

Algorithm and Architecture Optimization Targeting Memory Bandwidth Efficiency for Large Size 2D FFT

Berkin Akin

Electrical and Computer Engineering Department
Carnegie Mellon University



The “Memory Wall”

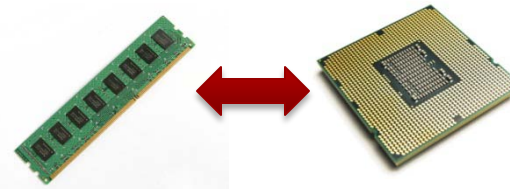


- “Growing gap between **on-chip processing throughput** and **off-chip memory bandwidth**”
 - “**Pin counts** increase at rate of **10% per year** in contrast to **doubling transistor densities** every **18 months**”
 - *International Technology Roadmap for Semiconductors (ITRS)*

- Memory bandwidth becomes the bottleneck



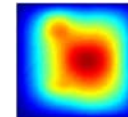
- Efficient use of memory bandwidth should be first class design consideration



Application: Large 2D-FFT on FPGA

- Important computational kernel in scientific computing, signal processing

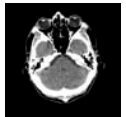
Solving
PDEs



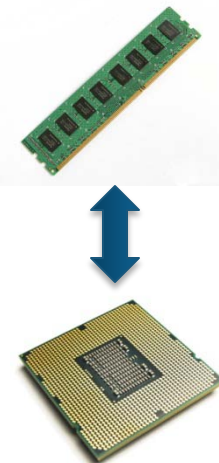
SAR



CT



- Typical datasets are *large* and *high precision*
 - 2K-by-2K double precision 2D-FFT:
 - Input dataset: **64 MB**
 - # of operations: **461 Mflops**
- Data is stored *off-chip* in *DRAM*
 - Conventional algorithms do not have efficient DRAM *access patterns*



Design Goals and Approach

- **Design goals**
 - Performance
 - Bandwidth efficiency
 - Power efficiency

- **Algorithm and architecture optimization**
 - Efficient DRAM access patterns
 - Maximum computation throughput
 - Balanced architecture
 - Scalable and parameterized hardware

H.T. Kung, "Memory requirements for balanced computer architectures," ISCA '86

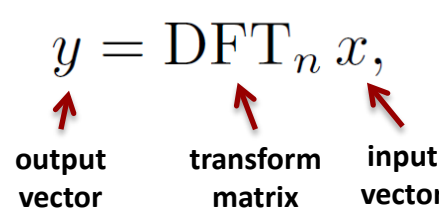
Outline

- Motivation
- **Background**
- Existing Implementations
- Our DRAM Optimized Design
- Results
- Conclusion

Discrete Fourier Transform (DFT)

- DFT as a matrix-vector multiplication:

$$y = \text{DFT}_n x, \quad \text{DFT}_n = [\omega_n^{k\ell}]_{0 \leq k, \ell < n}, \quad \omega_n = e^{-2\pi i/n}$$



output vector transform matrix input vector

- Fast Fourier Transform (FFT) is a fast *algorithm* for DFT
of operations: $O(n^2) \Rightarrow O(n \log n)$
- Similarly there exists FFT algorithms for 2D-DFT

Fast Fourier Transform (FFT)

■ History

- First discovered by Gauss in 1805
- Fourier publishes in 1807
- Rediscovered by Cooley-Tukey in 1965

■ Workload characteristics for “N-point FFT”

- # of arithmetic ops: $5N\log(N)$
- # of memory transfers: N
- Arithmetic intensity: $5\log(N)$ ops per access

Heideman et al, “Gauss and the History of the Fast Fourier Transform,” Arch. Hist. Sc. ‘85

Cooley and Tukey, “An algorithm for the machine calculation of complex Fourier series,” Math of Comp. ‘65

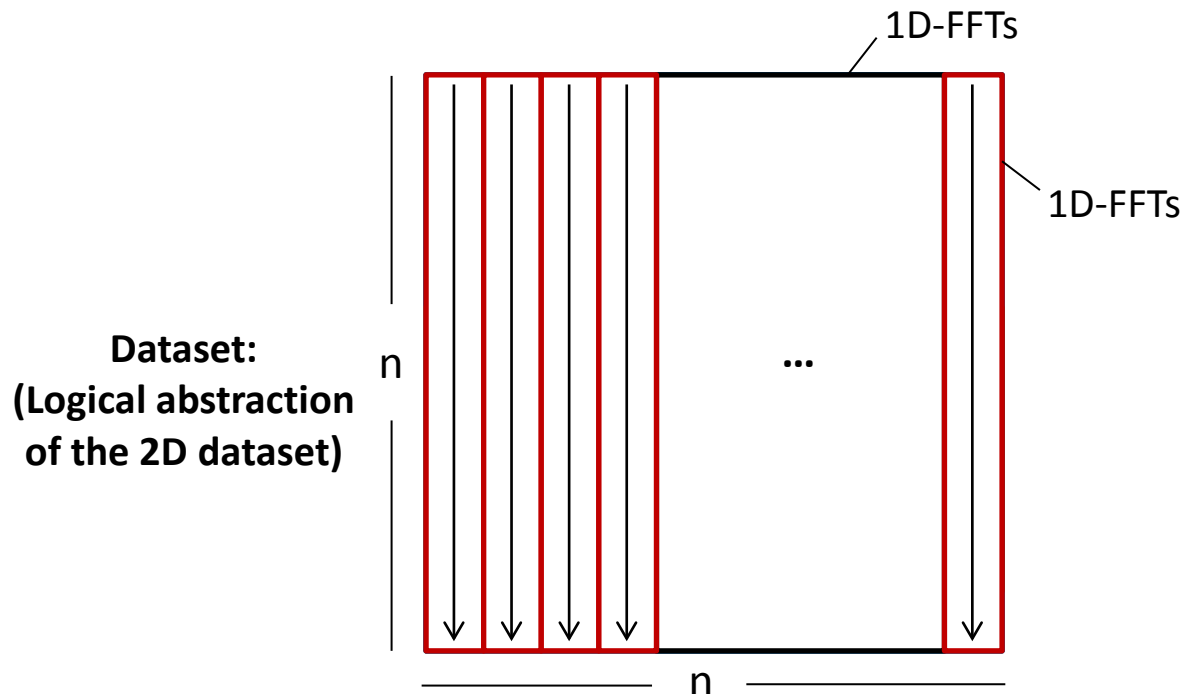
A 2D-FFT Algorithm

- Row-column algorithm:

C. Van Loan, "Computational frameworks for the fast Fourier transform", SIAM '92

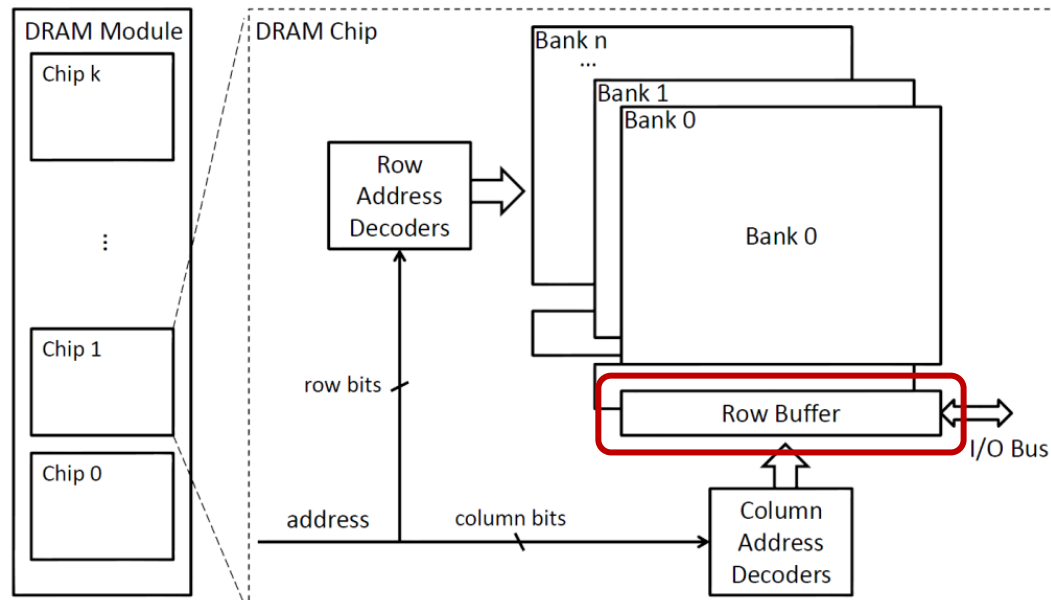
$$\text{DFT}_{n \times n} = \underbrace{(\text{DFT}_n \otimes \text{I}_n)}_{\text{Column Stage}} \underbrace{(\text{I}_n \otimes \text{DFT}_n)}_{\text{Row Stage}}$$

