SPIRAL DSP Transform Compiler:

Application Specific Hardware Synthesis

Peter A. Milder[1] (peter.milder@stonybrook.edu)
Franz Franchetti, James C. Hoe, and Markus Pueschel[2]

Department of ECE
Carnegie Mellon University

now with [1]SUNY Stonybrook and [2]ETH

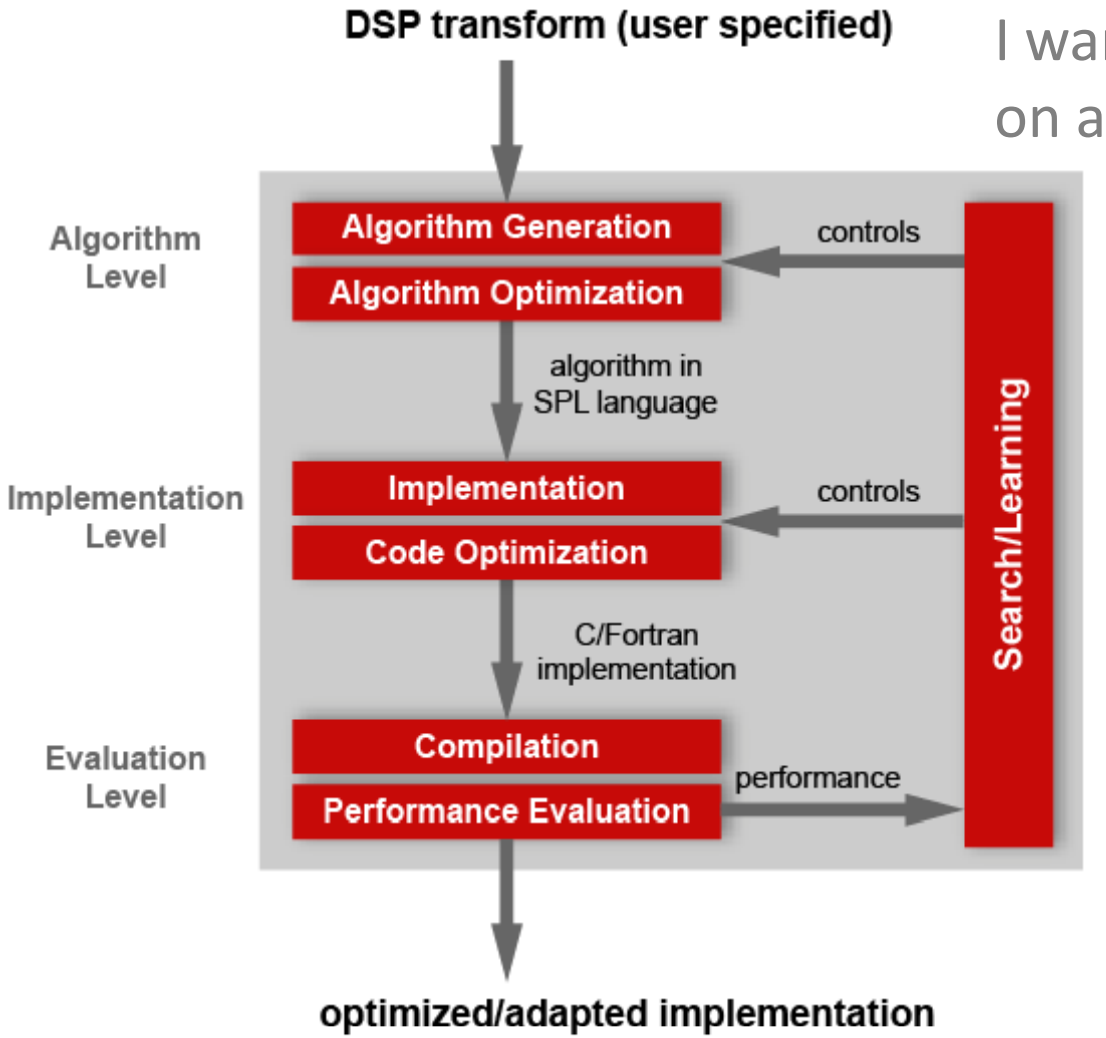
The SPIRAL Project

- High performance implementations of linear DSP transforms (DFT, DCT, DWT, filters, etc) are an important class of design problems
- Hand design and tuning is tricky and expensive
  - needs both math and implementation knowledge
  - time-consuming and tedious
  - needs to repeat effort for every new context
- SPIRAL research goal: A flexible push-button design generator that produces SW & HW implementations comparable with expert hand design

Why we can do better than hand design

- SPIRAL is only focused on linear DSP transforms
- These transforms are highly structured, highly regular and very well understood mathematically
- Algorithmic implementations of a transform can be enumerated following a known set of rules
- For a given objective function and mapping target, a computer generates a solution at least as good as the best human effort—by trying enough implementations

SPIRAL Framework



I want a DFT of size 1024 on a {Xilinx, P4, Cell....}

SPIRAL automation starts here

where most tools begin automating the problem

Principle 1: **Domain knowledge in the system**

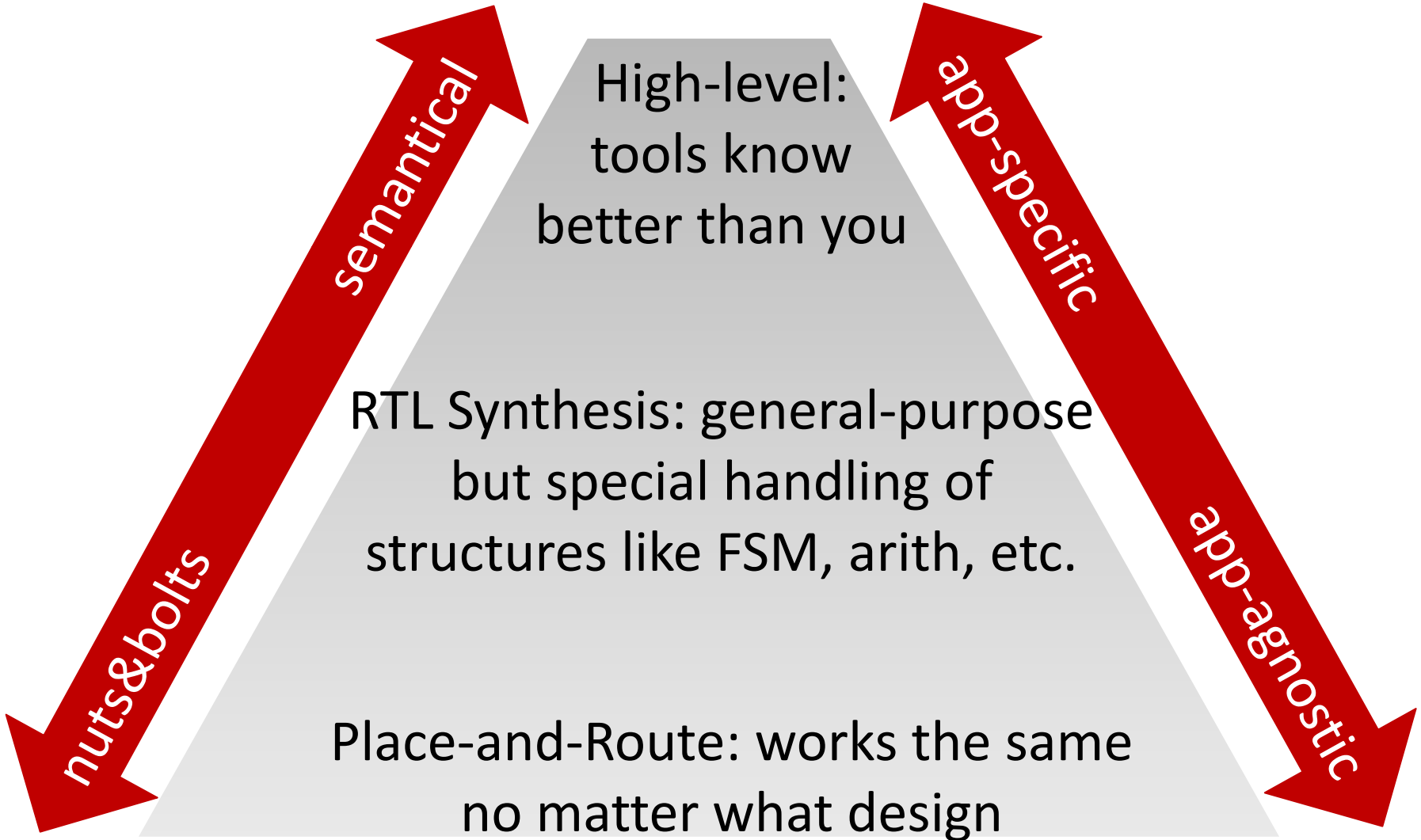
Principle 2: **Optimization at a high level of abstraction**

www.spiral.net/hardware/dftgen.html

parameter	value	range	explanation
Problem specification			
transform size	64	4–32768	Number of samples (?)
direction	forward		forward or inverse DFT (?)
data type	fixed point		fixed or floating point (?)
	16 bits	4–32 bits	fixed point precision (?)
	unscaled		scaling mode (?)
Parameters controlling implementation			
architecture	fully streaming		iterative or fully streaming (?)
radix	2	2, 4, 8, 16, 32, 64	size of DFT basic block (?)
streaming width	2	2–64	number of complex words per cycle (?)
data ordering	natural in / natural out		natural or digit-reversed data order (?)
BRAM budget	1000		maximum # of BRAMs to utilize (-1 for no limit) (?)
Permutation method	JACM'09 [3] (patented)		Please click for more information

Generate Verilog Reset

High-Level, Quality, and Specialization



Outline

- SPIRAL Formula Framework
- SPIRAL for HW FFT cores
- SPIRAL for HW FFT “un”-core

Linear Transforms

- Linear transform is a matrix-vector multiplication
  - computing by definition takes $O(N^2)$ operations
  - the matrix has structure

- E.g. discrete Fourier transform: $y = DFT_N \cdot x$

$$\begin{bmatrix} y_0 \\ y_1 \\ \cdot \\ y_j \\ \cdot \\ \cdot \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} & k \rightarrow 0 \,..\, N\text{-}1 & \\ j \leftarrow 0 \,..\, N\text{-}1 & e^{-i 2\pi\, jk/N} & \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ \cdot \\ x_k \\ \cdot \\ \cdot \\ x_{N-1} \end{bmatrix}$$

