

UC Riverside

UC Riverside Electronic Theses and Dissertations

Title

Simulation for Reliability, Hardware Security, and Ising Computing in VLSI Chip Design

Permalink

<https://escholarship.org/uc/item/4pz0538r>

Author

Cook, Chase William

Publication Date

2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
RIVERSIDE

Simulation for Reliability, Hardware Security, and Ising Computing in VLSI Chip
Design

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Chase W. Cook

December 2019

Dissertation Committee:

Dr. Sheldon X.-D. Tan, Chairperson
Dr. Daniel Wong
Dr. Nael Abu-Ghazaleh

Copyright by
Chase W. Cook
2019

The Dissertation of Chase W. Cook is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

This dissertation represents the culmination of work that would not have been possible if not for the support, mentorship, and collaboration from numerous individuals.

Firstly, I'd like to thank my PhD adviser, Dr. Sheldon Tan for making me the engineer and scholar that I am today and for giving me the opportunity to pursue this rigorous degree program in the first place. His years of experience and his immense wealth of technical and theoretical knowledge have been invaluable to my work and career.

I'd also like to thank my committee members, Dr. Daniel Wong and Dr. Nael Abu-Ghazaleh, both of whom inspired me during my early coursework and helped shaped my work that would follow.

I would like to also thank the members of the VLSI Systems and Computation Lab. Specifically, I'd like to thank Taeyoung Kim, Zeyu Sun, Hengyang Zhao, Shaoyi Peng, Han Zhou, Sheriff Sadiqbatcha, and Wentian Jin all for their collaboration and advice which has led to this work. Their fellowship has been invaluable and is greatly appreciated.

Additionally, I would have never pursued this degree if it were not for the following professors who pushed me to be more than I thought I could be: Dr. Hani Mehrpouyan, Dr. Alice Parker, and Dr. Shahnam Mirzaei.

Lastly, I want to thank my family, especially my wife Christina, my daughter Olivia, my parents Greg and Joleen, and my brother Clayton for their constant love and support.

To my parents Greg and Joleen, for their unwavering love and support.

ABSTRACT OF THE DISSERTATION

Simulation for Reliability, Hardware Security, and Ising Computing in VLSI Chip Design

by

Chase W. Cook

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2019
Dr. Sheldon X.-D. Tan, Chairperson

The continued scaling of VLSI circuits has provided a wealth of opportunities and challenges to the VLSI circuit design area. Both these challenges and opportunities, however, require new simulation tools that can enable their solution or exploitation as classical methods typically dealt with problem domains with smaller scales or less complexity. In this dissertation, simulation methods are presented to address the emerging VLSI design topics of Electromigration induced aging and Ising computing and are then applied to the application areas of hardware security and graph partitioning respectively.

The Electromigration aging effect in VLSI circuits is a long-term reliability issue affecting current carrying metal wires leading to IR drop degradation. Typically, simple analytical equations can determine a wire's effective age or if it will be affected by the EM aging effect at all. However, these classical methods are overly conservative and can lead to over design or unnecessary design iterations. Furthermore, it is expected that the EM aging effect will become more severe in future Integrated Circuits (ICs) due to increasing current densities and the prevalence of polycrystalline copper atom structures seen at small wire

dimensions. For this reason, more comprehensive simulation techniques that can efficiently simulate the EM effect with less conservative results can help mitigate over design and increase design margins while reducing design iterations.

The area of Hardware Security is becoming increasingly important as the chip supply chain becomes more globalized and the integrity of chips becomes more difficult to verify. Utilizing the accurate simulation techniques for EM, we can utilize this reliability effect to demonstrate how a reliability based attack could be perpetrated. Furthermore, we can utilize this aging effect as a defense mechanism to help us validate the integrity of an IC and detect counterfeit chips in the component supply chain market.

Ising computing is an emerging method of solving combinatorial optimization problems by simulating the interactions of so-called spin glasses and their interactions. Borrowing concepts from quantum computing, this methods mimics the quantum interaction between spin glasses in such a way that finding a ground state of these spin glass models leads to the solution of a particular problem. In this dissertation, effective methods of simulating the spin glass interactions using General Purpose Graphics Processing Units (GPGPUs) and finding their ground state are developed.

In addition to the GPU based Ising model simulations, important combinatorial problems can be mapped to the Ising model. In this dissertation the Ising solver is applied to graph partitioning which can be utilized in VLSI design and many other domains as well. Specifically, solvers for the max-cut problem and the balanced min-cut partitioning problem are developed.

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Electromigration	1
1.2 Hardware Security	4
1.2.1 Hardware Trojans	5
1.2.2 Counterfeit IC Detection	7
1.3 Ising Computing and Graph Partitioning	8
1.4 Contributions	11
2 Review of EM physics and stress modeling	13
3 Finite Difference Method for Simulating Electromigration in Multi-Branch interconnects	17
3.1 Void Volume Calculation	18
3.2 Finite Difference Method for EM Analysis	18
3.3 Numerical result and discussions	21
3.3.1 Validation for single 2-terminal wire	22
3.3.2 Validation for three wire T-shape intersection	23
3.4 Summary	25
4 Krylov Subspace Method for Fast Electromigration Simulation	31
4.1 The linear time invariant ordinary differential equations for EM stress evolution	32
4.1.1 Steady-state analysis for nucleation phase	36
4.2 The proposed Krylov fast EM stress analysis	38
4.2.1 Singularity mitigation for EM ODE matrices	38
4.2.2 Fast Krylov subspace-based stress analysis	40
4.2.3 EM simulation under time-varying temperature	45
4.2.4 Scaling schemes for numerical stability	47
4.3 Numerical results and discussions	48

4.3.1	Accuracy study	49
4.3.2	Eigenvalue analysis	51
5	Electromigration Based Hardware Trojans Design	58
5.1	Three-phase EM model	59
5.2	EM-based hardware attack modeling	62
5.2.1	Challenges in EM-based attacks	64
5.2.2	Electromigration topology effects	65
5.3	EM attack methods	69
5.3.1	EM as a Trojan payload	69
5.3.2	EM as a Trojan trigger	74
5.4	Mitigation techniques for EM-based Trojans	76
5.4.1	Split-fabrication	76
5.4.2	Burn-in testing	78
5.5	Conclusion	80
6	On-Chip Counterfeit IC Detection Using Electromigration Aging Sensor	81
6.1	Existing on-chip aging sensor design	82
6.2	EM aging sensor based on configurable multi-segment wire structure	84
6.2.1	Chip authentication methodology	87
6.2.2	Age sensing by wire delay estimation	88
6.2.3	New configurable hybrid multi-segment wire design	89
6.2.4	The complete aging sensor design	91
6.3	Numerical results and comparison	95
7	GPU-based Ising Computing for Max-cut	99
7.1	Review of existing work	100
7.2	Ising model and Ising computing	102
7.2.1	Ising model overview	102
7.2.2	Annealing method for Ising model solution	106
7.3	Ising model for the max-cut problem	109
7.4	GPU Implementation	111
7.4.1	GPU Architecture	111
7.4.2	Ising model implementation	113
7.5	Experimental Results	116
7.5.1	Accuracy study	117
7.5.2	Performance study	120
7.6	Summary	129
8	GPU-based Ising Computing for Balanced Min-cut Bi-partitioning	130
8.1	Ising model for balanced min-cut partitioning	131
8.2	GPU-based Ising solver for balanced min-cut problem	133
8.2.1	Standard implementation	133
8.2.2	Globally Decoupled Ising implementation	135
8.2.3	Further discussion and comparison	137

8.3	Experimental results and discussions	138
8.3.1	Solution time study	138
8.3.2	Solution quality study	142
8.4	Summary	144
9	Conclusion	146
9.1	Summary of Contributions	146
9.1.1	Reliability	146
9.1.2	Hardware security	147
9.1.3	Ising computing	148
	Bibliography	149

List of Figures

2.1	Two-segment wire with steady-state EM induced stress.	14
2.2	EM-stress evolution in the nucleation phase (a) and growth phase (b). . . .	16
3.1	The testing interconnect structures: (a) 1-wire (b) 2 connected wires	23
3.2	EM-stress distribution change over time with $j = 5 \times 10^9 A/m^2$ in single 2-terminal wire for void growth	24
3.3	EM-stress distribution change over time with $j = 5 \times 10^9 A/m^2$ in single 2-terminal wire for void growth	25
3.4	T-shape nucleation phase for horizontal wire	26
3.5	T-shape nucleation phase for vertical wire	27
3.6	T-shape growth phase for horizontal wire	28
3.7	T-shape growth phase for vertical wire.	29
3.8	Growth of void volume over time (t=1E7)	30
4.1	Discretization of the two-segment wire with length L , and segment lengths $\frac{L}{2}$	32
4.2	A two-terminal wire with the electron flow indicated by the arrow.	36
4.3	Example piecewise constant current density \mathbf{j} input as a function of time t	41
4.4	Cathode stress over time for multiple wire temperatures	46
4.5	Nucleation stage validation for two wire segments. <i>FastEM</i> solved using $q = 7$ with 0.0155% average error.	50
4.6	Growth stage validation for two wire segments. <i>FastEM</i> solved using $q = 7$ with 0.0123% average error.	51
4.7	Cathode stress comparison under piecewise constant current density input.	52
4.8	Cathode stress under varying temperature T= 360K, 370K, 365K, 390K.	53
4.9	Eigenvalue plot showing all eigenvalues.	54
4.10	Eigenvalue plot without the two largest values.	55
4.11	Stress near nucleation time for 174-segment tree with different numbers of computed poles.	57
5.1	IC design and manufacturing flow showing attack and detection opportunities	63
5.2	Via-below (a) and Via-above (b) wire structures showing void formation lo- cations.	66

5.3	A two segment wire structure with the same current density (below) and different current densities (above)	67
5.4	Illustrations of the reservoir reduction and sink insertion attacks	70
5.5	An example reservoir reduction attack	71
5.6	An example sink insertion attack	72
5.7	The originally up-stream wire is moved to a lower level of metalization to put it in the down-stream configuration in the layer demotion attack	73
5.8	Example circuit of an EM Trojan wire being used as a trigger for a Trojan gate payload	75
5.9	Burn-in testing reduces the range of failure times available to an attacker by inducing failure in aggressive TTF targets.	79
6.1	The existing EM-based aging sensor design [37, 38]	85
6.2	The EM aging sensor self calibration methodology	87
6.3	The IC supply chain life cycle depicting the pre-distribution self calibration stage, potential counterfeiting, and authentication during the IC distribution.	88
6.4	Transient SPICE simulation of the RC model while sweeping the resistance value showing the gradual degradation of rise time, and thus signal delay	90
6.5	The proposed hybrid multi-segment wire structure design	91
6.6	The EM induced delay-based aging sensor architecture	93
6.7	SPICE simulation of the differential amplifier showing the detection of the delay degradation caused by increased resistance in an EM failed wire	93
6.8	The proposed EM induced delay-based aging sensor architecture	94
7.1	The 2D nearest neighbor Ising model.	103
7.2	An example of a generally connected Ising model.	106
7.3	Depiction of the local minima and global minimum in the energy minimization problem.	107
7.4	The Nvidia CUDA programming model showing the Host (CPU) and Device (GPU) and the relation between threads, blocks, and grid s.	112
7.5	Convergence region of the GPU-based annealing for the Ising model on the G-set G47 problem.	120
7.6	Speedup results of the GPU against the CPU for the G-set benchmark problems.	121
7.7	Graphic of the performance in seconds of the GPU and CPU for increasingly large graphs.	124
7.8	The performance results for the GPU against the CPU, separated by the graph type, torus and non-torus.	125
7.9	CPU performance comparing torus and non-torus graphs.	127
7.10	GPU performance comparing torus and non-torus graphs.	128
8.1	Performance gains of the GDI solver min-cut algorithm.	140
8.2	Performance comparison of the GDI GPU Ising solver and METIS.	141
8.3	Solution quality of the proposed GPU-based Ising solvers.	142
8.4	Cut quality results for the G-set benchmark problems.	143

List of Tables

3.1	Notations and typical value in our transient simulation	22
4.1	Single-threaded performance comparison between FDTD and <i>FastEM</i> on interconnect trees from the IBM power grid benchmark <i>ibmpg2</i>	53
4.2	Multi-threaded performance comparison between FDTD and <i>FastEM</i> on interconnect trees from the IBM power grid benchmark <i>ibmpg2</i>	54
4.3	Scalability performance results comparing FDTD and <i>FastEM</i> using increasingly large n-segment trees.	56
6.1	The designed hybrid wire sensor parameters	96
6.2	Acceleration operating conditions of designed wire	97
6.3	Comparison with the EM-sensor in [38]	98
7.1	Accuracy comparison of the GPU max-cut value against the best known cut values for the G-set benchmark.	118
7.2	Accuracy comparison of the GPU max-cut value against the cut values obtained by CPLEX.	118
7.3	Performance results comparing GPU performance against CPU performance for the G-set benchmark (G prefix) problems and large custom problems (C prefix).	122
8.1	Summary of results for the standard (std.) and Globally decoupled (GDI) Ising solver compared with the METIS results. Graphs with less than 1000 nodes are omitted	139

Chapter 1

Introduction

This dissertation focuses on the efficient and practical simulation of physical phenomenon in ICs and applies these simulations to practical problem domains. Specifically, simulation of the Electromigration aging effect is firstly explored and then applied to the area of Hardware Security. The dissertation then explores the simulation of Ising Spin glasses as a method of solving combinatorial optimization problems. An introduction to each of these subjects is presented in the following.

1.1 Electromigration

Electromigration (EM) is the primary long term aging effect in Integrated Circuits (IC) affecting the metal interconnects subjected to electrical current. It has been estimated that the EM induced lifetime of interconnects will be halved for each new generation of process technology node [44]. This is due to the increasing current density and also because the metal deposition process of copper dual damascene interconnects at small scales leads

to polycrystalline grain structures which are more susceptible to EM failure. As a long term aging effect, electromigration affects the expected lifetime of an IC and must be considered during reliability sign-off.

The electromigration effect comes from the momentum transfer between electrons colliding with the metal atoms in a confined metal wire [14,51]. In a perfectly uniform atomic lattice the number of collisions is minimal, however; in the presence of discontinuities in this lattice, which is the scenario involving polycrystalline grain structures described previously, the collisions are more frequent leading to more momentum transfer resulting in stress generation in the wire. Typically, this stress will remain in an equilibrium state, with tensile stress in the cathode of a wire and compressive stress in the anode of the wire. However, if the stress reaches a critical level, the atoms in the cathode of the wire can diffuse and end up deposited at the anode. This metal migration can then cause the resistance of the wire to increase as a void forms in the cathode of the wire which decreases the volume of the conducting metal volume and ultimately may even force electrons to conduct through the highly resistive liner of the metal wire.

Traditionally, semi-empirical analytical methods are used to predict a wire's susceptibility to EM. Primarily, Black's equation [14] is used to estimate a wire's Mean Time To Failure (MTTF) as shown below in (1.1).

$$MTTF = Aj^{-n}exp\{E_a/k_B T\} \tag{1.1}$$

In this equation, j is the current density, k_B is the Boltzmann constant; T is the absolute temperature, E_a is the EM activation energy, A is a process dependent constant. A

is typically determined from a variety of parameters such as the metal grain structure, wire geometry, wire test conditions and others. The current density exponent n is also dependent on a number of parameters but primarily is material dependent. However, it has been found to be a variable parameter that heavily depends on the stressing conditions and is overly conservative which leads to over design and between 2X and 3X larger guard-bands than necessary [9]. This comes from the fact that parameters such as n are highly dependent on the stressing conditions and EM testing, which is used to determine these parameters, is done at high stressing conditions and then extrapolated to normal use conditions. In addition to resulting in over design, certain applications require accurate Time To Failure estimations, in which these traditional methods are not adequate.

To mitigate these issues, Physics-based models have been proposed which more accurately describe the EM effect and its subsequent failure mechanics. A good summary of these models can be found in [28]. The primary models in use are the stress-based Korhonen equation [51] and the metal atom vacancy concentration based model by Clement [25]. In this work the stress based Korhonen model is used. However, it should be noted that these models are actually the same and can both be used to derive each other but the stress based method is more intuitive for conceptualization of the EM mechanics.

The Korhonen model has been extended in [72] to describe a two phased failure process consisting of a nucleation phase and a growth phase. In the nucleation phase stress is generated in the wire until the wire reaches a critical tensile stress level which causes atoms to migrate. Once atoms begin to migrate, the second phase known as the growth phase starts and describes the growing void volume due to atom depletion [72].

Efficiently solving this physics model can generate less conservative EM assessments leading to less over-design and less design iterations. Furthermore, specific applications that require high accuracy EM simulation can benefit greatly. This manuscript presents a numerical solution technique for the EM-induced stress dynamics PDE for both the void nucleation and void growth phases in multi-branch interconnects. Additionally, a model reduction technique is employed to vastly accelerate simulation time while also considering time-varying temperature and current densities.

1.2 Hardware Security

The integrated circuit (IC) supply chain has become increasingly globalized giving semiconductor companies unprecedented access to over-seas markets. A consequence of this proliferation is that it is becoming increasingly difficult to validate the integrity of the component supply chain [81] giving rise to illicit markets that deal in counterfeit ICs or even the malicious alteration or otherwise unsanctioned uses of electronics components.

The counterfeiting of ICs pose significant financial and security risks. The International Chamber of Commerce estimated losses due to counterfeiting and IP theft for G20 nations to be as high as \$1.7 trillion in 2015 [20]. Furthermore, the illicit market discourages innovation as companies become more weary of disclosing technologies with the public. The concern over counterfeit ICs in these critical application areas was validated when the U.S. military discovered counterfeit ICs in several defense systems [85] which highlights the need to develop countermeasures.

Furthermore, the risk of malicious attacks from so-called hardware Trojans is an increasing concern. Often it is the case that IC design companies utilize 3rd party foundries to fabricate their ICs. Many of these dedicated foundries are over-seas and out of reach for domestic regulatory oversight and are difficult for their clients to directly monitor the manufacturing process. Because of this, the potential for bad actors at these 3rd party foundries to maliciously alter a device to create a hardware Trojan exists.

The security concern over both counterfeit ICs and hardware Trojans, primarily for critical applications (e.g. aerospace, defense, utility, and medical) has resulted in numerous calls for research from government agencies such as the IARPA TIC program [4], and the DARPA TRUST [3] and IRIS [2] programs. To combat these threats, developing novel methods of attack and defense are required to build an effective defensive toolbox for IC designers. The work in this dissertation builds upon the modeling and simulation methods developed for EM and applies these tools to create novel contribution in the space of hardware security.

1.2.1 Hardware Trojans

Hardware Trojans are malicious alterations or additions to an IC. They may be implemented at a foundry without the original IC designer's knowledge. The scope of these malicious alterations can vary and may be as small as a single gate or may contain more complex logic modules. Hardware Trojans typically have two main components, a trigger and a payload [79, 88]. The Trojan trigger activates the Trojan so it can carryout the task it was designed for. Trojan Triggers may be activated by a certain bit pattern in the IC's logic, they may use external sensors to utilize temperature or other environmental effects, or

they may even be timer based. The Trojan payload is the part of the Trojan that actually performs the attack. The attack can be as simple as disabling a logic gate or may be as complicated as to leak sensitive information to the outside world.

One method of attack, reliability-based Trojans, utilizes the semiconductor aging effects to carry out an attack [69, 71, 83]. Attacks utilizing reliability effects may attempt to accelerate IC aging so as to cause premature failure of the victim chip. Primarily, reliability-based attacks utilize device reliability effects such as Bias Temperature Instability (BTI) and Hot Carrier Injection (HCI). However, on a couple attacks have been proposed that utilize long term interconnect aging [69, 71]. These works propose both EM and Time-Dependent Dielectric-Breakdown (TDDB) type attacks but they only rely on current density based methods and semi-empirical aging models, such as Black's equation, which are meant for conservative reliability sign-off and are not suitable for Trojan designs.

Reliability-based Trojan attacks have a tremendous advantage over classical Trojans, that rely on sensors and logic, as they are more difficult to activate. The most commonly proposed methods of detecting hardware Trojans involves functional testing designed to trigger the Trojan during testing and allowing the observation of the attack and may also measure chip parameters such as temperature and power [79, 88]. However, if a Trojan utilizes reliability effects, it can be difficult to produce the conditions required to cause the activation of such a Trojan.

In this dissertation, the EM effect and the simulations methods presented are utilized to conceptualize and design EM-based hardware Trojans that can be utilized as a trigger, payload, or combination of both. Advanced EM failure dynamics are also considered which allow for finer control of the failure process.

1.2.2 Counterfeit IC Detection

Counterfeit ICs represent the portion of the illicit IC market that sales ICs that do not conform to the original design specification, are old ICs that are sold as new, or are remarked to misrepresent the IC as something it is not. It often is the case that counterfeit ICs are combinations of these attributes, e.g., an old used IC remarked as a new IC with enhanced features that the actual IC does not have [7, 34]. Counterfeit ICs represent a significant financial and security threat.

The typical method of detecting a counterfeit IC involves visually inspecting a batch of electronics components [85]. More advanced visual inspection methods have been proposed such as the work in [33] that relies on thermal imaging in conjunction with statistics to increase the efficiency of visual inspection. However, these are time consuming processes requiring special expertise and are not always reliable [85]. For this reason, on-chip aging sensors have been proposed to combat counterfeit ICs [80]. Early aging sensors used counters but these are vulnerable to attack. This led to the development of sensors based on the semiconductor aging effects. The first proposed aging sensor used transistor aging in a ring oscillator which, as it aged, would have frequency variations which could be used to indicate a used IC. However, these methods can only detect short lifetimes, on the order of weeks, making false positives due to testing during manufacturing possible.

To measure long term aging, EM-based aging sensors were proposed [37,38]. Unlike the BTI effect, EM is a long term aging effect which makes it ideal for measure age in units of years rather than weeks. The initially proposed EM-based aging sensors had the draw back of being highly area inefficient, requiring a large number of long redundant wires and multiple modules for detecting multiple ages.

The work presented in this manuscript shows a novel self-calibrating EM-based aging sensor that vastly improves upon the previously proposed aging sensors by removing the need for redundant wires and also for multiple sensors to measure larger ranges of ages.

1.3 Ising Computing and Graph Partitioning

There are many hard combinatorial optimization problems such as max-flow, max-cut, graph partitioning, satisfiability, and tree based problems, which are important for many scientific and engineering applications. With respect to VLSI design automation, these problems translate to finding optimal solutions for cell placement, wire routing, logic minimization, via minimization, and many others. The vast complexity of modern integrated circuits (ICs), some having millions or even billions of integrated devices, means that these problems are almost always computationally intractable and require heuristic and analytical methods to find approximate solutions. It is well-known that traditional von Neumann based computing can not deterministically find polynomial time solutions to these hard problems [67].

To mitigate this problem, a new computing paradigm utilizing the *Ising spin glass model* or *Ising model* has been proposed [61]. The Ising model is a mathematical model

describing interactions between magnetic spins in a 2D lattice [59]. The model consists of *spins*, each taking one of two values $\{+1,-1\}$ (to represent up and down states of a spin along a preferred axis) and are generally arranged in a 2D lattice. The spin's value is determined so that its energy is minimized based on interactions with its neighbor spins. Such local spin updates will lead to the ground state (globally lowest energy configuration) of the Ising model. It was shown that many computationally intractable problems (such as those in class NP complete or NP hard) can be converted into Ising models [56]. Some natural processes, such as quantum annealing process, were proposed as an effective way for finding such a ground state [17, 48]. D-Wave [6] is one such quantum annealing (also called adiabatic quantum computation) solver based on the Ising model and it shows 10^8 speedup over simulated annealing on the weak-string cluster pair problem [30]. However, existing quantum annealing requires close to absolute zero temperature operating on superconductive devices, which are very complicated and expensive. Furthermore, these machines currently are very limited on the size and complexity of the problems they can solve.

While quantum computing has yet to reach maturity, there exists a number of other hardware-based annealing solutions which have been proposed to exploit the highly parallel nature of the annealing process used to solve the Ising model. In [89], a novel CMOS based annealing solver was proposed in which an SRAM cell is used to represent each spin and thermal annealing process was emulated to find the ground state. In [35, 90], the FPGA-based Ising computing solver has been proposed to implement the simulated annealing process. However, those hardware based Ising model annealing solvers suffer several problems as detailed in Section 7.1.

In this dissertation, a GPU-based Ising model solver is proposed, using a modified simulated annealing heuristic, that can handle any combinatorial problem that is mapped to the Ising model as well as any general problem case associated with it. The maxcut solution is presented first as it is relevant to many VLSI design automation problems. Furthermore, max-cut is a relatively difficult problem to solve without highly efficient heuristic solvers. This manuscript will show that Ising computing by the simulated annealing process is very amenable to fine-grain GPU-based parallel computing. Further proposed is an update method that utilizes the GPU scheduler to achieve a random update pattern enabling independent parallel spin updates. This allows us to maximize thread utilization while also avoiding sequential and deterministic update patterns for a more natural annealing process.

In addition to the max-cut problem, which has a trivial mapping to the Ising model, a solution method is presented for the balanced min-cut bi-partitioning problem. In this problem, the balance constraint will lead to a complete graph in the resulting Ising model. The reason is that the *balance* constraint is a global constraint. As a result, each Ising spin glass is connected to all the Ising spins glasses in the graph, therefore; each local spin update becomes a global update. Current Ising model solvers and quantum annealing computers often do not have architectures amenable to the embedding of problems with complete graphs, e.g., the Chimera graph architecture used in D-wave computer [6]. However, recent study shows that balanced min-cut problems on the D-wave computer indeed yields better results than the state of the art partitioning solvers like METIS [82], but the problem sizes solved are still limited to thousands of nodes and requires co-processing on traditional hardware due to the limited number of available qubits in the QA machine.

In this dissertation, an efficient method for mapping the balanced min-cut bipartitioning problem is presented that handles the balancing constraint more efficiently avoiding the complete graph structure and essentially decoupling the spins globally.

Furthermore, unlike other hardware accelerated solvers for the Ising model (e.g. FPGA and ASIC implementations), the proposed method for both max-cut and min-cut solvers do not require solving the NP-hard graph embedding problem which would drastically increase the computational complexity of the solver.

1.4 Contributions

The work presented in this manuscript presents several contributions in the area of simulation and the application of those simulation techniques to the area of reliability based hardware security:

- The first proposed numerical solution to the Korhonen PDE for transient simulation of EM-induced Stress Dynamics for multi-branch interconnects considering both void nucleation and void growth phases.
- A Krylov subspace based model reduction technique to drastically speed-up the numerical solution of the Korhonen PDE. This novel approach is also the only known model reduction method for transient EM simulation that can consider time varying current and temperature.

- The first proposed comprehensive EM-based Hardware Trojan design utilizing physics based models and advanced EM failure dynamics for both Trojan triggers and payloads.
- A Novel self-calibrating EM-based aging sensor to combat counterfeit ICs. The presented approach drastically improves upon previously proposed EM-based aging sensors, and aging sensors in general, in terms of circuit area, accuracy, and range of lifetimes measurable.
- A GPU accelerated method for solving generally connected Ising spin-glass models with applications in graph partitioning. The solver presented avoids costly graph embedding required for other hardware-based solutions when considering complex graph structures. The work addresses both the max-cut and balanced min-cut bipartitioning problems.

Chapter 2

Review of EM physics and stress modeling

This chapter presents a review of the Electromigration (EM) aging effect. EM is a physical phenomenon of the migration of metal atoms along the direction of the applied electrical field. Atoms (either lattice atoms or defects / impurities) migrate along the trajectory of conducting electrons. During the migration process, hydrostatic stress is generated inside the embedded metal wire due to momentum exchange between lattice atoms and electrons resulting in tension at the cathode and compression at the anode ends of the line. Void and hillock formation are created by the depletion and subsequent deposition of atoms respectively. Fig. 2.1 shows the typical copper dual damascene interconnect structure, which has three terminal vias connecting to other interconnect wires, and the steady-state stress distributions before void nucleation. As time goes on, the lasting unidirectional electrical load will increase hydrostatic stress, as well as the stress gradient which

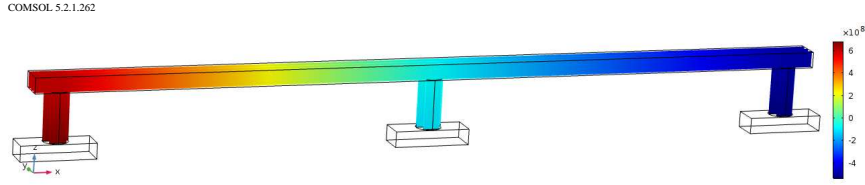


Figure 2.1: Two-segment wire with steady-state EM induced stress.

acts as a counter-force for atomic migration along the metal line. Generally, when a wire is long, this stress can reach a critical level, resulting in void nucleation at the cathode and/or hillock formation at the anode end of line.

The currently employed method for estimating time to failure is based on Black's equation [14],

$$MTTF = Aj^{-n}exp\{E_a/k_B T\} \quad (2.1)$$

where j is the current density, k_B is the Boltzmann's constant; T is the absolute temperature; and E_a is the EM activation energy. The symbol A is a constant, which depends on a number of factors, including grain size, line structure and geometry, test conditions, current density, thermal history, etc. The current exponent n was found to be 2 for aluminum interconnects in [14]. However, Black's equation is under growing criticism as the extracted parameters, for example, the current exponent n and activation energy E_a , are not constant and are stress condition-dependent. This is problematic because these parameters are typically found using high stressing accelerated test conditions which are extrapolated to the real use case. This essentially means the parameters used at testing and not realistic representations of the parameters that should be used in real use conditions. Furthermore, this equation is only applicable to a single wire segment.

Physics based EM models have been proposed [25, 28, 51] which can be used to mitigate the issues with Black's equation. For a general interconnect wire in two dimensions, transient hydrostatic stress evolution due to EM effects is analyzed in the Korhonen model [51] and stress $\sigma(x, t)$ is described by Korhonen's partial differential equation (PDE) with the following zero-flux boundary conditions (BC) and initial stress condition (IC):

$$\begin{aligned}
PDE : \frac{\partial \sigma}{\partial t} &= \frac{\partial}{\partial x} \left[\kappa \left(\frac{\partial \sigma}{\partial x} + G_x \right) \right] + \frac{\partial}{\partial y} \left[\kappa \left(\frac{\partial \sigma}{\partial y} + G_y \right) \right] \\
BC : \frac{\partial \sigma}{\partial x}(0, t) &= G_x, \quad \frac{\partial \sigma}{\partial y}(0, t) = G_y \\
BC : \frac{\partial \sigma}{\partial x}(L, t) &= -G_x, \quad \frac{\partial \sigma}{\partial y}(L, t) = -G_y
\end{aligned} \tag{2.2}$$

at $0 < t < t_{nuc}$

$$IC : \sigma(0) = [\sigma_1(0), \sigma_2(0), \dots, \sigma_n(0)] \text{ at } t = 0$$

Here, $\kappa = D_a B \Omega / k_B T$ and $G = \frac{eZ\rho j}{\Omega}$, which is a function of current density j and $D_a = D_0 \exp\left(-\frac{E_D - \Omega^* \sigma_T}{k_B T}\right)$ is the effective atomic diffusivity where E_D is the activation energy of the atom diffusion, T is the absolute temperature and k is the Boltzmann constant. B is the effective bulk elasticity modulus. And $G = \frac{eZ\rho j}{\Omega}$, where e is the electron charge, eZ is the effective charge of the migrating atoms, ρ is the wire electrical resistivity, and j is the current density.

When tensile stress reaches a critical level, a void is formed at the cathode. When this void is formed, the stress at the void location will immediately go to zero, however; the stress around the void will be close to the same stress level as it was immediately prior to the void nucleation [51, 72]. This creates a large stress gradient around the void at nucleation time as described by [51]:

$$\frac{\partial \sigma}{\partial x}(x_{nuc}, t) = \frac{\sigma(x_{nuc}, t)}{\delta}, \text{ at } t_{nuc} < t < \infty \quad (2.3)$$

In Equation (2.3), x_{nuc} is the location of the void nucleation at a boundary and δ is the width of the void interface [73].

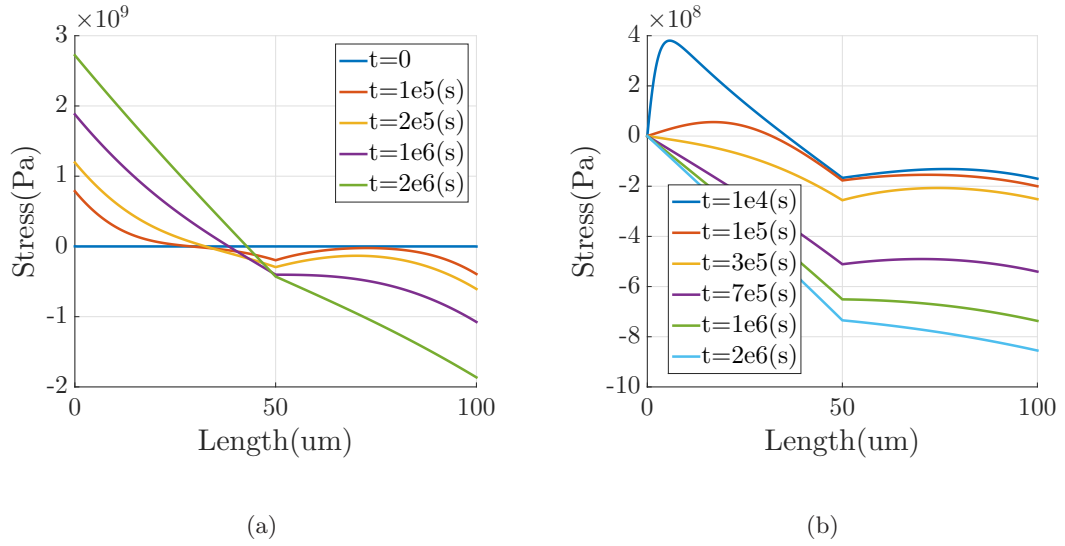


Figure 2.2: EM-stress evolution in the nucleation phase (a) and growth phase (b).