tensor product
identity matrix
DFT matrix



DRAM Organization & Operation

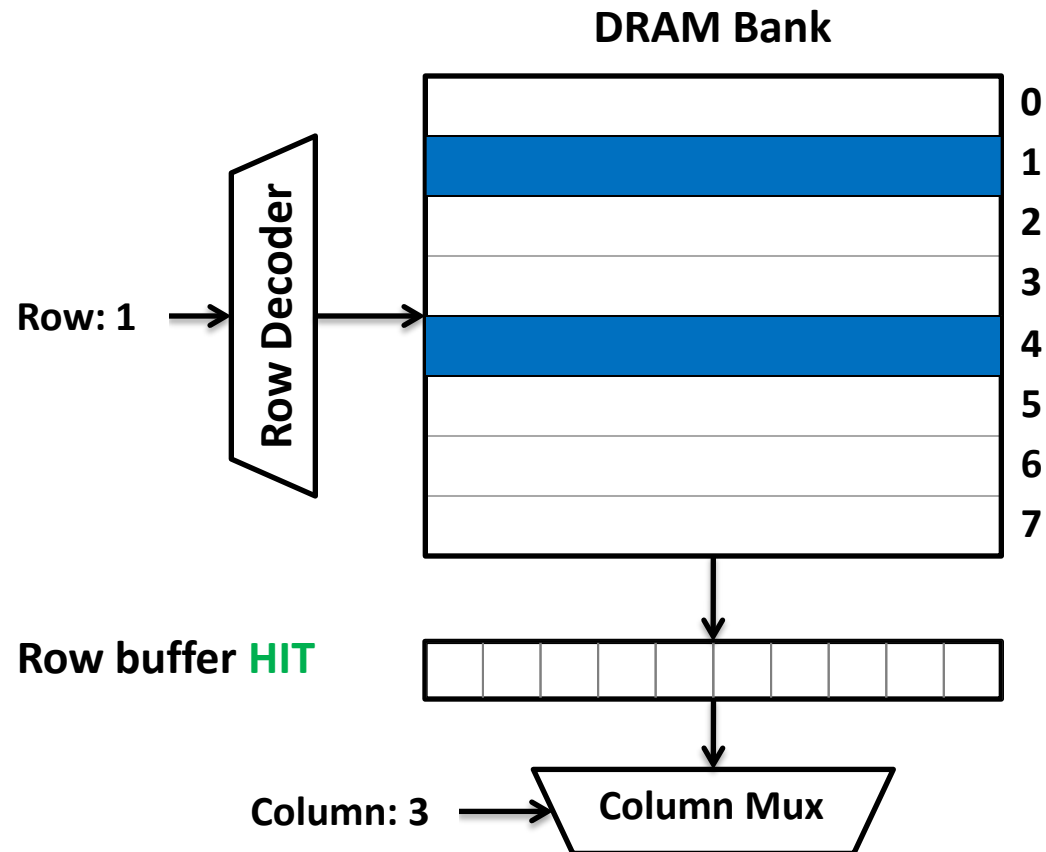
- Commonly used off-chip storage medium
- Divided hierarchically into: Rank, chip, bank, row, column



DRAM Organization & Operation

DRAM addresses:

- 1) Row 4, Column 0
- 2) Row 4, Column 1
- 3) Row 4, Column 7
- 4) Row 1, Column 2
- 5) Row 1, Column 3

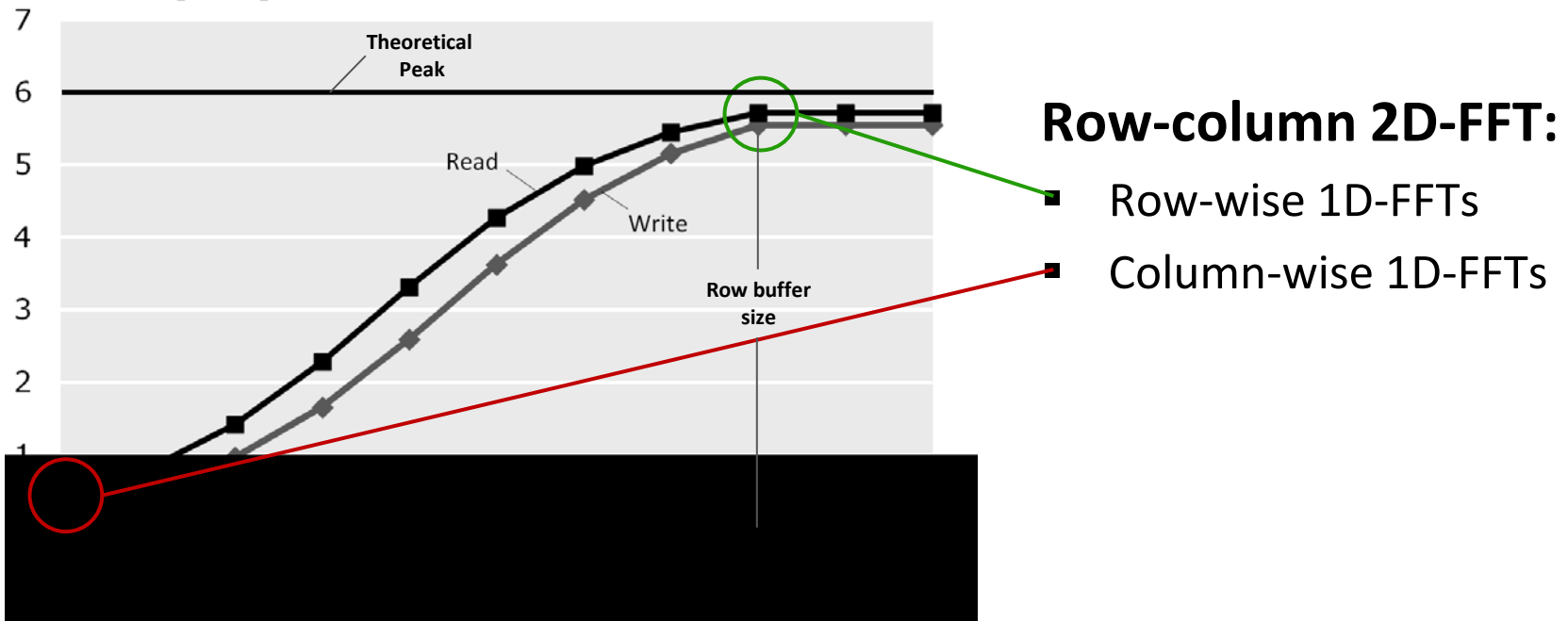


DRAM Bandwidth

- Experiment on DE4 FPGA platform w/ DDR2-800 SO-DIMM
 - “Packet” = Aligned consecutive data
 - Transfer packets that are separated by large strides

DDR2-800 Bandwidth on DE4 (per channel)

Bandwidth [GB/s]



Outline

- Motivation
- Background
- **Existing Implementations**
- Our DRAM Optimized Design
- Results
- Conclusion

