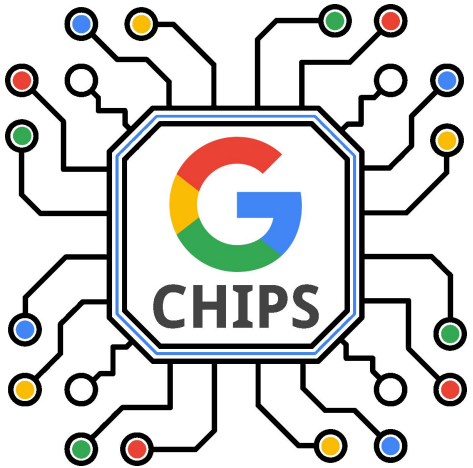
Google

Pixel Visual Core: Google's Fully Programmable Image, Vision, and AI Processor For Mobile Devices

Jason Redgrave, Albert Meixner, Nathan Goulding-Hotta, Artem Vasilyev, and Ofer Shacham



gChips's Silicon Implementation

Motivation: The Vision

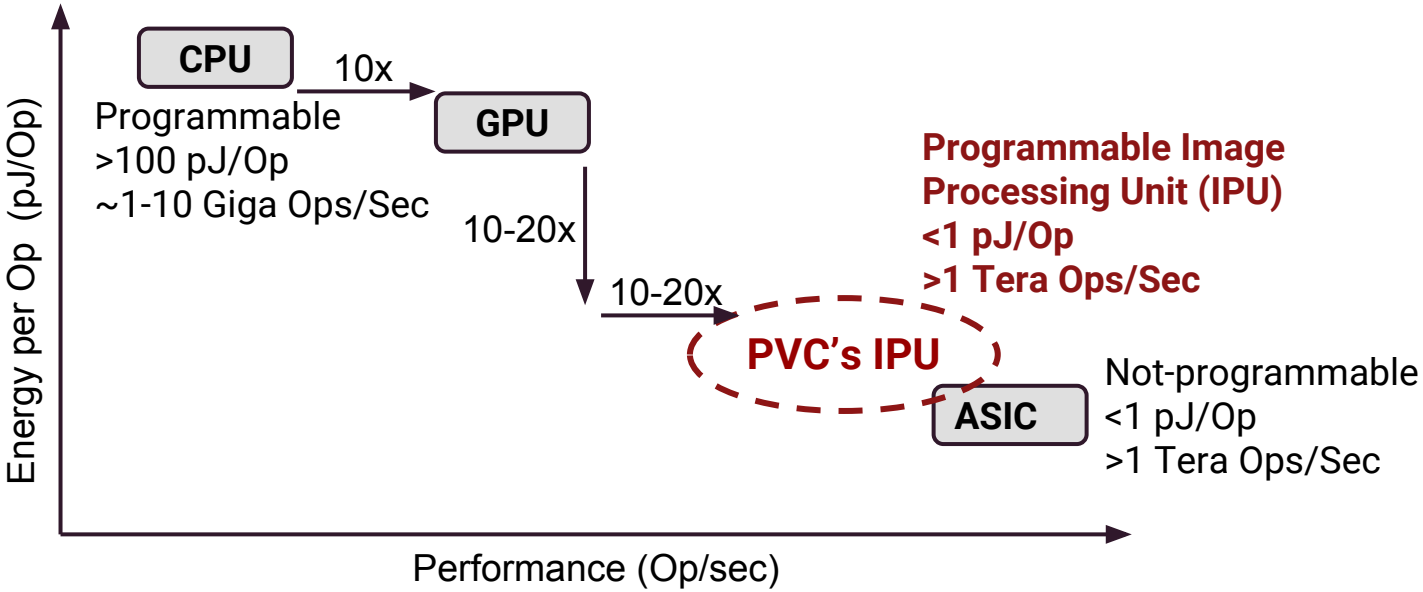
No HDR+

HDR+

Software Motivation

- Imaging, Vision, and AI are fast evolving fields
- Productizing new algorithms is difficult
  - ASIC design adds long delay into Research→Product cycle
  - CPU efficiency limits complexity in mobile space
  - Reliance on external vendors limits changes to the stack
- Software wants control, flexibility, and efficiency

