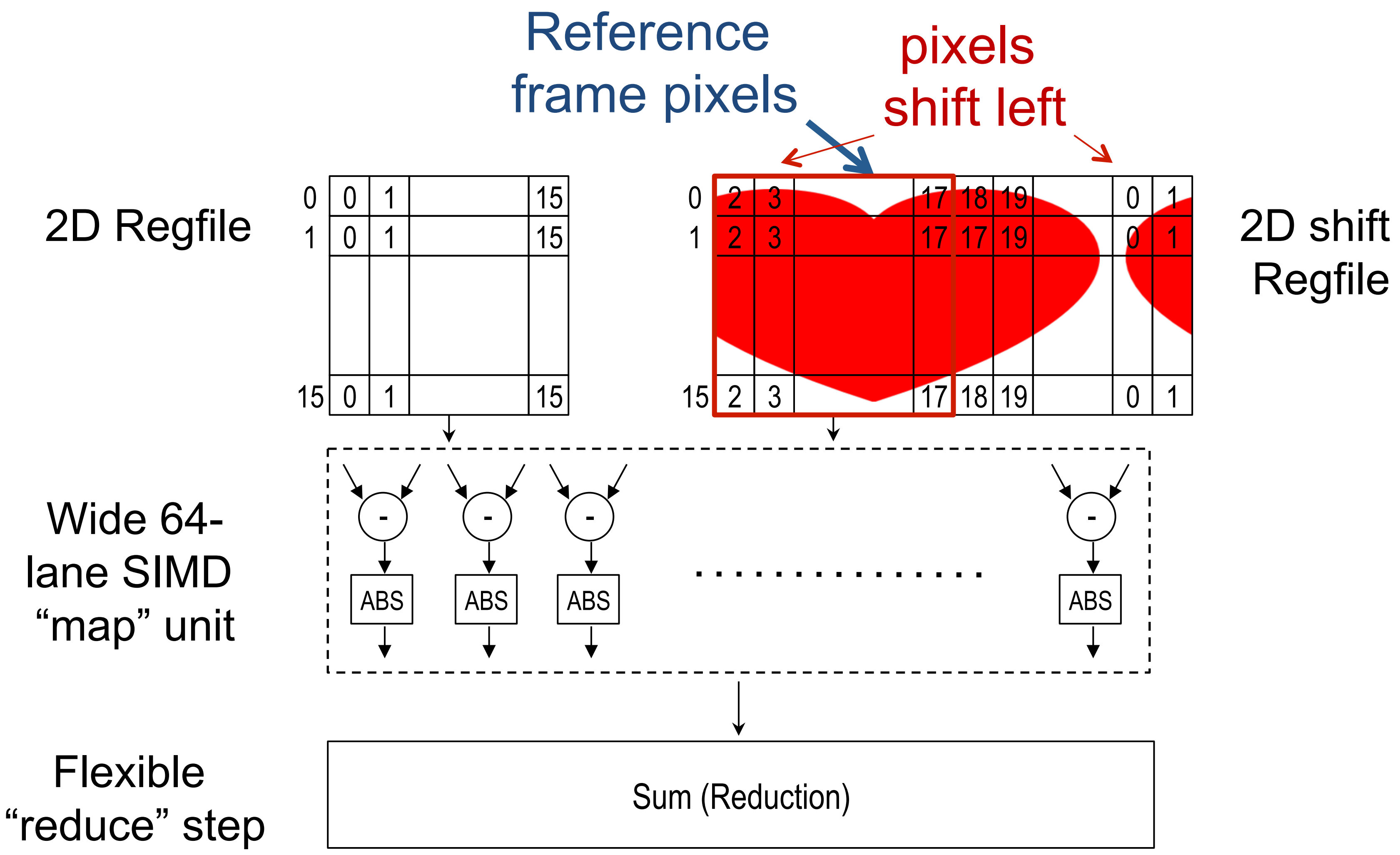
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