Baseline Design

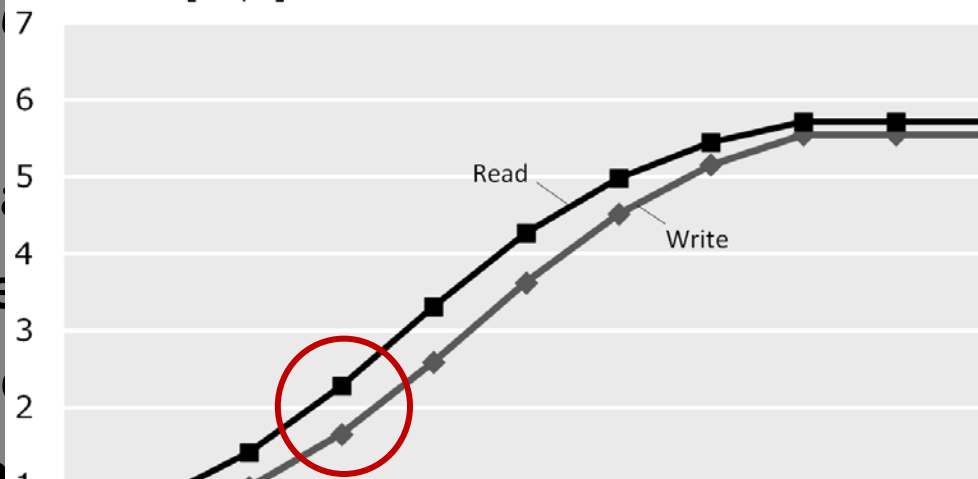
- I.S. Uzun, A. Amira, A. Bouridane, “FPGA implementations of fast Fourier transforms for real-time signal and image processing,” *IEE Proc.-Vision, Image and Signal Processing*, 2005
- **Off-chip problem sizes are considered**
- **Off-chip SRAMs -> No penalty for access pattern**
- **Limitations:**
 - Off-chip SRAMs are expensive
 - 2048-by-2048 double precision 2D-FFT has **64 MB** dataset
 - Used FPGA platform has **8 MB** off-chip SRAM
 - Limited precision (16-bit fixed point)
 - Algorithm is not efficient for platforms with DRAMs

DRAM Optimized Design

- Chi-Li Yu, K. Irick, C. Chakrabarti, V. Narayanan, "Multidimensional DFT IP generator for Embedded Systems, 2007

DDR2-800 Bandwidth on DE4 (per channel)

Bandwidth [GB/s]



- Targets
- Addressed
- Row-wise
- Limitations

- Suboptimal

- Inefficient

- Imbalanced algorithm

- Asymmetric stages (uneven computational load)
 - Different size on-chip FFTs

Outline

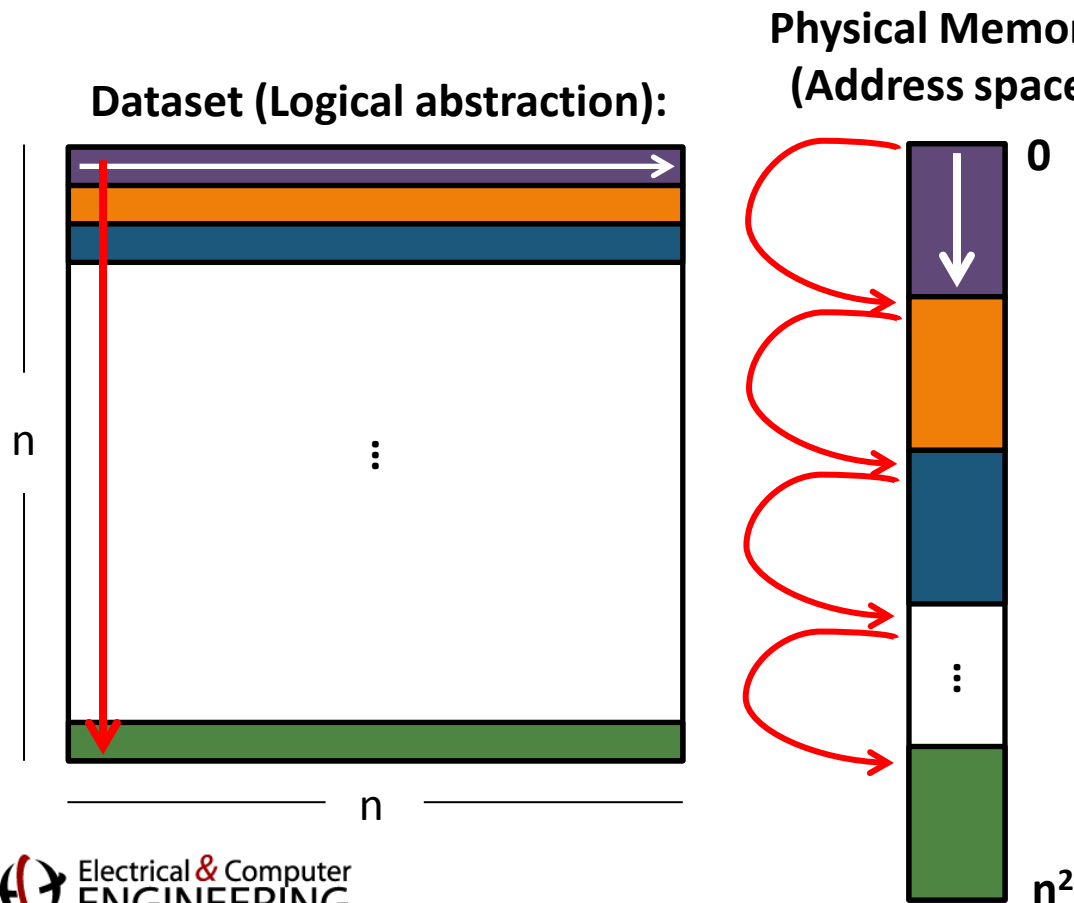
- Motivation
- Background
- Existing Implementations
- **Our DRAM Optimized Design**
- Results
- Conclusion

Our Design - Overview

- B. Akin, P. Milder, F. Franchetti, J.C. Hoe, “Memory bandwidth efficient two-dimensional fast Fourier transform algorithm and implementation for large problem sizes,” *FCCM, 2012*
- **DRAM bandwidth is efficiently utilized**
 - Row buffer locality aware access patterns
- **Balanced algorithm and architecture**
 - Symmetric algorithm (identical operations across stages)
 - Off-chip bandwidth and on-chip throughput is matched
 - Maximum computation throughput is extracted

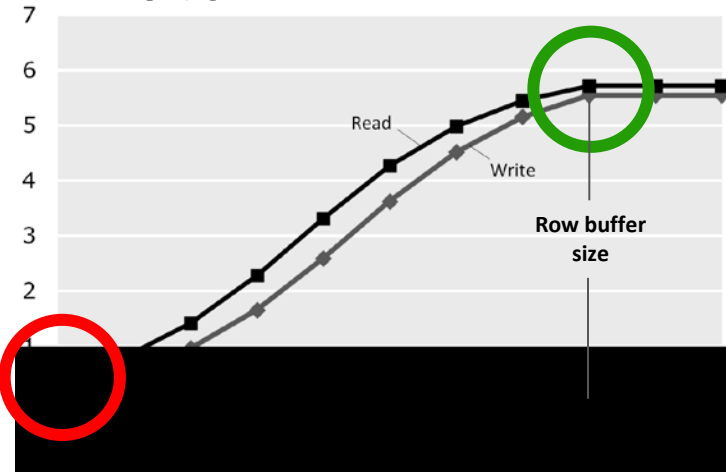
Inefficient DRAM Access Patterns

- Row-wise traversal -> Sequential accesses
- Column-wise traversal -> Large strided accesses