"Fast" Algorithms

- a "fast" algorithm factors the matrix into a sequence of structured, sparse matrices

  cheaper sparse multiplies $\Rightarrow O(N \log(N))$ *operations*

- E.g. Cooley-Tukey Factorization of DFT_4

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} = \begin{bmatrix} 1 & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & 1 \\ 1 & \cdot & -1 & \cdot \\ \cdot & 1 & \cdot & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & i \end{bmatrix} \begin{bmatrix} 1 & 1 & \cdot & \cdot \\ 1 & -1 & \cdot & \cdot \\ \cdot & \cdot & 1 & 1 \\ \cdot & \cdot & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot \\ \cdot & 1 & \cdot & \cdot \\ \cdot & \cdot & \cdot & 1 \end{bmatrix}$$

- Matrix formula representation

$$DFT_4 = \left(DFT_2 \otimes I_2\right) D_2^4 \left(I_2 \otimes DFT_2\right) L_2^4$$

Factorization Rules

E.g. Cooley-Tukey

$$DFT_{n\cdot m} = \left(DFT_n \otimes I_m\right) D_n^{n\cdot m} \left(I_n \otimes DFT_m\right) L_n^{n\cdot m}$$

- DFT_2 is $\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
- D is a diagonal matrix of twiddle factors
- L is a stride permutation matrix
- $A \otimes B = [a_{j,k} B]$ is the tensor (or kronecker) product

e.g., $I_n \otimes B \Rightarrow \begin{bmatrix} B & & & 0 \\ & B & & \\ & & \ddots & \\ 0 & & & B \end{bmatrix}$

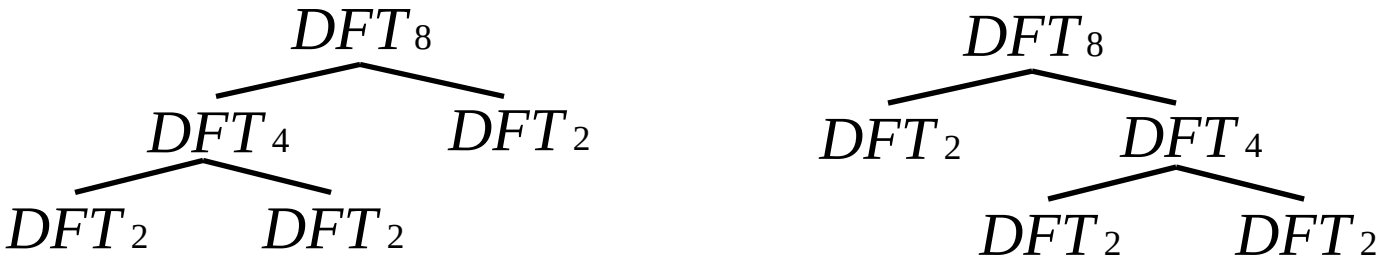
$A \otimes I_n \Rightarrow \begin{bmatrix} a_{0,0} & 0 & a_{0,1} & 0 & \cdots \\ 0 & a_{0,0} & 0 & a_{0,1} & \\ a_{1,0} & 0 & a_{1,1} & 0 & \\ 0 & a_{1,0} & 0 & a_{1,1} & \\ \vdots & & & & \ddots \end{bmatrix}$

"Fast" Fourier Transform Algorithms

- Recursively factorize by the Cooley-Tukey rule until only leaf cases remain (e.g. DFT_r for radix-r)

$$DFT_8 = (DFT_2 \otimes I_4) D_2^8 (I_2 \otimes DFT_4) L_2^8$$
$$= (DFT_2 \otimes I_4) D_2^8 (I_2 \otimes ((DFT_2 \otimes I_2) D_2^4 (I_2 \otimes DFT_2) L_2^4)) L_2^8$$

- Exponential number of alternatives



- Each ruletree corresponds a different algorithm
- All cost $O(N\ log(N))$

A System of Transforms and Rules

$$DCT_2^{(II)} \rightarrow diag\left(1, 1/\sqrt{2}\right) \cdot F_2$$

$$DCT_n^{(II)} \rightarrow P \cdot \left(DCT_{n/2}^{(II)} \oplus DCT_{n/2}^{(IV)}\right) \cdot \left(I_{n/2} \otimes F_2\right)^Q$$

$$DCT_n^{(IV)} \rightarrow S \cdot DCT_n^{(II)} \cdot D$$

$$DCT_n^{(IV)} \rightarrow M_1 \cdots M_r$$

$$DFT_n \rightarrow B \cdot (DCT_{n/2}^{(I)} \oplus DST_{n/2}^{(I)}) \cdot C$$

50+ transforms

150+ rules

$$DFT_{nm} \rightarrow \left(DFT_n \otimes I_m\right) \cdot D \cdot \left(I_n \otimes DFT_m\right) \cdot P$$

$$F_n(h) \rightarrow (I_{n/d} \otimes^k I_{d+k}) \cdot (I_{n/d} \otimes F_d(h))$$

$$F_n(h) \rightarrow Circ(\bar{h}) \cdot E$$

$$DWT_n(W) \rightarrow (DWT_{n/2}(W) \oplus I_{n/2}) \cdot P \cdot (I_{n/2} \otimes_k W) \cdot E$$

$$WHT_{2^n} \rightarrow \prod_{i=1}^{n} (I_{2^{n_1+\ldots+n_{i-1}}} \otimes WHT_{2^{n_i}} \otimes I_{2^{n_{i+1}+\ldots+n_t}})$$

• • • • •

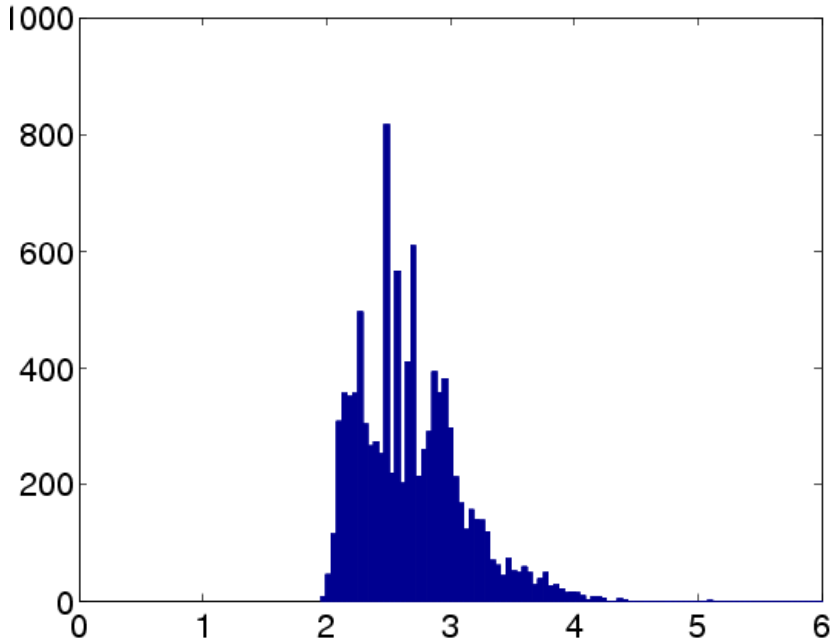
Algorithmic Design Space

size	# of DFT	# of DCT-IV
2	1	1
4	6	10
8	40	126
16	296	31242
32	27744	1924443362
64	162570361280	73438151216313542 42
128	$\sim 1.01 \times 10^{27}$	$\sim 1.07 \times 10^{38}$
256	$\sim 2.31 \times 10^{61}$	$\sim 2.30 \times 10^{76}$
512	$\sim 2.86 \times 10^{133}$	$\sim 1.06 \times 10^{153}$

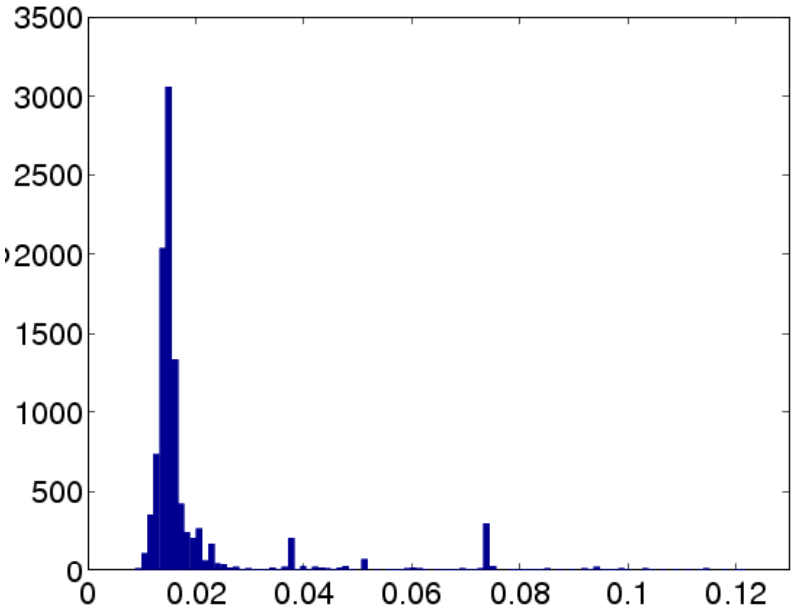
➡ Different characteristics: data flow, numerical stability, operation ordering, working set size, datapath regularity

Design Space: SW DCT 32 on P4

Histogram of 10,000 randomly selected algorithms



histogram by runtime
(P4, 3.2 GHz)



histogram by num. accuracy

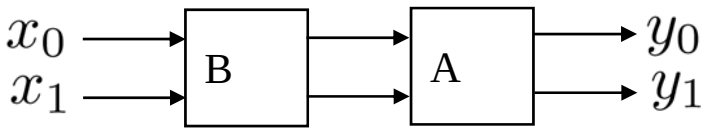
Outline

- SPIRAL Formula Framework
- SPIRAL for HW FFT cores
- SPIRAL for HW FFT “un”-core

