AFRL-RI-RS-TR-2009-279
Final Technical Report
December 2009

HYBRID TECHNIQUES FOR OPTIMIZING COMPLEX SYSTEMS

University of Michigan

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2009-279 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
STEVEN L. DRAGER
Work Unit Manager

/s/
EDWARD J. JONES, Deputy Chief
Advanced Computing Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
DECEMBER 2009	Final	January 2006 – June 2009

4. TITLE AND SUBTITLE

HYBRID TECHNIQUES FOR OPTIMIZING COMPLEX SYSTEMS

5a. CONTRACT NUMBER
N/A

5b. GRANT NUMBER
FA8750-05-1-0282

5c. PROGRAM ELEMENT NUMBER
62702F

6. AUTHOR(S)

John P. Hayes and Igor L. Markov

5d. PROJECT NUMBER
459T

5e. TASK NUMBER
HT

5f. WORK UNIT NUMBER
OC

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Michigan
503 Thompson Street
Ann Arbor, MI 48109-1340

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RITA
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2009-279

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA# 88ABW-2009-5050 Date Cleared: 4-December-2009

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This is the final technical report for a three year research project on Hybrid Techniques for Optimizing Complex Systems conducted at the University of Michigan, Ann Arbor and sponsored by the Air Force Research Laboratory. The project's overall goal was to investigate novel hybrid techniques that combine concepts from quantum and classical computer science to solve hard computational problems, including the handling of uncertainty. The research problems considered include design optimization and simulation of conventional CMOS and quantum systems, fault tolerance, resource allocation and scheduling, strategy optimization and related challenges facing the Air Force. The research focuses on accurate modeling of practical metrics of performance, robustness and cost, and their optimization in both linear and non-linear domains, using fast exact and heuristic methods, along with highly efficient data representations. Errors in data and control due to environmental effects, as well as uncertainty in the problem formulation, are taken into account during system modeling and optimization.

15. SUBJECT TERMS
Hybrid Techniques, Quantum Computing, Design optimization, Simulation CMOS Circuits, Fault Tolerance, Resource Allocation, Strategy Optimization, Error Management

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Steven L. Drager
U	U	U	UU	101	19b. TELEPHONE NUMBER (*Include area code*) N/A

Table of Contents

List of Figures

1. Executive Summary

The overall objective of this project was to investigate novel hybrid techniques that combine concepts from quantum and classical computer science to solve hard computational problems, including the handling of uncertainty. The research problems considered include design optimization and simulation of conventional CMOS and quantum systems, fault tolerance, resource allocation and scheduling, strategy optimization and related challenges facing the Air Force. The research focuses on accurate modeling of practical metrics of performance, robustness, and cost, and their optimization in both linear and non-linear domains, using fast exact and heuristic methods, along with highly efficient data representations. Errors in data and control due to environmental effects, as well as uncertainty in the problem formulation, are taken into account during system modeling and optimization.

The project's main accomplishments include the following:

- An analytical study of probabilistic fault models in digital logic and their impact on overall circuit performance. Probabilistic faults affect electronics in high-altitude and high-radiation environments, especially state-of-the-art deep-submicron CMOS chips.
- New algorithmic methodologies and tools for circuit synthesis and test to mitigate the impact of probabilistic faults. These methodologies include fast evaluation of circuit reliability based on functional simulation, and incremental modification of a circuit to improve its robustness.
- Analysis of mobile (wireless) ad hoc communication networks, focused on network throughput and total power consumption.
- A non-linear programming framework for spatial optimization of mobile networks, its empirical evaluation, and visualization of results.
- A new algorithm to simulate quantum circuits which exhibits polynomial-time performance in several important cases. This algorithm and accompanying theoretical results have been subsequently used by other researchers to show that the Quantum Fourier Transform (QFT) can be simulated in polynomial time on conventional computers.
- Several techniques for verification of correctness of quantum circuits. These techniques are based on computational engines frequently used in Electronic Design Automation Boolean satisfiability (SAT) and binary decision diagram (BDD), fall under the category of equivalence-checking, and can verify the results of adapting known circuits to specific device architectures, such as linear ion traps.

2. Introduction

This project encompasses several topics of interest to the Air Force—from near-term to long-term—that concern hybrid classical-quantum systems affected by uncertainty. The work reported includes algorithmic techniques and methodologies to simulate, compare and evaluate hybrid systems, as well as to optimize them for robustness, performance and resource utilization.

2.1. Topics Addressed

One of the near-term efforts focused on probabilistic (non-deterministic) faults and errors in electronic devices, particularly in semiconductor chips that are manufactured with increasing device densities and miniature devices susceptible to transient particle strikes. Our results include an extensive suite of algorithmic techniques to represent transient faults and quickly evaluate their impact on a large circuit. We also developed a methodology for hardening a given circuit by inserting a small amount of redundancy in carefully chosen sections of the circuit. This methodology improves robustness of the circuit, while decreasing area and power overhead. We also carried out one of the first studies of circuit testing for non-deterministic faults. Observing that existing test patterns may have to be repeated in order to observe transient faults, we developed algorithms for calculating replication factors for given tests.

A second near-term effort focused on mobile (wireless) ad hoc networks (MANETs), whose structure can be determined by the locations and power levels of nodes and relays. We studied the impact of these locations on network throughput and total power consumption, as well as run-time reconfiguration of MANETs. To this end, we developed a non-linear programming framework which determines locations so as to achieve the best compromise between resource consumption and network throughput. The technique was evaluated on a number of realistic test-cases under dynamic scenarios where network parameters change. The incremental changes in network structure can be visualized and communicated to a human operator in real time.

Longer-term topics studied in our work include algorithms for simulation and comparison of quantum circuits. This research pursued several complementary goals, such as exploring the limitations of quantum computing, developing engineering tools to aid in the construction of prototype quantum processors, and exploring possible applications of quantum techniques. In particular, we developed new algorithms for simulating quantum circuits that achieve polynomial-time efficiency in important cases, such as all depth-three quantum circuits and certain approximate circuits for the Quantum Fourier Transform. These results show that more sophisticated quantum circuits are required to achieve computational speed-ups.

In conjunction with the need to optimize quantum circuits to particular device architectures, such as linear ion traps, we developed several techniques for quantum circuit verification. One of these is based on the quantum information decision diagram (QuIDD) data structure introduced in our past DARPA-funded project. This technique can handle several phase-equivalence relations relevant to quantum circuits, but is relatively slow. Another technique is the use of quantum and reversible miters, in conjunction with circuit simplification. It can significantly reduce the complexity of quantum verification instances and is compatible with other techniques, including QuIDD-based methods. A third technique is based on Boolean satisfiability, and capitalizes on the ongoing success of SAT techniques in industrial verification of digital logic.

We also started exploring simulation of atomic-scale systems represented by Ising models, especially in conjunction with number-factoring through quantum adiabatic optimization. To this end, we developed two techniques for energy minimization in Ising spin-glasses. One is a branch-and-bound algorithm that finds ground states on 100 spins in one day, and one is a local-search heuristic that approximates ground states on 1,000,000 spins in one day. Both can be easily parallelized to multi-core CPUs and distributed systems.

2.2. Project Participants

Faculty at the University of Michigan who led this research were Professor John Hayes and Professor Igor Markov. Current and former graduate students participating in this project at the University of Michigan included Dr. Smita Krishnaswamy, Dr. George Viamontes, and Héctor García. Lastly, Dr. Ilia Polian from the University of Freiburg and Professor Shigeru Yamashita of the Ritsumeikan University collaborated with us.

3. Logic Circuits Subject to Uncertainty

3.1. Summary

Integrated circuits (ICs) are becoming increasingly susceptible to uncertainty caused by soft errors, inherently probabilistic devices, and manufacturing variability. To address these issues, we have developed methods for analyzing, designing, and testing circuits subject to probabilistic effects. The main contributions of this work are: fast, soft-error rate (SER) analysis methods and software tools that use functional-simulation signatures to capture error effects, novel design techniques and software tools that improve reliability using little area and performance overhead, a matrix-based reliability-analysis framework (probabilistic transfer matrices or PTMs) that can capture many types of probabilistic faults, and test generation and compaction methods aimed at probabilistic faults in logic circuits.

Further details concerning the material in this chapter can be found in Smita Krishnaswamy's 2008 Ph.D. dissertation [12] and related publications [13-21].

3.2. Introduction

Digital systems have always been vulnerable to a variety of manufacturing and wearout defects. Over time, IC technology scaling has increased device sensitivity to soft errors caused by external noise or radiation that temporarily affects circuit behavior without permanently damaging the hardware. With the advent of nanoscale computing, soft errors are beginning to affect not only memory but also combinational logic. Unlike memory, errors in combinational logic cannot be easily corrected and can lead to potentially disastrous failures in error-critical systems such as aircraft, medical devices, and servers. New device technologies such as carbon nanotubes and quantum computers exhibit inherently probabilistic behavior due to quantum-mechanical effects. Resilience under these sources of uncertainty is vital for continued technology and performance improvements. Due to the high cost and power consumption of ICs, the widespread addition of redundancy is not a practical option for curtailing error rates. Instead, careful circuit analysis and low-cost methods of improving reliability are necessary. Further, circuits must be tested post-manufacture for their vulnerability to both transient faults and manufacturing defects.

A soft error is a signal that has an incorrect logic value but does not imply a permanent defect. Such errors are one of the main causes of uncertainty and failure in logic circuits [34]. They can be caused by cosmic rays, α-particles, and even thermal noise. When a particle strikes the sensitive area of a logic gate, it can cause an ionized track known as a single-event upset (SEU), as shown in Figure 1. An SEU is a transient or soft fault, as opposed to a permanent fault. SEU effects do not propagate if the charge deposited is below the critical charge Q_{crit} required to switch the corresponding transistor on or off. If an SEU deposits enough charge to cause a spurious signal pulse or *glitch* in the circuit, it produces a soft error. Error propagation from the fault site to a flip-flop or primary output is stopped if there is no sensitized path for error propagation. If a soft error is transmitted to and captured (latched) by a flip-flop, it can persist in a system for many clock cycles.

A single latched error can also fan out to multiple flip-flops. Unlike errors in memory, errors in combinational logic cannot be rectified by error-correcting codes without incurring significant area overhead. Hence, it becomes vital to find ways to accurately analyze and decrease the soft-error rate of a circuit through careful design

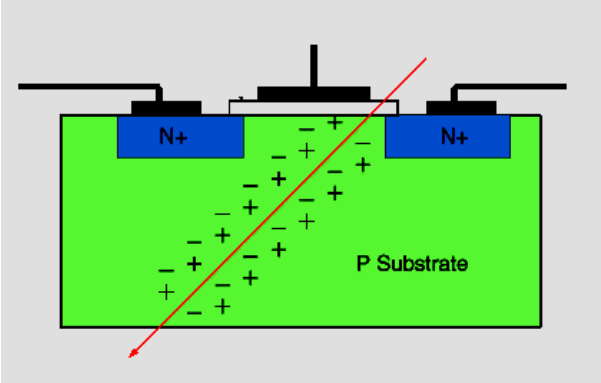


Figure 1. Ionized track in a transistor caused by cosmic radiation [34].

3.3. Soft-Error Rate Estimation

Several factors determine the SER of a logic circuit. An SEU must have sufficient energy to change a signal and propagate the erroneous signal value through subsequent gates; if not, the fault is *electrically* masked. The signal change must propagate through the logic to affect a primary output; if not, the fault is *logically* masked. The fault must reach a flip-flop during the sensitive portion (latching window) of a clock cycle; if not, the fault is *temporally* masked. These various masking effects depend on the characteristics of the gates encountered by a fault on its way to the primary outputs, as well as on the particular paths taken. Any path taken must have non-controlling values on side inputs, so different input vectors can sensitize different sets of paths.

The SER can be computed using the basic algorithm shown in Figure 2. Here, P_{err} is the probability of an error on a signal node. It is computed using the following variables: $P(i)$, the probability of vector i being applied to the input; $P_{strike}(n)$, the probability of a fault at n; $P_{attenuate}(path(p))$, the probability of attenuation along path p; and $P_{latch}(p;o)$, the probability of an error on p arriving at output o during clock latching. Neglecting to model any of these factors leads to overestimation of the SER. This algorithm is only practical for the smallest of circuits, as the number of possible sensitized paths grows exponentially in the size of the circuit [30]. Therefore, even determining the probability of logical masking is NP-hard.

```
compute_SER(circuit C)
{
 for(input vector i)
  for(node n ∈ C)
    for(output o ∈ C)
        for(sensitized path p ∈ path(i,n))
          P_prop(n) = (1 − P_attenuate(p))
          Perr(C)+ = P(i)P_strike(n)P_prop(n)P_latch(p,o)
 return Perr(C)
}
```

Figure 2. Basic SER computation algorithm.

Several software tools exist for estimating the SER of combinational circuits. Soft-error rate analysis (SERA) [38] follows Figure 2 and, using designer-specified input vectors, finds all paths from each gate to an output. SEU-induced glitches are simulated on representative inverter chains of the same lengths as the target paths to determine the probability of electrical masking. Fast analysis of soft-error (FASER) [39] uses binary decision diagrams to enumerate all possible input vectors. A BDD is created for each gate in a circuit: a static BDD for gates outside a glitch's cone of influence, and duration and amplitude BDDs for gates within that cone. The BDDs are combined in a way that allows the width and amplitude of glitches to be systematically analyzed with respect to electrical masking. FASER's BDD representations can consume a lot of memory space. Single event transient (SET) [31] proceeds in topological order and considers each gate only once. For each gate, SET encodes the probability and shape of a glitch as a Weibull probability density function called a SER descriptor (SERD). The Weibull parameters are modified at each gate to account for electrical attenuation, and the new output SERDs are passed on to succeeding gates. The SET algorithm is similar to static timing analysis (STA) and so does not consider false paths. Figure 3 summarizes the characteristics of the tools described above, as well as their methods for incorporating masking mechanisms. Because of their different assumptions and vastly different ways of computing SER, they can yield very different SER values for the same circuit.

Attribute	SERA	FASER	SET
Logic masking	Vector simulation	BDD-based analysis	Vector simulation
Timing masking	SER derating	No details given	SER derating
Electrical masking	Inverter-chain simulation	Gate characterization	Gate characterization
Fault assumptions	Single	Single	Multiple

Figure 3. Summary of differences between three SER evaluation tools.

Our work aims to build SER analysis tools that are scalable and can be used early in the logic design phase [15, 18, 19]. Due to our emphasis on reliability-driven logic design, we focus on modeling logical masking both accurately and efficiently. We then use our tools to guide several design techniques to improve circuit resilience against soft errors.

3.3.1. Reliable Design Methods

Techniques for transient-fault tolerance have been developed for use at nearly all stages of the design flow. They rely on enhancing masking mechanisms to mitigate error propagation. Here, we discuss several techniques and highlight their masking mechanisms. Faults can be detected at the architectural level via some costly form of redundancy and can be corrected by rolling back to a checkpoint to replay instructions from that checkpoint. Techniques that involve replicating an entire circuit increase chip area significantly and, therefore, decrease chip yield. For example, Mohanram and Touba [26] proposed to partially triplicate logic by selecting regions of the circuit that are especially susceptible to soft errors. Such regions are selected by simulating faults with random test vectors. More recently, Almukhaizim et al. [2] presented a design modification technique, called rewiring, to increase reliability. In the spirit of [2], our work focuses on lightweight modifications that increase reliability without requiring large amounts of redundancy.

Chip manufacturers routinely test their chips for SER [23, 36, 37]. This is normally accomplished through field testing or accelerated testing. In field testing, many devices are connected to testers and evaluated over several months under normal operating conditions. In accelerated testing, devices are irradiated with neutron or α-particle beams, thus shortening test time to a few hours. There is some difficulty, however, in translating the SER obtained by accelerated testing to that of field testing [11]. For instance, intense radiation can cause multiple simultaneous errors, triggering system failures more often than normal.

As an alternative to field testing, Hayes, Polian and Becker [9] propose a non-concurrent built-in self-test (BIST) architecture for online testing. They define the impact of various soft faults on the circuit in terms of frequency, observability, and severity. For instance, more frequent and observable faults are considered more influential than rare faults. With this fault characterization, integer linear programming (ILP) is used to generate tests for various objectives, such as ensuring a minimum fault-detection probability. Researchers have sought to accelerate testing by employing test patterns that sensitize faults. Conceptually, the main difference between testing for hard errors rather than soft errors is that soft errors are only present for a fraction of the test time. Therefore, test vectors must be repeated to detect faults, and they must be selected to sensitize the most frequent faults. Sanyal et al. [33] accelerate testing by selecting a set of error-critical nodes and deriving test sets that, using ILP, sensitize the maximum number of these faults. In our work, which preceded [33], we developed a way of identifying error-sensitive test vectors for multiple faults, and we devised algorithms for generating test sets to accelerate SER testing [13, 14].

3.3.2. Probabilistic Circuit Analysis

Circuit design and testing calls for new types of probabilistic analysis that go beyond soft error analysis only. In our work, we developed a novel probabilistic matrix-based model for gates, and we use matrix operations and symbolic methods to evaluate overall circuit error probabilities, as discussed later. In earlier work, Bahar et al. [5] propose to model and design carbon nanotube (CNT)-based neural networks using Markov random fields (MRFs). MRFs specify joint probability distributions in terms of local conditional probabilities, but they can also describe cyclic dependencies. A neural network is described by an MRF with node values computed by a weighted sum of conditional probabilities of a neighboring clique of nodes. This is known as the Gibbs formulation and lends itself to optimizing for clique energy, which is translated into low node error probabilities in [5]. Related to this, Nepal et al. [27] present a method for implementing MRF-based circuits in CMOS, while Bhadhuri et al. [4] describe a software tool called Nanolab, which uses the algorithm from [5] to automate the design of fault-tolerant architectures.

In more recent work than our's, Rejimon and Bhanja [32] propose capturing errors in nanoscale circuits by means of Bayesian networks. (A Bayesian network is a directed graph with nodes representing variables and edges representing dependence relations among the variables.) If there is an edge from node *a* to node *b*, then *a* is a parent of *b*. The joint probability distribution for *n* variables x_1:x_n is represented as the product of the conditional probability distributions:

$$\Pi_{i=1}^{n} P[x_i | parents(x_i)]$$

To carry out numerical calculations on a Bayesian network, each node x_i is labeled with a probability distribution, conditioned on its parents. The probability distribution can be given in tabular form. Primary inputs are given pre-defined probabilities and the probabilities of other nodes are then computed using a method called belief propagation. Joint probabilities are computed in large Bayesian networks using sampling methods such as *importance sampling*. Many tools [28] exist for Bayesian network analysis.

3.4. Signature-based Soft-error Analysis

As discussed above, it is important to be able to efficiently and accurately analyze SERs during the actual design process. We now present our SER analyzer called analysis of soft-error rate (AnSER). This tool uses functional simulation to estimate logic masking and to account for the input-vector dependence in timing and electrical masking. Since exact analysis is impractical for all but the smallest of circuits, we estimate parameters like signal probability and observability, which are closely connected to the probability of error propagation, using a new and very efficient signature-based method.

Figure 4 illustrates the flow of computation in AnSER. Functional-simulation signatures are computed from logical information, error-derating factors from gate-characterization information, and error-latching windows from static-timing analysis. These smaller computations are combined to form an estimate of circuit SER. Since AnSER is intended to be used alongside logical and physical design tools, we pay particular attention to runtime, memory requirements, and the incremental-use model.

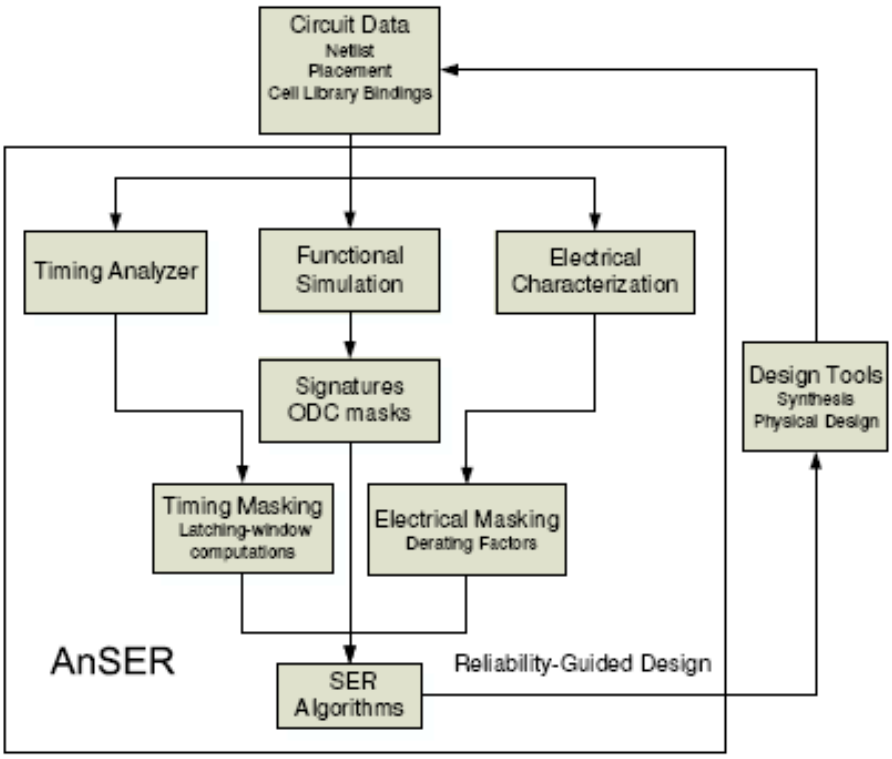


Figure 4. Overall design flow of AnSER.

The remainder of this chapter develops our method for computing the SER of logic circuits by accounting for logic masking, extends this methodology to sequential circuits, and incorporates timing and electrical masking into our SER estimates. Further details of the techniques and results appear in [12].

3.4.1. SER in Combinational Logic

We now consider a SER analysis method for combinational logic which, by definition, contains no memory. We first develop fault models for soft errors. Then, we provide background on functional-simulation signatures, which are used extensively in AnSER. Next, we derive SER algorithms for single and multiple fault assumptions using signal probability and observability measures that are computed using signatures.

3.4.2. Faults, Signatures and Observability Don't Cares

We first formulate a model for transient faults that is suitable for SER analysis. Formally, a *transient stuck-at* (TSA) fault is a triple (g, v, $Perr(g)$) where g is a circuit node, v is a logic value, and $Perr(g)$ is the probability per clock cycle of a stuck-at-0/1 fault when the node has correct value v. The advantage of basing a fault model on the standard stuck-at (SA) model for permanent faults is that the same automatic test pattern generation (ATPG) tools can be used for SA and TSA faults. The TSA fault model assumes that at most one fault occurs in any clock cycle. This assumption is common in SER research because for most technologies, the intrinsic error rate is very low. The contribution of each gate to the SER depends on the SEU rate of the particular gate, as captured by $Perr(g)$ and on the observability of the error. The TSA model can readily be extended to several types of multiple transient multiple faults.

A *signature*, denoted, $sig(g) = F_g(X_1)\, F_g(X_2)\ldots F_g(X_K)$, is the sequence of logic values observed at node g in response to applying a sequence of K input vectors $X_1, X_2\ldots X_K$ to the circuit. We use node signatures for three purposes: to compute the SER, to identify error-sensitive areas of the circuit, and to identify redundant nodes for resynthesis. Here, $F_g(X_i)$ indicates the value appearing at g in response to X_i, so the signature partially specifies the Boolean function F_g realized by g. Applying all possible input vectors (exhaustive simulation) generates a signature that corresponds to a full truth table. In general, $sig(g)$ can be seen as a kind of "supersignal" composed of individual binary signals that are defined by some current set of vectors. Like the individual signals, $sig(g)$ can be processed by EDA tools such as simulators and synthesizers as a single entity. This processing can take advantage of bitwise operations available in CPUs to speed up the computation compared to processing the signals that compose $sig(g)$ one at a time.

Signatures with thousands of bits can be useful in pruning non-equivalent nodes during equivalence checking [29, 40]. A related speedup technique is also the basis for "parallel" fault simulation [7]. Figure 5 shows a 5-input circuit where each of the 10 nodes is labeled by an 8-bit signature computed with eight input vectors. These vectors are randomly generated, and conventional functional simulation propagates signatures to the internal and output nodes. In a typical implementation such as ours, signatures are stored as logical words and manipulated with 64-bit logical operations, ensuring high simulation throughput. Therefore 64 vector simulations are conducted in parallel with each signature processed. Generating K-bit signatures in an N-node circuit takes $O(NK)$ time.

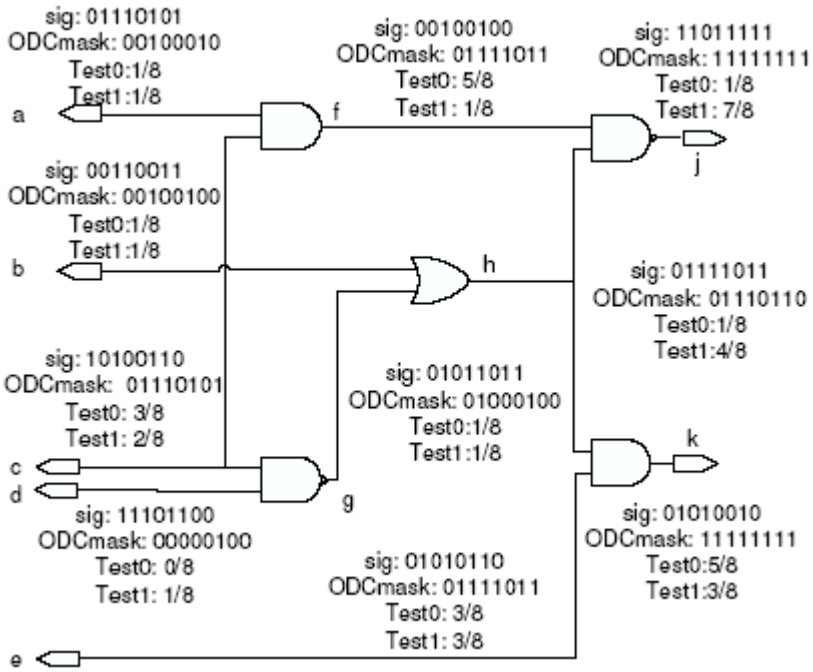


Figure 5. Signatures, ODC masks and testability information in a small circuit.

Observability don't-cares (ODCs) occur at a node *g* for input vectors for which the value at *g* does not affect the primary outputs. For example, in the circuit AND(*a*; OR(*a*,*b*)), the output of the OR gate is inconsequential when $a = 0$. Hence, input vectors 00 and 01 are ODCs for *b*. Corresponding to the *K*-bit signature *sig*(*g*), the ODC mask of *g* is the *K*-bit sequence whose *i*-th bit is 0 if input vector X_i, is in the don't-care set of *g*; otherwise the *i*-th bit is 1, that is, $ODCmask(g) = X_1 \notin ODC(F_g)\ X_2 \notin ODC(F_g) \ldots X_K \notin ODC(F_g)$. The ODC mask is computed by bitwise inverting *sig*(*g*) and re-simulating through the fan-out cone of *g* to check if the changes are propagated to any of the primary outputs. ODC masks can be computed exactly in time $O(N^2)$ for a circuit with *N* gates. We found the faster heuristic algorithm presented in [29], which has only O(*N*) complexity, convenient to use. Figure 5 shows a sample 8-bit signature and the accompanying ODC mask for each node of the example 10-node circuit.

3.4.3. SER Evaluation

We compute the SER by counting the number of test vectors that propagate the effects of a transient fault to outputs. Figure 6 summarizes our algorithm for SER computation. It involves two traversals of the target circuit: one to propagate signatures forward and another to propagate ODC masks backwards. The fraction of 1s in a node's signature is an estimate of its signal probability, while the relative proportion of 1s in an ODC mask indicates observability. The two measures are combined to obtain a testability figure-of-merit for each node of interest, which is then multiplied by the probability of the associated TSA to obtain the SER for the node. This approach can be contrasted with technology-dependent SER estimates, which include timing and electrical masking.

```
compute_TSA_SER(Circuit C, int K)
{
 compute_sigs(C,K)
 compute_odc_approx(C,K)
 for(all nodes g ∈ C)
  test0(g) = zeros(sig(g)&ODCmask(g))/K
  test1(g) = ones(~sig(g)&ODCmask(g))/K
  Perr(C)+ = Perr0(g)test1(g)
  Perr(C)+ = Perr1(g)test0(g)
 return Perr(C)
}
```

Figure 6. Algorithm to compute SER under the TSA fault model.

We estimate the probability $P[g = 1]$ of signal g having logic value 1 by the fraction of 1s in the signature $sig(g)$; this is often called the *controllability* of g. The *observability* $P[obs(g)]$ of g is the probability that a change in the signal's value changes a primary output, and is approximated by the number of 1s in g's ODC mask. The 1-*testability* of g is the probability that its correct value is 1 and that it is observable.

$$P[test1(g)] = \text{ones}(sig(g)\&ODCmask(g)/K$$

Similarly, *0-testability* is the number of positions where the ODC mask is 1 and the signature is 0. Consider again the circuit in Figure 5. For node g we have $sig(g) = 01011011$ and $ODCmask(g) = 01000100$. Hence, $P[g = 1] = 5/8$, $P[g = 0] = 3/8$, $P[obs(g)] = 2/8$, and $P[test0(g)] = P[test1(g)] = 1/8$.

In a circuit C, we sum the SER contributions of the gates, and weight the gate error probabilities by the testability for each TSA fault. Hence, we can write:

$$Perr(C) = \sum_{g \in C} P[test_1(g)]Perr_0(g) + P[test_0(g)]Perr_1(g)$$

For example, if each gate in Figure 5 has TSA-1 probability $Perr0 = p$ and TSA-0 probability $Perr1 = q$, then the SER is given by $Perr(C) = 2p + (13/8)q$. The metrics $test_0$ and $test_1$ implicitly incorporate error sensitization and propagation conditions. Hence, the last equation accounts for the possibility of an error being logically masked.

3.5. SER in Sequential Logic

We can extend our SER analysis to handle sequential circuits, which contain state storage elements (D flip-flops). The combinational logic computes state information and primary output values as a function of the current state and primary inputs. Three factors to consider while analyzing sequential-circuit reliability are: steady-state probability distribution, state reachability, and sequential observability.

3.5.1. Steady-State and Reachability Analysis

In order to approximate the steady-state distribution, we perform sequential simulation using signatures. Assume that a circuit with m flip-flops $L = \{l_1,l_2,\ldots l_m\}$ is in state $S_L = \{s_1,s_2,\ldots s_m\}$. Our method starts in state S_0 for each simulation run (sets of 64 states are processed in parallel), then we simulate the circuit for n cycles. Each cycle propagates signatures through the combinational logic and stops when flip-flops are reached. Primary input values are generated randomly from some fixed probability distribution. At the end of each simulation cycle, flip-flop inputs are transferred to flip-flop outputs, which are, in turn, fed into combinational logic for the subsequent cycle. All intermediate signatures are erased before the next simulation cycle starts. The K-bit signatures of the flip-flops at the end of n simulations cycles define K states. We claim that for large enough n, these states are sampled from the steady-state probability distribution. Empirical results suggest that most ISCAS benchmarks reach steady-state in 10 or fewer cycles, under the above operating conditions [25].

