Reversibility for Efficient Computing

(Ph.D. Thesis)

Michael P. Frank

MIT AI Lab, Rm. 821
545 Technology Sq.
Cambridge, MA 02139
http://www.ai.mit.edu/~mpf

Final Version
May 14, 1999

Available online through
http://www.ai.mit.edu/~mpf/thesis/phdthesis.html

Reversibility for Efficient Computing

by

Michael Patrick Frank

S.B., Stanford University (1991)
S.M., Massachusetts Institute of Technology (1994)

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 1999

© 1999 Massachusetts Institute of Technology. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 14, 1999

Certified by .....
Thomas F. Knight, Jr.
Senior Research Scientist
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Reversibility for Efficient Computing

by
Michael Patrick Frank

Submitted to the Department of Electrical Engineering and Computer Science
on May 14, 1999, in partial fulfillment of the Requirements for the degree of
Doctor of Philosophy

Abstract

Today's computers are based on *irreversible* logic devices, which have been known to be fundamentally energy-inefficient for several decades. Recently, alternative *reversible* logic technologies have improved rapidly, and are now becoming practical.

In traditional models of computation, pure reversibility seems to decrease overall computational efficiency; I provide a proof to this effect. However, traditional models ignore important physical constraints on information processing.

This thesis gives the first analysis demonstrating that in a realistic model of computation that accounts for thermodynamic issues, as well as other physical constraints, the judicious use of reversible computing can strictly *increase* asymptotic computational efficiency, as machine sizes increase. I project real benefits for supercomputing at a large (but achievable) scale in the fairly near term. And with proposed future computing technologies, I show that reversibility will benefit computing at all scales.

Next, the thesis demonstrates that reversible computing techniques do not make computer design much more difficult. I describe how to design asymptotically efficient processors using an "adiabatic" reversible electronic logic technology that can be built with today's microprocessor fabrication processes. I describe a simple universal reversible parallel processor chip that our group recently fabricated, and a reversible instruction set for a more traditional RISC-style uniprocessor.

Finally, I describe techniques for programming reversible computers. I present a high-level language and a compiler suitable for coding efficient reversible algorithms, and I describe a variety of example algorithms, including efficient reversible sorting, searching, arithmetic, matrix, and graph algorithms. As an example application, I present a linear-time, constant-space reversible program for simulating the Schrödinger wave equation of quantum mechanics.

Thesis Supervisor: Thomas F. Knight, Jr.
Title: Senior Research Scientist

Acknowledgments

First and foremost, I am enduringly grateful to my advisor Tom Knight and also to Norm Margolus, for taking me into their fold, for their extensive and wise guidance in all areas of this research, for the many things I learned from them, and for providing an exceptionally supportive and stimulating work environment. Thanks very much also to the other readers Gill Pratt and Gerry Sussman, for kindly agreeing to serve on my thesis committee and lend their expertise to the evaluation of this work.

I would also like to thank the Pendulum group's head graduate student, Carlin Vieri, for inventing the Pendulum architecture, which provided the context for much of this work; for taking it upon himself to do most of the project's administrative chores; and for the fun we had working together on the nuts and bolts of various aspects of the project such as the Pendulum ISA and the FlatTop chip. (Also, thanks for all the Nolios!)

I should also thank a number of other MIT students: Matt DeBergalis created many helpful software tools, such as the Pendulum assembler and emulator. Josie Ammer assisted with chip design, and made important contributions to the theory and algorithms work. Scott Rixner and Nicole Love were my primary design partners on the Tick and FlatTop processors, respectively. Matt Becker often popped into my office for a friendly greeting and an interesting conversation.

Prof. Michael Sipser deserves thanks for his informal but very helpful guidance during the development of the proof in §3.4. I am also grateful to Alain Tapp and Pierre McKenzie of the University of Montreal, for providing much helpful feedback on that result, and for inviting me to visit their lab. Thanks also to Prof. Tom Leighton for communicating the reversible shortest-path algorithm, and to both him and Prof. Charles Leiserson for feedback on the scaling results of chapter 5.

Finally, I would like to thank Carl R. Witty and Warren D. Smith for helpful editorial comments on drafts of this document, my family and my wife Lori for their love, and all my friends throughout my years at MIT for their companionship and encouragement.