Fig. 2.2 shows stress development over time in a two-segment wire (shown in Fig. 2.1) for Korhonen's equation for both nucleation phase (Fig. 2.2(a)) and growth phase (Fig.2.2(b)). For nucleation phase, over time, tensile (positive) stress will be developed at the cathode (left) node and compressive (negative) stress will be developed at the anode (right) node. The built-up stress (its gradient) will serve as the back force for atomic flux. If the highest stress at the cathode node exceeds the critical stress, voids will be created. The time to reach the critical stress is called nucleation time(t_{nuc}).

Chapter 3

Finite Difference Method for Simulating Electromigration in Multi-Branch interconnects

This chapter presents the Finite Difference Time Domain method (FDTD) used for simulating dynamic stress evolution in the IC back-end-of-line. By solving the PDE for dynamic stress evolution from 2 that leads to EM failure, this method avoids the conservativeness introduced by the semi-empirical Black's equation and the Blech Limit [15](unlike Black's equation, the Blech limit does not estimate a time to failure but rather determines EM mortality). The new simulation method captures both the nucleation and growth stress dynamics and also enables the simulation of EM in multi-branch interconnects. This is an important consideration as it has been shown that wires in an interconnect tree are not independently effected by EM but instead their stress distribution and generation are cou-

pled with neighboring segments in the tree [21, 58]. We further calculate the void volume growth allowing for the determination of resistance degradation in the interconnect tree.

3.1 Void Volume Calculation

During the void growth phase, a void has actually nucleated in the wire and begins to grow. This void causes changes to the electrical properties of the wire, resistance for example, and leads to the eventual failure of the interconnect. For this reason, being able to measure the void volume growth is critical to the proper analysis of the interconnect. Using the dynamic stress evolution of the wire, which will be calculated by the proposed FDTD method, the volume of the void can be calculated as it grows. This is accomplished by calculating the atom drift volume in the wire [76]. Essentially, the stress σ over bulk elasticity modulus B is integrated over the length L of the wire and multiplied by the wire's cross-sectional area A .

$$V(t) = A \int_0^L \frac{\sigma(t)}{B} dx \quad (3.1)$$

From this equation we can then begin to make inferences about the effect the growth will have on the interconnect's electrical properties.

3.2 Finite Difference Method for EM Analysis

The Finite Difference Time Domain (FDTD) method is a numerical method of solving a Partial Differential Equation (PDE) [66]. A Numerical method finds an approx-

imate solution by iteratively obtaining values as apposed to analytically finding a solution to a PDE. FDTD gives us the ability to quickly and easily find a solution for complicated PDEs that would otherwise be difficult to solve using analytical methods. Furthermore, it has the ability to handle complex geometries.

FDTD works by discretizing both the time and spatial variables in a PDE. This discretization is accomplished by employing the local Taylor expansion to PDE, in conjunction with the wire geometry being discretized, which yields a system of linear equations which are solved for the discrete values within the geometry. While there are several discretization schemes, this work uses the central difference method to discretize the spatial variable x and the first order backward method to discretize time t .

$$\frac{\sigma_i^{n+1} - \sigma_i^n}{\Delta t} = \kappa \frac{\sigma_{i+1}^{n+1} - 2\sigma_i^{n+1} + \sigma_{i-1}^{n+1}}{\Delta x^2} \quad (3.2)$$

In (3.2), the superscript n indicates the time step and subscript i is the space index, x .

$$-S\sigma_{i+1}^{n+1} + (1 - 2S)\sigma_i^{n+1} - S\sigma_{i-1}^{n+1} = \sigma_i^n \quad (3.3)$$

where $S = \kappa \frac{\Delta t}{\Delta x^2}$. This discretization method gives us an implicit scheme for solving the PDE numerically. This Implicit scheme allows us more freedom on how large the time step Δt can be as we don't need to worry about stability, which is a problem in explicit methods [66].

For the boundary conditions, stress is dependent on the derivative of the stress function. Thus, Neumann boundaries are used and discretized [53].

$$\sigma_x(0, t) = G = \frac{\partial \sigma_i^n}{\partial x} \quad (3.4)$$

$$\sigma_x(0, t) = \frac{\partial \sigma_i^n}{\partial x} = \frac{\sigma_i^{n+1} - \sigma_{i-1}^{n+1}}{\Delta x} = G \quad (3.5)$$

We can now use this discretization scheme and plug it into (3.3) to obtain the following.

$$(S + 1)\sigma_i^{n+1} - S\sigma_{i+1}^{n+1} = \sigma_i^n - SG\Delta x \quad (3.6)$$

Note that the derivation for the second boundary condition $\sigma_x(L, t)$ is omitted. The resulting system of equations can be mapped to an equation in the form of $A\sigma^{n+1} = \sigma^n$. A is a tri-diagonal coefficient matrix with the diagonal elements equal to $(1 - 2S)$ and the lower and upper diagonal elements equal to $(-S)$. The vector σ^{n+1} is a vector of unknown stress along the wire and σ^n is a vector of previously solved for stress. The first element in σ^n is $\sigma_{left}^n + \beta$ and the last element is $\sigma_{right}^n - \beta$. These correspond to the boundary conditions at each end of the wire where β is equal to $SG\Delta x$.

Each solution of the system of equations results in a vector containing the stress of the wire at a single time step. Subsequent solution of this system, using previous time step solution as the σ^n vector, can produce a vector containing the stress at the respective time steps. By iteratively solving the system of equations, we obtain the transient stress evolution for the entire wire length.

This previous section shows the basic FDTD scheme for a one dimensional wire, however; this can be easily expanded to the two or three dimensional case. The discretization for the two dimensional case used in this work is presented below, albeit without derivation.

$$\begin{aligned}
 -S_y\sigma_{i+1,j}^{n+1} + (1 + 2S_y + 2S_x)\sigma_{i,j}^{n+1} - S_y\sigma_{i-1,j}^{n+1} - \\
 S_x\sigma_{i,j+1}^{n+1} - S_x\sigma_{i,j-1}^{n+1} = \sigma_{i,j}^n
 \end{aligned}
 \tag{3.7}$$

In this equation, i is the discrete variable in the x - *axis* and j is the discrete variable in the y - *axis*. The value S also is specified as S_x or S_y to differentiate discretization steps used in either direction.

The void growth phase follows the void nucleation phase. For void growth phase, the A matrices are similar (with different boundary and initial conditions). When solving, an arbitrary vector of initial conditions can be used or the output stress distribution from the previous void nucleation phase can be used. Once this has been determined by the user, the growth phase portion of the framework operates the same as the void nucleation phase. That is, the new growth phase discretized equation $A\sigma^{n+1} = \sigma^n$ is formed and is then solved iteratively to generate transient data for the wire stress.

3.3 Numerical result and discussions

The proposed FDM-based EM analyzer was prototyped in MATLAB. COMSOL multiphysics [1] is used to validate the numerical solver.

In order to validate our result, a FEA tool, COMSOL [1] is used. In the nucleation phase, the initial conditions are set to be zero and default zero flux boundary conditions are used. In the growth phase, initial conditions come from the time at which the cathode in

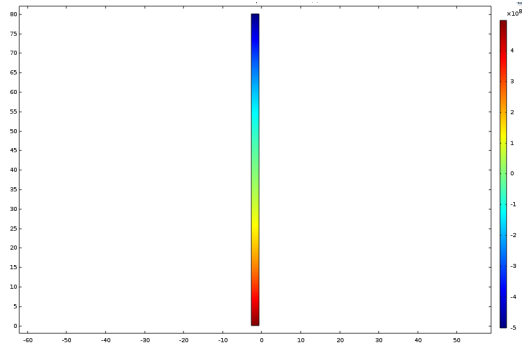
the nucleation phase reaches critical stress. We summarize the major notations and typical parameter values in Table 3.1.

Table 3.1: Notations and typical value in our transient simulation

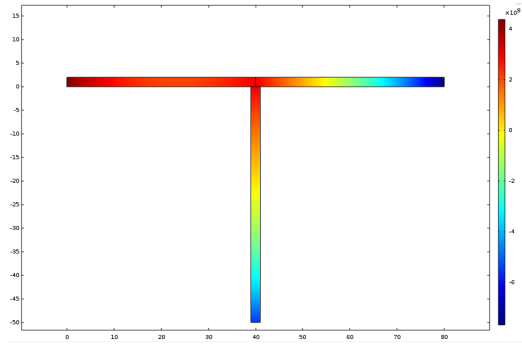
Term	Typical value	Description
ρ	3.00e-8 $\Omega \cdot m$	Electrical resistivity
e	1.60e-19C	Electric charge
Z^*	10	Effective valence charge
Ω	1.18e-29 m ³	Atomic volume
k	1.38e-23J/K	Boltzmann constant
B	1.10e11Pa	Back flow stress modular
D_0	7.56e-5m ² / s	Self-diffusion coefficient
E_a	1.76e-19J	Activation energy
σ_{crit}	500MPa	Critical stress
T	373K	Absolute temperature
δ	3e-7m	Effective thickness of the void interface

3.3.1 Validation for single 2-terminal wire

For the 2-terminal single wire as shown in Fig 3.1(a), tensile stress was generated in the cathode (left node) while the anode node experiences compressive stress. This agrees with our expectations which say that a void should be nucleated at the cathode end of the single wire. Root Mean Squared Error(RMSE) when compared to COMSOL for the void nucleation phase, seen in Fig 3.3.1, and void growth phase, seen in Fig 3.3.1, are 0.8033% and 0.4435% respectively.



(a)



(b)

Figure 3.1: The testing interconnect structures: (a) 1-wire (b) 2 connected wires

3.3.2 Validation for three wire T-shape intersection

In the three wire T-shaped interconnect shown in Fig 3.1(b), currents were applied as: $j_1 = 5 \times 10^9 A/m^2$, $j_2 = -6 \times 10^9 A/m^2$, $j_3 = -7 \times 10^9$. With current flowing in through wire one, we expect the void to nucleate here. Results show that our expectations are met and the numerical data agrees with the COMSOL results for both nucleation, seen in Fig 3.3.2 and Fig 3.3.2 and growth phases, Fig 3.3.2 and Fig 3.3.2. RMSE for the nucleation

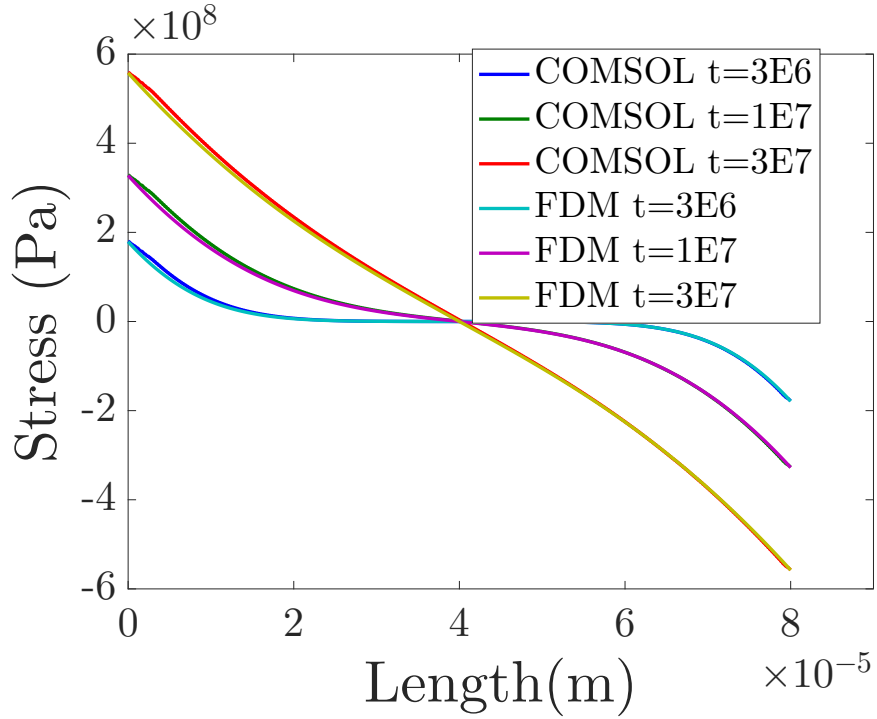


Figure 3.2: EM-stress distribution change over time with $j = 5 \times 10^9 A/m^2$ in single 2-terminal wire for void growth

phase is 2.011% and 2.23% for the growth phase. Results for other current configurations are omitted for space but produce similar results.

Once the dynamic stress evolution data has been collected from the FDTD analysis, we can apply the void volume calculation to see the transient void volume growth. These results are shown in Fig 3.8(a).

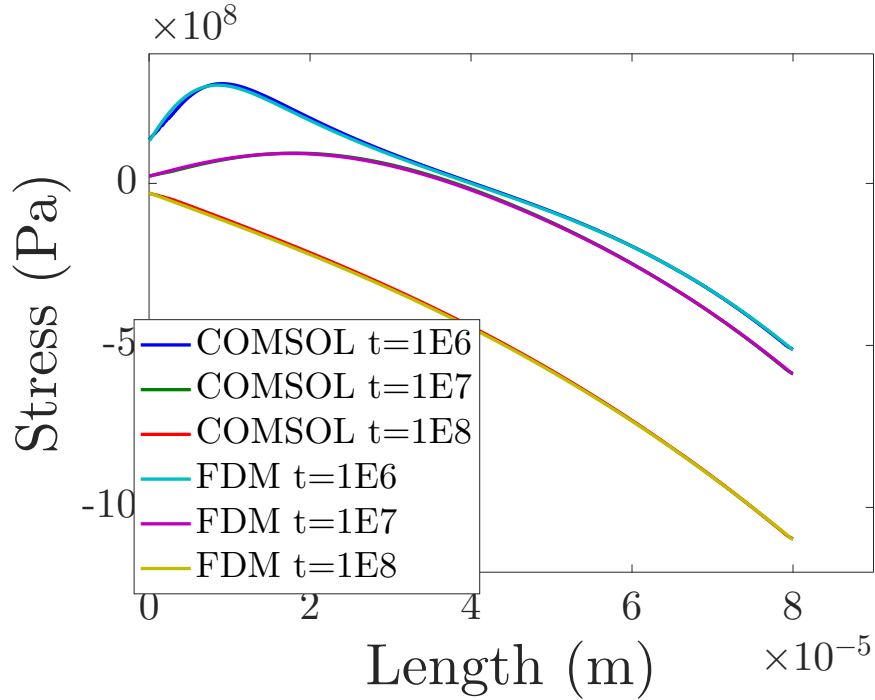


Figure 3.3: EM-stress distribution change over time with $j = 5 \times 10^9 A/m^2$ in single 2-terminal wire for void growth

3.4 Summary

In this chapter, an accurate EM simulation method using FDTD in multi-branch interconnects based on the first principle of EM physics was presented. The presented numerical method is the first consider both the nucleation and growth phase simulations for multi-branch interconnects. Void volume during the void growth phase was also calculated. The presented method also can easily accommodate existing non-uniform residual stress distribution which is another new contribution. lastly, Numerical results showed that the proposed method agrees with the COMSOL based Finite Element Analysis in terms of accuracy which provides validation of the proposed method.

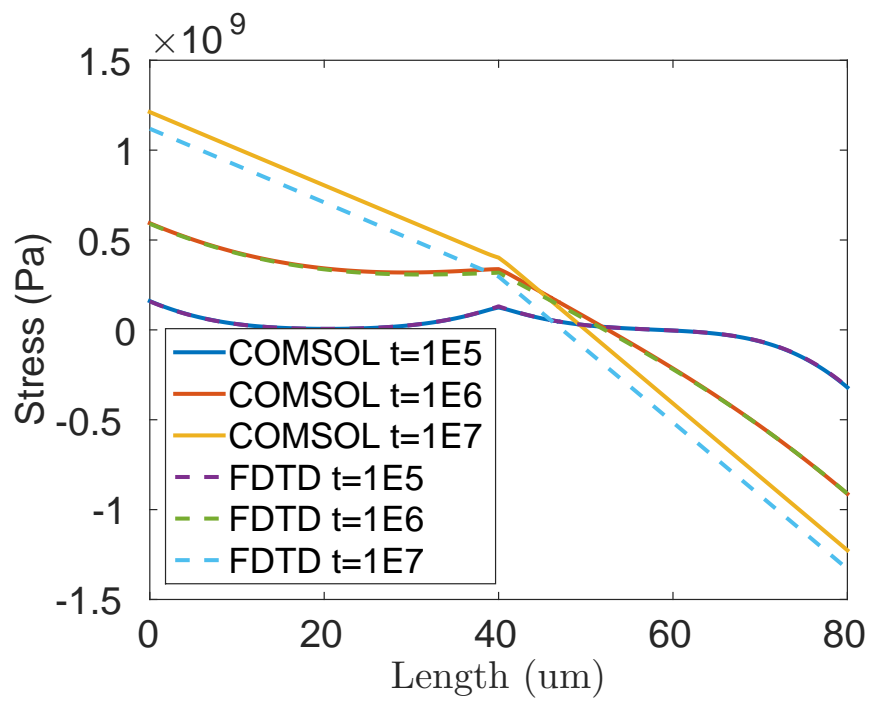


Figure 3.4: T-shape nucleation phase for horizontal wire

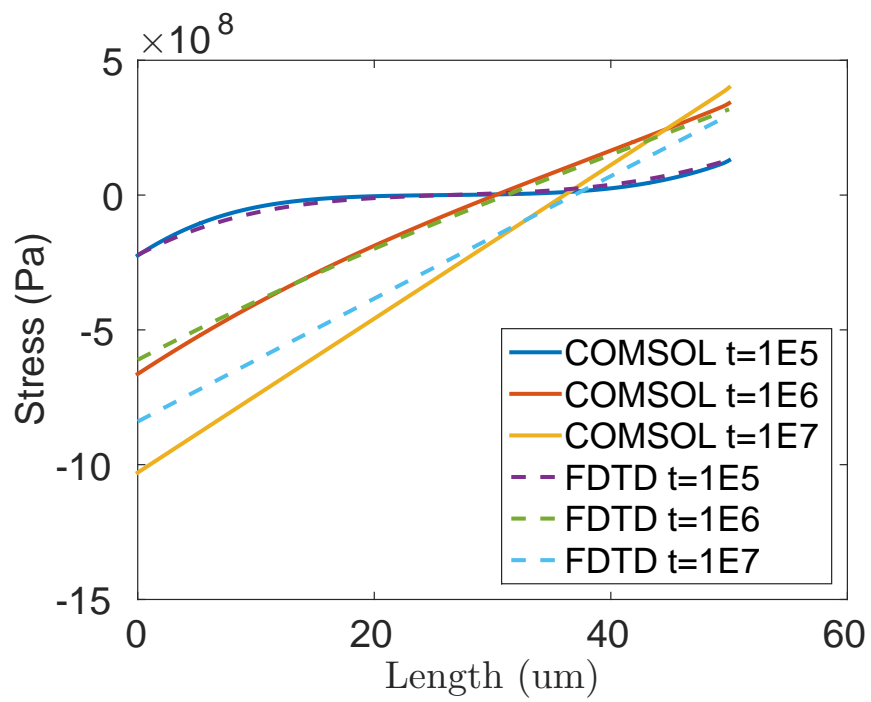


Figure 3.5: T-shape nucleation phase for vertical wire

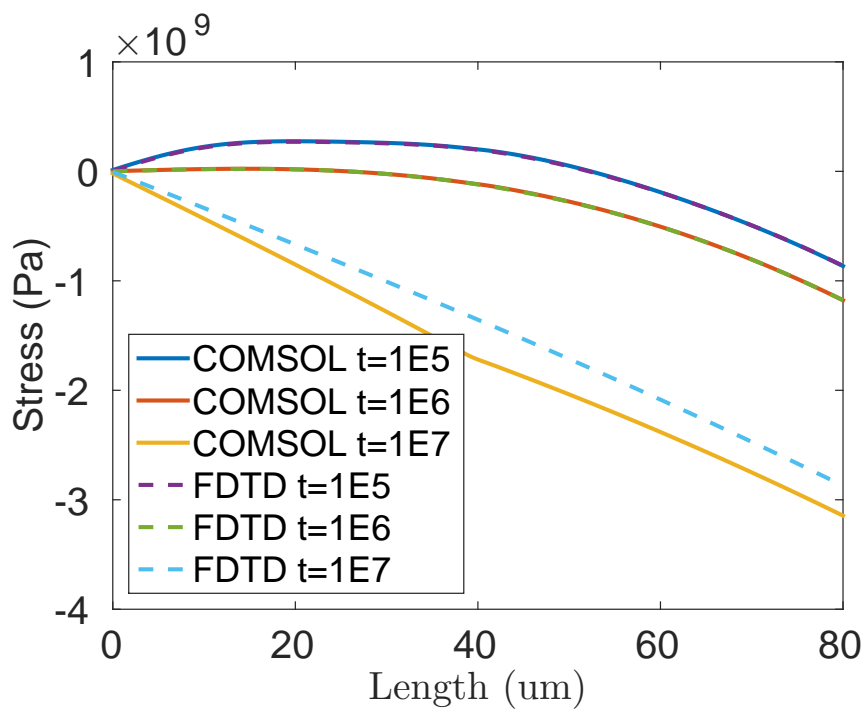


Figure 3.6: T-shape growth phase for horizontal wire

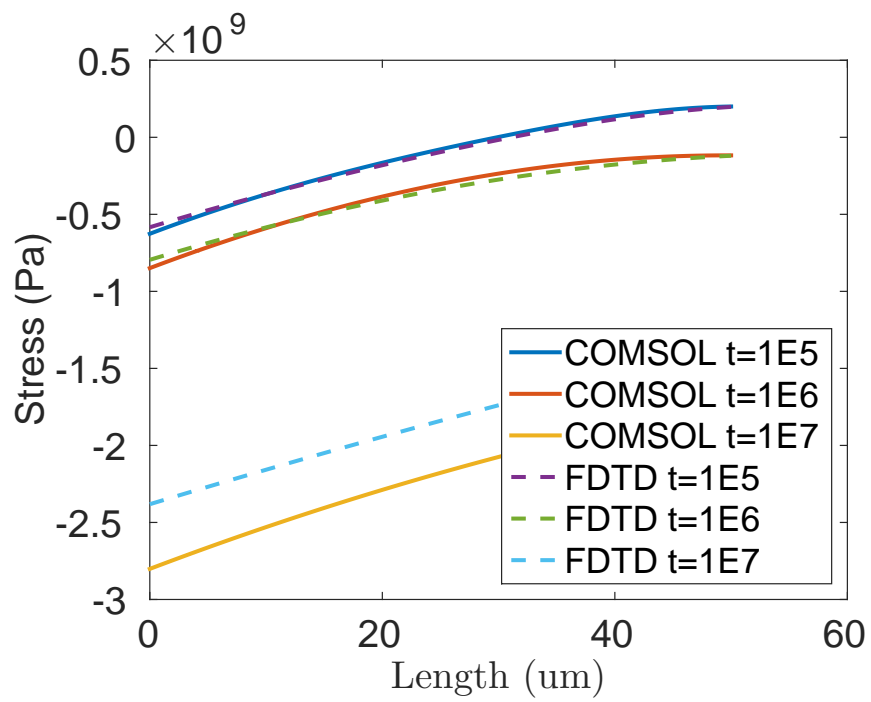
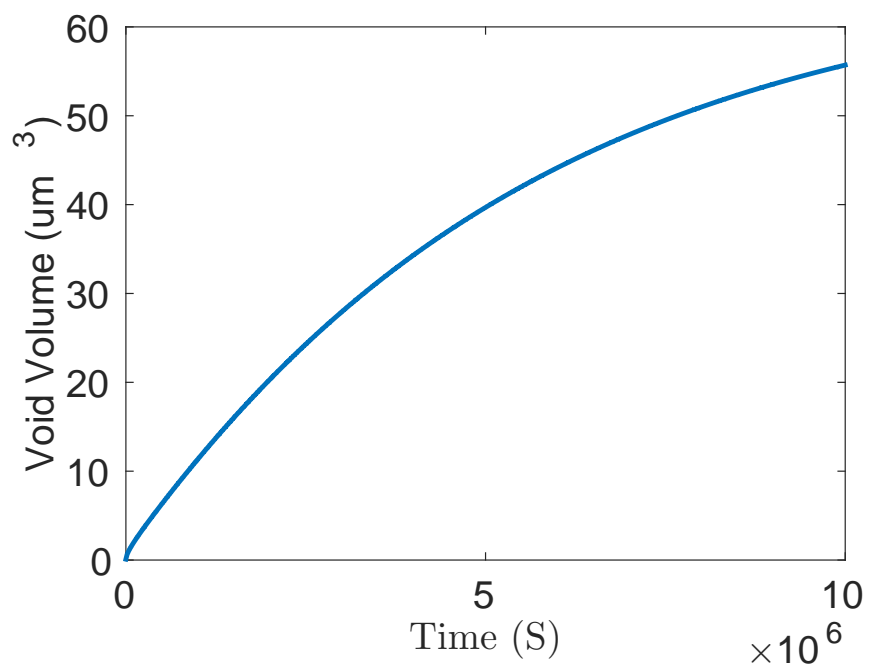


Figure 3.7: T-shape growth phase for vertical wire.



(a)

Figure 3.8: Growth of void volume over time ($t=1E7$)

Chapter 4

Krylov Subspace Method for Fast Electromigration Simulation

This chapter builds upon the previously presented FDTD method for simulating dynamics stress evolution for Electromigration analysis. While the FDTD based method provides a simulation framework for simulating nucleation and growth phases in multi-branch interconnects, the numerical solution time can scale poorly for large interconnect trees. To combat this, this chapter proposes a Krylov subspace-based model reduction technique, called “FastEM”, to accelerate simulation time while maintaining a high degree of simulation accuracy. Furthermore, this chapter also presents a method for transient current density and thermal simulation in the model reduced simulation framework. Numerical results show that the proposed method can lead to 1-2 orders of magnitude speed-up over the existing method.

4.1 The linear time invariant ordinary differential equations for EM stress evolution

In this section, we show how to perform the finite difference discretization for the given stress partial differential equation, also known as the Korhonen equation in (5.1), to create the LTI ODE system. While the derivation is similar to the FDTD presented in the previous chapter, we utilize an implicit method to formulate an LTI System of equations this time.

A two-segment wire example is used throughout the section for demonstration with total length L and separate G values for each segment as shown in Fig. 4.1. The wire is discretized into five nodes; two edge boundary nodes at each end of the wire, one junction node at the middle of the wire, and two non-boundary nodes, each between the junction and an edge node.

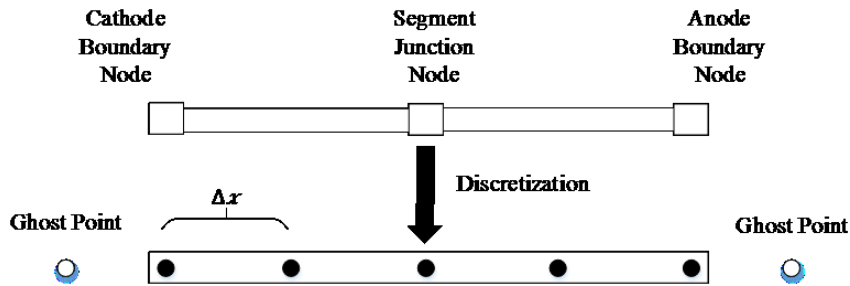


Figure 4.1: Discretization of the two-segment wire with length L , and segment lengths $\frac{L}{2}$.

The finite difference method (FDM) is a method of finding a numerical solution to partial differential equations (PDEs) [66]. The PDE can be discretized using many different methods; in our implementation, a central difference method (4.1) is used to discretize

the spatial variable x (and y in the two-dimensional case as shown later). We use the central difference method due to the low truncation error compared to other discretization methods, such as the forward and backward differences, at the cost of only adding one term to each equation. Note that this is different from the Finite Difference Time Domain method (FDTD) in [26], where time is also discretized.

$$\frac{\partial \sigma}{\partial t}(x, t) = \kappa \frac{\sigma_{i+1} - 2\sigma_i + \sigma_{i-1}}{\Delta x^2}, \quad \kappa = \frac{D_a B \Omega}{kT} \quad (4.1)$$

Boundary conditions are discretized depending on location (internal junctions or edges) and EM phase (nucleation or growth). Edge boundaries are introduced during the handling of ghost points in the discretization scheme. These ghost points are terms in the central difference scheme that do not correspond to physical points on the wire structures. Boundary conditions are discretized using the backward difference scheme shown in (4.2).

$$\begin{aligned} \text{Nucleation : } \frac{\partial \sigma}{\partial x}(0, t) &= \frac{\sigma_i - \sigma_{i-1}}{\Delta x} = G \\ \frac{\partial \sigma}{\partial x}(L, t) &= \frac{\sigma_{i+1} - \sigma_i}{\Delta x} = -G \\ \text{Growth : } \frac{\partial \sigma}{\partial x}(0, t) &= \frac{\sigma_i - \sigma_{i-1}}{\Delta x} = \frac{\sigma(0, t)}{\delta} \\ \frac{\partial \sigma}{\partial x}(L, t) &= \frac{\sigma_{i+1} - \sigma_i}{\Delta x} = -G \end{aligned} \quad (4.2)$$

By isolating the ghost point term to one side of the equation, we can replace it in the original central difference equation (4.1) allowing us to eliminate the non-existent point while also introducing the boundary condition. Equation (4.3) demonstrates the new central difference equation when the ghost point is eliminated at the cathode.

$$\frac{\partial \sigma}{\partial t}(0, t) = \frac{\kappa}{\Delta x^2} (G_1 \Delta x - \sigma_i + \sigma_{i+1}) \quad (4.3)$$

Internal junctions require no ghost point replacement and instead use the fact that flux is continuous at wire junctions to introduce the boundary conditions as in (4.4).

$$\frac{\partial \sigma}{\partial t} \left(\frac{L}{2}, t \right) = \frac{\kappa}{\Delta x^2} (\sigma_{i-1} - 2\sigma_i + \sigma_{i+1} + (G_2 - G_1)) \quad (4.4)$$

In (4.4), G_1 and G_2 belong to the two respective wire segments that meet at the junction. Additionally, $\frac{L}{2}$ indicates that the example is a single wire with two segments where the junction boundary occurs at half the length of the whole wire.

As previously mentioned, we preserve the continuity of the time domain term which allows us to rewrite these equations as an ODE and LTI dynamic system. Using the previously derived equations (4.1), (4.2), and (4.4) for boundary and internal nodes, we can rewrite these equations into matrix format:

$$\begin{aligned} \begin{bmatrix} \dot{\sigma}_1 \\ \dot{\sigma}_2 \\ \dot{\sigma}_3 \\ \dot{\sigma}_4 \\ \dot{\sigma}_5 \end{bmatrix} &= \frac{\kappa}{\Delta x^2} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \\ &\times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \end{bmatrix} + \begin{bmatrix} \frac{\kappa\beta\rho}{\Delta x} & 0 \\ 0 & 0 \\ \frac{-2\kappa(\beta\rho)}{\Delta x} & \frac{2\kappa(\beta\rho)}{\Delta x} \\ 0 & 0 \\ 0 & -\frac{\kappa\beta\rho}{\Delta x} \end{bmatrix} \begin{bmatrix} j_1 \\ j_2 \end{bmatrix} \end{aligned} \quad (4.5)$$

where $\beta = \frac{eZ}{\Omega}$.

For the growth phase, the void is nucleated at the cathode node. Then, the resulting LTI system for the two-segment wire case becomes:

$$\begin{aligned}
\begin{bmatrix} \dot{\sigma}_1 \\ \dot{\sigma}_2 \\ \dot{\sigma}_3 \\ \dot{\sigma}_4 \\ \dot{\sigma}_5 \end{bmatrix} &= \frac{\kappa}{\Delta x^2} \begin{bmatrix} (-\frac{\Delta x}{\delta} - 1) & 1 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \\
&\times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -\frac{2\kappa(\beta\rho)}{\Delta x} & \frac{2\kappa(\beta\rho)}{\Delta x} \\ 0 & 0 \\ 0 & -\frac{\kappa\beta\rho}{\Delta x} \end{bmatrix} \begin{bmatrix} j_1 \\ j_2 \end{bmatrix}
\end{aligned} \tag{4.6}$$

As a result, in both the void nucleation and growth phases, we can write the LTI ODE for stress evolution in the following general form:

$$\mathbf{C}\dot{\sigma}(t) = \mathbf{A}\sigma(t) + \mathbf{B}\mathbf{j}(t), \tag{4.7}$$

$$\sigma(0) = [\sigma_1(0), \sigma_2(0), \dots, \sigma_n(0)]$$

In the case of (4.5), \mathbf{A} is the 5×5 coefficient matrix, \mathbf{C} is a 5×5 identity matrix, \mathbf{B} is 5×2 input matrix, and $\mathbf{j}(t)$ is the 2×1 column vector containing the current density of each wire segment for the respective time t .