Additionally, our signature-based SER analysis methods can handle systems that are decomposable. Such systems pass through some transient states and are then confined to a set of strongly connected closed (SCC) states. That is, the system can be partitioned into transient states and sets of SCC states. For such systems, the steady-state distribution strongly depends on the initial states. We address this implicitly by performing reachability analysis starting at a reset state. Thus, each bit of the signature corresponds to a simulation that starts from a reset state and propagates through the combinational logic, moves to adjacent reachable states, and, for a large enough n, reaches steady-state within the partition.

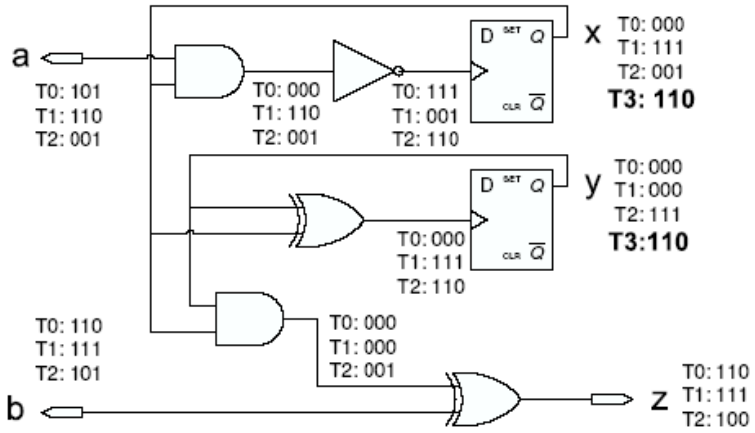


Figure 7. Illustration of bit-parallel sequential simulation.

Using our algorithm, simulating a circuit with g and K-bit signatures for n simulation cycles takes time $O(Kng)$. Note that it does not require matrix-based analysis, which is often the bottleneck in other methods. For example, Markov matrices are used to encode state-transition probabilities explicitly, and so can be large due to state-space explosion [25]. Figure 7 shows an example of sequential simulation with 3-bit signatures. The flip-flops with outputs x and y are initialized to 000 in cycle $T0$, then the combinational logic is simulated. For cycle $T1$, the inputs of x and y are transferred to the output, and the process continues. At the end of the simulation, the values for x and y at $T3$ are saved for sequential-error analysis, as explained below.

3.5.2. Error Persistence and Sequential Observability

To assess the impact of soft faults on sequential circuits, we analyze several cycles through which faults persist, using time-frame expansion. This means making n copies of the circuit $C_0, C_1, \ldots C_{n-1}$, thereby converting a sequential circuit into a pseudo-combinational one with n time frames. The outputs of the flip-flops of the k-th frame are connected to the primary inputs of frame $k + 1$ for $0 < k < n - 1$. Flip-flop outputs that feed into the first frame ($k = 0$) are treated as primary inputs, and flip-flop inputs of frame n are treated as primary outputs. Figure 8 shows a three-time-frame circuit that corresponds to Figure 7. Note how new primary inputs and outputs are created, corresponding to the inputs from flip-flops for frame 0 and outputs of flip-flops for frame 3. Intermediate flip-flops are represented by buffers.

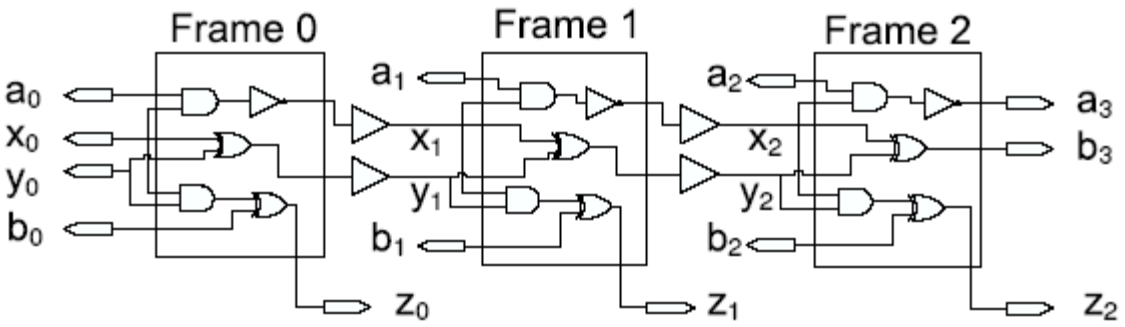


Figure 8. Illustration of time-frame expansion into three frames: C_0, C_1, C_2.

Observability is analyzed by considering all *n* frames as a single combinational circuit, thus allowing the single-fault SER analysis described in the previous section to be applied to sequential circuits. Other useful information, such as the average number of cycles during which faults persist, can also be determined using time-frame expansion. After the multi-cycle sequential simulation described above we store the signatures of the flip-flops and use signatures to stimulate the newly created primary inputs in the time-frame expanded circuit. For instance, the x_0 and y_0 inputs of the circuit in Figure 4 are simulated with the corresponding signatures, marked *T*3 (the final signature after multi-cycle simulation is finished). Randomly generated signatures are used for primary inputs not corresponding to flip-flops (such as a_0 and b_0 in Figure 8).

After simulation, we perform ODC analysis starting from the primary outputs and flip-flop inputs of the *n*-th frame C_n and moving all the way to the inputs of the 0-th frame. In other words, errors in primary outputs and flip-flops are considered to be observable. Figure 9 gives our algorithm for sequential SER computation. The value of *n* can be varied until the SER stabilizes, i.e., does not change appreciably from frame *n* to frame *n* +1. The *n*-frame ODC-analysis can lead to different gates being seen as critical for SER. For instance, the designer can deem errors that persist longer than *n* cycles as more critical than errors that are quickly flushed at primary outputs. In this case, the ODC analysis only considers the fan-in cones of the primary outputs of C_n.

```
compute_seq_SER(Circuit C,int K,int n,int f)
{
 for(i < n)
  seq_simulate(C,K)
 C' = time_frame_expand(C,f)
 copy_flipflop_inputs(C',C)
 compute_sigs(C',K)
 compute_odc_approx(C',K)
 for(all nodes g ∈ C0)
  test0(g) = zeros(sig(g)&ODCmask(g))/K
  test1(g) = ones(~sig(g)&ODCmask(g))/K
  Perr(C')+ = (Perr0(g)test1(g) + Perr1(g)test0(g))
 return Perr(C')
}
```

Figure 9. Algorithm to compute SER in sequential circuits under TSA faults.

The SER algorithm in Figure 6 runs in linear time with respect to circuit size, since each simulation is linear and ODC analysis (even with *n* time frames) runs in linear time as well. To capture electrical masking in AnSER, we derate gate-error probabilities by a factor dependent upon the characterization of successor gates. Previous research has shown that electrical masking eliminates low-energy SEUs in 3-4 levels of logic and has little effect thereafter [26]. This implies that considering paths of limited length starting from the gate in question is often sufficient to approximate this effect. We have also developed a linear-time algorithm in the spirit of static timing analysis for computing the error-latching window (ELW) of every gate in a circuit; see [12] for details.

3.5.3. Empirical Validation

We now report some empirical results for SER analysis using AnSER and our SER-aware synthesis techniques. The experiments were conducted on a 2.4 GHz AMD Athlon 4000+ workstation with 2GB of RAM. The algorithms were implemented in C++. For validation purposes, we compare AnSER with complete test-vector enumeration using the ATPG tool ATALANTA [22]. We provided ATALANTA with a list of all possible SA faults in the circuit to generate tests in "diagnostic" mode, which calculates all test vectors for each fault. We used an intrinsic gate-fault value of $gerr0 = gerr1 = 1 \times 10^6$ on all faults. Since TSA faults are SA faults that last only one cycle, the probability of a TSA fault causing an output error is equal to the number of test vectors for the corresponding SA fault, weighted by their frequency. Assuming a uniform input distribution, the fraction of vectors that detect a fault provides an exact measure of its testability. Then, we computed the SER by weighting the testability with a small gate fault probability. While the exact computation can be performed only for small circuits, Figure 10 suggests that our algorithm is accurate to about 3% for 2,048 simulation vectors.

Circuit	No. gates	ATALANTA	AnSER	% Error	AnSER Exact-ODC	% Error
c17	13	6.96E-7	6.96E-7	0.01	6.96E-7	0.01
majority	21	6.25E-6	6.63E-6	6.05	6.57E-6	4.87
decod	25	2.60E-5	2.62E-5	0.83	2.60E-5	0.83
b1	25	1.28E-5	1.31E-5	2.81	1.27E-5	0.78
pm1	68	2.86E-5	3.00E-5	4.70	2.97E-5	3.5
tcon	80	5.30E-5	5.39E-5	1.67	5.35E-5	0.94
x2	86	3.78E-5	3.87E-5	2.38	3.93E-5	3.97
z4ml	92	5.29E-5	5.37E-5	1.50	5.41 E-5	2.20
parity	111	7.60E-5	7.69E-5	1.24	7.71E-5	1.45
pcle	115	5.38E-5	5.34E-5	0.75	5.35E-5	0.56
pcler8	140	7.06E-5	7.24E-5	2.52	7.23E-5	2.41
mux	188	1.58E-5	1.38E-5	12.54	1.63E-5	3.16
Ave.				3.06		2.65

Figure 10. Comparison of SER (FIT) data for AnSER and ATALANTA.

We isolate the effects of two possible sources of inaccuracy: sampling inaccuracy, and inaccuracy due to approximate ODC computation. Sampling inaccuracy is due to the incomplete enumeration of the input space. Approximate ODCs computed using the algorithm from [29] incur inaccuracy due to mutual masking. When an error is propagated through two reconvergent paths, the errors may cancel. However, the results in Figure 10 indicate that most of the inaccuracy is due to sampling, not approximate ODCs. The last two columns, corresponding to exact ODC computation, show an average error of 2.65%. Therefore, only 0.41% of the error is due to the approximate ODC computation. On the other hand, while enumerating the entire input space is intractable, our use of bit-parallel computation enables significantly more vectors to be sampled than other techniques for the same computation time.

To characterize the gates in the circuits accurately, we adapted data from [31], where several gate types are analyzed in a 130nm, 1.2 V_{DD} technology via simulation program with integrated circuit emphasis (SPICE) simulations. We use an average SER value of $gerr0 = gerr1 = 8 \times 10^{-5}$ for all gates. The SER analyzers from [31, 38, 39] report error rates that differ by orders of magnitude, SERA tends to report error rates on the order of 10^{-3} for 180nm technology nodes, and FASER reports error rates on the order of 10^{-5} for 100nm. Furthermore, although our focus is logic masking, we also approximate electrical masking by scaling our fault probabilities at nodes by a small derating factor to obtain trends similar to those of [31]. In Figure 11, we compare AnSER and SERD when computing SER for inverter chains of varying lengths. Since one path is always sensitized in this circuit, it helps us estimate the derating factor.

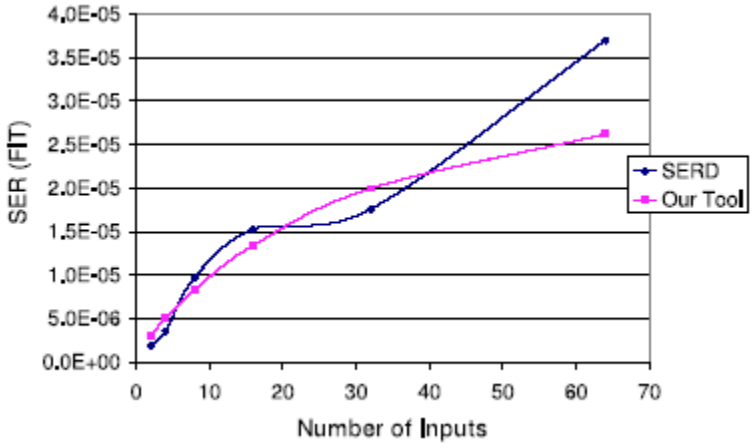


Figure 11. SER trends on inverter chains produced by SERD and AnSER.

Figure 12 compares with the previous work on some ISCAS-85 benchmarks. While the runtimes in [8] include 50 runs, the runtimes in [31] are reported per input vector. Thus, we multiply data from [31] by the number of vectors (2,048) used there; our runtimes appear better by several orders of magnitude. We believe that this is due to the use of bit-parallel functional simulation to determine logic masking, which has a strong input-vector dependency. Most other work uses fault simulation or symbolic methods.

Circuit	No. gates	Time (s)			
		AnSER	SERD[100]	FASER [135]	[24]
c432	246	**<0.01**	10	22	—
c880	591	**<0.01**	10	—	—
c1355	746	**0.014**	20	40	2.09
c1908	760	**0.015**	20	66	0.781
c3540	1951	**<0.01**	60	149	5m42s
c6280	4836	**1.00**	120	278	—

Figure 12. Runtime comparisons of four SER analyzers.

Figure 13 shows changes in SER when timing masking is considered. Incorporating timing masking into SER is useful in guiding physical synthesis operations, while considering logic masking alone suffices for technology-independent logic synthesis.

Circuit	No. gates	Clock Period (s)	Logic SER (FIT)	Time (s)	Timing SER (FIT)	Time (s)	Potential % improvement
aes_core	20265	5.68E-07	0.1654	6	9.33E-05	3	37.57
spi	2998	3.19E-07	0.05722	1	4.23E-05	1	15.28
s35932	5545	6.18E-07	0.1363	2	6.03E-05	1	26.73
s38417	6714	3.56E-07	0.1360	2	1.22E-04	1	37.83
tv80	6802	6.79E-07	0.05602	2	2.64E-05	1	37.50
mem_ctrl	11062	6.44E-07	0.2185	2	8.45E-05	3	19.64
ethernet	36227	1.46E-06	0.7010	9	1.31E-04	9	91.68
usb_funct	10357	5.06E-07	0.1852	3	8.79E-4	3	36.59

Figure 13. SER evaluation with logic and timing masking.

In summary, efficient analysis methods are necessary for assessing and reducing the SER of a circuit. AnSER, our linear-time method for the logic-level soft-error analysis, achieves its low runtimes by functional simulation signatures, which enable a fast and accurate method for computing signal probability and observability, even in the presence of reconvergent fan-out. We analyzed sequential circuits using AnSER and employing multicycle simulation and time-frame expansion. In addition, we incorporated timing masking through error-latching windows which were computed using timing analysis information. Our results on the standard benchmarks generally showed 2 to 3 orders of magnitude speed-up over previous SER analyzers, and high accuracy when validated against the ATALANTA ATPG tool.

3.6. Design for Robustness

At the gate level, soft errors have traditionally been eliminated via costly time or space redundancy. However, as we show here, it is possible to achieve major improvements in reliability without resorting to massive redundancy. In combinational logic, an SEU only affects the primary outputs if it is propagated through the intermediate gates. A basic way that designers can improve a circuit's reliability is to ensure that faults are logically masked with high probability. We target logic and timing masking to obtain soft-error-tolerant circuits in the following ways: 1) by identifying and using partial redundancy already present within the circuit, to mask errors; 2) by selecting error-sensitive areas of the circuit for replication or hardening; 3) by generating many candidate rewrites for each subcircuit and selecting among them for improvements in area and SER; and 4) by increasing timing masking during physical design.

3.6.1. Signature-Based Design

We now describe a technique called signature-based design for reliability (SiDeR), which identifies redundancy already present in the circuit and utilizes it to increase logic masking. As previously discussed, signatures provide partial information about the Boolean function of a node. Therefore, candidate nodes with similar functionality can be identified by matching signatures. We exploit the fact that nodes need not implement identical Boolean functions to bolster reliability. Any node that provides predictable information about another can be used to mask errors. For instance, if two internal nodes x and y satisfy the property $(y = 1) \Rightarrow (x = 1)$, where $\Rightarrow$ denotes "implies", then y gives information about x whenever $y = 1$. More generally, if $f(x_0,x_1, x_2,\ldots x_n) = x$, then x can be replaced by f to logically mask errors that are propagated through x. However, errors at x are only masked in cases where x does not control f.

We can increase the number of potential candidates that can replicate x by taking ODCs into account. In terms of signatures, this corresponds to bitwise ANDing $sig(f)$ and $sig(x)$ by $ODCmask(x)$ to check for the following relation:

$$sig(f)\&ODCmask(x) = sig(x)\&ODCmask(x)$$

Figure 14(a) shows an example of replicated logic for node a, derived by utilizing don't-care values and signatures.

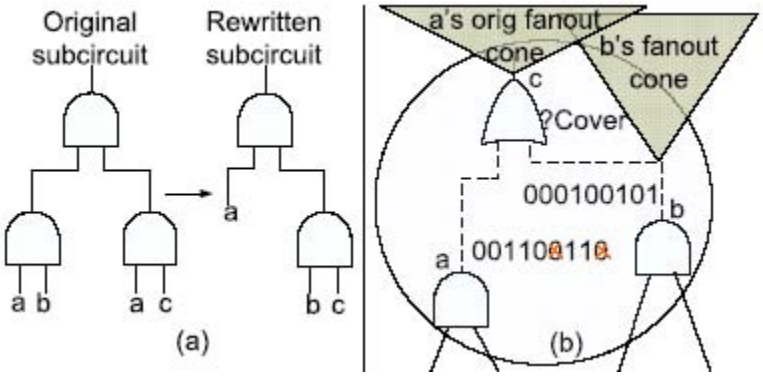


Figure 14. (a) Rewriting a sub-circuit to improve area, and (b) Finding a candidate cover for *a*.

In order to limit area overhead, the function f must be efficiently constructed from $x_0,x_1,\ldots x_n$. Therefore, we only consider cases where f is implemented by a single AND or OR gate. We add redundant logic by transforming node x into $OR(x; y)$. This means that either $(y = 1) \Rightarrow (x = 1)$ or $(x = 1) \Rightarrow (y = 1)$, which makes candidate pairs x and y easy to identify. When $OR(x; y) = x$, it follows that $sig(x) > sig(y)$ lexicographically; otherwise, $sig(y)$ is 1 in a position where $sig(x)$ is not. Therefore, sorting the signatures can narrow the search for candidate signals y. Also, $sig(x)$ must contain more 1s than $sig(y)$, so maintaining an additional list of size-sorted signatures and intersecting the two lists can prune the search. Multiple lexicographical sorts and multiple size sorts of signatures starting from different bit positions can further narrow the search. Consequently, signature-based redundancy identification can efficiently perform logic implication. Generally, several candidates satisfy implication relations for each node x.

Among the candidates, we choose a node *y* that most often controls the output of the additional OR/AND gate, and whose fan-in cone is maximally disjoint from that of *x*. Once we find candidates for resynthesis, a SAT solver is used to verify the implication relation. Our basic process for verifying circuit optimizations with SAT follows that of [6].

AnSER also makes use of local rewriting, a general synthesis technique that optimizes small subcircuits to obtain overall design improvements. Rewriting relies on the fact that different irredundant circuits corresponding to the same Boolean function can exhibit different properties. We optimize circuits for SER and area simultaneously by using AnSER to accept or reject rewrites, following the implementation of rewriting in [1, 24]. This technique first derives a 4-input cut for a selected node defining a one-output subcircuit. Next, replacement candidates are looked up in hash tables that store several alternative implementations of each function. To ensure global reliability improvement, we resimulate the circuit and update SER estimates. Computational efficiency is achieved through fast incremental updates. As shown in Figure 14(a), the original subcircuit with three gates can be rewritten with just two gates.

		With exact covers		With approx. covers	
Circuit	Area	% SER decrease	% Area increase	% SER decrease	% Area increase
cordic	84	1.7	1.2	27.3	45.2
b9	114	18.1	14.9	30.7	31.6
C432	215	37.6	14.0	38.7	14.9
C880	341	9.6	0.9	13.1	2.3
C499	432	1.0	3.2	32.2	20.6
C1908	432	5.9	9.0	32.4	24.1
C1355	536	25.3	9.0	30.7	8.6
alu4	740	55.9	0.9	55.9	1.6
i9	952	65.4	6.6	65.4	6.6
C3540	1055	31.1	2.2	49.4	3.6
dalu	1387	74.3	1.2	74.3	1.2
i10	2824	40.4	5.4	40.4	5.6
des	4252	11.4	2.9	26.7	4.4
Ave.		**29.1**	**5.5**	**39.8**	**13.1**

Figure 15. Improvements in SER obtained by SiDeR.

3.6.2. Empirical Validation

We now report some empirical results for the various design techniques presented in this section. Figure 15 shows SER and area overhead improvements obtained by SiDeR. The first set of results is for exact implication relationships, i.e., not considering ODCs. The second column shows the use of ODCs to increase the number of candidates. In both cases, AND/OR gates are added based on the functional relationship satisfied. We see an average 29% improvement in SER with only 5% area overhead without ODCs. The improvements for the ODC covers are 40% with area overhead of 13%, suggesting a greater gain per additional unit area than the partial triple modular redundancy (TMR) techniques in [26], which achieve a 91% improvement but increase area by 104% on average.

Figure 16 illustrates the use of AnSER to guide the local rewriting method implemented in the ABC logic-synthesis package [1]. AnSER calculates the global SER impact of each local change to decide whether or not to accept the change. After checking hundreds of rewriting possibilities, those that improve SER and have limited area overhead are retained. The data indicate that, on average, SER decreases by 10.7%, while area decreases by 2.3%. For instance, for alu4, a circuit with 740 gates, we achieve 29% lower SER while reducing area by 0.5%. Although area optimization is often thought to hurt SER, these results show that carefully guided logic transformations can eliminate this problem.

Circuits	No. gates	No. rewrites	% SER decrease	% Area decrease	Time (s)
alu4	740	13	29.3	0.5	24.5
b1	14	0	0.0	0.0	0.2
b9	114	8	6.8	0.9	0.3
C1355	536	97	1.2	9.0	37.6
C3540	1055	23	5.8	0.9	51.5
C432	215	68	5.5	1.4	12.1
C499	432	37	0.0	0.5	13.0
C880	341	7	0.2	0.0	5.4
cordic	84	5	1.2	1.2	0.5
dalu	1387	58	24.0	3.2	35.0
des	4252	282	11.2	0.1	12.3
frg2	1228	96	27.9	2.0	8.9
i10	2824	143	5.0	0.6	16.7
i9	952	83	31.4	11.7	35.3
Ave.			**10.7**	**2.3**	**18.1**

Figure 16. Improvements in SER obtained with local rewriting.

3.7. Probabilistic Transfer Matrices

We now move to a more general reliability-analysis framework that treats circuits entirely probabilistically. While this is useful for analyzing soft errors, it is also useful for analyzing devices that periodically fail or behave probabilistically during regular operation. Examples of such devices include probabilistic CMOS, molecular logic circuits, and quantum computers. In general, accurate reliability analysis involves computing not just a single output distribution but, rather, the output error probability for each input pattern. In cases where each gate experiences input-pattern dependent errors, even if the input distribution is fixed, simply computing the output distribution does not give the overall circuit error probability. For instance, if an XOR gate experiences a bit-flip error, then the output distribution is unaffected, but the wrong output is paired with each input. Therefore, we need to separately compute the error associated with each input vector.

3.7.1. PTM Basics

We analyze non-deterministic circuit behavior using a representation we introduced [20, 21] called the probabilistic transfer matrix (PTM). A PTM for a gate or circuit gives the probability of each output combination, conditioned upon the input combinations. PTMs can model gates exhibiting varying input-dependent error probabilities. PTMs form an algebra, that is, a set closed under specific operations, where the operations in question are matrix multiplication and tensor products. These operations may be used to compute overall circuit behavior by combining gate PTMs to form circuit PTMs. Matrix products capture serial connections, and tensor products capture parallel connections. Also, PTM-based computations implicitly capture signal correlations that are caused by fan-out.

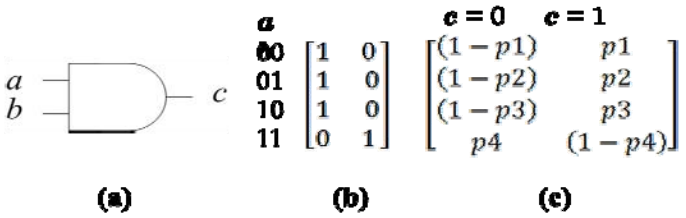


Figure 17. (a) Two-input AND gate, (b) Its ITM, and (c) A PTM with various error probabilities for each input vector.

A PTM for an n-input m-output component is a $2^n \times 2^m$ matrix M whose $(i,j)^{th}$ element is the probability of output j occurring in response to input i. The PTM of a fault-free component is called its ideal transfer matrix (ITM) and the probability of every correct output value is 1. Figure 17 shows a two-input AND gate, its ITM, and a PTM with different probabilities of erroneous output for each input combination. For example, the probability that $\{a, b\} = (1,1)$ yields the wrong output $c = 0$ is $p4$.

PTM algorithms involve several types of matrix operations, one of which is the tensor product. Given an $n \times m$ matrix A and a $p \times q$ matrix B, their tensor product $MT = A \otimes B$ is an $np \times mq$ matrix whose elements are:

$$MT(i_0 \ldots i_{n+p-1}, j_0 \ldots j_{m+q-1}) = A(i_0 \ldots i_{n-1}, i_0 \ldots j_{m-1}) \times B(i_n \ldots i_{n+p-1} \ldots j_{m+q-1})$$

The basic tensor operation on A and B needs $nmpq$ scalar multiplications, and consumes $nmpq$ memory for storing the results, for example,

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix} \otimes \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 & 0 & 2 & 4 \\ 3 & 4 & 0 & 0 & 6 & 8 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 3 & 4 & 0 & 0 \end{bmatrix}$$

Next we describe how the PTM of a *p*-level combinational circuit can be constructed from the PTMs of its component gates and wires. First, derive ITMs or PTMs for all components. Then for each topological level *i* containing *k* component PTMs $\{M_{ij}\}$, form the level PTM $M_i = M_{i1} \otimes M_{i2} \otimes \ldots M_{ik}$ by repeated application of the tensor product. Finally, using ordinary matrix multiplication multiply all *p* level PTMs together to form the circuit PTM $M = M_1 \cdot M_2 \cdot \ldots \cdot M_p$.

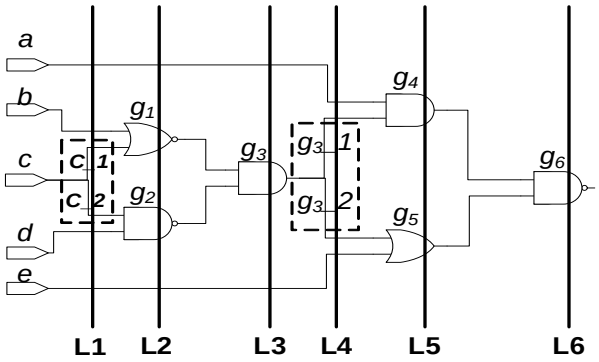


Figure 18. Circuit demonstrating PTM construction; dashed lines enclose fanout gates.

The six-level circuit *C* in Figure 18 shows how a circuit PTM is constructed. First, insert explicit wiring and fanout "gates" into *C* as needed. Then construct level PTMs M_1:M_6 for each level of logic. The PTM of gate g_i is denoted by G_i; PTMs of a single wire and an *n*-branch fanout gate are denoted by the identity matrix I_2 and F_n, respectively in the following symbolic representations: $M_1 = I_2 \otimes I_2 \otimes F_2 \otimes I_2 \otimes I_2$; $M_2 = I_2 \otimes G_1 \otimes G_2 \otimes I_2$; $M_3 = I_2 \otimes G_3 \otimes I_2$; $M_4 = I_2 \otimes F_2 \otimes I_2$; $M_5 = G_4 \otimes G_5$; and $M_6 = G_6$. The final circuit PTM is:

$$M = M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5 \cdot M_6$$

which corresponds to the 32 × 2 matrix outlined in Figure 19.

Once the overall PTM is known, output signal probabilities can be calculated very easily by multiplying the input signal distribution (row) vector *V* by the circuit PTM *M* thus: $J = V \cdot M$. For example, if all gates in *C* have the same error probability, 0.1, and all input signal probabilities are 0.5, the output probability of a 1 is 0.81.

$$\begin{array}{c} abcde \\ 00000 \\ 00001 \\ \cdots \\ 11111 \end{array} \begin{array}{c} g_6\!-\!0 \\ \left[\begin{array}{c} p0_{00} \\ p0_{01} \\ \cdots \\ p0_{31} \end{array}\right. \end{array} \begin{array}{c} g_6\!-\!1 \\ \left.\begin{array}{c} p1_{00} \\ p1_{01} \\ \cdots \\ p1_{31} \end{array}\right] \end{array}$$

Figure 19. PTM structure of the circuit in Figure 18.

Directly constructing a circuit PTM from its level PTMs may consume a large amount of runtime and memory. To enhance the scalability of PTM algorithms, we proposed using algebraic decision diagrams (ADDs) to reduce the memory needed for storing matrices, and developed heuristics such as dynamic evaluation ordering and hierarchical estimation to avoid unnecessary matrix operations [12]. Such heuristics can reduce the computational complexity by several orders of magnitude. However, the modified PTM calculations are still restricted to relatively small circuits because they still construct the circuit PTM by multiplying level PTMs. In addition, PTM size is limited by the number of inputs and outputs of the circuit, so deriving a PTM from a large circuit without a simplification scheme such as circuit partitioning may be impractical.

3.7.2. Applications

Besides the basic operations of matrix multiplication and tensor product, we introduced the following three operations to increase the scope and efficiency of PTM-based computation:

eliminate variables: This computes the PTM of a subset of inputs or outputs, starting from a given PTM. It can also be used to compute the probability of error of individual outputs.

eliminate redundant variables: This eliminates redundant input variables that result from tensoring matrices of gates that are in different fan-out branches of the same signal.

fidelity: This measures the similarity between an ITM and a corresponding PTM. It is used to evaluate the reliability of a circuit.

The PTM model can represent a wide variety of faulty circuit behaviors, including both hard and soft errors. The fact that there are separate probabilities for each input and output, and the fact that they are propagated simultaneously make this possible. Figure 20 lists some error types that can be precisely represented by PTMs.