This research was supported by DARPA (the Defense Advanced Research Projects Agency of the United States of America) as part of the Scalable Computing Systems research program [37], under contract number DABT63-95-C-0130.

to my father, Patrick Gene Frank

Contents

1 Introduction and background **17**
- 1.1 What this thesis is about . . . 17
- 1.2 Motivation . . . 18
- 1.3 Brief history of reversible computing . . . 20
  - 1.3.1 Early thermodynamics of computation . . . 20
  - 1.3.2 Development of reversible models of computation . . . 21
  - 1.3.3 Development of physically reversible logic devices . . . 21
  - 1.3.4 Previous reversible computing theory . . . 22
  - 1.3.5 Optimal scaling of physical machines . . . 22
  - 1.3.6 Programming reversible machines . . . 23
- 1.4 Major contributions of this thesis . . . 23
- 1.5 Overview of thesis chapters . . . 24
- 1.6 Overall message of thesis . . . 26

I Foundations of reversible computing **29**

2 Physical constraints on computation **31**
- 2.1 Propagation speed limits . . . 31
- 2.2 Information density limits . . . 32
  - 2.2.1 Entropy bounds from black hole physics. . . . 33
  - 2.2.2 Entropy bounds for a photon gas. . . . 34
  - 2.2.3 Entropy bounds at normal temperatures/pressures. . . . 34
- 2.3 Information flux rate limits . . . 35
- 2.4 Computation rate limits . . . 36
- 2.5 Reversibility of physics . . . 37
  - 2.5.1 Physical reversibility and information erasure . . . 39
  - 2.5.2 Reversibility, entropy, and the second law . . . 39
  - 2.5.3 Entropy and energy . . . 42
  - 2.5.4 Logical irreversibility and energy dissipation . . . 43

2.6 Quantum computation 45
2.7 Physical constraints—conclusion 45

3 Reversible computing theory **47**
3.1 Models of computation 48
3.1.1 Computability 50
3.2 Computational complexity and efficiency 50
3.2.1 Computational efficiency vs. computational complexity 50
3.2.2 Characterizing computational complexity 51
3.2.3 Complexity classes 56
3.3 Review of existing reversible computing theory 56
3.3.1 Reversible models of computation 56
3.3.2 Computability in reversible models 56
3.3.3 Time complexity in reversible models 58
3.3.4 Reversible entropic complexity 59
3.3.5 Reversible space complexity 60
3.3.6 Miscellaneous developments 64
3.4 Reversible vs. irreversible space-time complexity 64
3.4.1 General definitions 66
3.4.2 Oracle results 69
3.4.3 Non-relativized separation 81
3.4.4 Decompression algorithm 83
3.4.5 Can this proof be carried farther? 84
3.5 Summary of reversible complexity results for traditional models . . . 84

4 Ultimate physical models of computation **89**
4.1 What is a model of computation? 90
4.2 Existing models of computation 92
4.3 Problems with the existing models 94
4.4 Some candidates for an ultimate model 97
4.4.1 The reversible 3-D mesh (R3M) model 98
4.4.2 The ballistic 3-D mesh (B3M) model 99
4.4.3 The quantum 3-D mesh (Q3M) model 99
4.5 A "tight" Church's thesis 100
4.6 Ultimate computational complexity 101
4.7 Summary of discussion of ultimate models 102

5 Reversibility and physical scaling laws 103
5.1 Types of architectures studied . . . 104
5.1.1 Shared properties . . . 104
5.1.2 Fully irreversible architecture . . . 104
5.1.3 Time-proportionately reversible architecture . . . 105
5.1.4 Ballistic reversible architecture . . . 105
5.2 Analyses under various physical costs . . . 106
5.2.1 Entropy cost . . . 107
5.2.2 Area-time product . . . 109
5.2.3 Time cost . . . 113
5.2.4 Spacetime cost . . . 119
5.2.5 Mass-time product . . . 121
5.2.6 (Area + mass) × time . . . 121
5.2.7 Entropy + mass-time . . . 121
5.3 Generalizing the results . . . 121
5.3.1 Speedups for irreversible computations on reversible machines 122
5.4 Summary of scaling results . . . 124

II Engineering reversible computational systems 127

6 Adiabatic circuits 129
6.1 Maximizing the efficiency of iCMOS . . . 130
6.1.1 Basic iCMOS review . . . 130
6.1.2 iCMOS entropy generation. . . . 133
6.1.3 The SIA semiconductor roadmap . . . 139
6.1.4 Minimizing permanent energy dissipation in iCMOS . . . 144
6.1.5 Maximizing per-area processing rate for iCMOS . . . 145
6.1.6 Maximizing iCMOS cost-efficiency . . . 147
6.2 Historical development of adiabatic circuits . . . 148
6.3 A comment on terminology . . . 152
6.4 Basic principles of adiabatic circuits . . . 153
6.5 The SCRL technique . . . 156
6.5.1 Basic SCRL components . . . 156
6.5.2 SCRL pipelines . . . 160
6.5.3 Timing disciplines . . . 162
6.6 SCRL circuit analyses . . . 162
6.6.1 A simple SCRL model for analysis . . . 165
6.6.2 Switching losses as a function of technology parameters . . . 166
6.6.3 Minimizing the sum of switching and leakage energy . . . 172

