The FFT
Via Matrix Factorizations

A Key to Designing High Performance Implementations

Charles Van Loan
Department of Computer Science
Cornell University

A High Level Perspective...

Blocking For Performance

$$A = \begin{array}{c} \left[\begin{array}{cccc} A_{11} & A_{12} & \cdots & A_{1q} \\ A_{21} & A_{22} & \cdots & A_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ A_{p1} & A_{p2} & \cdots & A_{pq} \end{array}\right] \begin{array}{l} \}\, n_1 \\ \}\, n_2 \\ \\ \}\, n_q \end{array} \\ \underbrace{\qquad}_{n_1} \; \underbrace{\qquad}_{n_2} \qquad \underbrace{\qquad}_{n_q} \qquad\qquad \end{array}$$

A well known strategy for high-performance $Ax = b$ and $Ax = \lambda x$ solvers.

Factoring for Performance

One way to execute a matrix-vector product

$$y \;=\; F_n x$$

when $F_n = A_t \cdots A_2 A_1$ is as follows:

$y = x$
for $k = 1{:}t$
  $y = A_k x$
end

A different factorization $F_n \;=\; \tilde{A}_{\tilde{t}} \cdots \tilde{A}_1$ would yield a different algorithm.

The Discrete Fourier Transform ($n = 8$)

$$y = F_8 x = \begin{bmatrix} \omega_8^0 & \omega_8^0 & \omega_8^0 & \omega_8^0 & \omega_8^0 & \omega_8^0 & \omega_8^0 & \omega_8^0 \\ \omega_8^0 & \omega_8^1 & \omega_8^2 & \omega_8^3 & \omega_8^4 & \omega_8^5 & \omega_8^6 & \omega_8^7 \\ \omega_8^0 & \omega_8^2 & \omega_8^4 & \omega_8^6 & \omega_8^8 & \omega_8^{10} & \omega_8^{12} & \omega_8^{14} \\ \omega_8^0 & \omega_8^3 & \omega_8^6 & \omega_8^9 & \omega_8^{12} & \omega_8^{15} & \omega_8^{18} & \omega_8^{21} \\ \omega_8^0 & \omega_8^4 & \omega_8^8 & \omega_8^{12} & \omega_8^{16} & \omega_8^{20} & \omega_8^{24} & \omega_8^{28} \\ \omega_8^0 & \omega_8^5 & \omega_8^{10} & \omega_8^{15} & \omega_8^{20} & \omega_8^{25} & \omega_8^{30} & \omega_8^{35} \\ \omega_8^0 & \omega_8^6 & \omega_8^{12} & \omega_8^{18} & \omega_8^{24} & \omega_8^{30} & \omega_8^{36} & \omega_8^{42} \\ \omega_8^0 & \omega_8^7 & \omega_8^{14} & \omega_8^{21} & \omega_8^{28} & \omega_8^{35} & \omega_8^{42} & \omega_8^{49} \end{bmatrix} x$$

$$\omega_8 \;=\; \cos(2\pi/8) \;-\; i \cdot \sin(2\pi/8)$$

The DFT Matrix In General...

If $\omega_n = \cos(2\pi/n) - i \cdot \sin(2\pi/n)$ then

$$\begin{aligned} [F_n]_{pq} &= \omega_n^{pq} \\ &= (\cos(2\pi/n) \; - \; i \cdot \sin(2\pi/n))^{pq} \\ &= \cos(2pq\pi/n) \; - \; i \cdot \sin(2pq\pi/n) \end{aligned}$$

Fact:

$$F_n^H F_n \; = \; nI_n$$

Thus, $F_n/\sqrt{n}$ is unitary.

Data Sparse Matrices

An n-by-n matrix A is data sparse if it can be represented with many fewer than n^2 numbers.

Example 1.

A has lots of zeros. ("Traditional Sparse")

Example 2.

A is Toeplitz...

$$A = \begin{bmatrix} a & b & c & d \\ e & a & b & c \\ f & e & a & b \\ g & f & e & a \end{bmatrix}$$

More Examples of Data Sparse Matrices

A is a Kronecker Product $B \otimes C$, e.g.,

$$A = \left[\begin{array}{c|c} b_{11}C & b_{12}C \\ \hline b_{21}C & b_{22}C \end{array}\right]$$

If $B \in \mathrm{IR}^{m_1 \times m_1}$ and $C \in \mathrm{IR}^{m_2 \times m_2}$ then $A = B \otimes C$ has $m_1^2 m_2^2$ entries but is parameterized by just $m_1^2 + m_2^2$ numbers.

Extreme Data Sparsity

$$A \;=\; \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{\ell=1}^{n} S(i,j,k,\ell)\cdot \underbrace{(\text{2-by-2})\otimes\;\cdots\;\otimes(\text{2-by-2})}_{d\ \text{times}}$$

A is 2^d -by-2^d but is parameterized by $O(dn^4)$ numbers.

Factorization of F_n

The DFT matrix can be factored into a short product of sparse matrices, e.g.,

$$F_{1024} \;=\; A_{10} \cdots A_2 A_1 P_{1024}$$

where each A-matrix has 2 nonzeros per row and P_{1024} is a permutation.