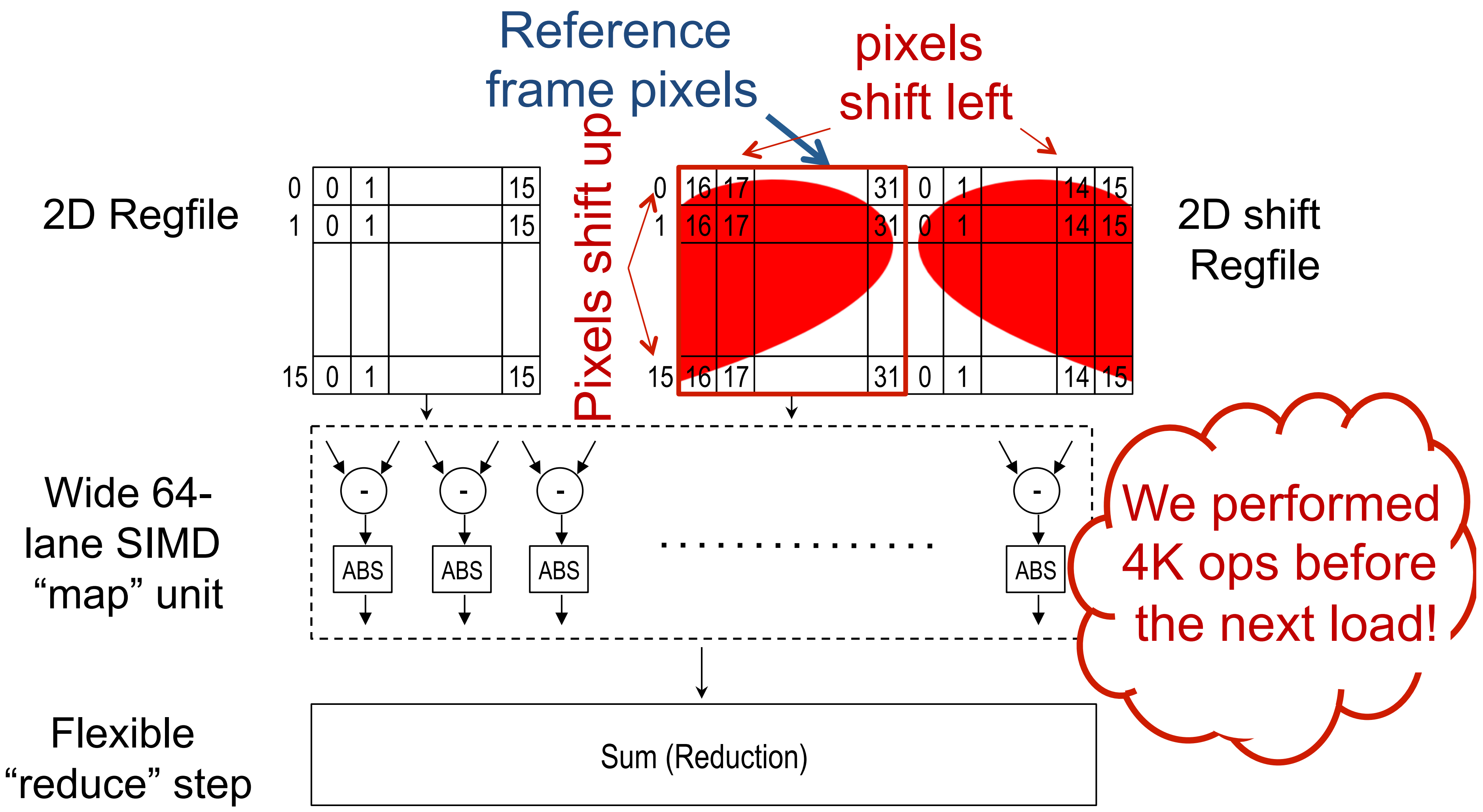
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