Hardware Motivation



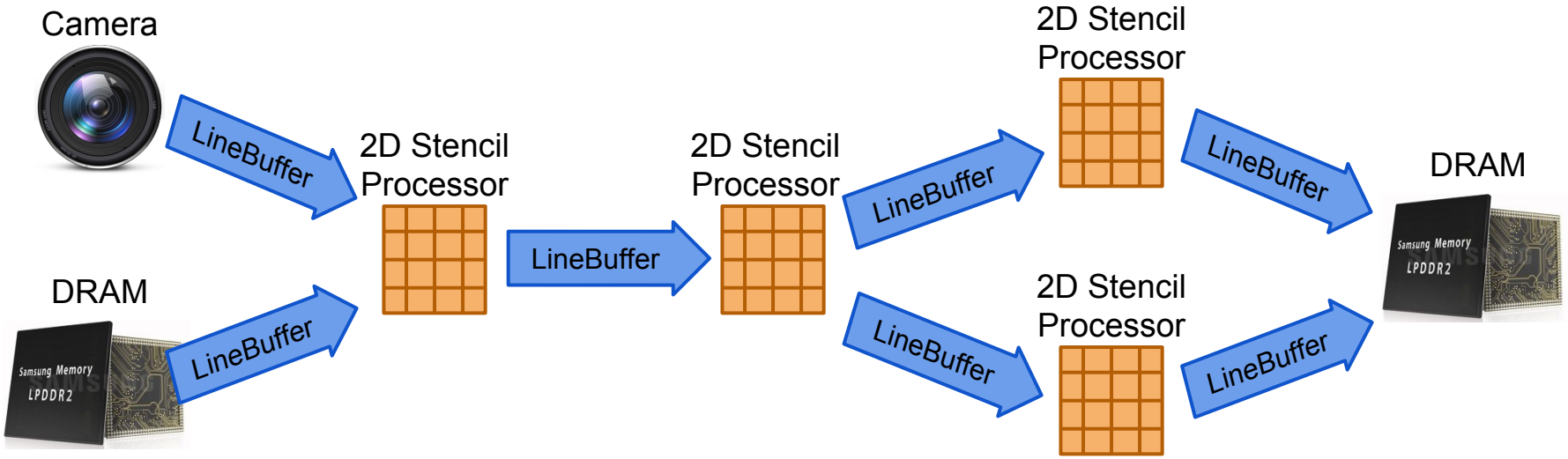
Domain-specific Software

- High-level programming model is Halide
  - Domain-specific language for Image Processing
- IPU supports a subset of Halide language
  - No floating point
  - Limits on available memory access patterns
- Halide backend generates kernels and all API calls
  - Proprietary API for resource allocation and execution control