From Factorization to Algorithm

If $n = 2^{10}$ and

$$F_n \;=\; A_{10} \cdots A_2 A_1 P_n$$

then

$y = P_n x$

for $k = 1{:}10$

$\qquad y = A_k x \qquad\qquad \leftarrow$ $2n$ flops.

end

computes $y = F_n x$ and requires $O(n \log n)$ flops.

Recursive Block Structure

$$F_8(:,[\,0\ 2\ 4\ 6\ 1\ 3\ 5\ 7\,]) \;=$$

$$\left[\begin{array}{cccc|cccc}
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & \omega_8 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & \omega_8^2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & \omega_8^3 \\
\hline
1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & -\omega_8 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & -\omega_8^2 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & -\omega_8^3
\end{array}\right]
\left[\begin{array}{cc} F_4 & 0 \\ 0 & F_4 \end{array}\right]$$

$F_{n/2}$ "shows up" when you permute the columns of F_n so that the odd-indexed columns come first.

Recursion...

We build an 8-point DFT from two 4-point DFTs...

$$F_8\,x \;=\; \left[\begin{array}{cccc|cccc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \omega_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \omega_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \omega_8^3 \\ \hline 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -\omega_8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -\omega_8^2 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -\omega_8^3 \end{array}\right] \left[\begin{array}{c} F_4 x(0{:}2{:}7) \\ F_4 x(1{:}2{:}7) \end{array}\right]$$

Radix-2 FFT: Recursive Implementation

function $y =$ **fft**(x, n)

    if $n = 1$

$$y = x$$

    else

$$m = n/2; \quad \omega = \exp(-2\pi i/n)$$

$$\Omega = \mathrm{diag}(1, \omega, \ldots, \omega^{m-1})$$

$$z_T = \mathbf{fft}(x(0{:}2{:}n-1), m)$$

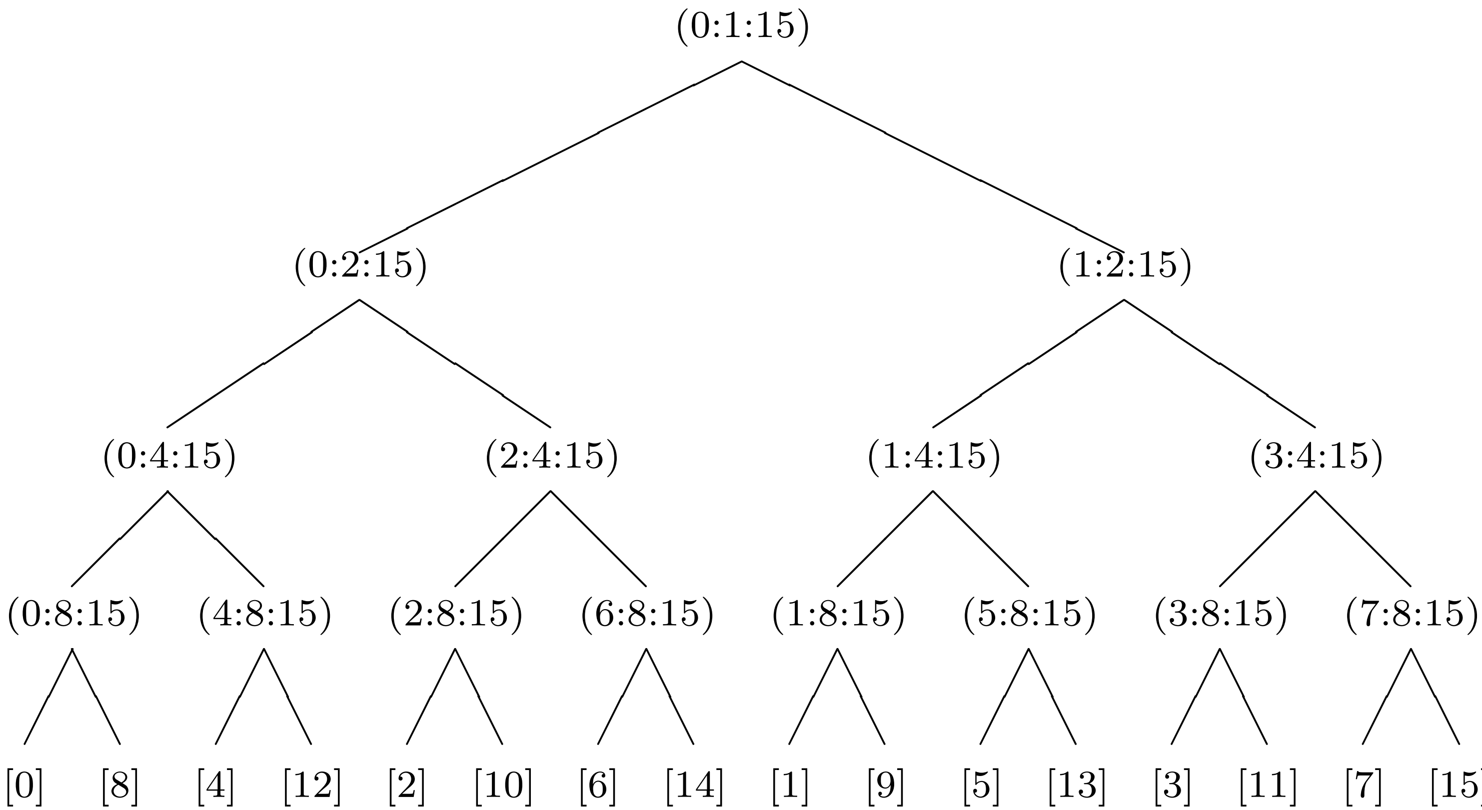
$$z_B = \Omega \cdot \mathbf{fft}(x(1{:}2{:}n-1), m)$$

