OpenVX Graph Acceleration

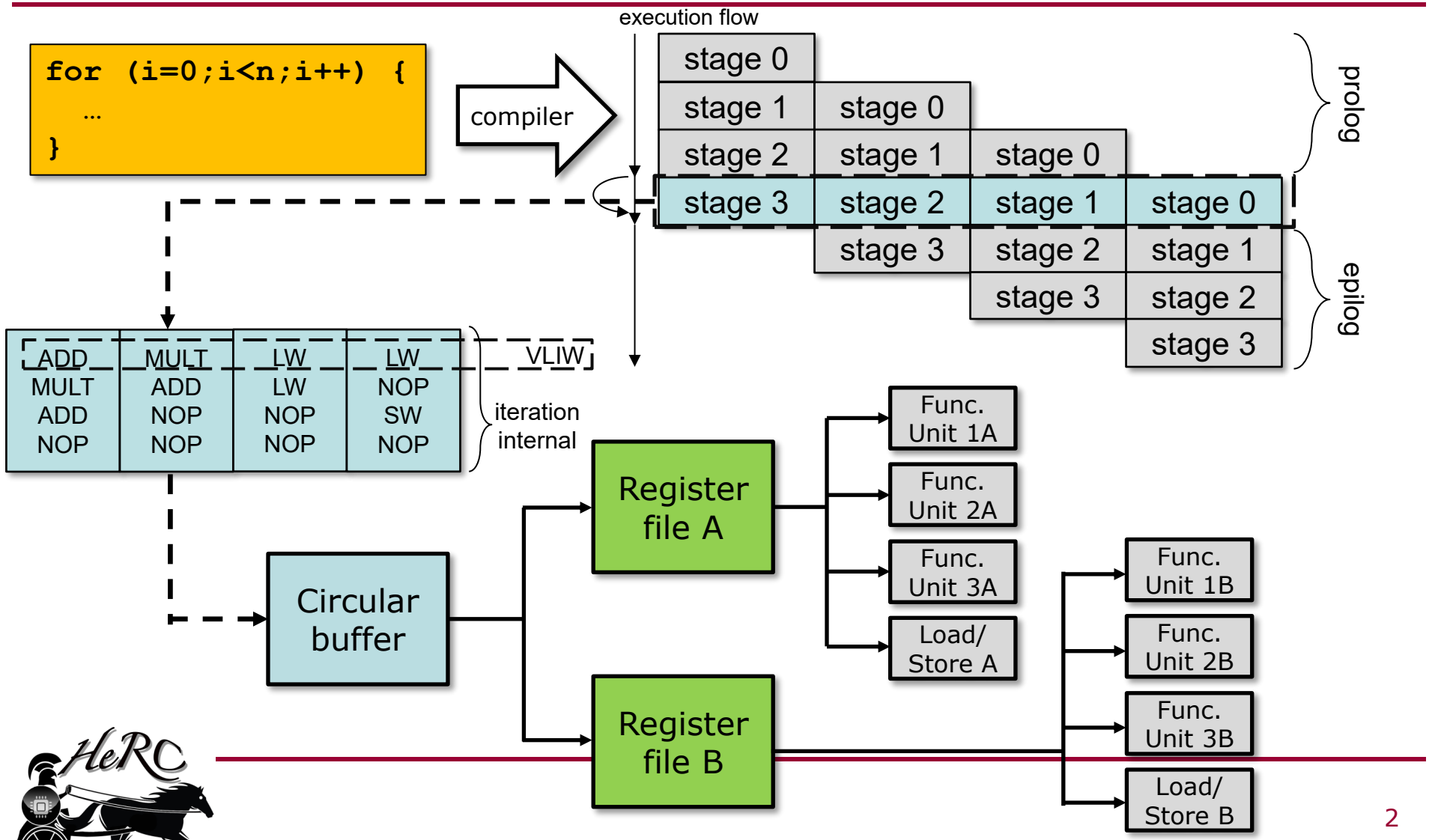
Madushan Abeysinghe, Jesse Villarreal,
Lucas Weaver, Jason D. Bakos

Heterogeneous and Reconfigurable Computing Group

UNIVERSITY OF SOUTH CAROLINA

TEXAS INSTRUMENTS

TI C66 Digital Signal Processor



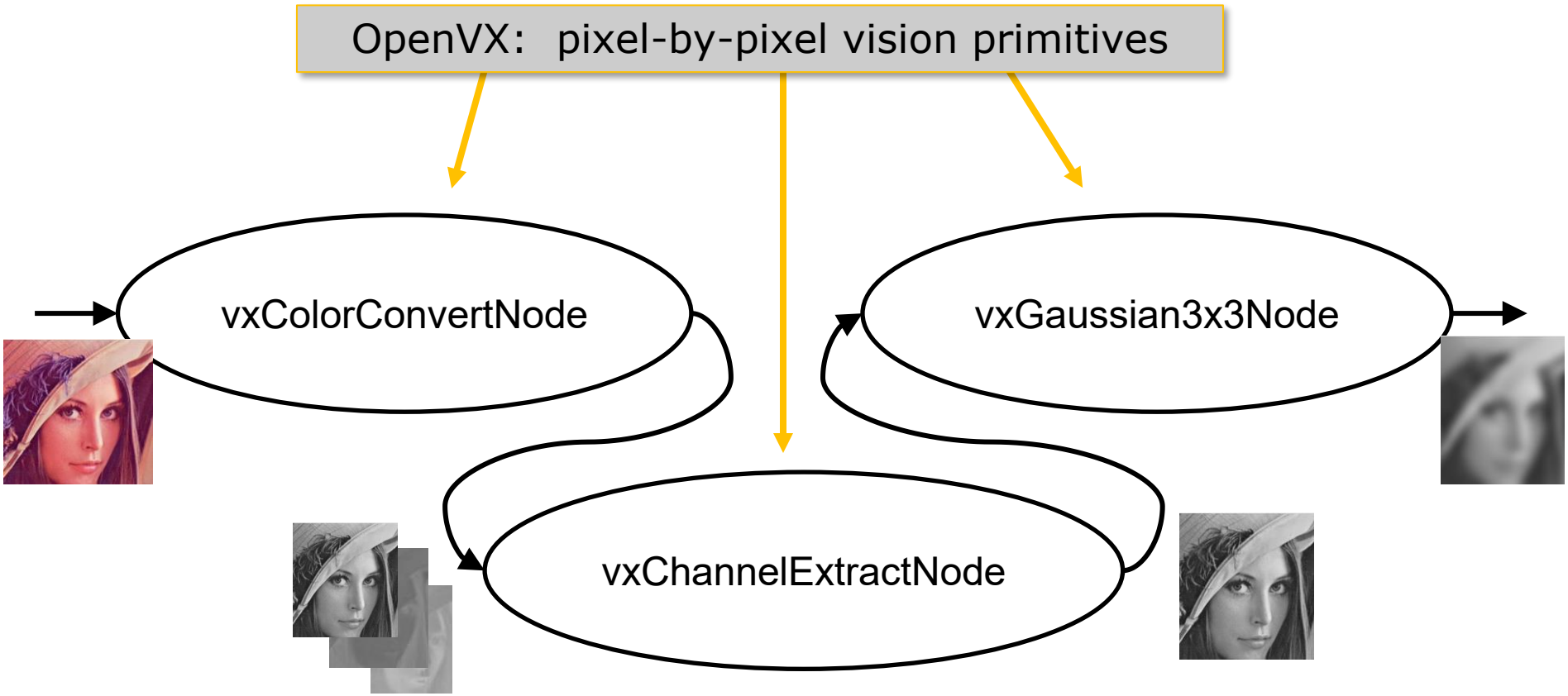
OpenVX Framework

- OpenVX is a standardized, cross platform framework to aid in development of accelerated computer vision, machine learning, and other signal processing applications.

