Algorithm and Architecture Optimizations for Large Size Two Dimensional Discrete Fourier Transform

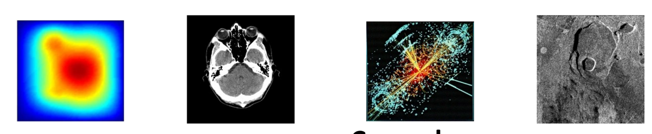
Berkin Akin, Peter Milder, Franz Franchetti and James Hoe
Carnegie Mellon Unieristy, Pittsburgh PA USA
{bakin, pam, franzf, jhoe}@ece.cmu.edu

SPIRAL www.spiral.net
Carnegie Mellon
Electrical & Computer ENGINEERING

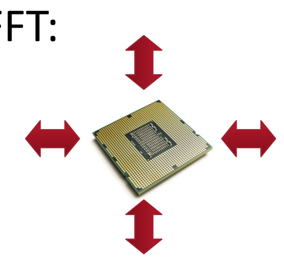
Overview

Large size 2D Fast Fourier Transform

- Used in image processing, scientific computing



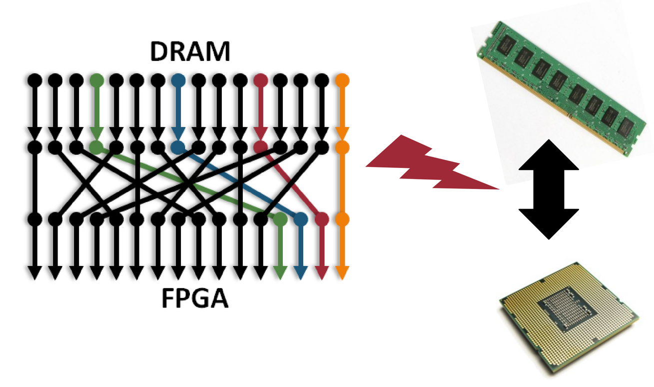
- Typical datasets are ***large*** and ***high precision***!
  - *e.g.* 2K-by-2K double precision 2D-FFT:
    - Input dataset: **64 MB**
    - # of operations: **~461.4 Mflop**



- Does not fit on-chip
- Stored ***off-chip***

Memory access pattern and achieved bandwidth

- Have large ***strided*** DRAM access pattern
- Does not exploit DRAM ***row-buffer locality***