Compilation

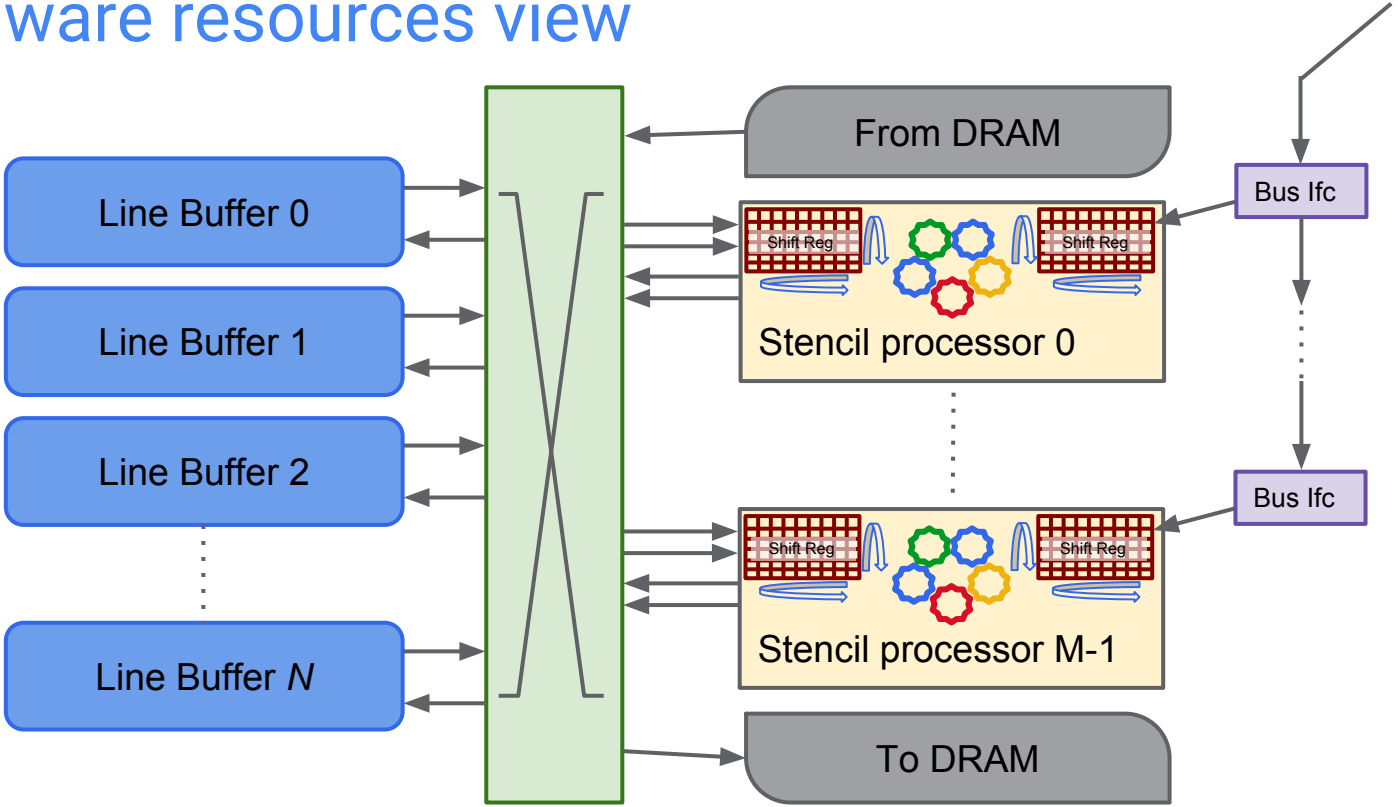
- Halide backend generates high-level, virtual ISA (vISA)
  - RISC ISA with streaming-friendly memory model
  - Cross-generational & Architecture-independent
- Final compilation into physical ISA (pISA)
  - Compilation can be offline or on-device
  - Generation-specific VLIW ISA
  - All memory movements are explicit (no caches)