- OpenVX allows the programmer to define an application using a graph-based programming model
  - Nodes: highly optimized kernels for the specific architecture
  - Edges: represents data flow

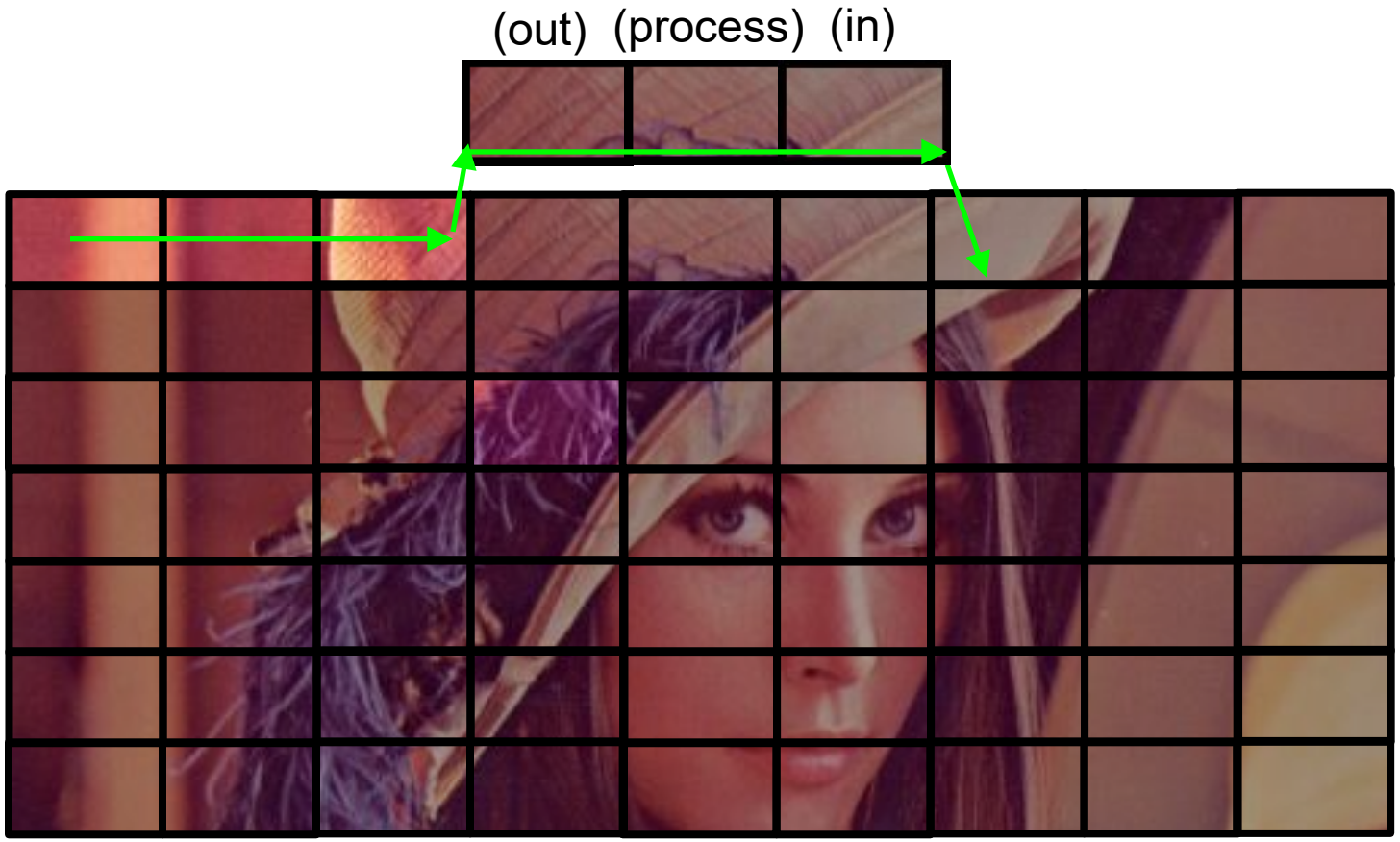
- Code-portable and Performance-portable !

OpenVX/BAM on ADAS TDA2x

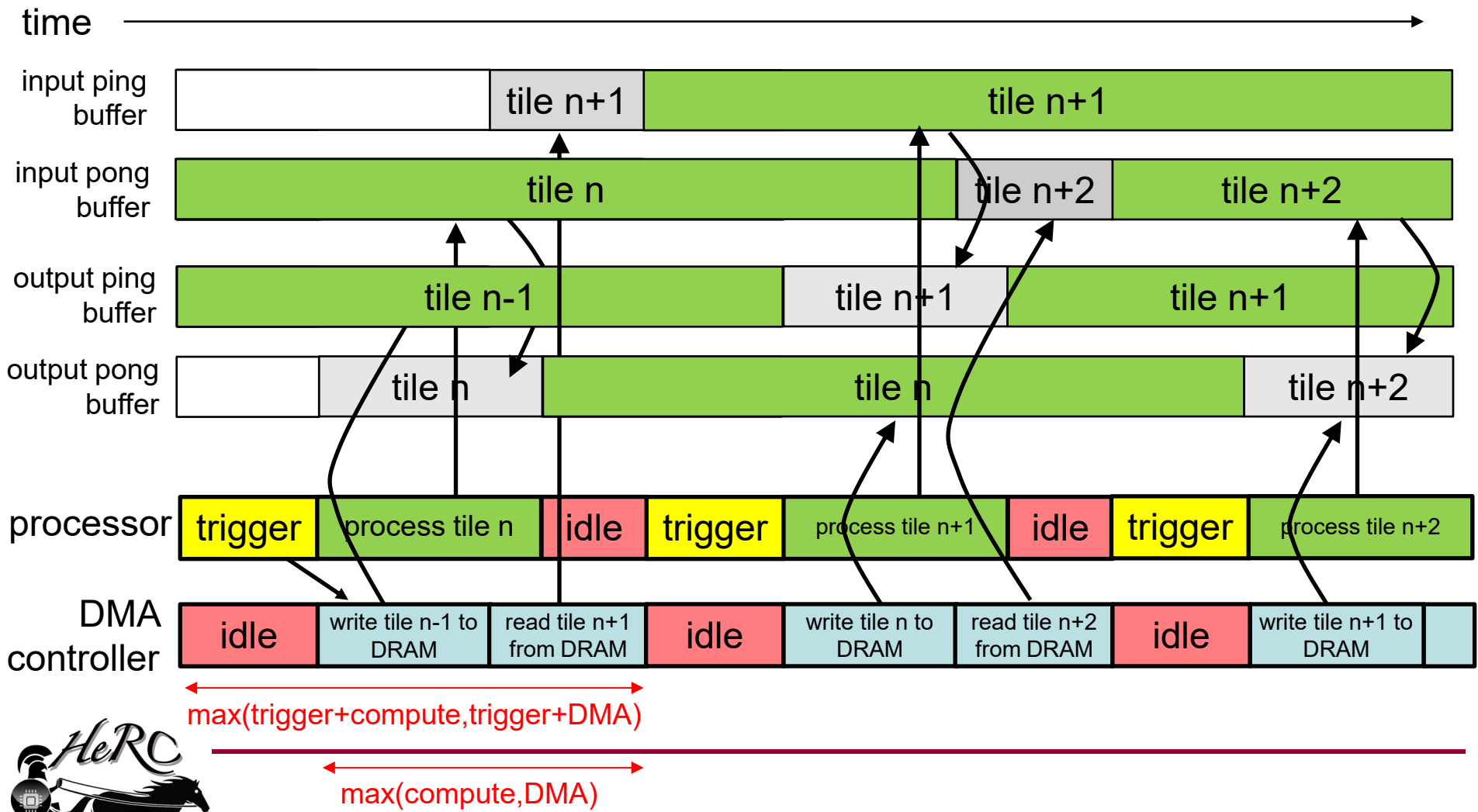


OpenVX and BAM (current work)