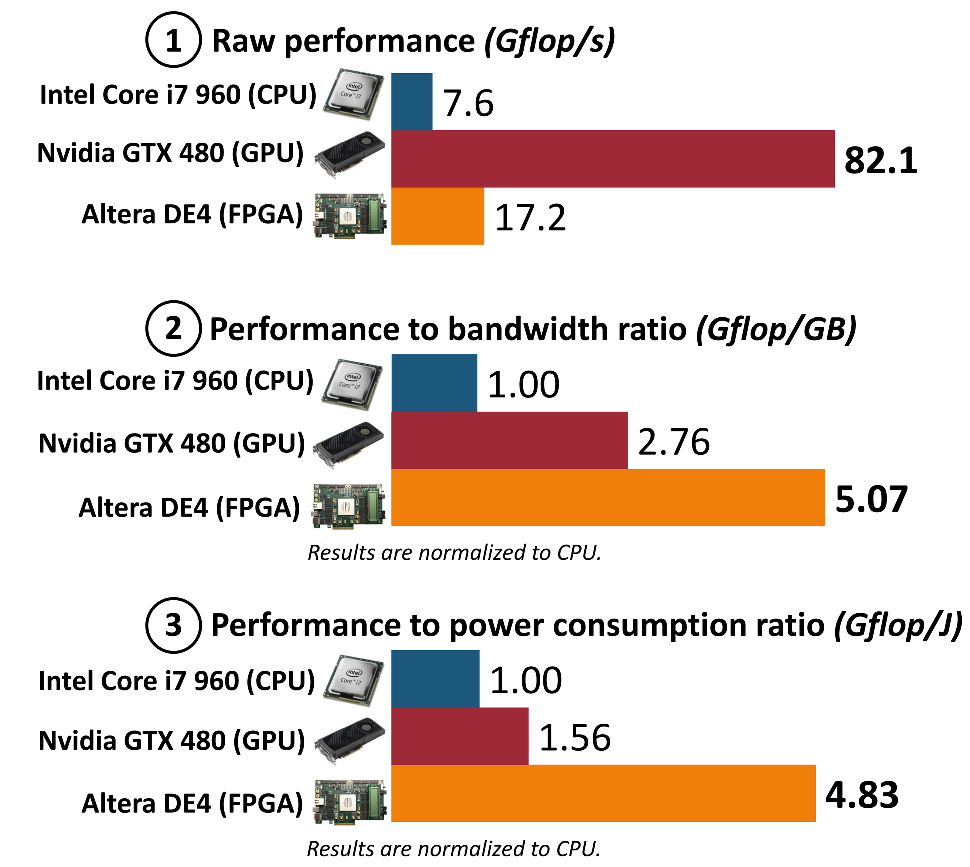


- Results in ***low*** memory bandwidth utilization!
- Memory bandwidth becomes ***bottleneck*** for achieving high performance

- Effective ***bandwidth orchestration*** is required for:

1 Performance 2 Bandwidth Efficiency 3 Power Efficiency

1024-by-1024 double precision 2D-FFT



Background

DFT is matrix-vector multiplication

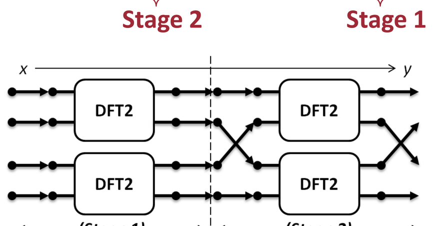
$$y = \mathrm{DFT}_n \cdot x, \quad \mathrm{DFT}_n = [e^{-2\pi i k\ell/n}]_{0\le k,\ell<n}$$

FFT algorithm is factorization of DFT matrix

$$\mathrm{DFT}_4 = (\underbrace{\mathrm{DFT}_2 \otimes \mathrm{I}_2}_{\text{tensor}})\,\underbrace{\mathrm{T}_2^4}_{\text{twiddle factors}}(\mathrm{I}_2 \otimes \mathrm{DFT}_2)\,\underbrace{\mathrm{L}_2^4}_{\text{permutation}}$$

2D-FFT operates on 2D data, *e.g. images*

$$\mathrm{DFT}_{2\times 2} = \underbrace{(\mathrm{DFT}_2 \otimes \mathrm{I}_2)}_{\text{Stage 2}}\underbrace{(\mathrm{I}_2 \otimes \mathrm{DFT}_2)}_{\text{Stage 1}}$$

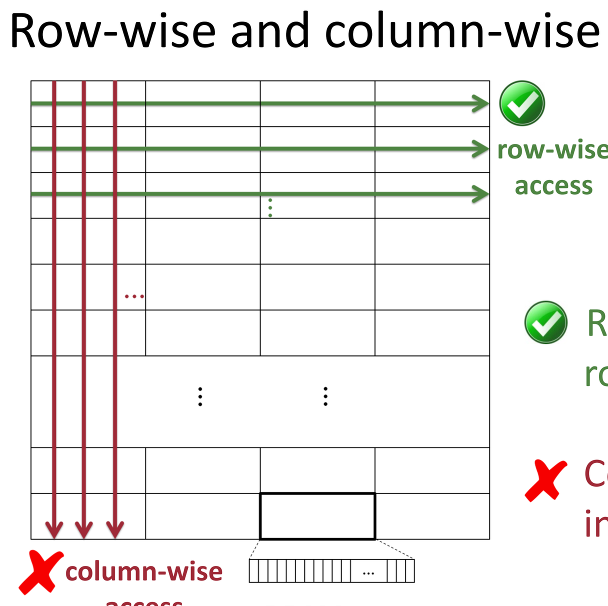


2D-FFT algorithms

- Row column algorithm:

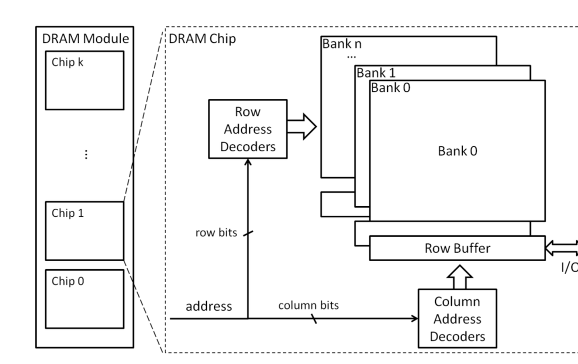
$$\mathrm{DFT}_{n\times n} = (\mathrm{DFT}_n \otimes \mathrm{I}_n)(\mathrm{I}_n \otimes \mathrm{DFT}_n)$$

Row-wise and column-wise accesses!

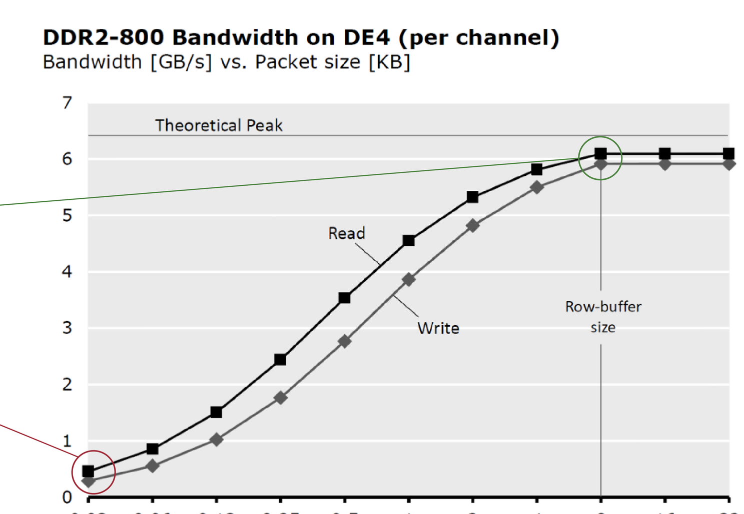


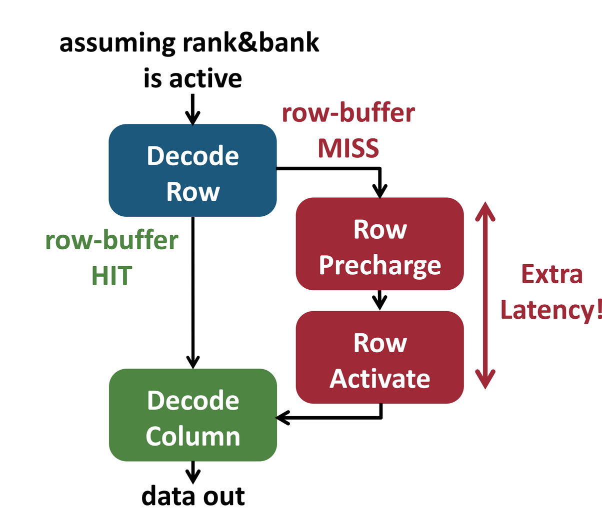
- Row-wise access exploits row-buffer locality
- Column-wise access results in row-buffer misses

DRAM operation



- Need to make use of every row touched to ***maximize bandwidth***
- Large strides result in ***small packets*** of transferred data