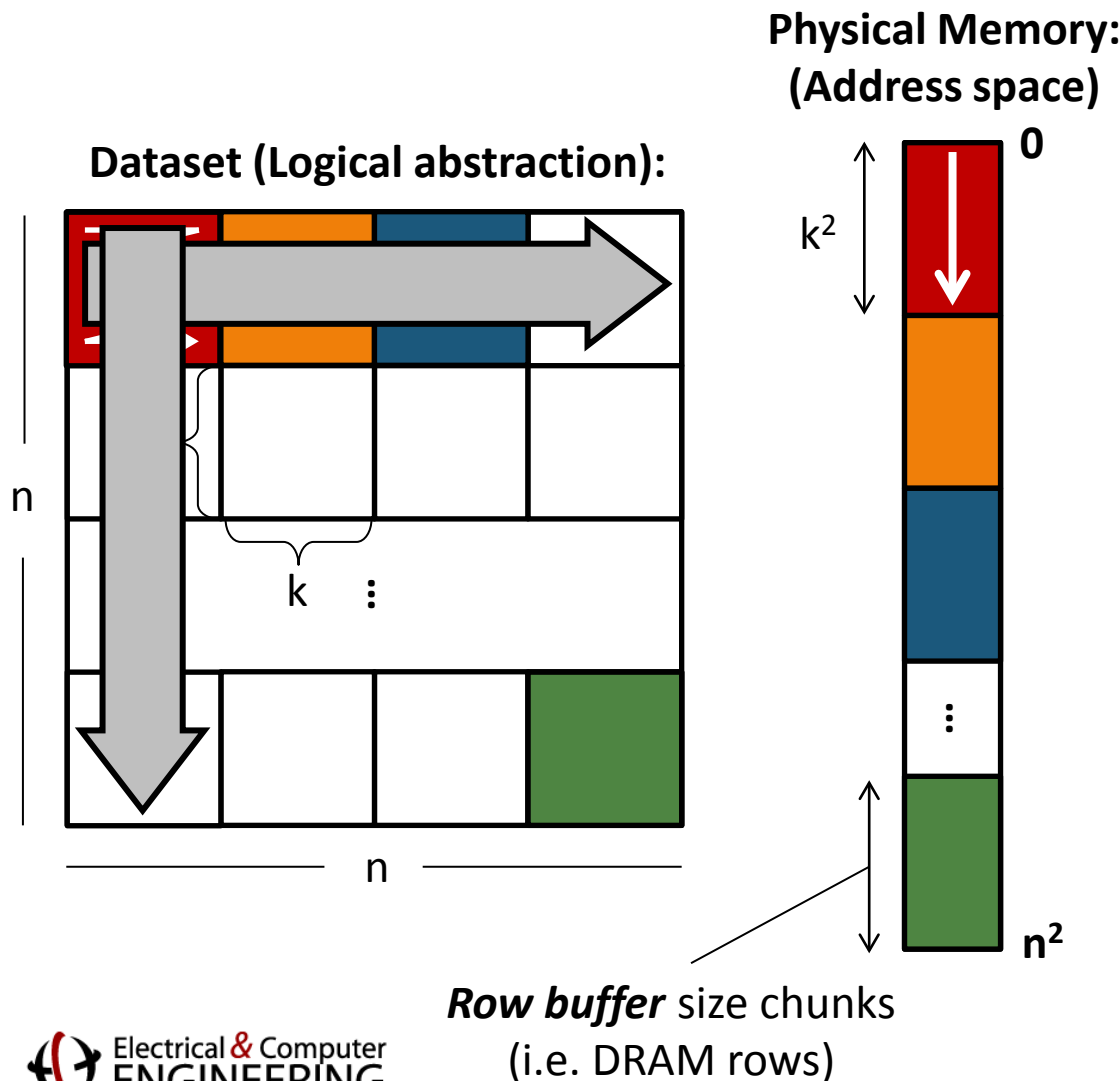


DDR2-800 Bandwidth on DE4 (per channel)

Bandwidth [GB/s]

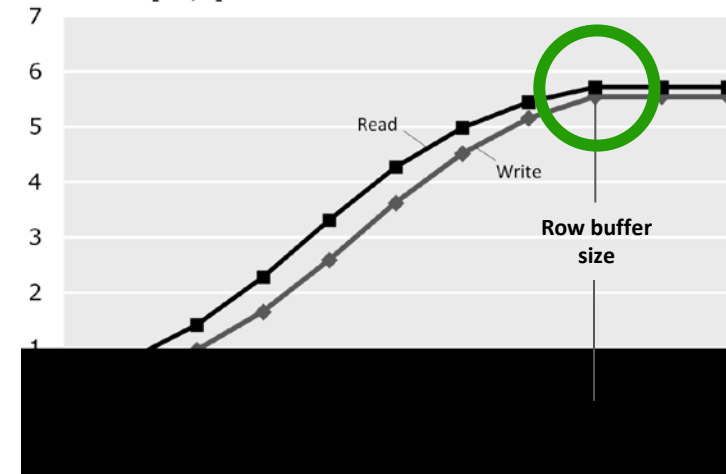


How to Optimize the Access Patterns

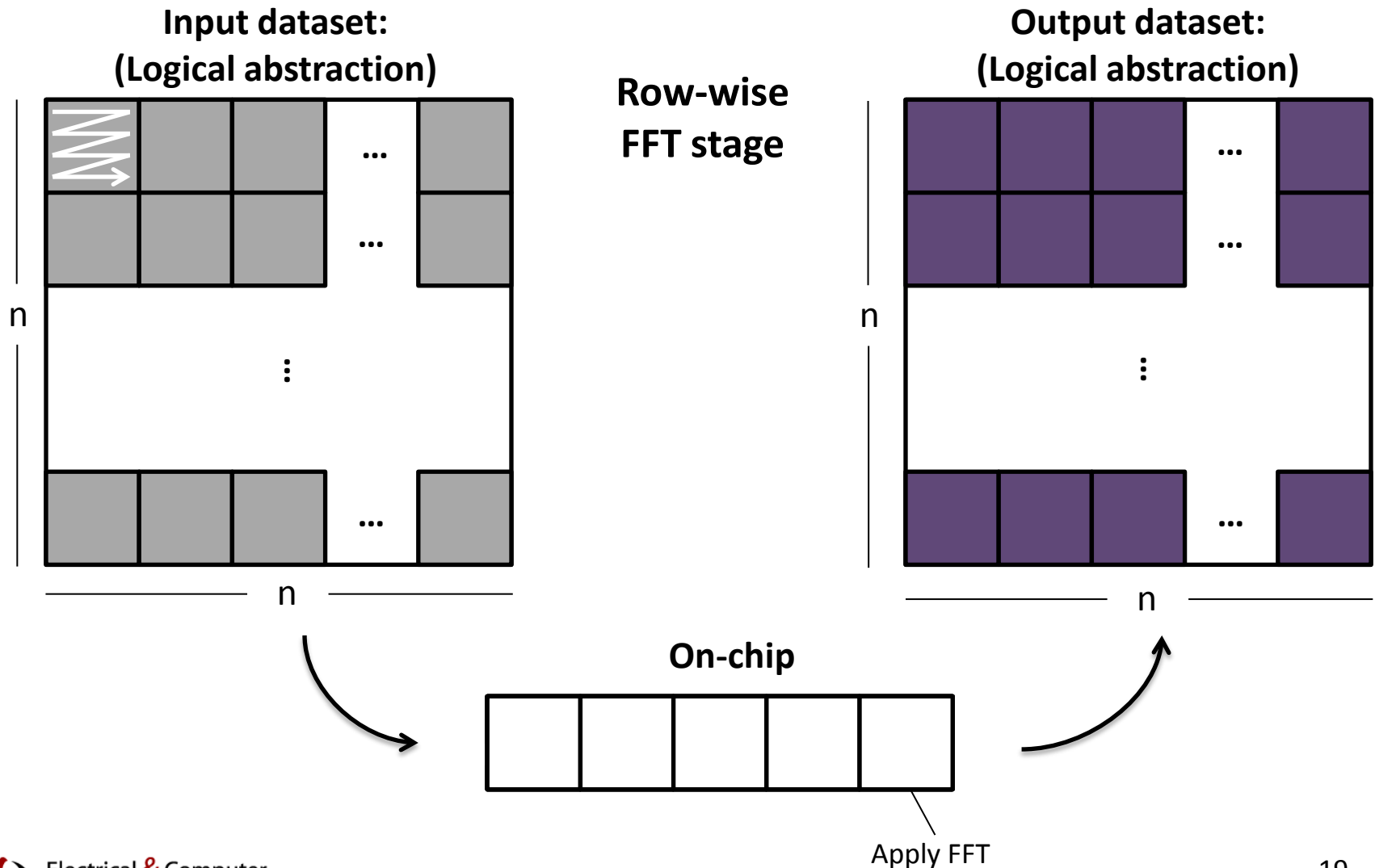


DDR2-800 Bandwidth on DE4 (per channel)

