Contents

I Preliminaries

1 An Elementary Introduction to the Discrete Fourier Transform 3
- 1.1 Complex Numbers . . . 3
- 1.2 Trigonometric Interpolation . . . 5
- 1.3 Analyzing the Series . . . 8
- 1.4 Fourier Frequency Versus Time Frequency . . . 9
- 1.5 Filtering a Signal . . . 10
- 1.6 How Often Does One Sample? . . . 11
- 1.7 Notes and References . . . 12

2 Some Mathematical and Computational Preliminaries 13
- 2.1 Computing the Twiddle Factors . . . 14
- 2.2 Multiplying Two Complex Numbers . . . 15
  - 2.2.1 Real floating-point operation (FLOP) count . . . 15
  - 2.2.2 Special considerations in computing the FFT . . . 16
- 2.3 Expressing Complex Multiply-Adds in Terms of Real Multiply-Adds . 17
- 2.4 Solving Recurrences to Determine an Unknown Function . . . 17

II Sequential FFT Algorithms

3 The Divide-and-Conquer Paradigm and Two Basic FFT Algorithms 21
- 3.1 Radix-2 Decimation-In-Time (DIT) FFT . . . 22
  - 3.1.1 Analyzing the arithmetic cost . . . 23
- 3.2 Radix-2 Decimation-In-Frequency (DIF) FFT . . . 24
  - 3.2.1 Analyzing the arithmetic cost . . . 25
- 3.3 Notes and References . . . 25

4 Deciphering the Scrambled Output from In-Place FFT Computation 27
- 4.1 Iterative Form of the Radix-2 DIF FFT . . . 28

4.2 Applying the Iterative DIF FFT to a $N = 32$ Example. 30
4.3 Storing and Accessing Pre-computed Twiddle Factors 32
4.4 A Binary Address Based Notation and the Bit-Reversed Output . . . 34
4.4.1 Binary representation of positive decimal integers 34
4.4.2 Deciphering the scrambled output 34
4.5 Shorthand Notation for the Twiddle Factors 37

5 Bit-Reversed Input to the Radix-2 DIF FFT 39
5.1 The Effect of Bit-Reversed Input 40
5.2 A Taxonomy for Radix-2 FFT Algorithms 42
5.3 Shorthand Notation for the $\mathrm{DIF}_{\mathrm{RN}}$ Algorithm 42
5.3.1 Shorthand notation for the twiddle factors 43
5.3.2 Applying algorithm 5.2 to a $N = 32$ example. 43
5.4 Using Scrambled Output for Input to the Inverse FFT 45
5.5 Notes and References 45

6 Performing Bit-Reversal by Repeated Permutation of Intermediate Results 47
6.1 Combining Permutation with Butterfly Computation 48
6.1.1 The ordered radix-2 $\mathrm{DIF}_{\mathrm{NN}}$ FFT 48
6.1.2 The shorthand notation 53
6.2 Applying the Ordered DIF FFT to a $N = 32$ Example 53
6.3 In-Place Ordered (or Self-Sorting) Radix-2 FFT Algorithms. 56

7 An In-Place Radix-2 DIT FFT for Input in Natural Order 57
7.1 Understanding the Recursive DIT FFT and its In-Place Implementation 57
7.2 Developing the Iterative In-Place DIT FFT 62
7.2.1 Identifying the twiddle factors in the DIT FFT 64
7.2.2 The pseudo-code program for the $\mathrm{DIT}_{\mathrm{NR}}$ FFT algorithm 65
7.3 Shorthand Notation and a $N = 32$ Example 65

8 An In-Place Radix-2 DIT FFT for Input in Bit-Reversed Order 69
8.1 Developing the Iterative In-Place $\mathrm{DIT}_{\mathrm{RN}}$ FFT 74
8.1.1 Identifying the twiddle factors in the $\mathrm{DIT}_{\mathrm{RN}}$ FFT 75
8.1.2 The pseudo-code program for the $\mathrm{DIT}_{\mathrm{RN}}$ FFT 77
8.2 Shorthand Notation and a $N = 32$ Example 78

9 An Ordered Radix-2 DIT FFT 81
9.1 Deriving the (Ordered) $\mathrm{DIT}_{\mathrm{NN}}$ FFT From Its Recursive Definition . . 84
9.2 The Pseudo-code Program for the $\mathrm{DIT}_{\mathrm{NN}}$ FFT 87
9.3 Applying the (Ordered) $\mathrm{DIT}_{\mathrm{NN}}$ FFT to a $N = 32$ Example 89