6.6.4 Fixing a problematic case for plain SCRL . . . 179
6.7 Experimental SCRL Circuits . . . 181
6.7.1 The Billiard Ball Model . . . 183
6.7.2 The Billiard Ball Model Cellular Automaton . . . 184
6.7.3 Logic minimization . . . 185
6.7.4 FlatTop array design . . . 186
6.7.5 Minimum energy estimation . . . 188
6.8 Resonant power supply techniques . . . 189
6.9 Scaling SCRL to future technology generations . . . 190
6.10 Mostly reversible computation . . . 192
6.11 Adiabatic Circuits—Conclusion . . . 193

7 Future reversible device technologies 195
7.1 Cooling technologies . . . 195
7.2 Irreversible device technologies . . . 197
7.3 Reversible technologies . . . 198
7.4 Future device technologies—Conclusion . . . 202

8 Design and programming of reversible processors 205
8.1 Context of this work . . . 206
8.1.1 Previous reversible architectures . . . 206
8.1.2 Pendulum architecture . . . 206
8.2 Reversible instruction set architectures . . . 207
8.2.1 Asymptotic efficiency . . . 207
8.2.2 Use of paired branches . . . 207
8.2.3 Reversible logic/arithmetic operations . . . 210
8.2.4 Data transfer operations . . . 211
8.2.5 Hardware-guaranteed reversibility . . . 211
8.3 Simple example PISA program: Multipication algorithm . . . 212
8.3.1 Discussion . . . 215
8.4 Reversible programming languages . . . 216
8.4.1 General issues . . . 216
8.4.2 "R," a reversible language . . . 218
8.4.3 The R compiler . . . 219
8.5 Reversible algorithms . . . 219
8.5.1 Sorting . . . 220
8.5.2 Arithmetic . . . 220
8.5.3 Matrices . . . 220
8.5.4 Searches . . . 221

8.5.5 Graph problems . . . 221
8.5.6 Physical simulations . . . 221
8.6 Operating system issues . . . 224
8.7 Parallelism . . . 226

9 Alternative applications for reversibility 229
9.1 Auditable/verifiable/trustable computation . . . 229
9.1.1 Detecting transient errors . . . 230
9.1.2 Logging or limiting effects of unwelcome intrusions . . . 231
9.2 Program debugging . . . 232
9.3 Transaction processing and database rollback . . . 233
9.4 Speculative execution in multiprocessors . . . 233
9.5 Numerical stability in physics simulations . . . 234
9.6 Alternative applications: Conclusion . . . 234

10 Conclusion and Future Work 237
10.1 Summary of Contributions . . . 237
10.2 Major areas for future research . . . 239
10.3 Final words . . . 243

III Appendices 245

A FlatTop processor schematics and layouts 247
A.1 High-level blocks . . . 247
A.2 Detailed gate schematics . . . 249
A.3 Cell layout . . . 249

B The Pendulum instruction set architecture (PISA) 255
B.1 Overall organization . . . 255
B.2 List of Instructions . . . 259
B.3 Arithmetic/logical ops . . . 260
B.4 Ordinary branches . . . 268
B.5 Special instructions . . . 271

C The R reversible programming language 275
C.1 Introduction . . . 275
C.2 What type of language is R? . . . 276
C.3 Overview of R Syntax . . . 276
C.4 User-level Constructs . . . 276
C.4.1 Program Structure . . . 277

C.4.2 Control Structure 278
C.4.3 Variables 281
C.4.4 Data Modification 282
C.4.5 Expressions 284
C.4.6 Static Data 287
C.4.7 Input/Output 288
C.5 Example Programs 289
C.6 Compiler Internals 289
C.7 Conclusions 289