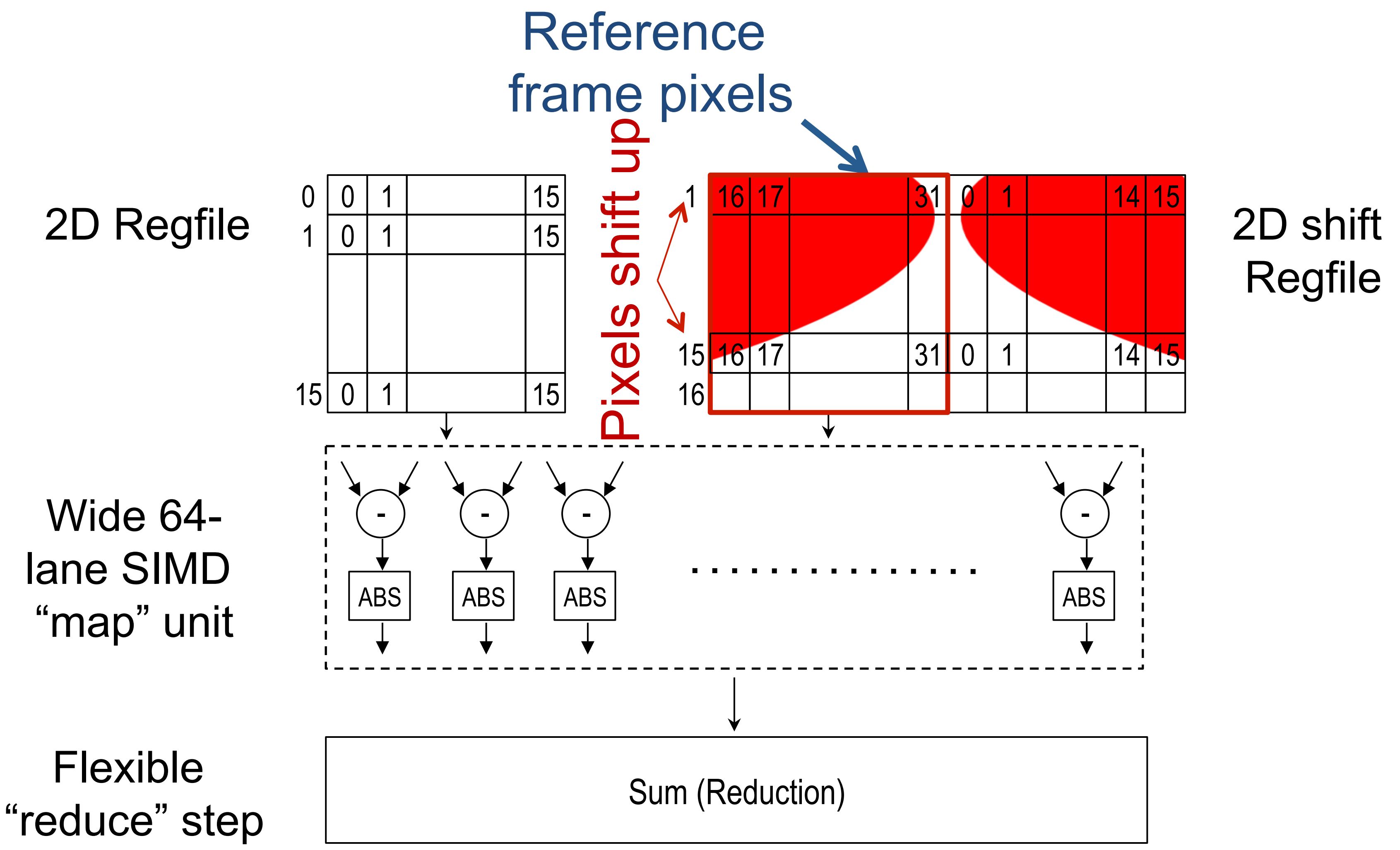
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