We note that the presented example only requires equations for the one-dimensional case. However, to handle more general cases, these equations can simply be extended to the two-dimensional domain as shown in (4.8).

$$\frac{\partial \sigma}{\partial t}(x, y, t) = \kappa \frac{\sigma_{i+1,j} - 2\sigma_{i,j} + \sigma_{i-1,j}}{\Delta x^2} + \kappa \frac{\sigma_{i,j+1} - 2\sigma_{i,j} + \sigma_{i,j-1}}{\Delta y^2} \quad (4.8)$$

4.1.1 Steady-state analysis for nucleation phase

In this subsection, we show that the ODE for the nucleation phase which we derived from Korhonen's equation shown in (4.5) has the same steady-state stress result as the recently proposed voltage-based EM method in [74]. We demonstrate this using one simple example, a two-terminal wire as shown in Fig. 4.2. We let the total length be L . We

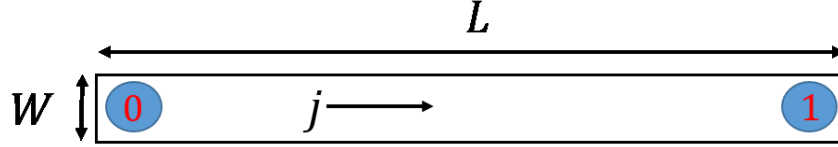


Figure 4.2: A two-terminal wire with the electron flow indicated by the arrow.

then use this wire segment length as the spatial step size and use the backward difference method shown in (4.2) for boundary derivation. The resulting system of equations for the two-terminal case is presented in equations (4.9).

$$\begin{bmatrix} \dot{\sigma}_0 \\ \dot{\sigma}_1 \end{bmatrix} = \frac{\kappa}{L^2} \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} \sigma_0 \\ \sigma_1 \end{bmatrix} + \begin{bmatrix} \frac{\kappa G}{L} \\ -\frac{\kappa G}{L} \end{bmatrix} \quad (4.9)$$

We then rewrite these equations into the following format:

$$\dot{\sigma}(t) = \mathbf{A}\sigma(t) + \mathbf{B} \quad (4.10)$$

$$y(t) = \mathbf{E}\sigma(t)$$

In (4.10), $\mathbf{E} = (1, 0)$, meaning we select node 0, the cathode, as the output node for which we are obtaining the steady-state stress. Then, a Laplace transform can be applied and the resulting transfer function becomes

$$\mathbf{F}(s) = \mathbf{E}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} \quad (4.11)$$

Then we go back to (4.9), the resulting transfer function for the single wire case becomes:

$$F(s) = \frac{\kappa GL}{(sL^2 + 2\kappa)} \quad (4.12)$$

Under step response, which is $1/s$ in frequency domain, we can then use the Final Value Theorem to obtain the stress at $t = \infty$, which is the steady-state result of the system under step response as:

$$\sigma_{steady} = \lim_{t \rightarrow \infty} f(t) = \lim_{s \rightarrow 0} sF(s) \frac{1}{s} = \frac{GL}{2} \quad (4.13)$$

We then can compute the steady-state stress based on the voltage-based method [74]:

$$\sigma_{steady} = V_E \frac{eZ}{\Omega} = \frac{jL\rho eZ}{2\Omega} = \frac{GL}{2} \quad (4.14)$$

where $V_E = jL\rho/2$ is the *EM voltage* at the cathode node (node 0) [74]. As we can see, the results from the two methods are identical.

In general, this is the case for general multi-segment interconnects and the steady-state EM stress can be computed by either method. Furthermore, the voltage-based EM method [74] can provide an important relationship for stress values at different nodes as shown in (4.15) in the next section.

4.2 The proposed Krylov fast EM stress analysis

In this section, we will present our new Krylov subspace-based fast EM stress analysis method. The following section contains several steps necessary for explanation of the proposed method and are outlined below:

- We first show that the linear time-invariant system describing the dynamic stress evolution must be pre-processed to handle the inherent singularity of the \mathbf{A} matrix.
- We then compare the steady-state response of the LTI system with a recently proposed steady-state method for EM stress evolution using frequency domain methods.
- Next, we show the proposed Krylov-based model reduction technique using a modified Arnoldi process.
- The temperature dependence of the EM effect and our method for handling time-varying temperatures is presented.
- Lastly, we outline our method for normalization of the results to maintain numeric stability during model reduction and simulation.

4.2.1 Singularity mitigation for EM ODE matrices

Before we introduce our Krylov subspace-based method, we notice that the EM matrix \mathbf{A} in (4.5) or in (4.8) in general is singular for our case. We notice that this is typically true for the nucleation phase; however, this will cause problems for the Krylov subspace-based method, which requires computing the inverse of \mathbf{A} to obtain the Krylov subspace. The reason is that the stress variables for the wire nodes are not independent as

there is no “ground” stress node. As a result, one more independent equation is required to make this matrix non-singular and we will show the mitigation method below. We note that this singularity issue has also been observed in [22].

To mitigate this problem, we need to introduce one more independent equation (to replace one dependent equation) into the stress LTI system (4.8).

It turns out that such an independent stress equation can be found in the dynamic and steady-state stress of the LTI system (4.8) as a result of mass conservation and stress-strain relationship, and it has been shown in [29, 74] that

$$\sum_k a_k \sigma_k = 0 \tag{4.15}$$

where a_k is the total area of branches connected to the node k . This equation represents the conservation in the stress kinetics. Equation (4.15) is independent of any rows in the A matrix. As an example, for the two-segment wire in Fig. 4.1 with same width for all the segments, we have

$$\sigma_1 + 2\sigma_2 + 2\sigma_3 + 2\sigma_4 + \sigma_5 = 0 \tag{4.16}$$

Therefore, we can use this equation to replace a dependent row (for example the middle row of the \mathbf{A} matrix). With the new row, the A matrix becomes an invertible matrix. As an example, the modified equation (4.17), where the second equation or row is replaced, is shown below:

$$\begin{aligned}
& \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\sigma}_1 \\ \dot{\sigma}_2 \\ \dot{\sigma}_3 \\ \dot{\sigma}_4 \\ \dot{\sigma}_5 \end{bmatrix} \\
&= \frac{\kappa}{\Delta x^2} \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 1 & 2 & 2 & 2 & 1 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \\
&\times \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \end{bmatrix} + \begin{bmatrix} \frac{\kappa\beta\rho}{\Delta x} & 0 \\ 0 & 0 \\ -\frac{2\kappa(\beta\rho)}{\Delta x} & \frac{2\kappa(\beta\rho)}{\Delta x} \\ 0 & 0 \\ 0 & -\frac{\kappa\beta\rho}{\Delta x} \end{bmatrix} \begin{bmatrix} j_1 \\ j_2 \end{bmatrix}
\end{aligned} \tag{4.17}$$

We notice that the \mathbf{A} for the growth phase is not singular any more and no mitigation is required for growth phase analysis.

4.2.2 Fast Krylov subspace-based stress analysis

In this subsection we present our Krylov subspace-based complexity reduction and simulation method, which is based on the the similar principles in the traditional model order reduction methods [65, 78]. After the stress evolution PDE has been discretized into

the ODE as shown in (4.1), (4.2), and (4.3), it can be written into the following linear time-invariant (LTI) dynamic system:

$$\mathbf{C}\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{j}(t), \quad (4.18)$$

$$\mathbf{x}(0) = [x_1(0), x_2(0), \dots, x_n(0)]$$

where the stress vector is represented by $\mathbf{x}(t)$, $\mathbf{x}(0)$ is the initial stress at $t = 0$ due to thermal-mechanical interaction. \mathbf{C} , \mathbf{A} are $n \times n$ matrices and \mathbf{B} is the $b \times p$ input matrix, where p is the number of inputs or the size of driving current density sources, $\mathbf{j}(t)$, which can be time-varying and is represented by the piecewise constant linear waveform as shown in Fig. 4.3. The piecewise constant linear input current density $\mathbf{j}(t)$ can be represented by

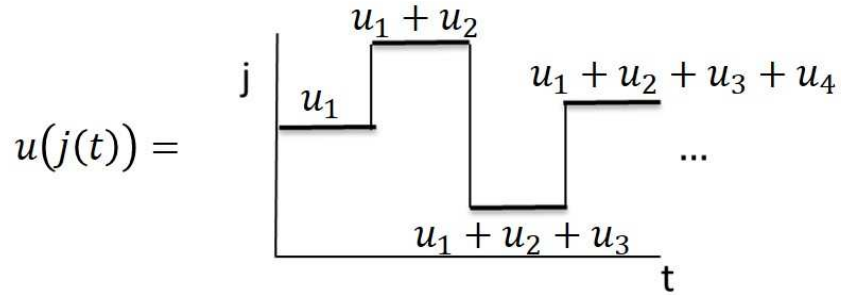


Figure 4.3: Example piecewise constant current density \mathbf{j} input as a function of time t .

$$\mathbf{u}(t) = \mathbf{u}_1(t) + \mathbf{u}_2(t - t_1) + \mathbf{u}_3(t - t_2) + \dots + \mathbf{u}_N(t - t_{N-1}) \quad (4.19)$$

We transform the problem domain into the frequency domain using the Laplace transformation of the state equation (4.19), which can be rewritten as

$$s\mathbf{C}\mathbf{X}(s) - \mathbf{C}\mathbf{x}(0) = \mathbf{A}\mathbf{X}(s) + \frac{1}{s}\mathbf{B}\mathbf{J}_1, \quad (4.20)$$

The Laplace transformation of $\mathbf{j}(t)$ is computed as

$$J(s) = \frac{1}{s} \left(\sum_{i=1}^N u_i e^{t_{i-1}} \right) = \frac{1}{s} J_1 \quad (4.21)$$

$$J_1 = \left(\sum_{i=1}^N u_i e^{t_{i-1}} \right). \quad (4.22)$$

It may be noted that in contrast to traditional model order reduction of the LTI systems, where the inputs are the impulse function and we perform reduction on the transfer functions [8], here the input is piecewise constant linear (or any arbitrary waveform represented by the piecewise linear function ¹). As a result, we have to consider the input signal subspace during the reduction process. Essentially the reduction process is no longer the traditional **model order** reduction, but is just the reduction step for a given signal input and is the pre-process step of the whole simulation. Notice that the extended Krylov subspace (EKS) method has been previously proposed for fast power grid network analysis [86]. In this paper, we follow a similar idea, but we use a simple Arnoldi-like orthonormalization process to compute the Krylov subspace of the response space instead of using the more complicated EKS method. Specifically, let $\tilde{\mathbf{X}}(s) = s\mathbf{X}(s)$, then the above equation becomes:

$$s\mathbf{C}\tilde{\mathbf{X}}(s) - s\mathbf{C}\mathbf{x}(0) = \mathbf{A}\tilde{\mathbf{X}}(s) + \mathbf{B}\mathbf{J}_1, \quad (4.23)$$

We then expand the $\tilde{\mathbf{X}}(s)$ using Taylor's series at $s = 0$, to get:

$$\begin{aligned} s\mathbf{C}(m_0 + m_1s + m_2s^2 + \dots) - s\mathbf{C}\mathbf{X}(0) \\ = \mathbf{A}(m_0 + m_1s + m_2s^2 + \dots) + \mathbf{B}\mathbf{J}_1 \end{aligned} \quad (4.24)$$

¹For the EM-induced stress analysis, piecewise constant linear current density input is sufficient as most of the power models of a real chip can be modeled as a piecewise constant linear waveform.

We then obtain the recursive response moment computation formula as follows:

$$\begin{aligned}
\mathbf{m}_0 &= -\mathbf{A}^{-1}\mathbf{B}\mathbf{J}_1 \\
\mathbf{m}_1 &= \mathbf{A}^{-1}\mathbf{C}(\mathbf{m}_0 - \mathbf{x}(0)) \\
\mathbf{m}_2 &= \mathbf{A}^{-1}\mathbf{C}\mathbf{m}_1 \\
&\vdots \\
\mathbf{m}_{q-1} &= \mathbf{A}^{-1}\mathbf{C}\mathbf{m}_{q-2}
\end{aligned} \tag{4.25}$$

For the Krylov subspace method, instead of computing the raw moments as shown in (4.25), a modified Arnoldi process is used to compute the orthonormalized response moment space. We let

$\mathbf{G} = \mathbf{A}^{-1}\mathbf{C}$ and $\mathbf{b} = -\mathbf{A}^{-1}\mathbf{B}\mathbf{J}_1$, and the modified Arnoldi process is shown in Algorithm 1.

We call this a modified Arnoldi process, as the computed space \mathbf{V}_q is not strictly a Krylov subspace which is defined as:

$$K_q(\mathbf{G}, \mathbf{b}) = \text{span}(\mathbf{b}, \mathbf{G}\mathbf{b}, \mathbf{G}\mathbf{b}^2, \dots, \mathbf{G}^{q-1}\mathbf{b}) \tag{4.26}$$

However, due to line 4 and lines 6-8 [8] our algorithm does not satisfy this. Instead, the subspace \mathbf{V}_q is the response signal space for $\tilde{\mathbf{X}}(s)$ as it considers the initial condition $\mathbf{x}(0)$ and the input signal vector $J(S)$. In this paper, we call it the *response Krylov subspace*. In our case, the signal space in the frequency domain only has the $\frac{1}{s}$ moment term, but in general, a piecewise linear input has non-zero component coefficients in all moments. Hence, lines 7 and 9 will be changed accordingly in this case. Due to the orthonormalization process

Algorithm 1 Modified Arnoldi method for orthonormalization of moment space.

```
1: Modified Arnoldi process ()

2: input: ( $\mathbf{G}, \mathbf{b}, \mathbf{x}(0), q$ )

3: output: ( $\mathbf{V}_q, \mathbf{H}_q$ )

4:  $\mathbf{v}_1 = \mathbf{b}/\|\mathbf{b}\|_2$ 

5: for ( $j = 1; j \leq q; j++$ ) do

6:   if ( $j == 1$ ) then

7:      $\mathbf{w} = \mathbf{G}(\mathbf{v}_j - \mathbf{x}(0))$ 

8:   else

9:      $\mathbf{w} = \mathbf{G}\mathbf{v}_j$ 

10:  end if

11:  for ( $i = 1; i \leq j - 1; i++$ ) do

12:     $h_{i,j} = \mathbf{w}^T \mathbf{v}_i$ 

13:     $\mathbf{w} = \mathbf{w} - h_{i,j} \mathbf{v}_i$ 

14:  end for

15:   $h_{j+1,j} = \|\mathbf{w}\|_2$ 

16:  if ( $h_{j+1,j} \neq 0$ ) then

17:     $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+1,j}$ 

18:  end if

19: end for

20:  $\mathbf{V}_q = [\mathbf{v}_1 \cdots \mathbf{v}_q]$ 

21:  $\mathbf{H}_q = (h_{i,j}), \quad i, j = 1, \dots, q$ 
```

in Algorithm 1, the moment computation process will become much more numerically stable than the raw moment computation as shown in (4.25).

Once we obtain the projection matrix \mathbf{V}_q , the original circuit matrix \mathbf{A} , \mathbf{C} , \mathbf{B} and initial stress $\mathbf{x}(0)$ in (4.19) can be order-reduced to the following matrices by the *congruence transformation*:

$$\hat{\mathbf{A}} = \mathbf{V}_q^T \mathbf{A} \mathbf{V}_q, \hat{\mathbf{C}} = \mathbf{V}_q^T \mathbf{C} \mathbf{V}_q, \hat{\mathbf{B}} = \mathbf{V}_q^T \mathbf{B}, \hat{\mathbf{x}}(0) = \mathbf{V}_q^T \mathbf{x}(0)$$

where $\hat{\mathbf{A}}$ and $\hat{\mathbf{C}}$ are $q \times q$ matrices and $\hat{\mathbf{B}}$ is the reduced $q \times p$ input matrix. $\hat{\mathbf{x}}(0)$ is reduced initial condition $q \times 1$ vector. Then the resulting reduced ODE LTI stress evolution system with the initial condition can be written as:

$$\hat{\mathbf{C}} \dot{\hat{\mathbf{x}}}(t) = \hat{\mathbf{A}} \hat{\mathbf{x}}(t) + \hat{\mathbf{B}} \mathbf{j}(t), \quad (4.27)$$

$$\hat{\mathbf{x}}(0) = [\hat{x}_1(0), \hat{x}_2(0), \dots, \hat{x}_q(0)]$$

Then transient simulations in the time domain using Back Euler time integration method can be performed on (4.28), which will be much more efficient to simulate than the original ODE LTI system in (4.19). After the reduced response is obtained $\hat{\mathbf{x}}(t)$, then the original response can be obtained by $\mathbf{x}(t) = \mathbf{V}_q \hat{\mathbf{x}}(t)$.

4.2.3 EM simulation under time-varying temperature

To accommodate the time-varying temperature impact on the stress evolution in the Korhonen equation (5.1), which can effectively accelerate or decelerate the stress build-up as shown in Fig. 4.4, we need to consider the time-varying diffusion parameter $\kappa(T)$ as it is a function of temperature T . As a result, the elements of matrix \mathbf{A} in the resulting

LTI system in (4.8) will be a function of the temperature. As a result, the response Krylov subspace computation has to be carried out for each different temperature value, which will not be a viable solution in our case.

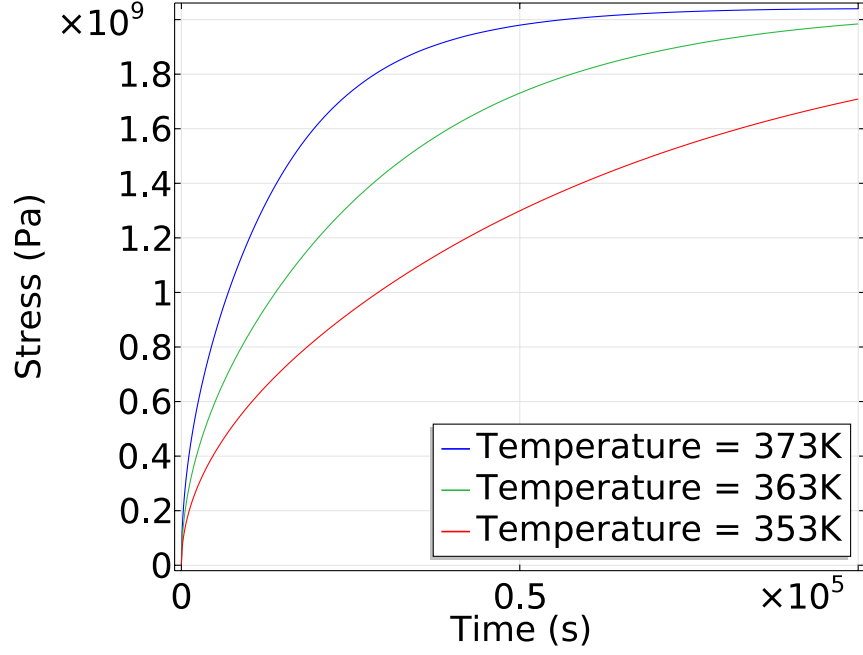


Figure 4.4: Cathode stress over time for multiple wire temperatures

Fortunately, there is a better way to deal with this situation. Specifically, Let $\sigma_i(x, t, \kappa(T_1))$ and $\sigma_i(x, t, \kappa(T_2))$ ($i = 1, 2$) be the solutions to the stress evolution equation (4.8) with the diffusivities $\kappa(T_1)$ and $\kappa(T_2)$, due to different temperature T_1 and T_2 , for the same initial and boundary conditions, respectively. Let Δt be the time period, then we can have following relationship [23, 55]:

$$\sigma_i(x, \Delta t, \kappa(T_2)) = \sigma_i(x, \frac{\kappa(T_2)}{\kappa(T_1)} \Delta t, \kappa(T_1)). \quad (4.28)$$

From (4.28), we can see that the temperature impact on the stress $\sigma(T, t)$ during the period Δt can be translated to the time period change for a metal wire. In other words, stress development in a metal wire over a period Δt under temperature T_2 will be equal to stress development for a metal wire over a period $\frac{\kappa(T_2)}{\kappa(T_1)}\Delta T$ under temperature T_1 , where $\Delta T = T_2 - T_1$. As a result, we convert the temperature-varying stress computation problem into a constant temperature problem.

Going back to our problem when we perform the time-domain stress simulation on the reduced model described in (4.28), a constant temperature T_0 is used for all the response Krylov subspace generation and reduction steps. Then during the simulation, we first create a virtual i th time step τ_i :

$$\tau_i = \sum_{i=1}^i \frac{\kappa(T_i)}{\kappa(T_0)} \Delta t_i \quad (4.29)$$

to perform the analysis under different temperatures T_i over the virtual time. However, the real time step for the simulated stress results are still $t_i = \sum_{i=1}^i \Delta t_i$.

4.2.4 Scaling schemes for numerical stability

One important implementation issue is that the parameters used in the Korhonen equation can differ by a few orders of magnitude. As a result, direct application of modified Krylov subspace-based method on the resulting PDE may not be stable. This problem can be mitigated by parameter scaling. After the scaling, calculated stress and time should be scaled back to obtain the real stress condition and time information. In the following, we discuss two scaling schemes:

Stress scaling scheme

For the stress scaling, steady-state stress is used as a reference. As mentioned in equation (4.14), the steady-state stress is proportional to GL . The equation $G = \frac{eZ\rho j}{\Omega}$, in eq. (5.1) is one of the dependents for the steady-state stress and is scaled to 1 when current density j is set to $5MA/cm^2$. In other words, $G_{scale} = j/j_0$ after scaling where $j_0 = 5MA/cm^2$ and we can call the corresponding G value G_0 when $G_{scale} = 1$.

The scaled length of the branch is given by $L_{scale} = L/L_0$ where L_0 is $100\mu m$ in our example case. After the scaled stress value σ_{scale} is calculated, the real stress $\sigma = \sigma_{scale}G_0L_0$.

Time scaling scheme

The time scale is dependent on $\frac{\kappa}{\Delta x^2}$ as shown in equation (4.1). That is, time is proportional to $\frac{\Delta x^2}{\kappa}$. Δx is one segment of the branch and is scaled using the length L_0 , which means $\Delta x = \Delta x_{scale}L_0$. Additionally, κ is scaled to 1 in our analysis, and the relationship between time and scaled time is $t = t_{scale}L_0^2/\kappa$.

4.3 Numerical results and discussions

We have implemented the proposed *FastEM* method in MATLAB and this section presents the numerical results for accuracy of the proposed fast Krylov subspace-based EM analysis method and speed-up over the finite difference time domain (FDTD) method, which was also implemented in MATLAB. The experiments were carried out on a Linux server with dual 3.3GHz Xeon processors and 316GB memory, with each processor having

2x22 cores (44 threads each). We want to stress that the both *FastEM* and FDTD were implemented in MATLAB. As a result, their performance comparison based on the same numerical package and computing server is fair.

4.3.1 Accuracy study

In our numerical analysis, a netlist for the interconnect tree is first discretized to form the LTI ODE system which is then used as input for the solver. In addition to constant current density, we also show the results from the piecewise constant current density inputs as well as simulation under dynamic temperature variations. To validate our proposed method, the two-segment case is simulated with the *FastEM* method and compared to the Finite Element Analysis (FEA) and the FDTD method proposed in [26]. This two-wire structure contains two $50\mu\text{m}$ wire segments connected in series as presented in section 2. Each simulation is conducted using 7 poles for reduction ($q = 7$). All FEA simulations are performed using COMSOL Multiphysics software using 2D structures.

In Fig. 4.5 the nucleation stage stress under asymmetrical current density distribution ($1E10A/m^2$ in segment 1 and $8E10A/m^2$ in segment 2) for three time steps is shown. The results show agreement between all three methods with less than 0.02% error. We use the stress results from the nucleation phase to generate a critical stress profile (the stress in the wire when $\sigma(0, t) \leq 500\text{MPa}$) which is then used as the initial condition for the growth phase simulation while also applying the same current density distribution. Validation of the three methods in the growth phase is shown in Fig. 4.6 for three time steps. Again, the methods agree almost perfectly with negligible errors.

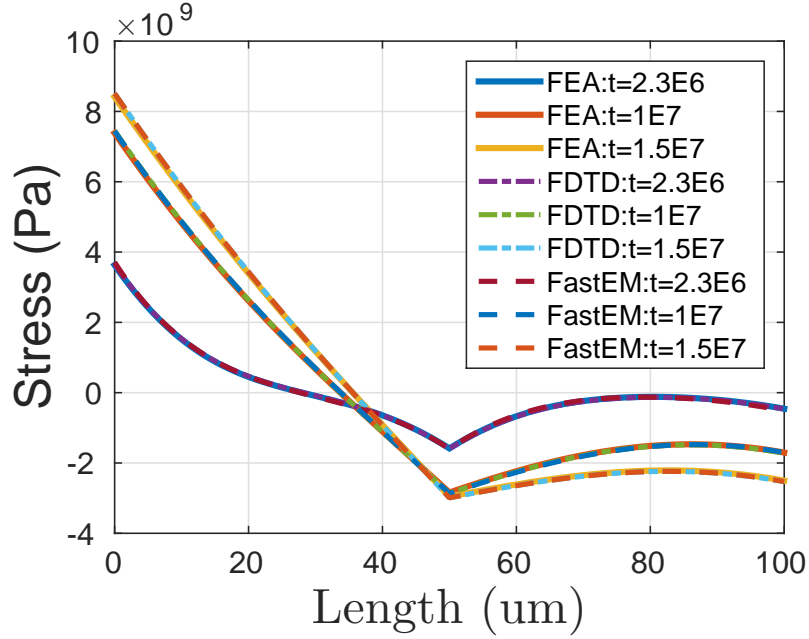


Figure 4.5: Nucleation stage validation for two wire segments. *FastEM* solved using $q = 7$ with 0.0155% average error.

The results for both the nucleation and growth phases show good agreement with the other methods under constant asymmetric current density conditions, which validates our *FastEM* method. To validate the time-dependent current density handling, a periodic piecewise constant current density input, oscillating from $1E10A/m^2$ to $0A/m^2$, is applied and cathode stress during the nucleation phase is compared for the three methods and presented in Fig.4.7. The three methods again agree with negligible error. Simulation results are also compared for the temperature-dependent modeling. Results show that both FDTD and *FastEM* agree with the COMSOL result using different temperatures during the simulation and are presented in Fig. 4.8.

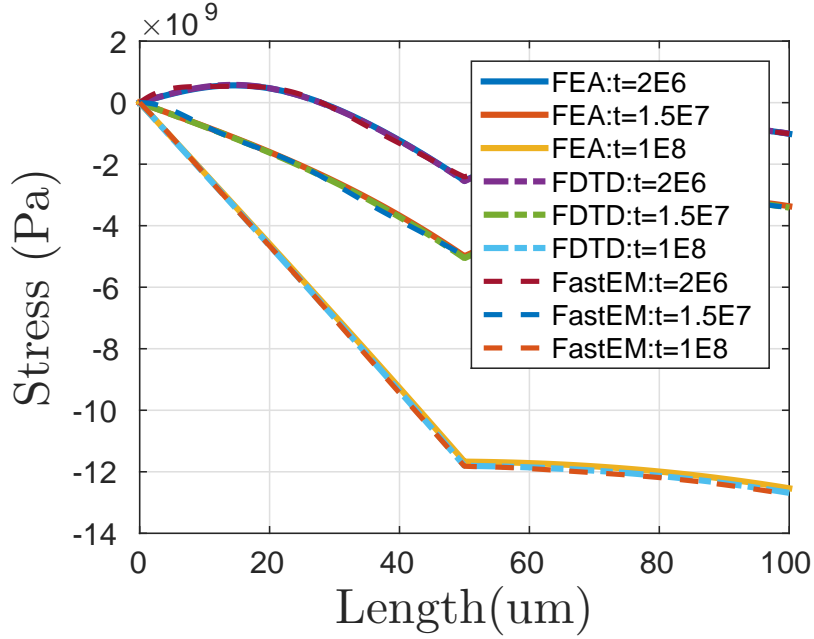


Figure 4.6: Growth stage validation for two wire segments. *FastEM* solved using $q = 7$ with 0.0123% average error.

4.3.2 Eigenvalue analysis

The modified Krylov subspace-based method relies on the computation of several Eigenvalues for reduction. To better understand how many poles q need to be computed in the *FastEM* method, we compute and plot the Eigenvalues in Fig.4.9 and Fig.4.10.

In Fig.4.9 we can see that the first Eigenvalue, the smallest pole of our system, determines the time constant of the system, and thus the trend of the stress development. Additionally, we see that only the first few Eigenvalues, and thus poles, have an effect on the system. Fig.4.10 omits the first two Eigenvalues allowing us to better see the values that follow.

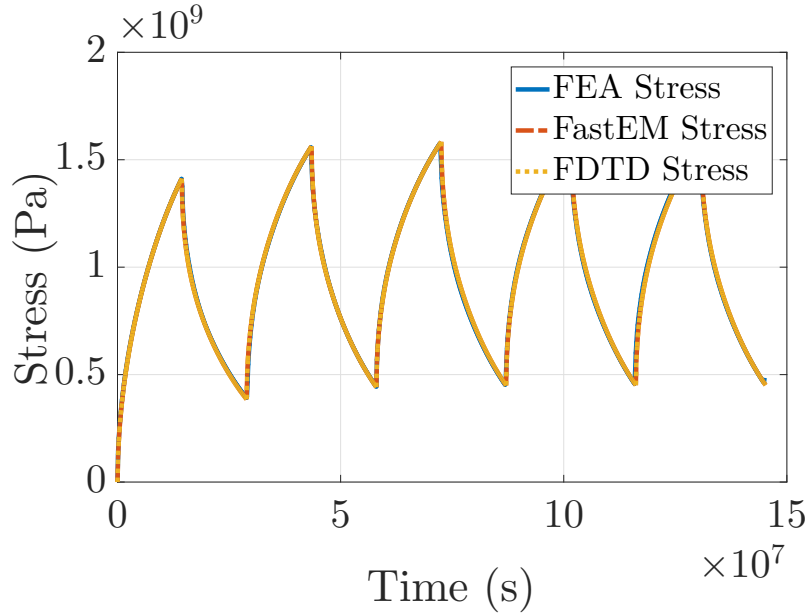


Figure 4.7: Cathode stress comparison under piecewise constant current density input.

Additionally, the driving current and the structure of the tree being analyzed will also affect the number of poles needed. In this work, we use an ad-hoc method of determining the poles required by seeing how many dominant poles exist in our system. This type of analysis would need to be performed for each structure under test, however; we find that no more than 17 poles were ever needed for the trees that we tested in this work.

For demonstration, we show the stress simulation for a large 174-segment tree with non-uniform current density distribution in Fig.4.11, using different numbers of computed poles. Using 7 and 9 poles proves inaccurate as the large number of different current densities and segments require more poles for accurate simulation. However, the use of 11

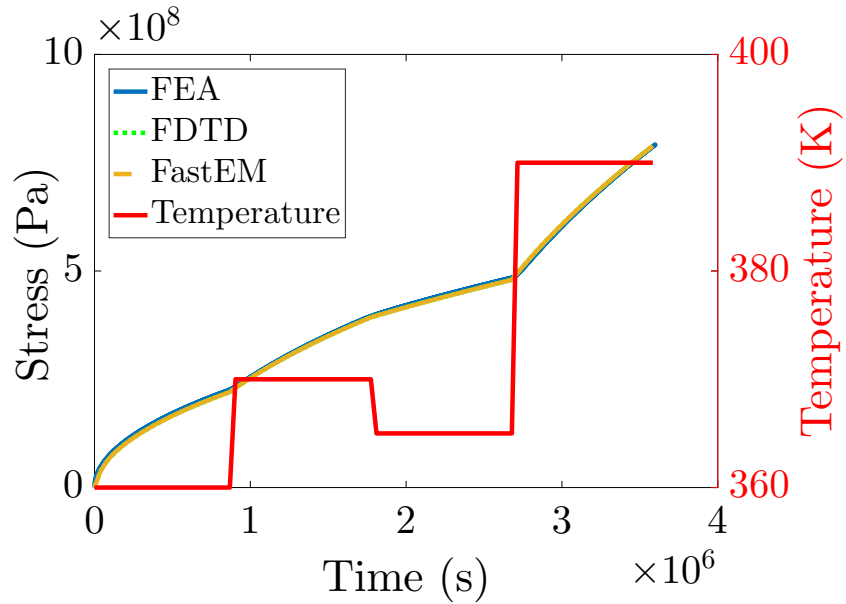


Figure 4.8: Cathode stress under varying temperature $T= 360\text{K}, 370\text{K}, 365\text{K}, 390\text{K}$.