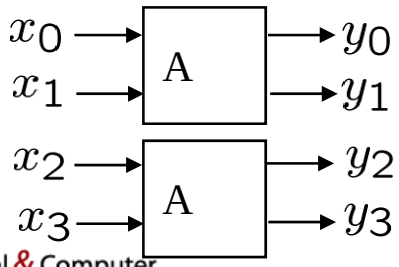
Formula to HW (Combinational)

- Given $y = M \cdot x$ where M is:
  - $M = A \cdot B$ apply B, then A
  - M is a permutation permute x
  - $M = I_n \otimes A$ apply A, n times in parallel
  - M is a diagonal scale x

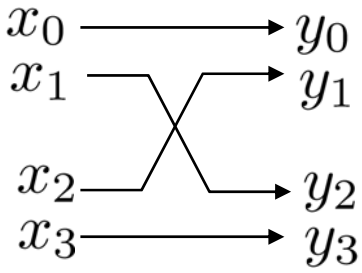
$y = (A \cdot B) \cdot x$



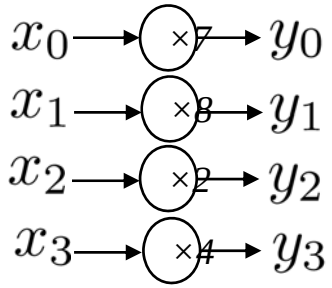
$y = (I_2 \otimes A) \cdot x$



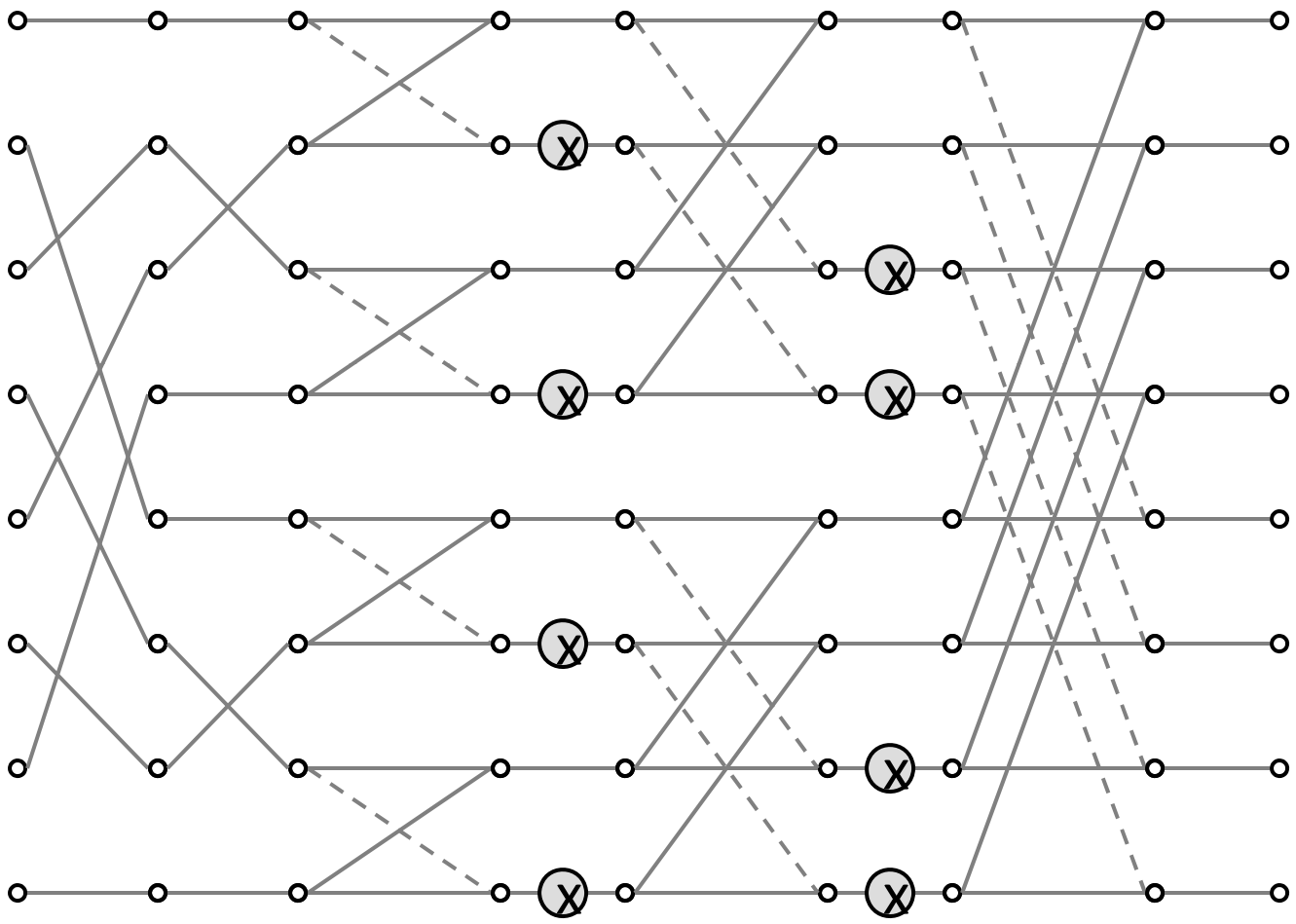
$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot x$$



$$y = \begin{bmatrix} 7 & 0 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix} \cdot x$$



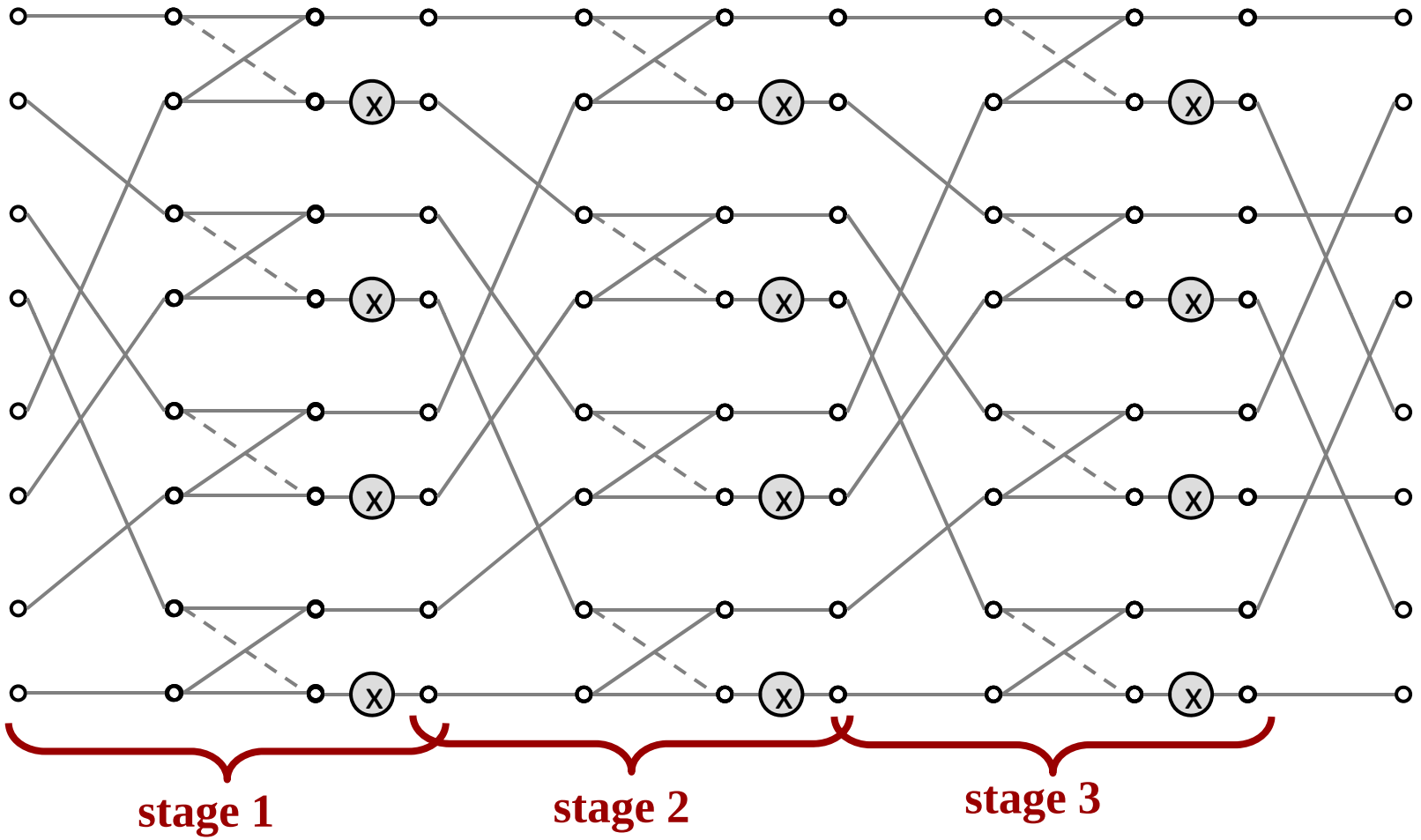
DFT_8 Datapath Example



$$DFT_8 = \left(DFT_2 \otimes I_4\right) D_2^8 \left(I_2 \otimes \left(\left(DFT_2 \otimes I_2\right) D_2^4 \left(I_2 \otimes DFT_2\right) L_2^4\right)\right) L_2^8$$

(formula is applied from right to left)

Pease DFT_8 Example



$$\mathrm{DFT}_8 = R_8 \cdot T_2(I_4 \otimes F_2)L_4^8 \cdot T_1(I_4 \otimes F_2)L_4^8 \cdot T_0(I_4 \otimes F_2)L_4^8$$

How about good HW?