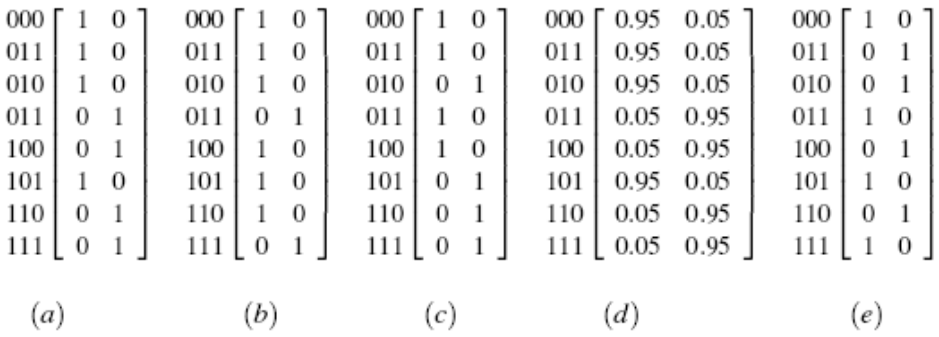
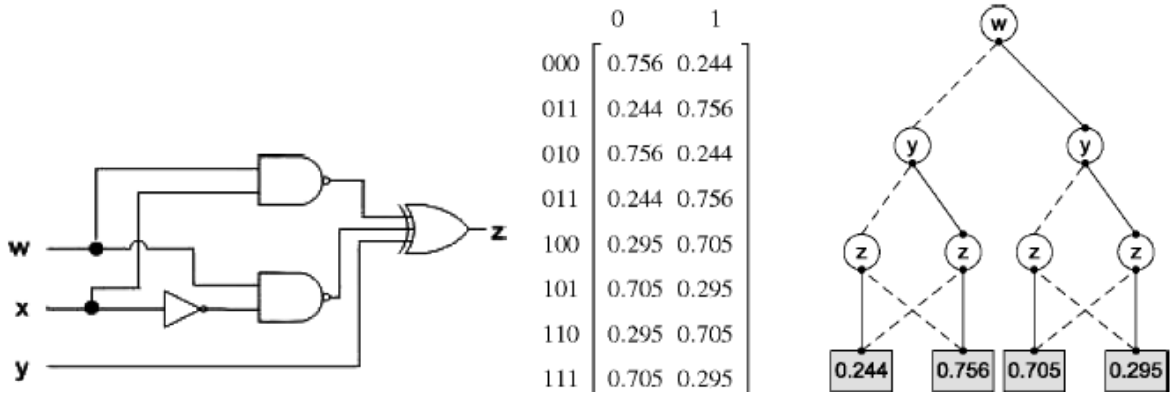


Figure 20. PTMs for several error types: (a) Fault free 2-1 MUX, (b) First input stuck-at 1, (c) Two inputs swapped, (d) Probabilistic output bit flip with *p*=0.05, and (e) MUX replaced by XOR.

We can also use PTMs to derive polynomial approximations for circuit error probabilities in terms of gate error probabilities for the purpose of determining thresholds of acceptable gate error for specific circuits [12].

3.7.3. Computing with PTMs

Circuit PTMs have exponential space complexity because they contain information about all possible input vectors. This complexity makes direct numerical computation with PTMs impractical for circuits with more than about 15 inputs. In order to improve scalability, we developed an implementation of the PTM framework that uses algebraic decision diagrams to compress matrices. We also derived several ADD algorithms to combine PTMs directly in their compressed forms. Figure 21(b) gives a PTM for the circuit in Figure 21(a) representing the case where all gates experience output bit-flips with probability $p = 0.05$. Figure 21(c) shows the corresponding ADD. As the latter figure indicates, the same values occur multiple times in the matrix and suggest a possibility of compression. Due to the canonicity of ADD/BDD representation, identical subgraphs, corresponding to identical submatrices, can be automatically identified and eliminated during the process of ADD construction. In some cases, ADDs contain exponentially fewer nodes than the number of entries in the explicit matrix representation. In such cases, linear-algebraic transformations can be applied exponentially faster to the ADD than to the matrix.



(a) (b) (c)

Figure 21. (a) Sample logic circuit, (b) PTM where each gate experiences error with probability p=0.05, and (c) ADD encoding of the PTM.

Developing efficient ADD algorithms for PTM operations is a significant technical challenge that we have addressed. We adapted previous ADD algorithms from [3] and [35] for tensor and matrix products. The original versions of their algorithms handle only square matrices, while PTMs are generally rectangular. In addition, we developed ADD algorithms for the new PTM operations defined earlier. These operations are needed for computing marginal-probability distributions, reconciling dimensions, and estimating overall circuit-error probabilities. For details of this work, see [12].

3.8. Testing for Probabilistic Faults

Circuits have to be tested in order to ensure that their soft error rates do not exceed an acceptable threshold. To estimate the expected soft error rate in the field, chips are typically exposed to intense beams of protons or neutrons and the resulting error rate is measured. However, these types of tests take a long time to conduct because random patterns may not be sensitive to vulnerabilities in the circuit. We have developed methods for selecting test vectors such that test application time is minimized [13, 14].

Generating tests for probabilistic faults is fundamentally different from existing testing techniques. Probabilistic testing requires a *multiset* (a set with repetitions) of test patterns, since a given fault is only present for a fraction of the computational cycles. Another difference is that some test vectors detect transient faults with higher probability than others due to path-dependent effects like electrical masking. Therefore, one can consider the likelihood of detection, or the sensitivity, of a test vector to a fault.

3.8.1. Test-Vector Sensitivity

The sensitivity $sens(F,t)$ of a test vector t to a multi-fault set $F = \{f_i\}$ which occurs with probability $P = \{p_i\}$ in circuit C with PTM M_F and ITM M is defined as the total probability that the output under t is erroneous. Test t can be represented by the vector v_t, with 0's in all but the index corresponding to t's input assignments. For instance, if t assigns 0s to all input signals and C has 3 inputs, then v_t = [1 0 0 0 0 0 0 0]. The sensitivity of t is the probability that the ideal and faulty outputs are different, and it can be computed by taking the norm of the element-wise product of the correct and faulty output vectors. This operation is similar to our *fidelity* operation defined for vectors rather than matrices, and the computation takes the form:

$$sens(F,t) = 1 - ||(v_t M_f) .* (v_t M)||_{l_1}$$

A second method of sensitivity computation begins with a pre-selected complete set of test vectors for the permanent stuck-at faults corresponding to those in F. For each test vector in this set, we compute the faulty output at each gate using vector-PTM multiplication through intermediate gates. We also compute the ideal output at each gate. The ideal output is v_tM, and the faulty output vector is v_tM_f. The advantage of this method is that we do not have to explicitly compute the circuit PTM and ITM, steps which are computationally expensive.

A caveat in output vector computation is that fan-out branches result in inseparable probability distributions of the branch signals. If these signals are marginalized or treated as separate, then inaccuracies can occur in the output probabilities. A simple method of handling this problem is to jointly store the probabilities of these signals and then enlarge any gate PTM the signals encounter. We accomplish gate enlarging by adding inputs to the gate that pass through unchanged, i.e., tensoring the gate matrix with an identity matrix I.

```
compute_faulty_output(Circuit C, testvector T)
{
 for(all inputs i ∈ C)
  vector[i] = create_row_vector(T[i])
 insert_fanout_gates(C)
 sort_topological(C)
 for(each node g ∈ C)
  for(each input j ∈ inputs(g))
   inputvector[g] = inputvector[g] ⊗ PTM[j]
   enlarge(g, size(j) − 1)
  outputvector[g] = inputvector[g] × PTM[g]
  for(each outputs o ∈ outputs(g))
   vector[o] = outputvector[g]
}
```

Figure 22. Algorithm for output computation.

An algorithm for computing the output of a test vector under probabilistic faults (encoded in gate PTMs) is shown in Figure 22. The primary input values determined by the given test vectors are converted into input vectors. Then, in topological order, the inputs for each gate are tensored together to form the input vector for the gate. If any of the input signals are stored jointly with other signals, the gate in question is enlarged by the number of additional signals. The gate PTM is multiplied by the input vector to obtain the output vector. In the case of a multiple-output gate such as a fan-out gate, the output vector stays as a joint probability distribution. In practice, output distributions can become very large, through the accumulation of correlated signals. However, the joint signals can be separated by using the *eliminate variables* operation, which may entail some loss of accuracy.

This process can be repeated with gate ITMs (or functional simulation) to obtain the ideal output vector. Finally, test vector sensitivity is computed according to the foregoing equation for *sens*(*F*,*t*) using the *fidelity* operation applied to the ideal and faulty primary-output vectors.

3.8.2. Test Generation

We use the test-vector sensitivity information computed in the previous section to generate compact multisets of test vectors for detecting transient faults. Test-set compaction is closely related to the standard SET COVER problem [10]. In that problem, elements of a set *S* are to be covered by subsets. A minimal set of subsets must be chosen such that every member of *S* belongs to at least one of the chosen subsets. In the context of test generation, the set *S* consists of all possible faults, and each test vector represents a subset of faults, namely the subset of faults that it detects. When testing for soft errors, tests may have to be repeated to increase the probability of fault detection, therefore multisets of tests are selected.

This connection between SET COVER and test compaction allows us to modify algorithms designed for SET COVER and introduce related ILP formulations whose linear programming (LP) relaxations can be solved in polynomial time. Furthermore, modifying the test-multiset objective simply amounts to altering the ILP objective function.

Suppose a single fault f in a circuit C has an estimated probability p of occurrence. We confirm its probability as follows:

1. Derive a test vector t with high sensitivity $sens(f,t)$.

2. Apply t $k = \lfloor 1/sens(f,t) \rfloor$ times to C for one expected detection.

3. If we have $d(f) >> 1$ detections, we conclude that the actual probability of f is higher and reject the estimated probability. We can estimate the probability that there are $d(f)$ detections in k trials using the binomial theorem. If the probability of $d(f)$ detections is low, then it is likely that the actual sensitivity $sens(f,t)$ is higher than the estimate.

4. If $sens(f,t)$ is higher than estimated, we can update our estimate and repeat this process.

We can extend the above method to multiple faults under two different assumptions: 1) there are several probabilistic faults but the circuit experiences only a single fault in a clock cycle, and 2) each circuit component has an independent fault probability, implying that multiple faults at different locations can occur in the same clock cycle. The goal in either case is to pick a multiset of vectors T' taken from T such that $|T'|$ is minimal. Recall that each test vector t_i represents a subset of F, i.e., each test vector detects a subset of faults. Under assumption 1, we minimize the size of the multiset by using test vectors that are either especially sensitive to one fault or somewhat sensitive to many faults. Therefore, to achieve a given detection probability of p_{th} we need n tests, where n satisfies $(1 - p)^n \leq 1 - p_{th}$. Figure 23 gives the greedy algorithm for generating such a multiset of test vectors, starting from a compacted set of test vectors. Our ILP formulation for minimal test multiset generation is shown in Figure 24.

```
select_test_multiset(faults F, tests T, prob p_th)
{
 UF = F
 while(!isempty(UF))
  Tmax = find_maximal_test(UF, T, p_th)
  add_selected_test(ST, Tmax)
  UF = remove_new_covered(UF, ST, p_th)
 return ST
}

remove_new_covered(faults UF, test ST, prob p_th)
{
 for(each fault f ∈ UF)
  for(each test t ∈ ST)
  Pdet += ∏_i(1 − sens(f,t))
  if(1 − Pdet = p_th)
   remove_fault(UF, f)
 return UF
}
```

Figure 23. Greedy algorithm for minimizing the number of test vectors (with repetition) required for fault detection.

Figure 25 shows the number of test vectors required to detect probabilistic stuck-at faults using the method of Figure 23, and assuming probability p_f = 0.05. *Rand* is the average number of test vectors selected during random test generation. These results show that our algorithm requires 53 to 64% fewer test vectors than random selection, even with a small complete test vector set (generated by ATALANTA) used as a base set.

(a)

$$\text{Minimize } \sum_{i=0}^{m} x_i$$
$$\text{subject to:}$$
$$\forall j, \left(\sum_{i=1}^{0} x_i \times sens(f_j, t_i)\right) \geq n$$
$$\forall i, x_i \geq 0,\ x_i \text{ is an integer}$$

(b)

$$\text{Minimize } \sum_{j=0}^{n}\sum_{i=0}^{m} x_i \times sens(f_j, t_i)$$
$$\text{subject to:}$$
$$\forall j, \left(\sum_{i=1}^{m} x_i \times sens(f_j, t_i)\right) \geq n$$
$$\forall i, x_i \geq 0,\ x_i \text{ is an integer}$$

Figure 24. ILP formulations for test set generation with a fixed number of expected detections: (a) To minimize the number of test vectors, and (b) To maximize fault resolution.

Circuit	$p_{th} = .05$		$p_{th} = .75$		$p_{th} = .85$		$p_{th} = .95$		$p_{th} = .99$	
	Rand	Our	Rand	Our	Rand	Our	Rand	Our	Rand	Our
c6288	377	56	782	112	1034	148	1266	236	1998	360
c432	731	462	1415	924	1771	1221	2696	1947	3797	2970
c499	1643	518	2723	1036	3085	1369	4448	2183	8157	3330
c3540	907	411	1665	817	2256	1078	3589	1716	4975	2615
c5315	2669	854	4691	1708	6531	2557	8961	3599	13359	5490
c7552	3729	1680	6824	3364	8352	4445	12210	7082	18314	10805
c2670	3650	884	5699	1770	7755	2339	11104	3729	15961	5682
% improv.		64.5		59.7		57.26		53.71		53.05

Figure 25. Number of test vectors required to detect input signal faults with various threshold probabilities p_{th}.

Once a multiset of test vectors is generated, the actual probability of error can be estimated using Bayesian learning. This well-established artificial intelligence (AI) technique uses observation (data) and prior domain knowledge to predict future events. In our case, the prior domain knowledge is the expected or modeled fault probabilities in a circuit, and the data comes from testing.

4. Wireless Network Optimization

4.1. Summary

Mobile ad hoc networks (MANETs) are wireless communication networks which are of interest because of their flexibility and ease of deployment. MANET nodes are often powered by batteries, whose replacement is difficult. Internode transmission power thus constrains the network topology and the topology changes continuously due to mobility. Hence, understanding node mobility and efficiently managing transmission power are essential for successful network operation. First, we analyze mathematical models of node movement and propose a new metric to quantify mobility. Existing network control algorithms are usually evaluated using random mobility models. However, since such models employ incompatible mobility parameters, it is hard to compare the performance of different algorithms. We show that link duration has a nearly invariant relationship with route lifetime regardless of the adopted mobility model, and so is a good mobility metric. Second, we investigate the issues of power control and link maintenance. Existing power control schemes are mainly intended for (pseudo) static networks, and their effectiveness in highly mobile networks has not been demonstrated. We develop a novel algorithm, which adaptively controls transmission power and substantially reduces communication power. We analyze the impact of medium access control on performance, and show that the widely used request to send/clear to send (RTS/CTS) handshake protocol may adversely affect the network throughput. We further present a way to maximize the network throughput. Third, we investigate the problem of optimally placing base station and relay nodes to reduce power consumption and improve performance. We apply non-linear optimization techniques to node placement, and present distributed node placement techniques which place nodes among radio obstacles to minimize energy use. Simulation results confirm that the efficiency of the proposed algorithms is comparable to that of an existing centralized algorithm.

Further details concerning the material in this chapter can be found in Sungsoon Cho's 2009 Ph.D. dissertation [52] and related publications [53, 54, 55].

4.2. Introduction

A MANET is a multi-hop, wireless network consisting of a set of interacting hosts or nodes that move through space. Its many applications include networks for sensing the environment, vehicle tracking systems, and emergency communications in a disaster area; see Figure 26. When a node has to send a message to another node, the sender can either directly transmit the message to the recipient, or transmit the message to immediate nodes which relay the message to the final destination. MANET operation differs from traditional wired networks in several respects. The network topology constantly changes due to node movement. The links between node pairs can be created or deleted by adjusting the transmission power of the nodes. The communication medium (air) is shared by multiple hosts, so the transmitted data can be garbled or lost if channel access is not controlled appropriately. For these reasons, efficient operation of a MANET poses some unique challenges. Our research has focused on the following issues: the impact of node mobility on network topology, transmission power control, and medium access control.

Application	Objective
Emergency networks	To provide connectivity between distant devices where the network infrastructure is unavailable
Sensor networks	To monitor environmental conditions over a large area
Vehicular ad hoc networks	To enable real-time vehicle monitoring and adaptive traffic control
Personal area networks	To provide flexible connectivity between personal electronic devices or home appliances

Figure 26. Examples of MANET applications.

Much prior research has investigated ways to access the Internet using open access points. Balasubramanian et al. [43] proposed opportunistic connection techniques which enable web search in vehicles in urban areas where only intermittent connectivity is available. Banerjee et al. [4] proposed an energy-efficient data-forwarding architecture based on fixed and battery-powered data centers, which store and forward packets according to connection availability. However, a serious limitation with these techniques is that connectivity is available only when open access points exist nearby. To monitor environmental conditions over a large area, sensor networks can be used [51, 56]. For example, at Great Duck Island, Maine, 32 wireless sensor nodes are deployed [71] in a way that is manageable via the Internet; this habitat-monitoring network is expected to last 9 months with two AA batteries. In [83], a mobile sensor deployment project is presented, which delivers wireless sensors to a road using global positioning system (GPS) controlled unmanned aerial vehicles (UAVs). The sensor nodes, controlled by TinyOS [84], constitute a multi-hop communication network, track nearby vehicles passing on the road, and report tracking data to a base station via a UAV. An airborne communication network is another promising MANET application [86].

Next, we summarize the terminology and formal network models adopted in our work. Two nodes are connected if and only if they can directly communicate with each other. A network's connectivity or topology is represented by a graph $G = (V, E)$ where V denotes the network nodes and $E = \{(u, v)\}$, where $u, v \in V$. The data-forwarding process from a node to a neighboring node is called a hop. A data delivery path consisting of one or more hops forms a route. We also use the well-known 5-layer model [64] for the software and hardware parts of a network. Data generated in the top (application) layer at host A are transferred to lower layers, and the bottom (physical) layer transmits the data to the corresponding physical layer at host B. Then the data are transferred up to B's application layer via the data-link, network and transport layers. The middle (network) layer is responsible for maintaining appropriate routes from sources to destinations. The network topology can change over time due to node movements, so usually there is no entity that has global knowledge of the network's exact connection status. Hence, the delivery route from source to destination can frequently change, and the nodes need a method — a routing protocol such as dynamic source routing (DSR) [61] — to discover the routes to use.

The physical layer of a MANET handles conversion between data bits and radio signals. The received signal strength (RSS) of the radio signal at the receiver RX, can be expressed by:

$$\mathrm{RSS}(RX) = Pow(TX) \bullet Gain(TX, RX) = Pow(TX) \bullet [\, k_1 \bullet Dist(TX, RX)^{\alpha} + k_2]$$

where $Pow(TX)$ denotes the transmission power of node TX, $Dist(TX,RX)$ the distance between TX and RX, and α represents the radio attenuation of the environment. For successful data reception, RSS should exceed some threshold characterized by the receiver's sensitivity, so that receivers within the transmission range r_{TX} of the transmitter can successfully receive the message. In our work, we assume that a MANET node can receive data from at most one transmitter at a time. The main criteria for successful communication are: (1) there should be just one transmitting node within the transmission range of the receiver, and (2) the signal-to-interference-noise ratio (SINR) should be above some threshold.

Topology control algorithms	Methodology
LINT and LILT [76] K-neigh protocol [11]	Bounded number of neighbors
Cone-based topology control [54, 96]	At least one neighbor in every cone
MST-based topology control [55, 56]	Local minimum spanning tree

Figure 27. Representative topology control algorithms and their methodology.

MANET connections change over time due to node movement, and the resulting network topology changes critically affect network operation. Understanding the impact of mobility and topology on the performance is essential for designing network control algorithms. Analysis of the impact of mobility on network performance usually relies on simulations using artificial random mobility models [45, 46, 91]. Unlike wired networks, MANET topology can be actively controlled by adjusting the transmission power of the nodes. The range of the nodes should be assigned so that power consumption and signal interference are minimized, while connectivity is maintained.

There have been many studies of how transmission range affects the network performance and connectivity; see Figure 27. For instance, local information no topology (LINT) [79] simply allows each node to maintain a bounded number of neighbors by incrementally adjusting its transmission power. However, although it forms a network that is connected with a high probability, LINT does not guarantee global connectivity. To deal with this problem, the related local information link-state topology (LILT) method uses routing tables to maintain global connectivity. Most control algorithms attempt to fix the network connection after a topology change occurs, and cannot reduce the number of connection changes caused by node movements. To remove this limitation, we have developed a proactive topology control algorithm that adjusts the transmission power of communicating nodes, prevents frequent link breaks, and improves communication performance.

4.3. Impact of Mobility on Performance

MANET performance is highly sensitive to changes in node-to-node connections (communication links) caused by node movement. Link instability of this kind has proven very difficult to analyze mathematically, so previous work has relied heavily on simulation. We have constructed a mathematically tractable model of node motion, the constant velocity (CV) model, and used it to derive a precise relation between mobility and connection stability [52, 53]. Our analysis allows determination of the appropriate frame length for efficient single-hop communication. We investigated connection stability in multi-hop communication, and uncovered some underlying properties of previously proposed mobility metrics. In particular, we demonstrated that link duration has a nearly invariant relationship with the stability of multi-hop connections for a wide range of mobility models, and thus forms an excellent mobility metric.

We approach the problem as follows. First, we set up the relatively simple CV model, and derive two mobility metrics for it, link duration (LD) and link change rate (LCR) [65, 75, 82]. Then we derive an analytic expression for successful packet delivery via single-hop communication. We further investigate the relation between LD and multi-hop route stability. We quantify connection stability by the mean residual duration (RD) of routes, which measures how long multi-hop routes last under the given mobility conditions. We show that, among previously proposed mobility metrics [42], LCR is unsuitable for estimating link stability because the relation between LCR and RD depends on other network parameters. In contrast, by using the analytic expressions derived from our CV model, we find that RD is a function of LD; i.e., the multi-hop connection stability is mainly determined by single-hop link duration. We also derive simulation results which show that LD has a consistent relation with RD for a wide range of mobility conditions. Our analysis and simulations confirm, as suggested in [49], that LD constitutes a very good, unified metric for link stability with many types of mobility models.

4.3.1. Constant Velocity Model

Assume that nodes are randomly placed on an unbounded plane with a density ρ. The nodes move linearly at a constant velocity v in random directions, but do not change direction while moving; see Figure 28. We have shown that the average LCR λ_{LCR} is given by:

$$\lambda_{LCR} = 2\lambda_{gen} = \pi\rho r v$$

where λ_{gen} is the average link generation rate. For example, if nodes move at the average speed v =10 m/s with transmission range r =10 m, and the node density is $\rho = 0.02$ m^{-2}, then $\lambda_{LCR} = 2\lambda_{gen}$ =10.2 s^{-1}.

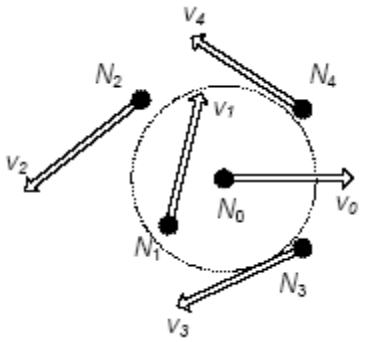


Figure 28. Illustration of node movements in a MANET.

The link duration T_{LD} is the time from link generation to link break, and is a measure of the stability of single-hop connections [42]. We can describe the event that node N_i passes through the transmission region of a source N_0 with two parameters (X,Θ). The link duration is then given by $T_{LD} = Y(X)/v_i\rho(\Theta)$ where $Y(X) = 2(r^2 - X^2)^{1/2}$, and the mean value of T_{LD} is:

$$\overline{T}_{LD} = \int_0^{\pi} \int_0^{r} T_{LD}(x,\theta) f_{X,\Theta}(x,\theta) dx d\theta = \frac{\pi^2}{8}(\frac{r}{v})$$

It should be noted that LD is not the reciprocal of LCR or the link generation rate. LCR is the reciprocal of the time between two successive link changes, whereas LD is defined as the time between link generation and link break.

Suppose a receiver N_i enters the communication region of a transmitter N_0 at time $t = 0$ and exits the region at time $t = T_{LD}$. For a packet with transmission time T_{comm} to complete communication before N_i moves out of range, the transmission should start at time t, where $0 \le t \le T_{LD}(x,\theta) - T_{comm}$. Hence, the conditional probability of complete transmission is:

$$p_{comp}(T_{comm}|X,\Theta) = max[T_{LD}(X,\Theta) - T_{comm},0] \,/T_{LD}(X,\Theta)$$

and the total probability of complete transmission is:

$$p_{comp}(T_{comm}) = \int_0^{\pi} \int_0^{r} p_{comp}(T_{comm}|x,\theta) g_{X,\Theta}(x,\theta) dx d\theta$$

where $g_{X,\Theta}(x,\theta)$ denotes the joint probability density function of random variables X and Θ. We can rewrite the last expression as:

$$p_{comp}(\tau) = \int_0^{\pi} \int_0^{1} p_{comp}(\tau|x,\theta) g_{X,\Theta}(x,\theta) dx d\theta$$

where τ denotes a normalized communication time that can be interpreted as the ratio of node mobility to communication speed.

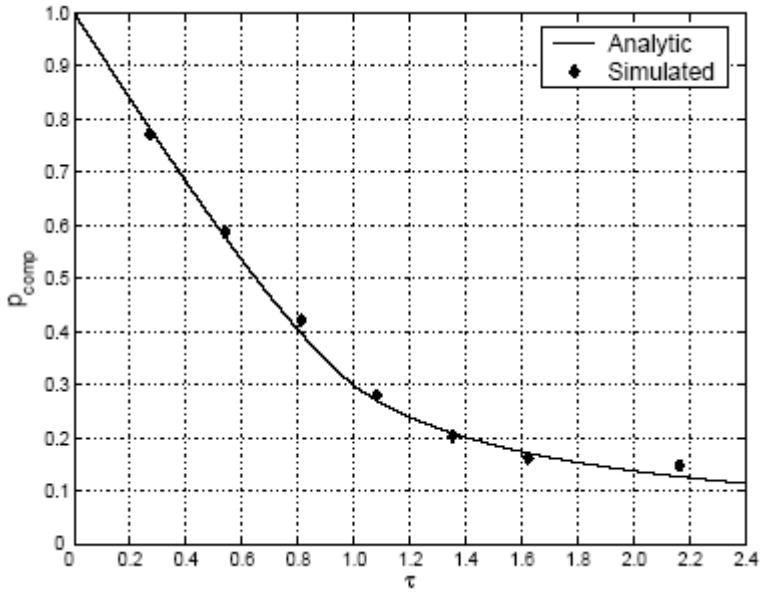


Figure 29. Plot of transmission probability p_{comp} vs. communication time τ.

Figure 29 compares the foregoing analysis with simulated results using 100 mobile nodes on a 1×1 plane. The network parameters such as the transmission range *r*, the node speed *v*, and the communication time *Tcomm* are varied so that τ ranges from 0 to 2.2. Comparison of the simulated and calculated p_{comp} values supports the accuracy of the CV model.

Although the success rate of single-hop communication is generally insensitive to mobility, successful data delivery over multi-hop routes critically depends on the connection stability. We define the residual duration of a multi-hop route as the mean time from the route discovery to the breaking of the route. RD is the key factor which determines the success of packet delivery over multi-hop routes, and so is an important parameter in MANET design [41]. For instance, when the RD of a multi-hop route is 100 ms, data delivery which takes 500 ms over the route will probably fail due to the connectivity change. Hence RD is an indicator of the stability of multi-hop routes, whereas link duration indicates the stability of single-hop links.

4.3.2. Mobility Metric Relationships

We now investigate the relationship between LD and RD. It is extremely difficult to derive rigorous expressions for RD using existing mobility models such as random waypoint, random walk, and boundless simulation area. Hence, we conducted a series of simulations with these mobility models, which show a strong correlation between LD and RD. Later we analyze the relationships between LD and RD by using our CV model. The simulation is organized as follows. First, 100 nodes are randomly placed on a 1x1 plane. The nodes start moving according to the given mobility model. At time $t = 0$, the simulator computes the shortest, multi-hop path from each node to the root node N_0. When a multi-hop route is broken at time t, all routes with later hops that pass along the broken route are also regarded as broken at time t. This procedure measures the RD of routes from the shortest path discovery to their break times.

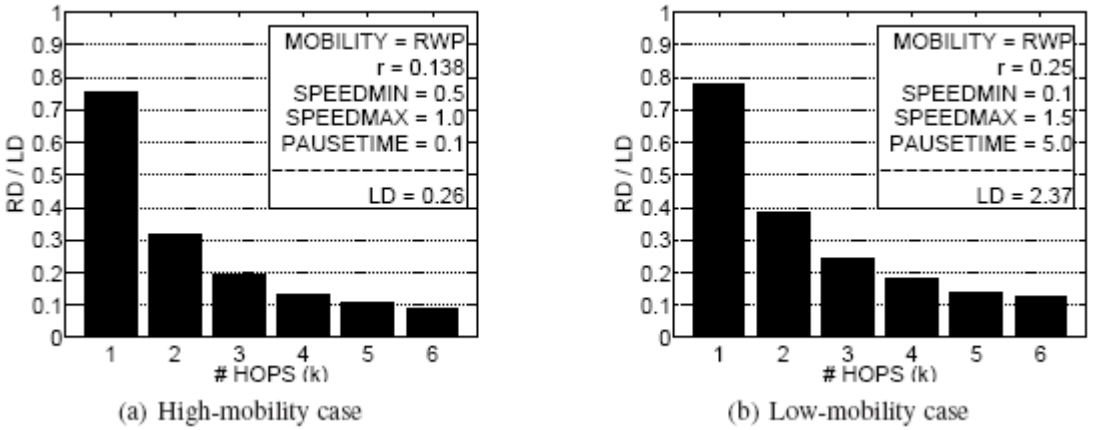


Figure 30. Simulation results with RWP model.

Here we only present results for the random waypoint (RWP); results for the other model can be found in [52]. Mobile nodes in the RWP model [61] behave as follows. First a node selects a random destination within a bounded movement area, and moves toward it at a random speed v. Once it arrives at the destination, it pauses for a predefined *pausetime*. The speed v is a random variable uniformly distributed between *speedmin* and *speedmax*. By varying r, *speedmin*, *speedmax*, and *pausetime*, we can control the node movements.

Figure 30(a) shows simulation results corresponding to a high-mobility case with parameters: r = 0.138, *speedmin* = 0.5, *speedmax* = 1.0, and *pausetime* = 0.1, for which the link duration $T_{LD} = 0.26$. The RD/LD ratio of 0.32 for 2-hop links, for example, indicates that the average residual duration of a 2-hop link is $0.32 \times T_{LD} = 0.0832$. Figure 30(b) corresponds to a low-mobility case: $r = 0.25$, *speedmin* = 0.1, *speedmax* = 1.5, and *pausetime* = 5.0, for which $T_{LD} = 2.37$. In this case, due to the relatively long *pausetime*, a considerable number of nodes are observed in their pause state. Hence, this case has a larger LD value than the previous case. Although the two cases differ in their mobility parameters and LDs, their RD/LD ratios are nearly identical. Instead of changing *pausetime* alone, we also varied *speedmin* and *speedmax* and obtained nearly the same distribution of RDs.

From our simulation studies, we saw that LD has a consistent relation with RD for a wide range of mobility models and parameter values. Using our CV model, we can explain the strong correlation between LD and RD, and also the relations between LD, LCR and RD. Consider the question: What makes LD a better mobility metric than LCR? Suppose at time $t = 0$, a mobile node N_0 observes a neighbor node N_1 within its transmission range, and at time $t = T_0$, the neighbor leaves the range. This exit time T_0 is a random variable, and its cumulative distribution function (CDF) is $F(t)$. Hence, the probability that a link remains connected until time t is given by:

$$p_{comp}(t.v/r) = 1 - F(t)$$

Next, suppose that a k-hop route consists of $k + 1$ mobile nodes $\{N_i\}$ and each node pair (N_{j-1}, N_j) is connected by a link L_j. At time $t = 0$, the route is connected, and at $t = T_1$, the route breaks. Let $G_k(t)$ and $g_k(t)$ denote the CDF and probability density function (PDF), respectively, of the random variable T_1. Then the probability that the route remains connected until time t is $1 - G_k(t)$), and the RD for this route is:

$$\overline{T}_{RD,k} = \int_0^\infty t \cdot g_k(t)dt$$

Assuming the generation/break processes of the links in a route are mutually independent [59], the probability that a k-hop route remains connected until time t becomes $1 - G_k(t) = [1 - F(t)]^k$ from which we can deduce that:

$$\overline{T}_{RD,k} = \int_0^\infty t \cdot [\frac{d}{dt}\{1 - p_{comp}(\frac{\pi^2}{8}\frac{t}{\overline{T}_{LD}})^k\}]dt$$

Hence, we see that RD is a function of LD rather than LCR, which is why LD is a good indicator of multi-hop connection stability.