Conceptual View of Hardware: DAG of Kernels



- Many cores, each fully programmable
- Configurable DAG topology

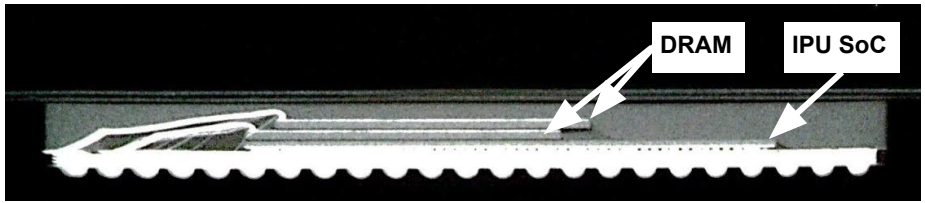
Hardware resources view

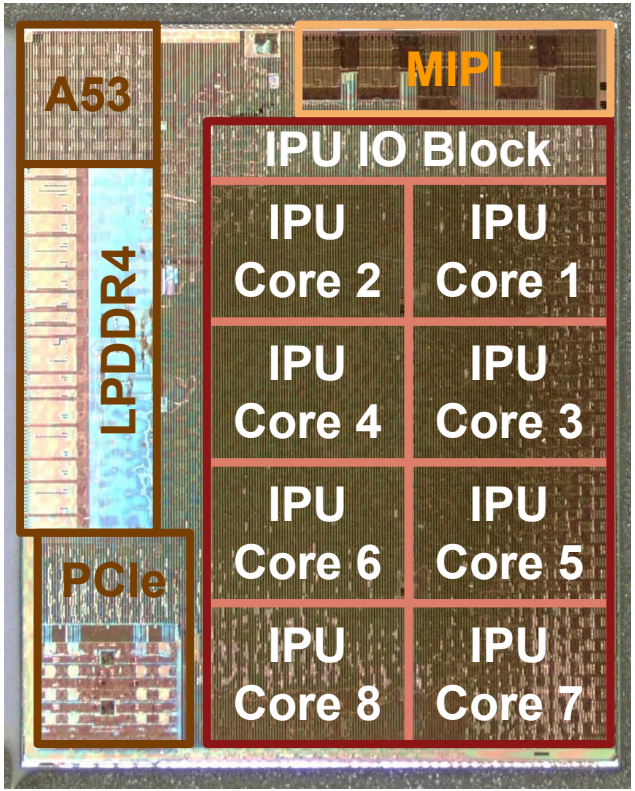


Pixel Visual Core Architecture

- A53
- LPDDR4
- MIPI
- PCIe
- IPU

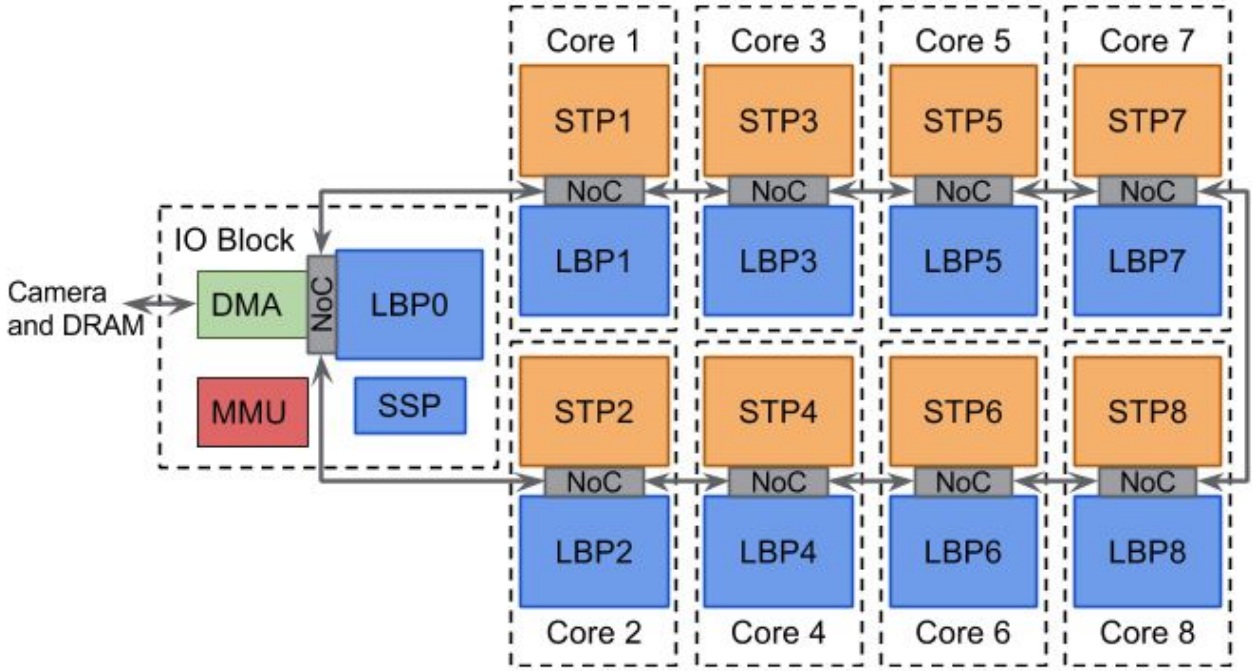
Chip Specs:
TSMC 28nm
6.0 x 7.2 mm
426 MHz
512 MB DRAM
Power <4500 mW