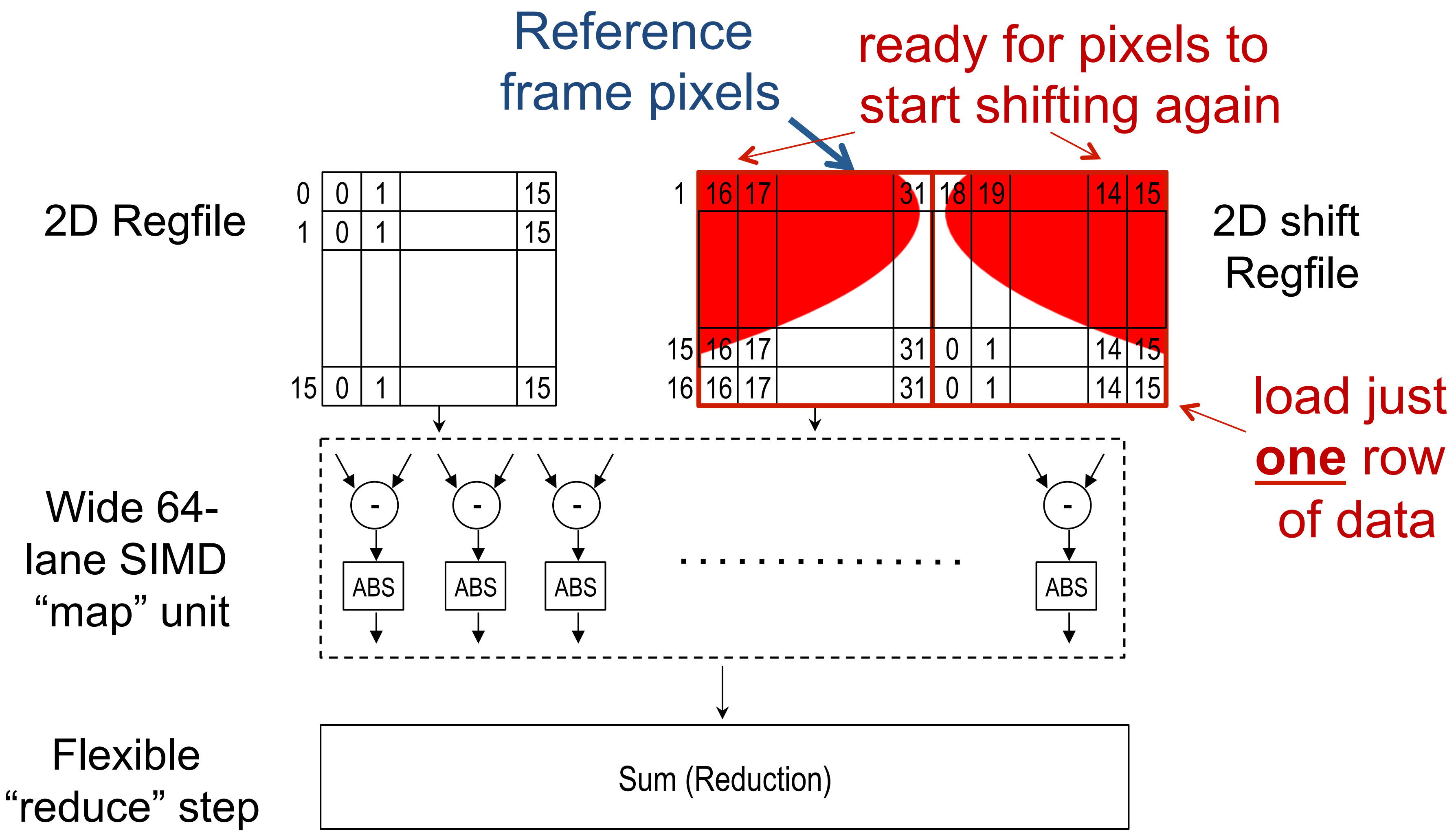
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