Table 4.1: Single-threaded performance comparison between FDTD and *FastEM* on interconnect trees from the IBM power grid benchmark *ibmpg2*

IBM Tree	FDTD (seconds)	<i>FastEM</i> (seconds)			Speed-up
		MOR	BE	Total	
tree1	16.985	0.1562	0.0312	0.1890	81x
tree2	63.727	0.6664	0.0345	0.7009	78x
tree3	185.224	1.8681	0.1960	1.9117	86x

and 17 poles leads to accurate results, with the 17 computed poles having less than 1% error.

To demonstrate the performance of our proposed method, we use both a real power grid benchmark and also arbitrary n-segmented trees. The IBM power grid benchmark *ibmpg3* structure [60] is firstly used for a realistic simulation using three different

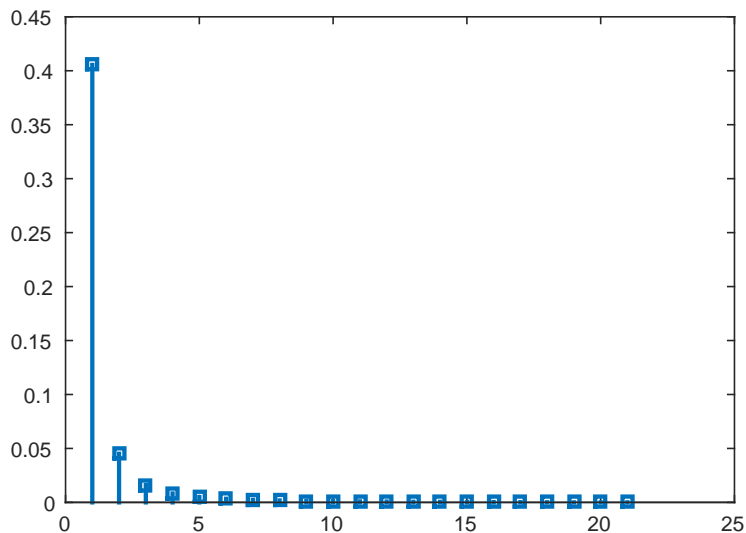


Figure 4.9: Eigenvalue plot showing all eigenvalues.

Table 4.2: Multi-threaded performance comparison between FDTD and *FastEM* on interconnect trees from the IBM power grid benchmark ibmpg2

IBM Tree	FDTD (seconds)	<i>FastEM</i> (seconds)			Speed-up
		MOR	BE	Total	
tree1	14.670	0.1530	0.1809	0.1809	90x
tree2	53.914	0.6512	0.0337	0.6849	90x
tree3	150.963	1.7437	0.0416	1.7575	97x

trees extracted from the ibmpg3 benchmark. Secondly, we perform FDTD and *FastEM* simulations using increasingly large n-segmented trees in order to demonstrate the scalability of the our method.

For the power grid simulation, we choose a small tree (tree 1) with 58 segments, a medium-sized tree (tree 2) with 109 segments, and the largest tree (tree 3) with 174 segments for the simulation structures. Both methods discretize each branch in the trees into 21 nodes and compute the stress for 1000 time steps. The proposed method used 17 poles to compute each simulation. We perform the simulation with all 88 threads available

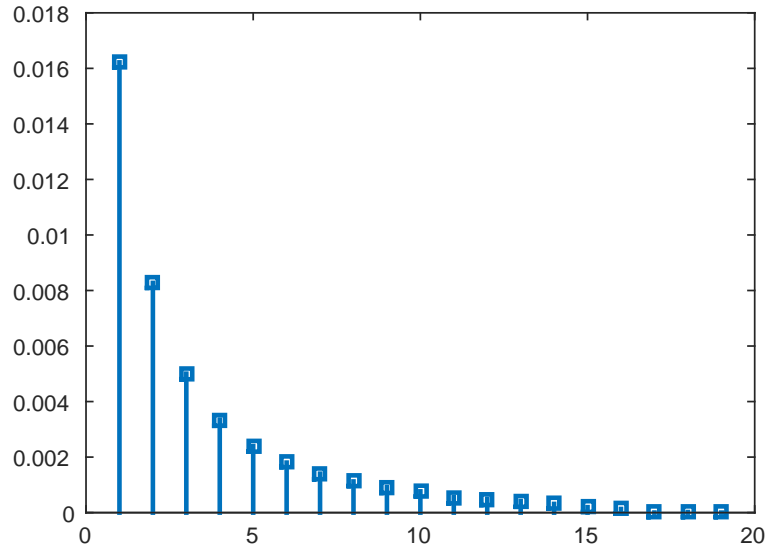


Figure 4.10: Eigenvalue plot without the two largest values.

and also while enforcing single-threaded computation. We do this to provide a better idea of the performance the simulation can achieve on various platforms as multi-threaded computation performance will be system-specific.

The results in Table 4.1 and Table 4.2 show the average computation time of the two methods using the three different *ibmpg3* trees. The results show significant speed up of over 90X and reaching close to 100X. We can also see from this table that the *FastEM* method is broken down into the time for the model order reduction (MOR) algorithm, the Backward-Euler solver (BE), and the total time.

The results show that the MOR portion of the proposed method dominates the computation time while the Backward-Euler solver takes much less time. We remark that MOR is dependent on the size of the input system and the number of poles that used for model reduction while BE is dependent on both. However, because we perform MOR, the

Table 4.3: Scalability performance results comparing FDTD and *FastEM* using increasingly large n-segment trees.

n-segments	FDTD (seconds)	<i>FastEM</i> (seconds)			Speed-up
		MOR	BE	Total	
20	6.26	0.0354	0.0309	0.663	94.41x
50	8.613	0.0820	0.0233	0.1053	81.79x
100	37.445	0.4058	0.0203	0.4261	87.87x
200	205.504	2.5983	0.0335	2.6318	78.08x
400	1519.715	20.2112	0.0407	20.2519	75.04x
500	2376.478	34.0633	0.0481	34.1114	69.66x
750	7609.128	107.6036	0.0501	107.6537	70.68x
1000	15354	252.8880	0.0958	252.9838	60.69x
1250	28840	460.5587	0.1167	460.6754	62.60x
1500	52096	791.4289	0.1372	791.5661	65.81x
1750	80542	1240.455	0.1451	1240.455	64.92x
2000	110810	1815.1380	0.1624	1815.3	61.04x
3000	363660	6023.0290	0.1711	6022.3	60.37x
4000	8.61E5	14236.81	0.01876	14237	60.45x
5000	1.7339E6	27729.8	0.1951	27730	62.52x

input to BE will always be a matrix of $q \times q$ where q is the number of poles. Since we have set the number of poles to 17 in this test, the BE computation time will remain about constant with only negligible increases in duration due to congruence transformations at the beginning and the end of BE. Furthermore, we compute mean errors of **0.0678%**, **0.0251%**, and **0.0362%** for test trees 1, 2, and 3 respectively. As a result, the error caused by the reduction is basically negligible when compared to the FDTD method.

In addition to the benchmark test, performance scalability is studied by simulating increasingly larger n-segmented trees. Each tree is a straight wire, with n segments. Each segment is discretized into 11 nodes, and *FastEM* is computed with 17 poles. We can see from Table 4.3 that *FastEM* maintains the performance improvement even with very large systems achieving a 60-94X performance increase over FDTD. We also observe that as the number of segments increases, performance gain converges to around 60X.

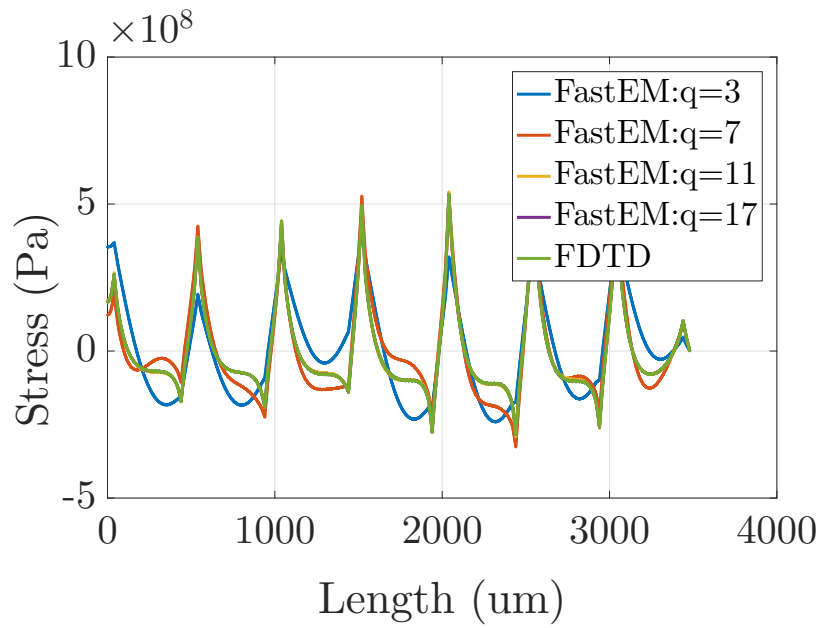


Figure 4.11: Stress near nucleation time for 174-segment tree with different numbers of computed poles.

Chapter 5

Electromigration Based Hardware Trojans Design

In recent years the concern over Hardware Trojans has come to the forefront of hardware security research as these types of attacks pose a real and dangerous threat to both commercial and mission-critical systems. One interesting threat model utilizes semiconductor physics, specifically aging effects such as Electromigration (EM). However, existing methods for EM-based Trojans rely on empirical Black's models can easily lead to performance degradation and less accuracy in Trojan activation time prediction. This chapter presents EM-based Trojan attacks based on recently developed physics-based EM models and the previously presented simulation techniques. This includes a novel EM attack techniques in which the EM-induced hydrostatic stress increase in a wire is caused by wire structure or layer changes without changing the current density of the wires. The proposed techniques consist of sink/reservoir insertion or sizing and layer switching techniques based

on the early and late failure modes of EM wear-out effects. As a result, the presented techniques can have minimal impact on circuit performance, which is in contrast with existing current-density-based EM attacks. The proposed techniques can serve as a trigger for the EM attack on power/ground networks and signal and clock networks. Furthermore, two potential EM attack mitigation techniques, namely, split fabrication and burn-in testing are discussed.

5.1 Three-phase EM model

Building upon the presented EM physics and models from chapter2, a more accurate three phase EM model has been recently developed [77]. In the new model, we have three phases including (1) the *nucleation phase* from $t = 0$ to t_{nuc} ; (2) the *incubation phase* from t_{nuc} to t_i ; and (3) the *growth phase* starting from t_i to t_{50} . The term t_{50} indicates the time-to-failure in statistical terms (50% of the samples fail). This model was later extended to consider more general multi-segment interconnect structures [75]. The following is a brief summary of this EM model.

In the **nucleation phase**, the void is formed at nucleation time (t_{nuc}). Hydrostatic stress increases from the initial value to critical level. In order to model that, a more complete physics based modeling of transient hydrostatic stress evolution was proposed by Korhonen [51]. We illustrate the equation in the one dimensional case for the sake of presentation. Then stress $\sigma(x, t)$ is described by Korhonen's partial differential equation (PDE) with liner(Ta) blocked boundary conditions (BC):

$$\begin{aligned}
PDE: \frac{\partial \sigma}{\partial t} &= \frac{\partial}{\partial x} \left[\kappa \left(\frac{\partial \sigma}{\partial x} + G \right) \right] \\
BC: \frac{\partial \sigma}{\partial x}(0, t) &= G, \quad \frac{\partial \sigma}{\partial x}(L, t) = -G
\end{aligned} \tag{5.1}$$

where, $\kappa = D_a B \Omega / kT$, B is the effective bulk elasticity modulus, Ω is the atomic lattice volume, $G = \frac{eZ\rho j}{\Omega}$ is the EM driving force, where e is the electron charge, eZ is the effective charge of the migrating atoms, ρ is the wire electrical resistivity. If the stress calculated by that model saturates before the critical level, the wire will never fail. Otherwise, the time in which stress reaches the critical level is t_{nuc} .

In the **incubation phase**, which is defined by the time period t_{nuc} to t_i , the void is nucleated, but its size is not significant. Hence the change in wire resistance will be very small and can be neglected. The incubation time ($t_i - t_{nuc}$) can be estimated as:

$$t_i - t_{nuc} = \frac{\Delta L_{crit}}{v_d} \tag{5.2}$$

Here ΔL_{crit} is the length of critical void size and v_d is the void's growth rate. For a single segment wire, v_d is expressed as a function of atomic flux J , $v_d = \Omega J$ [76], where Ω is atomic volume. Atomic flux, $J = \frac{D_a f}{\Omega kT}$, is the number of atoms crossing a unit area per unit time. Thus, the atoms crossing per unit length can be expressed as JW , where f is electron wind force per atom: $f = eZ\rho j$.

For a multi-segment tree, all segments that share a terminal with the void can contribute to its growth. Electron wind at each segment can accelerate or slow down the void growth based on their direction. Hence, the total atom flux can be expressed as a combination of all the fluxes on the segments. For multi-segment wires, the effective atomic flux per unit length $v_d W_m$ is the void's growth rate on the main segment. This is expressed

as:

$$v_d = \Omega J_m^* = \Omega \frac{1}{W_m} \sum_i J_i W_i = \frac{D_a e Z \rho}{k T W_m} \sum_i j_i W_i \quad (5.3)$$

Here j_i and W_i are the current density and width of the i th segment. W_m is the width of the main segment where the void is formed and J_m is the total flux contributing to the void. Here, we use $J_m^* = \frac{1}{W_m} \sum_i J_i W_i$ to compute the effective atomic flux J_m on the main segment. Note that if we only have one segment, then $v_d = \frac{D_a e Z \rho j}{k T}$ as shown in [39].

If the void volume saturates before it reaches the critical length, the wire will never fail. The **Incubation phase** ends when the void reaches L_{crit} at t_i . If the wire is a via-above wire, after the via is blocked by the void the current flow will also be blocked since the capping layer is fabricated with dielectrics such as Si3N4. This is referred to as early-failure and results in an immediate critical failure of the wire. However, if the wire is a via-below wire, the current flow can still pass but resistance of the wire will increase because current has to go through the liner which has much higher resistivity. This is referred to as late-failure. These concepts are explained in more detail further in the article.

Finally, in the **growth phase**, defined by time period from t_i to t_{50} the wire resistance starts increasing. Note that this phase is only possible in a via-below configuration and is unique to late-failure. After the via is blocked by the void, current is forced to flow through the liner. Since this liner is very thin, and its resistivity is much larger than copper, the current density and resistance on the liner will be very high. Resistance change can be expressed as [42]:

$$t - t_i = \frac{\Delta R(t)}{v_d \left[\frac{\rho_{Ta}}{h_{Ta}(2H+W)} - \frac{\rho_{Cu}}{HW} \right]} \quad (5.4)$$

where ρ_{Ta} and ρ_{Cu} are the resistances of the liner material (Ta for instance) and copper respectively, W is the line width of the segment where void is formed (main segment), H is the copper thickness, and h_{Ta} is the liner layer thickness.

A critical observation is that commercial tools do not utilize the Korhonen model or the three-phased model but instead still rely on the conservative Black's model. Using the simulation methods presented in this dissertation in conjunction with the three-phased Korhonen based method of modeling the EM-failure process is much more accurate. As such, commercial tools would be insufficient for the level of analysis we require to perform the attacks in this paper. Further, this three-phase model allows us to make use of the complex EM failure process to engineer specific wire topologies that can be leveraged as a reliability based attack while the currently available commercial tools could not consider these effects.

5.2 EM-based hardware attack modeling

Reliability-based attacks are made possible due to the vulnerabilities in the design and manufacturing process of modern IC's as depicted in Fig. 5.1.

Once the design house sends the final physical design to the third party foundry, masks are created from the design which the lithography tools use to fabricate the chip. An attacker at a foundry could modify these masks, without the design house knowing, and compromise the chip. :w

The challenge for an attacker creating an EM-based Trojan is to design a wire with structure and configuration such that the wire fails at a desired time and accomplishes some

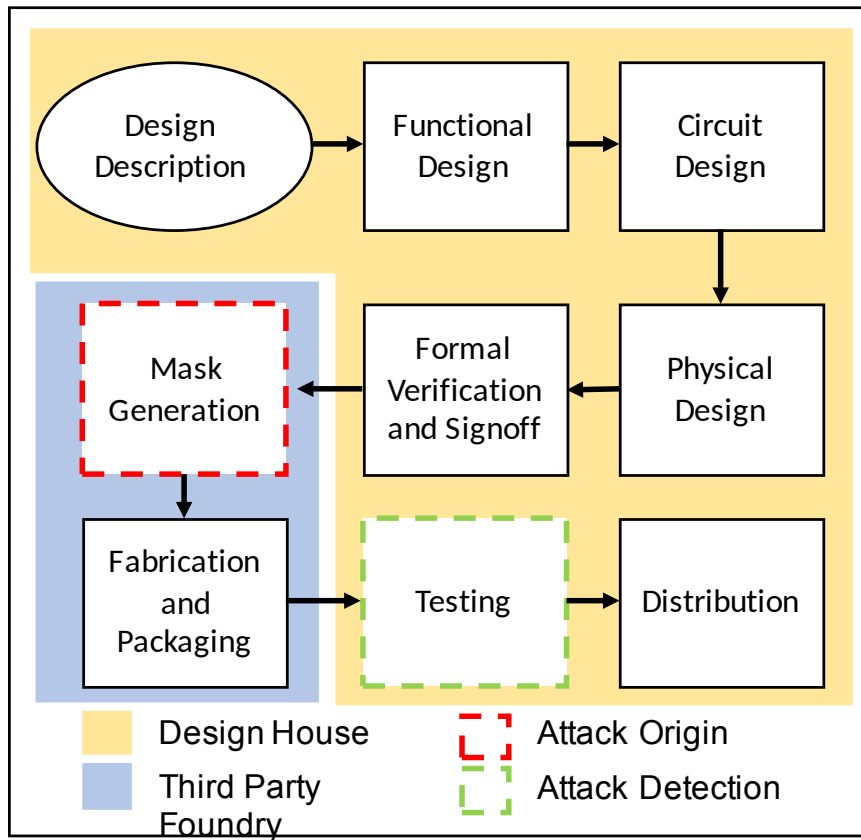


Figure 5.1: IC design and manufacturing flow showing attack and detection opportunities

malicious task without compromising the circuit performances and other design constraints prior to activation. Furthermore, with respect to the particular attacks presented in this article, an attacker must identify wires that are susceptible to these attacks. Primarily this requires an attacker to find wires with sufficient chip area around it or excess metal for modification and then find attack parameters that achieve the target life-time reduction. While automation tools could help in this regard, a determined attacker should not be too tightly constrained that they could not find candidate wires simply using visual inspection

of the mask or layout in combination with commercially available physical design tools and published EM models such as the one we employ in this work.

In the following sections we outline some primary challenges to EM-based attack design, attack opportunities based on newly proposed physics-based EM models, and the newly proposed attacks.

5.2.1 Challenges in EM-based attacks

In order for a wire to fail due to EM, it must be stressed by electrical current. An attacker must increase the EM stress conditions so that the EM-induced lifetime of the wire will be reduced. Furthermore, an EM Trojan must have minimal impact on circuit performance to maintain its stealthiness and effectiveness. For these reasons the stress source in a wire must be considered, as well as the method an attack uses to induce failure.

The wire's stress source, wire current, is a major contributor to its EM vulnerability. Furthermore, it is known that there exists a stress relaxation effect in a wire that becomes unstressed [41]. If the stress source is not considered, a wire may never generate enough stress to result in void nucleation, or the TTF of the wire may be much larger than estimated. When designing an EM attack, there are three primary stress sources: power/ground networks (p/g), clock trees, and signal nets. P/G networks have strong unidirectional currents giving them a good stress profile. Clock trees, while periodic, have high enough frequency ensuring long term averaging current providing a good stress source. Signal nets that are highly active are good stress sources but other nets, that have little activity, may not carry enough current to induce EM failure.

Additionally, simple alterations to a wire, such as altering its width to increase current density, can have unintended consequences on the wire's IR drop. In the case of p/g network wires, this can cause switching speed degradation to front end devices, thus affecting chip timing. This has two major consequences. Firstly, it can render the chip immediately inoperable. Secondly, it can cause enough change in performance that the EM Trojan is detectable through side-channel analysis. For this reason, novel techniques of inducing EM failure without degrading chip performance is required for effective EM attacks.

5.2.2 Electromigration topology effects

Multi-mode failure

EM induced atom migration results in parametric failure, e.g., causes resistance to change. However, depending on the wire topology, a wire may gradually experience resistance change once a void is nucleated (late failure), or the wire may immediately experience drastic resistance change causing an open circuit once a void has grown to a certain size (Early Failure) [39, 91].

Late failure typically occurs in a so-called via-below (or up-stream) structure when electron flow is from a lower layer of metalization to a higher level of metalization. In this case the void will form in the upper portion of the wire which will allow current flow for some time as shown in Fig. 5.2(a). Even after the void has saturated, current can still flow through the Ta barrier layer, albeit, with much higher resistance. This results in a gradual parametric failure. In contrast, early failure occurs in the via-above (or down-

stream) structure, where electron flow is from a higher layer of metalization to a lower level of metalization as shown in Fig. 5.2(b). In this case, the void will form in the upper part of the wire at the via interface. This void quickly grows to the diameter of the via, blocking current flow. Current cannot continue to flow as the only remaining path is the wire capping layer which is typically a dielectric such as Si_3N_4 and does not shunt the current flow. This causes immediate resistance change and effectively an open circuit.

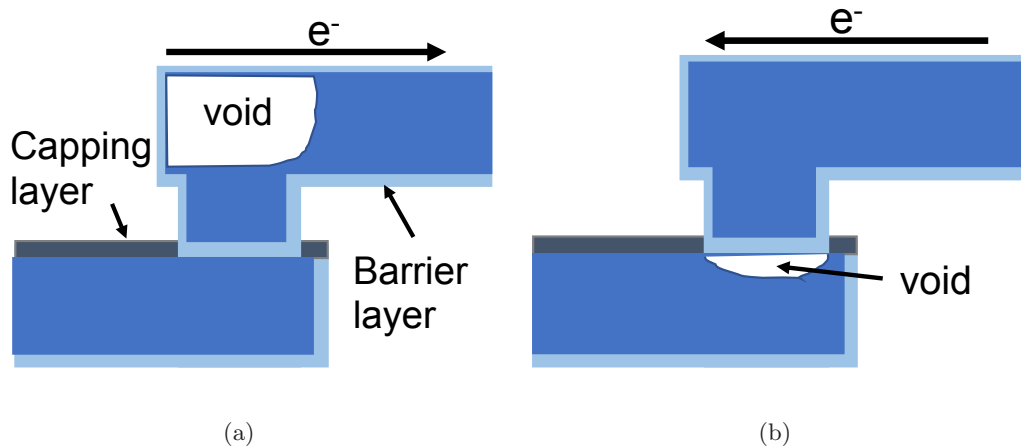


Figure 5.2: Via-below (a) and Via-above (b) wire structures showing void formation locations.

Multi-segment wires

Electromigration sign-off typically considers only single wire segments individually, however, the stress in neighboring wires can effect each other. Because of this, the Korhonen model has been expanded to handle these multi-segment interconnect trees. Depending on the wire topology and current flow in neighboring segments, the stress can vary drastically in the wire under test.

To illustrate this point, we consider a simple two-segment wire as shown in Fig. 5.3. We compare the TTF results using equal unidirectional current against equal opposing currents.

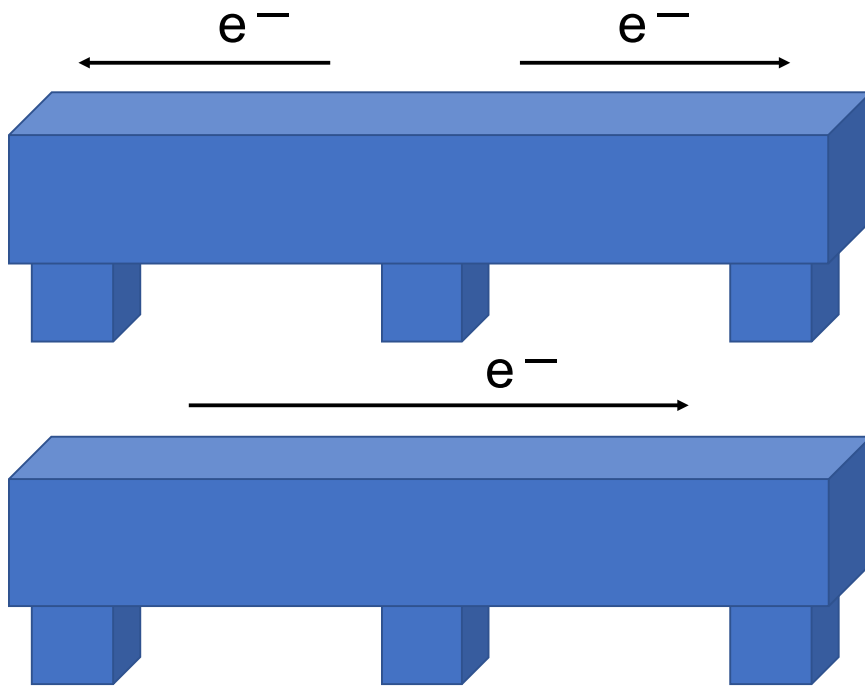


Figure 5.3: A two segment wire structure with the same current density (below) and different current densities (above)

Simulation results for these two structures show that the case with opposing currents has a TTF = 7.03 years and the case with a single unidirectional current has a TTF = 4.92 years which is quite a large difference.

Specific multi-segment configurations act as atomic reservoirs or sinks. These configurations have previously been observed to have effects on the TTF of a wire when either passive (having no current) or active (carrying current) [54]. The reservoir is situated

at the cathode end of a target wire and when passive, can extend the TTF of a wire. Consequently, reduction or removal of this reservoir can reduce the TTF of a wire. The sink is attached to the anode of a wire. When the sink is passive, the stress is increased in the cathode end of the wire which decreases the TTF of the overall wire.

Power/ground network redundancy

The p/g network is seemingly the best place to insert a Trojan due to its high constant unidirectional current flow which provides a great stressing source for an EM Trojan. At first inspection, it is also enticing to simply modify an existing wire in the p/g network to induce premature failure of the entire chip. However, this is also not as simple as it seems due to the inherent redundancy of the p/g network.

Redundancy in the p/g network comes from its mesh structure [42]. Wire failure may only cause minor IR degradation in the network and there may still exist other paths allowing current to reach the front-end devices.

This is both an advantage and disadvantage depending on the attacker's objective. If attempting to render the chip inoperable through an attack on the p/g network, an attacker must determine which wire(s) will result in enough IR degradation to meet this objective which is not trivial. However, if an attacker wishes to utilize the p/g network to stress an EM-based Trojan which has some other effect other than disabling the chip, then the redundancy works to the attackers' advantage. This is because the wire failure in the network can achieve the attacker's objective without compromising the power integrity of the chip.

5.3 EM attack methods

With the proper modeling, simulation, and design challenges in mind, we can formulate specific attacks using the EM wear-out effect. As mentioned previously, a naive method of attack relies on Black’s equation where current density is the only parameter available to implement an attack. Practically, this means an attacker can simply decrease wire widths. While this can be an effective method, it has the drawbacks of affecting other circuit performance parameters, e.g., wire delay and IR drop. However, in this work we utilize the wire structure impacts on EM as presented in 5.2.2. In the following presented attacks IR drop and delay are not affected because, as we will present, we only need modify non-current carrying metal to create the attack.

In the following sections, to demonstrate our attacks, we generated EM resilient wire configurations. Then, we ran several simulations, using Finite Element Analysis and the three-phase model, on the target wire while sweeping attack parameters to find effective attack formulations.

5.3.1 EM as a Trojan payload

As a payload, the EM-based attack results in performance or functionality degradation upon wire failure. This can be accomplished by modifying an existing wire to cause wire failure earlier than anticipated by the designers. An EM payload can be used to cause IR degradation in the p/g network, disrupt the functionality of the clock tree, or even disable highly active signal nets.

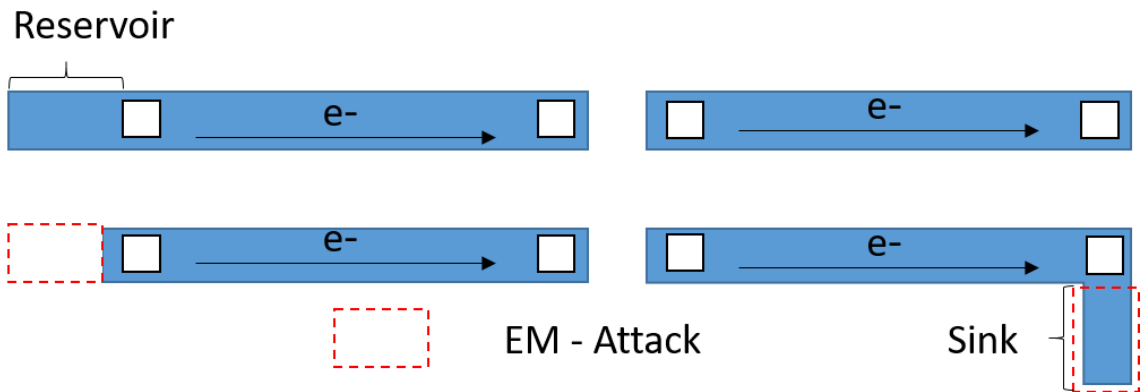


Figure 5.4: Illustrations of the reservoir reduction and sink insertion attacks

Reservoir reduction p/g network attack

As discussed in 5.2.2, a passive reservoir structure in a multi-segment wire can help increase the TTF of a wire. In practice, passive reservoirs are a common occurrence in the p/g network. Often it is the case that large reservoirs are added for reliability reasons, primarily in the p/g network, or simply as a consequence of power grid synthesis that results in excess metal. Thus, an effective and stealthy attack would be to reduce or remove the reservoirs from the p/g network of a chip. Because they are passive, their removal will not cause IR degradation.

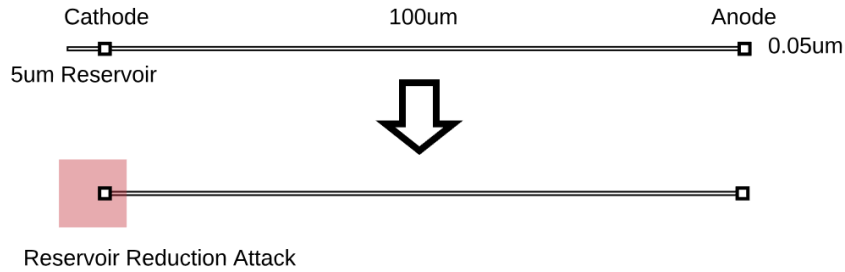


Figure 5.5: An example reservoir reduction attack

To demonstrate the reservoir reduction attack, shown in Fig. 5.4, we design an immortal wire (meaning it will never fail due to EM) with a passive reservoir and show the attack in Fig. 5.5. The wire is $0.05\mu m \times 100\mu m$ with a reservoir size of $0.05\mu m \times 5\mu m$. After removing the reservoir, the initially immortal wire has a TTF of 5.197 years, rendering the wire quite vulnerable to EM aging.

Sink insertion attack

Many wires in a chip will not have passive reservoirs already attached to them, typically these will be clock tree and signal nets. In these cases, a reservoir reduction cannot be attempted as reservoirs will likely be active and their removal will immediately cause chip failure at worst or performance degradation at best.

To target these wires, we can use a sink insertion attack, depicted in Fig. 5.4. As mentioned in 5.2.2, a passive sink added to a target wire will reduce its TTF. This type of attack is ideal for causing a target wire to fail when a passive reservoir is not present. Furthermore, like the previously mentioned reservoir reduction attack, this small addition

will not have any large effect on the IR drop of the net since we are only adding passive metal to the wire.

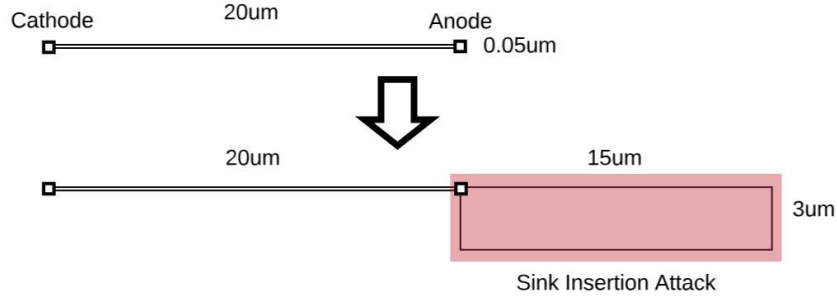


Figure 5.6: An example sink insertion attack

As a demonstration, we consider an immortal wire, shown in Fig.5.6, with periodic current density similar to that of a wire in a clock tree. We then insert a passive sink to the wire and observe the effects on TTF. It should be noted that for this simulation, we model the current density as the average current density due to the periodicity. It has been shown that for high frequency periodic signals, like we would find in a clock tree, the long term averaging effects mean that using the average current density is adequate [40]. The initially immortal wire is $0.05\mu m \times 20\mu m$ and the inserted passive sink is $3\mu m \times 15\mu m$. After inserting the sink, the initially immortal wire has a TTF of 0.7253 years, a drastic reduction in the TTF.