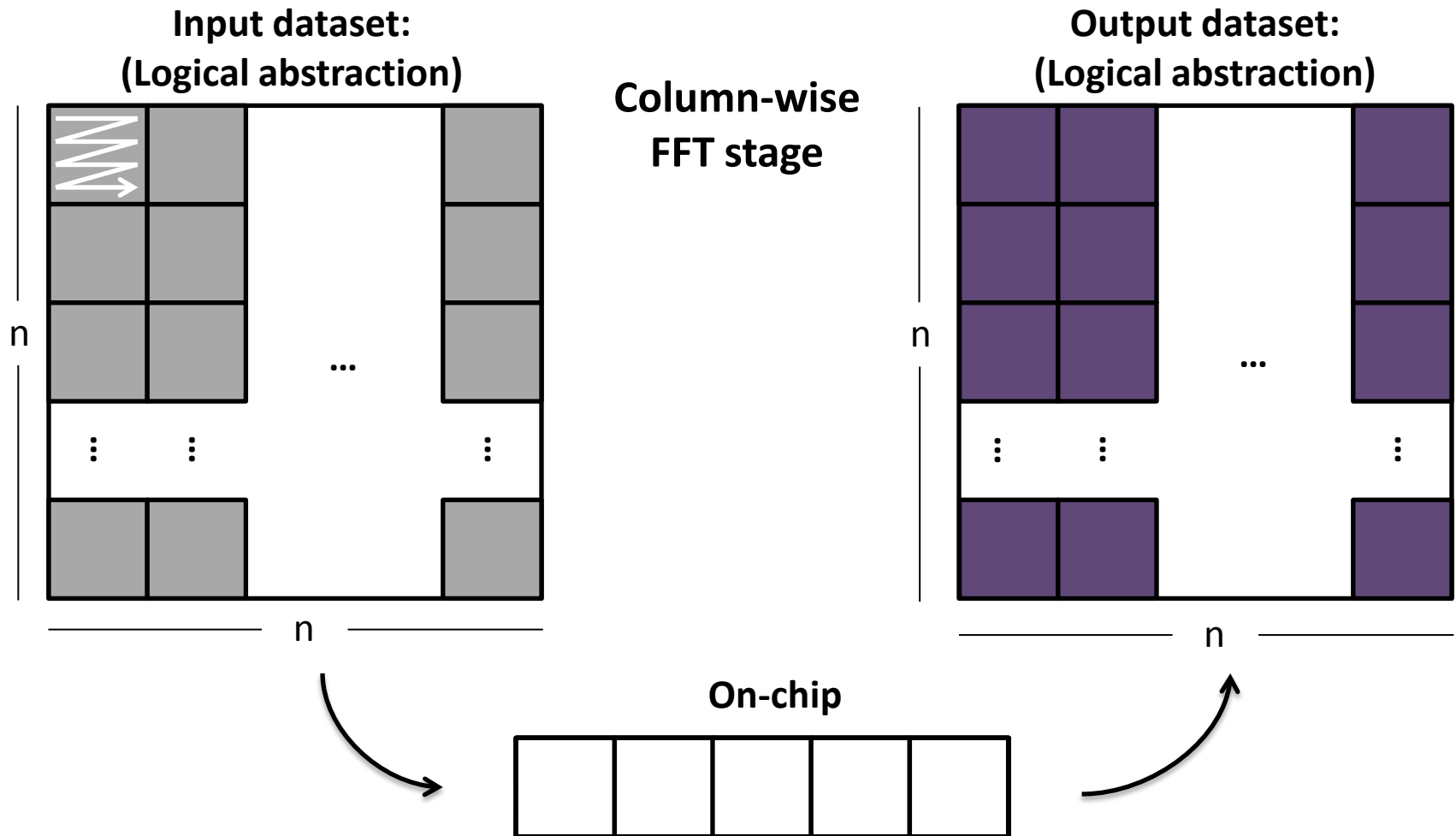
Bandwidth [GB/s]



Row-Column Algorithm with Tiling



Row-Column Algorithm with Tiling



Row-Column Algorithm with Tiling

- **DRAM access pattern problem is solved**
 - We make use of the whole row-buffer whenever we access DRAM
 - Row-buffer misses are minimized
- **Stages of the algorithm are not the same!**
 - Datapath can be simplified by having identical stages
- **Design generator based on tensor formalism**
 - We can handle more complicated but more elegant algorithms
 - Parameterized and scalable hardware
 - Can be tuned for different memory systems

Design Generator w/ Tensor Formalism

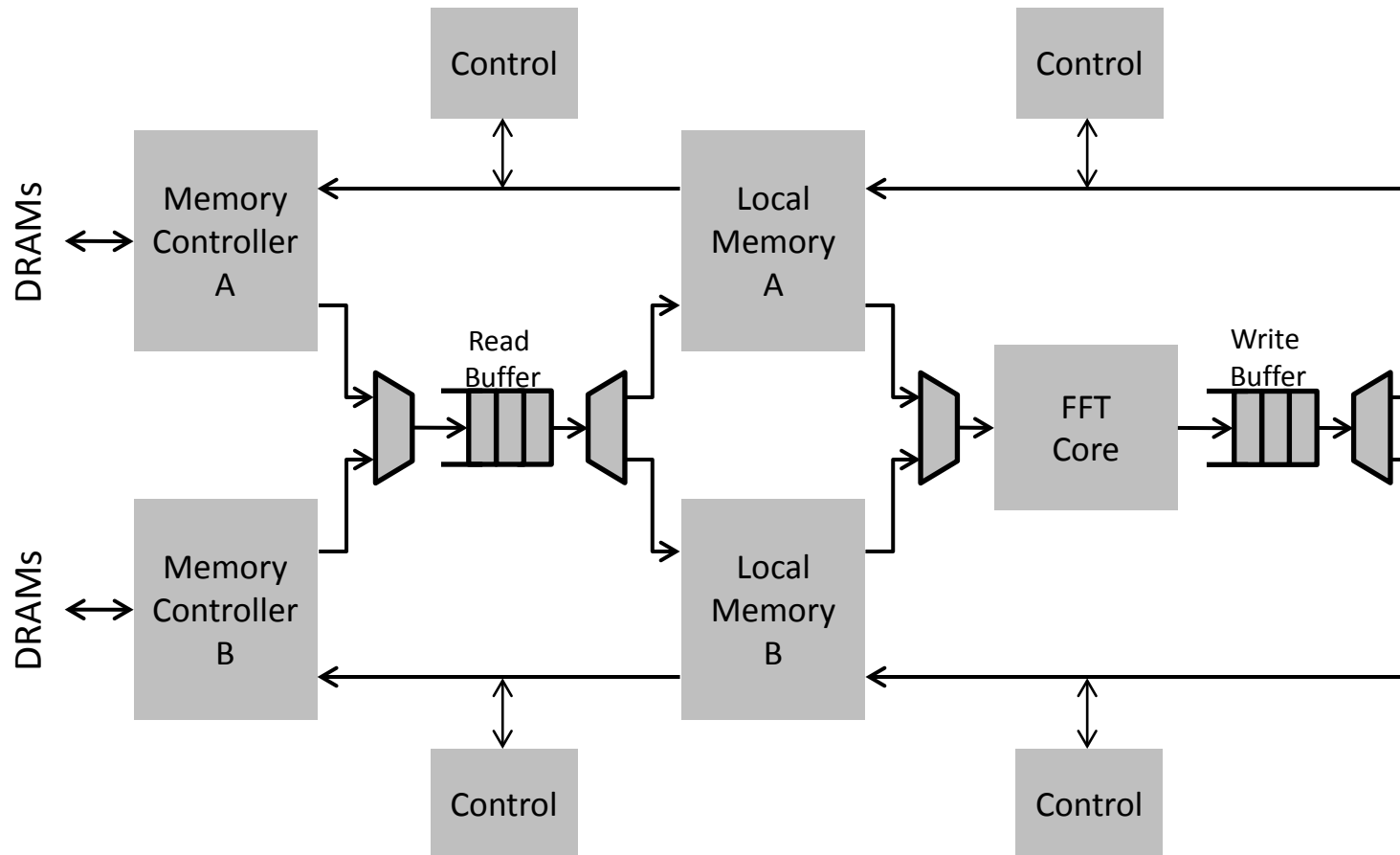
$$\text{DFT}_{n \times n} = \overbrace{(\text{DFT}_n \otimes \text{I}_n)}^{\text{Column Stage}} \overbrace{(\text{I}_n \otimes \text{DFT}_n)}^{\text{Row Stage}} \quad \text{Row-column algorithm}$$