- Matrix formulas have a natural mapping to dataflow and hence combinational datapath
- However, real hardware designs must fit a given resource constraint

  $\Rightarrow$ *sequential datapath that reuse available HW*

  – identify repeated kernels

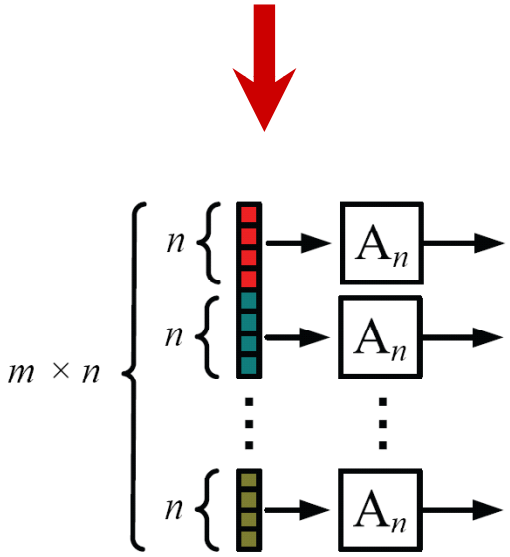
  – instantiate kernels under resource constraints

  – schedule computation to reuse instantiated kernels

  We want to do the analysis and mapping at formula level, with high-level algorithm knowledge

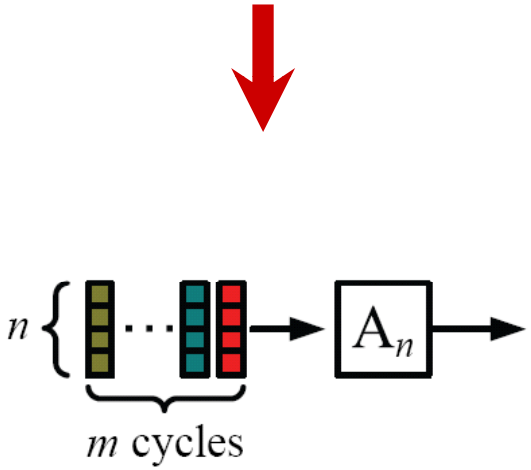
Tensor as Streaming Parallelism

$I_m \otimes A_n$



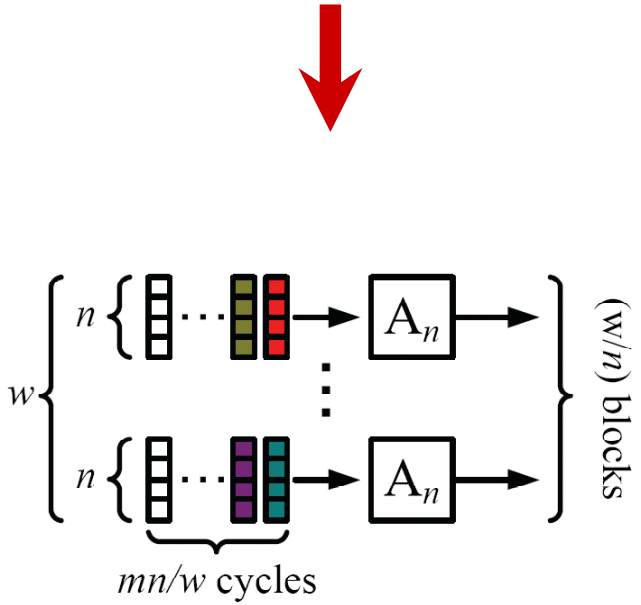
fully parallel

$I_m \otimes^{\mathbf{sr}} A_n$



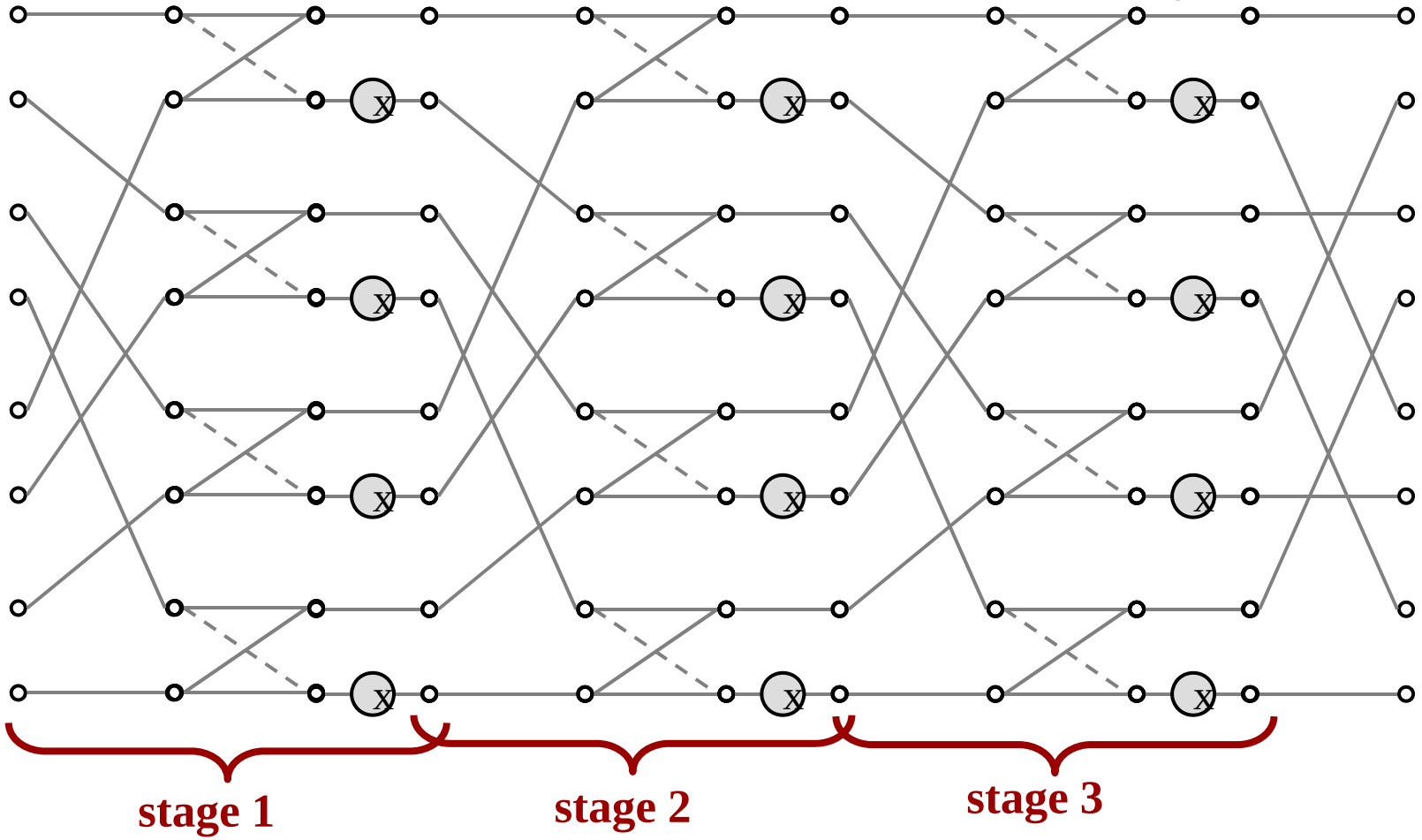
fully streamed

$I_{mn/w} \otimes^{\mathbf{sr}} (I_{w/n} \otimes A_n)$



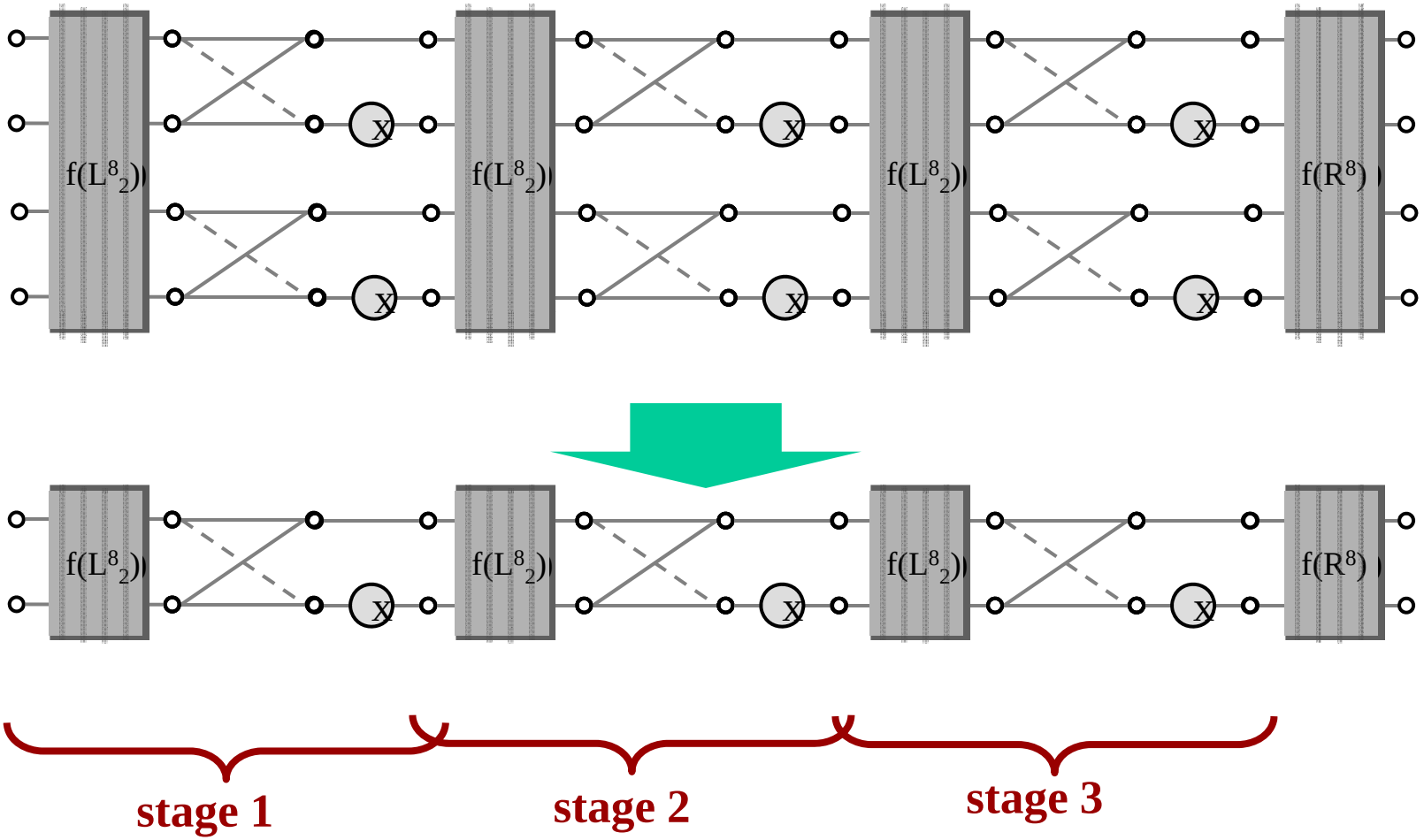
partially streamed

Pease DFT Example: DFT_8



$$\mathrm{DFT}_8 = R_8 \cdot T_2(I_4 \otimes F_2)L_4^8 \cdot T_1(I_4 \otimes F_2)L_4^8 \cdot T_0(I_4 \otimes F_2)L_4^8$$