We note that in this particular example, the size of the reservoirs required can be quite large compared to the target wire. However, the area taken up by the reservoir is relatively small compared to the entire chip. Still, an attacker could be constrained by high density routing. In this case, care must be taken to find appropriate target wires to attack

where room is available for a sink insertion. Additionally, sinks need not be rectangular but can take on all matter of shapes and sizes so long as the sink area is sufficient for the attack. Furthermore, it is not uncommon for large areas of a metal layer to be unoccupied requiring the insertion of passive dummy filler metal to maintain structural stability in the die during fabrication [70]. This provides an excellent opportunity for an attacker to attach the dummy metal to the anode of a wire, thereby creating a large passive reservoir without adding large amounts of metal.

Layer demotion attack

In 5.2.2, it was shown that depending on the wire positioning, either up-stream or down-stream, a wire can experience the Early or Late failure effects. While this is something we can leverage in any EM attack, it can also be used as an attack by itself while also maintaining all the advantages of the topological attacks presented previously.

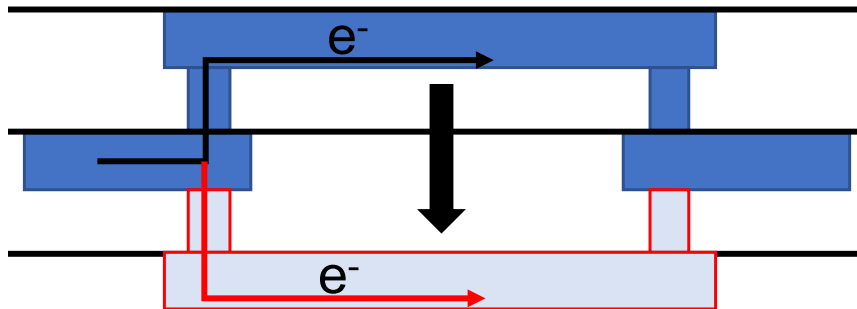


Figure 5.7: The originally up-stream wire is moved to a lower level of metalization to put it in the down-stream configuration in the layer demotion attack

In this attack, a mortal wire that is normally positioned in the up-stream configuration, may have an acceptable TTF. However, if the wire were to be in the down-stream

configuration, the Early failure mode would result in much more rapid failure. To achieve this, we can perform a layer demotion attack on an up-stream configured wire by moving the wire to a lower level of metalization than the wire its cathode is attached to. This will maintain the same electrical paths and IR drop of the circuit but will cause the wire to be in the down-stream configuration, and thus, experience Early failure.

To demonstrate this attack, we identify a mortal wire in the up-stream configuration with reasonably high TTF. In this case the wire has an initial TTF of 6.69 years. However, after reconfiguring the wire to a down-stream configuration, the TTF falls to 3.94 years.

5.3.2 EM as a Trojan trigger

In some cases, it may be desirable to activate a Trojan that does not render the chip inoperable. In this case, the challenge for an attacker is to embed a trigger for their payload in the chip that is difficult to activate or detect by the design house. EM-based Trojans offer a stealthy and lightweight option to triggering a Trojan payload due to their inherent stealthiness.

An EM-based trigger can utilize any of the modeling and attack techniques previously mentioned but are configured in such a way that their failure activates some other Trojan payload. In this case, it is best to use an early failure configured wire that, when activated, will quickly redirect current to (or from) the Trojan payload. We propose to use the EM-trigger to control current flow such that during the aging process, the functional

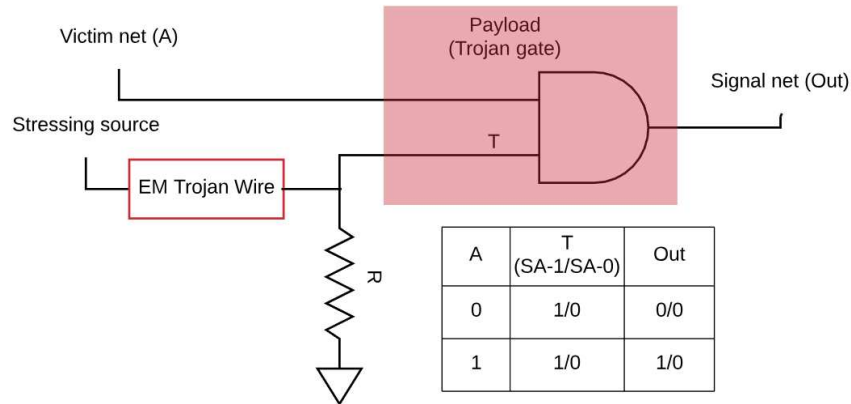


Figure 5.8: Example circuit of an EM Trojan wire being used as a trigger for a Trojan gate payload

behavior of the circuit is unchanged. However, after activation, when the Trojan wire fails, the current flow will be redirected such that the Trojan payload becomes activated.

In another example, shown in Fig. 5.8, an EM-based trigger wire is used in conjunction with pass logic transistors to attack a victim gate. In this case, the pass logic transistors allow for normal functional operation of the circuit, so long as voltage is sufficient at either input to the transistors. Prior to activation, the victim NAND gate will operate normally with inputs “A” and “B”. However, after the EM Trojan wire fails, the pass logic transistors will no longer allow the “B” net to pass and will create a SA-0 fault at the “B” input of the victim net. At this point, the victim gate’s output “Out” will always retain a high logic level.

The only drawback of using the EM-based Trojan wire as a trigger is that it likely requires the introduction of more circuitry to ensure the wire is stressed. That is, the wire needs a sufficient stressing source which requires the introduction of circuitry to create current flow which will have some effect on chip power consumption. This is because the

current generated on signal nets may not be sufficient to stress an EM-Trojan wire. In the above examples we utilized a resistor connecting to a reference which will complete a circuit and allow constant current flow through the EM Trojan wire while ensuring we avoid shorting the circuit all together. However, this resistor could potentially be any circuit or structure that ensures the Trojan wire is stressed. Furthermore, making use of the structural attacks presented earlier can help mitigate the added power consumption as current draw may not have to be as significant compared to a purely current density based EM Trojan design. Lastly, other methods could be utilized to stress the wire that would not result in significant additional power consumption, e.g., the current from the charging and discharging of gate capacitance generated by a periodic signal such as a clock net could be sufficient to stress the Trojan wire.

5.4 Mitigation techniques for EM-based Trojans

EM-based Trojans pose a real threat due to the difficulty in detecting them through conventional testing methodologies. However, there are measures that chip designers can take to mitigate the possibility of an EM-based attack and to also enhance traditional testing methodologies to increase their chance of detecting these types of attacks.

5.4.1 Split-fabrication

As discussed previously, the primary concern over hardware Trojan attacks stems from the decentralization of the design and fabrication of ICs, due to increasing costs associated with manufacturing these devices, thus making them vulnerable to attack by third

party fabs. The primary reason for the rising costs of manufacturing has to do with the highly advanced technology nodes, specifically the front end devices of the IC. However, the process for manufacturing the back-end-of-line interconnects is relatively unchanged from previous technology nodes. Split-fabrication has been proposed in several works [45, 46, 83] to take advantage of this fact and effectively makes the IC design process immune to EM-based attacks.

Split-fabrication separates the front-end and back-end-of-line manufacturing which provides several advantages with respect to the integrity of the fabrication process. The idea is to allow the third party fabs, which have invested in the advanced fabs with the tools and processes to manufacture at advanced technology nodes, continue to manufacture the front-end devices while the design house or other trusted manufacturer can finish the IC's back-end with a relatively less expensive fab.

This methodology provides a few advantages. Firstly, the design house no longer needs to provide detailed design files that reveal the actual architecture of their designs. This makes the insertion of hardware Trojans that target the chip logic extremely difficult while also protecting the design house's IP. Secondly, this methodology allows the design house to visually inspect the front-end of the chip before the back-end is manufactured, thus allowing the detection of any Trojan logic gates. Lastly, with respect to EM-based attacks, this methodology would be particularly effective. EM-based attacks target the back-end-of-line interconnects, however, if the design house is using a split-fabrication process, then any untrusted fab could not insert this type of attack.

While not yet adopted by industry, real ICs have been manufactured using split-fabrication by researchers [83] showing its feasibility.

5.4.2 Burn-in testing

EM-based attacks are particularly difficult to detect due to their passive behavior prior to activation due to aging. While test vectors and side-channel analysis will fail to detect these types of attacks on their own, they could be used for detecting an EM-based Trojan if the Trojan was forced to activate during test time.

Burn-in testing is already a common methodology for ensuring an IC or PCB is free of defects or excessive process variation. These tests subject a chip to high stress conditions, outside of the normal use conditions, designed to cause failure in chips that do not meet reliability specifications. After testing, the chips that pass are considered reliable and can be introduced into the market where the expectation is that they will be used in the normal use condition which is far less extreme than the high stressing conditions they have been subjected to in the burn-in testing. Additionally, this type of testing is often employed by fabs to judge the EM resilience of a particular fabrication process by subjecting wires with varying dimensions and current densities to high stress until they fail. These results are then extrapolated to real world use cases to determine design rules for EM sign-off.

By subjecting a chip to burn-in testing, an EM Trojan, which is designed to fail early already, can be subjected to high stress conditions which will accelerate their aging. Coupling this with traditional logic based testing and side channel analysis, EM-Trojans which are forced to fail early during this process can be detected. Not all EM-Trojans will be detected in this manner however.

This technique will only affect Trojans with aggressive failure targets. However, this effectively reduces the design margin for the attacker by limiting the aging range that can be selected for the EM-attack. In other words, an attacker cannot create a Trojan that will fail very early without risking its activation during a burn-in test. This may be enough to limit an attacker to such an extent that an EM-based Trojan may never even be activated, even if inserted and not detected.

The burn-in technique is illustrated in Fig. 5.9. We simulate the EM-induced TTF for a wire under high stressing conditions of 390K with a voltage scaled to +5% and assume a burn-in duration of one week. Trojan wires with failure times below the 7 day burn-in duration under these conditions would likely have been activated at test time and detected.

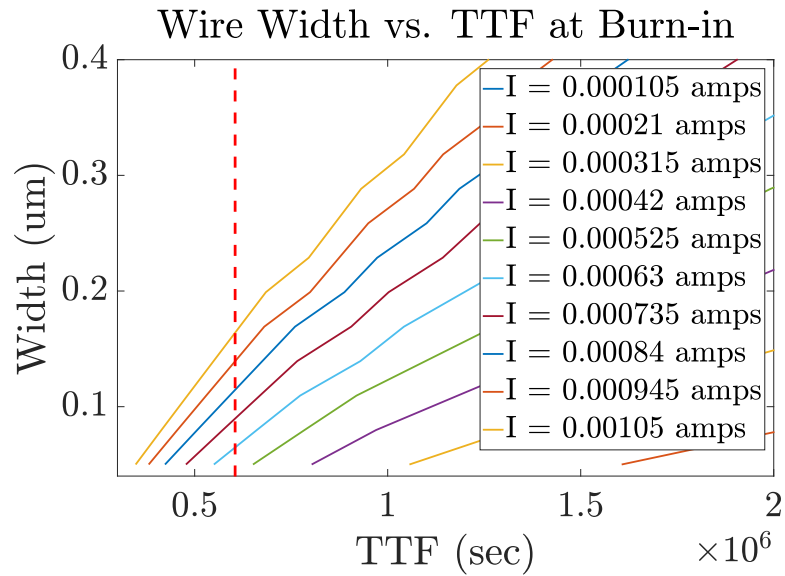


Figure 5.9: Burn-in testing reduces the range of failure times available to an attacker by inducing failure in aggressive TTF targets.

5.5 Conclusion

In this chapter, the recently proposed advanced EM modeling and simulation techniques were utilized to formulate novel reliability-based Trojan payloads. Two topology-based EM attacks were presented that leverage the multi-segment stressing dynamics from atomic sinks and reservoirs. Also presented is a payload that exploits multi-mode failure mechanisms, early and late failure, by converting a wire from the up-stream configuration to the down-stream configuration. Furthermore, we proposed an EM-based Trojan triggering mechanism for stealthy time-delayed activation of hardware Trojans. These Trojans utilize the topology and structure of wires which gives them an advantage over previously proposed current density based EM-Trojans which can affect circuit performance. Finally, a discussion is presented for two potential EM attack mitigation techniques including split fabrication and burn-in testing.

Chapter 6

On-Chip Counterfeit IC Detection

Using Electromigration Aging

Sensor

The counterfeiting of Integrated Circuits (ICs), especially recycled ICs, has become increasingly problematic for the electronics industry due to its financial impact and threat to the security of mission critical electronic systems. One viable way to detect counterfeit ICs is by means of on-chip aging sensors by leveraging natural aging and degradation processes such as Negative Bias Temperature Instability (NBTI) in devices and electromigration (EM) in interconnects. In this chapter, a novel EM based hardware aging sensor is presented to precisely record the aging process of a chip. Compared to existing EM-based aging sensors, the proposed aging sensor consists of three major improvements. First, the EM-aging sensor uses post-voiding resistance changes and subsequent signal delay as the measurement of

aging process instead of using the nucleation effects, which are difficult to detect and more susceptible to variations. As a result, the new design does not need many redundant wires to reduce variation. Second, a novel configurable multi-segment wire structure is utilized to significantly reduce the void formation time while preserving long void growth time for wire resistance change in the normal chip working conditions. The new sensor can perform a novel self-calibration process during fabrication, which can significantly improve the prediction accuracy of the aging sensor. Third, the new EM sensor measures the RC delay changes of the stressed wires instead of the resistance itself as in the existing EM aging sensor. This new method leads to a simplified hardware design. As resistance change has a linear dependency on stressing time, such a sensor design can give the continuous aging time estimations after the sensor is self-calibrated, which is not possible in existing EM-aging sensor designs. Lastly, the new sensor design makes use of advanced EM topology effects, as presented in 5.1, to accelerate the nucleation of a void to facilitate the self calibration prior to IC distribution. Experimental results show the proposed EM-aging sensors are more accurate and more area efficient at nearly $10\times$ smaller than previously proposed methods.

6.1 Existing on-chip aging sensor design

Many previous works have been proposed for lightweight aging detection sensor designs [37, 38, 84, 92, 93]. The method in [93] designed a Ring-Oscillator(RO)-based aging sensor that relies on the aging effects of MOSFETs to change an RO frequency in comparison with a reference frequency embedded in the chip. As the chip ages, due to the wear-out mechanisms such as Negative-Bias Temperature Instability (NBTI) and Hot Carrier

Injection (HCI), the threshold voltage of the MOSFET devices begins to shift, while also changing the frequency of the RO, and provides a simple indicator for the IC age. However, this method can only give a very rough estimation of the usage age of the chip as the shift in frequency depends on many factors.

In order to mitigate the inaccuracy problem, an antifuse (AF)-based sensor was developed in [80]. The AF-based sensor essentially is a counter, which counts the clocks or derivatives of the clock events to log the usage of the chip. The antifuse memory is used to make sure the data in the count will not be erased or altered by attackers. However, AF-based sensors suffer from large area overhead, especially when a more accurate indication of usage is required [80]. Another problem with this method is that it may not reflect the true aging-dependent usage of a chip. For instance, it will log the same usage time for different on-chip temperatures, however, the temperature has been shown to have a dramatic impact on the aging effects from electromigration, NBTI and HCI [47].

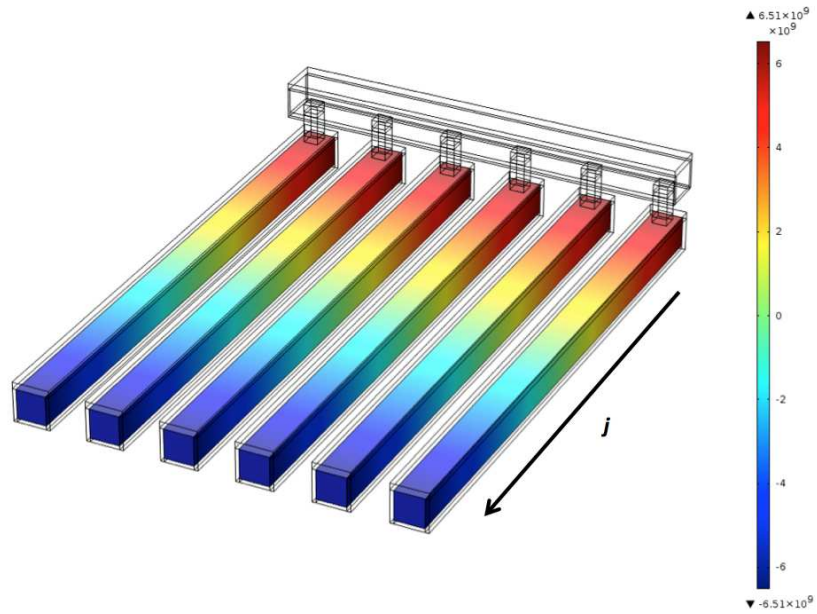
Recently He *et al.* proposed to use the electromigration (EM) aging effect in metal interconnects to create aging sensors that indicate when a certain lifetime has been reached on the chip [37, 38]. The sensor works by designing a copper wire to fail at a specified time, due to the EM effect, which then acts as an indicator that the specified age has been exceeded. We show in Fig. 6.1(b) the schematic of the EM-based aging sensor circuit from [38]. The sensor circuit utilizes a set of EM stressed wires (consisting of 10-20 wires [38]) which are identical and are designed to fail at the desired time. An unstressed reference wire provides the reference voltage which will be compared to the voltage of the aged wire set. The comparison of the voltage between the EM stressed wire set and the

reference wire is performed by a 1-bit ADC whose output is connected to a multiplexer which in turn outputs the status of the sensor when the input *Read_en* is set. If the sensor has been aged long enough, then the voltage across the EM stressed wires will have changed enough to cause the ADC to output a 1 instead of 0, indicating the specified age was reached.

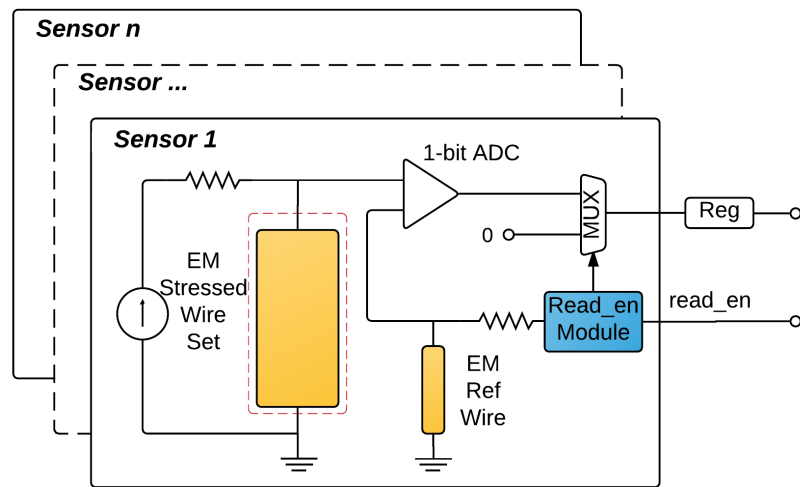
However this method also is not without its shortcomings. While able to accurately indicate a specific time of failure, this method relies on the voltage differential between a failed wire and a reference wire. Due to the small voltages and resistance inherent in modern ICs, this means a high accuracy ADC must be utilized to detect small differences in voltages requiring large area usage. Furthermore, to account for inherent variation in failure time, multiple redundant wires are used in parallel, as shown in Fig. 6.1(a), to increase accuracy of the sensor. Lastly, this sensor must be repeated to detect more than one failure time. For example, if the aging sensor needs to detect failure at 5 years and 7 years, then two separate aging modules are required as shown in Fig. 6.1(b), which can lead to large area overhead.

6.2 EM aging sensor based on configurable multi-segment wire structure

In this work we propose to create a self calibrating EM-based aging sensor module to indicate the age of an IC by designing a wire that will experience resistance degradation over a long period of time. A consumer, or the chip vendor can check the status of this module to determine if a chip has been used, and to what extent it has been used. This



(a) The existing multi-wire structure for the aging sensor and its stressed condition



(b) The schematic of the existing EM-based aging sensor

Figure 6.1: The existing EM-based aging sensor design [37, 38]

will assist in the identification of counterfeit, recycled and remarked ICs. This method can be employed quickly and on large batches of chips as only a single signal must be

checked rather than visually inspecting sample chips. In the following sections, we present the architecture of the proposed EM-based aging sensor and the associated models and methodologies. Parameters used are consistent with a 90nm process technology.

To facilitate the implementation of the aging sensor, we propose to utilize a pre-distribution calibration methodology. We make use of the fact that void growth, and thus resistance change, contains a linear region prior to its saturation. Because of this, by determining the resistance of the aged wire, we can calculate its age as long as we know the initial age and the resistance change rate.

The methodology to calibrate the aging module, shown in Fig. 6.2, contains several steps. Firstly, the chip, and its module are subjected to burn-in testing, a normal step in IC sorting during the IC fabrication process. This burn-in testing subjects the chip to high temperatures and voltages which will cause the stress generation in wires due to the EM effect to accelerate. We design the aging module such that it is more susceptible to void nucleation during this phase using topological modification as presented in [68] to accelerate this aging even further. Once burn-in is complete and the void has nucleated in the aging module wire, we measure its initial resistance R_1 (measurement of this resistance is explained in a later section). We then subject the wire to normal operating conditions for a certain amount of time (e.g. 1 week) and then measure the resulting resistance R_2 and calculate the resistance change ΔR . The values R_1 and ΔR are then stored on-chip in the aging module using an anti-fuse memory which is a one time writable memory difficult to tamper with. The host chip is then ready for distribution.

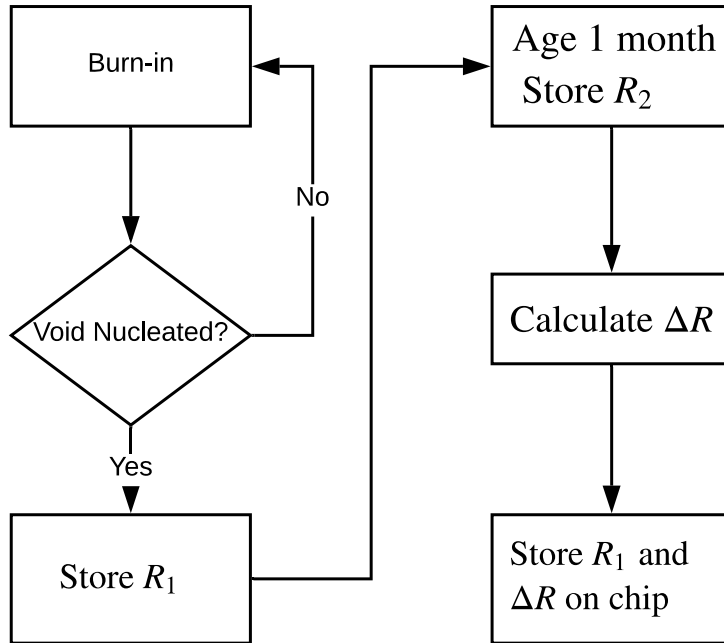


Figure 6.2: The EM aging sensor self calibration methodology

6.2.1 Chip authentication methodology

After the IC has been calibrated at the foundry, it is ready to be introduced into the electronics supply chain 6.3. Vendors, OEMs, consumers and suppliers can all check the status of the aging module to validate that the IC is indeed a new chip and not recycled.

As the chip is used, the resistance will continue to increase due to the void growth mechanics and the continued stressing. At any point in its life cycle R_{em} can be measured, and using the stored values of the initial resistance and the resistance growth rate, the age of the host chip t_{age} can be calculated. This allows the determination of not only if a chip has been used previously, but also to what extent the chip has been aged. If the chip is used

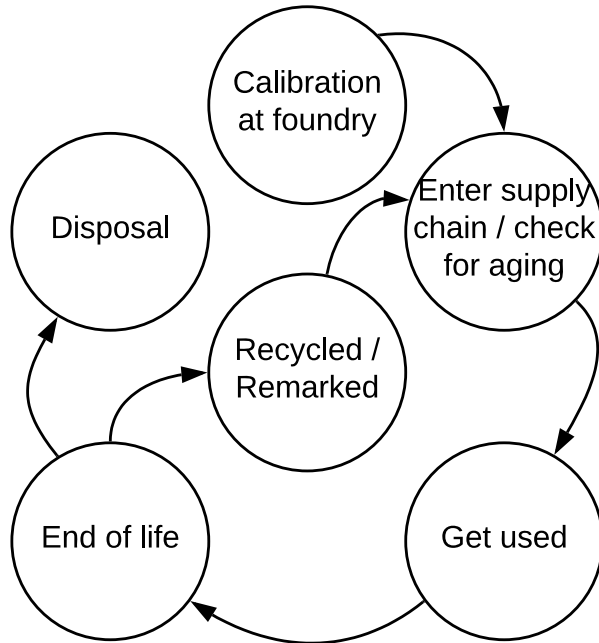


Figure 6.3: The IC supply chain life cycle depicting the pre-distribution self calibration stage, potential counterfeiting, and authentication during the IC distribution.

and then recycled and reintroduced into the electronics supply chain then the counterfeit part will show signs of aging determinable at the output of the aging module.

$$t_{age} = \frac{R_{em} - R_1}{\Delta R} \quad (6.1)$$

6.2.2 Age sensing by wire delay estimation

Unlike ideal wires, a wire in an IC can be modeled as an RC component, and that each wire will have some inherent delay associated with it. This is due to the natural resistivity of the metal wire itself and also the parasitic capacitance inherently present in ICs due to the close proximity of charged metal wires.

The voltage $V(t)$ of the RC model is given in (6.2). In this equation, V_0 is the voltage of the input signal and τ is the RC time constant given by the product of the resistor value R and the capacitor value C . From this equation, we can see that if R increases then so does τ which effectively increases the time it takes for $V(t)$ to reach the input voltage of V_0 . This shows that EM degradation not only effects the resistance of a wire but also has a delay degradation associate with it as well.

$$V(t) = V_0(1 - e^{-\frac{t}{\tau}}) \quad (6.2)$$

In Fig 6.4, we show the SPICE transient simulation of a step response in an RC modeled wire. In this simulation we show the results for a 30Ω wire and sweep the resistance to 40Ω , which is consistent with parametric failure of a wire due to EM, showing the effect that resistance degradation has on the wire delay. Specifically, it increases the rise time of the input signal.

6.2.3 New configurable hybrid multi-segment wire design

A critical part of this work is the ability to create a wire that can quickly nucleate a void but will have a long growth phase with reasonably large resistance change over time.

To facilitate this, we propose to utilize advanced EM models [77] and EM acceleration techniques [68]. As demonstrated in [68], we can create a configurable multi-segment wire that can vastly accelerate the void nucleation in a wire as shown in Fig. 6.5. Additionally, we can alter the operating mode of this wire such that the accelerated effects are disabled, allowing for non-accelerated void growth behavior needed for the age measurement

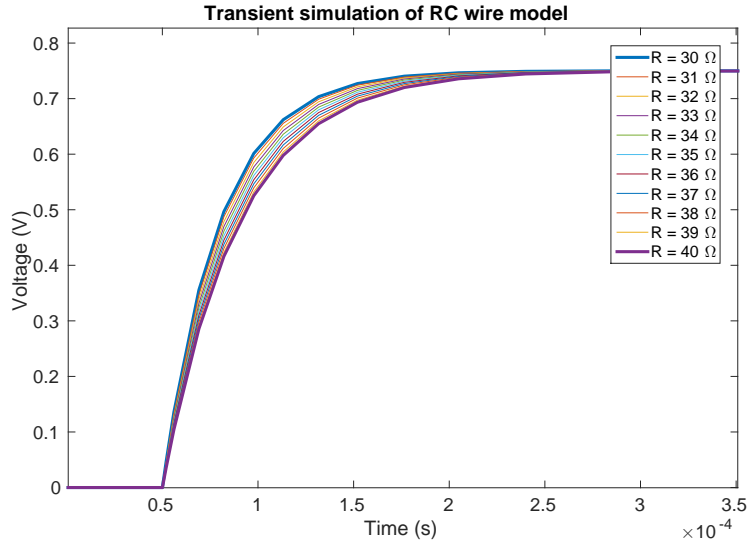


Figure 6.4: Transient SPICE simulation of the RC model while sweeping the resistance value showing the gradual degradation of rise time, and thus signal delay

in this work. Essentially, the acceleration of the EM effect in this wire works by creating a large stress gradient at the wire cathode by enabling current in both the sink and the two reservoirs in this structure. By disabling these currents, the EM effect is returned to normal.

In this work, we choose wire geometry parameters consistent with a wire that would be found in an IC power delivery network for $90nm$ technology nodes. We then can sweep these parameters in EM simulation to find appropriate wire, sink, and reservoir geometries to meet the needs of the design. To accomplish this, we use a Finite Element Analysis solver to solve the equation in (5.1) and determine the void nucleation and growth kinetics of these wires.

To ensure that the wire has a significantly long enough growth phase with measurable resistance change during normal aging operation, We can apply the following equation formulated in [42]:

$$t - t_i = \frac{\Delta R(t)}{v_d \left[\frac{\rho T_a}{h T_a (2H+W)} - \frac{\rho C_u}{HW} \right]} \quad (6.3)$$

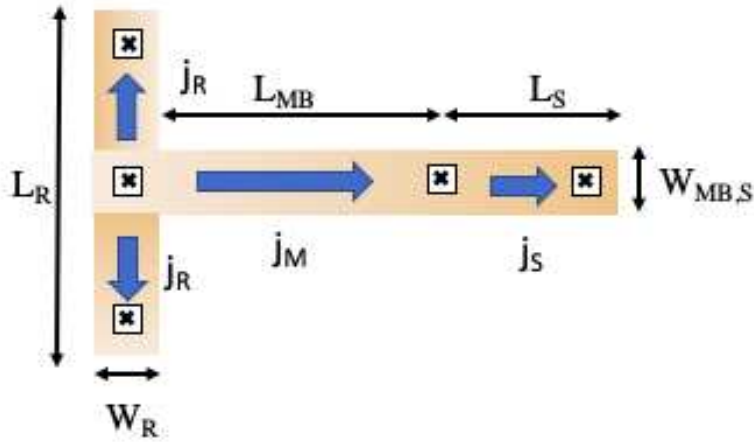


Figure 6.5: The proposed hybrid multi-segment wire structure design

6.2.4 The complete aging sensor design

Delay-based EM aging circuit

Using the previously proposed hybrid wire design, we can implement the EM aging and delay based circuit as shown in Fig. 6.6. The circuit works by stressing a wire in the EM stressed wire branch, which will result in resistance degradation, and then comparing the RC delay (essentially the rise time) between this aged branch and an unstressed reference branch which is designed to have the same resistance and delay as the aged wire. This delay

can then be used to calculate the resistance of the aged wire which can be used to infer the age of module.

The circuit has three modes which are controlled by the control signals *AccSignal* and *EN*. When initially accelerating the void nucleation at the foundry, *AccSignal* is set to 1 and *EN* is set to 0. This will allow an acceleration stress source V_{AS} to produce the current flow needed to put the hybrid structure into its accelerated aging mode by providing current in the main wire R_{MB} (previously referred to as R_{EM}) as well as activating the sink R_{sink} and reservoirs $R_{res,1}$ and $R_{res,2}$. This accelerated stress source V_{AC} will typically be larger than V_{stress} as it will be used during burn-in, however, this depends on the desired void nucleation time. During normal aging operation, *AccSignal* and *EN* are both set to 0. A multiplexer is used to provide a stressing source V_{stress} to R_{MB} during this aging mode. When measurement is needed, *EN* is set to 1. This will generate a signal (such as a unit-step) which will drive both the EM stressed wire branch, as well as the reference wire R_{ref} . The signal arrives at the inputs of the differential amplifier which then outputs the difference of these two signals V_{Diff} . As the resistance in R_{MB} increases due to aging, so to does the rise time of the signal on this branch which leads to larger voltage in V_{Diff} .

In Fig. 6.7 the HSPICE simulation of the response from the differential amplifier using resistance value for $R_{em} = 40\Omega$ and $R_{ref} = 30\Omega$. Wire capacitance consistent with parasitics found in the power delivery network were also used and were equal in each wire. The aging module components and the differential amplifier were built using a Synopsys 90nm PDK. From the results we can see that the differential amplifier is indeed capable of detecting the small timing variation in the two wires, validating our proposed approach.