It is now easily seen that the metrics LD and LCR are related by the formula $\lambda_{LCR} \bullet T_{LD} = 2\rho\pi r^2$, which implies that the product of half of LCR and LD equals the average node degree. This relation also follows from Little's theorem, which states that the average number of customers in a system, $\rho\pi r^2$, is equal to the product of the customer arrival rate, $\lambda_{gen} = \lambda_{LCR}/2$ and the average of the time T_{LD} customers spend in the system.

A recent study by Nayebi et al. cites our work and observes that the probability distribution of LD of the RWP model is similar to that of the CV model [73]. Furthermore, using the boundless random direction model (BRDM) derived from CV, they showed that the PDF of RWP can be approximated fairly accurately by adding stationary nodes. From these observations, it can be seen that LD with RWP is practically equivalent to that with CV. Therefore, we conclude that LD is a good unified mobility metric for most types of mobile ad hoc networks, and that CV is a useful model for mobility analysis.

4.4. Distributed Power-Aware Link Maintenance

We have developed a new management algorithm for MANETs called PALM (power-aware link maintenance), which simultaneously performs transmission power control and route connectivity maintenance. Unlike most topology control algorithms, PALM manages the transmission power of active nodes only, and thus eliminates considerable energy and channel resource waste. We also investigated how medium access control (MAC) parameters affect MANET performance, and found ways to maximize network throughput while maintaining a high energy efficiency.

4.4.1. Introduction

Topology controls [48, 67, 79, 87] have long been considered for MANET power management. They attempt to reduce the transmission power while maintaining a connected network topology. However, they have several drawbacks. First, they require periodic beaconing, which wastes energy and channel resources. Second, the actual connected routes still have to be discovered by the routing layer, so that even when the network is connected, frequent rediscovery of connected routes may occur. Third, in order to make the network insensitive to node mobility, topology control algorithms must allow longer transmission range, which may worsen the network performance.

The so-called BASIC power control [62] algorithm has been proposed to mitigate the above problems. The transmitter computes the minimum transmission power for data delivery based on received signal strength at the receiver. Then the data packets are transmitted at minimum power, while only control packets such as request-to-send and clear-to-send are transmitted at full power. While BASIC reduces overall energy consumption, it still has shortcomings. The nodes must abide by the data delivery routes discovered by the routing layer, which may not be power-efficient, and the control packets transmitted at full power may collide with other nodes' transmissions. The power control MAC (PCM) protocol [62] mitigates the latter problem by allowing periodic full-power transmission for DATA packets, but the former problem still persists.

The power aware routing optimization (PARO) algorithm [57, 58] attempts to discover power-efficient routes in a distributed manner without periodic beaconing. PARO assumes that all nodes are within transmission range of each other. At the start, a source node A directly sends packets to a destination node E. If another node *C* overhears the communication from *A* to *E*, and determines that route redirection via itself conserves energy, then the route becomes *ACE*. This redirection may be repeated many times. Sometimes the redirected routes become inefficient as a shorter route may exist. PARO attempts to remove unnecessary redirectors to discover power-efficient routes in a distributed manner. However, PARO still carries the risk of generating inefficient routes, since it cannot easily determine hop distance.

Our proposed PALM scheme resolves the above problems with BASIC and PARO. We first determine a *virtual* hop distance to efficiently assign and measure hop distances in the presence of route redirections. Second, we add an accumulated energy field to packets, and enable the removal of multiple redirectors. Third, we use different transmission power levels for control and broadcast packets to mitigate the problems with BASIC's power control.

4.4.2. PALM Method

The medium access control in PALM is based on IEEE 802.11. We assume ad hoc on-demand (AODV) [77] as the routing algorithm. We also assume that transceivers are equipped to detect received signal strength RSS, which was defined earlier and is proportional to the transmission power. PALM's power control method is similar to BASIC's except for the power levels of RTS/CTS packets. The transmitter TX records its transmission power level $Pow(TX)$ on outgoing packets, and receivers RX_i, including overhearing nodes, estimate the channel gain $Gain(TX, RX_i)$ from the RSS. TX sends data packets at the power level given by:

$$Pow(TX, RX_i) = \beta \cdot \frac{RSS_{\min}}{Gain(TX, RX_i)}$$

where RSS_{min} and β denote the minimum RSS for successful reception and a safety factor, respectively. PALM uses the minimum power for data packets, and relatively greater power for control packets.

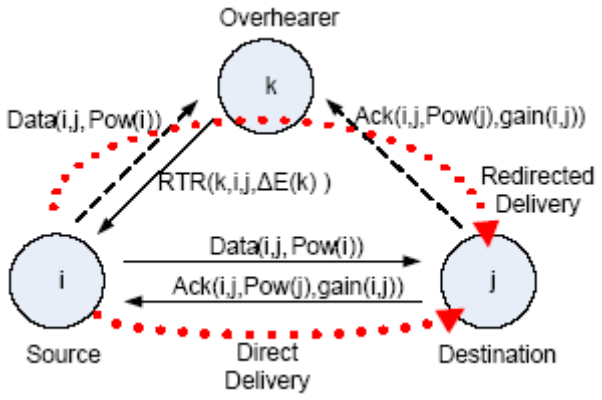


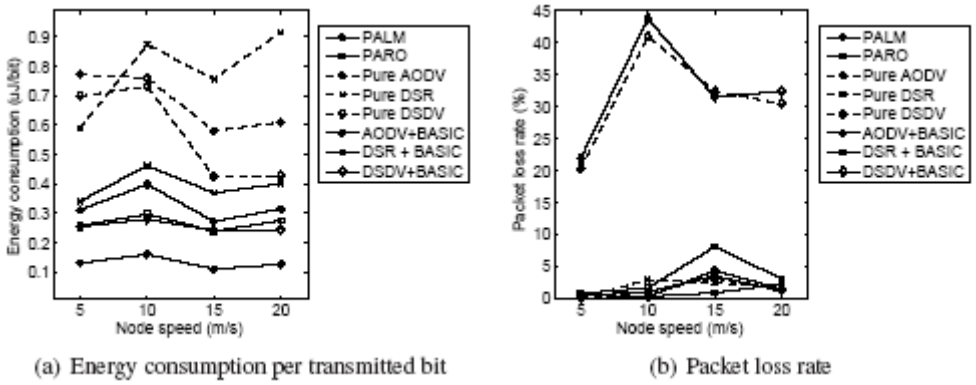
Figure 31. Communication between nodes in PALM, including route redirection.

PALM performs route redirection in a similar way to PARO: see Figure 31. Here node i is forwarding data packets to next-hop node j, and node k is overhearing them. Solid lines in Figure 31 indicate intended packet flow, and dotted lines, overheard flow. The DATA packets contain the source address i, the destination address j, and i's transmission power $Pow(i)$. Node j replies with an acknowledge (ACK) message containing i, j, $Pow(j)$ and $Gain(i, j)$. The overhearing node k now can determine which of the direct $i \rightarrow j$ and redirected $i \rightarrow k \rightarrow j$ routes consumes less energy, and it can estimate the energy savings ΔE. After overhearing ACK, node k sends a request-to-redirect (RTR) packet to node i. On receiving an RTR message, i modifies its routing table so that its new next hop node becomes k instead of j. Subsequent packets are delivered along $i \rightarrow k \rightarrow j$ instead of $i \rightarrow j$. Note that there might be more than one overhearing node that sends an RTR to the source. To address this, when a node that is to send an RTR message overhears another RTR with a larger ΔE value, it does not send an RTR message, and discards its own RTR.

PALM's redirection technique enables nodes to repair routing tables locally without propagating rerouting requests. However, this carries the risk of generating loops in the delivery path. PALM resolves loop problems in the following way. First, we define the virtual hop distance to the final destination as a rational number instead of an integer. After a route is discovered, the real-numbered virtual hop distance has the same value as the usual integer-numbered hop distance. When redirecting a link between nodes X and Y with hop distances h_X and h_Y, respectively, the redirector computes its (virtual) hop distance as $(h_X + h_Y)/2$. By taking this value between h_X and h_Y, PALM maintains monotonically decreasing hop distance numbers along routes without propagating the redirection event to other nodes. PALM has a locking method for use when packets are forwarded along the links or when a link entry is added. Since its route redirection scheme tends to increase the number of hops from source to destination, PALM also employs mechanisms to mitigate some associated problems. For example, it includes a "route-warping" technique to prevent an excessive number of hops from being created by the redirection procedure. Increasing the number of hops tends to increase channel contention between nodes. In particular, the long transmission range for RTS/CTS packets may worsen this problem. PALM addresses this by reducing transmission power for broadcast and control packets. All these techniques collectively enable PALM to continuously construct power-efficient and loop-free routes.

4.4.3. Performance Evaluation

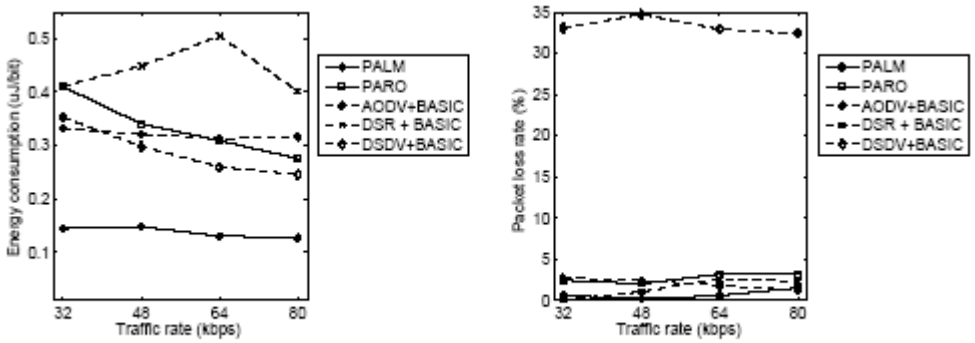
We conducted an extensive set of experiments to evaluate PALM. We used the *ns* simulator [85], and modified its AODV, MAC 802.11, and physical layer to implement PALM. A MANET model for the experiments was constructed as follows. The radio channel parameters are taken from [62]: the RSS threshold RSS_{th} = 0.3652 nW; the maximum transmission power Pow_{max} = 281.8 mW corresponding to 250 m transmission range; and the channel bandwidth is 2 Mbps. We assume that each node i can take any value between 0 and Pow_{max} as its transmission power. The following PALM specific parameters are also used: energy saving threshold $\alpha = 0.8$; safety factor $\beta = 1.5$; active neighbor window T_{act} = 0.5 s; usage expiration time T_{unused} = 0.5 s; and unlock time T_{unlock} = 0.5 s. Finally, the two-ray ground model [62] was adopted as the radio propagation model.



(a) Energy consumption per transmitted bit (b) Packet loss rate

Figure 32. Simulation rsults with varying node speed and a CBR of 80kbps.

First, we carried out simulations with varying node speeds. The network consists of 35 mobile nodes randomly distributed on a 700×700 m2 plane. Source and destination nodes are randomly chosen and a constant-bit-rate (CBR) source sends packets at 80 kbps. Packet size is 1000 bytes, and packet interval time is 0.1 s. The RWP was used with the following parameters: pause time = 0.0; minimum speed = maximum speed = *v* varying from 5 m/s to 20 m/s. Each simulation was run for 500 s. The maximum number of active neighbors is set to $k = 8$. Figure 32 presents the simulation results with respect to node speed ranging from 5 m/s to 20 m/s. For comparison, simulation results with AODV [77], dynamic source routing (DSR) [61], and destination sequence distance vectoring (DSDV) [76] with the BASIC power control scheme are included; also included is a variant of PARO. It can be seen that PALM's energy consumption is much lower than that of other algorithms including PARO, and its packet loss rate is comparable to others.



(a) Energy consumption per transmitted bit (b) Packet loss rate

Figure 33. Simulation results with varying traffic rate and a node speed of 20 m/s.

Next, we conducted simulations that varied the traffic rate from 32 to 80 kbps; see Figure 33. The packet size is 1,000 bytes, and the packet interval ranges from 0.1 s to 0.25 s. The node speed is set to 20 m/s. The energy consumption of DSDV decreases as the traffic rate grows because its beaconing energy is amortized by the increased data traffic. However, DSDV still results in high packet loss rate for the reason stated earlier. On the other hand, PALM produces low packet loss rate, while it still consumes much less energy than other algorithms.

We also measured the impact of the maximum number *k* of active neighbors on the energy consumption and the packet loss rate, and compared it with PARO's energy consumption and loss rate. The simulation was done over a stationary network to isolate the effect of transmission power. PALM's operation with a stationary network is identical with that of PARO except for the transmission power for control and broadcast packets. We found that PALM's energy consumption increases with *k*, but is still much lower than PAROs, even for fairly large *k*. On the other hand, PALM's packet loss rate is much lower than that of PARO. It can be concluded that transmitting control packets to only a few of the active neighbors is sufficient to prevent signal collision, and considerably improves communication concurrency.

4.4.4. Impact of MAC Parameters

We investigated how medium access control parameters affect the performance of MANETs in which the transmission power for data packets is controlled to the minimum value. We considered the impact of three factors in the MAC layer on network performance: RTS/CTS handshake, transmission power control, and carrier-sense threshold. The RTS/CTS handshake is used to reduce signal collisions due to simultaneous transmissions by nodes located outside each other's transmission range, which is called the hidden terminal problem [47]. It has been recently observed that while the RTS/CTS handshake effectively reduces the hidden terminal problem in local area networks, its effectiveness is limited in ad hoc networks [88, 89, 90].

For the purpose of reducing communication power consumption, the BASIC power control method has long been considered, which adjusts the power level for DATA packets to the minimum necessary value, while the RTS/CTS packets are transmitted at the maximum level. However, recent studies [63, 72, 81, 93] have shown that when the RTS/CTS handshake is used with BASIC, the overall network throughput may become worse than the networks without transmission power control, because the RTS/CTS packets can easily interfere with other DATA packets with the reduced power level. In addition, the carrier-sense (CS) threshold affects concurrency of network communication, and in consequence, the end-to-end network throughput also depends on the carrier-sense threshold [90].

We have confirmed through simulations that the optimal CS threshold is given by $CS_{th} = RSS_0/6z_0$, where z_0 is the minimum signal-to-interference ratio (SIR) for a successful signal capture. Since this condition does not depend on the communication distance or the radio attenuation exponent α, once CS_{th} is set to the optimal value, even when the network nodes continuously adjust the transmission power, and the radio channel property changes, the optimality condition can always be satisfied.

We have shown that use of the RTS/CTS handshake with power control protocols such as PALM may adversely affect the network performance. Our simulation results with variants of the BASIC power control scheme confirm that the use of maximum power for control packets can reduce the network throughput by 70%, and the total energy consumption with retransmission taken into account is much higher than that with carrier sense multiple access (CSMA) only. Therefore, when the transmission power for data packets in MANETs is controlled to the minimum necessary level through the channel gain feedback between nodes, the RTS/CTS packets with the maximum transmission power may worsen signal collision and increase energy consumption. We also have shown that a high network capacity can be obtained by reducing the transmission power for control packets, and proposed a means to determine the appropriate transmission power for control packets in a distributed manner.

4.5. Node Placement Optimization

Energy conservation is a key issue in ad hoc wireless network operation. Placing additional nodes at appropriate locations can substantially lower the power requirements of communicating nodes. This section investigates node placement for energy-constrained networks, and presents node placement algorithms that aim to minimize total energy consumption. We first consider the mobile base station (BS) placement problems for hierarchical wireless networks, and develop efficient heuristic solutions for them. We model the placement of multiple BSs as a clustering optimization problem in which BSs and user nodes are treated as clusterheads and cluster members, respectively. We also devise a heuristic that discovers the central area of a multi-hop network, and solves the BS placement problem with multi-hop connectivity. Our simulation results confirm that our methods reduce the energy consumption of wireless networks by up to 55% compared with grid networks. By using the PALM algorithm presented above, we devise a distributed relay placement algorithm for the flat network structure that discovers energy-efficient routes while maintaining connectivity in the presence of radio obstruction. Simulation results show that the power consumption of the distributed implementation is greater than that of an existing centralized algorithm by only 25%.

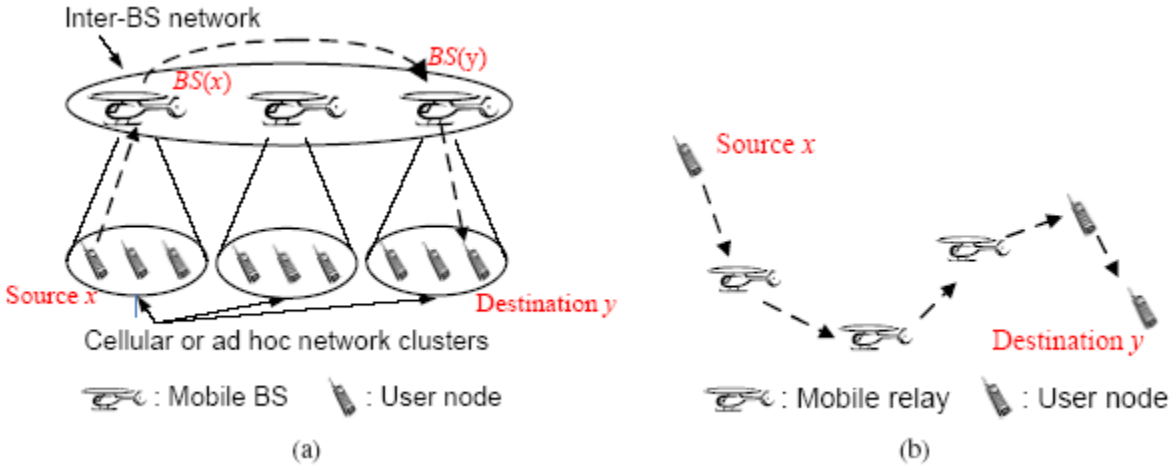


Figure 34. (a) Hierarchical network with mobile BSs, and (b) Flat network with mobile relays.

4.5.1. Introduction

Ad hoc networks can be categorized to two classes, hierarchical and flat; see Figure 34. In hierarchical networks as in Figure 34(a), nodes are grouped to clusters, and a clusterhead or BS node in each cluster takes responsibility for intercluster communication. In flat networks as in Figure 34(b), all nodes have the same communication ability, and data from the source user node are delivered to the destination via intermediate user nodes or relays.

We first consider a two-layered hierarchical network structure consisting of user nodes and mobile BSs, which can be moved to any location. We further assume that the BSs have a separate wireless channel for communication between them, and they have sufficiently large energy sources. Consider the following example. Suppose there are N sensor nodes with wireless transceivers in a forest. We dispatch R UAVs with wireless communication capability to assist network communication between the sensors. Where should we place the UAVs to minimize energy consumption of the sensor devices? We model this problem as a clustering problem composed of unconstrained convex optimization subproblems. We treat BSs as clusterheads, and aim to minimize the energy consumption of uplink communication, i.e., communication from user nodes to BSs. As the clustering problem is NP-hard, we develop an efficient heuristic method based on the K-means algorithm [70, 80], which is widely used for determining clusters that minimize the mean squared distance from points to the nearest cluster means. Simulation results show that our BS placement algorithm produces near-optimal solutions with high probability. Then, we investigate the relay placement problem for the flat network structure by modeling it as an analogous mechanical system. Our goal in relay placement is to place mobile relays at appropriate locations, and minimize the total power consumption, while maintaining network connectivity. By emulating the artificial forces exerted on mobile relays, and utilizing the PALM algorithm, we solve the relay placement problem in the presence of radio obstruction in a distributed manner.

Network structure	Number of base stations	Route connectivity
1-SH	1	Single-hop
K-SH	$K > 1$	Single-hop
1-MH	1	Multi-hop
K-MH	$K > 1$	Multi-hop

Figure 35. Four different network structures for BSP problems.

4.5.2. Base Station Placement Optimization

We investigate base station placement (BSP) optimization for hierarchical networks considering four different network structures of increasing complexity, which are summarized in Figure 35. For the node placement problem of hierarchical networks, we again adopt the radio model used earlier. We assume that nodes expend the minimum transmission power by using the BASIC power control scheme. We further assume that BSs have full information about the locations of all nodes. We also allow source and destination nodes to be randomly chosen. Unlike sensor networks in which most nodes send data to a single sink, data can be generated at any node, and can be delivered to any other node, and the source-destination pair can frequently change over time.

We use the following network architecture, which resembles that of the near-term digital radio (NTDR) network [78, 92]. A source node *x* either directly transmits data to the nearest base station BS(*x*), or sends it over the shortest multi-hop path to BS(*x*). Then BS(*x*) sends the data to the base station BS(*y*) of the destination *y* through a separate communication channel. Eventually, BS(*y*) sends the data to the final destination *y*. In this structure, the inter-BS network provides the network backbone, and the clusterhead serves as the gateway.

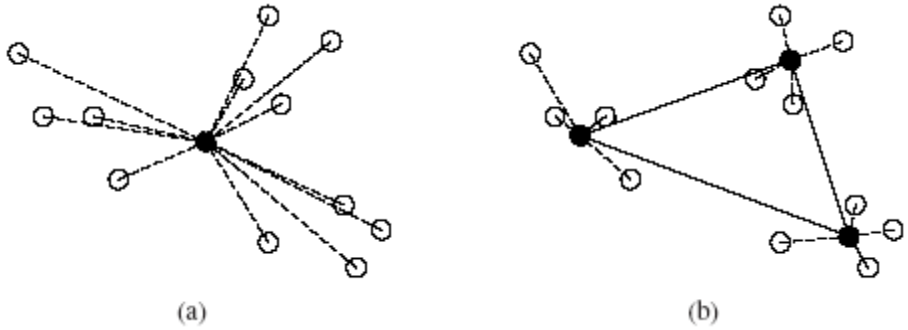


Figure 36. Network structure with single-hop communications: (a) User nodes (white) directly communicate with a single BS (black), and (b) User nodes communicate with their nearest BSs.

First we consider the single base station with single-hop links or 1-SH case. User nodes directly communicate via a single BS as in Figure 36(a). Suppose that *N* nodes are placed at locations denoted by vectors $\vec{p}_i$. The nodes directly communicate with a BS located at $\vec{x}_i$, and we want to minimize the uplink communication energy consumed by the user nodes by controlling the location the BS. The BSP problem is then: What is the optimal location $\vec{x}_i$? The power consumed by each node p_i can be written as:

$$E(p_i) = \tau(p_i)(k_1\|\vec{x} - \vec{p}_i\|^{\alpha} + k_2)$$

where $\tau(_{pi})$ denotes the transmission time of p_i, which is the number of bits to be sent by p_i divided by the bandwidth of the transceiver, and $\| \bullet \|$ denotes the Euclidian norm. Thus the 1-SH BSP problem becomes the following optimization problem:

$$\text{Minimize} \quad \sum_{i=1}^{N} \tau(p_i)(k_1 \|\vec{x} - \vec{p}_i\|^{\alpha} + k_2)$$

where BS location $\vec{x}_i$ is the optimization variable. This is an unconstrained optimization problem which turns out to be convex and can be efficiently solved with existing convex optimization techniques.

To extend the preceding case to multiple-BS placement (the *K*-SH problem), assume that there are *K* BSs and *N* nodes as in Figure 36(b). Each node communicates with its nearest BS, and there is a separate channel for inter-BS communication. The energy cost function of the 1-SH case does not change its form with respect to the BS location. However, such an invariant function cannot be used for the *K*-SH case, as user nodes change their BSs depending on the BS locations, which makes this problem NP-hard [74]. Define clusters $\{C_k\}$ as *K* disjoint subsets of $P = \{ p_i\}$. The *K*-SH BSP problem then becomes the following clustering optimization problem:

$$\text{Minimize} \quad \sum_{k=1}^{K} \sum_{i \in C_k} \tau(p_i)(k_1 \|\vec{x}_k - \vec{p}_i\|^{\alpha} + k_2)$$

where clusters C_k and BS locations $\vec{x}_k$ are the optimization variables. This is a generalization of the well-known *K*-means clustering problem. To solve it, we use the following straightforward heuristic. First groups nodes into their nearest clusters. Then compute the BS locations by solving the 1-SH optimization problem:

$$\text{Minimize} \quad \sum_{p_i \in C_k} \tau(p_i)(k_1 \|\vec{x}_k - \vec{p}_i\|^{\alpha} + k_2)$$

These steps are repeated until no change in clustering occurs.

Convergence of the K¬SH algorithm can be proved easily. Optimality depends on the initial BS locations $\vec{x}_k$, just as in the K-means case [60]. For this reason, in order to obtain satisfactory solutions, K¬SH needs to be repeated, for which we have also devised an efficient heuristic method. Our solution methods for the 1-MH and *K*-MH cases further extend these ideas, and can be found in [55].

4.5.3. Simulation Results

The solution quality of the proposed BSP algorithms depends on the initial clustering seeds. Figure 37 shows simple examples of optimal and suboptimal BSP placements produced by our *K*-SH algorithm. Figure 38 gives histograms of solutions produced by (a) random seeding, (b) the seeding method of Ostrovsky et al. [74], (c) the modified method of [74], and (d) the farthest-first method; the resolution of the data is 15 mW. Each seeding method was repeated 1000 times, and the solution quality was measured by the total transmission power. According to Figure 38(b), the probability that the method of [74] produces an optimal solution is approximately 1%, which is only slightly higher than that of the random seeding (0.4%). Figure 38(d) shows that the farthest-first method produces the optimal solution with high probability (38%). It performs clustering in a greedy manner, and consequently, outperforms the other seeding methods when nodes are almost uniformly distributed. Thus, the farthest-first method was adopted for the remaining experiments.

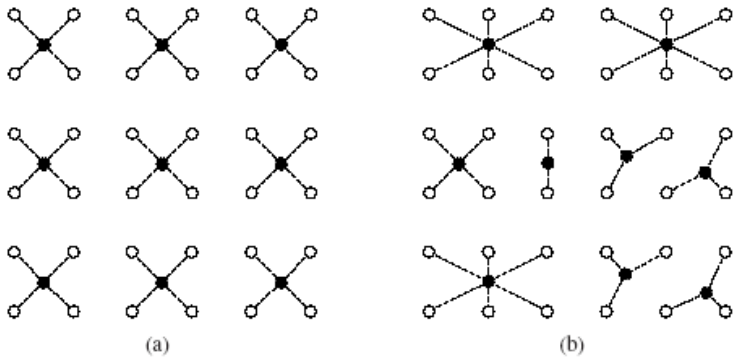


Figure 37. Examples of: (a) Optimal, and (b) Sub-optimal BS placement. User nodes (white) form a grid structure and BS locations (black) are computed by *K*-SH.

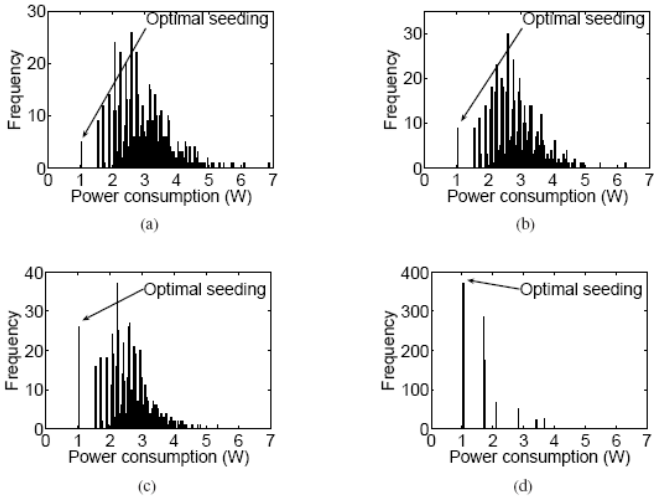


Figure 38. Total power consumption after: (a) Random seeding, (b) The seeding method of [34] (original), (c) The seeding method of [34] (modified, and (d) The farthest first method.

4.5.4. Distributed Relay Placement Optimization

While the BS placement algorithms presented above compute optimal locations that act as clusterheads with relatively large energy sources, the algorithms considered next try to place mobile relays that forward data from previous hop nodes to next hop nodes. We also investigate the distributed relay placement (DRP) problem in the presence of radio obstacles.

In this analysis, we assume that only one user node is the sink node that serves as the final destination of all data generated by other users. Sources send new data to a sink through direct or multi-hop routes. Relays only forward data from sources to the sink. The network attempts to construct energy-efficient routes using PALM. We assume that nodes adjust their transmission power to the minimum necessary level, and receiver sensitivity RSS_{min} is set to 1. The power cost for communication across distance d is $Cost(d) = k_1 \bullet d^{\alpha} + k_2$. We also assume that relays can sense the direction of incoming signals and the distance to the transmitter, and so can deduce the relative locations of their neighbors. For the current network topology, relays identify their immediate neighbors, and move toward locally optimal locations according to an artificial "force" function. When an object is located between two nodes, direct communication may be obstructed. We assume that when no line-of-sight communication is available, the radio gain between two nodes becomes zero. In addition, mobile relays have full information about the locations and the shapes of radio obstacles.

In order to formulate and solve the relay placement problem, we modeled the network as a mechanical system with springs and a viscous damper—a widely used approach for solving optimization problems [69]. Specifically, we model the communication energy cost as an artificial potential energy stored in springs, and nodes as objects with unit mass, moving according to the artificial force field. Movement of objects then resembles the progressive solution improvement of the steepest descent method [50] in convex optimization. Thus, through this model, we can obtain a locally optimal solution as the mechanical system converges to an equilibrium point.

Based on this mechanical analog, we designed a distributed controller of mobile relays as follows. The sink node determines the area where an additional relay is needed, and dispatches a mobile relay to that area. On arrival, the new relay starts to discover energy-efficient routes, and moves in the direction specified by the mechanical model so that the total energy consumption decreases. Then the sink dispatches another relay. These steps continue until the desired number of relays are placed in the network.

In the DRP algorithm, each relay performs the following operations repeatedly: route redirection, location sensing, energy cost estimation, and movement control. First, it continuously attempts to discover energy-efficient routes by performing route redirection as described for PALM. Second, both sources and relays continuously broadcast hello messages to their immediate neighbors. By sensing the incoming hello messages, each relay can estimate the energy cost to reach its neighbors. Third, the relay estimates the energy cost around itself, and forwards the estimated value to the next-hop node. Each source node records its data generation rate b(s) on the data packets it transmits. By inspecting the source address and $b(s)$ recorded on the received data packets, relay i can determine the data rate $w(i, j)$ across the link (i, j), where j is a neighbor node of i. Thus, the

relay can estimate the total energy cost $E_{p,total}(i)$ around itself. Source nodes perform a similar computation, and record their addresses and the energy cost on outgoing packets. Relay i inspects the $E_{p,total}(i')$ value recorded on the packet, and if $E_{p,total}(i) > E_{p,total}(i')$ then it records its own address and energy cost on the packet, and forwards it. In consequence, the sink node will receive the address of the node that has the greatest energy cost. Fourth, each mobile relay controls its movement according to the mechanical analog; see the references for details.