Solution: Algorithm and Architecture

Restructured algorithm

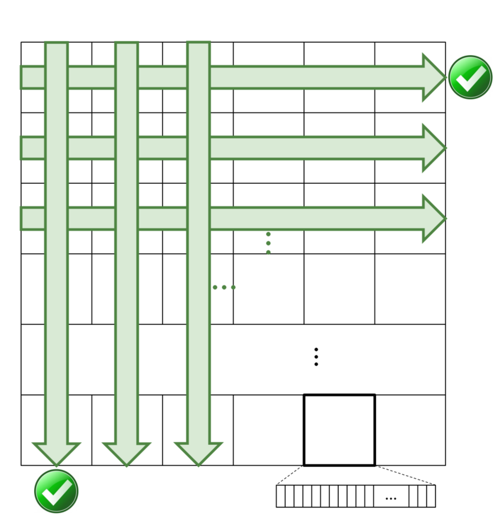
- ***Linear data mapping*** in DRAM causes row and column-wise accesses
- Use ***2D tiled*** data mapping where each tile is mapped to a DRAM row
- ***Restructure*** the algorithm given 2D data mapping

$$\mathrm{DFT}_{n\times n} = \prod_{i=0}^{1}\left(\mathrm{L}_n^{n^2}(\mathrm{I}_n \otimes \mathrm{DFT}_n)\,\mathrm{I}_{n^2}\right)$$

hardware aware manipulation

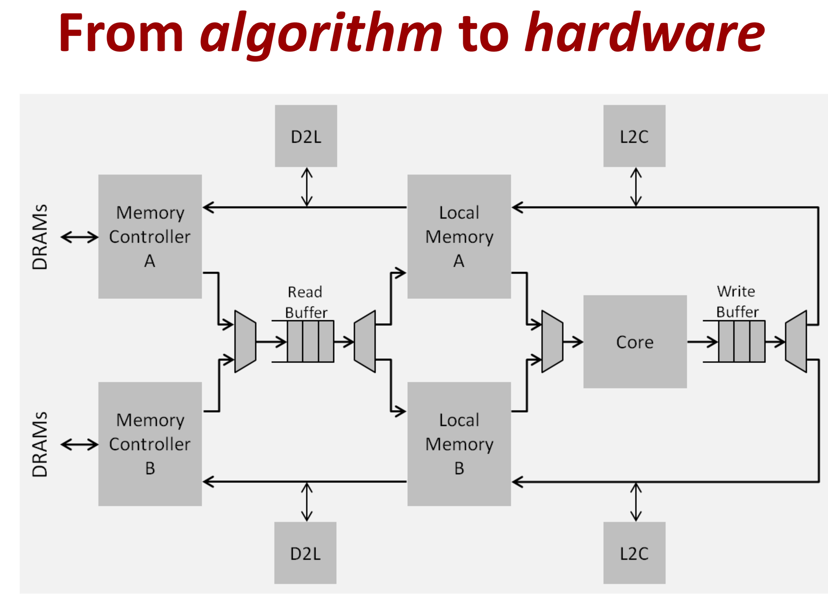
$$\mathrm{DFT}_{n\times n} = \prod_{i=0}^{1}\Big(\underbrace{(\mathrm{L}_n^{n^2/k}\otimes \mathrm{I}_k)}_{\text{write tiles into DRAM}}\underbrace{(\mathrm{I}_{n/k}\otimes \mathrm{L}_n^{nk})}_{\text{on-chip permutation}}\underbrace{(\mathrm{I}_{n/k}\otimes(\mathrm{I}_k\otimes \mathrm{DFT}_n))}_{\text{DFT computation}}\underbrace{(\mathrm{I}_{n/k}\otimes(\mathrm{L}_k^n\otimes \mathrm{I}_k))}_{\text{on-chip permutation}}\underbrace{(\mathrm{I}_{n/k}\otimes(\mathrm{L}_{n/k}^n\otimes \mathrm{I}_k))}_{\text{read tiles from DRAM}}\Big)$$

DRAM write address generation | Local Memory address generation | Streaming FFT Core | Local Memory address generation | DRAM read address generation