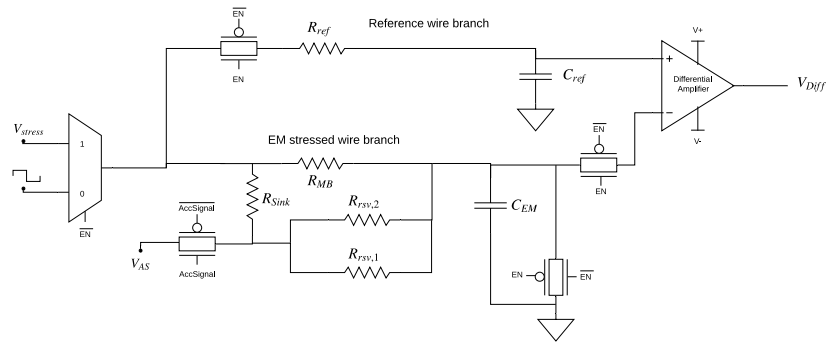


Figure 6.6: The EM induced delay-based aging sensor architecture

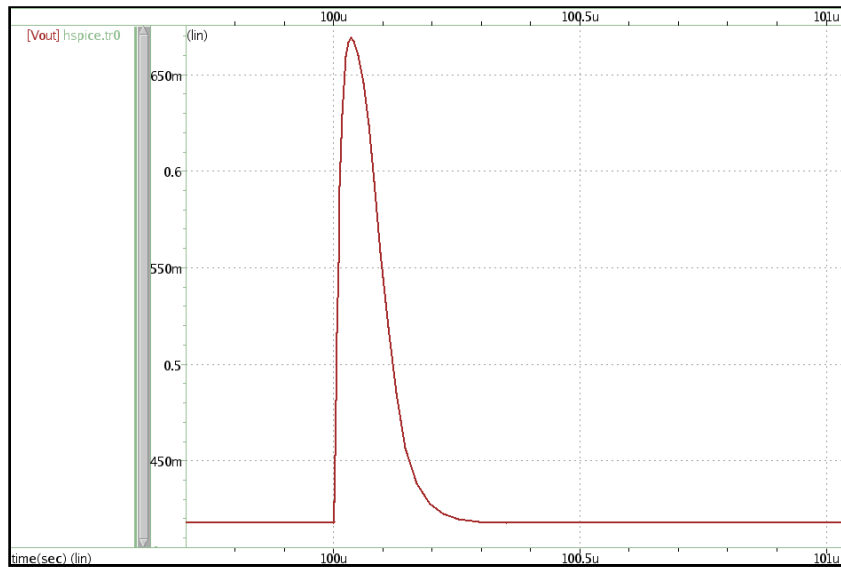


Figure 6.7: SPICE simulation of the differential amplifier showing the detection of the delay degradation caused by increased resistance in an EM failed wire

To calculate the resistance of R_{em} , we can write equation (6.2) for V_{em} and V_{ref} in addition to the equation for the output of the differential amplifier which is $V_{Diff} =$

$V_g(V_{ref} - V_{em})$ where V_g is the amplifier gain. We can then solve this equation for R_{em} yielding the following equation:

$$R_{em} = \frac{-t}{\ln(e^{\frac{-t}{R_{ref}C}} + \frac{V_{Diff}}{V_g V_0})C} \quad (6.4)$$

In this equation, t is the time when the output of the differential amplifier is measured with respect to the starting time of the test signal.

The overall aging sensor module architecture

Using the proposed circuit, the module architecture can be formulated as shown in Fig. 6.8

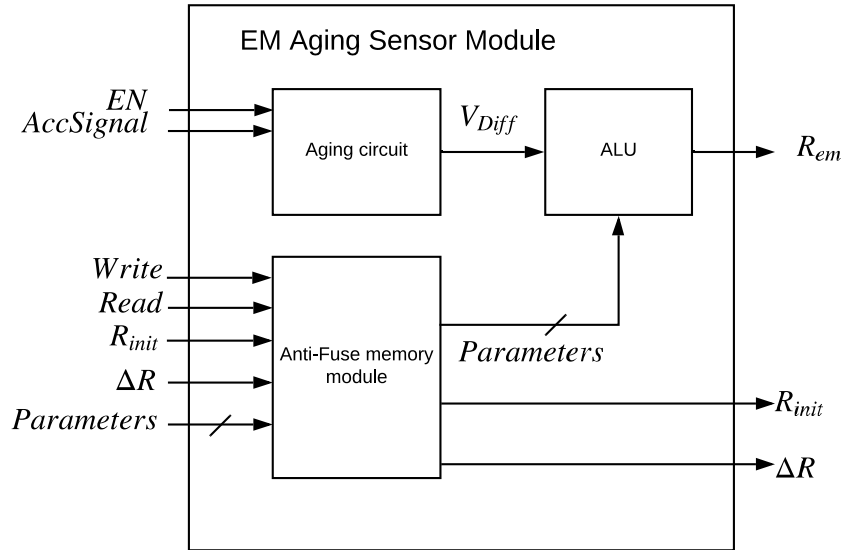


Figure 6.8: The proposed EM induced delay-based aging sensor architecture

In this module, the two control signals $AccSignal$ and EN are the signal inputs to the aging circuit which outputs V_{Diff} . This value is then sent to an ALU (simply performs multiplications with V_{Diff}) which calculates the resistance using parameter constants from 6.4. These parameters are stored in the Anti-fuse memory module (AF). The AF also stores the initial resistance R_{init} and ΔR . AF also has two control signals. One for writing to the AF module, which is a one time operation after the self-calibration, and one for reading from the module. We note that the parameter constants are known to the designers so these are not initially needed for calibration and can be stored as a single coefficient. Instead, the raw V_{Diff} can be used by the design team to perform calibration. Post-calibration, and after writing to the AF, the AF module sends the parameters necessary to calculate R_{MB} to the ALU. The aging sensor module outputs R_{init} and ΔR and the calculated R_{MB} which can be used by the distributor or consumer to calculate the age of the chip.

6.3 Numerical results and comparison

In this section we present a comparison of the proposed method against the previously proposed EM-base aging sensor in [38]. To facilitate the comparison, we set a specification that each module needs to detect up to 10 years of aging.

As a first step, we present the design of the hybrid structure to facilitate the design specification of 10 years aging measurement. To obtain the parameters of wires, we perform finite element analysis to solve the partial differential equations presented in (5.1) for the multi-segment wire structure shown in Fig. 6.5 and then sweep the physical parameters within the range of appropriate values. Additionally, void growth kinetics were

Table 6.1: The designed hybrid wire sensor parameters

Parameter	Value
Main branch length	$90\mu m$
Sink length	$30\mu m$
Reservoir length	$30\mu m$
Width	$0.24\mu m$
Thickness	$0.12\mu m$
Current density	$5E9A/m^2$
Temperature	$345K$
Resistance change per year	3.26Ω
Resistance when void saturate	40.9Ω
Void Saturation time	12.6 years

simulated to determine the ΔR and eventual R_{MB} saturation value which govern the ages that can be measured. Simulations were done under both accelerated conditions with the wire configured in the acceleration mode, to cause the rapid void nucleation and reduction of incubation time, and also under normal aging condition for void growth phase simulation with the accelerated configuration disabled. The results of the final wire parameters along with the normal aging conditions are shown in Table 6.1. Additionally, the burn-in conditions used in conjunction with the accelerated configuration are presented in Table 6.2. This design allows us to nucleate a void extremely quickly, after only 6.5 hours instead of the 126 hours (about 5 days) needed with just the hybrid structure configured in the acceleration mode, which is still reasonably quick. The resulting wire has a maximum measurable age of 12.6 years with a ΔR of $3.26\Omega/year$ which is easily detectable using the delay based sensor. As a result, we can start the self-calibration process in the foundry to get the void growth rate under typical or effective temperature conditions. Once we know the void growth rate, we can compute the ages based on the future resistance change measurements.

Table 6.2: Acceleration operating conditions of designed wire

Current densities ($j_{MB} = j_R = j_S$)	$2E10A/m^2$
Normal condition temperature	$373K$
Burn-in condition temperature	$423K$
Time before growth phase normal condition	126hr
Time before growth phase in burn-in condition	6.5hr

We also design aging modules following the methodology in [38]. However, a primary difference between the proposed method and the method in [38] is that the proposed method can measure age continuously throughout its lifetime while the previous method only indicates that single discrete age has been exceeded. As per the recommendation in [38], we can increase the granularity of this method by designing multiple timer modules that will fail at several different ages. To meet our requirement of measuring age up to 10 years, we chose to create three modules that will fail at 3, 5, and 10 years using the previous method. Again, we use FEA to solve the stress equations in (5.1) to design these wires to the correct specifications. Lastly, to ensure accurate nucleation time prediction, we employ 6 redundant wires for each aging module. A summary of the comparison between this method and our proposed method is presented in Table 6.3.

From the resulting designs, we can see that the proposed method achieves better area efficiency as well as better aging accuracy through fine granularity compared to the previous method. To match the aging accuracy, the previous method would need more modules than is realistic. Lastly, we remind the reader that the method in [38] is only able to achieve accurate nucleation time predictions by implementing redundant wires as discussed in section 6.1 which is the reason for using 6 wires per aging module. The result is a vastly larger aging sensor that is nearly $10\times$ larger than our proposed method. We

Table 6.3: Comparison with the EM-sensor in [38]

Parameter	Proposed method	Method in [38]
Target ages (t years)	$t \leq 10$	$t \in \{3, 5, 10\}$
Number of stressed wires	1	18 (6 wires, 3 modules)
Total area estimated	$43.2\mu m^2$	$460.8\mu m^2$
Single wire and module area breakdown	$R_{MB} = 21.6\mu m^2$, $R_{sink} = 7.6\mu m^2$, $R_{rsv,1+2} = 14.4\mu m^2$	10 year: $19.2\mu m^2$, 5 year: $26.4\mu m^2$, 3 year: $31.2\mu m^2$
Area overhead	N/A	966.66%

lastly note that only wire areas are reported as they dominate the area overhead of both our proposed method and the method in [38].

This chapter presented a novel on-chip aging sensor module for the detection of counterfeit ICs. The proposed module allows consumers and electronics component manufacturers to verify that an IC is genuinely new and has not been recycled and remarked. This module utilizes the electromigration failure effect to induce resistance degradation in a stressed wire which results in measurable changes in the RC delay and the subsequent calculation of the wire age. The new module makes use of advanced EM modeling techniques that enable the utilization of a configurable multi-segment wire structure to accelerate the void nucleation and facilitate the control of the void growth rate. This new module also relies on a novel self-calibration methodology that removes the necessity for large numbers of redundant wires that other methods require. The resulting design vastly increases measurable age granularity and has very high area efficiency, nearly $10\times$ smaller, compared to previously proposed methods.

Chapter 7

GPU-based Ising Computing for Max-cut

In VLSI physical design, many algorithms require the solution of difficult combinatorial optimization problems such as max/min-cut, max-flow problems etc. Due to the vast number of elements typically found in this problem domain, these problems are computationally intractable leading to the use of approximate solutions. This chapter explores the Ising spin glass model as a solution methodology for hard combinatorial optimization problems using the general purpose GPU (GPGPU). The Ising model is a mathematical model of ferromagnetism in statistical mechanics. Ising computing finds a minimum energy state for the Ising model which essentially corresponds to the expected optimal solution of the original problem. Many combinatorial optimization problems can be mapped into the Ising model. In this chapter the focus is on the max-cut problem as it is relevant to many VLSI design automation problems. Our method is inspired by the observation that

Ising annealing process is very amenable to fine-grain massive parallel GPU computing. The chapter will illustrate how the natural randomness of GPU thread scheduling can be exploited during the annealing process to create random update patterns and allow better GPU resource utilization. Furthermore, the proposed GPU-based Ising computing can handle any general Ising graph with arbitrary connections, which was shown to be difficult for existing FPGA and other hardware based implementation methods. Numerical results show that the proposed GPU Ising max-cut solver can deliver more than 2000X speedup over the CPU version of the algorithm on some large examples, which shows huge performance improvement for addressing many hard optimization algorithms for solving practical VLSI design automation problems.

7.1 Review of existing work

There have been several works previously proposed to utilize hardware acceleration techniques, other than quantum computers, as a solver for the Ising model to solve combinatorial optimization problems. The previous works have utilized accelerators such as GPUs [13, 16, 87], FPGAs [35, 90], and even ASIC implementations [89].

The Ising model for many practical problems can lead to very large connections among Ising spins or cells. Furthermore, embedding those connections into the 2-dimensional fixed degree spin arrays in VLSI chips is not a trivial problem. Doing so requires mitigation techniques such as cell cloning and splitting using graph minor embedding (another NP-hard problem) as proposed in [35, 36, 90]. This stems from the fact that these hardware based methods must convert every problem to a nearest-neighbor model to

ensure that their regularly arranged hardware spins can accommodate any graph problem. The solution to this NP-hard problem will drastically increase pre-processing time for the Ising solver. Additionally, graph embedding requires the cloning and splitting of nodes in the graph which means the number of spins in the model will also greatly increase. Furthermore, after the solution has been obtained, post-processing must be done to extract the real solution from the solver by deconstructing the graph embedding. This leads to another problem, that being hardware utilization, since the number of spins increases when a graph is embedded into the nearest neighbor model which will make it more difficult to fit larger problems on an FPGA or require more area in an ASIC implementation. Alternatively, hardware designs would require drastic increases in routing and connectivity hardware, however, even then it would not be guaranteed to handle every problem. In essence, ASIC implementations are not flexible and can only handle a specific problems or utilize graph embedding due to the fixed topology among spins, and FPGA implementations require architectural redesign, and thus recompilation, for each different problem if graph embedding is not utilized. Lastly, one has to design hardware for the random number generator for each spin cell and simulate the temperature changes, which occupies significant chip area resulting in scalability degradation.

Based on the above observations and the highly parallel nature of the Ising model, in this work, the General Purpose Graphics Processing Unit (GPGPU or simply GPU) is explored as the Ising model annealing computing platform. The GPU is a general computing platform, which can provide much more flexibility over VLSI hardware based annealing solutions as a GPU can be programmed in a more general way, enabling it to handle any

problem that can be mapped to the Ising model. That is, it is not restricted by the topology or complex connections that some problems may have. At the same time, it provides massive parallelisms compared to existing CPUs. The GPU is an architecture that utilizes large amounts of compute cores to achieve high throughput performance. This allows for very good performance when computing algorithms that are amenable to parallel computation while also having very large data sets which can occupy the computational resources of the GPU [63, 64]. The problem sizes in the design automation domain can easily accomplish this and heuristic methods can solve the Ising model in a parallel manner which makes the GPU ideal for this application. We remark that extensive work for Ising computing on GPUs have been proposed already [13, 16, 87], however; they still focus on physics problems which assume a nearest neighbor model only. This model is highly amenable to the GPU computing as it is easily load balanced across threads but is not general enough to handle problems such as max-cut without extensive pre-processing, again through graph embedding. Furthermore, many GPU-based methods use a checkerboard update scheme to ensure spin updates are uncorrelated, but this is still only practical for the nearest neighbor model.

7.2 Ising model and Ising computing

7.2.1 Ising model overview

The Ising model consists of a set of spins interconnected with each other by a weighted edge. For the general Ising model, spin connections can take on any topology. One of the connection topologies is the 2D lattice, referred to as the nearest neighbor model, shown in Fig. 7.1, which describes the ferromagnetic interactions between so-called spin

glasses. Many computationally intractable problems can be mapped to this Ising model. It was shown that finding the ground state in the 2D lattice Ising model is an NP-hard problem [12]. However, it has certain characteristics that make it more amenable to the annealing process as each local update results in energy minimization and spin glass updates can be performed in a highly parallel manner.

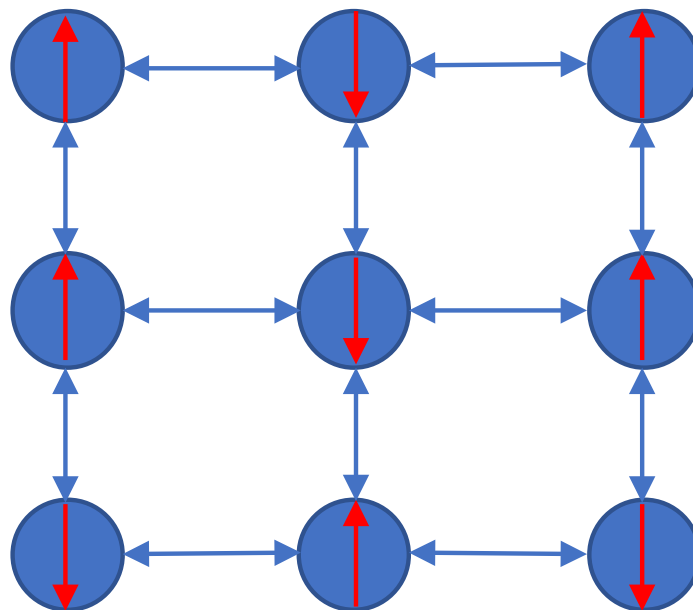


Figure 7.1: The 2D nearest neighbor Ising model.

Specifically, each spin σ_i , has two discrete spin values $\sigma_i \in \{-1, 1\}$ and some interaction with adjacent spins in the form of a weighted edge. Then the local energy or Hamiltonian of the spin is described by (7.1):

$$\mathcal{H}_i(\sigma_i) = - \sum_j J_{i,j} \sigma_i \sigma_j - h_i \sigma_i \quad (7.1)$$

In this equation, $J_{i,j}$ is the interaction weight between σ_i and σ_j , and h_i is a bias or external force acting on σ_i .

Local spin update: by finding the minimum value of $\mathcal{H}_i(\sigma_i)$, we can determine the local spin value σ_i . Specifically (7.1) can be written as

$$\mathcal{H}_i(\sigma_i) = \left(- \sum_j J_{i,j} \sigma_j - h_i \right) \sigma_i = -S \times \sigma_i \quad (7.2)$$

From (7.2), we can see that σ_i can be determined just from the sign of the S value. If $S > 0$, $\sigma_i = 1$, otherwise, $\sigma_i = -1$. If $S = 0$, it can take any value of $\{-1, 1\}$. This is called a **local spin update or update**, in Ising computing. We note that such an update for obtaining the minimum value of $\mathcal{H}_i(\sigma_i)$ only depends on its neighbors. That is, each individual spin only needs to know the energy of the spins that are directly connected to it for it to determine its next spin value. By ensuring that spin updates are not correlated, then all the spin updates can be done *independently* and thus *in parallel* [13, 16, 35, 89, 90].

The global energy of the model is simply the summation of the local energies. It is apparent then that minimizing the local energies will lead to the ground state of the model. The global energy of the whole Ising model is given by the following (7.3):

$$\mathcal{H}(\sigma_1, \sigma_2, \dots, \sigma_n) = - \sum_{\langle i,j \rangle} J_{i,j} \sigma_i \sigma_j - \sum_i h_i \sigma_i \quad (7.3)$$

Note that $\langle i, j \rangle$ indicates the combination of all spin interactions, and thus the energy of the whole model. Minimizing the energy of this equation essentially requires finding the optimal spin configurations given the interaction of each individual spin with their neighbors. While (7.2) describes the energy of a single spin, this new equation in (7.3) describes the total summation of off spins and their interactions.

In general, we refer the problem of finding the minimum energy of the Ising Model, or equivalently the ground state of Ising Hamiltonian, as the Ising problem. It can be shown that the Ising problem shown is equivalent to the problem of quadratic unconstrained Boolean optimization (QUBO) [18]. It was shown that many computationally intractable problems (such as those in class NP-complete or NP-hard) can be converted into Ising models [56].

Previous methods have focused on solving the nearest neighbor Ising model [35,89,90]. However, this model has the drawback of not being able to handle any general problem which may have arbitrary and complex connections. To do so, another NP-hard problem, the graph embedding problem, must be solved first to convert a generally connected graph to the nearest neighbor model which will be very computationally intensive and even resource prohibitive for certain hardware implementations. Therefore, in this work, we assume that a spin glass's connections, or edges, are able to connect to any other spin glass in the model, an example of which is shown in Fig. 7.2. Using this more general model removes the nearest neighbor restriction on the Ising model and allows us to handle more complex problems.

From the Hamiltonian equations presented in this section, we can see that if we minimize the local energy of each spin, we also minimize the global energy of the entire system which leads to the ground state of the model. We can then map a combinatorial optimization problem to this Ising model such that the ground state of the Ising model corresponds to the global solution of the corresponding optimization problem. In this way, finding the local energy of each spin, which leads to finding the ground state of the Ising model, is the same as finding the optimal solution to the problem in question.

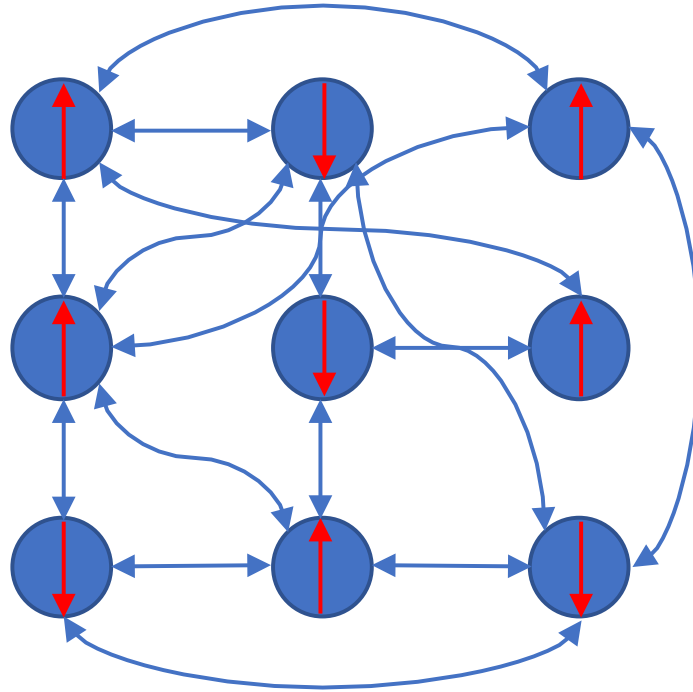


Figure 7.2: An example of a generally connected Ising model.

7.2.2 Annealing method for Ising model solution

A classical and well known heuristic for combinatorial optimization is Simulated Annealing (SA) [50]. This heuristic mimics the behavior of thermal annealing, found in metallurgy. Essentially, it works by setting the environment to a high “temperature,” giving the model high energy and allowing for higher probability of changing states, and then gradually decreases the temperature as the simulation runs. More precisely, it iteratively calculates and evaluates the global solution quality of neighbor states of a model and probabilistically allows the acceptance of a new state even if its solution quality is worse than the previous state by utilizing the Metropolis criteria. The probability of accepting a worse

state is dependent on the temperature of the system which gradually decays over time. This allows the heuristic to avoid local minima as depicted in Fig. 7.3.

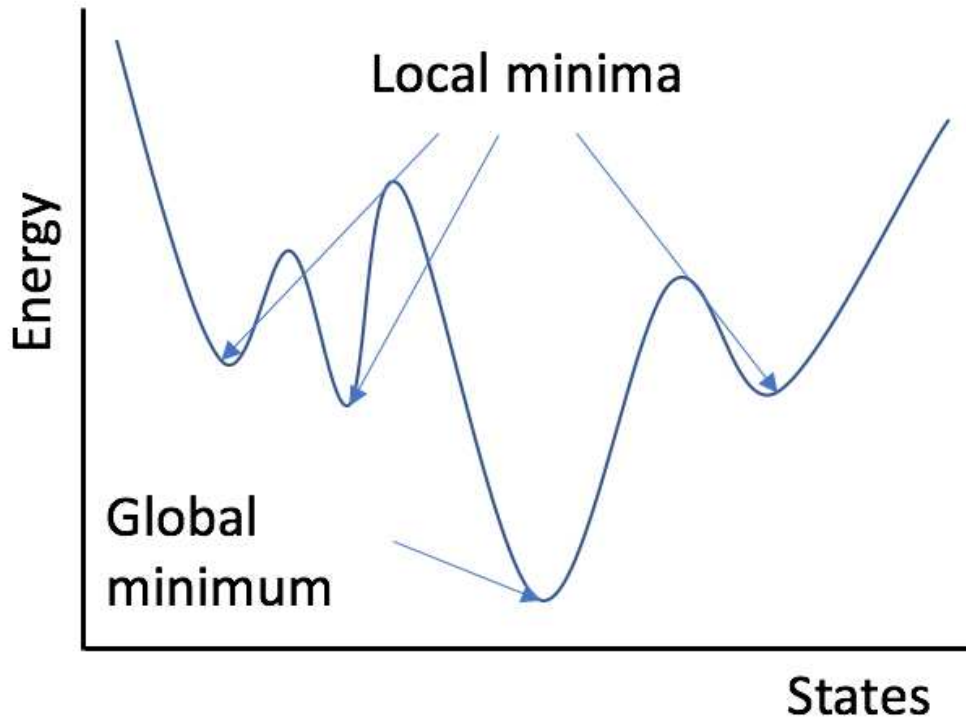


Figure 7.3: Depiction of the local minima and global minimum in the energy minimization problem.

In this proposed work, we use a simplified/modified Metropolis annealing algorithm to find the ground state as shown in Algorithm 2 that better exploits the features of the Ising model while also allowing us to avoid local minima. In our proposed method, we allow each spin update to minimize its own local energy and do not compute a global solution quality (which would add a large computational penalty). In order to avoid local

Algorithm 2 Modified Ising annealing algorithm

```
1: input:  $(M, N, \mathbf{S})$ 

2: initialize all  $\sigma_i$  in  $\mathbf{S}$ 

3: for sweep-id in  $\{1, 2, \dots, M\}$  do

4:   for  $\sigma_i$  in  $\mathbf{S}$  do

5:      $\sigma_i \leftarrow \operatorname{argmin}(H(\sigma_i))$  based on (7.2)

6:   end for

7:   randomly choose and flip  $N$  spin glasses in  $\mathbf{S}$ 

8:   decrease  $N$ 

9: end for
```

minima we add energy to our Ising model by utilizing random flip probabilities that decay over time.

In algorithm 2, M is the maximum number of sweeps, N is the number of spins to randomly flip, and \mathbf{S} is the set of all spin glasses. Once the spin glasses are initialized, all spin glasses are updated iteratively to propagate the interactions between each spin (a process we will call a “sweep”). When a spin glass updates based on (7.2), it computes and chooses the spin value that will minimize its local Hamiltonian. At the end of a sweep, N glasses are randomly flipped and N is decreased according to an annealing schedule. After this, the process is repeated for M sweeps or until convergence is achieved.

Following this process, the local energy of each spin glass, and thus the global energy of the model, is gradually decreased. Furthermore, we avoid the local minima by introducing energy to the model by using uncorrelated random flips which slowly decrease over time.

We want to remark that the modified Ising annealing process can be viewed as a simplified Metropolis Monte Carlo method as we essentially accept the positive energy changes of a spin flip based on a temperature dependent probability [57]. As a result, we should have the convergence properties of the Metropolis method [52], which means that the objective function will be minimized in a statistical way for a sufficient time. We notice that similar Ising annealing processes have been used for FPGA and ASIC based Ising computing [35, 36, 89].

7.3 Ising model for the max-cut problem

The Ising model and the method introduced in this paper can be applied to many NP class problems, however, we use the max-cut problem as a practical example. The max-cut problem, in practice, can help find solutions to several EDA and VLSI design problems. For example, the general via minimization problem, the act of assigning wire segments to layers of metalization such that the number of vias is minimized, can be modeled as a max-cut problem [10, 11, 24]. The max-cut problem is highly amenable to the Ising model, making it an ideal candidate to introduce the proposed methodology. Furthermore, we also note that the Ising spin model and max-cut have been used as a solution technique for the via-minimization problem in the past [11].

The max-cut problem is defined as partitioning a graph into two subsets S and \bar{S} such that the weighted edges between the vertices of one subset and the other subset are maximized. This is mathematically formulated by (7.4) assuming a graph $G = (V, E)$ has a variable x_i assigned to each vertex:

$$\begin{aligned} \max \frac{1}{2} \sum_{i,j \in V, i < j} w_{i,j} (1 - x_i x_j) \\ \text{s.t. } x_i \in \{1, -1\} \end{aligned} \tag{7.4}$$

In this equation, V is the set of vertices in the graph G , $w_{i,j}$ is the edge weights in E between the i th and j th elements in V , and x_i is an indication of which subset the vertex belongs to and can take the values $\{-1, 1\}$.

Intuitively, looking as the Ising spin glass problem in (7.3), we can see how the max-cut problem should map to the Ising model by associating the spin of a spin glass σ_i with a subset of the graph in the max-cut problem. That is, we can say that if a spin is 1 then the spin glass is in S and if a spin is -1 then it is in \bar{S} , which is analogous to x_i . Furthermore, the weights between vertices $w_{i,j}$ is the same as the interaction weights between spin glasses $J_{i,j}$ and, in this case, there is no bias or external force so the h term in the Ising model is simply zero. The global energy minimization of the Ising model for the max-cut problem is shown below in (7.5):

$$\mathcal{H} = - \sum_{\langle i,j \rangle} J_{i,j} \sigma_i \sigma_j \tag{7.5}$$

Once mapped to the Ising model, the max-cut problem can then be solved by finding the ground state of the model using the methods proposed in this paper. While there are other ways to solve this problem, the method we propose is highly amenable to parallel computation and large problem sets have great performance when implemented on the GPU, thus, giving our method an advantage in scalability.

7.4 GPU Implementation

7.4.1 GPU Architecture

The general purpose GPU is an architecture designed for highly parallel workloads which is leveraged by Nvidia's CUDA, Compute Unified Device Architecture, programming model [63]. GPUs offer massive parallelism favoring throughput oriented computing in contrast to traditional CPUs which focus primarily on latency optimized computation. This gives the GPU an advantage in scalability so long as a problem does not contain large amounts of sequential operations. The Nvidia GPU architecture is comprised of several Symmetric Multiprocessors (SMs), each containing a number of "CUDA" cores, and a very large amount of DRAM global memory [64]. The Kepler architecture based Tesla K40c GPU, for example, has 15 SMs for a total of 2880 CUDA cores (192 cores per SM), and 12GB DRAM global memory. Additionally, each SM has several special function units, shared memory, and its own cache.

The CUDA programming model, shown in Fig. 7.4, extends the C language adding support for thread and memory allocation and also the essential functions for driving the GPU [62]. The model makes a distinction between the host and device or the CPU and GPU respectively. The model uses an offloading methodology in which the host can launch a device kernel (the actual GPU program) and also prepare the device for the coming computation, e.g., the host will create the thread organization, allocate memory, and copy data to the device. In practice, a programmer must launch many threads which will be used to execute the GPU kernel. Thread organization is therefore extremely important in GPU programming. Threads are organized into blocks which are organized into grids. Each

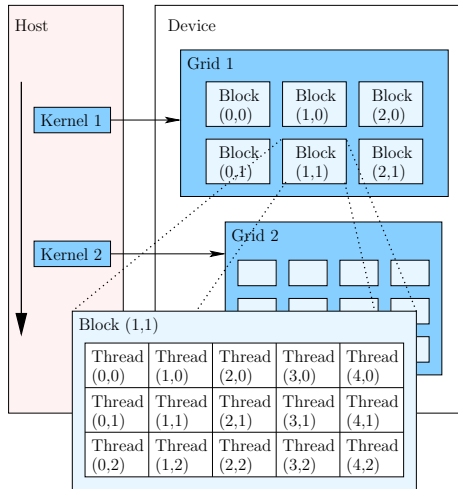


Figure 7.4: The Nvidia CUDA programming model showing the Host (CPU) and Device (GPU) and the relation between threads, blocks, and grids.

block of threads also has its own shared memory which is accessible to all the threads in that block. Additionally, the threads in the block can also access a global memory on the GPU which is available to all threads across all blocks.

The GPU fundamentally focuses on throughput over speed. This throughput is achieved through the massive compute resources able to be run in parallel. Because of this, it is important to realize that the GPU is not meant for small data sets or extremely complicated operations that may be better suited for a powerful CPU. Instead, the GPU is meant to execute relatively simple instructions on massive data in parallel that can occupy the GPU resources for an extended period of time. This computing paradigm gives the GPU a huge advantage in scalability so long as it has sufficient hardware resources for the problem being solved. So long as this is the case, and there is no significant sequential operations in the GPU kernel, the GPU computation time for a problem will not grow significantly.

7.4.2 Ising model implementation

The GPU, while lacking the massive scaling of a quantum computer, has much larger scaling capabilities than CPUs, allowing it to handle very large problem sizes. Indeed, smaller problems that are unable to fully utilize the GPU resources may achieve worse performance than a CPU.

In order to ensure the best utilization of GPU resources, it is necessary to devise a spin glass update scheme more amenable to parallel computation. Algorithm 2 relies on sequential updates to propagate the interactions between the spin glasses. However, this would be highly inefficient on the GPU as it would mean each thread would have to wait for previous threads to update.

In previous works that have addressed the nearest neighbor Ising model, a checkerboard update scheme is implemented which allows for many spin glasses to be updated in parallel [87]. While the spin glass updates are independent of their neighbors, they are not truly independent since the update pattern is deterministic and can introduce autocorrelation between spin updates but this autocorrelation generally does not affect the global balance of the model [87]. For the problem addressed in this work, however; interactions are not confined to nearest neighbors nor are they restricted to regular patterns. This results in very complex interactions in which spins can be dependent on many other spins across the entire model. Consequently, a different update scheme must be developed for such a general solver to ensure the independence of parallel updates.

To address the above mentioned issues, we modify the original algorithm in algorithm 2. Firstly, we assign each thread to a single spin glass, and make it responsible for

updating that glass. One may notice that since each spin glass may have a different number of neighbors, then the threads will not be perfectly load balanced. However, the alternative is to use graph minor embedding, another NP-hard problem [19], to create clone nodes such that every thread will have an equal number of updates [35]. However, this means that we need to have the CPU do some intensive pre-processing on the model, and it also means that after each update sweep, a reduction must be performed on each spin glass's clones so that the true spin value can be determined. For these reasons, it is much better to allow for some load imbalance and suffer some computational penalty on the GPU, instead of increasing the complexity of the algorithm. Furthermore, while each update sweep is synchronized, we do not synchronize the updates of each spin which makes them uncorrelated. In practice, this means that threads will update their assigned spin glass as soon as they are scheduled and will use whatever spin status their respective neighbors have at the time of data access. Therefore, there is no guarantee that a spin's neighbors will contain the spin value from the current sweep or from the previous sweep. This naturally propagates the updates of each spin glass in a non-deterministic pattern which ensures that each spin update can be done in independently of its neighbor and in parallel.