The final form of the propulsion force on the relay, including reaction forces when radio obstacles exist, is:

$$\vec{F}_i = \sum_{j \in N(i)} w_{i,j} \cdot \vec{F}_{i,j} + \sum_{j \in Obst(i)} \vec{f}_{i,j} - K_v \cdot \dot{\vec{x}}_i$$

where $Obst(i)$ denotes the set of nodes adjacent to i, and the distance between a radio obstacle and the communication link with i is less than $d_{critical}$.

4.5.5. Simulation Results

First, we first consider the network space without radio obstacles, and compare the energy efficiency of the network structure produced by the DRP algorithm with that of Li and Cassandras' algorithm [69]. Unlike our algorithm, their relay placement algorithm is a centralized scheme with the assumptions similar to our mechanical model. Its operation is as follows. First, a bottleneck node k is chosen, and their algorithm investigates all possible ways of linking nodes around the bottleneck node, the number of which is $3 \cdot 2^m - 2$, where m is the number of k's neighbors. Next, it selects the best connectivity with the minimum power cost, and adds a new relay according to the selected connectivity. Then, for the given network connectivity, optimal node locations are determined through the so-called inner-force method. The algorithm continues to add relays until the intended number of relay nodes are inserted. For a fair comparison, we adopt the radio parameters from [69]. Figure 39 shows simulation results for the DRP algorithm without radio obstacles.

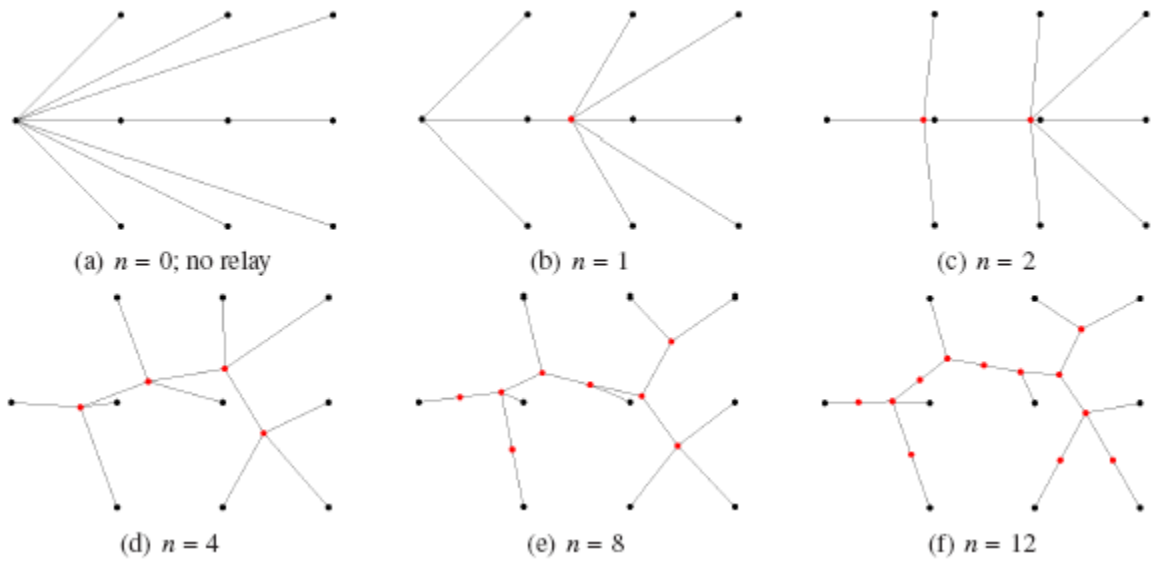


Figure 39. Simulation results for placing *n* relays with no radio obstacles.

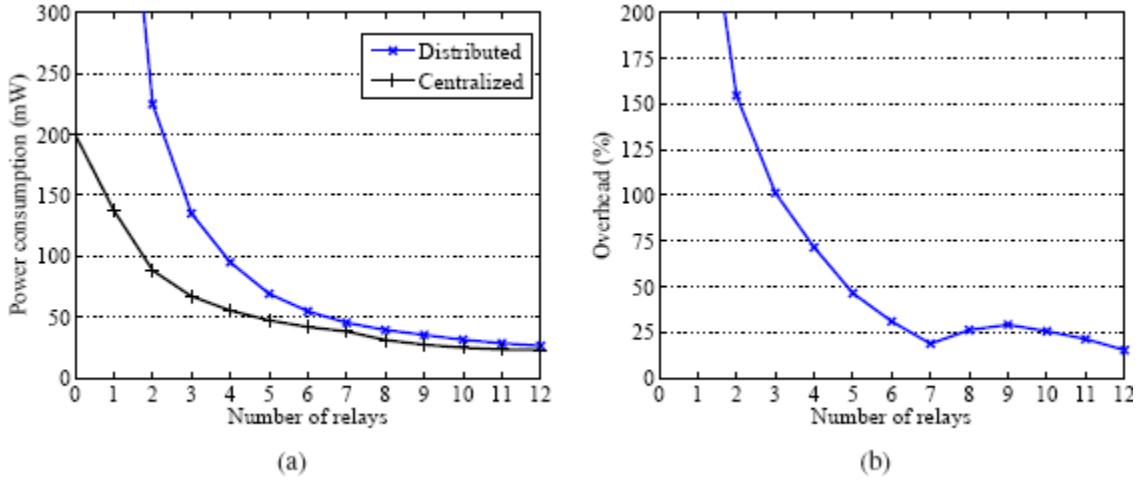


Figure 40. Simulation results for power consumption of relay placement algorithms: (a) Total power consumption, and (b) Power overhead due to the distributed implementation.

Figure 40 compares the power consumption of our DRP algorithm and that of Li and Cassandras. As Figure 40(a) shows, the power consumption of each algorithm decreases as the number of added relays n increases, and the gap between these algorithms also decreases. Let $P_{distributed}$ and $P_{centralized}$ denote the total power consumption of the algorithms. Figure 40(b) shows the relative overhead of the power consumption measured as $(P_{distributed} - P_{centralized})/P_{centralized}$. When n is small, the overhead is large, but as n grows, it decreases and becomes as small as 25%. The DRP algorithm constructs a power-efficient network structure in a distributed manner, and its power efficiency is comparable to that of Li and Cassandras' centralized algorithm.

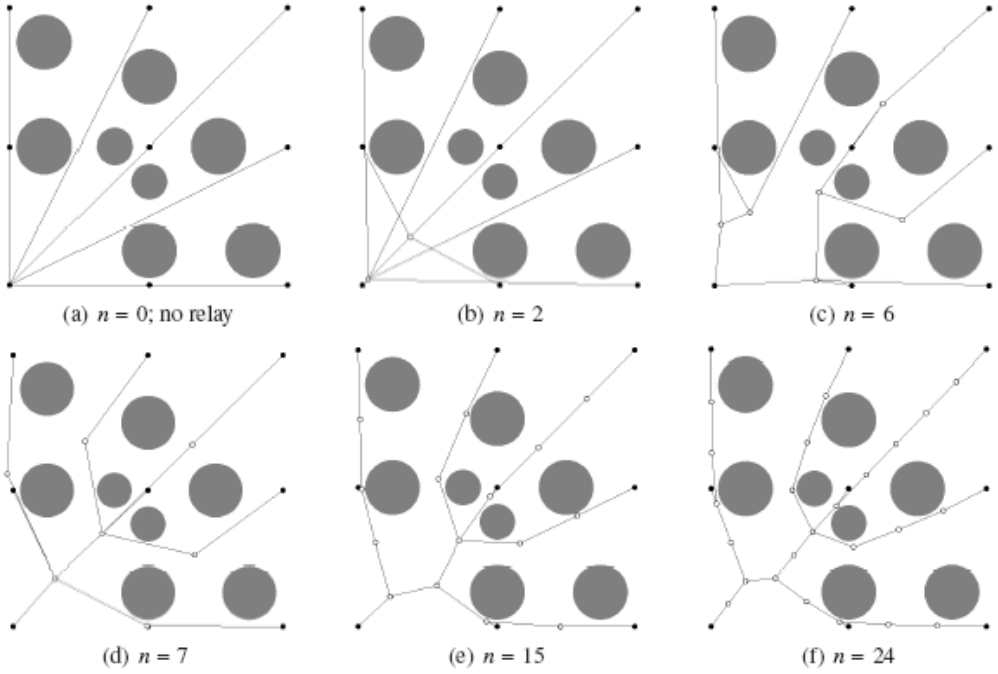


Figure 41. Relay placement with obstacles; *n* denotes the number of added relays.

Figure 41 provides simulation results for relay placement in the presence of radio obstacles. Nine source nodes (black) are placed in a grid structure on a plane. The sink node is at the bottom left, and the other eight source nodes are data sources. There are eight radio obstacles (grey disks), and *n* mobile relays (white) are inserted. It can be seen that the relay arrangement maintains connectivity in the presence of the radio obstacles.

Figure 42(a) shows simulation results for total power consumption obtained by the DRP algorithm in the presence of radio obstructions arranged as in Figure 41; results without radio obstructions are also shown. As expected, the total power consumption monotonically decreases as the number of relays increases. Figure 42(b) shows the ratio between the power consumption values with and without obstructions. It can be seen that if we add about six relays, the power overhead due to obstructions stays almost constant with respect to the case without obstructions. Thus, we conclude that the proposed DRP technique effectively reduces the total power consumption, even in the presence of obstructions, while maintaining the network's connectivity.

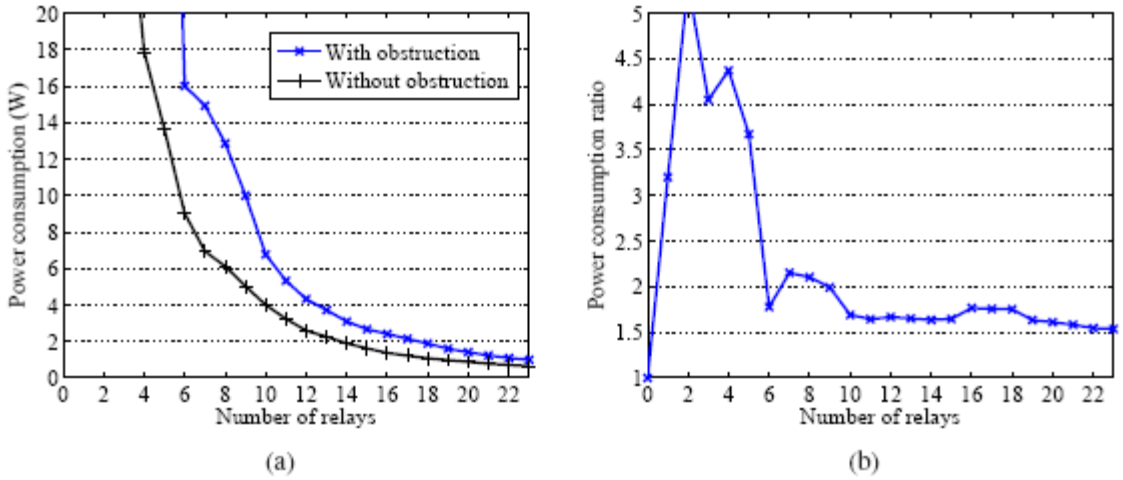


Figure 42. Simulation results for the proposed DRP algorithm: (a) Total power consumption, and (b) Ratio of power consumption levels with and without obstacles.

5. Simulation and Verification of Quantum Circuits

5.1. Introduction

The recent interest in quantum circuits is motivated by several complementary considerations. Quantum information-processing is rapidly becoming a reality as it allows manipulating matter at unprecedented scale. Such manipulations may create particular entangled states or implement specific quantum evolutions — they find uses in atomic clocks, ultra-precise metrology, high-resolution lithography, optical communication, etc. In August 2009, EE Times reported that "*researchers at the National Institute of Standards and Technology…demonstrated continuous quantum operations using a trapped-ion processor*" that stored quantum states in beryllium ions for up to 15 seconds. A large-scale architecture for quantum computing proposed in June 2009 by physicists from Michigan and Maryland suspends linear ion crystals in an anharmonic trap. Their design provisions for 100 ytterbium-based logical qubits and 20 additional ions for laser cooling. Commercial quantum communications and cryptography have so far relied on quantum-optical implementations, where qubits are stored in photons and can be transported over great distances.

Engineers traditionally simulate new designs before implementing them. Such simulation may identify subtle design flaws and save both costs and effort, and is therefore closely related to verification. Both simulation and verification techniques typically use well-understood host hardware, e.g., one can simulate a quantum circuit on a commonly-used conventional computer. However, working with even the smallest quantum circuits (2-3 qubits) requires powerful software tools. A quantum circuit can simultaneously process a *superposition* (linear combination) of many n-bit input combinations, and its functionality is specified by a complex-valued $(2^n \times 2^n)$-matrix. Because of the apparent data explosion, simulation and verification of quantum circuits are harder than in the case of conventional circuits. However, transformations of quantum circuits, e.g., adapting circuits to spin-chain architectures, may introduce errors that need to be verified.

Circuit equivalence checking is a particularly popular circuit verification technique. It is usually considered in conjunction with circuit optimization, where an initial circuit is believed to be correct, but the optimized circuit requires verification. One then needs to check if the two circuits produce equivalent outputs on all inputs. Equivalence of conventional circuits can be checked by random simulation, i.e., by looking for input combinations that would disprove equivalence. However, non-exhaustive simulation may overlook rare corner cases where the two circuits are different, and therefore cannot definitively conclude that two circuits are equivalent. Such conclusions can be produced instead by formal proof techniques. To this end, our work is the first to develop equivalence-checking techniques for quantum circuits and implement them in reusable software. We also found that the links between quantum circuit simulation and equivalence-checking are stronger than those for conventional circuits.

The construction of quantum information processors is often motivated by the hope that quantum circuits can compete with conventional computing and communication. Quantum-mechanical effects may lead to computational speed-ups, more secure or more efficient communication, better keeping of secrets, etc. To this end, Siemens and ID Quantique announced in August 2009 the commercial availability in Europe of quantumly-secure communication implemented in existing "dark" optical fiber. Current research seeks new circuits and algorithms with revolutionary behavior as in Shor's work on number-factoring or provable limits on possible behaviors. While

proving abstract limitations on the success of unknown algorithms appears more difficult, a common line of reasoning for such results is based on simulation. For example, if the behavior of a quantum circuit can be faithfully simulated on a conventional computer, then the possible speed-up achieved by the quantum circuit is limited by the cost of simulation. Thus, aside from sanity-checking new designs for quantum information-processing hardware, more efficient simulation can lead to sharper bounds on all possible algorithms.

Since the outcome of a quantum computation is probabilistic, we clarify our notion of simulation. By a *randomized* simulation, we mean a classical randomized algorithm whose output distribution on an input is identical to that of the simulated quantum computation. By a *deterministic* simulation, we mean a classical deterministic algorithm which, on a given pair of input x and output y of the quantum computation, outputs the probability that y is observed at the end of the quantum computation on x. To simulate a quantum circuit, one may use a naive brute-force calculation of quantum amplitudes that entails exponential overhead. Achieving significantly smaller overhead in the generic case appears hopeless — in fact, this has lead Feynman to suggest that quantum computers may outperform conventional ones in some tasks. Therefore, in the existing literature, theoretical results for simulating quantum circuits are mostly available for restricted classes of circuits. Our work offers new such classes and new results, with application to some of the most important quantum circuits in existence. In particular, follow-up work by other researchers (Aharonov and Short) applied our techniques to show that the QFT can be simulated in polynomial time on classical computers.

5.2. Algorithms for Quantum Circuit Simulation

Classes of quantum circuits that admit efficient simulation are often distinguished by a restricted "gate library", but do not impose additional restrictions on how gates are interconnected or sequenced. A case in point is the seminal Gottesman-Knill Theorem [106] and its recent improvement by Aaronson and Gottesman [94]. These results apply only to circuits with stabilizer gates—Controlled-NOT, Hadamard, Phase, and single-qubit measurements in the so-called Clifford group. Another example is given by *match gates* defined and studied by Valiant [127], and extended by Terhal and DiVincenzo [125].

A different way to impose a restriction on a class of quantum circuits is to limit the amount of entanglement in intermediate states. Jozsa and Linden [110], as well as Vidal [130] demonstrate efficient classical simulation of such circuits and conclude that achieving quantum speed-ups requires more than a bounded amount of entanglement. In this work, we pursue a different approach to efficient simulation and allow the use of arbitrary gates. More specifically, we assume a general quantum circuit model in which a gate is a general quantum operation (so called *physically realizable operators*) on a constant number of qubits. This model, proposed and studied by Aharonov, Kitaev and Nisan [95], generalizes the standard quantum circuit model, defined by Yao [134], where each gate is unitary and measurements are applied at the end of the computation. We also assume that (i) the computation starts with a fixed unentangled state in the computational basis, and (ii) at the end each qubit is either measured or traced-out.

Our simulation builds upon the framework of tensor network contraction. Being a direct generalization of matrices, tensors capture a wide range of linear phenomena including vectors, operators, multi-linear forms, etc. They facilitate convenient and fundamental mathematical tools in many branches of physics such as fluid and solid mechanics, and general relativity [108]. More recently, several methods have been developed to simulate quantum evolution by contracting variants of tensor networks, under the names of Matrix Product States, Projected Entangled Pairs States, etc. [118, 128, 129, 130, 131, 135]. In this framework, a quantum circuit is regarded as a network of tensors. The simulation contracts edges one by one and performs the convolution of the corresponding tensors, until there is only one vertex left. Having degree 0, this vertex must be labeled by a single number, which gives the final measurement probability sought by simulation. Unlike other simulation techniques, we do not necessarily simulate individual gates in their original order — in fact, a given gate may even be simulated *partially* at several stages of the simulation. While tensor network contraction has been used in previous work, little was known about optimal contraction orders. We prove that the minimal cost of contraction is determined by the *treewidth* $\mathrm{tw}(GC)$ of the circuit graph GC. Moreover, existing constructions that approximate optimal *tree-decompositions* (e.g. [122]) produce near-optimal contraction sequences. Intuitively, the smaller a graph's treewidth is, the closer it is to a tree, and a tree decomposition is a drawing of the graph to make it look like a tree as much as possible. Our result allows us to leverage the extensive graph-theoretical literature dealing with the properties and computation of treewidth.

Theorem 1.1. *Let C be a quantum circuit with T gates and whose underlying circuit graph is GC. Then C can be simulated deterministically in time $T^{O(1)} \exp[O(\mathrm{tw}(GC))]$.*

Hence given a function computable in polynomial time by a quantum algorithm but not classically, any polynomial-size quantum circuit computing the function must have super-logarithmic treewidth. The following corollary is an immediate consequence.

Corollary 1.2. *Any polynomial-size quantum circuit of a logarithmic treewidth can be simulated deterministically in polynomial time.*

Quantum formulas defined and studied by Yao [134] are quantum circuits whose underlying graphs are trees. Roychowdhury and Vatan [124] showed that quantum formulas can be efficiently simulated deterministically. Since every quantum formula has treewidth 1, Corollary gives an alternative efficient simulation. Our focus on the *topology* of the quantum circuit allows us to accommodate arbitrary gates, as long as their qubit-width (number of inputs) is limited by a constant. In particular, Corollary 1.2 implies efficient simulation of some circuits that create the maximum amount of entanglement in a partition of the qubits, e.g., a layer of two-qubit gates. Therefore, our results are not implied by previously published techniques. We now articulate some implications of our main result to classes of quantum circuits, in terms of properties of their underlying graphs. The following two classes of graphs are well-studied, and their treewidths are known. The class of *series parallel graphs* arises in electric circuits, and such circuits have treewidth ≤ 2. Planar graphs G with n vertices are known to have treewidth $\mathrm{tw}(G) = O(\sqrt{|V(G)|})$ [97].

Corollary 1.3. *Any polynomial-size parallel serial quantum circuit can be simulated deterministically in polynomial time.*

Corollary 1.4. *A size-T planar quantum circuit can be simulated deterministically in* $\exp[O(\sqrt{T})]$ *time.*

Another corollary deals with a topological restriction representative of many physical realizations of quantum circuits. A circuit is said to be *q*-local-interacting if under a linear ordering of its qubits, each gate acts only on qubits that are at most *q* distance apart. A circuit is said to be local-interacting if it is *q*-local interacting with a constant *q* independent of the circuit size. Such *local interaction* circuits generalize the restriction of qubit couplings to nearest-neighbor qubits (e.g., in a spin-chain) commonly appearing in proposals for building quantum computers, where qubits may be stationary and cannot be coupled arbitrarily. To this end, we observe that the treewidth of any local-interaction circuit of logarithmic depth is at most logarithmic.

Corollary 1.5. *Let C be a quantum circuit of size T and depth D, and is q-local-interacting. Then C can be simulated deterministically in* $T^{O(1)} \exp[O(qD)]$ *time. In particular, if C is a polynomial-size localy-linteracting circuit with a logarithmic depth, then it can be simulated deterministically in polynomial time.*

An important limitation of our techniques is that a circuit family with sufficiently fast-growing treewidth may require super-polynomial resources for simulation. In particular, this seems to be the case with known circuits for modular exponentiation. Therefore, there is little hope to efficiently simulate number-factoring algorithms using tree decompositions. As an extreme illustration, we found a depth-4 circuit—including the final measurement as the 4th layer—that has large treewidth.

Theorem 1.7. *There exists a depth-4 quantum circuit on n qubits using only one- and two-qubit gates such that its treewidth is* $\Omega(n)$.

Note that a circuit satisfying the assumption in the above theorem must have $O(n)$ size. Our construction is based on expander graphs, whose treewidth must be linear in the number of vertices. This finding is consistent with the obstacles to efficient simulation that are evident in the results of Terhal and DiVincenzo [126], later extended by Fenner et al. [107]. In contrast, we are able to efficiently simulate any depth-3 circuit *deterministically* while the simulation in [126] is probabilistic.

Theorem 1.8. *Assuming that only one- and two-qubit gates are allowed, any polynomial-size depth-*3 *quantum circuit can be simulated deterministically in polynomial time.*

Additional details are available in our extended paper published in the *SIAM Journal on Computing* 38(3), pp. 963-981, 2008.

5.3. Equivalence-Checking for Quantum Circuits

Equivalence checking is a basic task in the synthesis and verification of classical digital circuits. A hardware designer needs to know whether a circuit's implementation is functionally equivalent to its specification. In addition, the equivalence of different versions of the same (sub-)circuit must be checked throughout the complex computer-aided design process, which includes circuit optimization, technology mapping to specific device architectures and adaptation to compact and reliable layout. Combinational equivalence checking for conventional circuits is solved in practice with high-performance solvers for Boolean Satisfiability, and its negative version (non-equivalence) is NP-complete. Equivalence checking is likely to be just as important in quantum CAD, and the non-equivalence of quantum circuits is QMA-complete. However, the equivalence of quantum states and operators can be subtle. Unlike their classical counterparts, qubits and quantum gates can differ by global and relative phase, and yet be equivalent upon measurement.

Building upon our DARPA-sponsored work on QuIDD data structures for simulating quantum circuits, we developed QuIDD algorithms to check quantum states and operators for equivalence. Our research discovered a surprising variety of algorithms available to solve this problem. This variety is partly due to the fact that quantum circuits require the classical concept of equivalence to be extended to account for global and relative phase. This broader notion of equivalence creates several new opportunities in quantum circuit design, where minimizing the number of gate operations to achieve a given function is a fundamental goal. For example, the Toffoli gate can be implemented with fewer controlled-NOT (CNOT) and 1-qubit gates, if equivalence is interpreted as "equivalence up to relative phase" or, more briefly, "relative-phase equivalence." Normally the Toffoli gate requires an equivalent circuit of six CNOT and eight 1-qubit gates to implement it. Any relative-phase differences present in an equivalent circuit can be canceled out so long as every pair of these gates in the circuit is strategically placed. Since circuit minimization is being pursued for a number of key quantum arithmetic circuits with many Toffoli gates, such as the modular exponentiation occurring in Shor's algorithm, this type of phase equivalence could reduce the number of gates even further.

In our work we distinguish equivalence of quantum states up to global phase, from equivalence up to local phases. Neither affects measurement results if measurement is applied immediately, but may significantly alter the outcome if additional quantum gates are applied to given states. Furthermore, a pair of quantum operators or quantum circuits can be equivalent up to local or global phase if their outputs are respectively equivalent for all inputs.

5.3.1. Global-Phase Equivalence

We describe some algorithms that check equivalence up to global phase of two quantum states or operators. The first two are well-known linear-algebraic operations, while the remaining algorithms exploit QuIDD properties explicitly.

Inner Product. Since the quantum-circuit formalism models an arbitrary quantum state $|\psi\rangle$ as a unit vector, the inner product $\langle\psi|\psi\rangle = 1$. In the case of a global-phase difference between two states $|\psi\rangle$ and $|\varphi\rangle$, the inner product is the global-phase factor, $\langle\varphi|\psi\rangle = e^{i\theta}\langle\psi|\psi\rangle = e^{i\theta}$. Since $|e^{i\theta}| = 1$ for any θ, checking if the complex modulus of the inner product is 1 suffices to check global-phase equivalence for states. Although the inner product may be computed using explicit arrays, a straightforward QuIDD-based implementation is easily derived. The complex-conjugate transpose and matrix product with QuIDD operands were defined in our previous DARPA-funded work and implemented in software. Thus, the algorithm computes the complex-conjugate transpose of *A* and multiplies the result with I. The complexity of this algorithm is stated in the following lemma.

Lemma 5.2.1.A. Consider two state QuIDDs *A* and *B* with sizes |*A*| and |*B*|, respectively, in number of nodes. The global-phase difference can be computed in *O(|A||B|)* time and memory.

Matrix Product. The matrix product of two operators can be used for global-phase equivalence checking. In particular, since all quantum operators are unitary, the adjoint of each operator is its inverse. Thus, if two operators *U* and *V* differ by a global phase, then $UV\dagger = e^{i\theta}I$. With QuIDD representations of U and V, computing V† requires O(|V|) time and memory. The matrix product W = UV† requires $O((|U||V|)^2)$ time and memory. To check if $W = e^{i\theta}I$, any terminal value t is chosen from W, and scalar division is performed on W as W′ = Apply(W, t, /), which takes $O((|U||V|)^2)$ time and memory. Since QuIDDs are canonical, checking if W′ = I requires only *O(1)* time and memory. If W′ = I, then t is the global-phase factor.

Node-Count Check. The previous algorithms merely translate linear-algebraic operations to QuIDDs, but exploiting the following QuIDD property leads to faster checks.

Lemma 5.2.1.B. The QuIDD A′ = Apply(A, *c*, *), where *c* is a non-zero complex value, is isomorphic to A, hence |A′| = |A|.

This lemma states that two QuIDD states or operators that differ by a nonzero scalar, such as a global-phase factor, have the same number of nodes. Thus, the equality of node counts in QuIDDs is a necessary but not sufficient condition for global-phase equivalence. To see why it is not sufficient, consider two state vectors $|\psi\rangle$ and $|\varphi\rangle$ with elements w_j and v_k, respectively, where j, k = 0, 1, . . .$N - 1$. If some $w_j = v_k = 0$ such that $j \neq k$, then $|\varphi\rangle \neq e^{i\theta}|\psi\rangle$. The QuIDD representations of these states can, in general, have the same node counts. Despite this drawback, the node-count check requires only *O*(1*)* time since Apply is easily augmented to recursively sum the number of nodes as a QuIDD is created.

Recursive Check. Lemma 5.2.1.B implies that a QuIDD-based algorithm, which takes into account terminal value differences, checks a sufficient condition for global-phase equivalence. Pseudocode for such an algorithm called GPRC is shown in Figure 43.

GPRC returns *true* if two QuIDDs A and B differ by global phase and *false* otherwise. *gp* and *have_gp* are global variables containing the global-phase factor and a flag signifying whether or not a terminal node has been reached, respectively. The value of *gp* is only valid if *true* is returned. The first conditional block of GPRC deals with terminal values. The potential global-phase factor ngp is computed after handling division by 0. If $|ngp| \neq 1$ or if $ngp \neq gp$ when *gp* has been set, then the two QuIDDs do not differ by a global phase. Next, the condition specified by Lemma 5.2.1.B is checked. If the node of A depends on a different row or column variable than the node of B, then A and B are not isomorphic and thus cannot differ by global phase. Finally, GPRC is called recursively, and the results of these calls are combined via the logical AND operation.

```
GPRC(A, B, gp, have_gp) {
  if (Is_Constant(A) and Is_Constant(B)) {
    if (Value(B) == 0)  return (Value(A) == 0)
    ngp = Value(A)/Value(B)
    if (sqrt(real(ngp) * real(ngp)+
      imag(ngp) * imag(ngp)) != 1)
        return false
    if (!have_gp) {
      gp = ngp
      have_gp = true;
    }
    return (ngp == gp)
  }
  if ((Is_Constant(A) and !Is_Constant(B))
    or (!Is_Constant(A) and Is_Constant(B)))
      return false;
  if (Var(A)! = Var(B)) return false
  return (GPRC(Then(A), Then(B), gp, have_gp)
    and GPRC(Else(A), Else(B), gp, have_gp))
}
```

Figure 43. Pseudo-code for the recursive global-phase equivalence check.

Early termination occurs when isomorphism is violated or more than one phase difference is computed. In the worst case, both QuIDDs will be isomorphic, but the last terminal visited in each QuIDD will differ by more than a global-phase factor, causing full traversals of both QuIDDs. Thus, the overall runtime and memory complexity of GPRC for states or operators is $O(|A|+|B|)$. Also, the node-count check can be run before GPRC to quickly eliminate many non-equivalences.

Empirical Results. The first benchmark considered is a single iteration of Grover's quantum search algorithm, where the oracle searches for the last item in the database. One iteration is sufficient to test the effectiveness of the algorithms, since the state vector QuIDD remains isomorphic across all iterations, as was proven in our past DARPA-sponsored work.

Figure 44 shows the runtime results for the inner product and GPRC algorithms. Results for the node-count check algorithm are not shown since it runs in $O(1)$ time.