$$y = \begin{bmatrix} I_m & I_m \\ I_m & -I_m \end{bmatrix} \begin{bmatrix} z_T \\ z_B \end{bmatrix}$$

Overall: $5n \log n$ flops.

    end

The Divide-and-Conquer Picture



Towards a Nonrecursive Implementation

The Radix-2 Factorization...

If $n = 2m$ and

$$\Omega_m \;=\; \mathrm{diag}(1, \omega_n, \ldots, \omega_n^{m-1}),$$

then

$$F_n \Pi_n \;=\; \begin{bmatrix} F_m & \Omega_m F_m \\ F_m & -\Omega_m F_m \end{bmatrix} \;=\; \begin{bmatrix} I_m & \Omega_m \\ I_m & -\Omega_m \end{bmatrix} (I_2 \otimes F_m).$$

where $\Pi_n \;=\; I_n(:, [0{:}2{:}n \;\; 1{:}2{:}n])$.

Note: $\quad I_2 \otimes F_m \;=\; \begin{bmatrix} F_m & 0 \\ 0 & F_m \end{bmatrix}.$

The Cooley-Tukey Factorization

$$n = 2^t$$

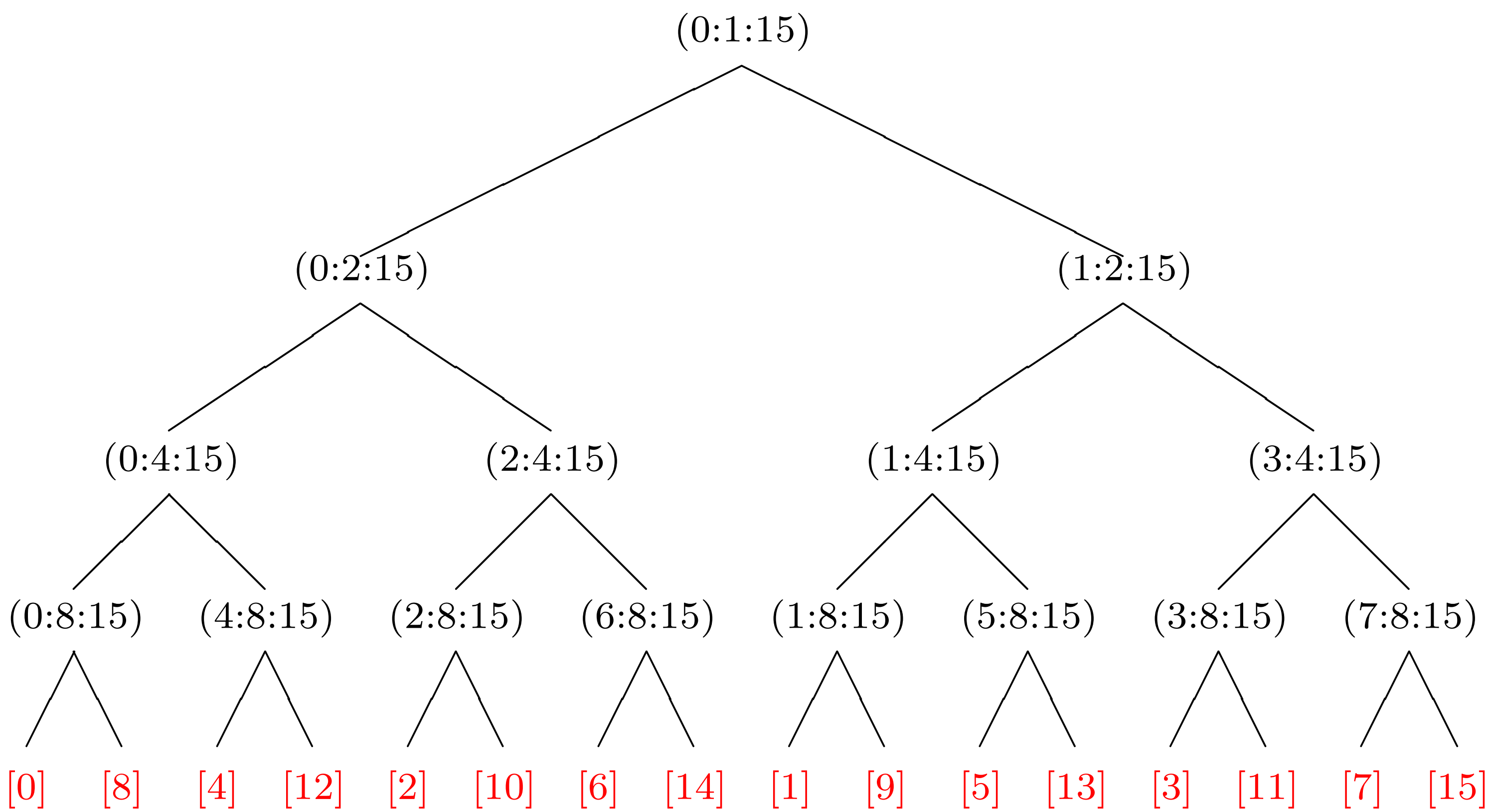
$$F_n = A_t \cdots A_1 P_n$$

$P_n =$ the n-by-n "bit reversal" permutation matrix

$$A_q = I_r \otimes \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix} \qquad L = 2^q,\ r = n/L$$

$$\Omega_{L/2} = \mathrm{diag}(1, \omega_L, \ldots, \omega_L^{L/2-1}) \qquad \omega_L = \exp(-2\pi i/L)$$