Another major change to the algorithm is the implementation of the random flips. Instead of randomly selecting a number of spin glasses to be flipped at the end of an update sweep, we let each thread independently decide if it should flip or not. A global variable, visible to all threads, gives the flip probability in the form of a floating point number between 0.0 and 1.0. Each thread then independently generates a random number between 0.0 and 1.0, generated by the CUDA cuRAND library for efficient random number generation, and

flips if the number is below the global flip probability. The cuRAND library used for the random number generations allows us to create a random number generator for each thread at the very beginning of the program. Each thread is then responsible for calling its own generator when it requires a random number. The advantage of using the cuRAND library is that the random numbers can be generated completely in parallel and independent of each other by allowing the GPU threads to do their own random number generation work. This also is amenable to our asynchronous update scheme since each thread is responsible for its own flipping and can decide to flip or not as soon as it finishes its update without waiting for other threads which may still be updating. Consequently, a thread may not only update using an already updated neighbor, it may actually update using a neighbor that has been randomly flipped also. The flip probability is then reduced as update sweeps are completed.

Algorithm 3 GPU Simulated Annealing method for Ising model

input: (F_p, \mathbf{S})

initialize ALL σ_i in \mathbf{S}

while $F_p > 0$ **do**

for all $\sigma_i \in \mathbf{S}$ **in parallel do**

$\sigma_i \leftarrow \operatorname{argmin}(H(\sigma_i))$

 flip σ_i with probability F_p

end for

 reduce F_p

end while

In algorithm 3, the parallel simulated annealing solver for the Ising model is presented. While mostly similar to the algorithm 2, There are some differences. We replace the number of random flips input with a variable F_p which represents the flip probability mentioned above. Next, we have each thread generate a random number, using the cuRAND library, between 0.0 and 1.0 and compare this to F_p . If it is less, then the thread will flip the spin value. While these changes are subtle, the effect is large as it allows the parallel computation of an entire update sweep.

With respect to GPU programming optimizations, the typical best practices were followed or considered during implementation. We observe that thread divergence is not an issue in our algorithm due to the fact that each thread performs the exact same operations. This means that when a group of threads are running in parallel, they will not be serialized at anytime. Furthermore, while memory coalescing is a standard method of enhancing memory access speeds, we recognize that memory accesses in the Ising model are not regular. This means implementing memory coalescing would result in unnecessary storage complexity.

7.5 Experimental Results

In this section, we present the experimental results showing both the accuracy and speed of our parallel GPU-based Ising model solver for the max-cut problem. The CPU-based solution is done using a Linux server with two Xeon E5-2698v2 2.3 GHz processors, each having 16 cores (2 threads per core for a total of 32 threads per CPU and 64 threads total in the server), and 72 GB of memory. On the same server, we also implement the GPU-based solver using the Nvidia Tesla K40c GPU which has 2880 CUDA cores and 12 GB

of memory. Test problems from the G-set benchmark [5] are used for testing as well as some custom made problems to show very large cases. The problems' edge counts are used to represent their size as the size of each problem is typically dominated by the number of edges. However, we remark that this is not always the case for every graph. That is, some graphs may have a very large number of edges but only a few nodes, or vice versa, which may make it easier or harder for a certain algorithm to compute. As such, some graphs with higher numbers of edges may take less time to compute. Furthermore, graphs with large numbers of edges may have more regular connections which will lead to more load balance and thus, quicker computation times. However, the general trend in computation time is that a larger number of edges will take more computation time.

7.5.1 Accuracy study

To test the accuracy of the method presented in this paper, we compare the max-cut value our method generates with that of the best known solution in the G-set benchmark. In addition to the G-set benchmark comparison, we generated several custom graphs and compared the solution quality of our GPU-based method against IBM's CPLEX mathematical programming solver [43], a state-of-the-art linear programming solver employed to solve combinatorial optimization problems. CPLEX solutions were implemented using a server with a 2.1 GHz Xeon Broadwell processor with 36 threads (18 cores with 2 threads per core) and 128 GB of memory.

For the results in Table 7.1 and Table 7.2, cut values were obtained by running the GPU solver 10 times per graph and taking the average solution quality. Each time the

Table 7.1: Accuracy comparison of the GPU max-cut value against the best known cut values for the G-set benchmark.

Graph	Best known cut	GPU cut (%accuracy)
G13	580	522(90.0%)
G34	1372	1191(86.8%)
G19	903	844(93.5%)
G21	931	880(94.6%)
G20	941	880(93.6%)
G18	988	938(94.9%)
G51	3846	3754(97.6%)
G53	3846	3756(97.7%)
G54	3846	3756(97.7%)
G50	5880	5803(98.7%)
G47	6656	6619(99.4%)
G40	2387	2267(95.0%)
G39	2395	2269(94.7%)
G42	2469	2325(94.2%)
G41	2398	2284(95.2%)
G9	2048	2004(97.9%)
G31	3288	3227(98.2%)

Table 7.2: Accuracy comparison of the GPU max-cut value against the cut values obtained by CPLEX.

# edges	CPLEX cut	GPU cut (%accuracy)
9999	9473	8884 (93.78%)
14999	13357	12776(95.65%)
24998	20206	19981(98.88%)
49995	35248	36228(100.29%)
39998	33605	32914(97.94%)
59997	46371	46510(100.29%)
99995	70566	72009(102.04%)
199990	128448	131930(102.71%)
249995	176556	179391(101.60%)
374993	248505	255078(102.64%)
626988	392912	400540(101.94%)
1249975	741709	751050(101.25%)

solver was run we used 1000 annealing steps which is the same number used to generate the performance data in the following performance study section. Additionally, the number of random flips for each solution is set to 200 which decays linearly each sweep by a factor of 0.01. The cut values generated by CPLEX were obtained by running the CPLEX solver

for each graph case. The solver time for CPLEX was capped at two days with most cases using all of the allocated time. We also note that we don't compare the solution time to our solver since the CPLEX solution time is on the order of days while the GPU Ising solver's solution time is on the order of seconds. The *accuracy%* is defined as the ratio of the cut values of the GPU Ising solver over the best known cut (Table 7.1) or the CPLEX solver cut (Table 7.2), i.e., the closer to 100% the better the *accuracy%*. If *accuracy%* is larger than 100%, then GPU Ising solver obtained a better result than the competing solver.

From the tables, we can see that the GPU consistently performs well with almost all results above 90%. Furthermore, in Table 7.2 our GPU-based solver is able to consistently beat CPLEX for larger cases which become too large for CPLEX to solve in a reasonable amount of time. We also note that in practice, the best result could be picked from a number of GPU simulations and some simulation parameters could be tuned to achieve better results, e.g., annealing schedule and initial flip probability. However, we present average results of a parameter configuration we found to be consistent across many graphs.

In Fig. 7.5 the region of convergence is shown for the GPU Ising solver and was obtained by running the solver several times for a particular problem. The red line shows the lowest observed accuracy while the green line shows the highest observed accuracy. One example run is also included to show the overall behavior of the convergence. Unlike classic simulated annealing, the solver does not converge to a single state but rather it continues to have minor variations in energy as the solver progresses. This is because of the utilization of the GPU scheduler as the random update pattern which means there will be

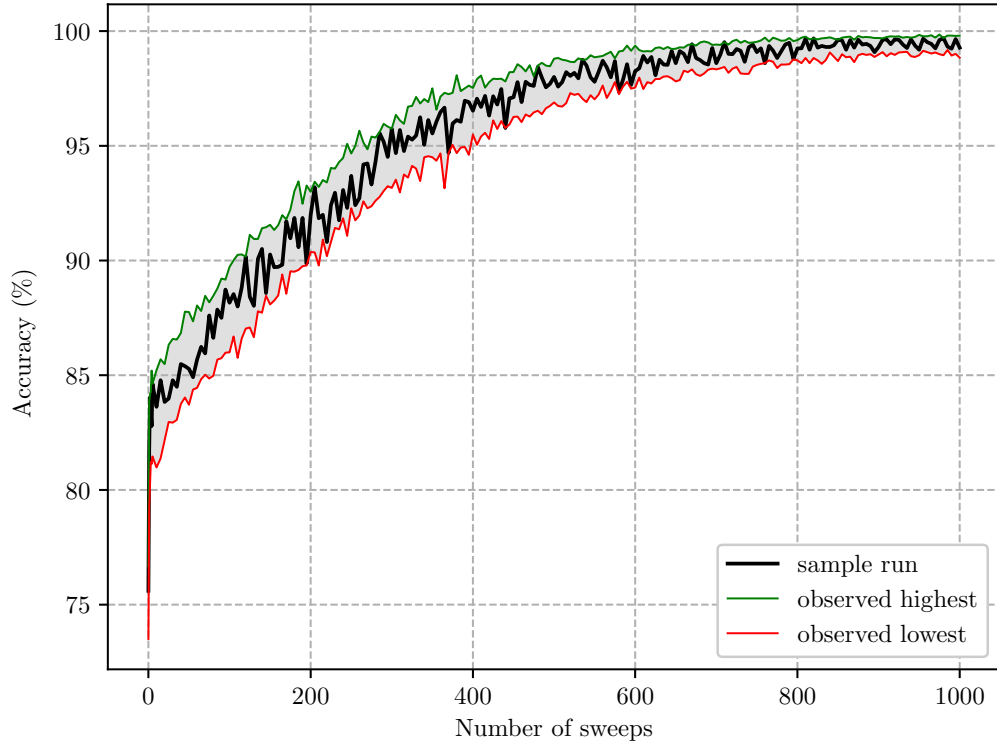


Figure 7.5: Convergence region of the GPU-based annealing for the Ising model on the G-set G47 problem.

some oscillations between solution states, even when the number of random flips is small or zero.

7.5.2 Performance study

Next, we look at the GPU Ising solver’s performance. The speed of the GPU Ising solver is judged by measuring how long it takes to perform a number of update sweeps on various sized models. We run both the proposed GPU Ising solver and the sequential CPU implementation of the algorithm for 1000 sweeps (which is the same number of sweeps used

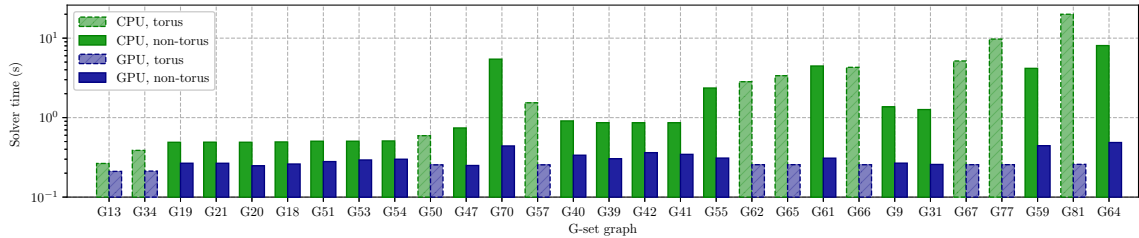


Figure 7.6: Speedup results of the GPU against the CPU for the G-set benchmark problems.

to generate the accuracy results above) and compare the computation time. Because the GPU performs best when its resources are fully utilized, and because it is not optimized for small workloads, we expect to see performance gain over the cpu to improve as the problem size increases. We also should not expect a large speed-up over a CPU version for smaller problems.

In Table 7.3, the time (in seconds) to complete 1000 simulation sweeps is shown for the CPU and GPU for various G-set problems of increasing edge counts. In this table, column *torus* indicates whether the graph is a torus or not. Columns $t_{CPU}(s)$ and $t_{GPU}(s)$ are the run times for CPU and GPU based solutions respectively. Column *speedup* is the speedup of GPU solution over CPU solution defined as $speedup = t_{CPU}(s)/t_{GPU}(s)$. Furthermore, we include several very large custom made and randomly generated non-torus graphs in the table to show the scalability of the proposed method against the CPU-based solution. It should be noted that accuracy results are not included for the custom graphs as there is no data for best known or optimal maximized cut values. Additionally, Fig. 7.7 graphically shows the speed results in seconds for increasing problem sizes and Fig. 7.6 shows a bar graph of the results in log scale (the large custom graphs are omitted from the figures). For all simulations, small and large, the GPU outperformed the CPU competition.

Table 7.3: Performance results comparing GPU performance against CPU performance for the G-set benchmark (G prefix) problems and large custom problems (C prefix).

graph	# vertices	# edges	torus	t_{CPU} (s)	t_{GPU} (s)	speedup
G13	800	1600	yes	0.17	0.11	1.5
G34	2000	4000	yes	0.29	0.11	2.6
G19	800	4661	no	0.39	0.17	2.3
G21	800	4667	no	0.39	0.17	2.3
G20	800	4672	no	0.39	0.15	2.6
G18	800	4694	no	0.39	0.16	2.5
G51	1000	5909	no	0.41	0.18	2.3
G53	1000	5914	no	0.41	0.19	2.1
G54	1000	5916	no	0.41	0.20	2.0
G50	3000	6000	yes	0.49	0.15	3.2
G47	1000	9990	no	0.64	0.15	4.3
G70	10000	9999	no	5.34	0.34	15.7
G57	7000	10000	yes	1.44	0.15	9.3
G40	2000	11766	no	0.81	0.24	3.4
G39	2000	11778	no	0.76	0.20	3.7
G42	2000	11779	no	0.76	0.26	2.9
G41	2000	11785	no	0.76	0.25	3.1
G55	5000	12498	no	2.26	0.21	10.8
G62	7000	14000	yes	2.73	0.16	17.5
G65	8000	16000	yes	3.27	0.16	21.0
G61	7000	17148	no	4.36	0.21	20.9
G66	9000	18000	yes	4.19	0.16	27.0
G9	800	19176	no	1.27	0.17	7.6
G31	2000	19990	no	1.16	0.16	7.4
G67	10000	20000	yes	5.06	0.16	32.5
G77	14000	28000	yes	9.64	0.16	61.8
G59	5000	29570	no	4.07	0.34	11.9
G81	20000	40000	yes	19.89	0.16	125.6
G64	7000	41459	no	7.97	0.39	20.6
C1	10000	100E3	no	26.63	0.18	147.9
C2	10000	250E3	no	59.81	0.38	157.39
C3	10000	500E3	no	121.5	0.61	199.5
C4	10000	750E3	no	179.87	0.91	197.65
C5	10000	1E6	no	234.86	1.18	198.69
C6	100E3	5E6	no	1.56E4	7.11	2200.81
C7	100E3	7E6	no	2.05E4	9.99	2254.74

For the very large custom graphs, the scalability of the GPU can really be seen as the CPU struggles to handle such large problems while the GPU is able to finish them quite quickly and even achieve over 2000X speedup for the largest problems.

The CPU based Ising solution time grew approximately quadratically with the problem size which is expected and is similar to the observations of the reported quantum adiabatic computing on the exact cover problem [31]. We can also see that the GPU solver time seems to remain quite constant until it starts working on the very large random graphs. This seemingly constant computation time trend can be explained by the GPU architecture. The GPU achieves its performance by utilizing a vast number of computational resources which enables high throughput computations. Because of this, we do not expect the computation time to rise dramatically with a rise in problem size so long as the computational resources of the GPU are sufficient to handle the problem size. In contrast, the CPU's computation time will be impacted regardless of the problem size.

For the most part, as the problem size increased, so did the GPU speedup. However, we note that there were some graphs which the CPU performed quite well and the speedup that the GPU achieved was much smaller than on other graphs. This is primarily seen in the smaller graphs where the large amount of compute resources cannot be fully utilized on the GPU. It is important to note that the GPU performance is achievable due to the large amount of compute resources that can be utilized in parallel. However, if the size of the graph is much larger than the amount of compute resources, then we would expect the performance to degrade but also to still be better than the CPU. Additionally, for very small graphs that don't occupy the GPU resources, we should expect the performance gain

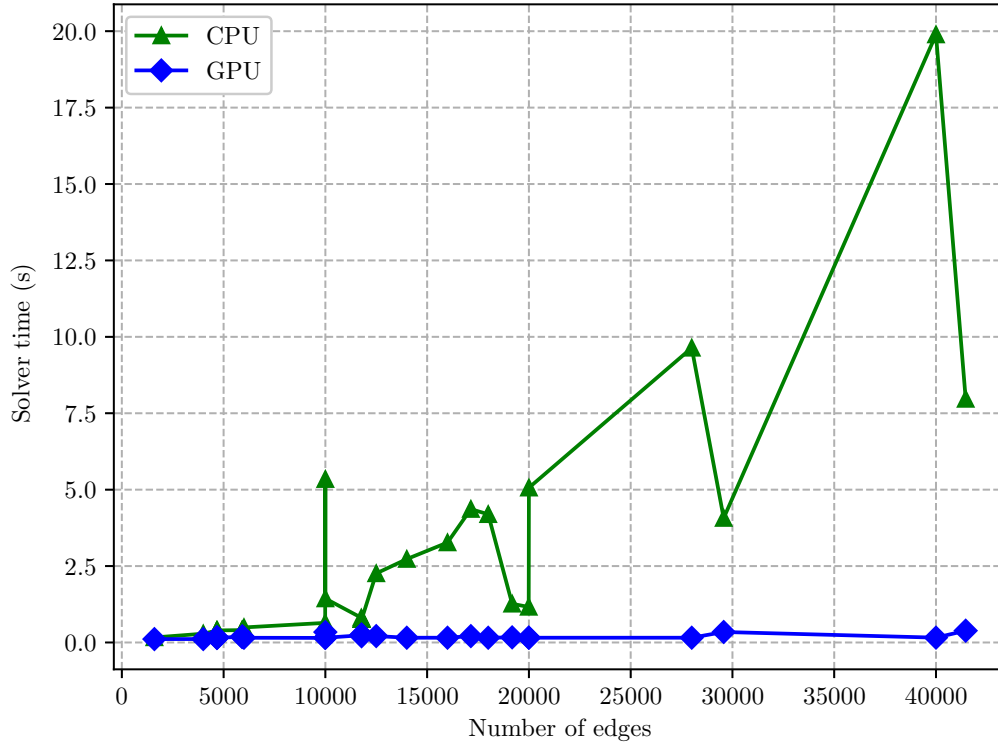


Figure 7.7: Graphic of the performance in seconds of the GPU and CPU for increasingly large graphs.

compared to a CPU based version to be much smaller after further inspection we identified that the solvers had interesting performance depending on the graph structure which prompted further investigation.

We immediately noticed that the performance of both the GPU and CPU was dependent on whether or not the graph was a 2D torus structure. As seen in Fig. 7.8, the speedup (GPU solver time over CPU solver time) is separated by the graph type, green for a torus and red for a non-torus. By examining this figure, we see that the speedup of the torus structure, which is highly regular, steadily increases as the problem size increases. However,

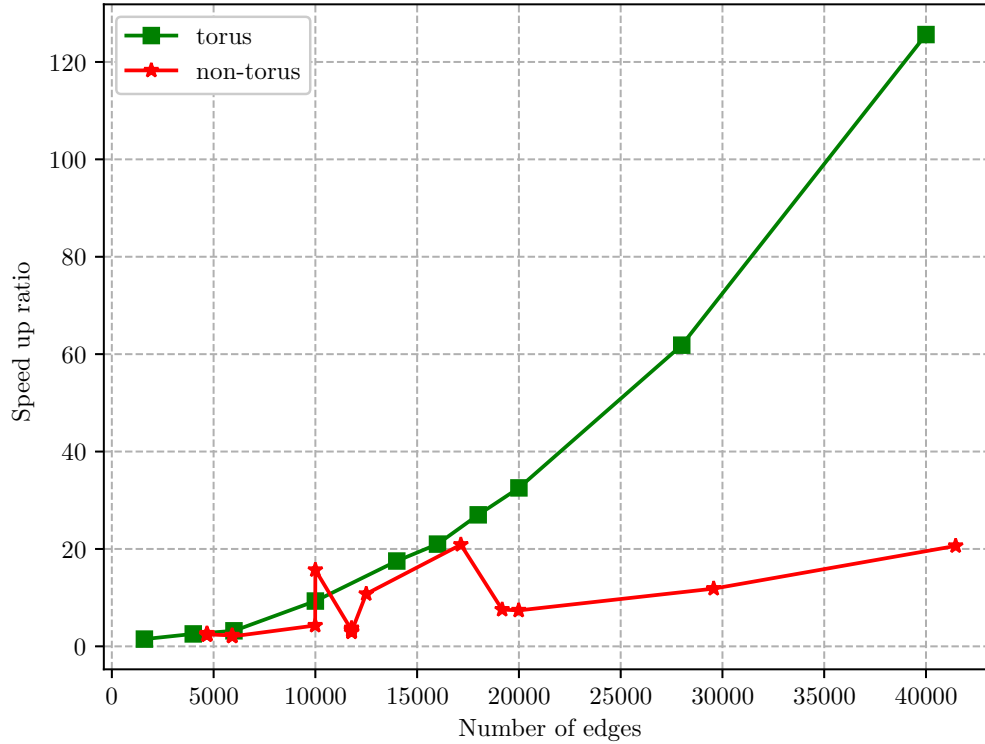


Figure 7.8: The performance results for the GPU against the CPU, separated by the graph type, torus and non-torus.

the non-torus structure, while still showing speedup, is less consistent but generally shows an increase in speedup as the problem size increases.

We further investigate the individual performance of the CPU and GPU by plotting their speed results individually and also separating these results by graph type in Fig. 7.9 and Fig. 7.10 respectively. From these graphs we can make a few observations. For both the CPU and GPU versions, we firstly notice that the speed trends when solving the torus structures is highly consistent with the increasing problem sizes while the non-torus structures have erratic behavior. This can be easily explained by the irregularity of the

graph structure. Each node will have different numbers of edges in the non-torus structure. Coincidentally, on the CPU, the torus structures take longer to solve than the non-torus structures due to the fact that the torus graphs in the g-set benchmark happen to have many more nodes than the non-torus structures(see 7.3 for node numbers). More interesting, we see that the GPU performance on the torus structures is highly efficient with almost no noticeable change in computation time between the graph with 6000 edges and the graph with over 40000 edges. This can be explained by the regularity of the graph structures. This regularity will lead to load balance amongst the GPU threads during computation which allows the GPU performance to really shine. We should expect this nearly constant compute complexity so long as the GPU has sufficient compute resources for the problem size. Any growth in computation time then, could be attributed to host to device memory transfers. However, as soon as the resources become insufficient, certain spin updates will need to be serialized, thus, decreasing performance.

We can also make the observation that the torus graphs are graphs that are similar to nearest neighbor models. That is, they result in highly load balanced workloads. In this way, we can also see that the cost of handling generally connected graph is present in these results by looking at the computation time of non-torus graphs compared to the torus graphs. However, we remark that this cost is still much better than if we had to pre-process the graph by solving a graph embedding problem which would also increase our node counts and finally require post-processing to extract the actual answer from the embedded graph.

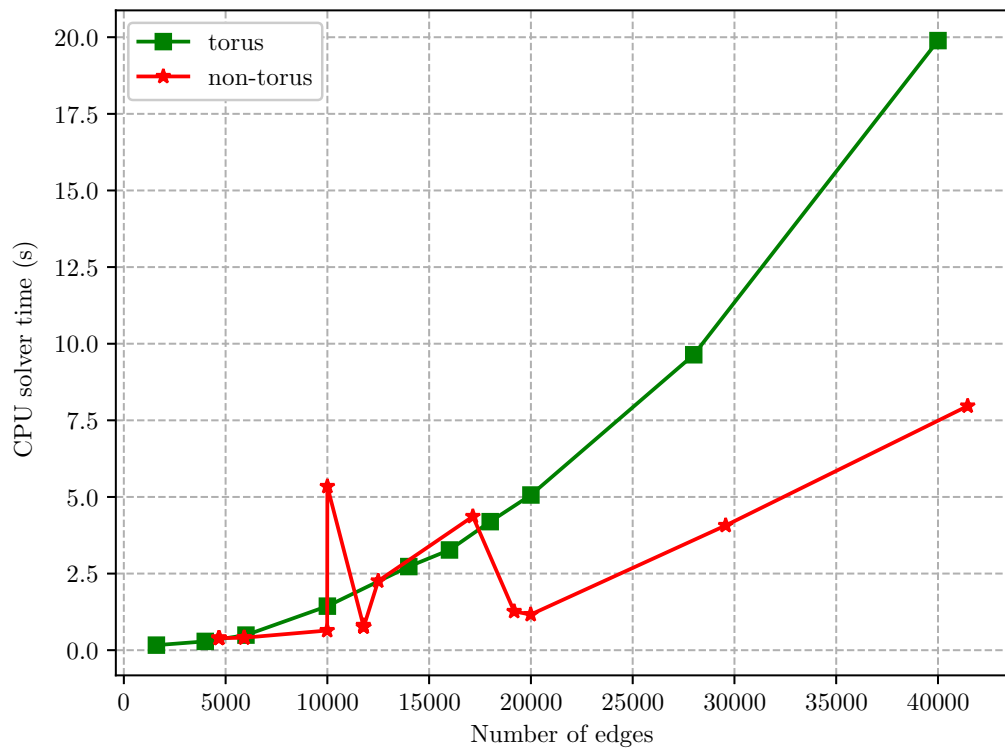


Figure 7.9: CPU performance comparing torus and non-torus graphs.

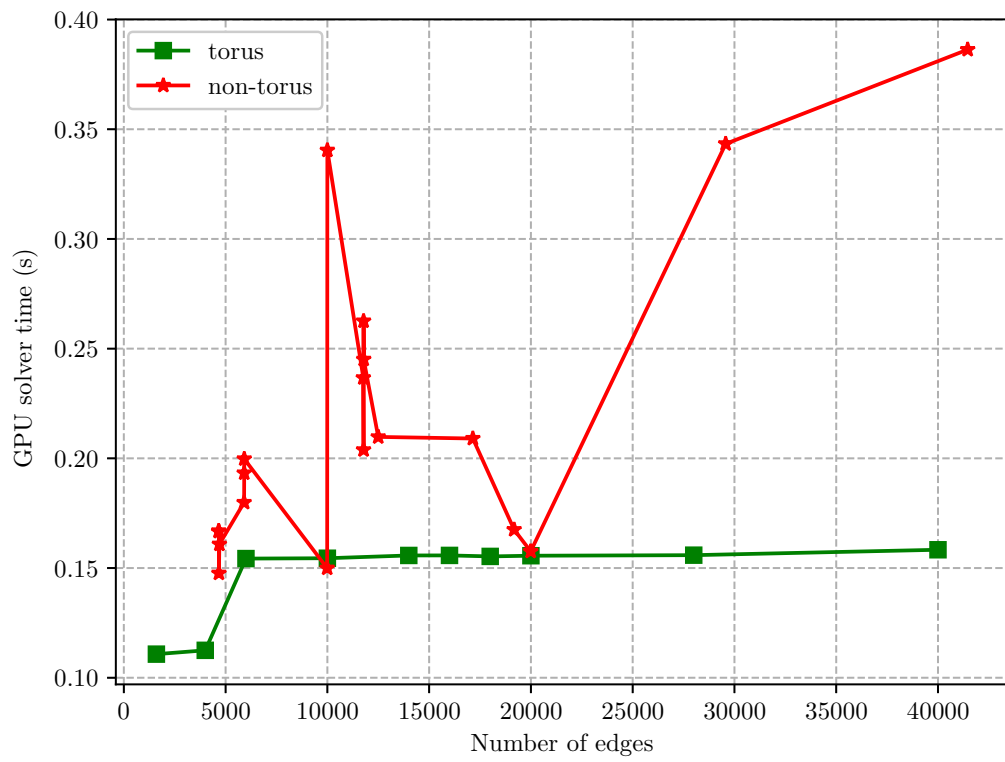


Figure 7.10: GPU performance comparing torus and non-torus graphs.

7.6 Summary

In this chapter we the Ising spin model based computing to solve the max-cut combinatorial optimization problem is presented, which is widely used method for VLSI design automation, on the general purpose GPU (GPGPU). The presented new algorithm is based on the observation that Ising computing by the simulated annealing process is very amenable to fine-grain GPU based parallel computing. GPU-based Ising computing provides clear advantage as a general solver over existing hardware-based Ising computing methods that utilize integrated circuits and FPGAs. We further illustrate how the natural randomness of GPU thread scheduling can be exploited during the annealing process to improve GPU resource utilization. We also showed that GPU-based computing can handle any general Ising graph with arbitrary connections, which was shown to be difficult for FPGA and other hardware based implementation methods. Numerical results show that the proposed GPU Ising max-cut solver can deliver over 2000X speedup over the CPU version of the algorithms over some large examples, which renders this method very appealing for solving many practical VLSI design automation problems.

Chapter 8

GPU-based Ising Computing for Balanced Min-cut Bi-partitioning

In this chapter the GPU-based Ising solver is further expanded to handle the balanced min-cut bi-partitioning problem. Unlike the max-cut solver, the balanced min-cut problem has a global balancing constraint that adds another layer of complexity to its solution. Primarily, this constraint couples each spin glass to every other spin glass causing the graph problem to generate a fully connected graph to solve resulting in exponential scaling of problem complexity. This chapter shows an efficient method for solving this problem and also shows that the newly proposed method achieves results that are competitive to the state-of-the-art graph partitioner METIS [49].

8.1 Ising model for balanced min-cut partitioning

In previous works, the Ising model has been used to solve the max-cut problem [27, 35, 89, 90], but as pointed out in [56], many other NP-complete and NP-hard problems can be mapped to this model as well. The choice to use max-cut in these other works stemmed from the fact that it is trivial to map to the Ising model, making it ideal as a proof of concept problem. However, very little work has been done to map and solve other, more practical problems.

For this reason, this article will show that the balance min-cut graph partitioning problem (min-cut) can be successfully mapped to the Ising model and solved with solution quality rivaling that of state of the art partitioners such as the METIS solver [49]. This problem is highly practical as partitioning is a key algorithm in a wide variety of applications such as VLSI Physical Design.

To formulate the Hamiltonian for the min-cut problem, we must first define min-cut. Given a graph $G(V, E)$, with edge set E and vertex set V , partition G into two subsets (V_1, V_2) such that the edges between V_1 and V_2 are minimized and that $|V_1| = |V_2|$. In other words, we want to assign each vertex in the graph to one of two sets such that we minimize the connections between the two sets while also keeping the size of each set equal. Unfortunately, this problem has been shown to be NP-hard [32].

To facilitate the formulation of the Hamiltonian and subsequent mapping to the Ising model, we need to modify certain constraints so that we can put the problem in a form that is amenable to unconstrained binary optimization. To do this, we will define our Hamiltonian to be the summation of two separate Hamiltonian functions, \mathcal{H}_A and \mathcal{H}_B .

Minimizing the energy of this Hamiltonian will be equivalent to solving the min-cut problem. Furthermore, we will relax the constraint that $|V_1| = |V_2|$ and allow some minor imbalance between each set (which we will now call partitions). Lastly, we will define the spins in our Ising model to be the indication of which partition a spin glass belongs in, i.e., if $\sigma_i = 1$ then $\sigma_i \in V_1$ and if $\sigma_i = -1$, then $\sigma_i \in V_2$.

$$\mathcal{H}_{cut} = \mathcal{H}_{cut,A} + \mathcal{H}_{cut,B} \quad (8.1)$$

The first Hamiltonian $\mathcal{H}_{cut,A}$ is a penalty function that adds energy to our system in (8.1) when the partition sizes are not equal and is defined as:

$$\mathcal{H}_{cut,A} = A\left(\left(\sum_i \sigma_i\right)^2\right) \quad (8.2)$$

Recall that $\sigma_i \in -1, 1$, then we can see from (8.2) that it will equal zero (be minimized) when there are an equal number of negative and positive spins. Otherwise, this Hamiltonian will evaluate to a positive number and add energy to the system. The constant A is a parameter that can be tuned to affect the weight of the penalty incurred from imbalance.

The Hamiltonian $\mathcal{H}_{cut,B}$ is responsible for partitioning the spin glasses such that the edges between the partitions are minimized and is defined as:

$$\mathcal{H}_{cut,B} = B\left(\sum_{\langle i,j \rangle} \frac{1 - \sigma_i \sigma_j}{2}\right) \quad (8.3)$$

In (8.3) we can see that when $\sigma_i = \sigma_j$ the energy is minimal and the effect is opposite when this is not the case. Effectively, this means that each vertex will produce the least amount of energy when it is placed in the same partition that the majority of its

neighbors occupy. The parameter B can be tuned to increase or decrease the penalty of this Hamiltonian.

In general, when choosing the value of A and B , the following rule, as outlined in [56], can be applied:

$$\frac{A}{B} \geq \frac{\min(2\Delta, N)}{8} \quad (8.4)$$

In this equation, Δ is the maximum degree of the graph G and N is the number of spins used to encode the problem, which for our implementation is just the number of vertices in G .

By minimizing the energy of (8.1), we can obtain a balanced partitioning of a graph problem while also achieving minimized cuts between the partitions. Unlike the traditional Ising model, this Hamiltonian contains two sub-Hamiltonians. Furthermore, we also must note that the Hamiltonian in (8.2) requires that each spin-glass have knowledge of every spin glass in the model. This effectively produces a fully connected graph and vastly increases the complexity of the problem, which will be addressed in the following sections.