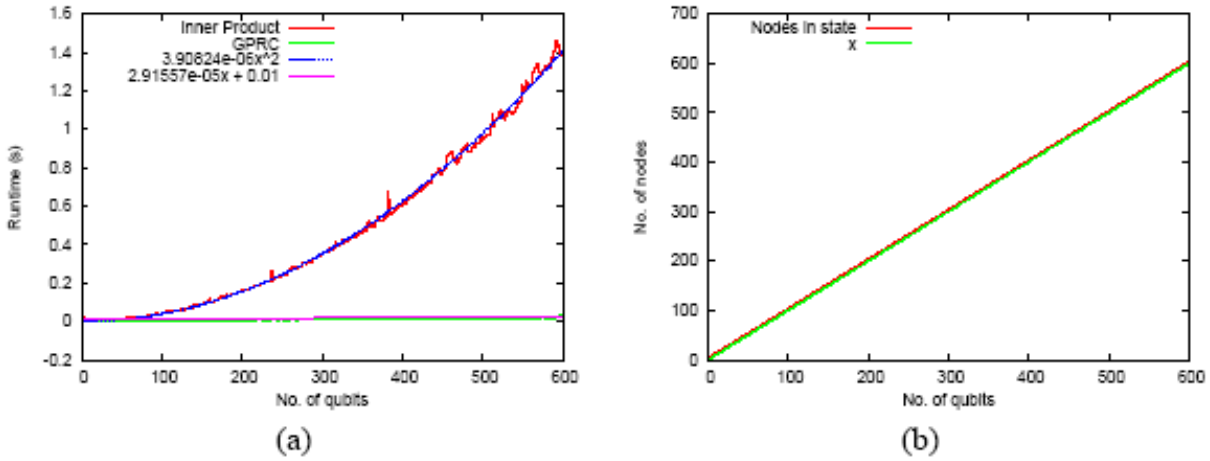


Figure 44. (a) Runtime results and regressions for the inner product and GPRC on checking global-phase equivalence in a Grover iteration, and (b) QuIDD size of the state vector.

These results confirm the asymptotic complexity difference between the inner-product and GPRC algorithms. Because the number of nodes in the QuIDD state-vector after a Grover iteration is $O(n)$, the runtime complexity of the inner product should be $O(n^2)$, which is confirmed by a regression plot within only a 1% error. In contrast, the runtime complexity of the GPRC algorithm should be $O(n)$, which is also confirmed by a regression plot, within the same error margin.

We next compared runtimes of the matrix-product and GPRC algorithms checking the Grover operator. Our previous work shows that the QuIDD representation of this operator grows in size as $O(n)$, which is confirmed in our empirical data. Therefore, the runtime of the matrix product should be quadratic in n, but linear in n for GPRC. Regression plots verify these complexities within 0.3% error. We also compared states that appear in Shor's integer factorization algorithm. In particular, we consider states created by the modular exponentiation sub-circuit that represent all possible combinations of x and $f(x,N) = a^x \bmod N$, where N is the integer to be factored. Each of the $O(2^n)$ paths to a non-0 terminal encodes a binary value for x and $f(x,N)$. This experiment demonstrates how the algorithms fare with exponentially-growing QuIDDs.

In our experiments, each N is an integer whose two non-trivial factors are prime (such numbers are often called semi-prime). a is set to $N-2$, but in general can be chosen randomly from the range $[2..N-2]$. In our experiments, states $|\psi\rangle$ and $|\varphi\rangle$ are equal up to global phase. The node counts for both states are equal as predicted by Lemma 5.2.1.B. Interestingly, both algorithms exhibit very similar performance. Further results were produced for the cases in which Hadamard gates are applied to the first, middle, and last qubits, respectively, of $|\varphi\rangle$. These results show that early termination in GPRC can enhance performance by a factor of roughly 1.5x to 10x. In almost every case, both algorithms represent far less than 1% of the total runtime.

Thus, checking for global-phase equivalence among QuIDD states appears to be easy once the QuIDDs are created. An interesting side note is that in modular exponentiation, some QuIDD states with more qubits have more exploitable structure than those with fewer qubits. For instance, the $N = 387929$ (19 qubits) QuIDD has fewer than half the nodes of the $N = 163507$ (18 qubits) QuIDD. We also compared the matrix-product and GPRC algorithms checking the inverse QFT

operator. While the inverse QFT is key to Shor's algorithm, its *n*-qubit QuIDD representation grows as $O(2^{2n})$, and the asymptotic differences in the matrix product and GPRC are very noticeable. Also, the memory usage indicates that the matrix product may need asymptotically more intermediate memory despite operating on QuIDDs with the same number of nodes as GPRC.

5.3.2. Relative-Phase Equivalence

Like the global-phase case, the relative-phase equivalence checking problem can be solved in several ways. Our first three algorithms adapt standard linear algebra to QuIDDs, while the last two exploit QuIDD properties directly, offering asymptotic runtime and memory improvements.

Modulus and Inner Product. Consider two state vectors $|\psi\rangle$ and $|\varphi\rangle$ that are equal up to relative phase and have complex-valued elements w_j and v_k, respectively, where $j, k = 0, 1, \ldots N-1$. Computing $|\varphi'\rangle = \Sigma_{i=0}^{N-1} |v_j| |j\rangle$ and $|\psi'\rangle = \Sigma_{k=0}^{N-1} |w_k| |k\rangle = \Sigma_{k=0}^{N-1} |e^{i\theta_k} v_k| |k\rangle$ sets each phase factor to 1, allowing the inner product to be applied. The complex modulus operations are computed as C = Apply(A, $|\cdot|$) and D = Apply(B, $|\cdot|$) with runtime and memory complexity $O(|A| + |B|)$, which is dominated by the $O(|A||B|)$ complexity of the inner product.

Modulus and Matrix Product. For operator equivalence up to relative phase, there are two cases depending on whether the diagonal relative-phase matrix appears on the left or right side of one of the operators. Consider two operators U and V with elements $u_{j,k}$ and $v_{j,k}$, respectively. The two cases in which the relative-phase factors appear on either side of V are described as $u_{j,k} = e^{i\theta_j} v_{j,k}$ (left side) and $u_{j,k} = e^{i\theta_k} v_{j,k}$ (right side). In either case, the matrix product check discussed above may be extended by computing the complex modulus without increasing the overall complexity. Note that neither this algorithm nor the modulus and inner-product algorithms calculate the relative-phase factors.

Element-wise Division. Given the states discussed for the modulus and inner product check, $w_k = e^{i\theta_k} v_k$, the operation w_k / v_j for each $j = k$ produces a relative-phase factor, $e^{i\theta_k}$. The condition $|w_k / v_j| = 1$ is used to check if each division yields a relative phase. If this condition is satisfied for all divisions, the states are equal up to relative phase.

The QuIDD implementation for states is simply C = Apply(A,B, /), where Apply is augmented to avoid division by 0 and instead return 1 when two terminal values being compared equal 0, and return 0 otherwise. Apply can be further augmented to terminate early when $|w_j / v_i| \neq 1$. *C* is a QuIDD vector containing the relative-phase factors. If *C* contains a terminal value 0, then *A* and *B* differ by more than a relative phase. Since one call to Apply implements this algorithm, the runtime and memory complexity are $O(|A||B|)$.

Element-wise division for operators is more complicated. For QuIDD operators U and V , W = Apply(U, V, /) corresponds to a matrix with the relative-phase factor $e^{i\theta_j}$ along row *j* in the case of phases appearing on the left side, and along column *j* in the case of phases appearing on the right side. In the first case, all rows of W are identical, meaning that the support of W does not contain any row variables. Similarly, in the second case the support of W does not contain any column variables. A complication arises when 0 values appear in either operator. In such cases, the support of W may contain both variable types, but the operators may in fact be equal up to relative phase.

We now present an algorithm based on Apply, which accounts for these special cases by using a sentinel value of 2 to mark valid 0 entries that do not affect relative-phase equivalence. These entries are recursively ignored by skipping either row or column variables with sentinel children (S specifies row or column variables), which effectively fills copies of neighboring row or column phase values in their place in W. The algorithm must be run twice, once for each variable type. The size of W is *O(|U||V|)* since it is created by a variant of Apply.

```
RP_DIV(A,B,S) {
  if (A == New_Terminal(0)) {
    if (B != New_Terminal(0))
      return New_Terminal(0)
    return New_Terminal(2)
  }
  if (Is_Constant(A) and Is_Constant(B)) {
    nrp = Value(A)/Value(B)
    if (sqrt(real(nrp) * real(nrp)+
      imag(nrp) * imag(nrp)) != 1)
        return New_Terminal(0)
    return New_Terminal(nrp)
  }
  if (Table_Lookup(R,RP_DIV,A,B,S)) return R;
  v = Top_Var(A,B)
  T = RP_DIV(A_v, B_v, S)
  E = RP_DIV(A_v', B_v', S)
  if ((T == New_Terminal(0)) or
    (E == New_Terminal(0)))
      return New_Terminal(0)
  if ((T != E) and (Type(v) == S)) {
    if (Is_Constant(T) and Value(T) == 2)
      return E
    if (Is_Constant(E) and Value(E) == 2)
      return T
    return New_Terminal(0)
  }
  if (Is_Constant(T) and Value(T) == 2)
    T = New_Terminal(1)
  if (Is_Constant(E) and Value(E) == 2)
    E = New_Terminal(1)
  R = ITE(v,T,E)
  Table_Insert(R,RP_DIV,A,B,S)
  return R
}
```

Figure 45. Pseudo-code for the element-wise division algorithm.

Non-0 Terminal Merge. A necessary condition for relative-phase equivalence is that zero-valued elements of each state vector appear in the same locations, as expressed by the following lemma.

Lemma 5.2.2. A necessary but not sufficient condition for two states $|\varphi\rangle = \Sigma_{j=0}^{N-1} v_j |j\rangle$ and $|\psi\rangle = \Sigma_{k=0}^{N-1} w_k |k\rangle$ to be relative-phase equivalent is that whenever $v_j = w_k = 0$, $j = k$.

QuIDD canonicity may be exploited with this condition. Let A and B be the QuIDD representations of states $|\psi\rangle$ and $|\varphi\rangle$, respectively. First, compute C = Apply(A, ⌈| · |⌉) and D = Apply(B, ⌈| · |⌉), which converts every non-zero terminal value of A and B into a 1. Since C and D have only two terminal values, 0 and 1, checking if C and D are equal satisfies Lemma 5.2.2. Canonicity ensures this check requires *O*(1) time and memory. The overall runtime and memory complexity of this algorithm is *O(|A|+|B|)* due to the unary Apply operations. This algorithm can also be applied to operators, since Lemma 5.2.2 also applies to $u_{j,k} = e^{i\theta_j} v_{j,k}$ (phases on the left) and $u_{j,k} = e^{i\theta_k} v_{j,k}$ (phases on the right) for operators U and V.

Modulus and DD Compare. A variant of the modulus and inner-product check, which also exploits the canonicity of QuIDDs, provides an asymptotic improvement when checking a necessary and sufficient condition for relative-phase equivalence of states and operators. As with the modulus and inner product check, compute C = Apply(A, |·|) and D = Apply(B, |·|). If A and B are equal up to relative phase, then C = D since each phase factor becomes a 1. Canonicity again ensures that this check requires *O(1)* time and memory. Thus, the runtime and memory complexity of this algorithm is dominated by the unary Apply operations, giving *O(|A| + |B|)*.

5.3.3. Empirical Validation

We now present experimental results for the relative-phase equivalence-checking algorithms. The first benchmark circuit creates a remote EPR pair between the first and last qubits, via nearest-neighbor interactions. Given an initial state |00 . . . 0>, it creates the remote EPR-pair state $(1/\sqrt{2})(|00\ldots0\rangle + |10\ldots1\rangle)$. The circuit size is varied, and the final state is compared to the state $(e^{0.345i}/\sqrt{2})\,|00\ldots0\rangle + (e^{0.457i}/\sqrt{2})\,|10\ldots1\rangle$. Our data show that all of our algorithms run quickly. For example, the inner product is the slowest algorithm, yet for a 1000-qubit instance, it runs in 0.2 seconds, a small fraction of the 7.6 seconds required to create the QuIDD state vectors.

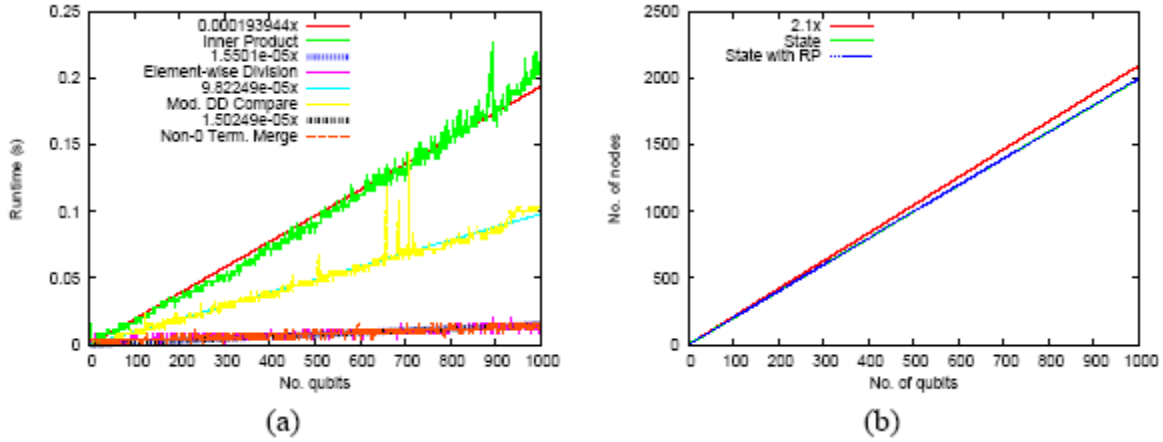


Figure 46. (a) Runtime results and regressions for various algorithms to check for relative phase equivalence of the remote EPR pair circuit, and (b) Size of the QuIDD states.

Curve-fitting of the runtime and memory data reveal linear complexity for all algorithms to within 1% error. This is not unexpected since the QuIDD representations of the states grow linearly with the number of qubits, and the complex modulus reduces the number of different terminals prior to computing the inner product. These results suggest that in practice, the inner-product and element-wise division algorithms perform better than their worst-case complexity. Element-wise division should be preferred when QuIDD states are compact since, unlike the other algorithms, it computes the relative-phase factors (this was also confirmed by another series of experiments reported in our ICCAD 2007 publication detailing this work).

5.3.4. Summary

Although QuIDD properties like canonicity enable exact equivalence checking in O(1) time, we have shown that such properties may be further exploited to develop efficient algorithms for the difficult problem of equivalence checking up to global and relative phase. In particular, the global-phase recursive check and element-wise division algorithms efficiently determine equivalence of states and operators up to global and relative phase, while also computing the phases. In practice, they outperform QuIDD implementations of the inner and matrix product, which do not compute relative-phase factors. Other QuIDD algorithms presented in this chapter, such as the node-count check, non-0 terminal merge, and modulus and DD compare, further exploit decision-diagram properties to provide even faster checks, but only satisfy necessary conditions for equivalence. Thus, they should be used to aid the more robust algorithms. A summary of our theoretical results on equivalence-checking is given in Figure 47.

Algorithm	Phase type	Finds phases?	Necessary & sufficient?	$O(\cdot)$ time complexity: best-case	$O(\cdot)$ time complexity: worst-case
Inner product	Global	Yes	Yes	$\|A\|\|B\|$	$\|A\|\|B\|$
Matrix product	Global	Yes	Yes	$(\|A\|\|B\|)^2$	$(\|A\|\|B\|)^2$
Node-count	Global	No	Nec. only	1	1
Recursive check	Global	Yes	Yes	1	$\|A\| + \|B\|$
Modulus and inner product	Relative	No	Yes	$\|A\|\|B\|$	$\|A\|\|B\|$
Element-wise division	Relative	Yes	Yes	$\|A\|\|B\|$	$\|A\|\|B\|$
Non-0 terminal merge	Relative	No	Nec. only	$\|A\| + \|B\|$	$\|A\| + \|B\|$
Modulus and DD compare	Relative	No	Yes	$\|A\| + \|B\|$	$\|A\| + \|B\|$

Figure 47. Key properties of the QuIDD-based phase-equivalence checking algorithms.

5.3.5. Adaptive Equivalence-Checking

To develop a more efficient methodology for equivalence-checking of quantum circuits, we exploit the reversibility of these circuits to define a new concept, called a *reversible miter* — a natural counterpart of miter circuits used in equivalence-checking of conventional circuits. In conjunction with existing techniques for iterative circuit simplification, reversible miters can drastically reduce the size of verification instances. Our method is *adaptive* in the sense that it utilizes multiple techniques appropriate for different classes of quantum circuit modules. In this context, we study reversible circuits which are a subset of quantum circuits that map conventional 0-1 bit-strings into other such bit-strings. In particular, the largest module in Shor's number-factoring algorithm—modular exponentiation—is implemented as a reversible circuit, but acting on entangled quantum states. It exceeds all other modules asymptotically in size, and thus requires most attention of CAD tools. To verify such logic modules, we adapt conventional state-of-the-art techniques in several ways, and significantly scale up quantum equivalence checking. Our empirical comparisons confirm that properties of reversible circuits can enable much faster SAT-based equivalence-checking. However, conventional techniques cannot be applied to, for example, the Quantum Fourier Transform. Therefore, we also study equivalence-checking of circuits with non-conventional gates (properly-quantum circuits), and the integration of heterogeneous techniques.

5.3.6. Preliminaries

Consider gates NOT, CNOT and TOFFOLI acting on classical bits. They can be implemented using NOT, XOR and AND gates. In the quantum case, they exchange basis states, which is why their matrices contain only 0s and 1s. As these gates obey the same algebraic rules in both cases, we term them *conventional* gates. In comparison, the matrix of the Hadamard gate contains $1/\sqrt{2}$, and its functionality cannot be expressed in Boolean logic. Therefore we call such gates *properly-quantum*. Each properly-quantum gate maps at least one 0-1 input combination (basis state) to a quantum superposition of more than one basis state. Measurement-free quantum computation is reversible in nature, thus quantum circuits are reversible in that (i) they map their input configurations to output configurations one-to-one, (ii) this property is also observed locally for every gate and sub-circuit.

Popular quantum algorithms contain large, application-specific sections dedicated to the computation of conventional Boolean functions. In some cases, the input to a quantum algorithm is read by these conventional sections and later factored into quantum states. In order to embed conventional computation into the quantum domain, it must be made reversible, and standard procedures exist for such transformations. The resulting circuits do not use quantum properties, except that they can be applied to quantum data (superposition states), allowing them to perform classical operations on many inputs at once. Leveraging this quantum parallelism in useful applications is difficult, but can be illustrated by Shor's polynomial-time algorithm for number-factoring [147]. This algorithm is dominated by a reversible module that performs modular exponentiation, applied before the QFT. We call such circuits without properly-quantum gates specifically reversible circuits here.

5.3.7. Reversible Miters

To check the equivalence of two conventional combinatorial circuits, C_1 and C_2, one checks if the conventional miter circuit XOR(C_1,C_2) implements the constant-0 function. For multi-output circuits, corresponding output bits are XORed and the results are ORed, so that even a single mismatch can be detected by existing powerful tools for Boolean circuit analysis. Conventional miters can be constructed for reversible circuits by treating these circuits as AND-OR-NOT circuits, except that the miter will not be reversible. However, this is not going to work for properly-quantum circuits. To address these limitations, we introduce *reversible miters* which exploit reversibility and can handle properly-quantum gates. We also discover synergies with simplification of reversible circuits [140], [142], [143]. Now consider quantum (or reversible) circuits C_1 and C_2. Recall that the concatenation $C_1 \cdot C_2$ of C_1 and C_2 is also a quantum (or reversible) circuit.

Observation 1: Given a quantum (or reversible) circuit $C = g_1 \cdot g_2 \cdot \ldots \cdot g_k$ where g_i is a gate, its copy where all gates are inverted and put in the reverse order, i.e., $g_k^{-1} \cdot \ldots \cdot g_2^{-1} \cdot g_1^{-1}$, implements the inverse transformation to what C implements. We therefore denote it by C^{-1}. Note that NOT, CNOT, and Toffoli gates are their own inverses. The circuit $C \cdot C^{-1}$ is equivalent to an empty circuit. This can be confirmed by iteratively cancelling out pairs of mutually-inverse adjacent gates. This observation motivates our new notion of *reversible miters*.

Definition 1: Given two quantum (or reversible) circuits C_1 and C_2, their *reversible miter* is defined to be one of the following circuits: $C_1 \cdot C_2^{-1}$, $C_2^{-1} \cdot C_1$, $C_2 \cdot C_1^{-1}$, $C_1^{-1} \cdot C_2$. In particular, for conventional miters one needs to check that the output functions implement the constant 0 function, whereas for reversible miters one checks that each output bit is equivalent to a corresponding input bit. Namely, C_1 and C_2 are functionally equivalent if and only if all of their reversible miters implement the identity transformation. In particular, if one miter implements the identity, then so do the remaining miters. If $C_1 = C_2$, then straightforward circuit simplification [140, 142, 143] cancels out all gates, resulting in an empty circuit. Some of the variant miters enable more cancellations than others, e.g., if C_1 and C_2 differ only in their first segments, $C_2 \cdot C_1^{-1}$ exhibits many gate cancellations. Reversible miters speed up equivalence-checking by exploiting similarities in circuits by two distinct mechanisms.

1) Local Reduction of Reversible Miters: When two conventional circuits end with identical gate sequences, one cannot cancel out these sequences because of observability don't-cares introduced by them. However, reversible circuits do not experience don't-cares, and identical suffixes always cancel out. Note that a reversible miter $C_1 \cdot C_2^{-1}$ places the last gate of C_1 next to the last gate of C_2. If these two gates cancel out, the second-to-last gates from C_1 and C_2 become adjacent, etc. Thus, no search is required to identify these gate cancellations, and they can be performed one at a time. Even if the last two gates are different, it may be possible to cancel out second-to-last gates, as long as the last and second-to-last gates do not act on the same (qu)bit lines. These are special cases of much more general *local reductions* discussed in [140, 142, 143]. If C_1 and C_2 are identical, an empty circuit will result, but this outcome is also possible when local reductions can prove equivalence of two structurally different circuits. A systematic procedure for applying reductions was introduced in [143]. Local reductions in reversible circuits are particularly easy to perform, are fast and do not consume much memory [140, 142].

In our experiments, even the simplest reduction rules can dramatically simplify reversible miters. More sophisticated reductions from [140, 142, 143] provide an additional boost.

We experimented with the following reduction procedure. In a miter circuit, consider one gate at a time, search for a matching inverse, and try to move them together to facilitate cancellation. Any two gates can be swapped if they do not act on the same (qu)bit lines. Two adjacent NOT, CNOT or Toffoli gates can be swapped if the control bit of one gate is not the target bit of the other gate (same for properly-quantum controlled-U gates). In our procedure, for the purposes of equivalence-checking, we temporarily consider the miter circuit to be "circular" by connecting its outputs to its inputs. Namely, we allow moving the last gate to the beginning of the. This transformation does not change the equivalence of the entire circuit to the identity. In other words, if $g_1 \cdot g_2 \cdot \ldots \cdot g_{k-1} \cdot g_k = I$ (Identity), then $g_k \cdot g_1 \cdot g_2 \cdot \ldots \cdot g_{k-1} \cdot g_k \cdot g_k^{-1} = g_k \cdot I \cdot g_k^{-1} = g_k \cdot g_k^{-1} = I$. Therefore, to check equivalence between $g_1 \cdot g_2 \cdot \ldots \cdot g_{k-1} \cdot g_k$ and $\underline{I}$ is the same as to check equivalence between $g_k \cdot g_1 \cdot g_2 \cdot \ldots \cdot g_k^{-1}$ and I.

2) Reduction of Canonical Forms: Iterative circuit simplification is not guaranteed to reduce $C_1 \cdot C_2^{-1}$ to the empty circuit in polynomial number of steps when such a reduction is possible. Finding a short reduction may be time-consuming. However, when constructing canonical forms (ROBDDs or QuIDDs) of reversible miters, a different kind of reduction may occur. Suppose that C_1 and C_2 end with functionally-equivalent but structurally distinct suffixes that do not admit local reductions — an example is given in [5]. In other words $C_1 = A_1 \cdot B_1$ and $C_2 = A_2 \cdot B_2$ where $B_1 \approx B_2$. Then $C_1 \cdot C^{-1}{}_2 = A_1 \cdot B_1 \cdot B^{-1}{}_2 \cdot A^{-1}{}_2 \approx A_1 \cdot A_2^{-1}$. As we traverse the miter $C_1 \cdot C_2^{-1}$, adding one gate at a time to the decision diagram (DD), the size of the intermediate DDs depends only on the transformation implemented by the current circuit prefix, i.e., the functions of the intermediate wires. The intermediate DD for $A_1 \cdot B_1 \cdot B_2^{-1}$ can be smaller than that for $A_1 \cdot B_1$ if $A_1 \cdot B_1 \cdot B_2^{-1} \approx A_1$. This phenomenon was observed in our experiments.

5.3.8. Methodology for Equivalence Checking

We now introduce equivalence-checking for quantum circuits based on several techniques appropriate for different classes of quantum circuits. The first class contains reversible circuits that arise as key modules in quantum algorithms. To check the equivalence of two reversible circuits, C_1 and C_2, one can pursue two strategies. The first strategy is to check that the conventional miter implements the constant 0 function. A conventional miter can also be applied to reversible circuits as explained below. The second strategy is to represent the transformations performed by C_1 and C_2 in a canonical form which supports efficient equivalence-checking. The latter strategy may use binary-decision diagrams, such as reduced ordered binary decision diagrams (ROBDDs), and QuIDDs [149] or quantum multiple valued decision diagrams (QMDDs) [146]. The former can be implemented with either decision diagrams or Boolean Satisfiability solvers by reducing Circuit-SAT to CNF-SAT. In particular, for conventional miters one needs to check that the output functions implement the constant 0 function. In addition to the basic SAT or BDD-based approaches, finding equivalent signals in two circuits is often very helpful [141]. Such techniques appear useful for reversible circuits as well, as shown in our experiments. Relevant computational engines are discussed next.

ROBDD. Calculate the output functions of miter circuits, using ROBDD as the primary data structure. This technique cannot handle properly quantum circuits.

QuIDD. Build functional representations of given circuits C_1 and C_2, and check if the results are identical. In particular, QuIDDPro [149, 150] builds multi-terminal decision diagrams called QuIDDs that can capture properly-quantum circuits.

SAT. Given two reversible circuits, construct a CNF-SAT formula that is satisfied only by those input combinations for which the two circuits produce different outputs. Then use a contemporary SAT solver [137] to check satisfiability.

SAT-based techniques can be dramatically improved by identifying intermediate equivalences and then using them to break up a large SAT instance for the miter circuit into several smaller instances. The state-of-the-art implementation found in the Berkeley ABC system [136] (the "cec" command) uses bit-parallel functional simulation to identify potentially equivalent signals, then proves or disproves the equivalence using (incremental) SAT. Established equivalences simplify future SAT calls, while counterexamples found are used to refine the results of functional simulation and often distinguish seemingly-equivalent signals. ABC also uses *fraiging* — a fast simplification technique based on hashing. To use ABC in our experiments, we construct a conventional (irreversible) circuit from a reversible circuit by replacing each gate with its AND/XOR/NOT equivalent. Common benchmarks for reversible synthesis can be verified in milliseconds by the above techniques. Instead, we focus on scalable blocks of standard quantum algorithms, whose optimization and equivalence-checking are critical to the success of quantum computers being designed today. More concretely, we performed experiments with n-bit *linear-nearest-neighbor (LNN)* CNOT gate circuits, a reversible ripple-carry adder circuit proposed in [148], *mesh* circuits [138] and reversible multipliers. Given a (qu)bit ordering, a linear-nearest-neighbor CNOT gate circuit is a circuit which realizes the functionality of a CNOT gate with target and control bits k bits apart, by using only LNN gates (gates that operate only on adjacent qubits). Studies of LNN architectures are important because several promising implementations of quantum computation require the LNN architecture (also called the *spin-chain* architecture in the physics literature) and allow only adjacent qubits to interact directly. Thus, standard quantum circuits must be adapted to such architectures and modified to use only LNN gates. Specific transformations and LNN circuits have been developed [138, 145]. The overhead of the LNN architecture in terms of the number of gates is often limited by a small factor (3-5). Such physical-synthesis optimization motivates the need for equivalence-checking against the original, non-LNN versions. Using important components of Shor's algorithm [138, 147] — adders, meshes and multipliers — we build three types of equivalence-checking instances.

Same. Two equivalent circuits.

Different 1. Randomly add Toffoli gates at the end.

Different 2. Randomly add Toffoli gates at the beginning.

Our empirical data for CNOT, adder and mesh circuits exhibits essentially the same trends. Hence, we report results only for adders in Figure 48. All runtimes are for a Linux system with a 2.40GHz Intel XeonTM CPU with 1GB RAM. We implemented n-bit reversible multipliers using *5n* bitlines, including *2n* bits for two inputs, *2n* bits for the results, and *n* ancillae. For example., the line $n = 6$ in the tables deals with 30-bit circuits. All methods other than "cec" timed out for $n = 8$, requiring more than 1,000s.

ADDER VERIFICATION PERFORMED BY SEVERAL TECHNIQUES.

Case	n	SAT	QuIDD	BDD	cec
Same	32	0.65	20.10	0.03	0.19
	64	2.91	115.85	0.11	0.23
	128	11.71	771.20	0.52	0.31
Diff. 1	32	1.00	31.93	0.04	0.02
	64	5.16	212.57	0.25	0.26
	128	15.25	> 1,000	1.67	0.38
Diff. 2	32	1.09	40.40	0.09	0.02
	64	10.98	318.62	0.76	0.03
	128	22.72	> 1,000	9.88	0.03

TABLE II

MULTIPLIER VERIFICATION PERFORMED BY SEVERAL TECHNIQUES.

	n	SAT	QuIDD	BDD	cec
Same	4	1.86	50.45	0.09	0.00
	6	392.74	> 1,000	39.19	0.01
Diff. 1	4	0.02	72.84	0.01	0.01
	6	0.11	> 1,000	0.03	0.02
Diff. 2	4	0.02	95.94	0.01	0.02
	6	0.17	> 1,000	0.01	0.02

Figure 48. Runtime results for equivalence-checking of reversible arithmetic circuits.

B. Checking Properly-Quantum Circuits

In this section, we show that our proposed techniques can handle properly-quantum gates, but remain compatible with fast special-case methods.

1) Utility of Reversible Miters: Earlier sections focused on equivalence-checking of reversible circuits which appear in modules of quantum algorithms and require physical synthesis optimizations [138] that must be verified. However, other important modules in quantum algorithms, such as the *Quantum Fourier Transform*, are properly-quantum, and conventional circuits, such as *modular exponentiation*, can be optimized for performance using properly-quantum gates. Fortunately, simple cancellations in reversible miters can be used with properly-quantum circuits. Reduced properly-quantum miters can be verified using symbolic simulation with QuIDDPro [149] or QMDD software [146]. Using reversible miters as pre-processors can dramatically decrease overall runtime. We empirically compare the following two methods.

With Local Reduction. Before invoking QuIDDPro, reduce the miter through local simplification.

W/o Local Reduction. Apply QuIDDPro directly to the miter.

For properly-quantum circuit benchmarks, we used QFT and *modular exponentiation* modules from circuits that implement Shor's factorization algorithm on an LNN architecture [138]. For each benchmark circuit with n inputs, we studied five cases (new gates were added in the middle).

Same. Two identical copies of a benchmark circuit.

Different 1. A circuit and its copy with one gate added.

Different 2. A circuit and its copy with two gates added.

Different 3. A circuit and its copy with one gate deleted.

Different 4. A circuit and its copy with two gates deleted.

Empirical results in Table III, Figure 49 and Table VI Figure 50 show that in the "Same" case, our local simplifications are sufficient to conclude equivalence. Otherwise, QuIDDPro is invoked after simplification, even though only several gates remain. However, in "Diff. 2" many gates remain after simplification. In columns "Same" and "Diff. 1" we report runtimes for local simplification and the QuIDDPro calls. Time-outs are shown with ">1,000". Local reduction is useful even for properly-quantum circuits.