The Bit Reversal Permutation



Bit Reversal

$$
\begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \\ x(4) \\ x(5) \\ x(6) \\ x(7) \\ x(8) \\ x(9) \\ x(10) \\ x(11) \\ x(12) \\ x(13) \\ x(14) \\ x(15) \end{bmatrix}
=
\begin{bmatrix} x(0000) \\ x(0001) \\ x(0010) \\ x(0011) \\ x(0100) \\ x(0101) \\ x(0110) \\ x(0111) \\ x(1000) \\ x(1001) \\ x(1010) \\ x(1011) \\ x(1100) \\ x(1101) \\ x(1110) \\ x(1111) \end{bmatrix}
\to
\begin{bmatrix} x(0000) \\ x(1000) \\ x(0100) \\ x(1100) \\ x(0010) \\ x(1010) \\ x(0110) \\ x(1110) \\ x(0001) \\ x(1001) \\ x(0101) \\ x(1101) \\ x(0011) \\ x(1011) \\ x(0111) \\ x(1111) \end{bmatrix}
=
\begin{bmatrix} x(0) \\ x(8) \\ x(4) \\ x(12) \\ x(2) \\ x(10) \\ x(6) \\ x(14) \\ x(1) \\ x(9) \\ x(5) \\ x(13) \\ x(3) \\ x(11) \\ x(7) \\ x(15) \end{bmatrix}
$$

Butterfly Operations

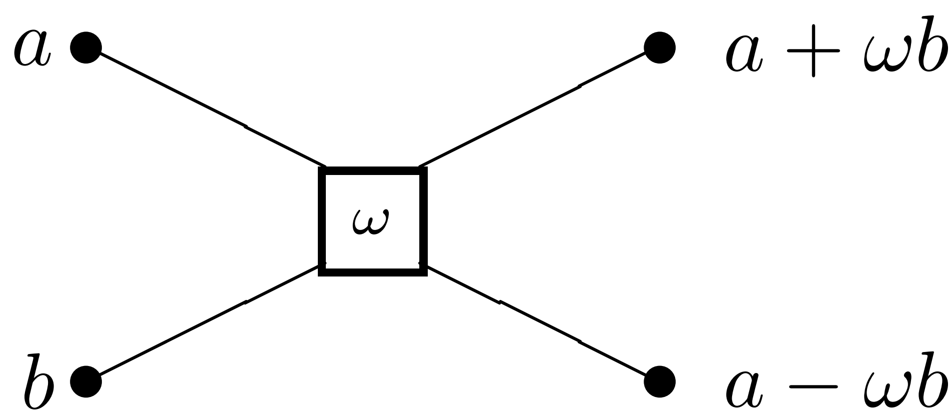
This matrix is block diagonal...

$$A_q = I_r \otimes \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix} \qquad L = 2^q,\ r = n/L$$

