

Introduction to Reversible Computing

Kalyan S. Perumalla

Oak Ridge National Laboratory
Knoxville, Tennessee, USA



CRC Press

Taylor & Francis Group

Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
A CHAPMAN & HALL BOOK

Contents

List of Figures	xiii
List of Tables	xv
List of Algorithms	xvii
Preface	xix
About the Author	xxi
Acknowledgments	xxiii
Book Organization	xxv
I Introduction	1
1 Scope	3
1.1 Notions of Computing	3
1.2 A Whole New Dimension	4
1.3 Related Terms and Synonyms	5
1.4 Similar yet Unrelated Concepts	7
2 Application Areas	9
2.1 General Reversible Computing Problem	9
2.2 Energy-Optimal Computing	10
2.3 Parallel Computing and Synchronization	11
2.3.1 Asynchronous Computing	11
2.3.2 Supercriticality	12
2.3.3 Performance Effects	14
2.4 Processor Architectures	15
2.4.1 Speculative Execution	15
2.4.2 Very Large Instruction Word	16
2.4.3 Anti-Memoization	16
2.5 Debugging	17
2.6 Source Code Control Systems	18
2.7 Fault Detection	19
2.8 Fault Tolerance	20

2.9	Database Transactions	22
2.10	Quantum Computing	23
2.11	Additional Applications	23
3	The Reversible Computing Spectrum	25
3.1	Spectrum	25
3.1.1	Components	25
3.1.2	Common Cases	26
3.2	Partial Reversibility	27
3.3	Unit of Reversibility	28
3.3.1	Reversing a Child's Play	28
3.3.2	Reversing the Movement of Library Books	29
3.3.3	Reversing Different Units of Computation	29
II	Theory	31
4	Systems and Principles	33
4.1	Logical Computations and Physical Processes	33
4.2	System Theoretic View of Computation	34
4.2.1	A Computation Example	35
4.2.2	Basic Components of Computational Energy	35
4.2.3	Dissipated Energy as Theoretical Energy Cost of Computation	37
4.2.4	Theoretical Lower Bound on Dissipated Energy	37
4.2.5	Reversibility for Zero Dissipated Energy	38
4.3	Reversible Circuits as Bit Compressors	40
4.3.1	Irreversible User Circuit within an Expanded Reversible Circuit	40
4.3.2	Clean and Dirty Bits	40
4.3.3	Custom Computation Circuit	41
4.3.4	General-Purpose Computation Circuit	41
4.3.5	Energy Cost of the Circuit	42
4.3.6	Analogy with Refrigeration	42
4.3.7	Reversibility in the Eye of the Beholder	43
4.4	Deterministic versus Non-Deterministic Reversal	44
4.4.1	Bit Erasure Cost versus Bit Reset Cost	45
4.4.2	Zero Energy Cost Schemes	45
5	Reversibility-Related Paradoxes	49
5.1	Entropy	49
5.2	Reversibility and Entropy	50
5.3	Ehrenfest's Urn Model	51
5.3.1	Model Configuration and Operation	51
5.3.2	Analysis	52
5.3.3	Forward and Reverse Algorithms	53
5.3.4	System versus Computational Reversibility	53

5.4	Kac-ring Model	55
5.4.1	Model Configuration and Operation	55
5.4.2	Analysis	56
5.4.3	Forward and Reverse Algorithms	57
5.4.4	An Entropy Function	57
5.4.5	System versus Computational Reversibility	57
5.5	Relation to Maxwell's Demon	59
5.5.1	Development	59
5.5.2	Setup and Operation	60
5.5.3	Operation as a Computer Program	60
5.5.4	Paradox Resolution	61
5.6	Relation to Other Paradoxes	63
5.6.1	Loschmidt's Paradox	63
5.6.2	Zermelo's Paradox	63
5.6.3	Berry's Paradox	64
5.7	Algorithmic Entropy	66
5.7.1	Definition	66
5.7.2	Non-Computability	67
5.8	Further Reading	68
6	Theoretical Computing Models	71
6.1	Overview	71
6.2	Turing Machine Model	72
6.3	Sources of Irreversibility in the Turing Machine Model	73
6.4	Definition of a Reversible Model	74
6.4.1	Rewriting Transition Rules: Quintuples to Quadruples	75
6.4.2	Adding History and Output Tapes	75
6.4.3	Canonical Turing Machine Model	76
6.5	Mapping Conventional Model Programs to a Reversible Model	77
6.6	Universality of Computation and Its Reversal	79
6.7	Space and Time Complexity of Reversible Execution	80
6.7.1	Complexity of Simple Reversal with One Segment	80
6.7.2	Partitioning Execution into Two Segments	80
6.7.3	Partitioning Execution into g Segments	83
6.7.4	Optimizing g for Minimal Total Space	83
6.7.5	Generalizing the Time-Space Trade-Off	84
6.8	Pebble Games	86
6.8.1	Rules and Objective	86
6.8.2	Analogies with the Reversible Turing Model	87
6.8.3	Complexity Analysis	88
6.8.4	Partial Reversibility	88
6.9	Further Reading	89