TABLE III
VERIFICATION OF QFT CIRCUITS WITH LOCAL REDUCTION.

n	Same		Diff. 1		Diff. 2		Diff. 3		Diff. 4	
	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD
4	0	0	0	0.03	0	0.05	0	0.04	0	0.05
8	0	0	0.01	0.03	0	0.17	0	0.04	0	0.26
16	0.05	0	0.07	0.05	0.08	0.26	0.06	0.04	0.07	0.05
32	0.73	0	1.11	0.04	1.13	9.17	0.99	0.04	1.22	0.08
64	17.29	0	24.32	0.05	25.48	0.52	24.33	0.06	30.35	0.12
128	354.52	0	366.2	0.04	497.21	> 1,000	522.57	0.04	580.11	0.39

TABLE IV
VERIFICATION OF QFT CIRCUITS WITHOUT LOCAL REDUCTION.

n	Same		Diff. 1		Diff. 2		Diff. 3		Diff. 4	
	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD
4	0	0.15	0	0.15	0	0.16	0	0.14	0	0.14
8	0	1.75	0	1.80	0	1.97	0	1.74	0	1.83
16	0.01	> 1,000	0.01	> 1,000	0	> 1,000	0.01	> 1,000	0	> 1,000
32	0.05	> 1,000	0.04	> 1,000	0.04	> 1,000	0.04	> 1,000	0.04	> 1,000
64	2.35	> 1,000	2.24	> 1,000	2.42	> 1,000	2.33	> 1,000	2.48	> 1,000

Figure 49. Runtime results for equivalence-checking of Quantum Fourier Transforms.

TABLE V

VERIFICATION OF MODULAR MULTIPLICATION WITH LOCAL REDUCTION.

n	Same		Diff. 1		Diff. 2		Diff. 3		Diff. 4	
	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD
4	0.58	0	0.98	0.04	1.07	0.85	0.98	0.05	1.02	0.39
8	2.13	0	3.72	0.04	3.69	0.37	3.73	0.04	3.45	1.19
16	6.03	0	10.11	0.05	11.29	$>$ 1,000	11.16	0.05	11.26	5.73
32	16.33	0	27.65	0.04	27.49	3.68	27.21	0.05	27.83	0.04
64	36.28	0	58.32	0.02	59.27	0.56	60.91	0.05	60.13	1.33
128	74.77	0	119.71	0.04	120.98	1.88	120.83	0.05	121.59	52.55

TABLE VI

VERIFICATION OF MODULAR MULTIPLICATION W/O LOCAL REDUCTION.

n	Same		Diff. 1		Diff. 2		Diff. 3		Diff. 4	
	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD	simp.	QuIDD
4	0.09	$>$ 1,000	0.09	$>$ 1,000	0.1	$>$ 1,000	0.09	$>$ 1,000	0.09	$>$ 1,000
8	0.23	$>$ 1,000	0.23	$>$ 1,000	0.23	$>$ 1,000	0.24	$>$ 1,000	0.24	$>$ 1,000

Figure 50. Runtime results for equivalence-checking of modular multiplication.

For a more convincing example, we check equivalence between an LNN and non-LNN implementation (without *measurement gates*) of Shor's algorithm for factoring the number 15. These equivalent properly-quantum circuits include 2,732 gates for the non-LNN version and 5,120 gates for the LNN version. Their structure is very different. For equivalence-checking, we used QuIDDPro with and without local reduction, and these runs completed in 59.07s and 64095.22s, respectively. The results unequivocally suggest the effectiveness of local reductions with reversible miters of properly-quantum circuits.

2) Proposed Method: Boosting Verification by Using SAT-based Combinational Tools: Local reduction may leave many gates around, after which QuIDDPro tends to consume significant time and memory. However, if very few properly-quantum gates remain, a more lightweight verification procedure may be used. Generic symbolic simulators, such as QuIDDPro, do not scale as well as state-of-the-art SAT-based combinational equivalence-checking. Hence we leverage SAT-based tools to boost equivalence-checking of quantum circuits.

FOR TWO CIRCUITS C_1 AND C_2, WE DO THE FOLLOWING.

Step 1. Construct the miter circuit $C = C_1 \cdot C_2^{-1}$.

Step 2. Perform simplification of the miter circuit.

Step 3. If properly-quantum gates remain, go to Step 4, else invoke state-of-the-art SAT-based combinational equivalence-checking (the "cec" command of ABC system [136]) to tell if the miter circuit is Identity.

Step 4. Find the longest sequence of conventional logic gates (NOT, CNOT, Toffoli) in the miter circuit. Label this sequence Ca. Let the simplified miter circuit be $Q_a \cdot C_a \cdot Q_b$.

Step 5. Transform $Q_a \cdot C_a \cdot Q_b$ to $C_a \cdot Q_b \cdot Q_a$. Note that $Q_a \cdot C_a \cdot Q_b = I$ (Identity) iff $C_a \cdot Q_b \cdot Q_a = I$ as shown earlier. Move conventional gates in $Q_b \cdot Q_a$ to the front of the miter as much as possible, creating a transformed miter $C'_a \cdot Q'_b$, where C_a and Q are a reversible circuit and a properly-quantum circuit, respectively.

Step 6. Check the functionality of Q'_b by lightweight iterated simulation. If it is not properly quantum, conclude that the miter circuit is not Identity. Else, go to Step 7.

Step 7. Exploit the functionality of Q'_b, and let C_b be a conventional circuit which corresponds to the exploited logic functionality. Then, check whether $C'_a \cdot C_b$ is Identity or not.

Consider the properly-quantum circuit as shown in the left-hand side of this figure.

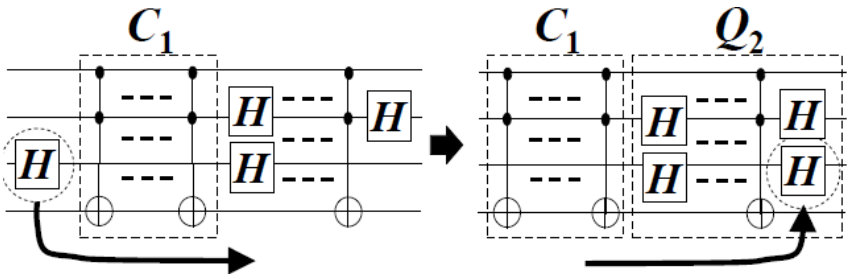


Figure 51. An example of circuit restructuring after simplification.

Here we assume that C_1 is relatively large. Then after Step 5, we can get the right-hand side circuit from the left-hand side circuit in Figure 51. Our miter becomes $C_1 \cdot Q_2$ where C_1 is reversible but Q_2 is properly-quantum. This avoids a heavy-duty generic quantum simulator for C_1. A key observation is that the functionality of Q'_b (at Step 6) should be classical (C'_a inverse) if the entire miter is Identity. Thus, if Q'_b is properly-quantum, the miter circuit is not Identity. When Q'_b has few gates, this can be checked efficiently by a quantum generic simulator. By Step 7, properly-quantum gates are reduced, and we can use state-of-the-art SAT-based combinational equivalence-checking. By avoiding heavy-duty generic quantum simulation, our adaptive method can achieve significant speed-ups when C_a' is large. To validate our method, we studied circuits implementing one iteration of Grover's quantum algorithm for search [147] as shown in Figure 51. A particular step of the algorithm, called *the oracle*, is implemented with a reversible circuit module C_f based on a user-defined Boolean function f (search predicate). To make verification more challenging, we configured a search predicate that contains a multiplier circuit. We then created an equivalent variant of C_f by applying a global, rather than local, circuit transform. Namely, we applied a certain wire permutation on inputs of C_f and its inverse on outputs of C_0. This permutation was implemented by applying SWAP gates to (all) pairs of adjacent wires and then breaking down each SWAP gate into three CNOT gates. In our case study, the proposed procedure goes as follows.

Step 1. Construct the miter circuit $C = C_1 \cdot C_2^{-1} = C_f^{1} \cdot W^{1} \cdot C_1^{0} \cdot W^{1} \cdot (W^{2})^{-1} \cdot (C_0^{2})^{-1} \cdot (W^{2})^{-1} \cdot (C_f^{2})^{-1}$.

Step 2. Simplify the miter circuit. Because of the inserted SWAP gates (if we use only naive cancellation rules), we cannot cancel the two pairs of C_f^1 and $(C_f^2)^{-1}$, or C_0^1 and $(C_0^2)^{-1}$. But we can remove the sequence $W^1 \cdot (W^2)^{-1}$, reducing the miter to $C_f^1 \cdot W^1 \cdot C_0^1 \cdot (C_0^2)^{-1} \cdot (W^2)^{-1} \cdot (C_f^2)^{-1}$.

Step 3. Since properly-quantum gates remain, go to Step 4.

Steps 4 and 5. Move $(C_f^2)^{-1}$ to the input side of the circuit to maximize the conventional logic part in the prefix. The miter becomes $C_a' \cdot Q_b'$ where $C'_a = (C_f^2)^{-1} \cdot C_f^1$ and $(Qb)' = W^1 \cdot C_0^1 \cdot (C_0^2)^{-1} \cdot (W^2)^{-1}$.

Steps 6. and 7. Using techniques described earlier, combine a quantum generic simulator (QuIDDPro [149, 150]) and state-of-the-art SAT-based combinational equivalence-checking (the "cec" command of ABC system [136]).

The above technique is compared to constructing a miter circuit and applying the symbolic simulator QuIDDPro [149, 150] to the miter. QuIDDPro alone does not finish in ten hours, but our technique completes in under seven seconds.

5.4. Summary

We have studied several techniques for equivalence-checking of reversible circuits, including the new concept of reversible miters. In particular, we have observed that state-of-the-art SAT-based combinational equivalence-checking (cec) can be adapted to this context and outperforms generic quantum techniques. Basic BDD-based techniques usually outperform SAT-based techniques, but not cec. As is the case with Automatic Test-pattern Generation, reversibility can significantly simplify equivalence-checking, while these simplifications are compatible with other techniques and amplify them. We then proposed an adaptive method to verify quantum circuits more efficiently than the existing quantum circuit verification tools by combining them with the state-of-the-art SAT-based combinational equivalence-checking tool for the conventional circuits. Experiments suggest that reversible miters are useful for the verification of reversible circuits as well as properly-quantum circuits.

6. Ising Systems and Quantum Adiabatic Computation

With the prospect of atomic-scale computing, we are studying cumulative energy profiles of spin-spin interactions in non-ferromagnetic lattices (spin-glasses)—an established topic in statistical and solid-state physics. Recent proposals suggest using Ising spin-glasses for non-traditional computing as a way to harness nature's ability to find min-energy states, and to take advantage of quantum tunneling to boost combinatorial optimization. We therefore developed EDA-inspired high-performance algorithms to simulate natural energy minimization in Ising systems. Unlike previous work, our algorithms are not limited to planar Ising topologies. In one CPU-day, our branch-and-bound algorithm finds min-energy (ground) states on 100 spins, while our local search approximates ground states on 1,000,000 spins. We use this computational tool to study the significance of hyper-couplings in recently implemented adiabatic quantum computers.

6.1. Introduction

As leading CMOS foundries are gearing for mass production of 22nm and 32nm CMOS chips, long-term EDA research has started to explore the use of atomic properties in computing. This exploration relies on established computational models of atomic-scale phenomena, but struggles to connect different levels of abstraction—spin-level models and energy-based statistical macro models. The spin-glass model was proposed by Edwards and Anderson [160] as a variation of the Ising model to study disorder and frustration in crystallized solids. Such systems are composed of particles that can be in either of two possible energy spin states. These spins interact in pairs to produce an energy landscape that describes the overall behavior of the system. The model is described in graph-theoretic terms by representing atoms in a crystal with vertices and bonds between atoms with edges. Since physical systems found in nature are often disordered, the bond edges are assigned random weight values. Furthermore, some physical and chemical properties of a crystal depend on the total energy of the bonds, which depend on atomic states. Thus, estimating total energy via a graph-based function facilitates the use of graph algorithms to study properties of solids. In particular, we are interested in finding the spin configuration that produces the least amount of energy. This configuration is known as the ground state of the system. Barahona [155] proved that for general Ising spin systems (spin-glasses), the ground-state determination problem (GSD) is NP-hard.

Since many physical systems have a natural ability to find least-energy states quickly, researchers are currently attempting to exploit this phenomenon to perform useful computation. At the atomic scale, energy minimization can be aided by quantum tunneling, which effectively reduces the number of local minima. Thus, GSD problems are of particular interest to quantum-information researchers because they are suitable candidates for evaluating the performance of adiabatic quantum computers (AQCs).

Recently developed AQCs employ an architecture based on Ising spin systems [167]. First, the spin system is configured to represent a given combinatorial problem, i.e., the spin interactions are carefully controlled rather than random as in spin glasses. The ground state is approximated via quantum annealing (the quantum analogue of thermal annealing), then read off as a bit sequence and interpreted as an answer to the problem. Since the complexity of GSD is universal with respect to both quantum and conventional forms of computation [155, 164], it is important to

consider how well an approximation to the ground-state energy can be obtained by purely classical techniques. Consequently, Bansal et al. [154] proposed an approximation algorithm for GSD on Ising spin lattices, which essentially simulates these AQC architectures [162], and thus limits their potential for quantum speed-ups. To approximate the least energy with ε accuracy, the algorithm from [154] requires runtime exponential in 1/ε, which is impractical. In contrast, we propose a branch-and-bound algorithm and a high-performance local search that quickly finds near-optimal energy values for arbitrary Ising topologies. Such techniques can be used to critically assess the performance of non-traditional computing devices based on energy minimization in Ising spin-glasses and also to determine the best implementation options.

6.2. Previous Work

In the Edwards-Anderson [160] model, spins are binary ±1 values, and the strengths of the atomic couplings are independent and identically distributed random variables according to some probability distribution. The most common distributions used are the Gaussian, and the ±1-bimodal distributions. Let $G_{ising} = (V,E)$ denote an Ising-model graph with n vertices (spins). Each vertex $u \in V$ is denoted by spin value $S_u \in \{\pm 1\}$ and is assigned a magnetization weight h_u. For $u,v \in V$, we define $(u,v) \in E$ to be an edge representing a bond between two adjacent spins with assigned weight $J_{u,v}$ chosen randomly from the standard Gaussian distribution ($\mu = 0$, $\delta = 1$). Thus, we expect that half of the bonds in the graph will be negative. Usually, the same positive-to-negative bond ratio is maintained when the ±1-bimodal distribution is used instead. The internal energy of the system for a particular configuration of spin values $\sigma = \{S_i\}$ is given by $E(\sigma) = -\Sigma_{i,j} J_{i,j} S_i S_j - \Sigma_i h_i S_i$ where the summation considers all pairs of adjacent spins.

Putting together the energies of all spin configurations gives the Hamiltonian of the system. Thus, the ground state is given by $E_{gs} = min(E(\sigma) \mid \sigma \in \pi_n)$, where π_n is the set of all possible n-spin configurations. Whether we are interested in the lowest-energy value or the n-spin configuration with such energy, $|\pi_n| = 2^n$ because each of the spins can take on one of two possible values. Energy minimization is typically NP-hard. Thus, calculating the ground state exactly using an exhaustive search algorithm is feasible only for small Ising models. To provide a scalable way of finding ground states or approximating their energies, heuristics are required, such as those developed in our work.

Ising-model graphs are not limited to a particular topology, but two- and three-dimensional lattices are most commonly considered in the literature. To simulate the behavior of infinite spin glasses, it is common to require that the spins lying on the dimensional boundary be connected to the spins on the opposing boundary on the same dimension. This can be viewed as a type of (periodic) boundary condition. In particular, only one periodic boundary condition is imposed for each dimension of the lattice. However, it is sometimes desirable to have boundary conditions on some but not all of the lattice dimensions.

Although most variations of GSD are known to be NP-hard [155], there are a few cases where the structure of the graph can be exploited to solve the problem in polynomial time. For example, Bieche et al. [158] proved that the GSD problem on planar graphs with zero magnetization can be solved in polynomial time by showing a reduction to the minimum-weight perfect matching (MWPM) problem. It follows from their work that GSD instances with zero magnetization ($h_i = 0$) and 0- or 1-periodic boundary conditions can be solved in $O(n^3)$ time. However, although MWPM is solvable in polynomial time, the runtime is impractical for large instances and suffers from a big memory footprint. To overcome these limitations, the work in [165] describes a heuristic based on the MWPM reduction. Another special case considered in the literature is that of ferromagnetic ($J_{i,j} > 0$) GSD instances. Barahona [156] reduced this particular problem to (s-t)-min-cut or max-flow.

6.3. Finding Exact Ground States

In order to better control trade-offs between runtime and solution quality obtained from heuristics, it is important to design algorithms that are guaranteed to find exact ground states. Although such algorithms are unlikely to scale to large instances, the exact solution obtained from smaller instances can be used to debug and evaluate the performance of more scalable heuristics. Branch-and-bound (B&B) algorithms consider incomplete or partial solutions, where only some variables are assigned values. Partial solutions are systematically constructed via a branching process that develops partial solutions that are deemed promising, i.e., those that may lead to the optimal solution. Partial solutions, whose cost is too high, are "bounded away" or pruned.

Our branching process proceeds as follows. First, all spins are labeled as unassigned–their value can be set in the future to either 1 or −1. The algorithm then calculates a lower bound E_{lb} for minimum energy. It then selects a spin I and branches on one of the possible values. In each branch, the incremental change in E_{lb} caused by the assignment is recorded as follows. For each spin j adjacent to i that has already been assigned, increase (decrease) E_{lb} by twice the amount of the positive (negative) bond connecting i and j if they have opposing (aligned) spin values. Similarly, the corresponding change due to the magnetization of the spin is also recorded (formulas for these updates are available in our technical papers and source code).

Once the spin is assigned, the algorithm branches to another spin and repeats the same procedure. When all spins have been assigned, E_{lb} represents the energy of the spin configuration generated by the branching process. To continue searching the configuration space, the branching process backtracks to the last assigned spin, flips its value and updates E_{lb}. If both spin values have already been tried, then the algorithm continues backtracking while relabeling the spins as unassigned. Since each spin can take one of two values, this branching process generates a full binary search tree where the leaves correspond to all possible spin configurations in the Ising system. Initially, we use a linear-time greedy approximation E as our bounding value. During the branching process, if the energy of the partial solution exceeds E, then we can safely prune this branch and backtrack without making any further assignments. The algorithm either tries the opposite spin value or backtracks again if both spin values have already been tried. If the search assigns all the spins in the graph and the corresponding minimal energy state is lower than E, then we set E to this new energy value. After searching all promising branches, E will assume the ground-state energy. In our experiments, this standard bounding technique alone improved the scalability of the branching process by an order of magnitude over exhaustive search.

To further improve the scalability of our B&B algorithm, we designed a prune-by-dominance technique that consists of identifying partial solutions whose partial energy can be improved (lowered) by modifying the configuration of currently assigned spins.

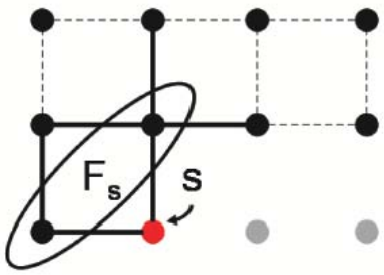


Figure 52. Pruning by dominance in our branch and bound algorithm.

Note that, whenever we assign a spin *s*, there is a set F_s of spins adjacent to *s* for which all neighboring spins have also been assigned. Figure 52 shows an example of s and F_s on a small grid. Note that $|F_s| \leq \text{degree}(s)$. The energy of the spins in F_s is localized in the sense that it will not be affected by further spin assignments (since all the neighbors of the spins in F_s are assigned). This allows us to evaluate a partial solution by flipping the values of the spins in F_s and comparing the partial energies. If any of the partial energies are lower, then we know that the current partial solution is unpromising and we can backtrack. Observe that in cases when different branches are statistically unlikely to have equal partial cost (e.g., when couplings and magnetizations are random), for two branches, the probability that the first branch dominates the second branch is approximately *1/2*. Let $0 < c \leq 1$ be the fraction of 2^k partial solutions that require branching. Then we can expect to prune $c2^k/2 = c2^{k-1}$ of these branches. This pruning technique improved the scalability of our B&B algorithm by 1-2 orders of magnitude on a variety of Ising benchmarks.

6.4. GSD through Local Search

Due to the difficulty of solving GSD, researchers have developed heuristic methods typically based on slow Monte Carlo simulations. However, because of the role that Ising models play in simulating real-world phenomena, it is desirable to have much faster techniques.

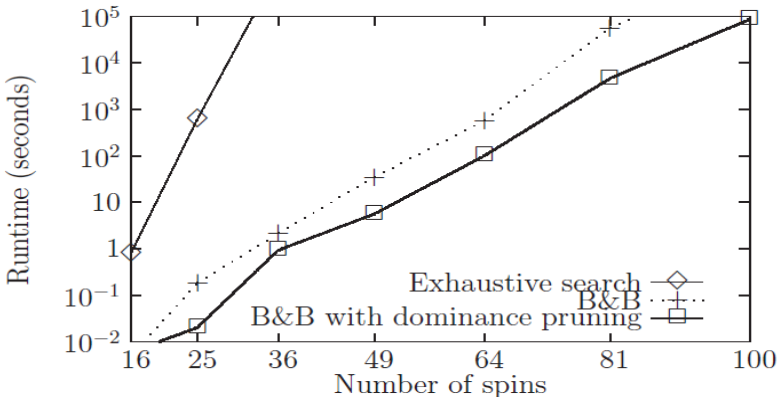


Figure 53. Runtime comparison of optimal Ising solvers.

Our local search is an iterative improvement algorithm that modifies the bipartition induced by an arbitrary spin configuration–positive spins are placed in one partition and negative spins in the other. Figure 53 compares the runtime of our approach with other optimal Ising solvers. The algorithm performs a sequence of incremental changes to the bipartition, organized as passes. These changes consist of spin-moves that remove a particular spin from its current partition and places it in the opposite one. At the beginning of each pass, the energy differential (gain) of performing each possible move is calculated. A positive gain implies that the move decreases the overall energy while a negative gain increases it. During a pass, the move that produces the largest gain is selected and executed. The corresponding spin is then labeled as locked, i.e., it cannot be selected again in the current pass. The pass continues selecting and executing the best moves until all spins have been locked. At the end of the pass, we save the best-seen bipartition produced by the sequence of moves. This bipartition is then used as the starting solution of the next pass. The entire algorithm terminates when a pass fails to obtain an improvement. Note that, in the absence of positive-gain moves, a negative-gain move can be selected. Thus, a pass may accept a solution that is worse than the existing solution (hill-climbing). This helps to reduce the probability of getting trapped in local minima. The initial random solution used in our local search is generated in linear time.

To increase the quality of solutions and the probability of finding the exact ground state, we repeat the algorithm using multiple random initial solutions and selecting the best result. The relationship between the number of random starts and solution quality is explored in our experiments. Each move causes a change in the local energy surrounding the selected spin, therefore, the gains of the neighboring spins need to be updated after each move. These gain updates are performed efficiently using a custom heap-based data structure. The data structure consists of two arrays. The first array implements a traditional binary heap while the second array allows quick access to the heap-array element that contains the gain-update value of a particular spin. To perform gain updates, we can access the specific value in $O(1)$ time, update the value, and perform the necessary swaps to maintain heap-order. Since only $\log(n)$ swaps are required in the worst case (where n is the number of spins), our data structure allows us to perform gain updates much faster than naive implementations that require scanning the entire set of n gain values. Since n moves are performed during a pass, and only a constant number of passes are performed, the runtime of our local search is $O(n \log(n))$.

6.5. Empirical Validation

We have tested single-threaded implementations of the algorithms proposed on a conventional Linux server. For small to medium-sized instances our local search finds exact ground states in 95% of independent random starts, otherwise solutions are 5% sub-optimal on average. We compared the average solution quality of local search for 2-dimensional Ising spin glasses with Gaussian-distributed couplings and two periodic boundary conditions. The benchmarks are differentiated by the number of spins included. For each instance we considered five different levels of effort with an increasing number of independent random starts (1, $\ln(n)$, 10, $\ln^2(n)$ and n) per instance per level of effort. To obtain the average solution quality we computed 1000 output samples per instance per case considered. The exact ground states were obtained via the University of Cologne's Ising Spin Glass Server [168] which uses a branch-and-cut algorithm to

find exact ground states on grid spin glasses with zero magnetization. As expected, the solution quality improves as the number of random starts increases. When we consider at least $\ln^2(n)$ random starts, our local search produces high-quality solutions (> 95%) for five of the benchmarks, while its runtime does not exceed 17 seconds for the largest benchmark with 2500 spins.

Note that the quality of the solutions decreases as we consider larger instances. This is expected since the energy landscape gets more complicated as the number of spins is increased and the probability of finding the exact ground state becomes exponentially small. However, this is true of all heuristics. We also experimented with spin-glasses that have ±1-bimodal coupling distributions. Compared to Gaussian-distributed instances, the expected solution quality observed is better. Our heuristic finds high-quality solutions for all but one of the benchmarks using a single random start.

Our heuristic scales to a million spins and empirical runtimes closely fit $n\log(n)$). We compared the runtime of our local search against that of the MWPM-based heuristic proposed in [165]. The solution quality of this heuristic depends on a particular choice of parameters and does not work on Gaussian-distributed instances. In contrast, our heuristic does not have such a dependency and works on all instances. Table IV in [165] shows the runtime and solution quality of the MWPM-based heuristic ±1 grid graphs of size 164×164. This heuristic takes 59*s* on average (with negligible deviation), producing the optimal value only 61% of the time. By comparison, our local search heuristic on a comparable benchmark with a single random start takes about 8.5*s*. Thus, we can perform 7 random starts in the same period of time.

7. Conclusion

This project investigated novel hybrid techniques, combining concepts from quantum information science and classical computer science to solve hard computational problems, including the handling of uncertainty, such as design optimization and simulation of conventional CMOS and quantum systems, fault tolerance, resource allocation and scheduling, strategy optimization and related computationally challenging problems facing the Air Force. The project accomplished:

- An analytical study of probabilistic fault models in digital logic and their impact on overall circuit performance. Probabilistic faults affect electronics in high-altitude and high-radiation environments, especially state-of-the-art deep-submicron CMOS chips.
- New algorithmic methodologies and tools for circuit synthesis and test to mitigate the impact of probabilistic faults. These methodologies include fast evaluation of circuit reliability based on functional simulation, and incremental modification of a circuit to improve its robustness.
- Analysis of mobile (wireless) ad hoc communication networks, focused on network throughput and total power consumption.
- A non-linear programming framework for spatial optimization of mobile networks, its empirical evaluation, and visualization of results.
- A new algorithm to simulate quantum circuits which exhibits polynomial-time performance in several important cases. This algorithm and accompanying theoretical results have been subsequently used by other researchers to show that the Quantum Fourier Transform can be simulated in polynomial time on conventional computers.
- Several techniques for verification of correctness of quantum circuits. These techniques are based on computational engines frequently used in Electronic Design Automation Boolean satisfiability (SAT) and binary decision diagram (BDD), fall under the category of equivalence-checking, and can verify the results of adapting known circuits to specific device architectures, such as linear ion traps.

8. References

1. Berkeley Logic Synthesis & Verification Group, *ABC*: *A System For Sequential Synthesis & Verification*, http://www.eecs.berkeley.edu/_alanmi/abc/
2. S. Almukhaizim et al., "Seamless Integration of SER in Rewiring-Based Design Space Exploration," *Proc. ITC*, 2006, pp. 1-9.
3. R.I. Bahar et al., "Algebraic Decision Diagrams and their Applications," *Journal of Formal Methods in System Design*, 1997, vol. 10, pp. 171-206.
4. D. Bhaduri, S. Shukla, "NANOLAB - A Tool for Evaluating Reliability of Defect-tolerant Nanoarchitectures,. *IEEE Trans. Nanotechnology*, 2005, vol. 4, pp. 381-394.
5. R.I. Bahar, J. Mundy, J. Chan, "A Probabilistic Based Design Methodology for Nanoscale Computation," *Proc. ICCAD*, 2003, pp. 480-486.
6. D. Brand, "Verification of Large Synthesized Designs," *Proc. ICCAD*, 1993, pp. 534-537.
7. M. Bushnell, V. Agrawal, *Essentials of Electronic Testing*, Kluwer, 2000.
8. M. Choudhury, K. Mohanram, "Accurate and Scalable Reliability Analysis of Logic Circuits," *Proc. DATE*, 2007, pp. 1454-1459.
9. J. P. Hayes, I. Polian, B. Becker, "An Analysis Framework for Transient-Error Tolerance," *Proc. VTS*, 2007, pp. 249-255.
10. J. Kleinberg, E. Tardos, *Algorithm Design*, Addison-Wesley, 2005, pp. 612-617.
11. H. Kobayashi et al., "Comparison between Neutron-induced System-SER and Accelerated-SER in SRAMs," *Proc. Intl Reliability Physics Symp.*, 2004, pp. 288-293.
12. S. Krishnaswamy, *Design, Analysis and Test of Logic Circuits under Uncertainty*, Ph.D. Dissertation, University of Michigan, Computer Science and Engineering Program, 2008.
13. S. Krishnaswamy, I. L. Markov, J. P. Hayes, "Testing Logic Circuits for Transient Faults," *Proc. ETS*, 2005, pp. 102-107.
14. S. Krishnaswamy, I. L. Markov, J. P. Hayes, "Tracking Uncertainty with Probabilistic Logic Circuit Testing," *IEEE Design & Test*, 2007, vol. 24, no. 4, pp. 312-321.
15. S. Krishnaswamy, I. L. Markov, J. P. Hayes, "On the Role of Timing Masking in Reliable Logic Circuit Design," *Proc. DAC*, 2008, pp. 924-929.
16. S. Krishnaswamy, I. L. Markov, J. P. Hayes, "When Are Multiple Gate Errors Significant in Logic Circuits?" *SELSE Workshop*, 2006.
17. S. Krishnaswamy, S. M. Plaza, I. L. Markov, J. P. Hayes, 'AnSER: A Lightweight Reliability Evaluator for use in Logic Synthesis," *Digest IWLS*, 2007, pp. 171-173.
18. S. Krishnaswamy, S. M. Plaza, I. L. Markov, J. P. Hayes, "Enhancing Design Robustness with Reliability-aware Resynthesis and Logic Simulation," *Proc. ICCAD*, 2007, pp. 149-154.
19. S. Krishnaswamy, S. M. Plaza, I. L. Markov, J. P. Hayes, "Signature-based SER Analysis and Design of Logic circuits,. *IEEE Trans. CAD*, vol. 28, Jan. 2009, pp.74–86.
20. S. Krishnaswamy, G. F. Viamontes, I. L. Markov, J. P. Hayes, "Accurate Reliability Evaluation and Enhancement via Probabilistic Transfer Matrices," *Proc. DATE*, 2005, pp. 282-287.
21. S. Krishnaswamy, G. F. Viamontes, I. L. Markov, J. P. Hayes, "Probabilistic Transfer Matrices in Symbolic Reliability Analysis of Logic Circuits," *ACM Trans. Design Automation of Electronic Systems*, 2008, vol. 13, no. 1, article 8.
22. H. K. Lee, D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," TR No. 12-93, EE Dept., Virginia Polytechnic Institute.
23. T.C. May, M.H. Woods, "Alpha-Particle-Induced Soft Errors in Dynamic Memories," *IEEE Trans. Electron Devices*, 1979, vol. 26, pp. 2-9.
24. Mishchenko, S. Chatterjee, R. Brayton, "DAG-aware AIG rewriting: A Fresh Look at Combinational Logic Synthesis," *Proc. DAC*, 2006, pp. 532-535.
25. N. Miskov-Zivanov, D. Marculescu, "Soft Error Rate Analysis for Sequential Circuits," *Proc. DATE*, 2007, pp. 1436-1441.