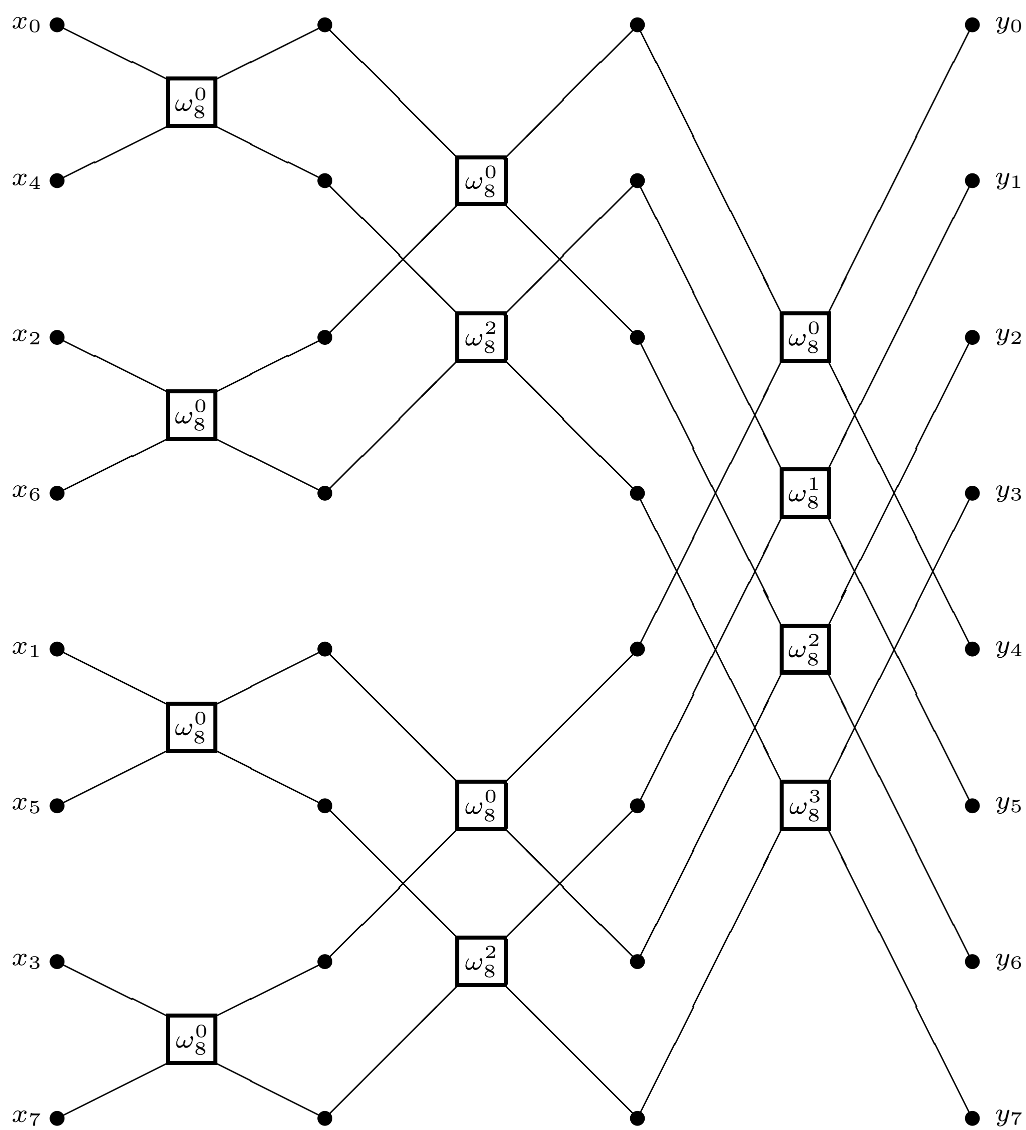
r copies of things like this

$$\left[\begin{array}{cccc|cccc} 1 & & & & \times & & & \\ & 1 & & & & \times & & \\ & & 1 & & & & \times & \\ & & & 1 & & & & \times \\ \hline 1 & & & & \times & & & \\ & 1 & & & & \times & & \\ & & 1 & & & & \times & \\ & & & 1 & & & & \times \end{array}\right]$$

At the Scalar Level...



Signal Flow Graph ($n = 8$)



The Transposed Stockham Factorization

If $n = 2^t$, then

$$F_n \;=\; S_t \cdots S_2 S_1,$$

where for $q = 1{:}t$ the factor $S_q = A_q \Gamma_{q-1}$ is defined by

$$A_q \quad = \; I_r \otimes B_L, \qquad\qquad L = 2^q,\; r = n/L,$$

$$\Gamma_{q-1} \; = \; \Pi_{r_*} \otimes I_{L_*}, \qquad\qquad L_* = L/2,\; r_* = 2r,$$

$$B_L \quad = \begin{bmatrix} I_{L_*} & \Omega_{L_*} \\ I_{L_*} & -\Omega_{L_*} \end{bmatrix},$$

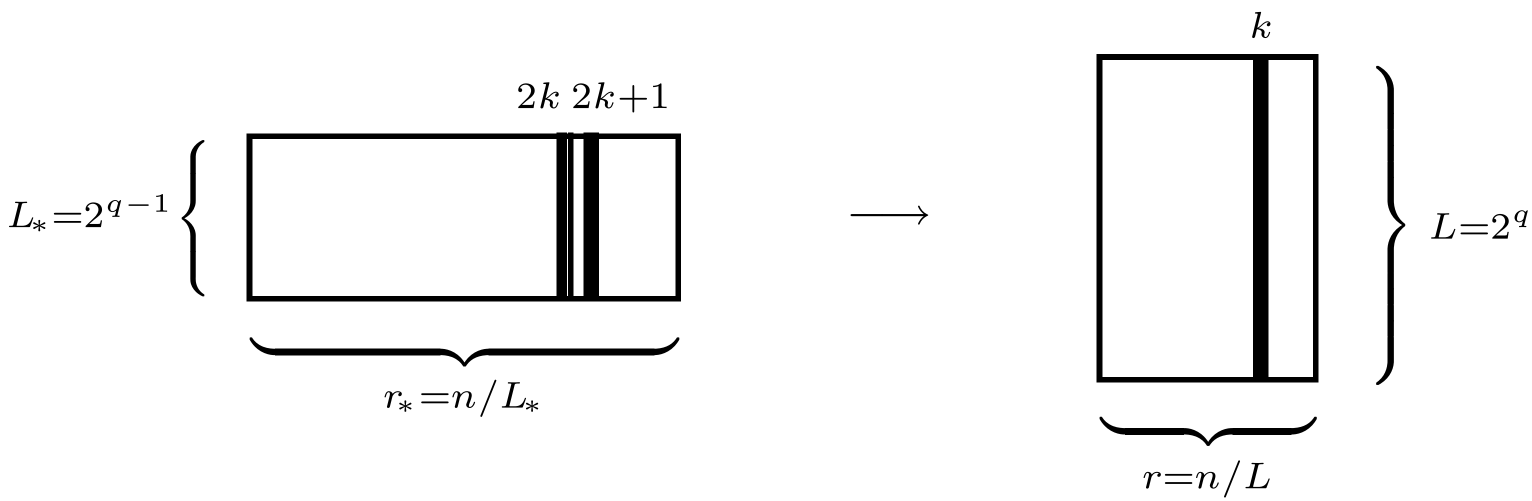
$$\Omega_{L_*} \quad = \mathrm{diag}(1,\; \omega_L, \ldots, \omega_L^{L_*-1}).$$