Pease DFT Example: DFT_8 Streaming



$$\mathrm{DFT}_8 = R_8 \cdot T_2(I_4 \otimes F_2)L_4^8 \cdot T_1(I_4 \otimes F_2)L_4^8 \cdot T_0(I_4 \otimes F_2)L_4^8$$

Regular Structure for HW

- Simple regular structure embodied in Pease FFT

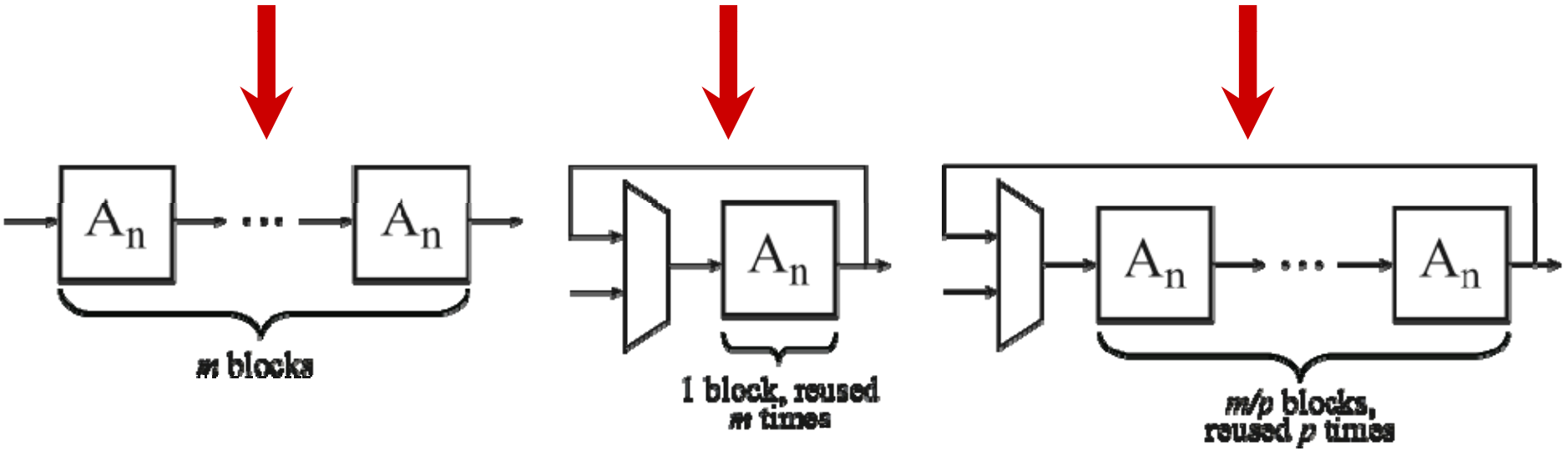
$$\mathrm{DFT}_{2^k} = R_{2^k} \left(\underbrace{\prod_{i=0}^{k-1}}_{} T_i \underbrace{(I_{2^{k-1}} \otimes F_2)}_{} L^{2^k}_{2^{k-1}} \right)$$

- Example:

$$\mathrm{DFT}_8 = R_8 \left(T_0(I_4 \otimes F_2)L^8_4\right) \left(T_1(I_4 \otimes F_2)L^8_4\right) \left(T_2(I_4 \otimes F_2)L^8_4\right)$$

Formally representing horizontal reuse

$$\prod_{\ell=0}^{m-1} A_n \qquad \prod_{\ell=0}^{m-1}{}^{\text{hr}} A_n \qquad \prod_{\ell=0}^{p-1}{}^{\text{hr}} \left(\prod_{k=0}^{(m/p)-1} A_n \right)$$

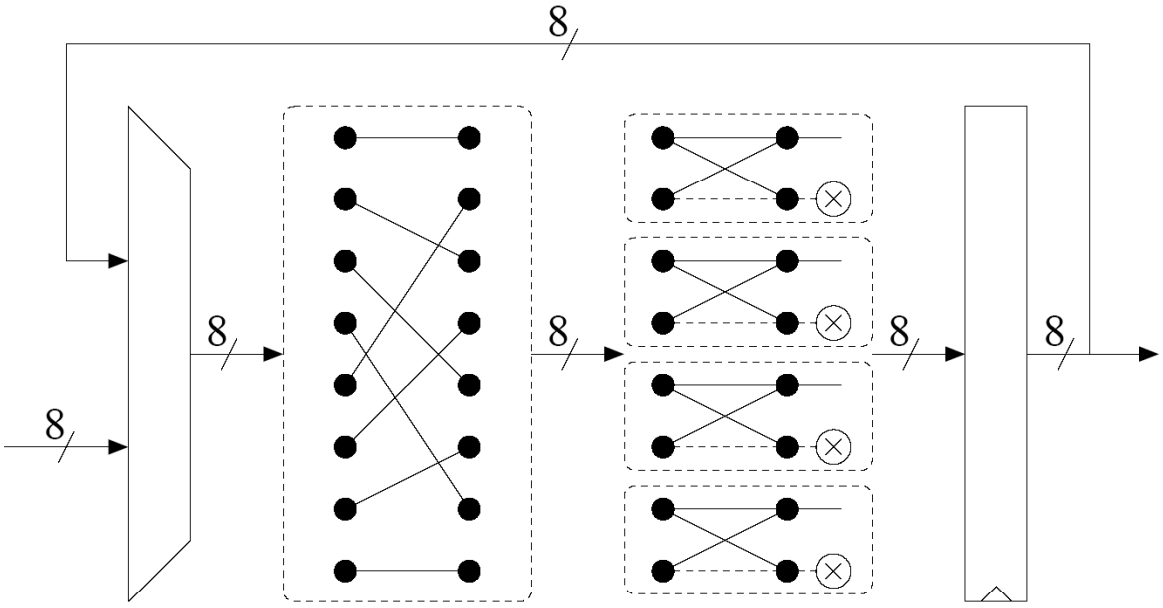


not horizontally reused

horizontally reused

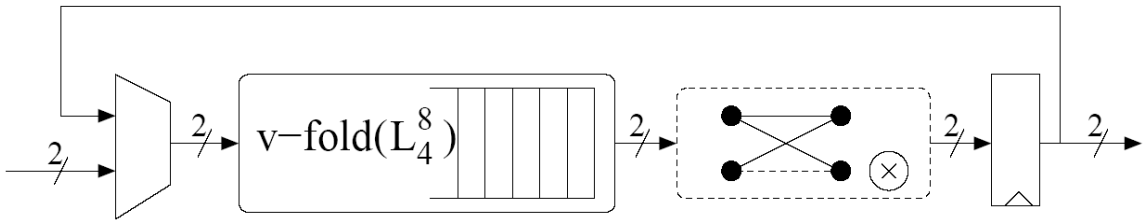
partially horizontally reused

Iterative Reuse of Logic



$p_{max} = \frac{n}{2}$

$p = 1, 2, 4, ...n/2$



$p_{min} = 1$

cost $\propto p$ **latency** $\propto \frac{1}{p}$

Fine-grained control over cost/latency tradeoff

Example: rewriting rules for streaming reuse

name	rule	condition
base-SR	$\underbrace{A_n}_{\text{stream}(n)} \to A_n$	
product-SR	$\underbrace{A_n \cdot B_n \cdots Z_n}_{\text{stream}(w)} \to \underbrace{A_n}_{\text{stream}(w)} \cdot \underbrace{B_n}_{\text{stream}(w)} \cdots \underbrace{Z_n}_{\text{stream}(w)}$	
stream-IR	$\underbrace{\prod^{\text{ir}} A_n}_{\text{stream}(w)} \to \prod^{\text{ir}} \underbrace{A_n}_{\text{stream}(w)}$	
stream1	$\underbrace{I_m \otimes A_k}_{\text{stream}(w)} \to I_{mk/w} \otimes^{\text{sr}} \left(I_{w/k} \otimes A_k\right)$	$mk > w$ and $k \leq w$
stream1-dep	$\underbrace{I_m \otimes_\ell A^k_\ell}_{\text{stream}(w)} \to I_{mk/w} \otimes^{\text{sr}}_{\ell_0} \left(I_{w/k} \otimes_{\ell_1} A^k_{\ell_0 \cdot w/k + \ell_1}\right)$	$mk > w$ and $k \leq w$
stream2	$\underbrace{I_m \otimes A_k}_{\text{stream}(w)} \to I_m \otimes^{\text{sr}} \underbrace{A_k}_{\text{stream}(w)}$	$k > w$
stream2-dep	$\underbrace{I_m \otimes_\ell A_k}_{\text{stream}(w)} \to I_m \otimes^{\text{sr}}_\ell \underbrace{A^k_\ell}_{\text{stream}(w)}$	$k > w$
stream-diag	$\underbrace{D_n}_{\text{stream}(w)} \to \text{StreamDiag}(D_n, w)$	$w \mid n$
stream-perm	$\underbrace{P_n}_{\text{stream}(w)} \to \text{StreamPerm}(P_n, w)$	$w \mid n$

Applicability to other transforms?