7	Relaxing Forward-Only Execution into Reversible Execution	91
7.1	Overview of Paradigms	91
7.2	Compute-Copy-Uncompute Paradigm	92
7.2.1	Equivalence Conditions	92
7.3	Forward-Reverse-Commit Paradigm	92
7.3.1	Equivalence Conditions	93
7.3.2	Sequence Conditions	94
7.3.3	Sequence Examples	95
7.4	Undo-Redo-Do Paradigm	95
7.5	Begin-Rollback-Commit Paradigm	97
III	Software	101
8	Reversible Programming Languages	103
8.1	The Role of Reversible Languages	103
8.2	Language-Level Reversibility Issues	105
8.2.1	Sequence and Recursive Reversal	105
8.2.2	Destructive Updates	106
8.2.3	Arithmetic	106
8.2.4	Conditionals	107
8.2.5	Loops	109
8.2.6	Additional Control Flow Issues	110
8.3	Procedural Languages	111
8.3.1	SRL and ESRL Reversible Languages	112
8.3.2	EPA Reversible Language	113
8.3.3	Janus Reversible Language	114
8.3.4	R Reversible Language	123
8.4	Functional and Logic Languages	125
8.5	Further Reading	126
9	Adding Reversibility to Irreversible Programs	127
9.1	Overview	127
9.2	Checkpointing	129
9.2.1	Full Checkpointing	129
9.2.2	Periodic Checkpointing	130
9.2.3	Incremental Checkpointing	131
9.2.4	Differential Checkpointing	133
9.2.5	Example Application	135
9.3	Reverse Computation	136
9.3.1	Automated: Compiler-Based	136
9.3.2	Automated: Interpreter-Based	140
9.3.3	Automated: Library-Based	143
9.3.4	Programmer-Assisted: Source Code-Based	143
9.3.5	Programmer-Assisted: Model-Based	144
9.4	Unified Composite	144

9.5	Further Reading	146
10	Reverse C Compiler	147
10.1	Reversibility of C Language Programs	148
10.2	Source-to-Source Method for Reversible C	150
10.2.1	Notation	151
10.2.2	Definition of Correctness of Reversible Execution	151
10.2.3	Runtime Tape Interface	152
10.2.4	Compilation Phases	152
10.3	Normalization	153
10.3.1	Declarations	153
10.3.2	Side-Effect Expressions	153
10.3.3	Function Calls	154
10.3.4	Arithmetic Expressions	154
10.3.5	for Statements	154
10.3.6	do-while Statements	155
10.3.7	while Statements	156
10.3.8	return Statements	157
10.3.9	continue Statements	158
10.3.10	break Statements	158
10.3.11	switch Statements	158
10.3.12	Post-Normalization State	159
10.4	Transformation	160
10.4.1	Expression Statements	160
10.4.2	Function Calls	161
10.4.3	Jump Statements	161
10.4.4	Compound Statements	163
10.4.5	Jumps across Nested Blocks	163
10.4.6	if Statements	164
10.4.7	switch Statements	165
10.4.8	while Statements	166
10.4.9	Libraries and I/O	167
10.4.10	Pragmas	168
10.5	Optimization	168
10.5.1	Value Recovery or Reconstruction	168
10.5.2	Irreversible and Environment Slices	168
10.5.3	Eliminating Reversal of Initialization	170
10.5.4	Invariant Expressions	171
10.5.5	Common Sub-Expression Elimination	172
10.5.6	Switch Statement Trade-Offs	172
10.5.7	Tape Compression	172
10.6	Tape Size Upper Bounds	173
10.7	Tape Size Determination	175