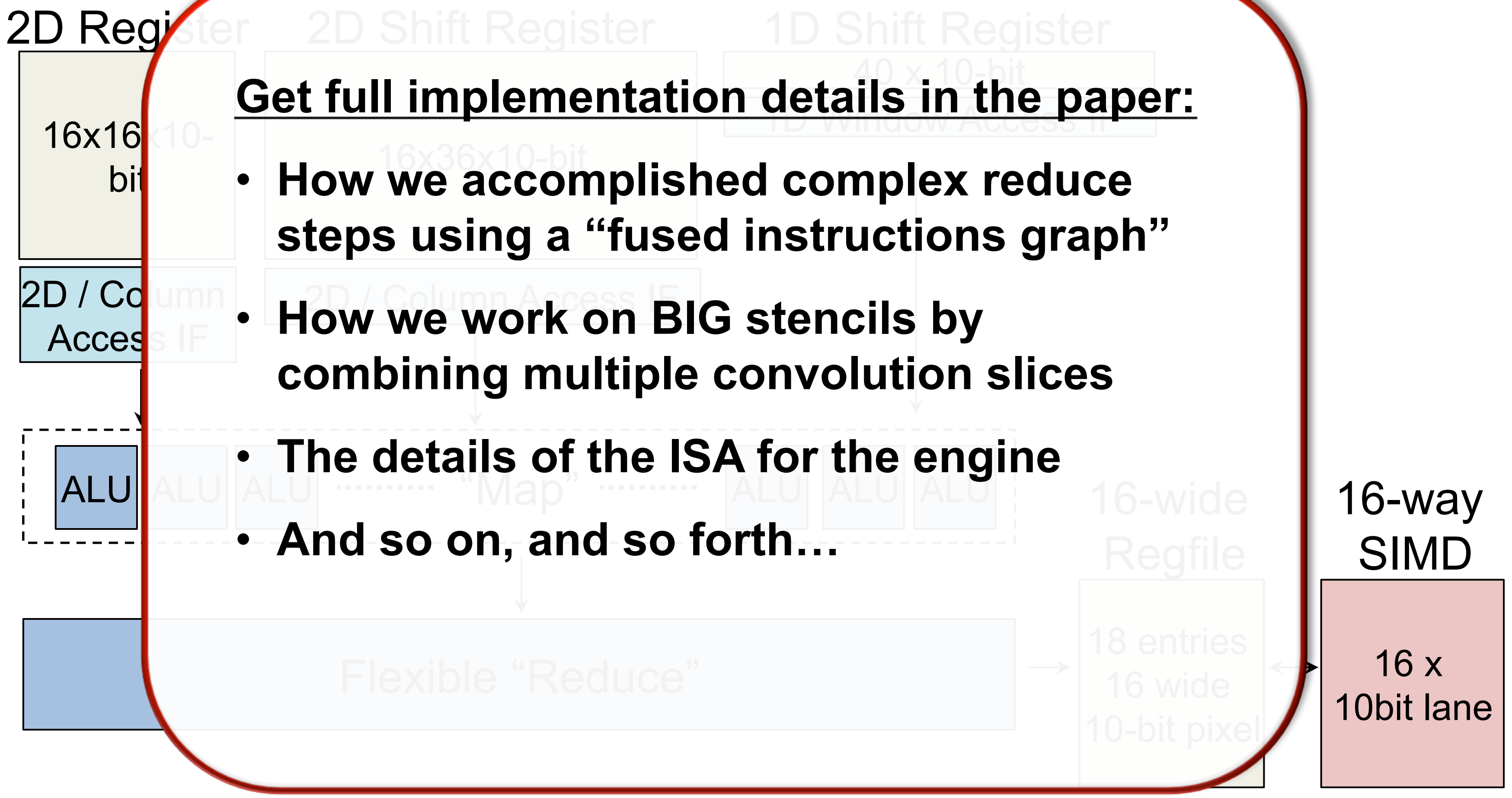
Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



Example from H.264's motion estimation: Mapping Sum of Absolute Differences (SAD)



Our Convolution Engine as implemented



Get full implementation details in the paper:

- **How we accomplished complex reduce steps using a "fused instructions graph"**
- **How we work on BIG stencils by combining multiple convolution slices**
- **The details of the ISA for the engine**
- **And so on, and so forth…**