26. K. Mohanram, N. A. Touba, “Partial Error Masking to Reduce Soft Error Failure Rate in Logic Circuits,.” *Proc. DFT*, 2003, pp. 433-440.
27. K. Nepal, et al., “Designing Logic Circuits for Probabilistic Computation in the Presence of Noise,” *Proc. DAC*, 2005, pp. 485-490.
28. Openbayes software, http://www.openbayes.org/.
29. S. Plaza, K-H. Chang, I. Markov, V. Bertacco, “ Node Mergers in the Presence of Don't Cares,.” *Proc. ASP-DAC*, 2007, pp. 414-419.
30. Ramalingam, et al., “An Accurate Sparse Matrix Based Framework for Statistical Static Timing Analysis,” .*Proc. ICCAD*, 2006, pp. 231-236.
31. R. Rao et al., “An Efficient Static Algorithm for Computing the Soft Error Rates of Combinational Circuits,” *Proc. DATE*, 2006, pp. 164-169.
32. T. Rejimon, S. Bhanja, “Probabilistic Error Model for Unreliable Nano-logic Gates,.”*Proc. NANO*, 2006, pp. 47-50.
33. Sanyal, K. Ganeshpure, S. Kundu, “On Accelerating Soft-Error Detection by Targeted Pattern Generation,” *Proc. ISQED*, 2008, pp. 723-728.
34. P. Shivakumar et al., “Modeling **t**he Effect of Technology Trends on Soft Error Rate of Combinational Logic,” *Proc. ICDSN*, 2002, pp. 389-398.
35. G.F. Viamontes, I.L. Markov, J. P. Hayes, “Improving Gate-Level Simulation of Quantum Circuits,” *Quantum Information Processing*, 2003, vol. 2, no. 5, pp. 347-380.
36. J. Wilkinson, S. Hareland,. “A Cautionary Tale of Soft Errors Induced by SRAM Packaging Materials,” *IEEE Trans. Device & Materials Reliability*, 2005, vol. 5, pp. 448-433.
37. J. F. Ziegler et al., “IBM Experiments in Soft Fails in Computer Electronics (1978- 1994),” *IBM Journal of Research and Development*, 1996, vol. 40, No. 1.
38. M. Zhang, N. R. Shanbhag, “A Soft Error Rate Analysis (SERA) Methodology,” *Proc. ICCAD*, 2004, pp. 111-118.
39. B. Zhang, W. S. Wang, M. Orshansky, “FASER: Fast Analysis of Soft Error Susceptibility for Cell-Based Designs,” *Proc. ISQED*, 2006, pp. 755-760.
40. Q. Zhu et al. “SAT Sweeping with Local Observability Don't-cares,” *Proc. DAC*, 2006, pp. 229-234.
41. S. Agarwal et al. , “Route-lifetime assessment based routing (RABR) protocol for mobile ad-hoc networks,” *Proc. Int. Conf. on Communications* (ICC), vol. 3, 2000, pp. 1697–1701.
42. F. Bai, N. Sadagopan, & A. Helmy, “Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for ad hoc networks,” *Proc. Infocom*, vol. 2. 2003, pp. 825–835.
43. Balasubramanian et al., “Web search from a bus,” *Proc. Workshop on Challenged Networks* (CHANTS), 2007, pp. 59–66.
44. N. Banerjee, M. D. Corner and B. N. Levine, “An energy-efficient architecture for dtn throwboxes,” *Proc. Infocom*, 2007, pp. 776–784.
45. C. Bettstetter, “Smooth is better than sharp: a random mobility model for simulation of wireless networks,” *Proc. 4th. Workshop on Modeling, Analysis & Simulation of Wireless and Mobile System*s (MSWiM), 2001, pp. 19–27.
46. C. Bettstetter, G. Resta & P. Santi, “The node distribution of the random waypoint mobility model for wireless ad hoc networks,” *IEEE Trans. Mobile Computing*, vol. 2, pp. 257–269, Jul. 2003.
47. V. Bharghavan et al.“MACAW: A media access protocol for wireless LAN’s,” *Proc. SIGCOMM*, 1994, pp. 212–225.
48. D. M. Blough et al., “The K-neigh protocol for symmetric topology control in ad hoc networks,” *Proc. MobiHoc*, 2003, pp. 141–152.
49. J. Boleng, W. Navidi and T. Camp, “Metrics to enable adaptive protocols for mobile ad hoc networks,” *Proc. Int. Conference on Wireless Networks* (ICWN), Jun. 2002.
50. S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.
51. Cerpa et al., “Habitat monitoring: Application driver for wireless communications technology,” *Proc. SIGCOMM Workshop Data Communications in Latin America & the Caribbean*, Apr. 2001.

52. S. Cho, *Adaptive Management Schemes for Mobile Ad Hoc Networks*, Ph.D. Dissertation, University of Michigan, Computer Science and Engineering Program, Sep. 2009.
53. S. Cho & J. P. Hayes, "Impact of mobility on connection stability in ad hoc networks," *Proc. Wireless Communications & Networking Conf.* (WCNC), vol. 3, 2005, pp. 1650–1656.
54. S. Cho & J. P. Hayes, "Power-aware link maintenance (PALM) for mobile ad hoc networks," *Proc. Local Computer Networks*, 2007, pp. 403–410.
55. S. Cho & J. P. Hayes, "Optimizing router locations for minimum-energy wireless networks," *Proc. Local Computer Networks*, 2008, pp. 544–546.
56. D. Estrin et al., "Scalable coordination in sensor networks," *Proc. MobiCom*, 1999, pp. 263–270.
57. Gomez, A.T. et al., "Conserving transmission power in wireless ad hoc networks," *Proc. Int. Conf. Network Protocols* (ICNP), 2001, pp. 24–34.
58. Gomez, A.T. et al., 'PARO: Supporting dynamic power controlled routing in wireless ad hoc networks," *Jour. Wireless Networks*, vol. 9, Sep. 2000.
59. Y. Han, R.J. La & A.M. Makowski, "Distribution of path durations in mobile ad-hoc networks - Palm's theorem at work," *Proc. 16th ITC Specialist Seminar*, Aug. 2004.
60. T. Hastie, R. Tibshirani & J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 3rd ed. Springer, 2003.
61. D. Johnson & D. Maltz, "Dynamic source routing in ad hoc wireless networks," *Proc. Mobile Computing*, vol. 353, 1996, pp. 153–181.
62. E.-S. Jung & N. H. Vaidya, "A power control MAC protocol for ad hoc networks," *Proc. MobiCom*, Sep. 2002, pp. 36–47.
63. M. Krunz, A. Muqattash, & S.-J. Lee, "Transmission power control in wireless ad hoc networks: Challenges, solutions, and open issues," *IEEE Network*, vol. 18, Sep. 2004 , pp. 8–14.
64. J. F. Kurose and K. W. Ross, *Computer Networking*, 3rd ed. Addison Wesley, 2005.
65. B.-J. Kwak, N.-O. Song & L.E. Miller, "A mobility measure for mobile ad hoc networks," *IEEE Communications Letters*, vol. 7, Aug. 2003, pp. 379–381.
66. L. Li et al., "Analysis of a cone-based distributed topology control algorithm for wireless multi-hop networks," *Proc. Symp. Principles of Distributed Computing* (PODC), 2001, pp. 264–273.
67. N. Li & J. C. Hou, "FLSS: a fault-tolerant topology control algorithm for wireless networks," *Proc. MobiCom*, 2004, pp. 275–286.
68. N. Li, J. C. Hou & L. Sha, "Design and analysis of an MST-based topology control algorithm," *Proc. Infocom*, vol. 3, Mar. 2003, pp. 1702–1712.
69. W. Li & C. G. Cassandras, "A minimum-power wireless sensor network self-deployment scheme," *Proc. Wireless Commun. & Networking Conf.* (WCNC), vol. 3, Mar. 2005, pp. 1897–1902.
70. J. B. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. Berkeley Symp. Mathematical Statistics and Probability*, vol. 1, 1967, pp. 281–297.
71. Mainwaring et al., "Wireless sensor networks for habitat monitoring," *Proc. Int. Workshop Wireless Sensor Networks and Applications* (WSNA), Sep. 2002.
72. P. C. Nar & E. Cayirci, "PCSMAC: A power controlled sensor-MAC protocol for wireless sensor networks," *Proc. European Workshop on Wireless Sensor Networks*, Jan. 2005, pp. 81–92.
73. Nayebi, A. Khosravi & H. Sarbazi-Azad, "The impact of stationary nodes on the performance of wireless mobile networks," *Proc. Int. Conf. Wireless and Mobile Commun.* (ICWMC), 2007.
74. R. Ostrovsky et al., "The effectiveness of Lloyd-type methods for the *k*-means problem," *Proc. Symp. Foundations of Computer Science* (FOCS), 2006, pp. 165–176.
75. X. Perez-Costa, C. Bettstetter & H. Hartenstein, "Toward a mobility metric for comparable & reproducible results in ad hoc networks research," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 7, no. 4, pp. 58–60, 2003.
76. E. Perkins & P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *Computer Commun. Reviews*, Oct. 1994, pp. 234–244.
77. E. Perkins & E. M. Royer, "Ad hoc on-demand distance vector routing," *2nd IEEE Workshop on Mobile Computing Systems & Applications*, 1999, pp. 90–100.

78. E. Perkins, *Ad Hoc Networking*. Addison Wesley, 2001.
79. R. Ramanathan & R. Rosales-Hain, "Topology control of multihop wireless networks using transmit power adjustment," *Proc. Infocom*, Mar. 2000, pp. 404–413.
80. K. Sayood, *Introduction to Data Compression*, 3rd ed. Morgan Kaufman, 2005.
81. K.¬P. Shih & Y.¬D. Chen, "CAPC: A collision avoidance power control MAC protocol for wireless ad hoc networks," I*EEE Commun. Letters*, vol. 9, pp. 859–861, Sep. 2005.
82. D. S. Tan et al., "Design and evaluation of an individually simulated mobility model in wireless ad hoc networks," *Proc. Commun. Networks & Distributed Systems Modeling & Simulation*, 2002.
83. Univ. of California, Berkeley. 29 Palms fixed/mobile experiment: Tracking vehicles with a UAV-delivered sensor network. http://www.eecs.berkeley.edu/˜pister/29palms0103, 2001.
84. Univ. of California, Berkeley, TinyOS Community Forum. http://www.tinyos.net.
85. USC Information Sciences Inst. The network simulator *ns*-2. http://www.isi.edu/nsnam/ns/.
86. Y. Wang & Y. J. Zhao, "Fundamental issues in systematic design of airborne networks for aviation," *Proc. Aerospace Conf.*, 2006, pp. 1–8.
87. R. Wattenhofer, L. Li, P. Bahl & Y.-M. Wang, "Distributed topology control for power efficient operation in multihop wireless networks," *Proc. Infocom*, vol. 3, Apr. 2001, pp. 1388–1397.
88. K. Xu, M. Gerla & S. Bae, "How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks?" *Proc. Global Telecommun. Conf.* (GLOBECOM), vol. 1, Nov. 2002, pp. 72–76.
89. K. Xu, M. Gerla & S. Bae, "Effectiveness *of RTS/CTS handshake in IEEE 802.11 based ad hoc networks,"* Ad Hoc Network Journal, vol. 1, pp. 107–123, Jul. 2003.
90. C. Yu, K. G. Shin & L. Song, "Link-layer salvaging for making routing progress in mobile ad hoc networks," *Proc. MobiHoc*, 2005, pp. 242–254.
91. D. Yu & H. Li, "Influence of mobility models on node distribution in ad hoc networks," *Proc. Int. Conf. Communication Technology* (ICCT), vol. 2, Apr. 2003, pp. 985–989.
92. J. Zavgren, "NTDR mobility management protocols and procedures," *Proc. Military Commun. Conf.* (MILCOM), Nov. 1997.
93. L. Zhitang et al., "A novel MAC protocol for wireless ad hoc networks with power control," *Proc. Int. Conf. Multimedia & Ubiquitous Engineering* (MUE), Apr. 2007, pp. 347–352.
94. S. Aaronson and D. Gottesman, "Improved simulation of stabilizer circuits," *Physical Review A*, 70:052328, 2004.
95. D. Aharonov, A. Kitaev, and N. Nisan, "Quantum circuits with mixed states," In *Proceedings of the 31th Annual ACM Symposium on the Theory of Computation (STOC)*, pages 20–30, 1998.
96. D. Aharonov, Z. Landau, and J. Makowsky, "The quantum FFT can be classically simulated," Preprint: quant-ph/0611156.
97. J. Alber, H. L. Bodlaender, H. Fernau, T. Kloks, and R. Niedermeier, "Fixed parameter algorithms for dominating set and related problems on planar graphs," *Algorithmica*, 33(4):461–493, 2002.
98. S. Arnborg, "Efficient algorithms for combinatorial problems on graphs with bounded decomposability - a survey," *BIT*, 25(1):2–23, 1985.
99. S. Arnborg, D. G. Corneil, and A. Proskurowski, "Complexity of finding embeddings in a *k*-tree," *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.
100. H.J. Briegel and R. Raussendorf, "Persistent entanglement in arrays of interacting particles," *Physical Review Letters*, 86, 910–913, 2001.
101. H. L. Bodlaender, "Treewidth: characterizations, applications, and computations," Technical Report UU-CS-2006-041, Universiteit Utrecht.
102. S. Bravyi and R. Raussendorf, "On measurement-based quantum computation with the toric code states," Preprint: quant-ph/0610102.
103. E. Broering and S. Lokam, "Width-based algorithms for SAT and Circuit-SAT (extended abstract)," In *Sixth International Conference on Theory and Applications of Satisfiability Testing (SAT 2003), Springer-Verlag Lecture Notes in Computer Science (LNCS)*, volume 2919, pp. 162–171, 2004.
104. R. Dechter, "Bucket elimination: a unifying framework for reasoning," *Artificial Intelligence*, 113(1-2):41–85, 1999.

105. L.-M. Duan and R. Raussendorf, “Efficient quantum computation with probabilistic quantum gates,” *Physical Review Letters*, 95:080503, 2005.
106. D. Gottesman, “The Heisenberg representation of quantum computers,” In S. P. Corney, R. Delbourgo, and P. D. Jarvis, editors, *Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics*, pages 32–43, Cambridge, MA, 1999. International Press. Long version: quant-ph/9807006.
107. S. F. F. Green, S. Homer, and Y. Zhang, “Bounds on the power of constant-depth quantum circuits,” Preprint: quant-ph/0312209, 2004.
108. A.W. Joshi. *Matrices and tensors in physics*. Halsted Press [John Wiley & Sons], New York-London-Sydney, 1975.
109. R. Jozsa, “On the simulation of quantum circuits,” Preprint: quant-ph/0603163.
110. R. Jozsa and N. Linden, “On the role of entanglement in quantum computational speed-up,” *Proceedings of the Royal Society of London, Series A*, 459: 2011-2032, 2003.
111. S. Krishnaswamy, G. F. Viamontes, I. L. Markov and J. P. Hayes, “Accurate reliability evaluation and enhancement via probabilistic transfer matrices,” *Proc. Design Automation and Test in Europ(DATE)*, pp. 282-287, Munich, Germany, March 2005.
112. Lubotzky, R. Phillips, P., “Sarnak. Ramanujan graphs,” *Combinatorica*, 8(3):261–277, 1988.
113. L. Markov and Y. Shi, “Simulating quantum computation by contracting tensor networks,” Pre-print: quant-ph/0511069.
114. L. Markov and Y. Shi, “Constant degree graph expansions and treewidth,” Manuscript.
115. M. van den Nest, W. D¨ur, G Vidal,and H J. Briegel, “Classical simulation versus universality in measurement based quantum computation,” *Physical Review A*, 75:012337, 2007.
116. M. van den Nest, A. Miyake, W. D¨ur and H. J Briegel, “Universal resources for measurement-based quantum computation,” *Physical Review Letters*, 97:150504, 2006.
117. M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, England, 2000.
118. D. Porras, F. Verstraete and J. I. Cirac, “Renormalization algorithm for the calculation of spectra of interacting quantum systems,” Preprint: cond-mat/0504717.
119. R. Raussendorf and H. J. Briegel, “A one-way quantum computer,” *Physical Review Letters*, 86, 5188–5191, 2001.
120. N. Robertson and P. D. Seymour, “Graph minors. II. Algorithmic aspects of tree-width,” *Journal of Algorithms*, 7(3):309–322, 1986.
121. N. Robertson and P. D. Seymour, “Graph minors. III. Planar tree-width,” *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.
122. N. Robertson and P. D. Seymour, “Graph minors. X. Obstructions to tree-decomposition,” *Journal of Combinatorial Theory, Series B*, 52(2):153–190, 1991.
123. D. J. Rose, “Triangulated graphs and the elimination process,” *Journal of Mathematical Analysis and Applications*, 32:597–609, 1970.
124. V. P. Roychowdhury and F. Vatan, “Quantum formulas: a lower bound and simulation,” *SIAM Journal on Computing*, 31(2): 460–476, 2001.
125. B. M. Terhal and D. P. DiVincenzo, “Classical simulation of noninteracting-fermion quantum circuits,” *Physical Review A*, 65:32325–32334, 2002.
126. B. M. Terhal and D. P. DiVincenzo, “Adaptive quantum computation, constant depth quantum circuits and Arthur-Merlin games,” *Quantum Information and Computation*, 4(2):134–145, 2004.
127. L. G. Valiant, “Quantum circuits that can be simulated classically in polynomial time,” *SIAM Journal on Computing*, 31(4):1229–1254, Aug. 2002.
128. F. Verstraete and J. I. Cirac, “Renormalization algorithms for Quantum-Many Body Systems in two and higher dimensions,” Preprint: cond-mat/0407066.
129. F. Verstraete, J. J. Garcia-Ripoll and J. I. Cirac, “Matrix product density operators: simulation of finite-temperature and dissipative systems,” *Physical Review Letters*, 93:207204, 2004.

130. G. Vidal, “Efficient classical simulation of slightly entangled quantum computations,” *Physical Review Letters*, 91:147902, 2003.
131. G. Vidal, “Efficient simulation of one-dimensional quantum many-body systems,” *Physical Review Letters*, 93:040502, 2004.
132. P. Walther, K.J. Resch, T. Rudolph, E. Schenck, H. Weinfurter, V. Vedral, M. Aspelmeyer, and A. Zeilinger, “Experimental one-way quantum computing,” *Nature*, 434, 169–176, 2005.
133. N. Yoran and A. Short, “Efficient classical simulation of the approximate quantum Fourier transform,” *Physical Review A*, 76:042321, 2007.
134. Yao, “Quantum circuit complexity,” *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, 352–361, 1993.
135. M. Zwolak and G. Vidal, “Mixed-State dynamics in one-dimensional quantum lattice systems: a time-dependent super-operator renormalization algorithm,” *Physical Review Letters*, 93:207205, 2004.
136. The ABC Home Page. http://www.eecs.berkeley.edu/~alanmi/abc/.
137. The MiniSat Home Page. http://minisat.se/MiniSat.html.
138. G. Fowler, S. J. Devitt and L. C. L. Hollenberg, “Implementation of Shor’s algorithm on a linear nearest neighbour qubit array,” *Quantum Information and Computation*, 4(4):237–251, July 2004.
139. R. C. Johnson, “NIST scales up quantum computing,” *EE Times*, 08/06/09.
140. K. Prasad, V. V. Shende, K. N. Patel, I. L. Markov and J. P. Hayes, “Algorithms and data structures for simplifying reversible circuits,” *ACM J. of Emerging Technologies in Computing*, 2(4):277–293, Oct. 2006.
141. Mishchenko, S. Chatterjee, R. Brayton, N. Een, “Improvements to combinational equivalence checking,” *ICCAD*, pp. 836–843, 2006.
142. D. Maslov, G. W. Dueck, D. M. Miller and C. Negrevergne, “Quantum circuit simplification and level compaction,” *IEEE Trans. on CAD*, 27(3):436–444, Mar. 2008.
143. K. Iwama, Y. Kambayashi, and S. Yamashita, “Transformation Rules for Designing CNOT-based Quantum Circuits,” *DAC*, pp. 419–424, 2002.
144. G.-D. Lin et al., “Large-scale quantum computation in an anharmonic linear ion trap,” *EPL*, 86 (2009) 60004.
145. D. Maslov, “Linear-depth stabilizer and quantum Fourier transformation circuits with no auxiliary qubits in linear neighbor quantum architectures,” *Phys. Review A*, 76(052310), Nov. 2007.
146. D. M. Miller and M. A. Thornton, “QMDD: A decision diagram structure for reversible and quantum circuits,” *IEEE Int’l Symp. on Multiple-Valued Logic*, p.30, May 2006.
147. M. A. Nielsen and I. L. Chuang, “Quantum Computation and Quantum Information,” Cambridge University Press, 2000.
148. S. A. Cuccaro et al., “A new quantum ripple-carry addition circuit,” arXiv:quant-ph/0410184.
149. G. F. Viamontes, I. L. Markov, and J. P. Hayes, “Improving gate-level simulation of quantum circuits,” *Quantum Information Processing*, 2(5):347–380, 2003.
150. G. F. Viamontes, I. L. Markov, and J. P. Hayes, “Equivalence checking of quantum circuits and states,” *ICCAD*, pp. 69–74, 2007.
151. W. Hung et al, “Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis,” *IEEE Trans. on CAD*, 25(9):1652–1663, Sept. 2006.
152. J. Yoshida, “35 people, places and things that will shape the future,” *EE Times*, Feb 29, 2008.
153. Bakk and J. S. Hoye, “One–dimensional Ising model applied to protein folding,” *Physica A*, 2003, 323(15), pp. 504–518.
154. N. Bansal et al., “Classical approximation schemes for the ground–state energy of quantum and classical Ising spin Hamiltonians on planar graphs,” *Quant. Inf. Comp.*, 2009(7).
155. F. Barahona, “On the computational complexity of the Ising spin glass models,” *J. Phys. A*, 1982, pp. 3241–3253.
156. F. Barahona, “Ground–state magnetization of Ising spin glasses,” *Phys. Rev. B*, 1994, 49(18), pp. 12864–12867.

157. R. H. Baran, H. Ko, “An Ising model of transcription polarity in bacterial chromosomes,” *Physica A*, 2006, 362(2), pp. 403.
158. L. Bieche et al. “On the ground states of the frustration model of a spin glass by a matching method of graph theory,” *J. Phys. A: Math. Gen.*, 13 (1980) pp. 2576.
159. E. Caldwell et al., “Design & Implementation of the Fiduccia-Mattheyses heuristic for VLSI netlist partitioning,” *Lec. Notes Comp. Sci.* 1999,1619,pp.177-193.
160. S. F. Edwards, P. W. Anderson, *J. Phys. F*, 1975, pp. 965.
161. K. Hartmann, H. Rieger, “New optimization algorithms in physics”, *Wiley-vch*, 2004.
162. W. M. Kaminsky et al., “Scalable superconducting architecture for adiabatic quantum computation,” 2004, arXiv:quant-ph/0403090v2.
163. K. K. Muller–Nedebock, H. L. Frisch, “Dynamics of stiff polymers mapped from an Ising model,” *Polymer*, 2003, 44 (10), pp. 2829–2831.
164. R. Oliveira, B. M. Terhal, “The complexity of quantum spin systems on a two-dimensional square lattice,” *Quant. Inf. Comp.*, 8(10), pp. 900–924(2008).
165. G. Pardella, F. Liers, “Exact ground states of large two-dimensional planar Ising spin glasses,” *Phys. Rev. E 78*, 056705 (2008).
166. K. Kim et al,“Entanglement and tunable spin-spin couplings between trapped ions using multiple transverse modes,” 2009, arXiv:quant-ph/0905.0225v1.
167. X. Peng et al. “A quantum adiabatic algorithm for factorization and its experimental implementation,” *Phys. Rev. Lett.* 101, 220405 (2008).
168. http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/.

9. List of Publications

This is a cumulative list of publications prepared under the project.

1. J.P. Hayes, I. Polian and B. Becker: "An analysis framework for transient error tolerance." *Proc. VLSI Test Symp.*, Berkeley, CA, pp.249–255, May 2007.
2. I. Markov and Y. Shi: "Simulating quantum computation by contracting tensor networks." Accepted for publication by the *SIAM Journal of Computing*. A version of this paper is also available at arXiv:quant-ph/0511069.
3. S. Cho and J.P. Hayes: "Power-aware link maintenance (PALM) for mobile ad hoc networks." *Proc. IEEE Conf. Local Computer Networks* (LCN), Dublin, pp.403–410, Oct. 2007.
4. R. Das and J.P. Hayes: "Monitoring transient errors in sequential circuits." *Proc.16th Asian Test Symp*., Beijing, pp.319–322, Oct. 2007.
5. I. Polian, J.P. Hayes and B. Becker: "Cost-efficient selection of gates for circuit hardening based on critical soft error rate." *Digest of Papers, IEEE Workshop on RTL and High Level Testin*g (WRTLT-07), Beijing, Oct., 2007.
6. G.F. Viamontes, I.L. Markov and J.P. Hayes: "Checking equivalence of quantum states and operators." *Proc. Intl. Conf. on Computer-Aided Design* (ICCAD-07), San Jose, pp.69–74, Nov. 2007. A version of this paper is also available at arXiv:0705.0017.
7. S. Krishnawamy, S. Plaza, I.L. Markov and J.P. Hayes: "Enhancing design robustness with reliability-aware resynthesis and logic simulation." *Proc. Intl. Conf. on Computer-Aided Design* (ICCAD-07), San Jose, pp.149–154, Nov. 2007.
8. S. Krishnaswamy, G. F. Viamontes, I.L. Markov and J. P. Hayes: "Probabilistic transfer matrices in symbolic reliability analysis of logic circuits." *ACM Trans. on Design Automation of Electronic Systems*, vol. 13, article 8, Jan. 2008. This is a journal version of a paper presented at the Design and Test in Europe Conf. (DATE-05) in June 2005.
9. S. Krishnawamy, I.L. Markov and J.P. Hayes: "On the role of timing masking in reliable logic circuit design." *Proc. Design Automation Conference* (DAC-08), San Diego, pp.924–929, June 2008.
10. S. Krishnaswamy, S.M. Plaza, I.L. Markov and J.P. Hayes: "Signature-based SER analysis and design of logic circuits." *IEEE Trans. on Computer-Aided Design*, vol. 28, pp.74–86, Jan. 2009.
11. S. Cho and J.P. Hayes: "Optimizing router locations for minimum-energy wireless networks." *Proc. IEEE Conf. Local Computer Networks* (LCN), Montreal, pp.544–546, Oct. 2008.
12. S. Yamashita and I.L. Markov: "Equivalence-checking for reversible circuits." Unpublished report.
13. V.V. Shende and I.L. Markov: "On the CNOT-cost of TOFFOLI gates." *Quantum Information and Computation*, vol. 9, no. 5-6, pp. 461-486, 2009. A version of this paper is also available at arXiv:quant-ph/08032316.
14. S. Krishnaswamy: *Design, Analysis and Test of Logic Circuits under Uncertainty*, Ph.D. Dissertation, University of Michigan, Computer Science and Engineering Program, Sept. 2008.
15. S. Krishnaswamy, S.M. Plaza, I.L. Markov and J.P. Hayes: "Improving testability and soft-error resilience through retiming." *Design Automation Conference* (DAC-09), San Francisco, pp. 508–513, July 2009.
16. S. Cho: *Adaptive Management Schemes for Mobile Ad Hoc Networks*, Ph.D. Dissertation, University of Michigan, Computer Science and Engineering Program, Sept. 2009.
17. G.F. Viamontes, I.L. Markov and J.P. Hayes: *Quantum Circuit Simulation*, Springer, Boston and Dordrecht, 2009. In press.
18. H. Garcia and I. L. Markov: "High-performance Algorithms for Energy Minimization in Ising Spin-glasses." *Digest of Papers, International Workshop on Logic Synthesis* (IWLS), Berkeley 2009.

10. List of Acronyms

ACK – Acknowledge

ADDs – Algebraic Decision Diagrams

AI – Artificial Intelligence

AnSER – Analysis of Soft-Error Rate

AODV – Ad Hoc On Demand

AQCs – Adiabatic Quantum Computers

ATPG – Automatic Test Pattern Generation

B&B – Branch-and-Bound

BDD – Binary Decision Diagram

BIST – Built-in Self-Test

BRDM – Boundless Random Direction Model

BS – Base Station

BSP – Base Station Placement

CBR – Constant Bit Rate

CDF – Cumulative Distribution Function

CNT – Carbon Nanotube

CS – Carrier Sense

CSMA – Carrier Sense Multiple Access

CTS – Clear to Send

CV – Constant Velocity

DD – Decision Diagrams

DRP – Distributed Relay Placement

DSDV – Destination Sequenced Distance Vectoring

DSR – Dynamic Source Routing

ELW – Error Latching Window

FASER – Fast Analysis of Soft-Error

GPS – Global Positioning System

GSD – Ground-State Determination

ICs – Integrated Circuits

ILP – Integer Linear Programming

ITM – Ideal Transfer Matrix

LCR – Link Change Rate

LD – Link Duration

LINT – Local Information No Topology

LILT – Local Information Link-State Topology

LNN – Linear Nearest Neighbor

LP – Linear Programming

MAC – Medium Access Control

MANETs – Mobile Ad-Hoc Networks

MRFs – Markov Random Fields

MWPM – Minimum Weight Perfect Matching

NTDR – Near Term Digital Radio

ODCs – Observability Don't Cares

PALM – Power Aware Link Maintenance

PARO – Power Aware Routing Optimization

PCM – Power Control MAC

PDF – Probability Density Function

PTMs – Probabilistic Transfer Matrices

QFT – Quantum Fourier Transform

QMDDs – Quantum Multiple Valued Decision Diagrams

QuIDD – Quantum Information Decision Diagram

RD – Residual Duration

ROBDDs – Reduced Ordered Binary Decision Diagrams

RSS – Received Signal Strength

RTS – Request to Send

RTR – Request to Redirect

RWP – Random Waypoint

RX -- Receive

SA – Stuck-At

SAT – Boolean Satisfiability

SCC – Strongly Connected Closed

SER – Soft-Error Rate

SERA – Soft-Error Rate Analysis

SERD – Soft-Error Rate Descriptor

SET – Single Event Transient

SEU – Single-Event Upset

SiDeR – Signature Based Design for Reliability

SINR – Signal to Interference Noise Ration

SIR – Signal to Interference Ratio

SPICE – Simulation Program with Integrated Circuit Emphasis

STA – Static Timing Analysis

TMR – Triple Modular Redundancy

TSA – Transient Stuck-At

TX – Transmit

UAV – Unmanned Aerial Vehicle