11 Reversal of Linear Codes	177
11.1 Automated Generation	177
11.2 Example: Fibonacci Sequence	178
11.3 General Linear Codes	179
11.4 Linear Code Reversal Algorithm	179
11.4.1 Preprocessing	180
11.4.2 Matrix Representation	180
11.4.3 Eliminating Singularity	182
11.5 Fast Backward	184
11.6 Other Common Linear Codes	185
12 Reversible Random Number Generation	187
12.1 Random Stream Traversal: Forward versus Reverse	187
12.2 Memory-Based Method to Make a Generator Reversible	189
12.3 Pseudorandom Numbers	192
12.3.1 Forward Generation	192
12.3.2 Reversible Generation	193
12.4 Reversible Generation from the Uniform Distribution	194
12.4.1 Open versus Closed Ranges	194
12.4.2 Linear Congruential Generators	195
12.4.3 Counting-Based Generators	196
12.5 Reversible Generation from Invertible Cumulative Distribu- tions	197
12.6 Reversible Generation from Probability Density Functions	197
12.6.1 Reversibility Problem	198
12.6.2 Upper-Bounded Rejection-Based Sampling	199
12.6.3 Upper- and Lower-Bounded Rejection-Based Sampling	202
12.7 Further Reading	204
13 Reversible Memory Allocation and Deallocation	207
13.1 The Problem: Reversible Dynamic Memory	207
13.2 A Simple Solution with Poor Memory Utilization	208
13.3 A Memory-Efficient Solution	209
13.3.1 Verification of Correctness of Allocation	210
13.3.2 Verification of Correctness of Deallocation	211
14 Reversible Numerical Computation	213
14.1 Software and Hardware Views	213
14.2 Sources of Irreversibility in Software	214
14.3 Considerations in Adding Reversibility	215
14.4 Defining Reversibility of Numerical Computation	216
14.4.1 Software-Level Operator Reversal	216
14.4.2 Operators with Constants	217
14.4.3 Operation Sequence Reversal	217
14.4.4 Hardware Circuit-Level Reversal	217

14.5	Reversal of Basic Arithmetic Operations in Software	218
14.5.1	Illustration of Basic Approach	218
14.5.2	Integer Operands	219
14.5.3	Floating Point Operands	225
14.6	Alternative Integer Framework for Reversibility	227
14.6.1	Internal Representation	227
14.6.2	Encoding Certain Error Conditions	228
14.6.3	Notation	229
14.6.4	Signed Values and Modulo Adjustment	229
14.6.5	Backward Compatibility	229
14.6.6	Computing \hat{Q} and R for Base v	231
14.6.7	Bit Representation Examples	231
14.6.8	Reversible Set of Arithmetic Operations	231
14.6.9	Combined Operation: A Simple Illustration	235
14.6.10	Reversal of Multiple Arithmetic Operations	237
14.7	Reversal of Basic Arithmetic in Hardware	237
14.8	Further Reading	239
15	Reversing a Sorting Procedure	241
16	Implementing Undo-Redo-Do	243
16.1	Application Model	243
16.2	Data Structures	244
16.3	Algorithms	245
16.4	Deletions and Memory Reclamation	246
16.5	Alternative Implementations	247
16.5.1	Undo and Redo Stacks	247
16.5.2	State Recreation via Reverse Computation	247
IV	Hardware	249
17	Reversible Logic Gates	251
17.1	Basic Concepts	251
17.1.1	Inadequacy of 2-bit Gates	252
17.1.2	w -bit Gate Candidates, $w \geq 3$	252
17.2	3-bit Reversible Gates	253
17.3	Fredkin Gate	254
17.3.1	Reversibility	254
17.3.2	Universality	255
17.4	Toffoli Gate	256
17.4.1	Reversibility	256
17.4.2	Universality	257
17.4.3	Increasing the Width to w Bits	257
17.5	Conservative Logic	259
17.6	Synthesis of Reversible Circuits	260