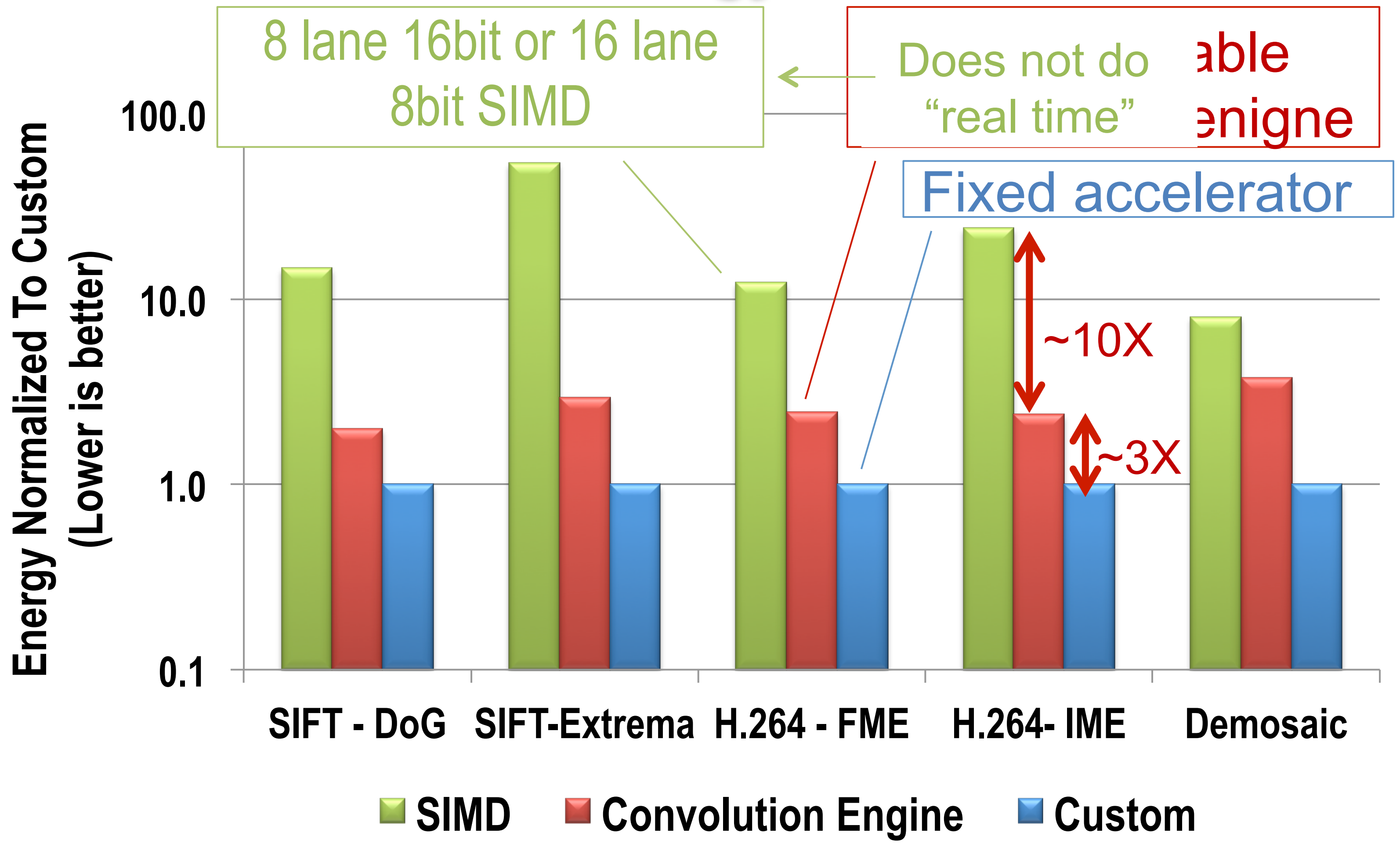
Result #1:
CE is user programmable in C!

```
SET_CE_OPS (CE_ABSDIFF, CE_ADD);  // Set map & reduce funcs to abs-diff and add
SET_CE_OPSIZE(16);                // Set convolution size 16x16

// Load the 16x16 current macroblock into 2D coefficients register
for (int i=0; i<16; i++ {
   LD_COEFF_REG_128(curMBPtr, i);   // Load 16 pixels to row i of coefficient register
   curMBPtr += imgWidth;
}
// Load the first 32x16 current reference window into 2D input register
for (int i=0; i<16; i++ {
   LD_2D_REG_128(refPtr, 0, SHIFT_ENABLED);        // Load & shift-up 16 pixels to 2D Reg
   LD_2D_REG_128(refPtr+16, 1, SHIFT_DISABLED);  // Load next 16 pixels
   refPtr += imgWidth;
}
// Calculate one row of SAD output
for (int x = 0; x < 16; x++) {
   CONVOLVE_2D(ROTATE_LEFT, x);   // 16x16 2D convolution step and shift left
}
// Store 16 output SAD results
ST_OUT_REG_128(outPtr);
```

IME code snippet

Result #2: CE is 100X more energy efficient than RISC



- All variations were implemented as Tensilica extensions (TIE)

Conclusions

- **There are classes of computations for which we can build efficient hardware, and we typically build them in ASIC**
- **Image and video are ubiquitous and represents one of those classes as their computation is convolution-like**
- **But when we restrict the domain, two orders of magnitude better programmable engines are also possible!**
- **Flexible specialized engines are <u>not</u> an oxymoron**
  - Flexible convolution engine improves power & performance by ~100X
  - Only 2-3X worse off than a dedicated (not flexible) accelerator

THANK YOU FOR LISTENING!