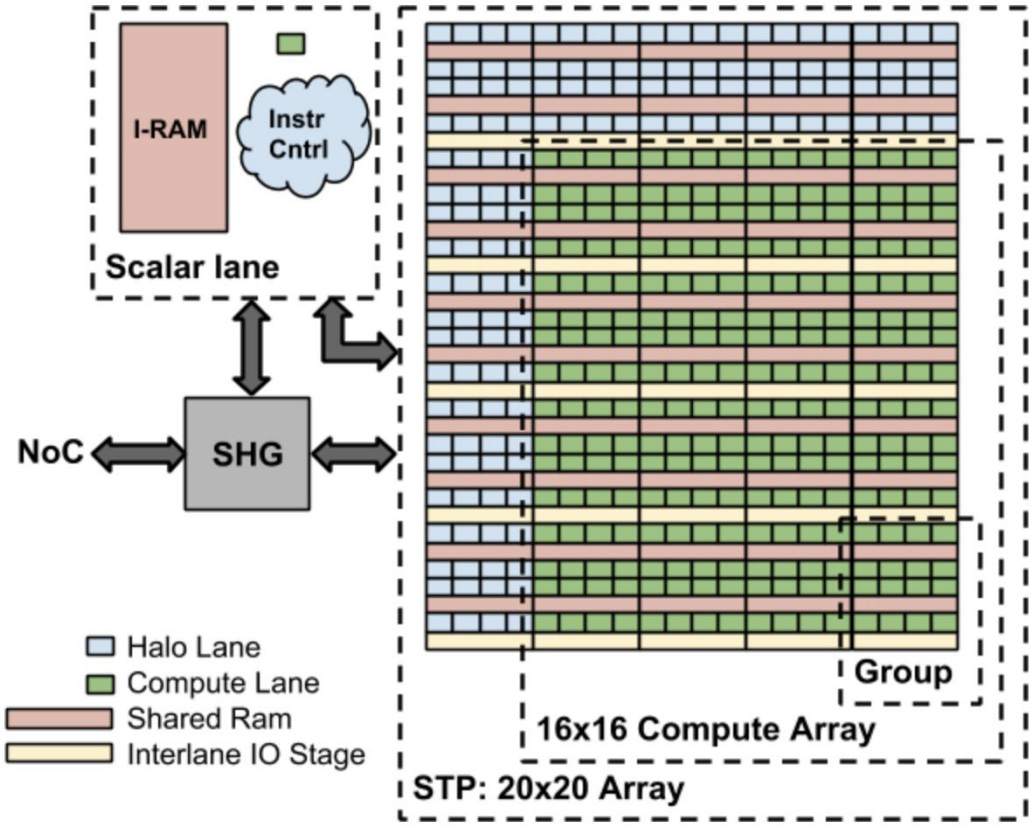
IPU Architecture

- 8x Cores
  - STP
  - LBP
- I/O
  - DMA
  - MMU
  - SSP (Buffer)
- Ring NoC



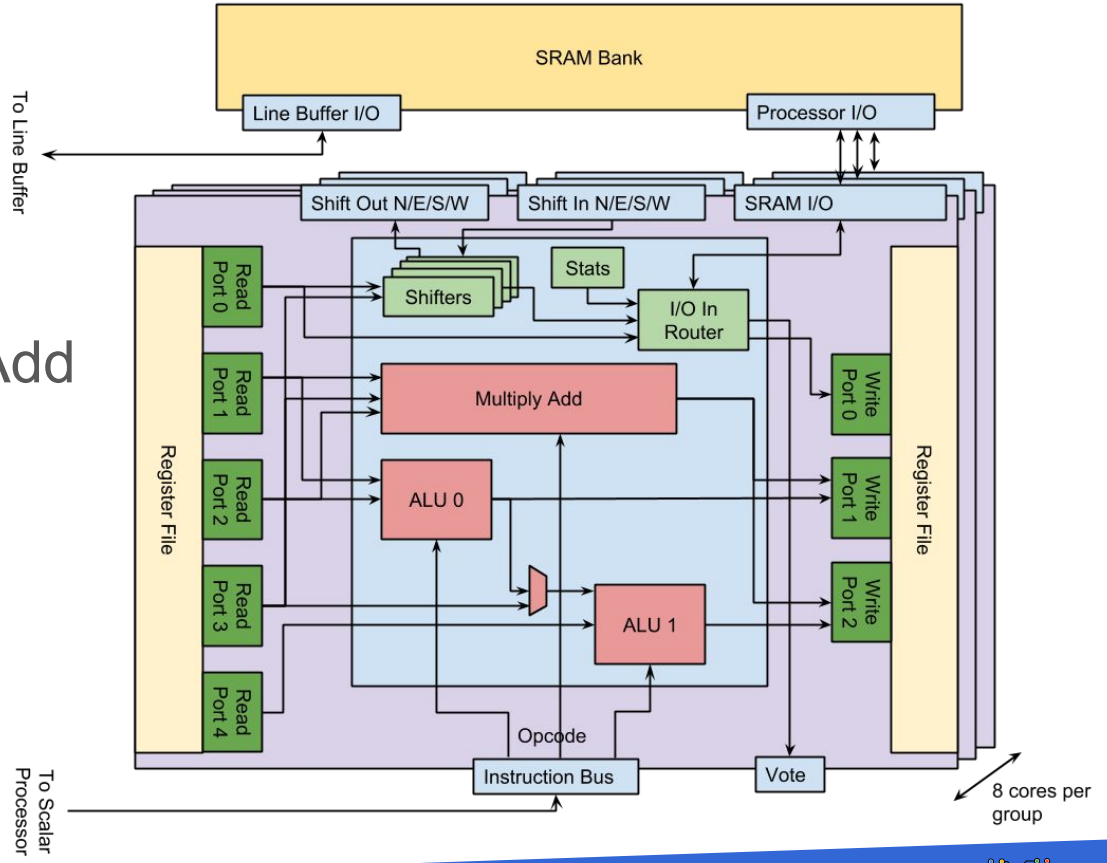
Stencil Processor (STP)