- DFT radix 2

$$R_{2^k}\prod_{i=0}^{k-1}\left[T_i\left(I_{2^{k-1}}\otimes DFT_2\right)L^{2^k}_{2^{k-1}}\right]$$

- DFT radix 2^r

$$R_{2^k}\prod_{i=0}^{k/r-1}\left[T_i\left(I_{2^{k-r}}\otimes DFT_{2^r}\right)L^{2^k}_{2^{k-r}}\right]$$

- 2-D DFT_{nxn}

$$\prod_{i=0}^{1}\left[L^{n^2}_{n}\left(I_n\otimes DFT_n\right)\right]$$

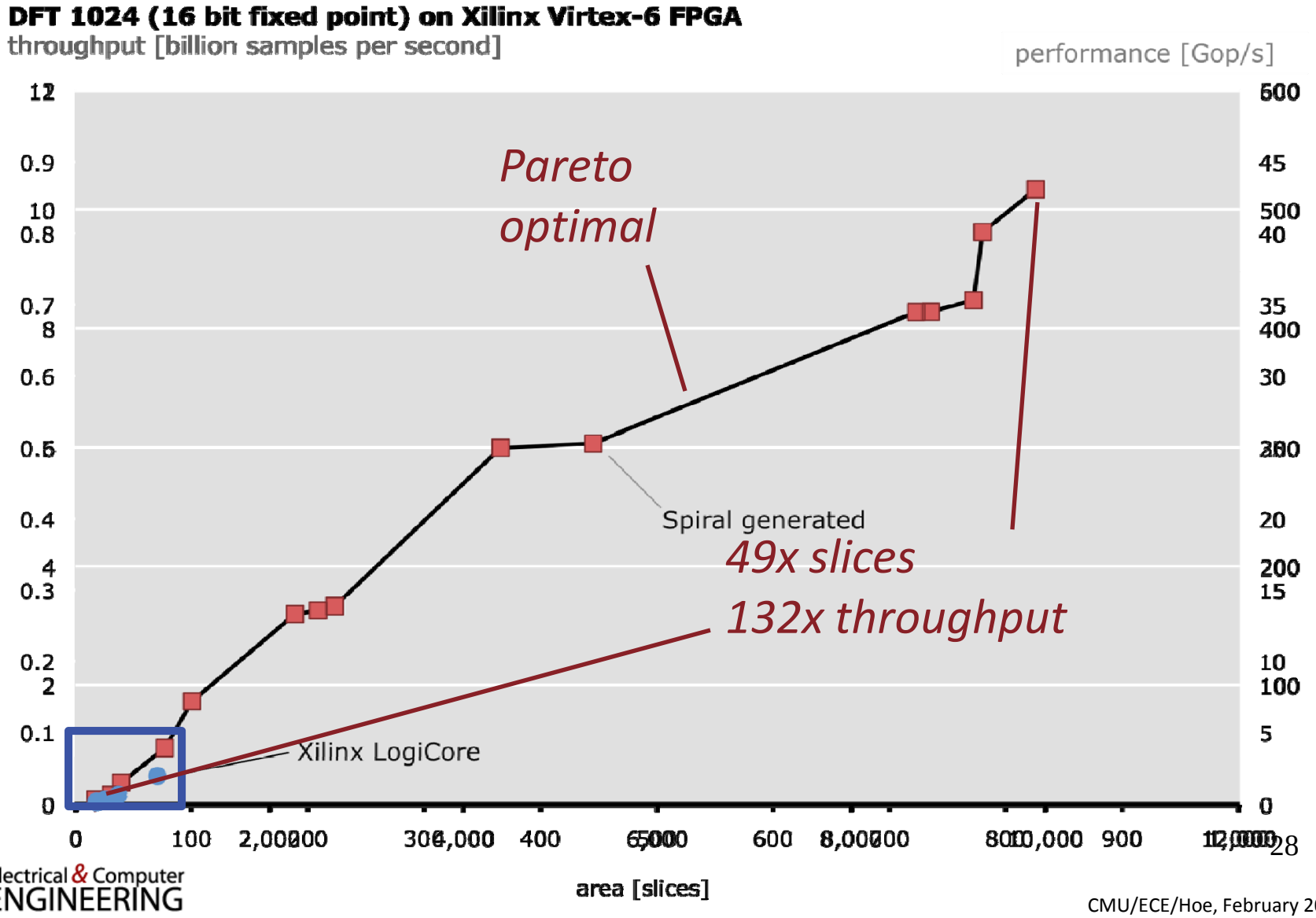
- WHT

$$\prod_{i=0}^{k/r-1}\left[\left(I_{2^{k-r}}\otimes WHT_{2^r}\right)L^{2^k}_{2^{k-r}}\right]$$

- DCT (type II)

$$DP\prod_{i=0}^{k-1}\left[A_{k-i}L^{2^k}_2\right]L^{2^k}_{2^{k-1}}L^{2^k}_{2^{k-1}}P^H_{2^k}$$

FPGA: Area vs. Throughput



Electrical & Computer ENGINEERING

Outline

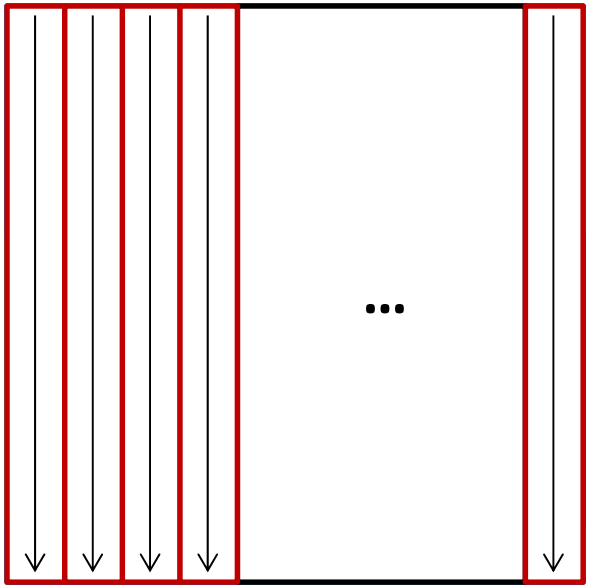
- SPIRAL Formula Framework
- SPIRAL for HW FFT cores
- SPIRAL for HW FFT “un”-core

A 2D-FFT Algorithm

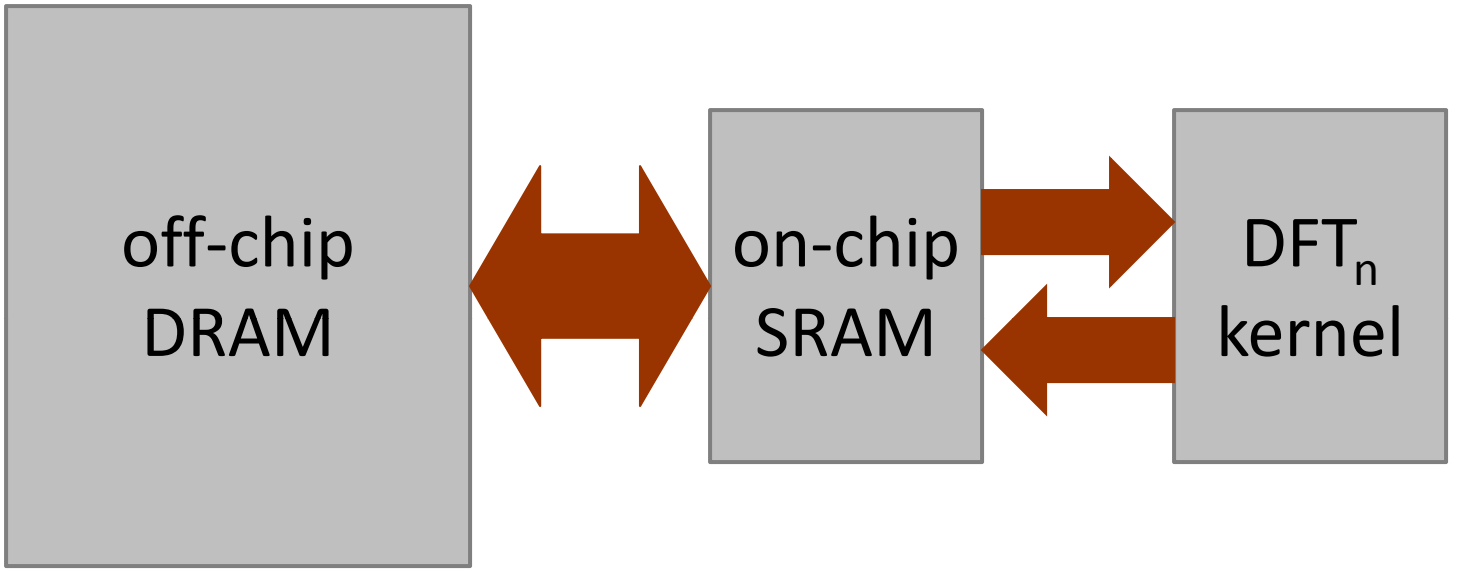
- Row-column algorithm:

$$\text{2D-DFT}_{n\times n} = \underbrace{(\text{DFT}_n \otimes \text{I}_n)}_{\text{Column Stage}}\underbrace{(\text{I}_n \otimes \text{DFT}_n)}_{\text{Row Stage}}$$

Dataset:
(Logical abstraction
of the 2D dataset)