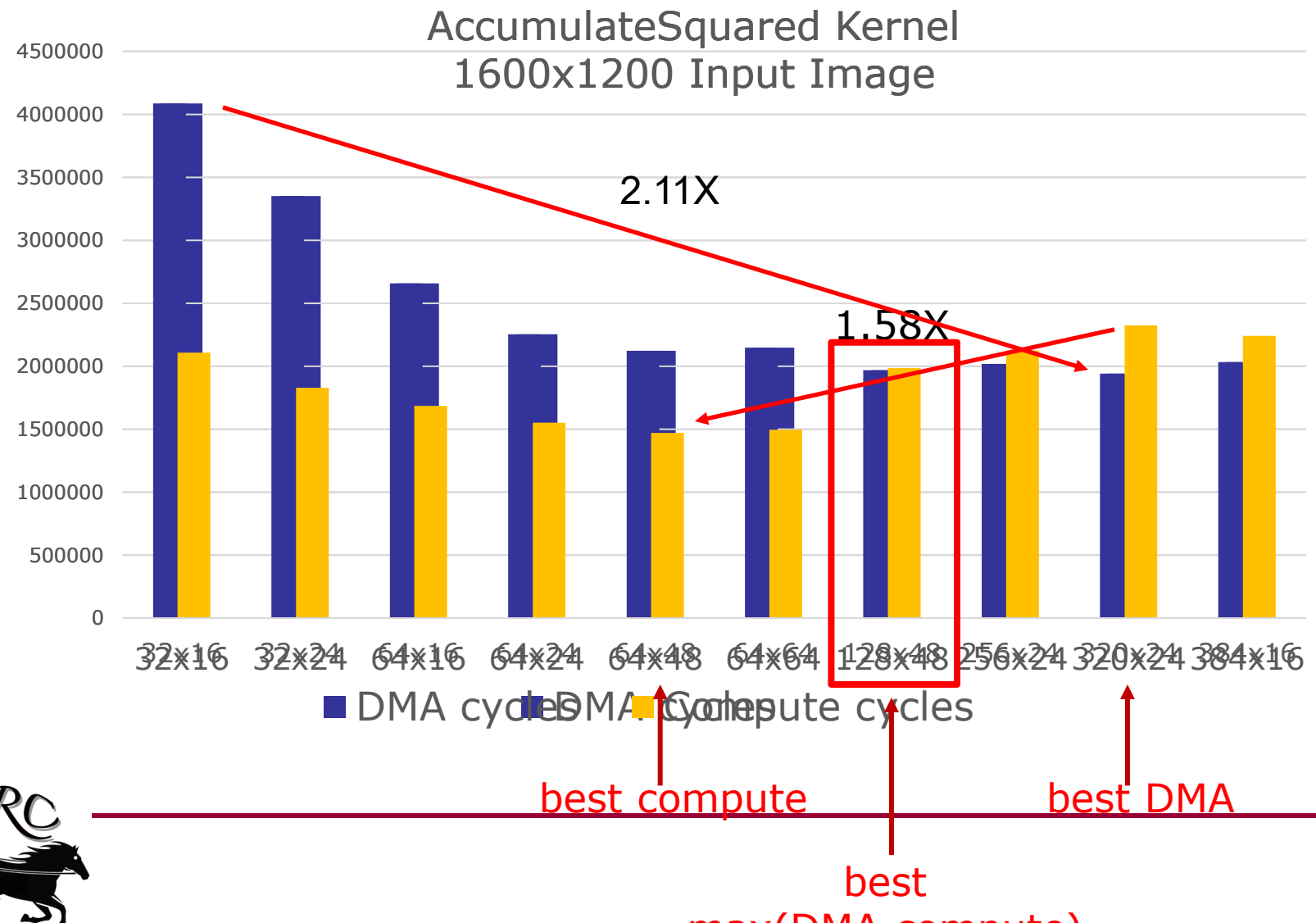
OpenVX and BAM (current work)