Perfect Shuffle

$$(\Pi_4 \otimes I_2) \left[\begin{array}{c} x_0 \\ x_1 \\ \hline x_2 \\ x_3 \\ \hline x_4 \\ x_5 \\ \hline x_6 \\ x_7 \end{array}\right] = \left[\begin{array}{c} x_0 \\ x_1 \\ \hline x_4 \\ x_5 \\ \hline x_2 \\ x_3 \\ \hline x_6 \\ x_7 \end{array}\right]$$

Cooley-Tukey Array Interpretation

Step q:

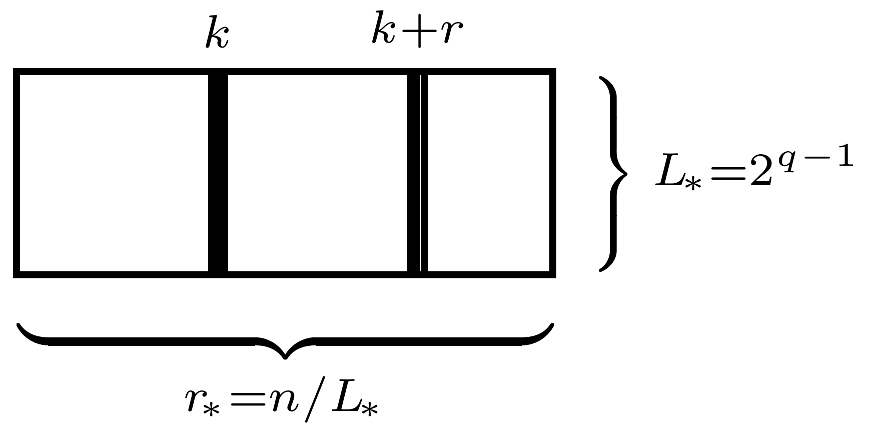


Reshaping

$$x = \begin{bmatrix} \times \\ \times \\ \times \\ \times \\ \times \\ \times \\ \times \\ \times \\ \times \end{bmatrix} \quad \longrightarrow \quad x_{2\times 4} = \begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

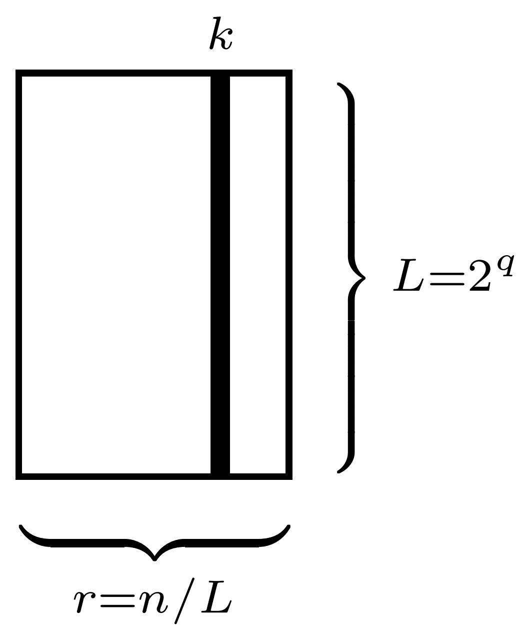
Transposed Stockham Array Interp

$$x^{(q-1)}_{L_*\times r_*} = F_{L_*}\, x^{T}_{r_*\times L_*} =$$



$$x^{(q)} = S_q x^{(q-1)}$$

$$x^{(q)}_{L\times r} = F_L\, x^{T}_{r\times L} =$$



$2 \times 2 \times 2$ Basic Radix-2 Versions

Store intermediate DFTs by row or column

Intermediate DFTs adjacent or not.