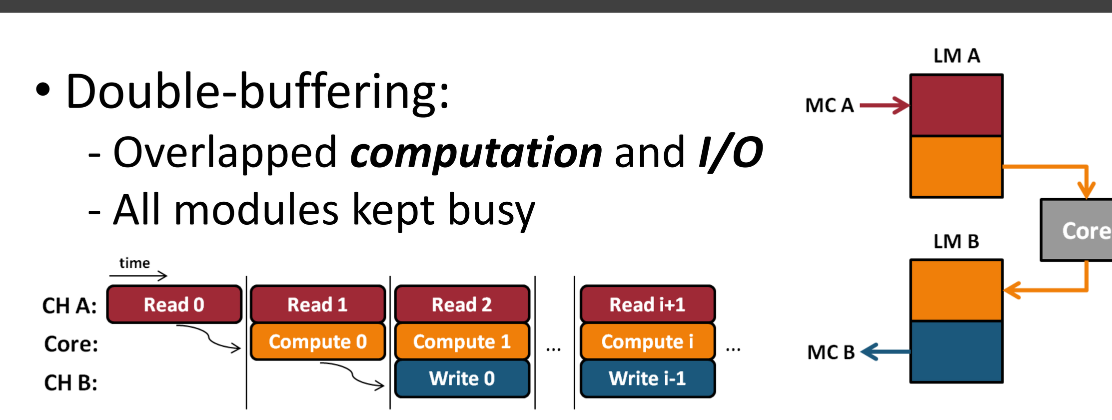


- Data is accessed as ***tiles***, not row and column-wise
- Row-buffer misses are minimized!

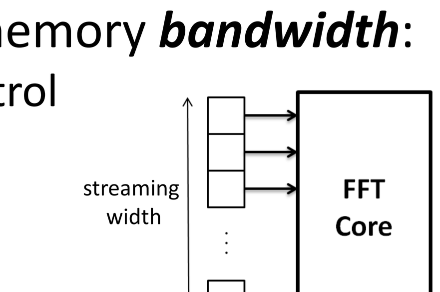
From *algorithm* to *hardware*



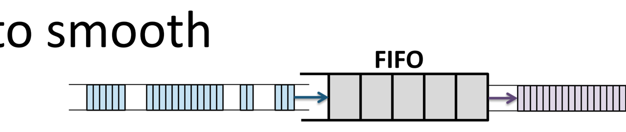
- Double-buffering:
  - Overlapped ***computation*** and ***I/O***
  - All modules kept busy



- Matching ***throughput*** to memory ***bandwidth***:
  - Achieved via fine-grain control over datapath ***parallelism***
  - Results in ***balanced*** design



- Ensuring ***continuous dataflow***:
  - Buffers are used to smooth the flow of data.



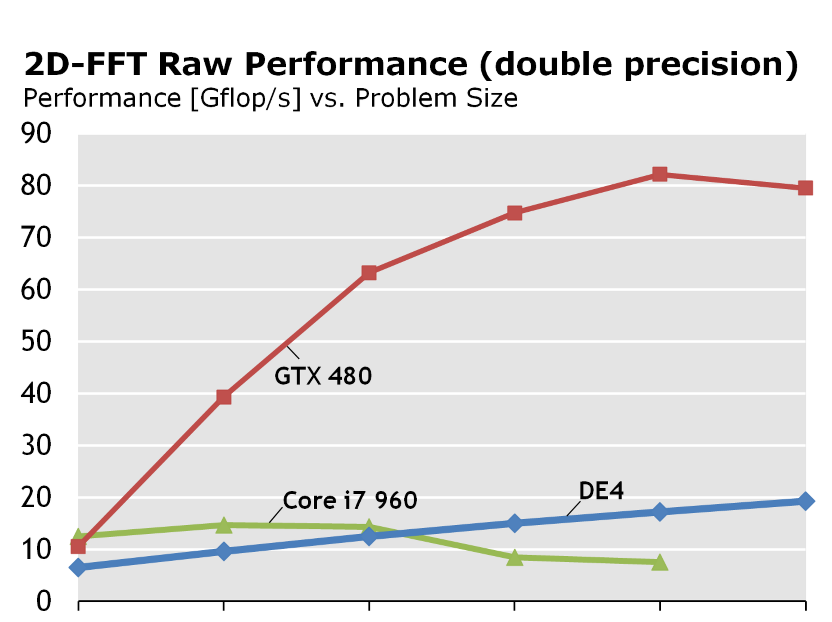
Evaluation

Target ***application***:
- Double precision complex 2D-FFT
- Data sizes up to 2,048-by-2,048