18 Reversible Instruction Set Architectures	261
18.1 Instruction Set Issues	261
18.1.1 Instruction Set for Memory Operations	262
18.1.2 Instruction Set for Simple Arithmetic	262
18.1.3 Instruction Set for Jumps	263
18.1.4 Implementation Considerations for Jumps	264
18.2 Reversible PDP-10-Like Instruction Set Architecture	264
18.2.1 Memory Operations	264
18.2.2 Arithmetic	265
18.2.3 Branches	266
18.3 Pendulum Instruction Set Architecture	266
18.3.1 Memory Operations	268
18.3.2 Arithmetic	268
18.3.3 Branches	268
18.3.4 Hardware Stacks	268
18.3.5 Input/Output	269
18.4 Hardware Interface to Reversible Memory	269
18.5 Further Reading	271
V Summary	273
19 Future Directions	275
19.1 Phased Transition from Irreversible to Reversible	275
19.2 Need for Additional Progress	276
19.3 Outlook	278
VI References	279
Bibliography	281
Index	295

List of Figures

2.1	Schematic of asynchronous recovery sequence using reverse computation-based rollback.	21
3.1	Reversible computing spectrum.	27
4.1	Reversibility at the intersection of computation and physics.	34
4.2	Computation and energy.	36
4.3	Problem-specific (custom) computational circuit C^s	41
4.4	General-purpose computational circuit C^g	42
4.5	Relation between the bits of interest to the user and the actual set of bits <i>reversibly</i> computed.	43
4.6	Irreversibility and non-determinism in state updates.	46
4.7	Bennett's scheme to incur zero energy cost in executing an irreversible deterministic program on a reversible <i>deterministic</i> machine.	48
4.8	Maroney's scheme to incur zero energy cost in executing an irreversible deterministic program on a reversible <i>non-deterministic</i> machine.	48
5.1	The Ehrenfest's Urn model illustrated with $N = 10$ balls.	52
5.2	Markov chain and probability distributions in the Ehrenfest's Urn model.	53
5.3	Illustration of the Kac-ring model with $N = 12$ sites containing $B = 7$ black balls, $W = 5$ white balls, and $n = 6$ markers.	55
5.4	Basic template of a Maxwell's Demon	61
5.5	Illustration of ambiguity for Maxwell's Demon regarding the source chamber of a particle.	62
6.1	Reversible execution with half trace size.	82
6.2	Reversible execution with g -fold smaller trace size.	84
6.3	Pebble game board structure.	86
6.4	Pebble board Rule 1: Adding a pebble to the board.	87
6.5	Pebble board Rule 2: Removing a pebble from the board.	88
7.1	Functional view of execution in the Compute-Copy-Uncompute paradigm.	93

7.2	Functional view of execution in the Forward-Reverse-Commit paradigm.	94
7.3	Functional view of execution in the Undo-Redo-Do paradigm.	97
7.4	Functional view of execution in the Begin-Rollback-Commit paradigm.	98
8.1	Classes of irreversible and reversible language programs.	104
9.1	Alternatives to convert irreversible programs into reversible ones.	128
9.2	Full checkpointing for snapshots-based reversal.	130
9.3	Periodic checkpointing for snapshots-based reversal.	131
9.4	Incremental checkpointing for snapshots-based reversal.	132
9.5	Differential checkpointing for snapshots-based reversal.	134
9.6	Example of the compiled approach to adding reversibility.	139
9.7	Compiler-based reversal using the source-to-source method.	139
9.8	Compiler-based reversal using the augmented compiler method.	140
9.9	Example of the interpreted approach to adding reversibility.	141
9.10	Interpreter-based reversal.	142
10.1	Reverse C Compilation Architecture.	150
10.2	Reverse compilation phases.	152
11.1	Sequence of transformations for Fibonacci sequence reversal.	182
12.1	Variable number of uniform distribution samples used for sampling a complex distribution.	199
12.2	Upper-bounded rejection-based sampling.	200
12.3	Upper- and lower-bounded rejection-based sampling.	204
14.1	Alternative internal representation of an integer amenable to reversible arithmetic.	228
16.1	Data structures to support Undo-Redo-Do mode of execution.	244
17.1	Realizing 2-bit <i>and</i> operation \otimes with the Fredkin gate.	255
17.2	Realizing 2-bit <i>or</i> operation \oplus with the Fredkin gate.	256
17.3	Realizing 1-bit <i>not</i> and fan-out with the Fredkin gate.	256
17.4	Realizing 2-bit <i>nand</i> operation $\bar{\otimes}$ with the Toffoli gate.	257
18.1	The irreversible and reversible constructs for jump instructions.	263
19.1	Transitioning from irreversible to reversible computing.	275