$$= \prod_{i=0}^1 \left(\text{L}_n^{n^2} \underbrace{(\text{I}_n \otimes \text{DFT}_n)}_{\text{Symmetric algorithm}} \text{I}_n^{n^2} \right)$$

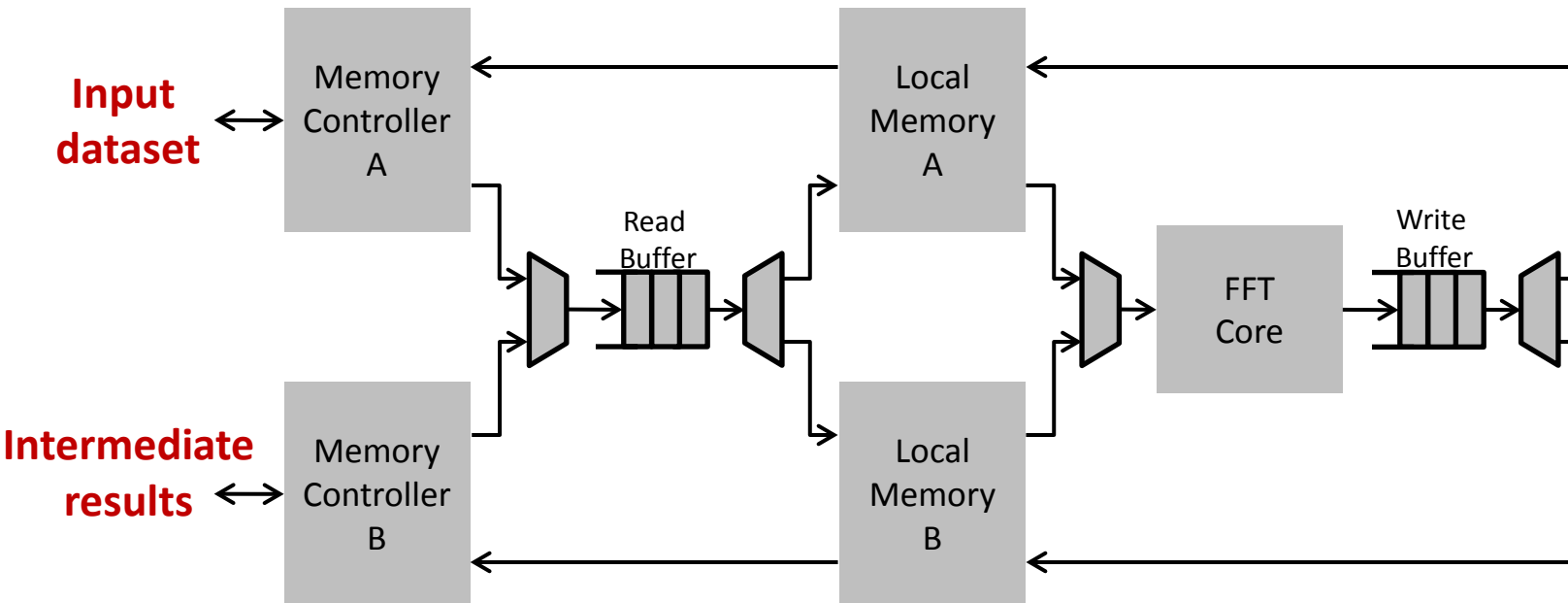
$$= \prod_{i=0}^1 \left(\underbrace{(\text{L}_n^{n^2/k} \otimes \text{I}_k)}_{\text{Write tiles column-wise}} \underbrace{(\text{I}_{n/k} \otimes \text{L}_n^{nk})}_{\text{Transpose and re-tile on-chip}} \underbrace{(\text{I}_{n/k} \otimes (\text{I}_k \otimes \text{DFT}_n))}_{\text{FFT processing}} \underbrace{(\text{I}_{n/k} \otimes (\text{L}_k^n \otimes \text{I}_k))}_{\text{Linearize on-chip}} \underbrace{(\text{I}_{n/k} \otimes (\text{L}_{n/k}^n \otimes \text{I}_k))}_{\text{Read tiles row-wise}} \right)$$