- Scalar lane
  - Instruction RAM
- SHG (Load/Store)
- 2D Array
  - 256 Compute lanes
  - 144 Halo lanes
  - Shared RAMs
  - Shift network



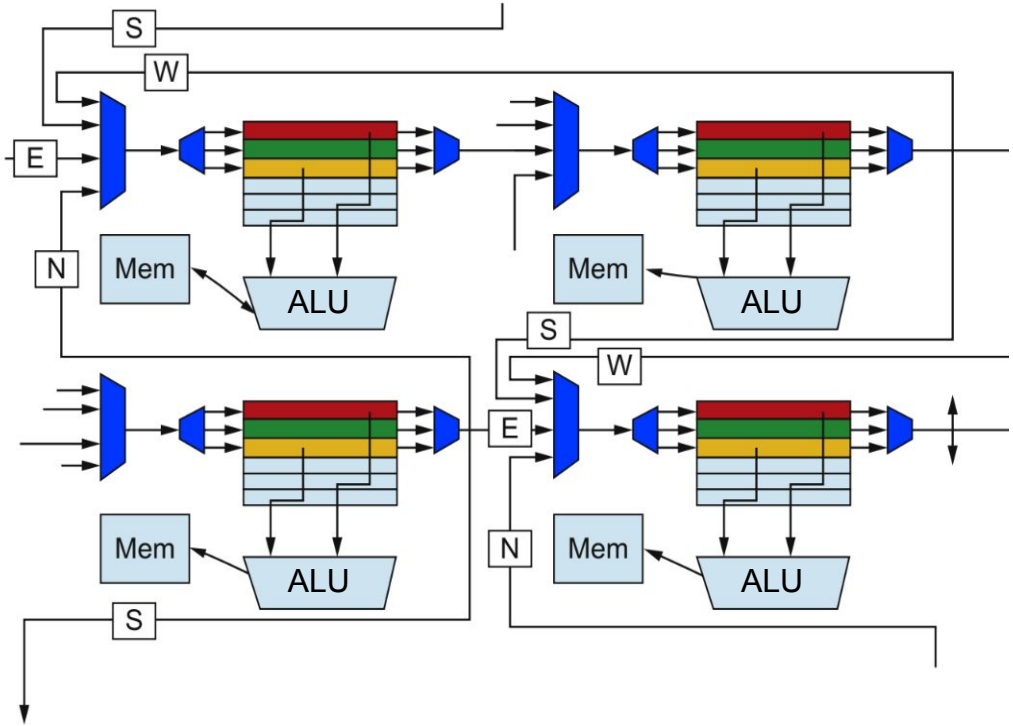
Compute Lane

- Single cycle!
- Dual 16b ALU
- 16b x 16b Multiply-Add
- Memory
  - Register File
  - Shared RAM (L0)