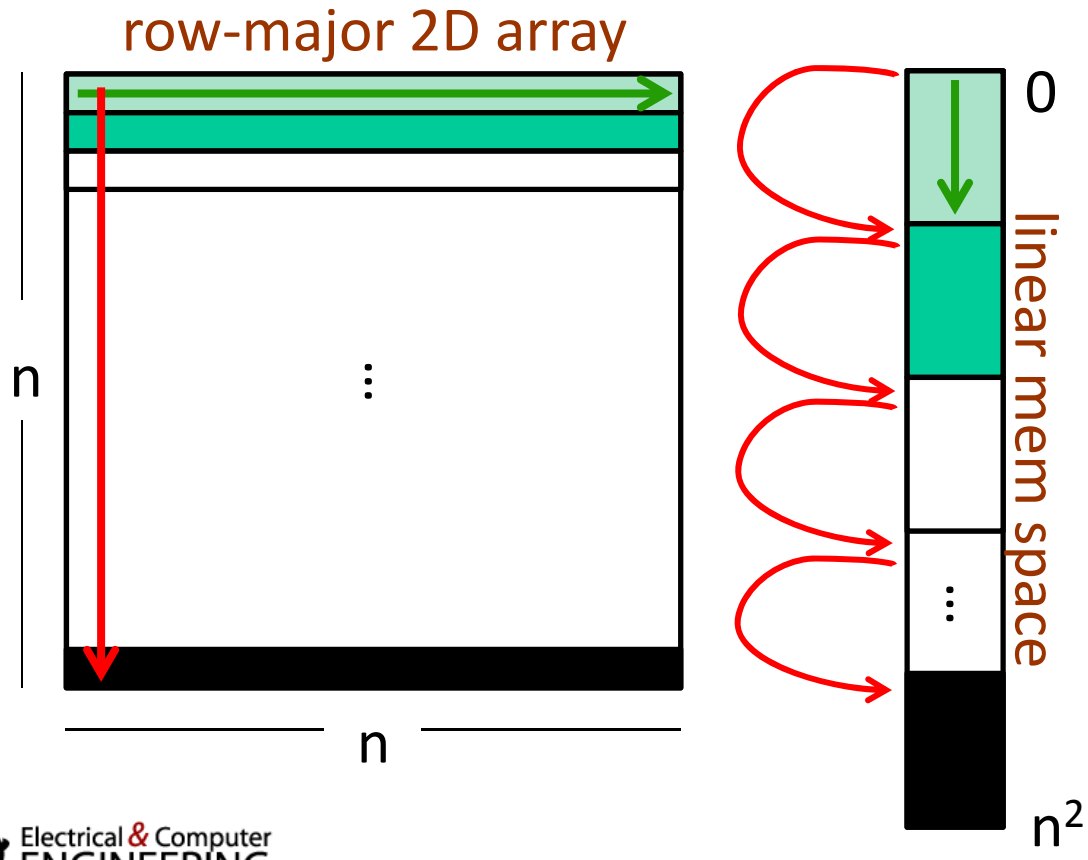
Off-chip Data Sets

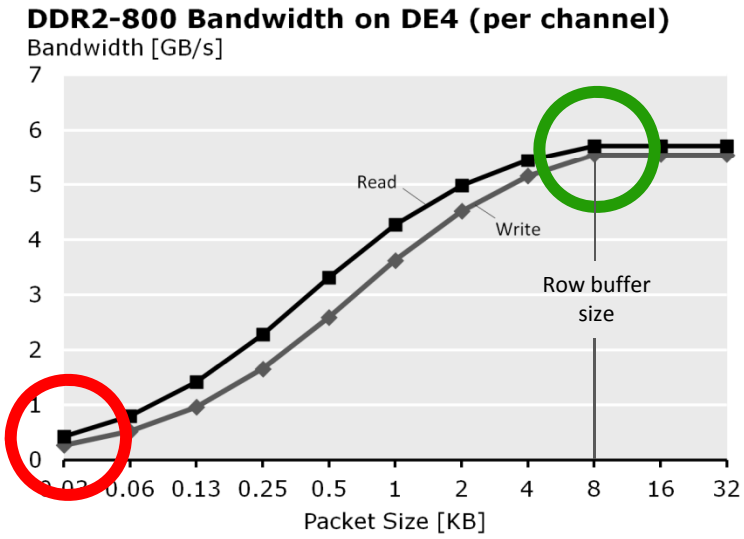


- Need to balance
  - kernel processing bandwidth
  - off-chip memory bandwidth
  - on-chip storage capacity

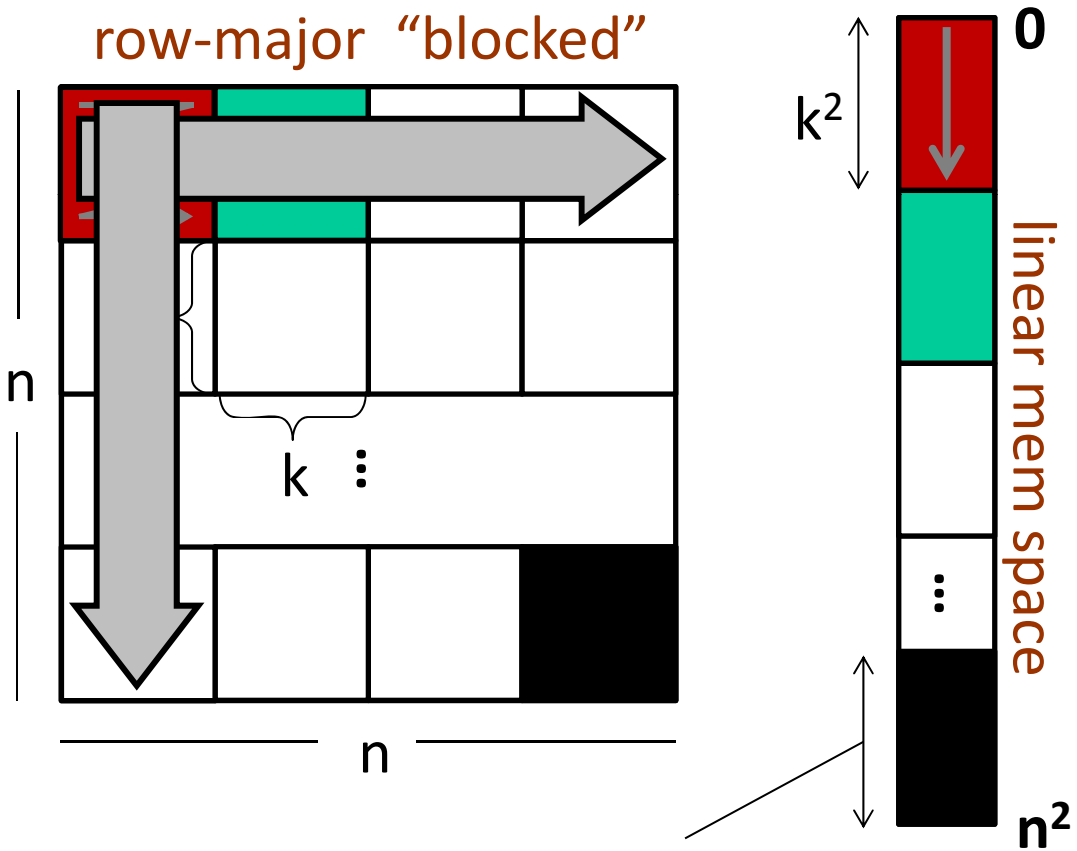
Inefficient DRAM Access Patterns

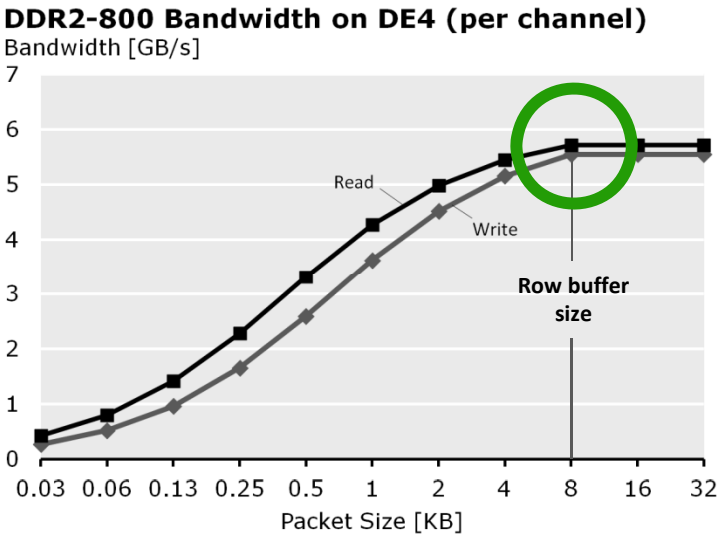
- Row-wise traversal -> Sequential accesses
- Column-wise traversal -> Large strided accesses





How to Optimize the Access Patterns





[Akin, et al., FCCM 2012]

Design Generator w/ Tensor Formalism

column stage row stage

$$\text{2D-DFT}_{n\times n} = (\overbrace{\text{DFT}_n \otimes \text{I}_n}^{\text{column stage}})(\overbrace{\text{I}_n \otimes \text{DFT}_n}^{\text{row stage}})$$

row-column algorithm

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

symmetric algorithm

symmetric algorithm with tiling

$$= \prod_{i=0}^{1} ((\text{L}_n^{n^2/k} \otimes \text{I}_k)(\text{I}_{n/k} \otimes \text{L}_n^{nk})(\text{I}_{n/k} \otimes (\text{I}_k \otimes \text{DFT}_n))(\text{I}_{n/k} \otimes (\text{L}_k^n \otimes \text{I}_k))(\text{I}_{n/k} \otimes (\text{L}_{n/k}^n \otimes \text{I}_k))$$