How the two butterfly loops are ordered.

$$x \;=\; \left(I_r \otimes \begin{bmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{bmatrix} \right) x \qquad L = 2^q,\; r = n/L$$

The Gentleman-Sande Idea

It can be shown that $F_n^T = F_n$ and so if

$$F_n \;=\; A_t \cdots A_1 P_n^T$$

then

$$F_n \;=\; F_n^T \;=\; P_n A_1^T \cdots A_t^T$$

and we can compute $y = F_n x$ as follows...

$$y \;=\; x$$

for $k = t{:}-1{:}1$

$$y = A_k^T x$$

end

$$y = P_n y$$

Convolution and Other Aps

From "problem space" to "DFT space" via

for $k = t{:} -1{:}1$

$$x = A_k^T x$$

end

$$x = P_n x$$

Do your thing in DFT space. Then inverse transform back to Problem space via

$$x = P_n^T x$$

for $k = 1{:}t$

$$x = A_k x$$

end

$$x = x/n$$

Can avoid the P_n ops by working in "scrambled" DFT space.

Radix-4

Can combine four quarter-length DFTs to produce a single full-length DFT:

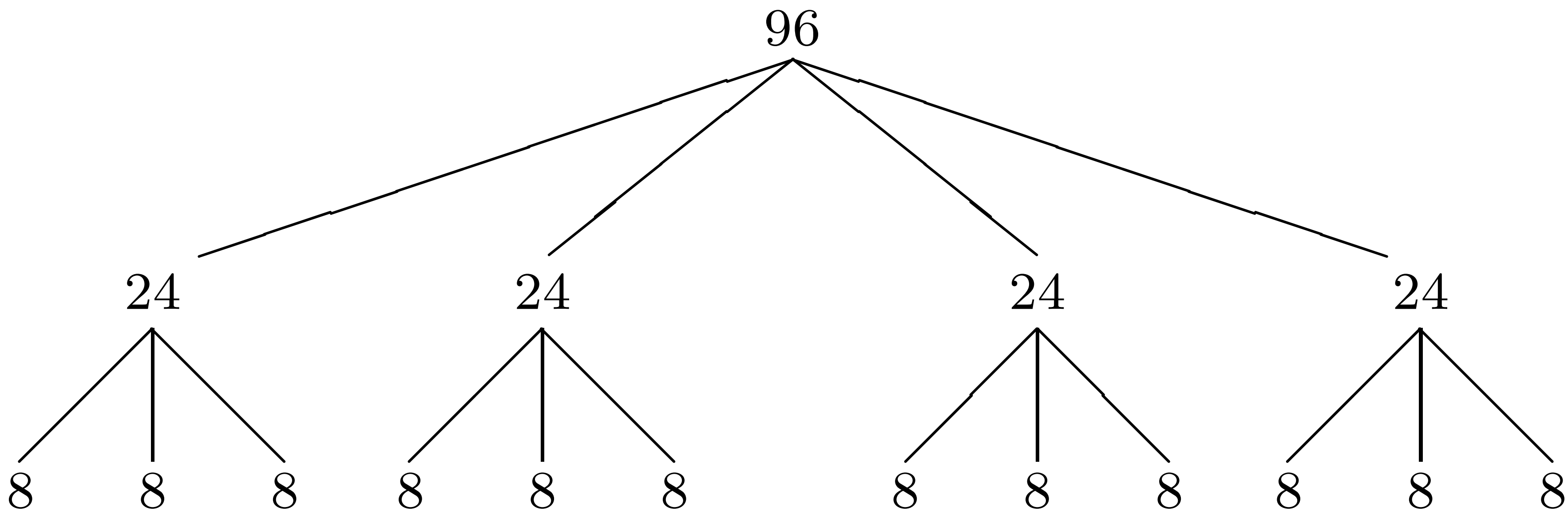
$$v = \begin{bmatrix} I & I & I & I \\ I & -iI & -I & iI \\ I & -I & I & -I \\ I & iI & -I & -iI \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} (a+c) + \ (b+d) \\ (a-c) - i(b-d) \\ (a+c) - \ (b+d) \\ (a-c) + i(b-d) \end{bmatrix},$$

The radix-4 butterfly.

Better re-use of data.

Fewer flops. Radix-4 FFT is $4.25n\log n$ (instead of $5n\log n$).

Mixed Radix



Multiple DFTs

Given: n_1-by-n_2 matrix X.

Multicolumn DFT Problem...

$$X \leftarrow F_{n_1} X$$

Multirow DFT Problem...

$$X \leftarrow X F_{n_2}$$

Blocked Multiple DFTs

$X \leftarrow F_{n_1} X$ becomes

$$\big[X_1 \mid X_2 \mid \cdots \mid X_p \big] \;\leftarrow\; \big[F_{n_1} X_1 \mid F_{n_1} X_2 \mid \cdots \mid F_{n_1} X_p \big]$$

The 4-Step Framework

A matrix reshaping of the $x \leftarrow F_n x$ operation when $n = n_1 n_2$:

$x_{n_1 \times n_2} \leftarrow x_{n_1 \times n_2} F_{n_2}$ Multiple row DFT

$x_{n_1 \times n_2} \leftarrow F_n(0{:}n_1 - 1, 0{:}n_2 - 1) .* \, x_{n_1 \times n_2}$ Pointwise multiply

$x_{n_2 \times n_1} \leftarrow x_{n_1 \times n_2}^T$ Transpose

$x_{n_2 \times n_1} \leftarrow x_{n_2 \times n_1} F_{n_1}$ Multiple row DFT .