D The R language compiler **291**
D.1 R Compiler User's Guide 291
D.2 Compilation technique 292
D.3 Internal compiler constructs 293
D.3.1 Intermediate-level internal constructs 293
D.3.2 Low-level constructs 301
D.4 Compiler LISP source code 313
D.4.1 `loader.lisp` 315
D.4.2 `util.lisp` 315
D.4.3 `infrastructure.lisp` 316
D.4.4 `location.lisp` 321
D.4.5 `environment.lisp` 322
D.4.6 `regstack.lisp` 326
D.4.7 `variables.lisp` 327
D.4.8 `branches.lisp` 330
D.4.9 `expression.lisp` 334
D.4.10 `clike.lisp` 339
D.4.11 `print.lisp` 342
D.4.12 `controlflow.lisp` 342
D.4.13 `subroutines.lisp` 344
D.4.14 `staticdata.lisp` 347
D.4.15 `program.lisp` 347
D.4.16 `library.lisp` 348
D.4.17 `files.lisp` 349
D.4.18 `test.lisp` 350

E Reversible Schrödinger wave simulation **357**
E.1 Derivation of discrete update rule 357
E.2 Reversible C implementation 364
E.3 Source code in R language 376

E.4 Compiled PISA code 378

F Units, Constants, and Notations **387**

List of Figures

2-1 Forward and reverse determinism 38
2-2 Information "erasure" under reversible physics 40
2-3 Venn diagram of entropy and information 43

3-1 Configuration graphs in reversible and irreversible models of computation 57
3-2 Bennett's 1989 reversible simulation algorithm 61
3-3 Euler tour of an irreversible machine's configuration tree 63
3-4 Structure of permutation oracles 68
3-5 Encoding outdegree-1 directed graphs in self-reversible oracles 71
3-6 Problem graph defined by oracle 73
3-7 Optimal reversible pebble game strategy 76
3-8 Triangle representation of oracle queries 77
3-9 Visualizing the definition of the set of pebbled nodes 78
3-10 Decompression algorithm 85

5-1 Speed limit for reversible machines of given dimensions 111
5-2 An optimal irreversible machine for 3-D CA simulations 116
5-3 A faster reversible machine for 3-D CA simulations 117
5-4 "Folding" a column of processors to minimize volume 120

6-1 Ordinary CMOS inverter 131
6-2 Energy dissipation in conventional switching 132
6-3 Charging with constant current 154
6-4 Adiabatic charging in CMOS 155
6-5 Voltage curves for slow and fast charging 156
6-6 SCRL inverter 158
6-7 SCRL generalized inverter 159
6-8 SCRL bidirectional latch 161
6-9 SCRL pipeline 163
6-10 Full SCRL timing diagram 164
6-11 Simplified SCRL circuit model 165
6-12 Scaling of energy/op in SCRL with speed, given nonzero leakage . . . 174

6-13 How minimum energy scales with threshold voltage 178
6-14 A problematic case for SCRL 180
6-15 Fixing the problem case 182
6-16 Two logic gates in the physical billiard ball model 183
6-17 The billiard-ball model cellular automaton 185
6-18 Boolean logic form of BBMCA update rule 186
6-19 Grid of FlatTop processing elements 187

8-1 Reversible control-flow structures 209
8-2 Reversible assembly-language multiplication routine. 213
8-3 Multiplication routine in R language 218
8-4 Initial state in Schrödinger simulation 222
8-5 State after 1000 simulation steps 223

A-1 Block diagram of PE cell 248
A-2 Icon for a single FlatTop cell 248
A-3 One corner of an array of FlatTop PEs 250
A-4 The full 20×20 array of PEs 251
A-5 Stage 1 logic gate 252
A-6 Logic for stages 2 and 3 253
A-7 Complete layout of a single FlatTop PE 253

B-1 "Non-expanding" arithmetic/logical operations 256
B-2 "Expanding" arithmetic/logical operations 257
B-3 Branch and I/O operations 258

List of Tables

2.1 Limits on entropy density from various analyses 35
2.2 Physical constraints on computation 46

3.1 Some existing theoretical models of computation 49
3.2 Some measures of cost or complexity 52

5.1 Three classes of architectures 106
5.2 Summary of asymptotic scaling results 124

6.1 SIA semiconductor roadmap 140
6.2 Minimum entropy generation for irreversible CMOS 141
6.3 Dielectric constants 141
6.4 Outer-area times time, for irreversible CMOS 146
6.5 Device parameters for the HP14 fabrication process 189

7.1 Maximum entropy flux estimates 196
7.2 Maximum per-area rate for various irreversible technologies 198
7.3 Entropy coefficients for various reversible technologies 200
7.4 Reversible device density to beat irreversible technolgies 201
7.5 Thicknesses above which reversible machines win 201

8.1 Registers used in MULT routine 214

F.1 Unit magnitude prefixes 388
F.2 Fundamental units used in this document 388
F.3 Fundamental physical constants used in this document 389
F.4 Asymptotic order-of-growth notation 389