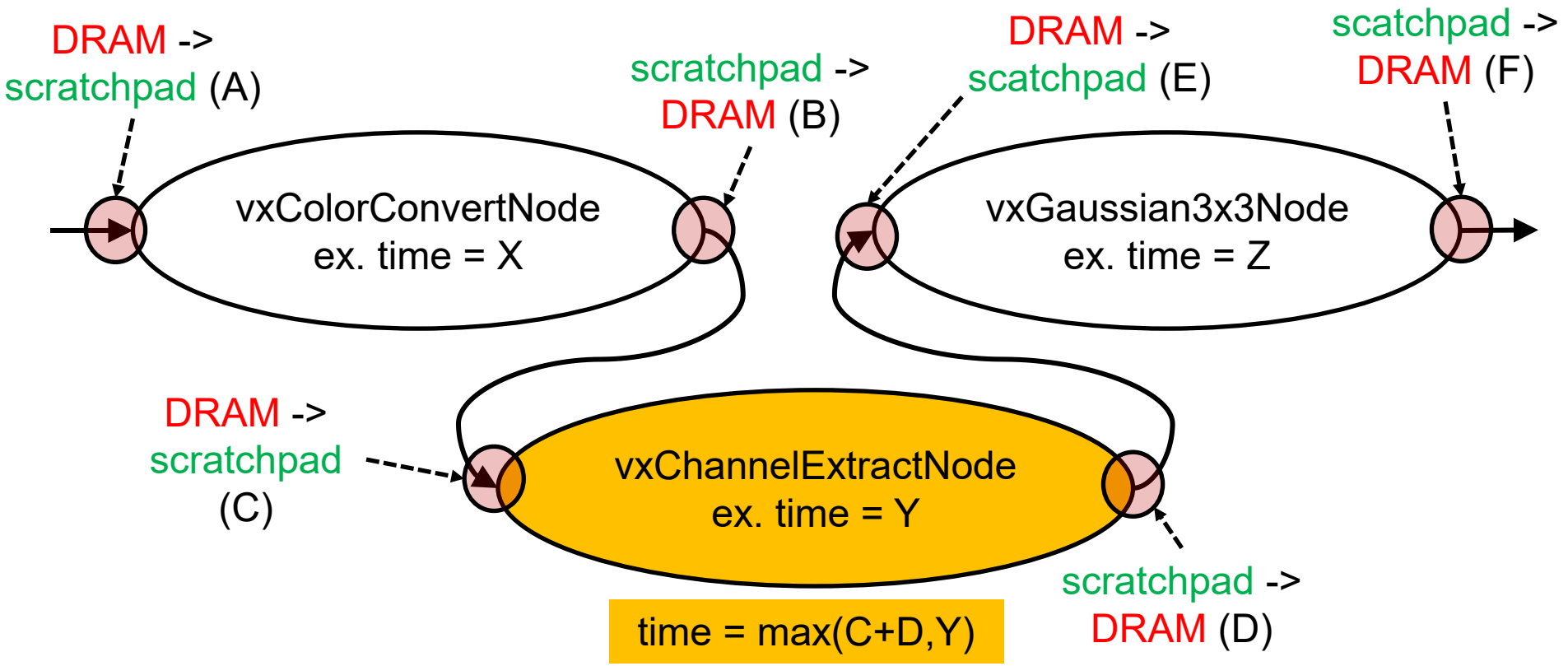
DMA and Scratchpad



Performance Factor 1: Tile Size

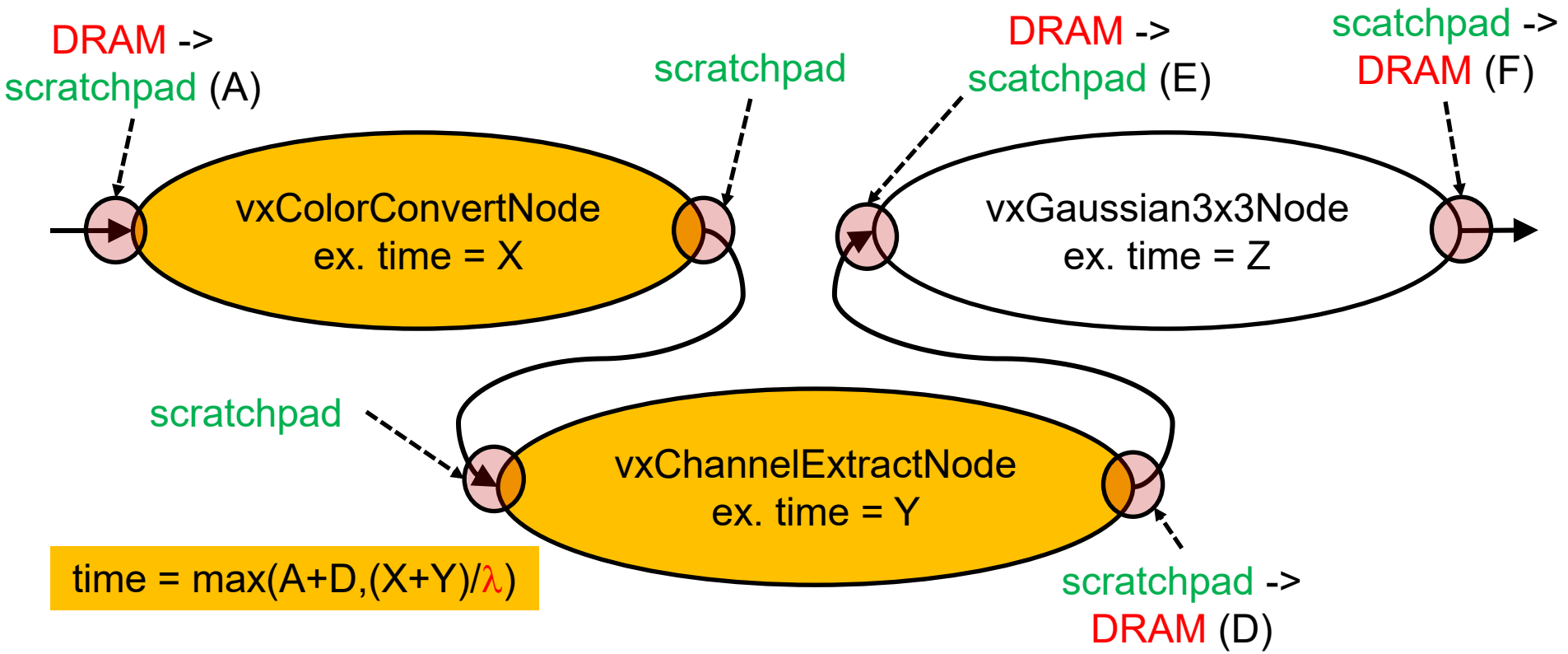


Performance Factor 2: Scratchpad Access Pattern



scratchpad buffers: 2, size = scratchpad/4

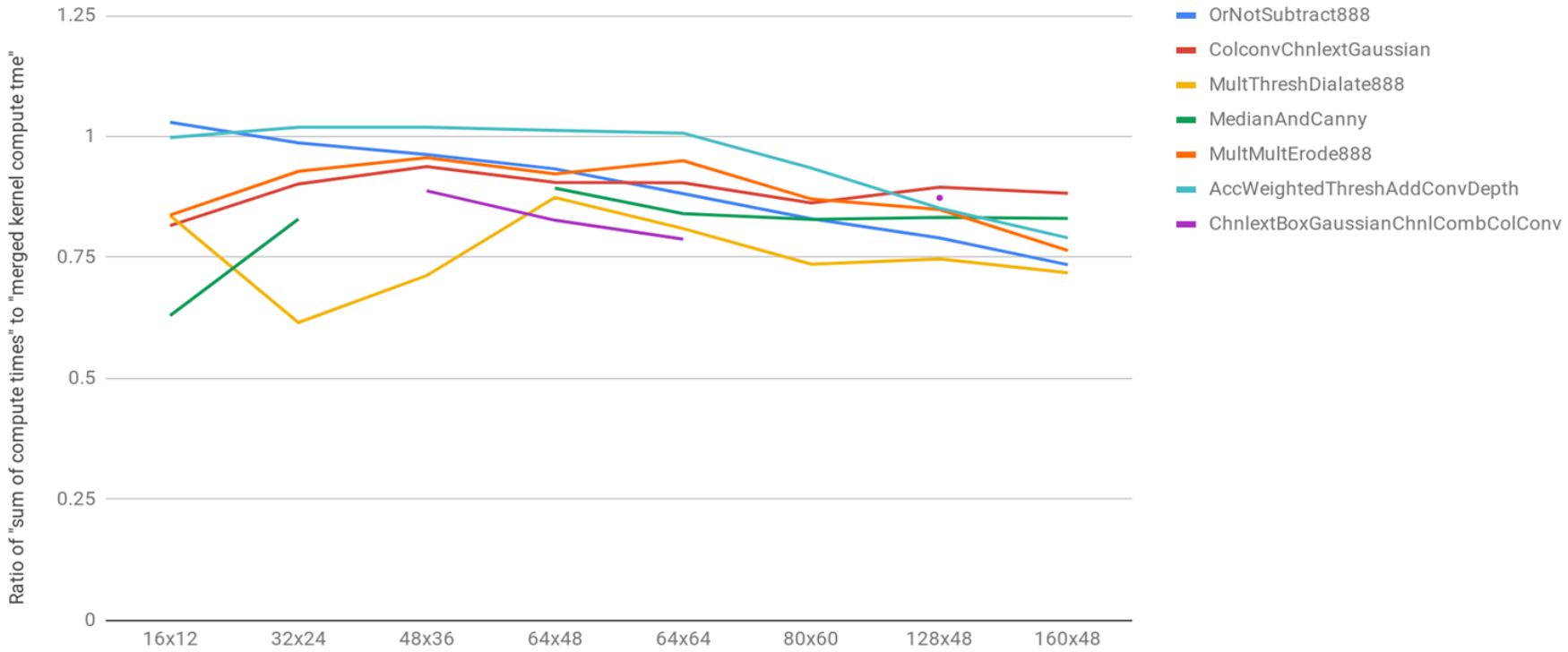
Performance Factor 2: Scratchpad Access Pattern



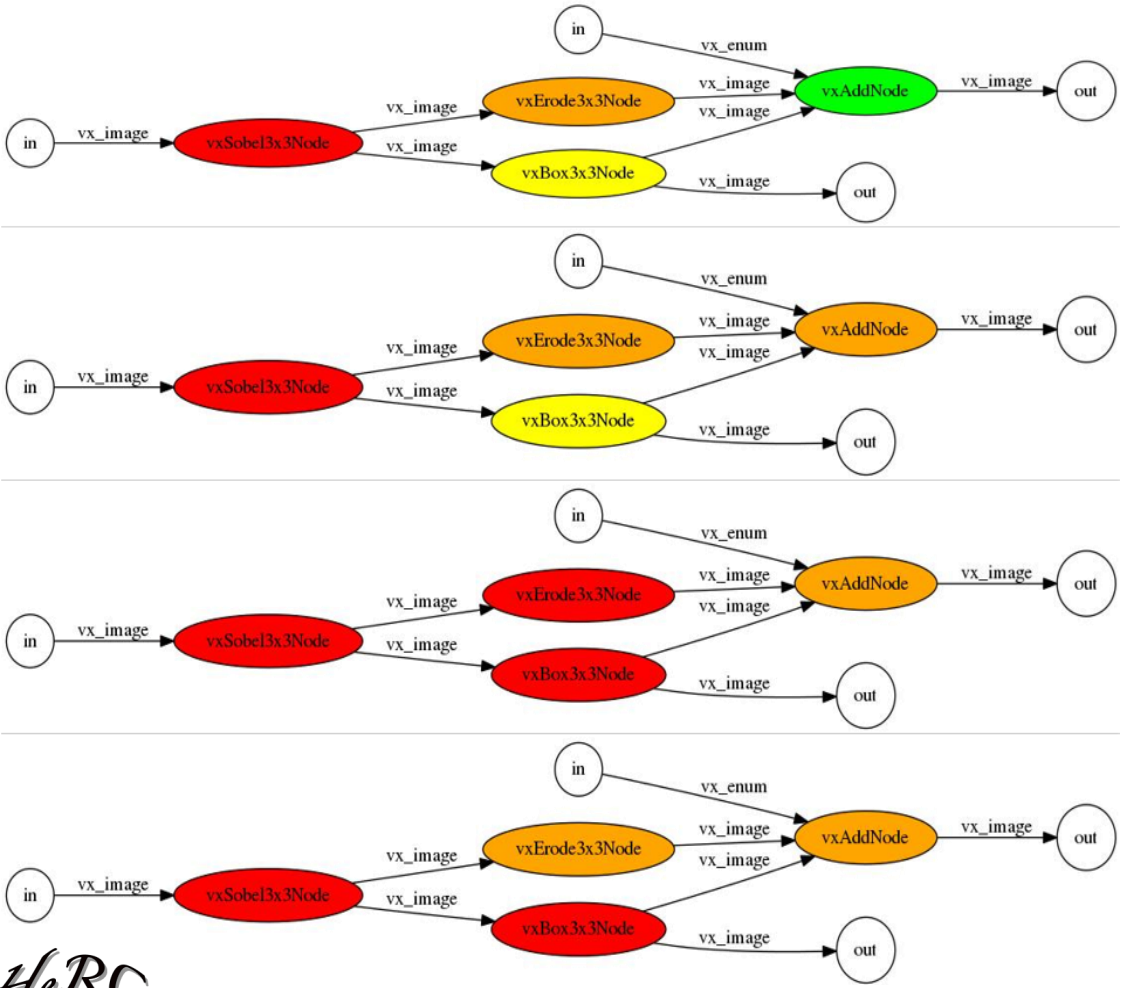
scratchpad buffers: 3, size = scratchpad/6

Lambda

λ Values for Merged Kernels (Observed)



Graph Partitioning



$$B_n = \sum_{k=0}^{n-1} \binom{n-1}{k} B_k$$

$$B_0 = 1$$

$B_6 = 203$
$B_7 = 877$
$B_8 = 4140$
$B_9 = 21147$
$B_{10} = 115975$
$B_{11} = 678570$
$B_{12} = 4213597$

Performance Considerations

- Problem:
  - Group size determines:
    - Number of DMA transactions
    - Maximum tile size
    - Lambda
  - Tile size determines:
    - DMA bandwidth
    - Compute throughput

- Approach:

  Train memory performance model and compute model to optimize grouping and tile size together