BACKUP SLIDES…

Energy dissipation in RISC machines

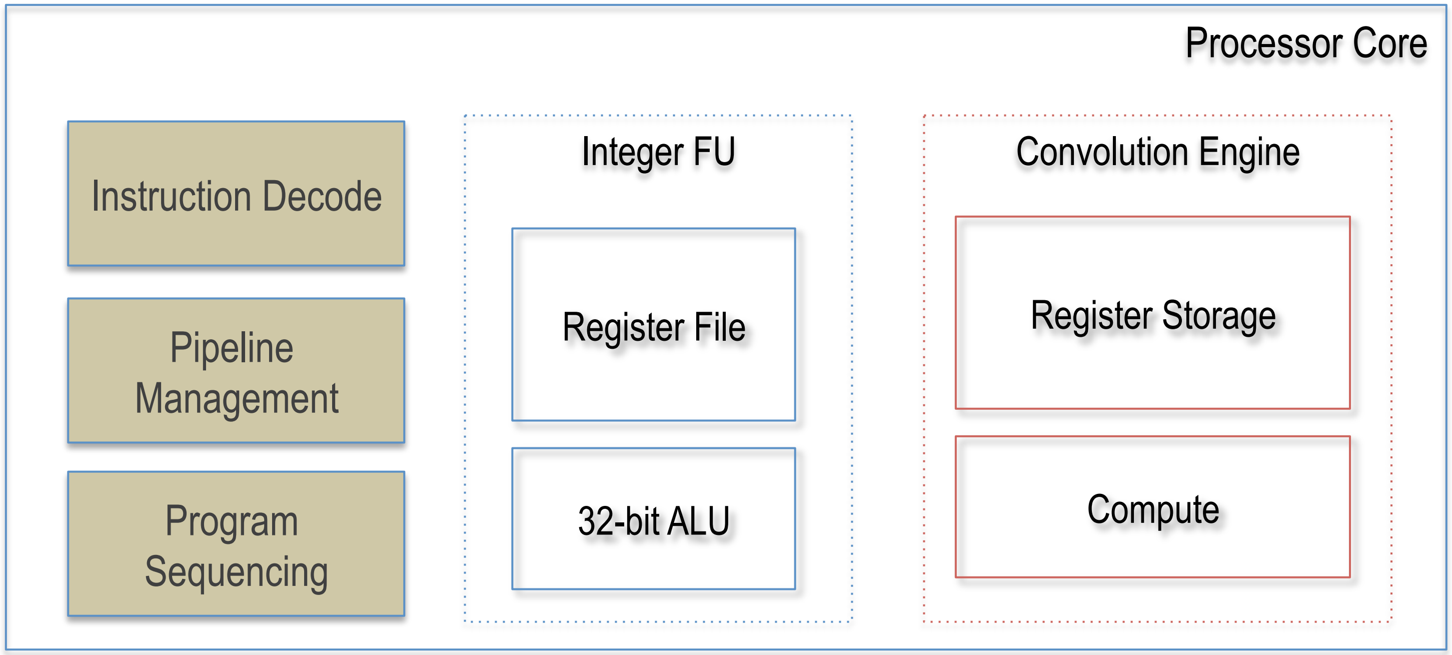
- **Let’s do a breakdown of a typical RISC Instruction**

- **Keep in mind (at 45nm):**
  - Addition is ~0.1pJ for 8bits (ASIC) or ~0.5pJ for 32bits (RISC)
  - Multiplication is ~0.2pJ for 8bits (ASIC) or ~3.1pJ for 32bits (RISC)
  - But a single RISC instruction is 70pJ