write tiles column-wise

transpose and re-tile on-chip

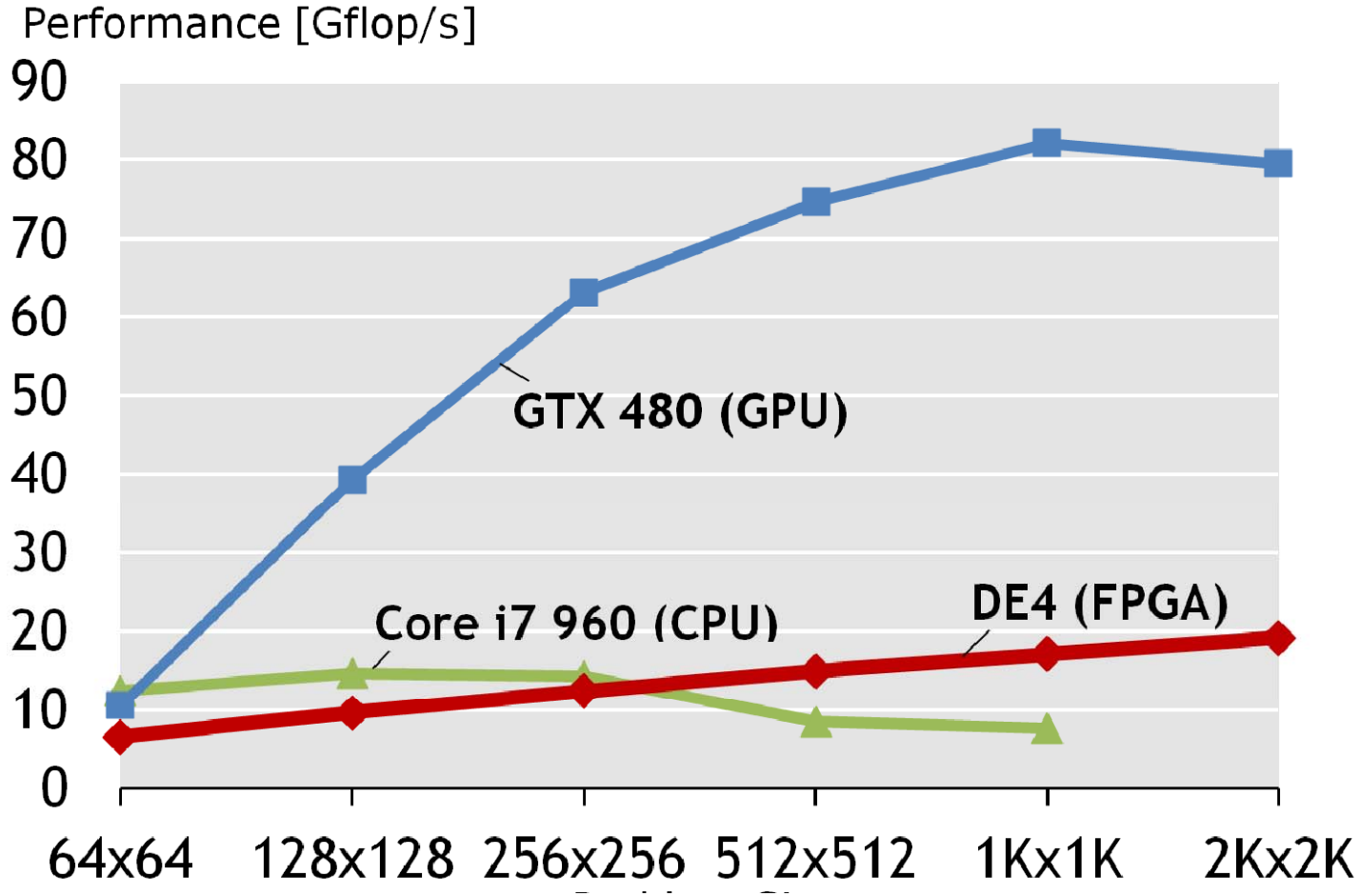
FFT processing

linearize on-chip

read tiles row-wise

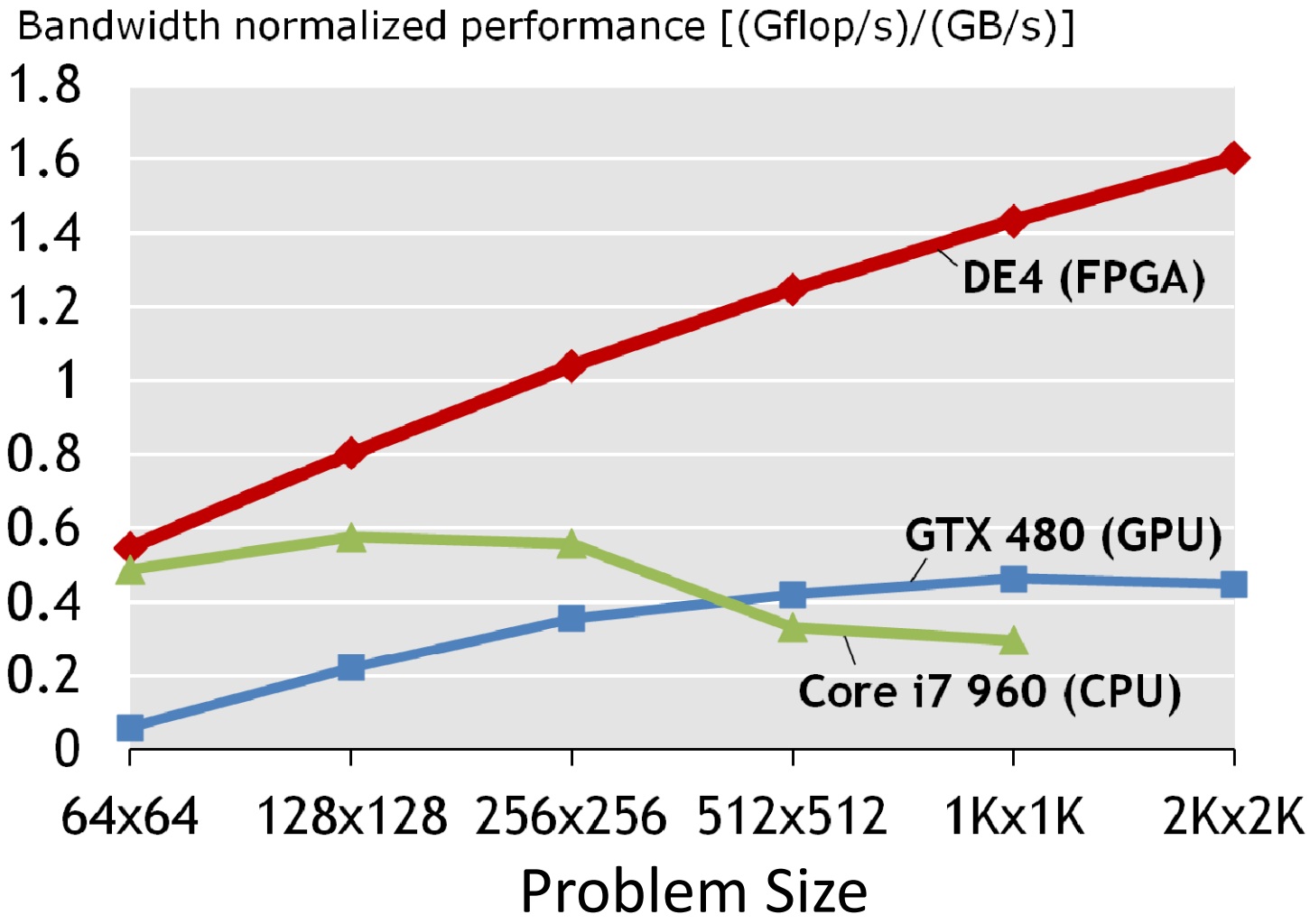
[Akin, et al., FCCM 2012]

2D-FFT (double) Raw Performance



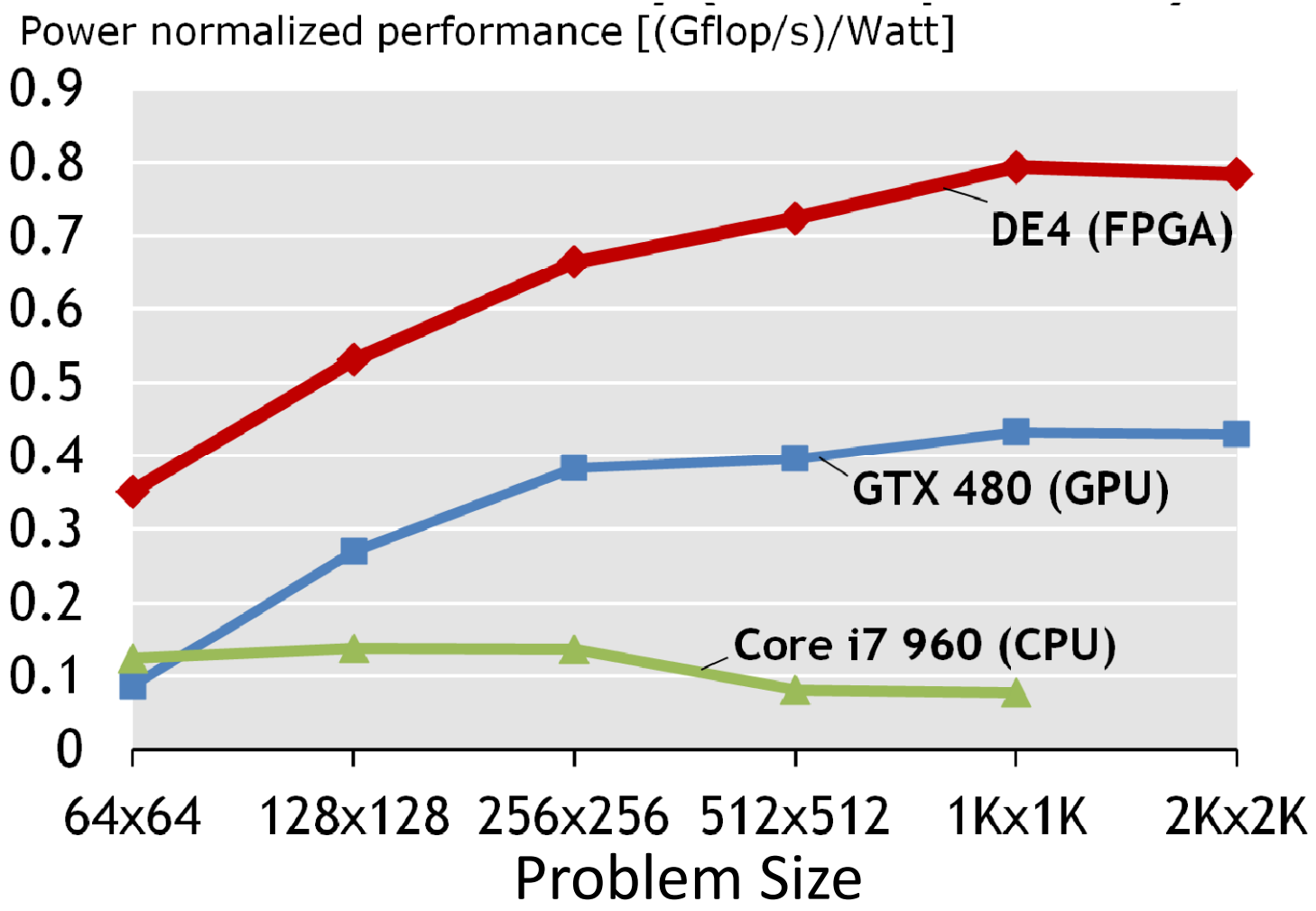
[Akin, et al., FCCM 2012]

2D-FFT (double) BW Efficiency



[Akin, et al., FCCM 2012]

2D-FFT (double) Power Efficiency



[Akin, et al., FCCM 2012]

Conclusions

- Encapsulating domain knowledge in a domain specific tool for high-level design automation
- SPIRAL
  - mathematical approach to DSP transform implementation (cores and “un”-core)
  - generalizable to other linear DSP transforms
  - as good as best expert designer
- Thank you