DMA/Memory System Performance Model

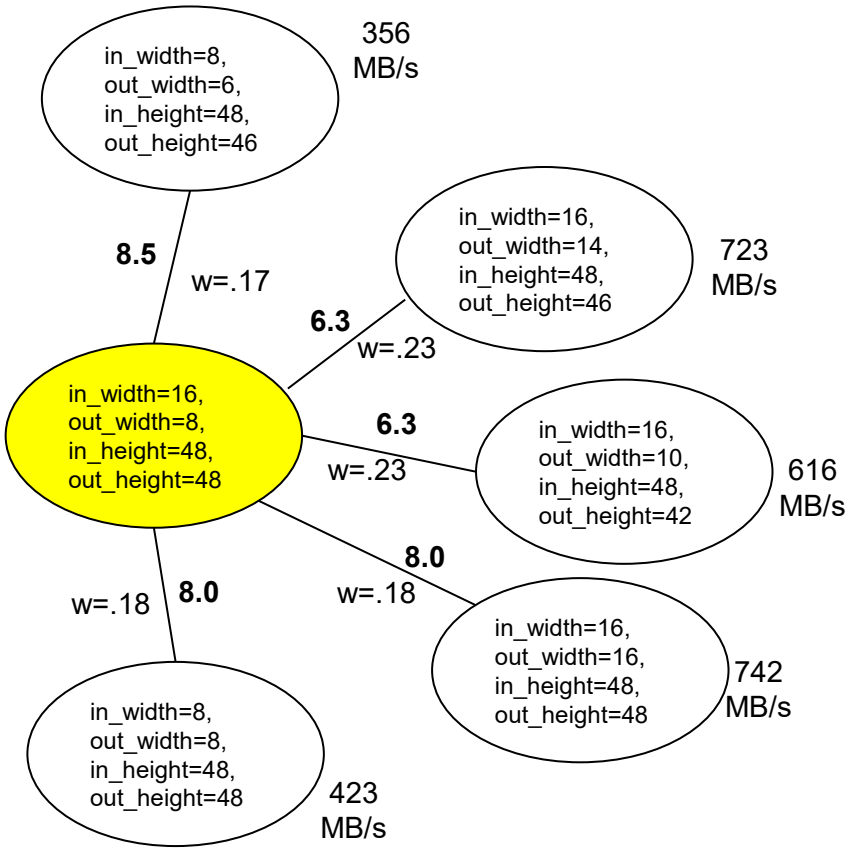
- Characterize:
  - 19 kernels (originally 35), 25 tile sizes, all possible pixel depths (per kernel)
  - **1460** total test cases (culled from 2819 original tests)
- Features:
  1. input tile width, height in pixels (**TIW, TIH**)
  2. output tile width, height in pixels (**TOW**, **TOH)**
  3. total input, output width (tile width x pixel size x number of inputs) (**TTIW, TTOW**)
  4. total input, output size, TTIW*TIH (**TIS, TOS**)
  5. number of inputs, outputs (**NI, NO**)
  6. pixel depth (**PD**)
  7. stencil neighborhood width, height (**SNW, SNH**)

- Best accuracy: **TTIW, TIH, TTOW, TOH**

Performance Models

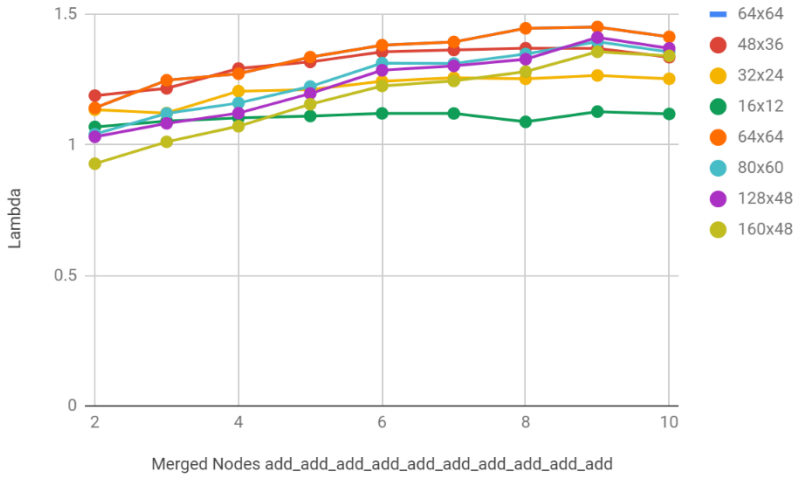
- **DMA: linear interpolant model of DMA b/w**
  - Find nearest 5 observations
    - Calculate weights
      - $w_i = \frac{\frac{1}{distance_i}}{\sum_j \frac{1}{distance_j}}$
    - Calculate weighted average
  - RMS training error = 32 MB/s
  - RMS testing error = 248 MB/s
  - Range = 189 MB/s to 4.33 GB/s

- **Compute: lookup kernel names and nearest tile WxH prod**

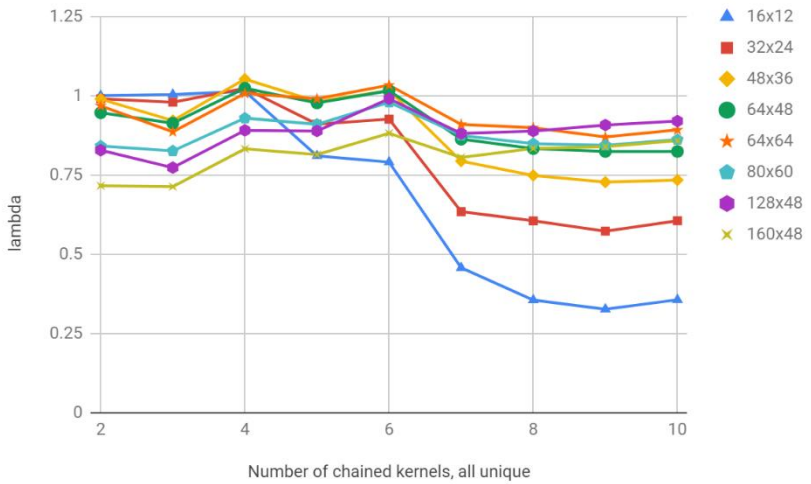


Lamdba Model

Lamdba vs graph size: same kernel

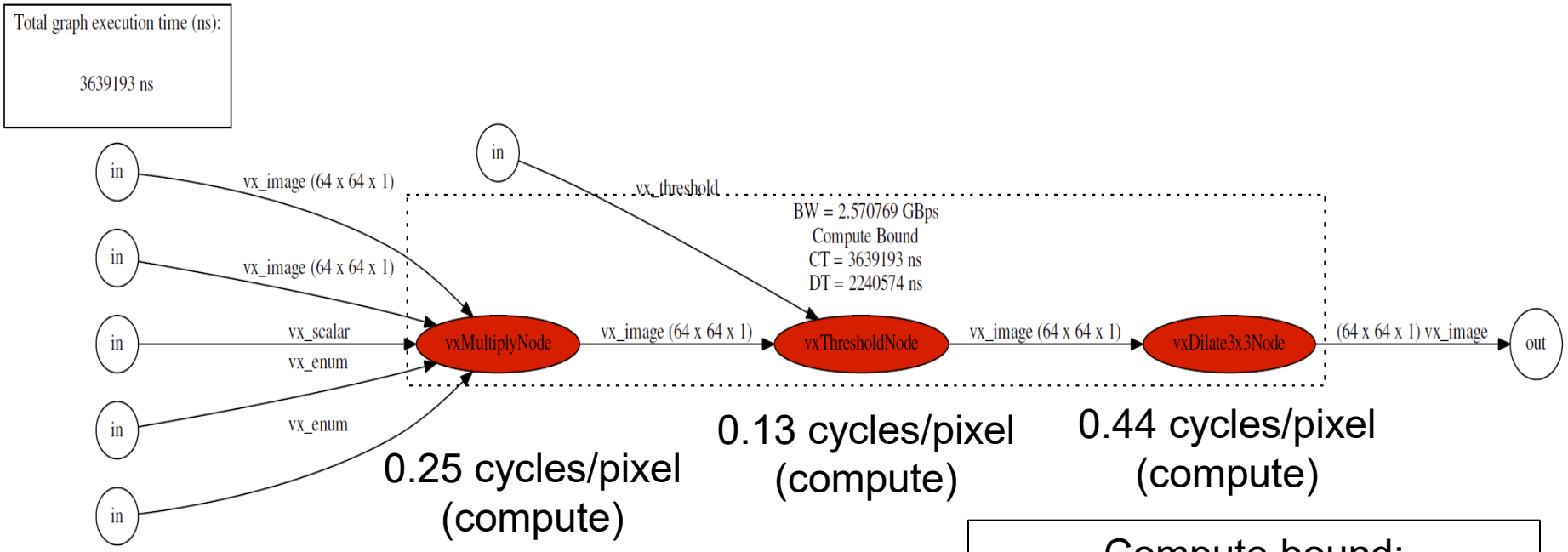


Lamdba vs graph size: unique kernels



- Lambda value depends on number of kernels and number of unique kernels
- Used a piecewise interpolated model
  - RMS error = 0.03 for training set of 43 test graphs