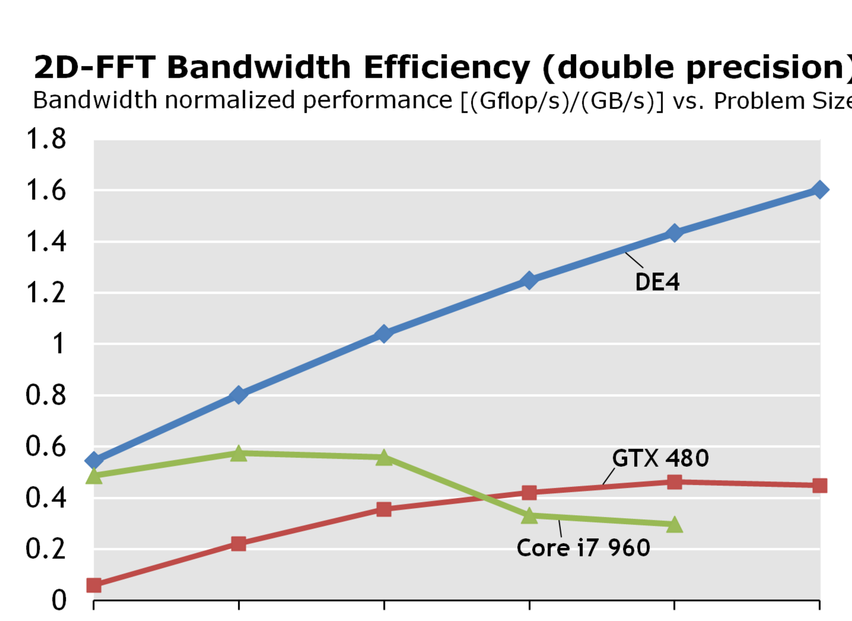
Target ***platforms***:

	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 Cores	4	480	N/A
Technology Node (nm)	45	40	40
Application Infrastructure	Spiral	CUDA 4.0	Spiral/Verilog

Raw performance:

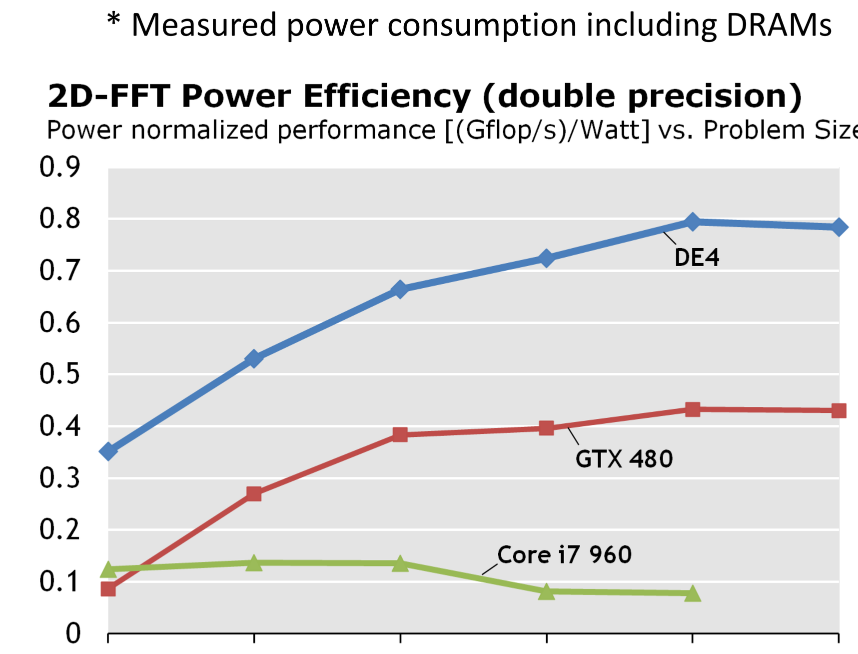


Bandwidth Efficiency:



Power* Efficiency:

* Measured power consumption including DRAMs



The authors acknowledge the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.