List of Tables

4.1	Types of Computation in Terms of Reversibility and Determinism	46
6.1	Definition of a Canonical Irreversible Turing Machine	73
6.2	Definition of a Standard Irreversible Turing Machine	77
6.3	Definition of a Standard Reversible Turing Machine	78
7.1	Relaxation of Forward-Only Execution into the Compute-Copy-Uncompute Paradigm	92
7.2	Relaxation of Forward-Only Execution into the Forward-Reverse-Commit Paradigm	94
7.3	Relaxation of Forward-Only Execution into the Undo-Redo-Do paradigm	96
8.1	Grammar of the Janus Time-Reversible Language	115
8.2	Reversible Calling Semantics of Subroutines in Janus	119
8.3	Reversal of the Janus Language Instructions	121
10.1	List of Example Pragma Specifications	169
10.2	Summary of State Bit Sizes for Various Statement Types	176
12.1	Summary of Reversible Random Number Generators	190
12.2	Example LCG Sequence for $m = 7$, $a = 3$, and $c = 2$	196
13.1	Simple Procedures for Dynamic Memory Management	208
13.2	Reversible Procedures for Dynamic Memory Management	210
14.1	Considerations in Adding Reversibility to Computer Arithmetic Operations	216
14.2	Computer Arithmetic Operations Considered for Reversibility	220
14.3	Internal Bit Widths for the Alternative Integer Representation Format Customized to Provide 2^k Bits of Usable Integer Precision	230
14.4	Usable Integer Precision in the Implementation of the Alternative Integer Representation Format Using 2^k Bits	230
14.5	A New Set of Alternative Arithmetic Operations Reversible without Generating History	232

17.1	Number of Candidate Permutations for w -Bit Reversible Gates	253
17.2	Input-Output Relations in a 3-Bit Fredkin Gate	254
17.3	Truth Table for a 3-Bit Fredkin Gate	254
17.4	Truth Table for a 3-Bit Toffoli Gate	257
17.5	Truth Table for a w -Bit Toffoli Gate	258
18.1	Memory Operations in PDP-10-Like Reversible Instruction Set Architecture	265
18.2	Simple Arithmetic Operations in PDP-10-Like Reversible Instruction Set Architecture	265
18.3	Reversible Memory Interface of a Rollback Chip	270

List of Algorithms

5.1	Forward and reverse algorithms for the Ehrenfest's Urn model.	54
5.2	Forward and reverse algorithms for the Kac-ring model	58
5.3	Program <i>q</i> to disprove the computability of algorithmic entropy	67
8.1	Reversible execution semantics of conditionals in Janus	117
8.2	Reversible execution semantics of looping in Janus	118
9.1	A simple irreversible program code fragment for adding reversibility	137
9.2	Example of reversal using the compiler-based approach	138
9.3	Example of reversal using the interpreter-based approach . .	142
11.1	Functions for reversible Fibonacci sequence generation	178
11.2	Linear code reversal algorithm	180
11.3	Preprocessing in linear code reversal algorithm	181
11.4	Automatically generated reversal for Fibonacci sequence . . .	182
11.5	Example code containing singularity preventing reversal . . .	183
11.6	Single- and double-stepping reverse functions for Fibonacci sequence	185
12.1	Reversing any forward random generator \mathcal{R}^* () using a finite <i>M</i> -sized circular buffer	191
12.2	Uniform random number generator	194
12.3	Reversible linear congruential generator	196
12.4	Update function \mathcal{S} () for counting-based generators	196
12.5	Random number generator for distributions with invertible cumulative distribution functions	198
12.6	Reversible upper-bounded rejection-based sampling	201
12.7	Reversible upper- and lower-bounded rejection-based sampling	205
14.1	Reversal of the Multiply operation	222
14.2	Reversal of the Semi-Scale operation	222
14.3	Reversal of the Divide operation when A or B is forgotten . .	224
14.4	Irreversible integer arithmetic for Celsius–Fahrenheit conversion	236
14.5	Reversible integer arithmetic for Celsius–Fahrenheit conversion	236
16.1	Algorithms to support Undo-Redo-Do mode of execution . . .	246
18.1	Realization of a reversible conditional statement using reversible jump instructions	267
18.2	Realization of a reversible looping statement using reversible jump instructions	267