10 Ordering Algorithms and Computer Implementation of Radix-2 FFTs 93
10.1 Bit-Reversal and Ordered FFTs 93
10.2 Perfect Shuffle and In-Place FFTs 94
10.2.1 Combining a software implementation with the FFT 95
10.2.2 Data adjacency afforded by a hardware implementation 96
10.3 Reverse Perfect Shuffle and In-Place FFTs 97
10.4 Fictitious Block Perfect Shuffle and Ordered FFTs 97
10.4.1 Interpreting the ordered $\mathrm{DIF}_{\mathrm{NN}}$ FFT algorithm 98
10.4.2 Interpreting the ordered $\mathrm{DIT}_{\mathrm{NN}}$ FFT algorithm 99

11 The Radix-4 and the Class of Radix-2^s FFTs 101
11.1 The Radix-4 DIT FFTs 101
11.1.1 Analyzing the arithmetic cost 103
11.2 The Radix-4 DIF FFTs 105
11.3 The Class of Radix-2^s DIT and DIF FFTs 108

12 The Mixed-Radix and Split-Radix FFTs 111
12.1 The Mixed-Radix FFTs 111
12.2 The Split-Radix DIT FFTs 111
12.2.1 Analyzing the arithmetic cost 113
12.3 The Split-Radix DIF FFTs 114
12.4 Notes and References 116

13 FFTs for Arbitrary N 117
13.1 The Main Ideas Behind Bluestein's FFT 117
13.1.1 DFT and the symmetric Toeplitz matrix-vector product 118
13.1.2 Enlarging the Toeplitz matrix to a circulant matrix 120
13.1.3 Enlarging the dimension of a circulant matrix to $M = 2^s$ 122
13.1.4 Forming the $M \times M$ circulant matrix-vector product 123
13.1.5 Diagonalizing a circulant matrix by a DFT matrix 124
13.2 Bluestein's Algorithm for Arbitrary N 126

14 FFTs for Real Input 129
14.1 Computing Two Real FFTs Simultaneously 129
14.2 Computing a Real FFT 130
14.3 Notes and References 132

15 FFTs for Composite N 133
15.1 Nested-Multiplication as a Computational Tool 134
15.1.1 Evaluating a polynomial by nested-multiplication 134
15.1.2 Computing a DFT by nested-multiplication 135
15.2 A 2D Array as a Basic Programming Tool 136

15.2.1 Row-oriented and column-oriented code templates 137
15.3 A 2D Array as an Algorithmic Tool 138
15.3.1 Storing a vector in a 2D array 138
15.3.2 Use of 2D arrays in computing the DFT 139
15.4 An Efficient FFT for $N = P \times Q$ 140
15.5 Multi-Dimensional Array as an Algorithmic Tool 142
15.5.1 Storing a 1D array into a multi-dimensional array 142
15.5.2 Row-oriented interpretation of ν-D arrays as 2D arrays 143
15.5.3 Column-oriented interpretation of ν-D arrays as 2D arrays . . . 144
15.5.4 Row-oriented interpretation of ν-D arrays as 3D arrays 144
15.5.5 Column-oriented interpretation of ν-D arrays as 3D arrays . . . 145
15.6 Programming Different ν-D Arrays From a Single Array 145
15.6.1 Support from the FORTRAN programming language 146
15.6.2 Further adaptation 147
15.7 An Efficient FFT for $N = N_0 \times N_1 \times \cdots \times N_{\nu-1}$ 148
15.8 Notes and References 152

16 Selected FFT Applications **153**
16.1 Fast Polynomial Multiplication 153
16.2 Fast Convolution and Deconvolution 155
16.3 Computing a Toeplitz Matrix-Vector Product 157
16.4 Computing a Circulant Matrix-Vector Product 158
16.5 Solving a Large Circulant Linear System 159
16.6 Fast Discrete Sine Transforms 159
16.7 Fast Discrete Cosine Transform 162
16.8 Fast Discrete Hartley Transform 164
16.9 Fast Chebyshev Approximation 166
16.10 Solving Difference Equations 168

III Parallel FFT Algorithms

17 Parallelizing the FFTs: Preliminaries on Data Mapping **177**
17.1 Mapping Data to Processors 178
17.2 Properties of Cyclic Block Mappings 179
17.3 Examples of CBM Mappings and Parallel FFTs 181

18 Computing and Communications on Distributed-Memory Multiprocessors **183**
18.1 Distributed-Memory Message-Passing Multiprocessors 184
18.2 The d-Dimensional Hypercube Multiprocessors 185
18.2.1 The subcube-doubling communication algorithm 186
18.2.2 Modeling the arithmetic and communication cost 187
18.2.3 Hardware characteristics and implications on algorithm design 188

18.3 Embedding a Ring by Reflected-Binary Gray-Code 190
18.4 A Further Twist—Performing Subcube-Doubling Communications on a Ring Embedded in a Hypercube 192
18.5 Notes and References 194
18.5.1 Arithmetic time benchmarks 194
18.5.2 Unidirectional times on circuit-switched networks 194
18.5.3 Bidirectional times on full-duplex channels 195