- **Need to see where the overhead is, and how we can mitigate it**

Processor Integration

- **Specialized Functional Unit**
  - Adds about 30 instructions to the processor ISA
  - The execution flow is controlled by the processor

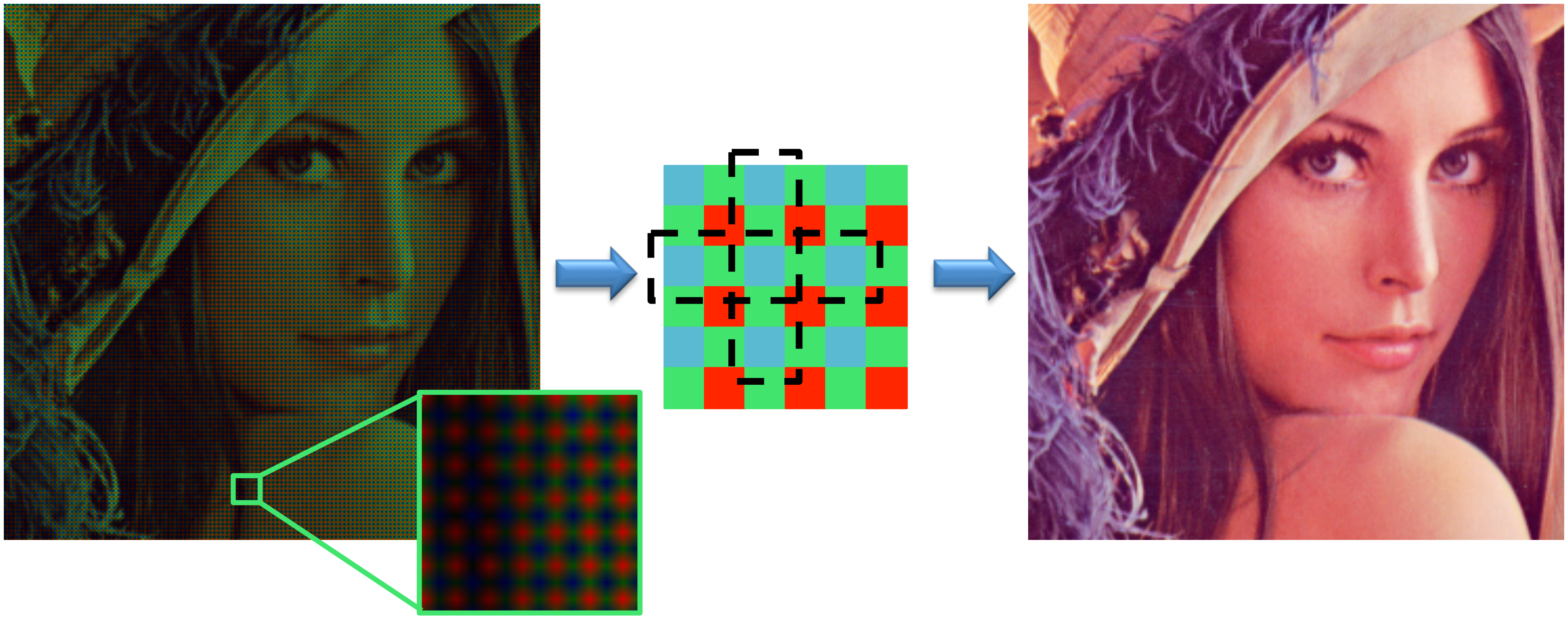


Evaluating the Convolution Engine

- **Applications**
- **SIFT Feature extraction**
  - Often a basic step for computational photography algorithms
    - HDR Imaging
    - Panorama stitching
    - Smart zoom / Super resolution
    - Multi-frame noise reduction
    - Synthetic aperture
    - Augmented reality
    - Flash – No-Flash photography
    - Video de-shake
    - ……
- **H.264 encoder**
  - Every video system has one

Let's look at some of the workloads

- **De-mosaic:**
  - Adaptive color plane interpolation (ACPI)*: image gradients followed by a three-tap filter in the direction of smallest gradient.



* Y. Cheng et. al. An adaptive color plane interpolation method based on edge detection. Journal of Electronics (China), 2007.