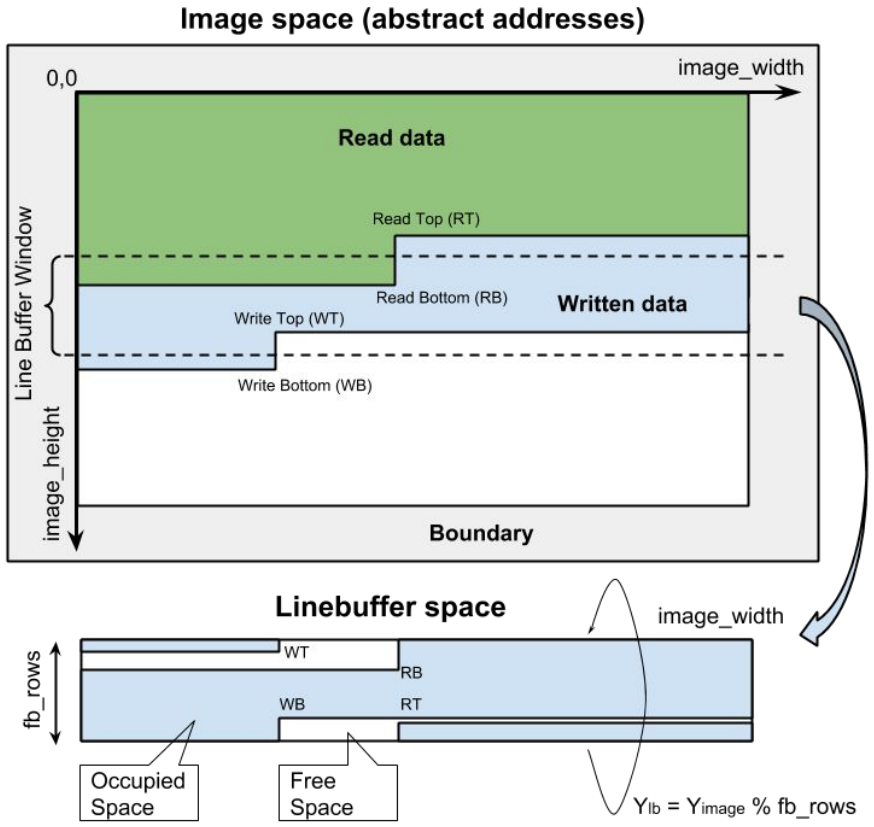
Read Neighbor Network

- Output pixel depends on neighboring inputs
- One vertical or horizontal shift per cycle
- Shifts are circular (toroidal)
- 1/2/3/4 hops in each direction



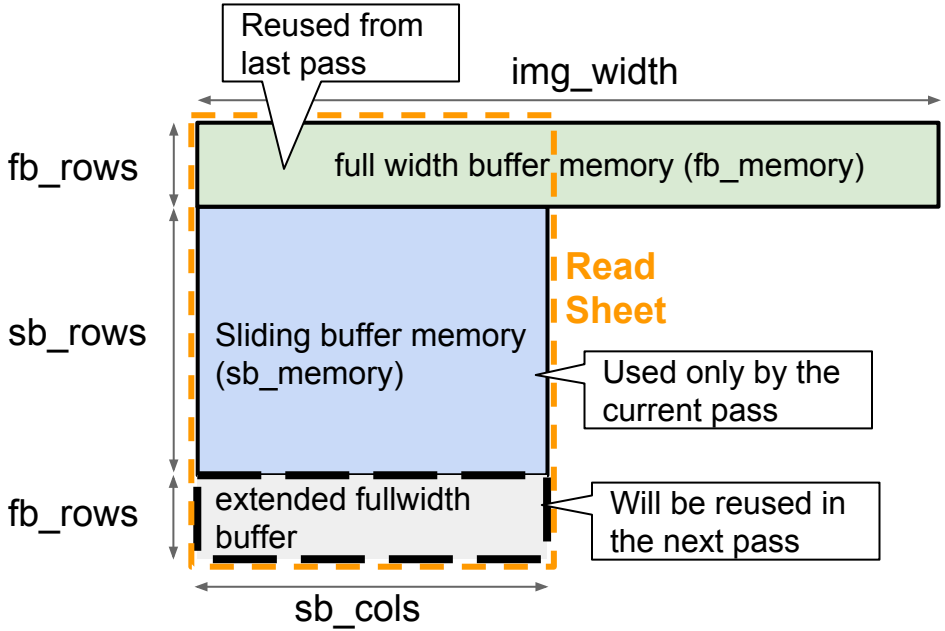
Line Buffer Pool (LBP)

- Data storage
- Synchronization
- 2D Line Buffer abstraction (2D FIFO)