19 Parallel FFTs without Inter-Processor Permutations **197**
19.1 A Useful Equivalent Notation: | PID |Local M 197
19.1.1 Representing data mappings for different orderings 200
19.2 Parallelizing In-Place FFTs Without Inter-Processor Permutations . . 203
19.2.1 Parallel DIF$_{NR}$ and DIT$_{NR}$ algorithms 203
19.2.2 Interpreting the data mapping for bit-reversed output 209
19.2.3 Parallel DIF$_{RN}$ and DIT$_{RN}$ algorithms 209
19.2.4 Interpreting the data mapping for bit-reversed input 209
19.3 Analysis of Communication Cost 210
19.4 Uneven Distribution of Arithmetic Workload 210

20 Parallel FFTs with Inter-Processor Permutations **211**
20.1 Improved Parallel DIF$_{NR}$ and DIT$_{NR}$ Algorithms 211
20.1.1 The idea and a modified shorthand notation 211
20.1.2 The complete algorithm and output interpretation 215
20.1.3 The use of other initial mappings 220
20.2 Improved Parallel DIF$_{RN}$ and DIT$_{RN}$ Algorithms 220
20.3 Further Technical Details and a Generalization 221

21 A Potpourri of Variations on Parallel FFTs **223**
21.1 Parallel FFTs without Inter-Processor Permutations 224
21.1.1 The PID in Gray code 224
21.1.2 Using an ordered FFT on local data 225
21.1.3 Using radix-4 and split-radix FFTs 226
21.1.4 FFTs for Connection Machines 226
21.2 Parallel FFTs with Inter-Processor Permutations 228
21.2.1 Restoring the initial map at every stage 228
21.2.2 Pivoting on the right-most bit in local M 228
21.2.3 All-to-all inter-processor communications 229
21.2.4 Maintaining specific maps for input and output 230
21.3 A Summary Table 233
21.4 Notes and References 233

22 Further Improvement and a Generalization of Parallel FFTs **235**
22.1 Algorithms with Specific Mappings for Ordered Output 235

22.1.1 Algorithm I 235
22.1.2 Algorithm II 236
22.2 A General Algorithm and Communication Complexity Results 238
22.2.1 Phase I of the general algorithm 238
22.2.2 Phase II of the general algorithm 240

23 Parallelizing Two-dimensional FFTs 243
23.1 The Computation of Multiple 1D FFTs 243
23.2 The Sequential 2D FFT Algorithm 244
23.2.1 Programming considerations 245
23.2.2 Computing a single 1D FFT stored in a 2D matrix 245
23.2.3 Sequential algorithms for matrix transposition 246
23.3 Three Parallel 2D FFT Algorithms for Hypercubes 248
23.3.1 The transpose split (TS) method 248
23.3.2 The local distributed (LD) method 249
23.3.3 The 2D block distributed method 250
23.3.4 Transforming a rectangular signal matrix on hypercubes 250
23.4 The Generalized 2D Block Distributed (GBLK) Method for Subcube-grids and Meshes 251
23.4.1 Running hypercube (subcube-grid) programs on meshes 252
23.5 Configuring an Optimal Physical Mesh for Running Hypercube (Subcube-grid) Programs 253
23.5.1 Minimizing multi-hop penalty 253
23.5.2 Minimizing traffic congestion 254
23.5.3 Minimizing channel contention on a circuit-switched network . 258
23.6 Pipelining Subcube-doubling Communications on All Hypercube Channels 258
23.7 Changing Data Mappings During Parallel 2D FFT Computation . . . 265
23.8 Parallel Matrix Transposition By Changing Data Mapping 265
23.9 Notes and References 267

24 Computing and Distributing Twiddle Factors in the Parallel FFTs 271
24.1 Twiddle Factors for Parallel FFT Without Inter-Processor Permutations 272
24.2 Twiddle Factors for Parallel FFT With Inter-Processor Permutations 277

IV Appendices

A Fundamental Concepts of Efficient Scientific Computation 281
A.1 Time and Space Consumed by the DFT and FFT Algorithms 281
A.1.1 Relating operation counts to execution times 281
A.1.2 Relating MFLOPS to execution times and operation counts . . 282

A.2 Comparing Algorithms by Orders of Complexity 283
A.2.1 An informal introduction via motivating examples 283
A.2.2 Formal notations and terminologies 284
A.2.3 The big-Oh and big-Omega notations 285
A.2.4 Some common uses of the Θ-notation 286

B Solving Recurrence Equations by Substitution **287**
B.1 Deriving Recurrences From a Known Function 287
B.2 Solving Recurrences to Determine an Unknown Function 287
B.3 Mathematical Summation Formulas 290
B.4 Solving Generalized Recurrence Equations 292
B.5 Recurrences and the Fast Fourier Transforms 298

Bibliography **301**

Index **309**