Can be arranged so communication is concentrated in the transpose step.

Distributed Transpose: Example

Initial:

$$X = \begin{bmatrix} X_{00} & X_{01} & X_{02} & X_{03} \\ X_{10} & X_{11} & X_{12} & X_{13} \\ X_{20} & X_{21} & X_{22} & X_{23} \\ X_{30} & X_{31} & X_{32} & X_{33} \end{bmatrix}.$$

Transpose each block:

$$X \leftarrow \begin{bmatrix} X_{00}^T & X_{01}^T & X_{02}^T & X_{03}^T \\ X_{10}^T & X_{11}^T & X_{12}^T & X_{13}^T \\ X_{20}^T & X_{21}^T & X_{22}^T & X_{23}^T \\ X_{30}^T & X_{31}^T & X_{32}^T & X_{33}^T \end{bmatrix}.$$

Now regard as 2-by-2 and block transpose each block:

$$X \leftarrow \left[\begin{array}{cc|cc} X_{00}^T & X_{10}^T & X_{02}^T & X_{12}^T \\ X_{01}^T & X_{11}^T & X_{03}^T & X_{13}^T \\ \hline X_{20}^T & X_{30}^T & X_{22}^T & X_{32}^T \\ X_{21}^T & X_{31}^T & X_{23}^T & X_{33}^T \end{array}\right] .$$

Now do a 2-by-2 block transpose:

$$X \leftarrow \left[\begin{array}{cc|cc} X_{00}^T & X_{10}^T & X_{20}^T & X_{30}^T \\ X_{01}^T & X_{11}^T & X_{21}^T & X_{31}^T \\ \hline X_{02}^T & X_{12}^T & X_{22}^T & X_{32}^T \\ X_{03}^T & X_{13}^T & X_{23}^T & X_{33}^T \end{array}\right] .$$

Factorization and Transpose

$$x_{n\times m} \leftarrow x^T_{m\times n}$$

corresponds to

$$x \leftarrow P(m,n)x$$

where $P(m,n)$ is a perfect shuffle permutation, e.g.,

$$P(3,4) \; = \; I_{12}(:,[0\ 3\ 6\ 9\ 1\ 4\ 7\ 10\ 2\ 5\ 8\ 11])$$

Different multi-pass transposition algorithms correspond to different factorizations of $P(m,n)$.

Two-Dimensional FFTs

If X is an n_1-by-n_2 matrix then is 2D DFT is

$$X \leftarrow F_{n_1} X F_{n_2}$$

Option 1.

$$X \leftarrow F_{n_1} X$$

$$X \leftarrow X F_{n_2}$$

Option 2. Assume $n_1 = n_2$ and $F_{n_1} = A_t \cdots A_1$.

for $q = 1{:}t$

$$X \leftarrow A_q X A_q^T$$

end

Interminlgling the column and row butterfly computations can result in better locality.

3-Dimensional DFTs

Given $X(1{:}n_1, 1{:}n_2, 1{:}n_3)$, apply DFT in each of the three dimensions.

If

$$x = \text{reshape}(X(1{:}n_1, 1{:}n_2, 1{:}n_3), n_1 n_2 n_3, 1)$$

then the problem is to compute

$$x \;\leftarrow\; (F_{n_3} \otimes F_{n_2} \otimes F_{n_1})x$$

i.e.,

$$\begin{aligned}
x \;&\leftarrow\; (I_{n_3} \otimes I_{n_2} \otimes F_{n_1})x \\
x \;&\leftarrow\; (I_{n_3} \otimes F_{n_2} \otimes I_{n_1})x \\
x \;&\leftarrow\; (F_{n_3} \otimes I_{n_2} \otimes I_{n_1})x
\end{aligned}$$

d-Dimensional DFTs

Sample for $d = 5$:

$\mu = 1$	$X(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$ $X(\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_1)$	F_{n_1} $\Pi^T_{n_1,n}$
$\mu = 2$	$X(\alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_1)$ $X(\alpha_3, \alpha_4, \alpha_5, \alpha_1, \alpha_2)$	F_{n_2} $\Pi^T_{n_2,n}$
$\mu = 3$	$X(\alpha_3, \alpha_4, \alpha_5, \alpha_1, \alpha_2)$ $X(\alpha_4, \alpha_5, \alpha_1, \alpha_2, \alpha_3)$	F_{n_3} $\Pi^T_{n_3,n}$
$\mu = 4$	$X(\alpha_4, \alpha_5, \alpha_1, \alpha_2, \alpha_3)$ $X(\alpha_5, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$	F_{n_4} $\Pi^T_{n_4,n}$
$\mu = 5$	$X(\alpha_5, \alpha_1, \alpha_2, \alpha_3, \alpha_4)$ $X(\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5)$	F_{n_5} $\Pi^T_{n_5,n}$

Intemingling of component DFTs and tensor transpositions.

References

FFTW: `http:www.fftw.org`

C. Van Loan (1992). *Computational Frameworks for the Fast Fourier Transform*, SIAM Publications, Philadelphia, PA.