Symmetric algorithm with tiling

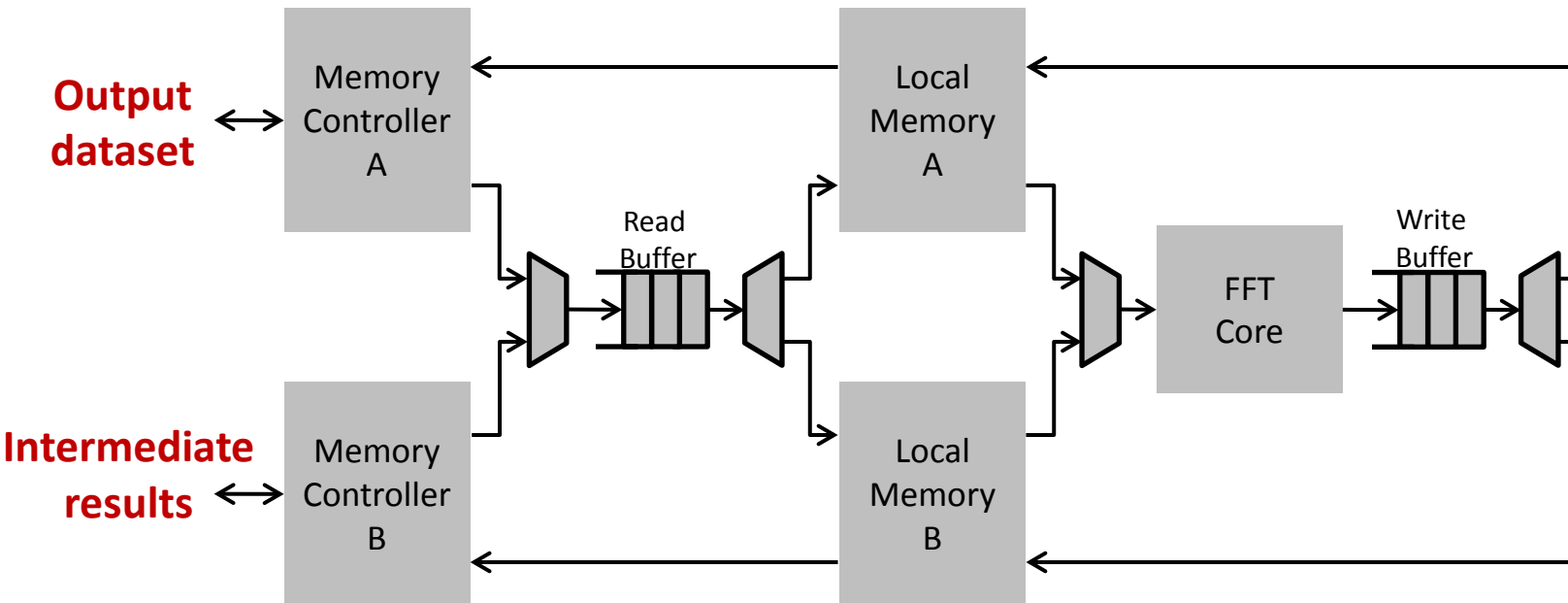
23



Overall Architecture



Overall Architecture



Read & Write Buffers

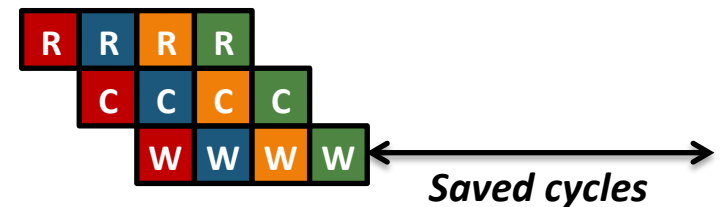
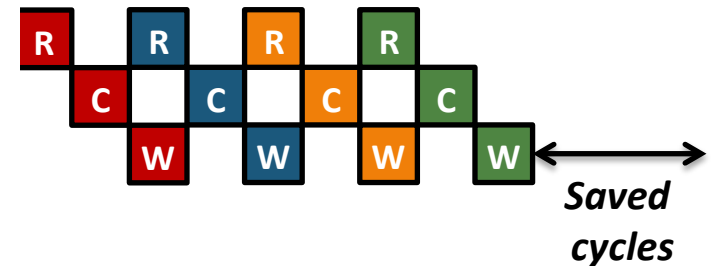
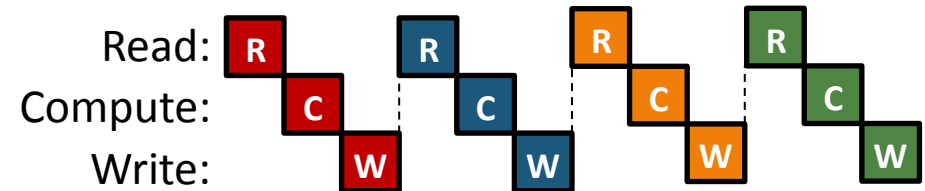
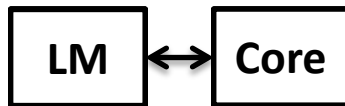
- Ensuring continuous flow of data to and from the core
- Compensates the bubbles in the data-flow due to DRAM



- Also used to monitor imbalances between read, write and compute bandwidth and to throttle up/down memory controllers if necessary

Maximizing the Throughput

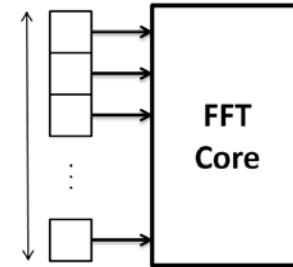
- Single memory



Implementation Details

■ FFT Core

- Generated using Spiral FFT IP generator
- Fully pipelined, scalable throughput



■ Local Memories

- Constructed out of embedded block RAMs on FPGA
- Double-buffered to overlap computation and data accesses
- Multi-banked, supports streaming permutations

P. Milder et al, "Automatic Generation of Streaming Datapaths for Arbitrary Fixed Permutations," DATE 2009

■ Memory controllers

- Altera High Performance Controller (HPC)
- Achieved above 90% of theoretical peak with efficient access patterns

Outline

- Motivation
- Background
- Existing Implementations
- Our DRAM Optimized Design
- **Results**
- Conclusion

Evaluation - I

- Summary of three reference works

FFT Size	FPGA	Off-chip Memory	Precision	Runtime (ms)	Source
1K-by-1K	Virtex-E	4xSRAM	16 (fixed)	62.5	Baseline
1K-by-1K	Virtex-5	1xDDR2-400	32 (single)	102.6	DRAM-opt
1K-by-1K	Stratix IV	2xDDR2-800	64 (double)	6.1	Ours

Evaluation - II

- **Platforms:**

- FPGA: Altera DE4 development board featuring Stratix IV FPGA
- CPU: Intel Core i7 960 Quad Core
- GPU: Nvidia GeForce GTX 480

- **Application:**

- 2D-FFT of sizes up to 2,048-by-2,048
- Double-precision, complex values

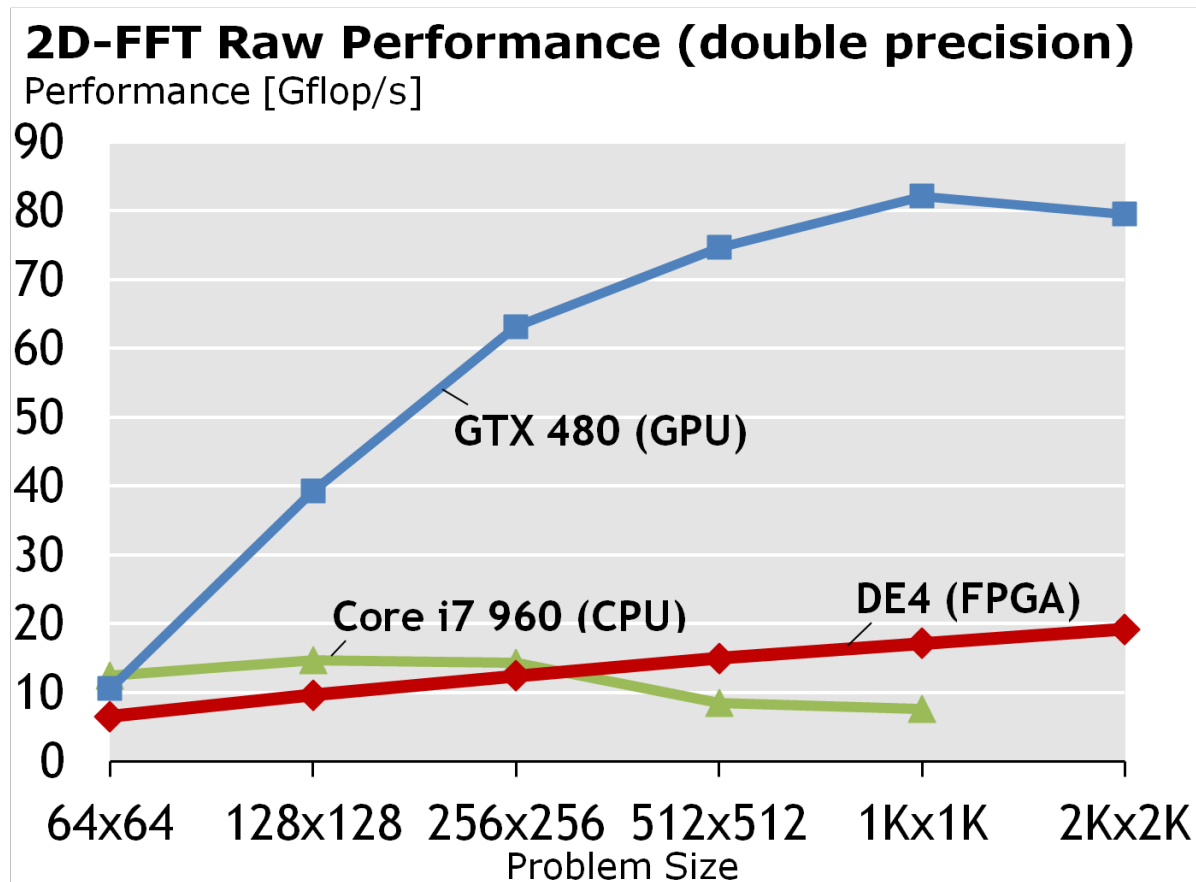
- **Metrics:**

- Raw performance (Gflop/s)
- Bandwidth efficiency $[(\text{Gflop/s})/(\text{Gbyte/s})]$
- Power efficiency $[(\text{Gflop/s})/\text{Watt}]$