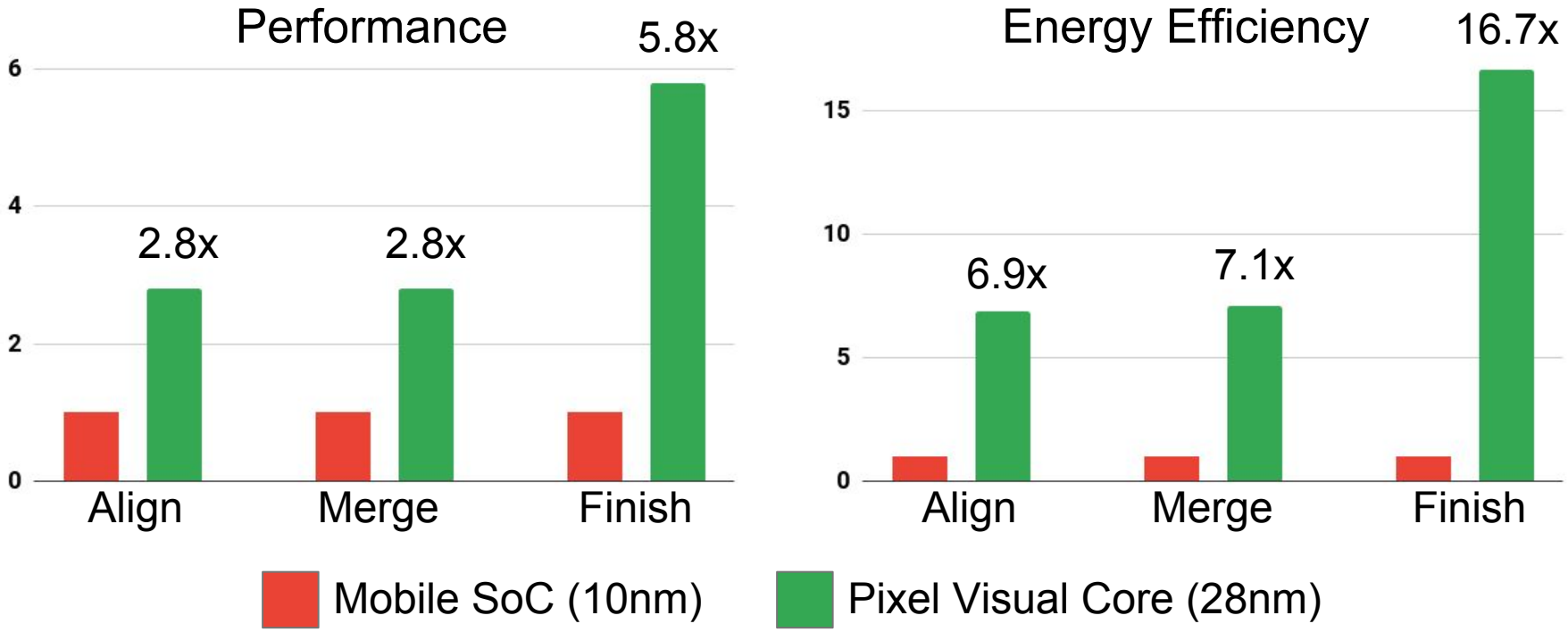


Virtually Tall Line Buffer

- Memory saving over full buffer
- Elasticity set by SB width
- FB height set by kernel reuse



Results



7-16x more energy-efficient despite 3-generation process gap!

T.J. Alumbaugh, Ke Bai, Fabrizio Basso, Trevor Bunker, Dinesh Chandrasekaran, Ed Chang, Robert Chapman, Arun Chauhan, Albert Chen, Chien-Yu Chen, Matt Cockrell, Jamison Collins, Joe Dao, Neeti Desai, Dusty DeWeese, Andrea Di Blas, Daniel Finchelstein, Arnd Geis, Filippo Gioachin, Richard Goe, Nathan Goulding-Hotta, Ben Gribstad, Cheng Gu, Penny Gur, Nico Hailey, Ashok Halambi, Adam Hampson, Mahdi Hamzeh, Vince Harron, Sam Hasinoff, Zhijun He, Viresh Hingarh, Sean Howarth, Richard Hsu, John Keen, Asif Khan, Ji Yun Kim, Timothy Knight, Victor Leung, Marc Levoy, Yuan Lin, Joshua Litt, Chenjie Luo, Bill Mark, Albert Meixner, Jon Michelson, Rob Mohr, Michael Moreno, Ben Mossawir, Rolf Mueller, Sean O'Boyle, Sarang Padalkar, Hyunchul Park, David Patterson, Alexander Perez, Karthika Periyathambi, Bob Pflederer, Theodore Popp, Todd Poynor, Jing Pu, Shahriar Rabii, Ramesh Ramarao, Jason Redgrave, Masumi Reynders, Shac Ron, Sabarish Sankaranarayanan, Vaidyanathan Seetharaman, Ofer Shacham, Dillon Sharlet, Sean Silva, Saravana Soundararajan, Don Stark, Eino-Ville Talvala, Pei Zhao Tang, David Tolnay, Michelle Tomasko, Artem Vasilyev, Jaakko Ventela, Nate Voorhies, David Warren, Kevin Watts, Huachang Xu, Catherine Yu, Rong Zhou, Qiuling Zhu

Questions?