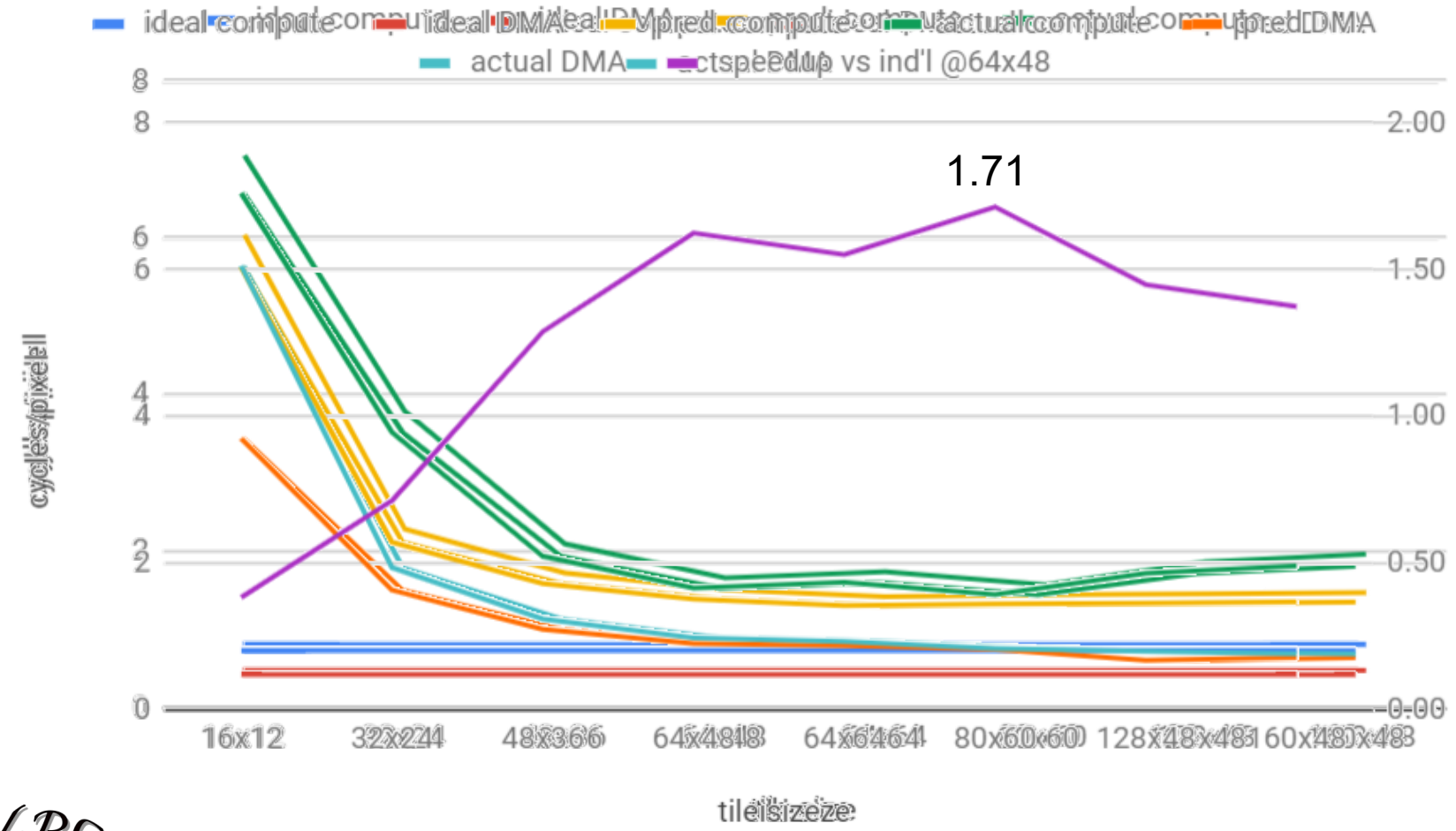
Model Accuracy and Performance



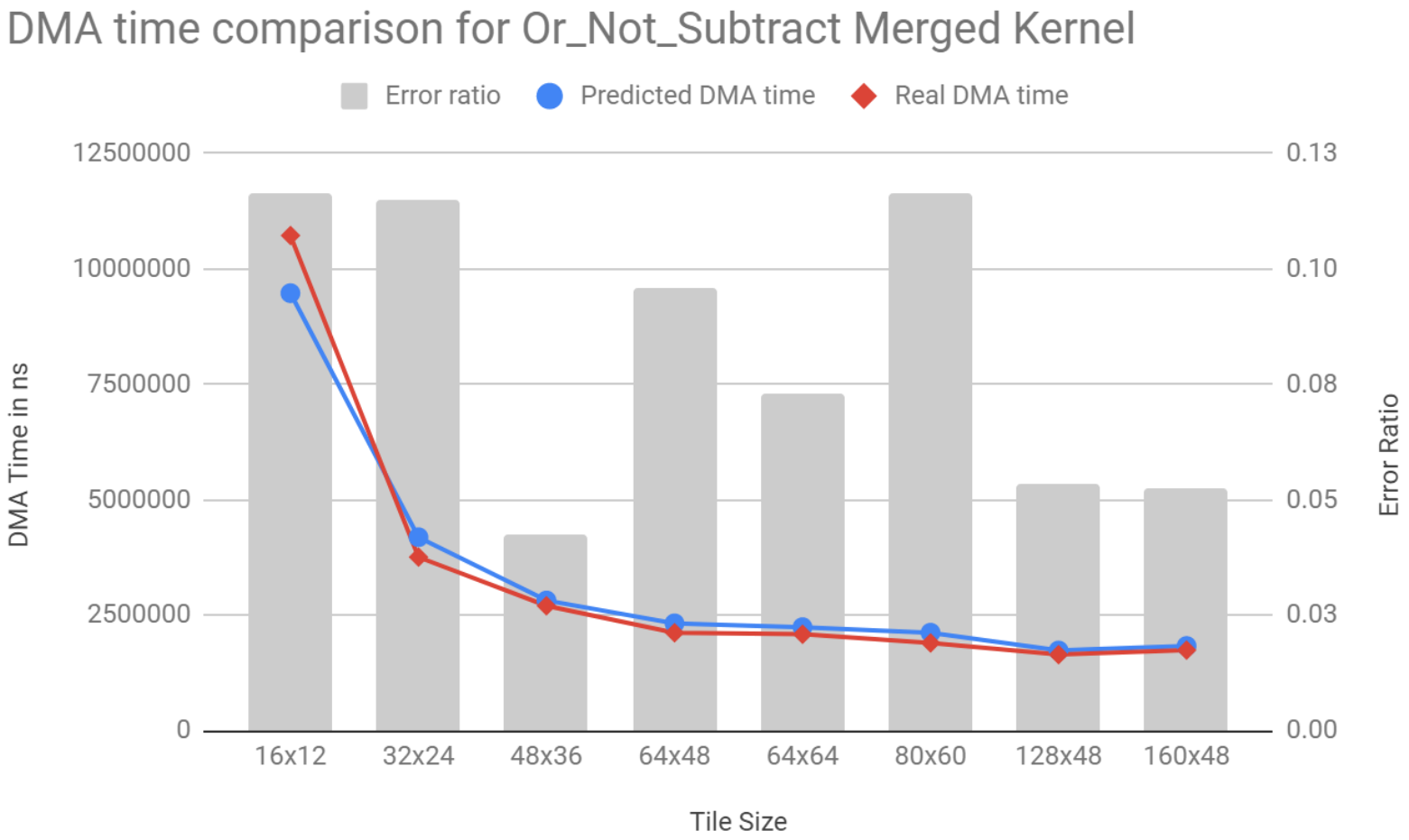
Compute bound:
0.81 cycles/pixel

DMA bound:
3 x 1600x1200x1 DMA @ 4.33 GB/s @ 750 MHz
0.48 cycles/pixel

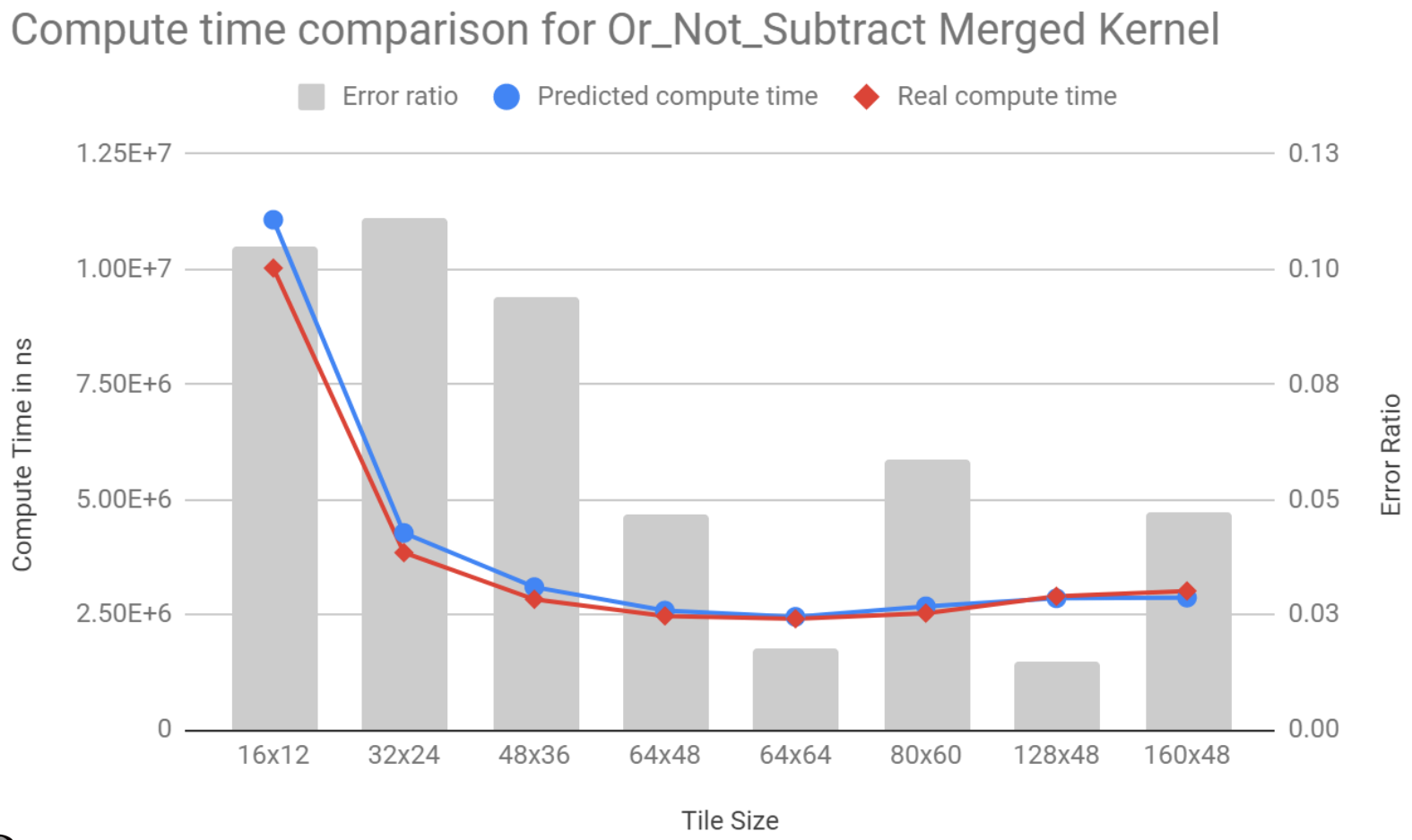
Impact of Tile Size on Merged Kernels



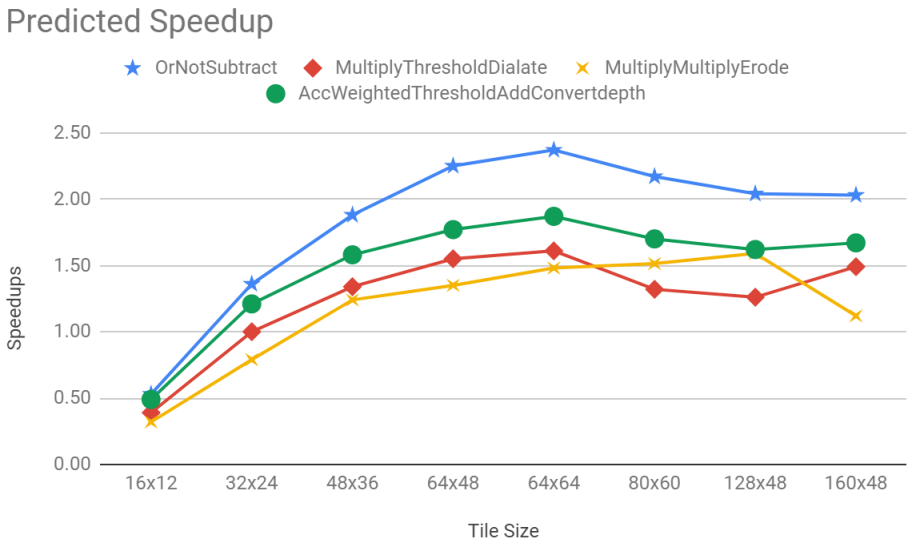
Model Accuracy

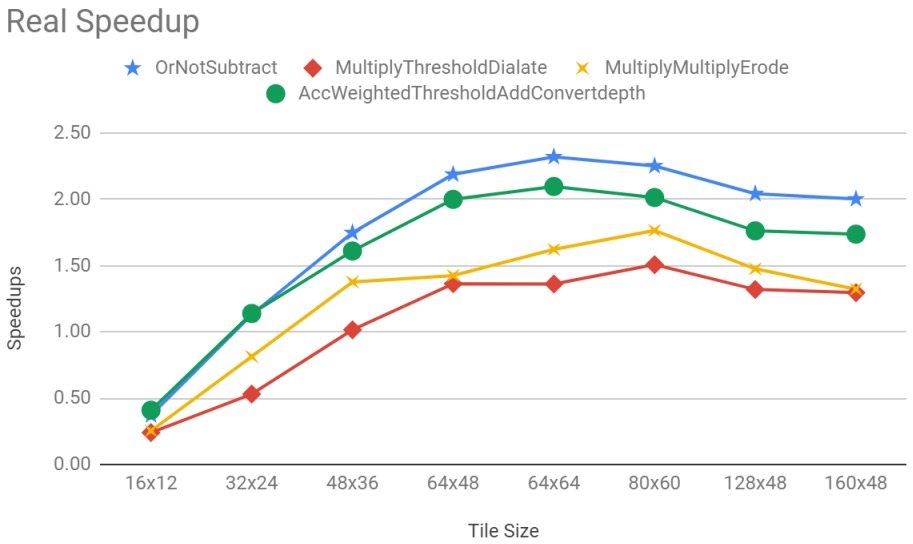


Model Accuracy



Speedups