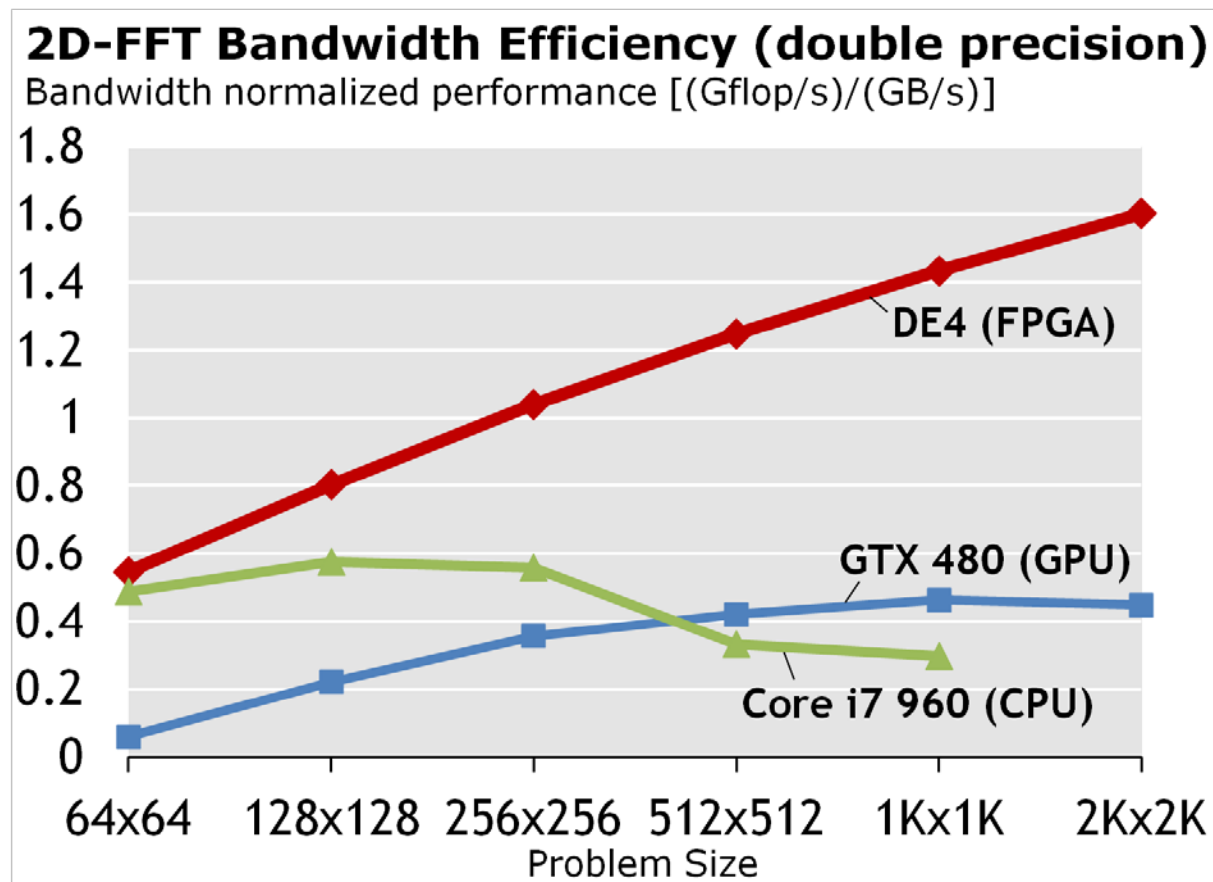
Platform Comparison

	Core i7 960	GTX 480	Stratix IV (DE4) EP4SGX530
DRAM Type	DDR3	GDDR5	DDR2
# of Memory Channels	3	6	2
Memory BW (GB/s)	25.6	177.4	12
On-chip Memory (MB)	8	1.69	2.53
Proc. Freq. (MHZ)	3,200	1,401	200
# of PEs (Cores)	4	480	N/A
Application Infrastructure	Spiral	CUDA 4.0	Spiral/Verilog

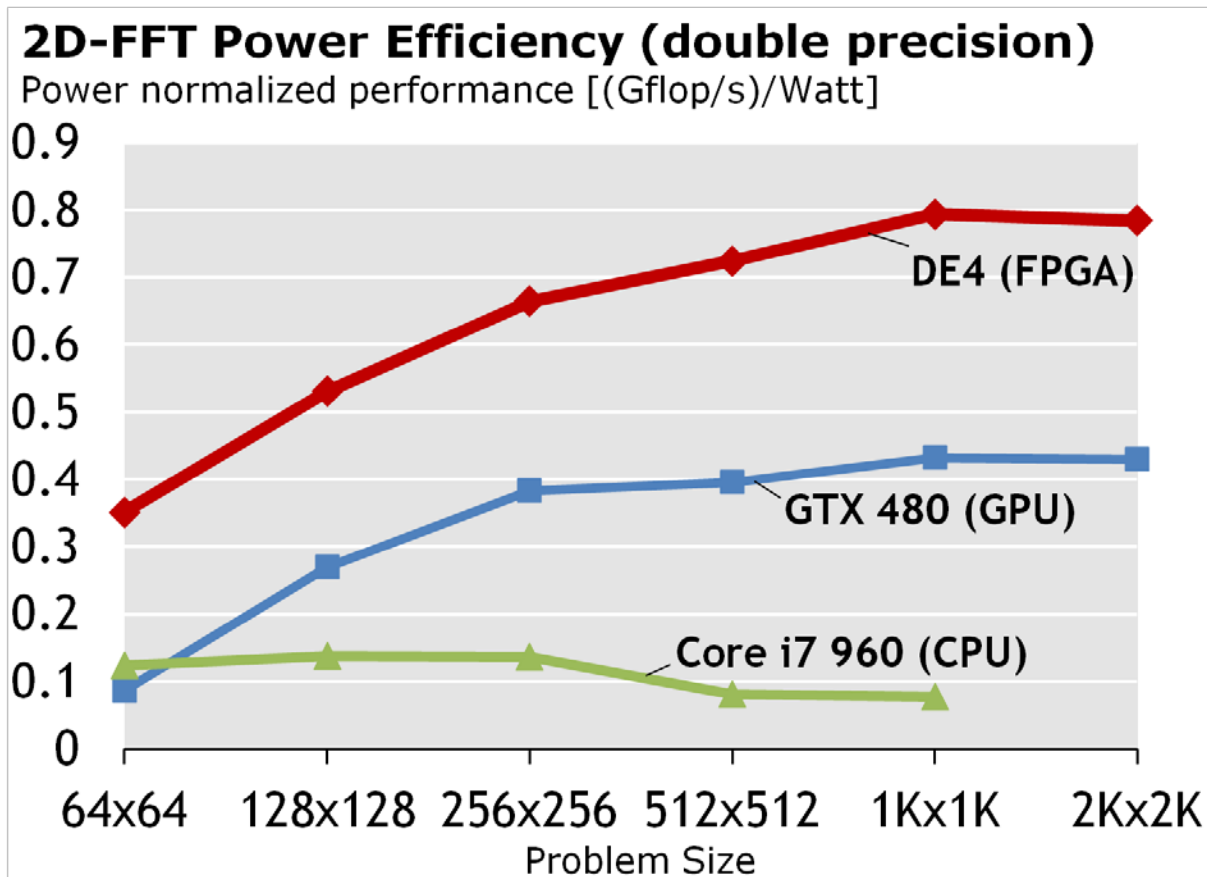
Evaluation - II



Evaluation - II



Evaluation - II



Summary & Conclusion

- **Memory bandwidth is a precious resource**
 - Becomes performance bottleneck
 - Efficient use of memory bandwidth enables high performance
 - Has to become first class design consideration
- **Algorithm and architecture for large size 2D-FFT**
 - Memory bandwidth is efficiently utilized
 - Computation throughput is maximized
 - Scalable and parameterized design
- **Outperforms existing FPGA implementations**
 - More bandwidth and power efficient than GPUs and CPUs

QUESTIONS?