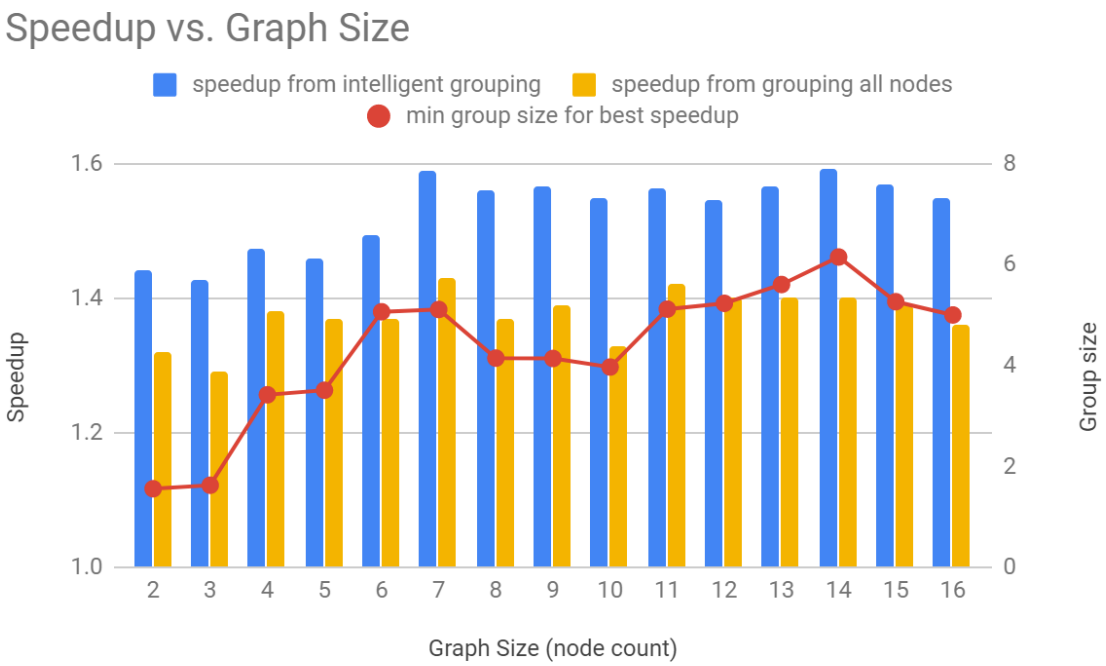




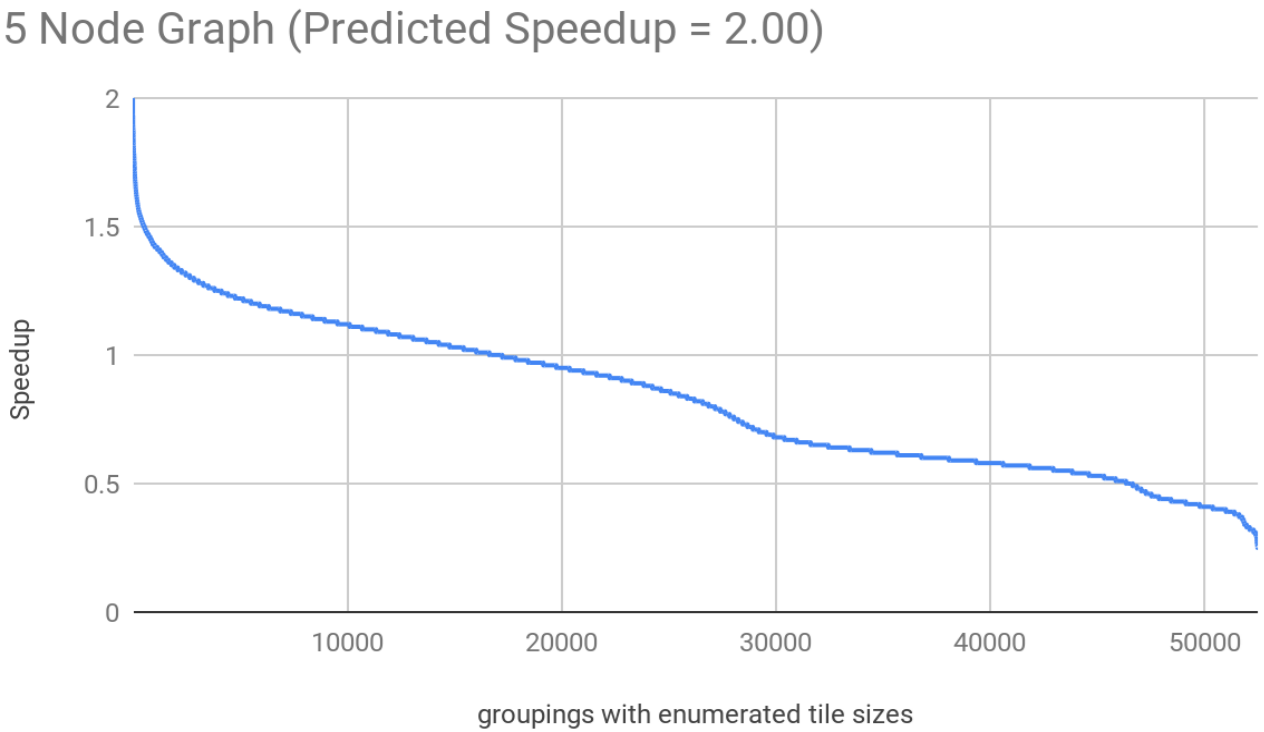
Comparison of Predicted and Real Speedups Speedups

Randomly Generated Graphs

- For each graph
  - For each grouping
    - For each allocable tile size (constrained by scratchpad capacity)…
      - Find predicted speedup (vs individual nodes with 64x48 tiles)

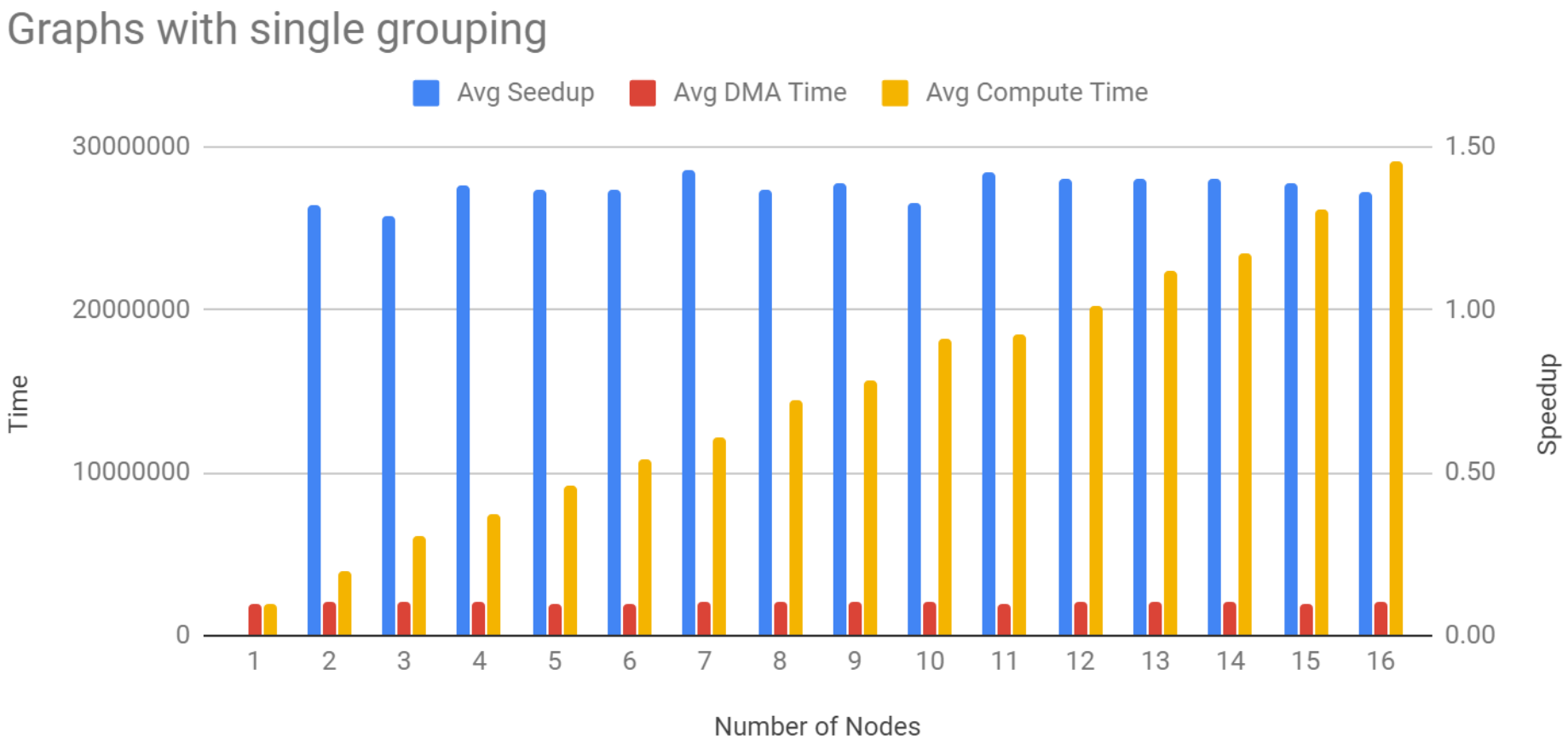


Speedup Distribution



- Speedups above 1.5 = 0.95%
- Speedups between 1 and 1.5 = 30.74%
- Speedups below 1 = 68.30%

Compute/DMA Makeup



Conclusions

- Performance models for DMA time and compute time for OpenVX graphs on C66 DSP
  - Accurate to within ~10 to 15%
- When used to automatically group random graphs, achieves a speedup of ~1.5 to 2

- Future Work:
  - Develep models to predict performance from merging the loops for each OpenVX kernel

Thank you!

Questions?