8.2 GPU-based Ising solver for balanced min-cut problem

8.2.1 Standard implementation

To first validate the GPU implementation of the Ising model solver for the min-cut problem, we construct a parallel algorithm from the one shown in Algorithm 2 but with the modified Hamiltonians for the min-cut problem.

As mentioned before, spin glass updates can be performed in parallel during a sweep. However, the restriction on this is that the updates must be performed indepen-

dently. Furthermore, the updates of spins must happen asynchronously to avoid bi-phasic oscillation. Fortunately, this can be accomplished in the GPU simply by allowing race conditions on read operations. Effectively, this just means that when a spin calculates its own local energy, it will use the current energy of its neighbors which may, or may not, have already minimized their own local energy. To facilitate this, the following algorithm for the GPU implementation has been used:

Algorithm 4 Standard implementation for GPU Ising model annealing for balanced min-cut problem

```

input:  $(M, P_f, \mathbf{S})$ 

initialize ALL  $\sigma_i$  in  $\mathbf{S}$ 

for sweep-id in  $\{1, 2, \dots, M\}$  do

    for all  $\sigma_i \in \mathbf{S}$  in parallel do

         $\sigma_i \leftarrow \operatorname{argmin}(\mathcal{H}_{cut,A}(\sigma_i) + \mathcal{H}_{cut,B}(\sigma_i))$ 

        flip  $\sigma_i$  with probability  $P_f$ 

    end for

    reduce  $P_f$ 

end for

```

In this algorithm the spins for each glass are initialized and M update sweeps are performed. During each sweep, every spin glass, in parallel, chooses the value of σ_i that minimizes the Hamiltonian in (8.1). In effect, this means each spin will read the spin values of all of its connected neighbors (to evaluate (8.3)) as well as the spin values of all spins in the entire graph (to evaluate (8.2)). After a spin glass finishes its update, it will randomly

flip with a probability determined by P_f . Once every spin has updated and decided to flip or not, the flip probability P_f is reduced according to a cooling schedule. This process is then repeated.

The flip probability P_f is initially determined by the number of nodes that should be randomly flipped, which we will call N_{rf} . That is, if there are 500 nodes, and the N_{rf} is 100, then P_f will initially be calculated to be 0.20. When deciding to flip or not, each spin generates a uniformly distributed random number between 0.00 and 1.00. If this number is higher than P_f then the spin will not flip, however, if it is equal or lower, then it will.

Practically, each spin is handled by a single thread in the GPU. If the number of spin glasses exceeds the number of threads available, then threads will be assigned multiple spin glasses to update. For the random number generation, the CUDA cuRAND library is used which allows for efficient random number generation in parallel. We also note, as mentioned previously, that each spin update should not be correlated. For this reason, updates are done asynchronously. In effect, this means that a spin glass will check the status of its neighbor spins during an update but it will not be guaranteed that its neighbors have not already finished their own update operation.

8.2.2 Globally Decoupled Ising implementation

While Algorithm 4 does successfully calculate a balanced min-cut partition, as we will show in the results section, it is also heavily constrained by the direct computation of $\mathcal{H}_{cut,A}$ as shown in (8.2). This means that for each spin glass in the model, it must traverse the entire model to assess the status of every spin glass and determine the balancing penalty. This is obviously a heavy computational step and will drastically limit the performance of

the algorithm as the problem size increases. *This is actually is one of major challenges for Ising based computing as many constraints will lead to dense or complete Ising graphs.*

One of the major contributions of this work is to find a way to mitigate this challenging problem. To mitigate this problem, we have to find a better way to deal with the balance constraints instead of using the standard Ising model. In this way, we still keep the spin updates local (each spin glass no longer has to traverse the entire model), which is critical to maintaining the efficiency and scalability for the GPU-based computing, while making the global connections decoupled. Therefore, we propose to use a Globally Decoupled Ising (GDI) solver.

Algorithm 5 Globally Decoupled Ising (GDI) implementation for GPU Ising model annealing for balanced min-cut problem

input: (M, P_f, \mathbf{S})

initialize ALL σ_i in \mathbf{S}

$$G = \sum_i \sigma_i$$

for sweep-id **in** $\{1, 2, \dots, M$ **do**

for all $\sigma_i \in \mathbf{S}$ **in parallel do**

$$\sigma_i \leftarrow \operatorname{argmin}(\mathcal{H}_{cut,B}(\sigma_i) + A(G + \sigma_i)^2)$$

$$G = \operatorname{atomicAdd}(G + \sigma_i)$$

 flip σ_i with probability P_f

end for

reduce P_f

end for

Specifically, in Algorithm 5, we pre-compute the global balance before starting the first update sweep and store this value in a global variable G . Then, as each thread performs its update during a sweep, it uses this value to determine the balancing penalty it's spin glass may incur. After deciding which partition will minimize the spin glass local energy, the thread updates the G value so that all other spins glasses have knowledge of the new balance. It should be noted that because this is happening in parallel, the balance that some threads see may not be the actual balance when that thread updates the spin glass it is responsible for. Furthermore, an atomic addition is used to update G which will incur some computational penalty (but at much less cost than the method in Algorithm 4). This atomic operation ensures the validity and integrity of the G value at the end of each sweep as it eliminates "data race" between threads updating this value.

8.2.3 Further discussion and comparison

The primary difference in the two algorithms is the modification to the calculation of the balancing Hamiltonian. The major effect is that we no longer require each node to traverse the entire graph at each update step to calculate the global balance. Rather, the global balance is pre-computed and then individually read, while modified (through atomic operations) by one node at a time.

With respect to implementation, we notice that the standard version requires a higher random flip probability than the enhanced version to achieve good cut results. The reason for this is that the enhanced version naturally introduces more noise, or randomness, to the update scheme while the standard version is more constrained by the global update. That is, it is less willing to violate the global constraint as each node has nearly perfect

knowledge of the state of the global balance. For this reason, during implementation, we typically need a $5X$ increase to the value of P_f when using the standard implementation.

8.3 Experimental results and discussions

In this section, we present the experimental results showing the quality of our parallel GPU-based Simulated Quantum Annealing solver for the balanced min-cut problem. The CPU-based solution is done using a Linux server with 2 Xeon processors, each having 8 cores (2 threads per core) and a total of 32 threads, and 72 GB of memory. On the same server, we also implement the GPU-based solver using the Nvidia Tesla K40c GPU which has 2880 CUDA cores and 12 GB of memory. Test problems from the G-set benchmark [5] are used for testing.

8.3.1 Solution time study

We firstly investigate the performance of the standard Ising solver and the globally decoupled method developed in the paper, followed by discussion of its performance compared to the state of the art partitioning software.

The direct Ising solver implementation of the min-cut partitioning problem leads to a complete graph requiring a global update for each spin glass in the model. In other words, when each node updates during an annealing sweep, it must visit every other node in the graph. To address this, we proposed the balance constraint efficient annealing algorithm in 5. In the GDI version, we mitigate the issue of global update by utilizing a global variable to store the balance of the graph partition which is updated using atomic operations by each

Table 8.1: Summary of results for the standard (std.) and Globally decoupled (GDI) Ising solver compared with the METIS results. Graphs with less than 1000 nodes are omitted

Graph-ID	# nodes	# edges	Density	t_{Metis}	$t_{\text{Ising,GDI}}$	$t_{\text{Ising,std.}}$	cut_{Metis}	$cut_{\text{Ising,GDI}}$	$cut_{\text{Ising,std.}}$	bal_{Metis}	$bal_{\text{Ising,GDI}}$	$bal_{\text{Ising,std.}}$
G70	10000	9999	2.0E-4	1.6E-2	0.34929	31.88844	487	507	555	5	0	0
G81	20000	40000	2.0001E-4	1.6E-2	0.15046	22.40281	210	240	242	5	0	0
G77	14000	28000	2.8573E-4	1.2E-2	0.15686	15.67784	220	212	232	3	0	0
G67	10000	20000	4.0004E-4	0.08	0.15579	11.20150	216	216	244	1	0	0
G72	10000	20000	4.0004E-4	0.08	0.15707	11.23208	216	212	230	1	0	0
G66	9000	18000	4.4449E-4	8.0E-3	0.15810	10.08537	196	220	242	2	0	0
G65	8000	16000	5.0006E-4	8.0E-3	0.15766	9.01416	178	168	178	3	0	0
G62	7000	14000	5.7151E-4	8.0E-3	0.15730	7.87871	144	146	148	1	0	0
G60	7000	17148	7.0002E-4	2.8E-2	0.21494	9.31080	3330	3115	3321	2	0	0
G57	5000	10000	8.0016E-4	4.0E-3	0.15679	5.64559	140	110	252	0	0	0
G55	5000	12498	1.0001E-3	0.02	0.21558	6.45212	2463	2308	2391	0	0	0
G48	3000	6000	1.33378E-3	4.0E-3	0.15601	3.45138	120	124	194	1	0	0
G49	3000	6000	1.33378E-3	4.0E-3	0.15632	3.45322	74	60	136	0	0	0
G50	3000	6000	1.33378E-3	0	0.15617	3.45544	58	50	92	1	0	0
G64	7000	41459	1.6924E-3	3.2E-2	0.38691	5.89811	9946	9787	9967	2	2	0
G32	2000	4000	2.001E-3	0	0.12023	2.30673	50	40	54	1	0	0
G33	2000	4000	2.001E-3	0	0.12019	2.30543	54	50	78	1	0	0
G34	2000	4000	2.001E-3	0	0.12044	2.31117	112	86	150	1	0	0
G58	5000	29570	2.366E-3	2.4E-2	0.34457	3.98248	7226	6921	7302	1	0	0
G36	2000	11766	5.8859E-3	8.0E-3	0.23601	1.63955	2896	2722	2898	0	0	0
G35	2000	11778	5.8919E-3	1.2E-2	0.20426	1.77944	2942	2771	2841	1	0	0
G38	2000	11779	5.8924E-3	1.2E-2	0.26296	1.63863	2866	2688	2801	1	0	0
G37	2000	11785	5.8954E-3	1.2E-2	0.24608	1.63948	2921	2781	2834	1	0	0
G22	2000	19990	0.01	1.2E-2	0.15961	0.87029	6925	6739	6803	0	0	0
G23	2000	19990	0.01	1.6E-2	0.15989	0.86660	6946	6702	6743	0	0	0
G24	2000	19990	0.01	1.6E-2	0.16047	0.96611	7022	6719	6809	1	0	0
G28	2000	19990	0.01	1.2E-2	0.16016	0.86641	6961	6753	6781	0	0	0
G30	2000	19990	0.01	1.6E-2	0.16170	0.86767	6978	6702	6774	1	0	0
G31	2000	19990	0.01	1.2E-2	0.16086	1.00123	6957	6683	6798	1	0	0
G51	1000	5909	1.183E-2	4.0E-3	0.18105	0.92766	1513	1383	1458	0	0	0
G53	1000	5914	1.184E-2	4.0E-3	0.19386	0.92421	1498	1366	1368	0	0	0
G52	1000	5916	1.1844E-2	4.0E-3	0.19584	0.92683	1446	1394	1399	0	0	0
G54	1000	5916	1.1844E-2	4.0E-3	0.20007	0.92612	1465	1356	1430	0	0	0
G43	1000	9990	0.02	8.0E-3	0.15092	0.51411	3542	3350	3382	0	0	0
G44	1000	9990	0.02	8.0E-3	0.15365	0.51538	3565	3361	3402	0	0	0
G45	1000	9990	0.02	8.0E-3	0.15322	0.51169	3522	3347	3397	0	0	0
G46	1000	9990	0.02	4.0E-3	0.15210	0.51384	3573	3353	3386	0	0	0
G47	1000	9990	0.02	8.0E-3	0.15191	0.51398	3520	3350	3396	0	0	0

node. This means each node only needs to perform a single read and atomic add operation on this variable.

In Fig 8.1, the graph problems solved are sorted from lowest to highest density where density is calculated as $density = (2 \times \#edges) / (\#nodes \times (\#nodes - 1))$. The nature of the G-set graphs used are such that low density graphs have many more nodes and edges than the high density graphs. For this reason, the graph can be looked at as being sorted from large complexity to smallest complexity. As we can see, the direct implementation quickly increases in computation time as the problem become more and

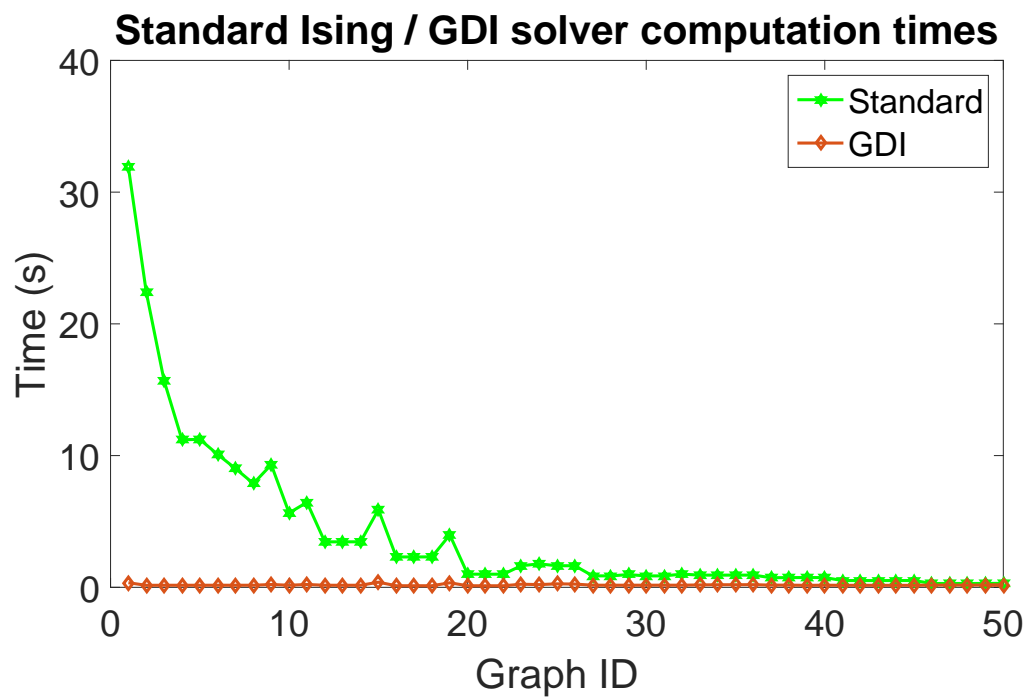


Figure 8.1: Performance gains of the GDI solver min-cut algorithm.

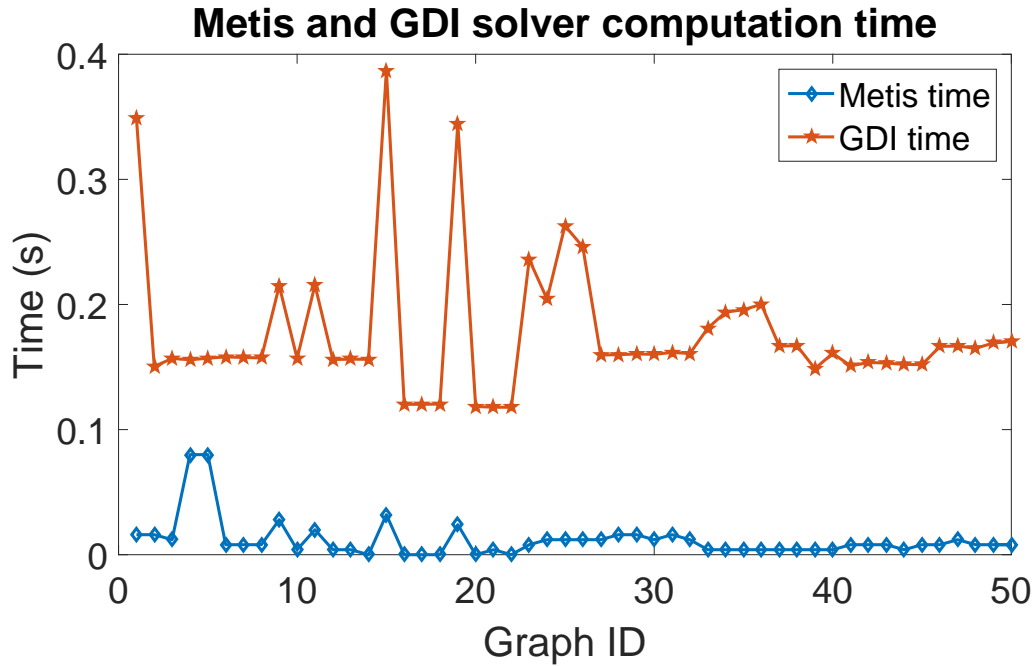


Figure 8.2: Performance comparison of the GDI GPU Ising solver and METIS.

more complex. Comparatively, the GDI version appears constant in these results as the computations times are not comparable.

We further show the performance results of the GDI solver algorithm and the performance of METIS, widely considered the gold standard for partitioning software.

We can see from Fig. 8.2 that, while METIS is still able to beat the proposed solver in terms of speed, the proposed Ising solver is quite close and comparable in time with all solutions taking less than a second, even for the larger graphs.

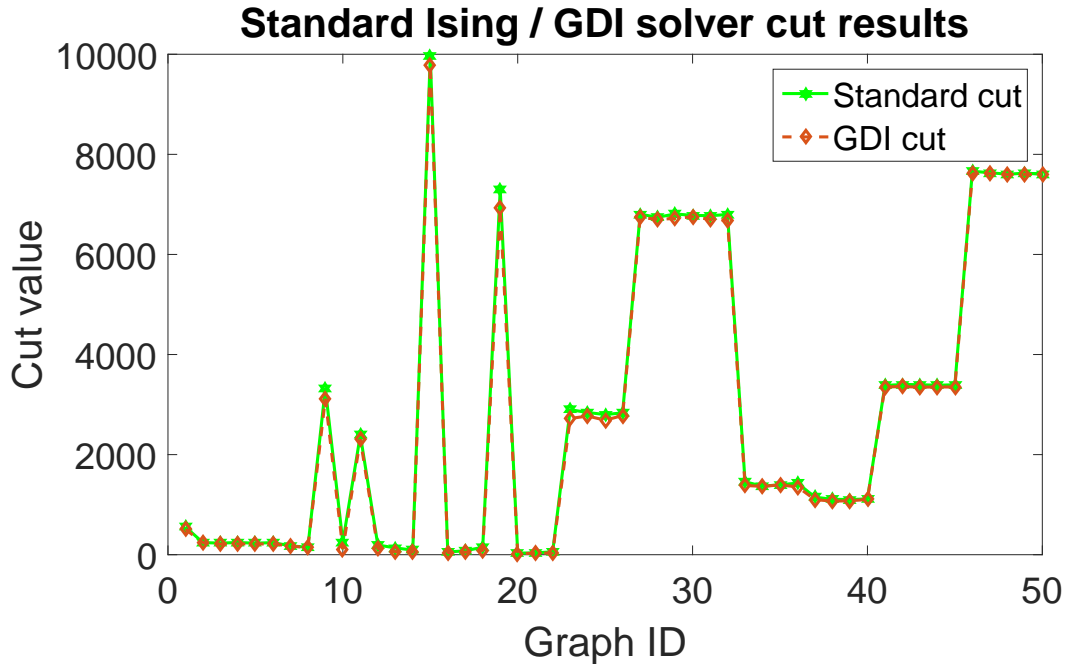


Figure 8.3: Solution quality of the proposed GPU-based Ising solvers.

8.3.2 Solution quality study

To study the solution quality of the proposed method, we first show that the solution quality of the direct implementation of the GPU-based Ising solver and the GDI solver are similar.

As seen from Fig. 8.3 the solution quality of the GDI method not only achieves similar solution results, it also consistently achieves cut values marginally lower than the standard implementation.

To measure the solution quality produced by the proposed method, we compare balanced min-cut partitioning results of the proposed method with the results produced by METIS. We omit the solution quality results of the direct implementation as they are

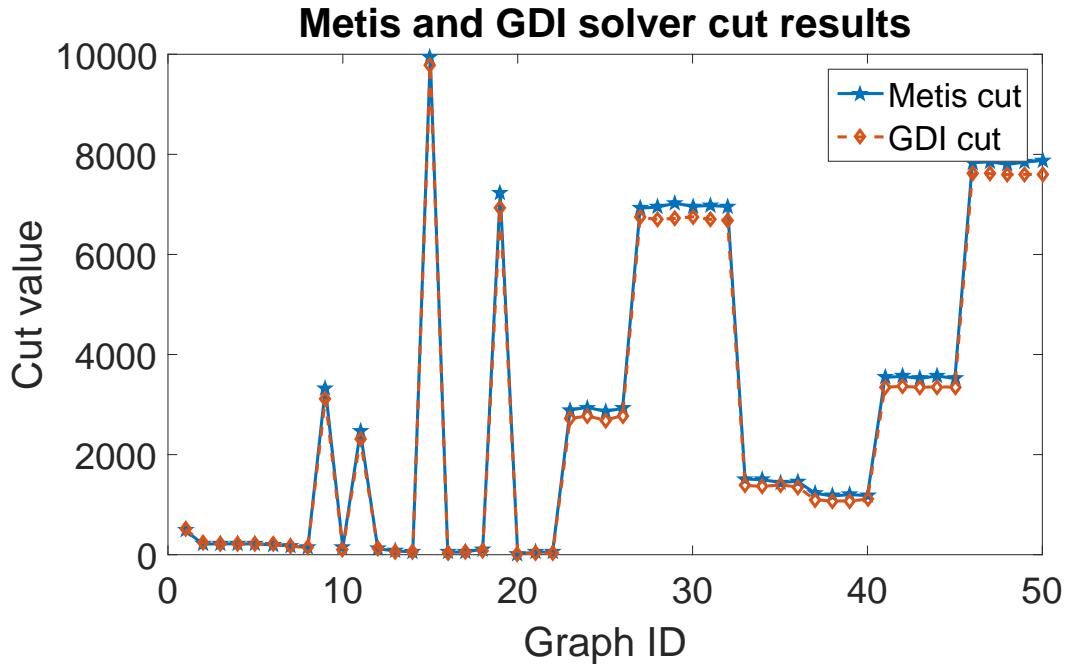


Figure 8.4: Cut quality results for the G-set benchmark problems.

similar to the GDI method proposed. For fairest comparison, we ensured both METIS and the GPU-based Ising solver used a highly constrained balance criteria. For each graph, we run both solvers 10 times and use the best solution quality found. The quality is measured by comparing the cut values of the solvers. Cut values are defined as the number of edges that connect one partition to the other.

As seen from Fig. 8.4, the proposed Ising solver GPU method achieves the same or better quality results for almost every graph tested. Of the 51 graphs, the Ising solver method achieved worse quality in only 5 graphs and produced the exact same quality in 1 graph. Even then, the Ising solver results were very close to METIS. We note that the all of the graphs where Ising solver achieved worse quality were very sparse graphs, i.e.,

their graph density was very low, which suggests that utilization of the Ising model can be difficult for these types of graphs.

While the balancing constraint on METIS was set to be as restrictive as possible, 22 of the the 51 graphs tested had imbalanced partitions. In contrast, the proposed GDI solver achieved perfect balance for all but one graph tested. We also remark that the balances achieved by METIS were still quite good with the worst imbalance being only 5 nodes.

These results, as summarized in Table 8.1, show that the proposed method is able to achieve better solution quality than the state of the art METIS solver, both in terms of balanced partitions and minimized cut values. Furthermore, even for large and dense graphs, the proposed solver finished in a time comparable to METIS.

8.4 Summary

This chapter presented a GPU-based Ising spin glass model solver applied to the balanced min-cut graph partitioning problem. The work presented the annealing solution method for solving Unconstrained Quadratic Binary Optimization problems while additionally showing the method for mapping the min-cut problem to this model. A standard GPU implementation is presented in addition to a Globally Decoupled Ising annealing algorithm which mitigates the costly global updates during each annealing sweep. Numerical results show that both the standard and enhanced annealing algorithms achieve high quality, and nearly perfectly balanced, partitioning results that compete with and exceed the partitioning quality achieved by the state of the art partitioning solver, METIS. Furthermore, the

proposed GDI method can produce results much faster than the standard method, resulting in computation times comparable to the METIS solver. Experimental results show that the proposed Ising-based min-cut partitioning method outperforms the state of art partitioning tool, METIS, on G-set graph benchmark in terms of partitioning quality with similar CPU/GPU times.

Chapter 9

Conclusion

The continued scaling of process technology and advances in hardware and architecture have created a need for more advanced simulation techniques. Where previously certain assumptions about simulating physical effects, such as Electromigration induced aging, causes negligible errors or over estimations, current advanced technology requires the tightest margins. This dissertation focuses on the presentation of new simulations techniques that take advantage of the newest physical models and acceleration techniques.

9.1 Summary of Contributions

9.1.1 Reliability

To address the overly conservative nature of the classical EM sign-off methodology, this dissertation presents a method for solving advanced physics-based EM models. Specifically, a numerical method, the Finite Difference Method, was used to solve the dynamic stress evolution PDE. This was the first general multi-branch EM simulator that considered

both void nucleation and growth phases while also incorporating void saturation volume. Furthermore, a model reduction technique was utilized to accelerate the numerical solution, resulting in a simulation method, *FastEM*, with a 90X performance gain making the solver more practical for use in real world scenarios. Furthermore, *FastEM* is the only accelerated numerical solution for EM that can consider both temperature and current density transients in multi-branch interconnects. The resulting work enables less conservative long-term reliability assessment leading to less guardbanding and design iterations due to reliability sign-off failure while also being efficient enough for practical use.

9.1.2 Hardware security

Hardware security is an increasingly important topic as more attack surfaces are developed every year requiring more research and development to find and mitigate these avenues of attack. Furthermore, due to the proliferation of electronic components across the globe, illicit markets have become more common and also require design houses to develop novel techniques to help mitigate IP theft and counterfeiting. Using the simulation techniques and advanced EM models, this work address both of these areas. Firstly, a novel reliability-based hardware Trojan was developed that utilizes the EM aging effect and advanced failure modes. The EM-based project requires actual IC usage and aging to become effective making it particularly difficult to detect using functional testing and can be implemented as either a Trojan trigger or payload. Next, the EM-aging effect was used to create an on-chip aging timer. Unlike previous EM-aging timers, this mother uses a self-calibration technique and also relies on the RC delay degradation due to EM-failure rather than voltage drop. This enables a highly area efficient aging sensor, that has a wide

range of age sensing in a single sensor, and does not require multiple sensors or wires in each sensor.

9.1.3 Ising computing

This dissertation explores the Ising spin glass model as a solution methodology for hard combinatorial optimization problems using the general purpose GPU (GPGPU). The Ising model is a mathematical model of ferromagnetism in statistical mechanics. Ising computing finds a minimum energy state for the Ising model which essentially corresponds to the expected optimal solution of the original problem. Many combinatorial optimization problems can be mapped into the Ising model. In this manuscript, parallel GPGPU based solutions are developed to solve max-cut as well as the balanced min-cut graph partitioning solvers. The max-cut solver achieved a large performance gain over the non-GPU annealing algorithm while also beating CPLEX in terms of accuracy. Unlike previous methods that solve the Ising model, the presented model can work directly on a general graph problem while avoiding the use of complicated graph embedding, which is another NP-hard problem. The min-cut problem presented a unique challenge as it involves a global constraint that results in a fully connected graph when a problem is mapped to the Ising model. To address this, the work develops a Globally Decoupled Ising (GDI) solver which mitigates this issue by pre-calculating global balance and keeping a running total of balances. The resulting algorithm is faster than the CPU-based method and achieves accuracy on par with the state-of-the-art METIS graph partitioning software.

Bibliography

- [1] Comsol multiphysics. <https://www.comsol.com/> [Oct. 16, 2013].
- [2] Defense Advanced Research Projects Agency, "Integrity and Reliability of Integrated Circuits (IRIS). <https://www.darpa.mil/program/integrity-and-reliability-of-integrated-circuits>.
- [3] Defense Advanced Research Projects Agency, "Trusted Integrated Circuits (TRUST)". <https://www.darpa.mil/program/trusted-integrated-circuits>.
- [4] Intelligence Advanced Research Projects Activity, "Trusted Integrated Chips (TIC)". <https://www.iarpa.gov/index.php/research-programs/tic/baa>.
- [5] G-set, 2003. <http://web.stanford.edu/yyye/yyye/Gset/>.
- [6] D-Wave Computer, 2018. <http://www.dwavesys.com>.
- [7] U.S. Department of Commerce, Defense Industrial Base Assessment: Counterfeit Electronics,, Jan. 2010.
- [8] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. The Society for Industrial and Applied Mathematics (SIAM), 2005.
- [9] Brian Bailey. Thermally Challenged. *Semiconductor Engineering*, pages 1–8, December 2013.
- [10] Francisco Barahona. On via minimization. *IEEE Transactions on Circuits and Systems*, 37(4):527–530, Apr 1990.
- [11] Francisco Barahona, Martin Grtschel, Michael Jnger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988.
- [12] Fransisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.
- [13] Lev Yu. Barash, Martin Weigel, Michal Borovský, Wolfhard Janke, and Lev N. Shchur. Gpu accelerated population annealing algorithm. *Computer Physics Communications*, 220:341 – 350, 2017.

- [14] J. R. Black. Electromigration-A Brief Survey and Some Recent Results. *IEEE Trans. on Electron Devices*, 16(4):338–347, 1969.
- [15] I. A. Blech. Electromigration in thin aluminum films on titanium nitride. *Journal of Applied Physics*, 47(4):1203–1208, 1976.
- [16] Benjamin Block, Peter Virnau, and Tobias Preis. Multi-gpu accelerated multi-spin monte carlo simulations of the 2d ising mode. *Computer Physics Communications*, 181(9):1549–1546, 2010.
- [17] Sergio Boixo, Froels F. Ronnow, Sergei V. Isakov, Zihui Wang, David Wecker, Daniel A. Lidar, John M. Martinis, and Matthias Troyer. Evidence for quantum annealing with more than one hundred qubits. *Nature Physics*, 2014.
- [18] Endre Boros, Peter L. Hammer, and Gabriel Tavares. Local search heuristics for quadratic unconstrained binary optimization (qubo). *Journal of Heuristics*, 13(2):99–132, April 2007.
- [19] Jun Cai, William G. Macread, and Aidan Roy. A practical heuristic for finding graph minors. 2014.
- [20] D. Chardonnal. Impacts of counterfeiting and piracy to reach US\$1.7 trillion by 2015, 2011.
- [21] S. Chatterjee, M. B. Fawaz, and F. N. Najm. Redundancy-Aware Electromigration Checking for Mesh Power Grids. In *Proc. IEEE/ACM Int. Conf. on Computer Aided Design (ICCAD)*, pages 540–547, 2013.
- [22] S. Chatterjee, V. Sukharev, and F. N. Najm. Power Grid Electromigration Checking using Physics-Based Models. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 37(7):1317–1330, 2018.
- [23] H.-B. Chen, S. X.-D. Tan, X. Huang, and V. Sukharev. New electromigration modeling and analysis considering time-varying temperature and current densities. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 352–357. IEEE, Jan. 2015.
- [24] Jun-Dong Cho, S. Raje, and M. Sarrafzadeh. Fast approximation algorithms on max-cut, k-coloring, and k-color ordering for vlsi applications. *IEEE Transactions on Computers*, 47(11):1253–1266, Nov 1998.
- [25] J. J. Clement. Reliability Analysis for Encapsulated Interconnect Lines under DC and Pulsed DC Current Using A Continuum Electromigration Transport Model. *Journal of Applied Physics*, 82(12):5991–6000, 1997.
- [26] C. Cook, Z. Sun, T. Kim, and S. X.-D. Tan. Finite difference method for electromigration analysis of multi-branch interconnects. In *Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 1–4. IEEE, June 2016.

- [27] C. Cook, H. Zhao, T. Sato, M. Hiromoto, and S. X.-D. Tan. GPU-based ising computing for solving max-cut combinatorial optimization problems. *Integration, the VLSI Journal*, 2019. In press.
- [28] RL De Orio, Hajdin Ceric, and Siegfried Selberherr. Physically based models of electro-migration: From black’s equation to modern tcad models. *Microelectronics Reliability*, 50(6):775–789, 2010.
- [29] E. Demircan and M. D.Shroff. Model based method for electro-migration stress determination in interconnects. In *2014 IEEE International Reliability Physics Symposium*, pages IT.5.1–IT.5.6, June 2014.
- [30] Vasil S. Denchev, Sergio Boixo, Sergei V. Isakov, Nan Ding, Ryan Babbush, Vadim Smelyanskiy, John Martinis, and Hartmut Neven. What is the computational value of finite-range tunneling? *Phys. Rev. X*, 6:031015, Aug 2016.
- [31] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science*, 292(5516):472–475, 2001.
- [32] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
- [33] E. Thomas Gilmore, Preston D. Frazier, Isaac J. Collins, William Reid, and M. F. Chouikha. Infrared analysis for counterfeit electronic parts detection and supply chain validation. *Environment Systems and Decisions*, 33(4):477–485, Dec 2013.
- [34] U. Guin, D. DiMase, and M. Tehranipoor. Counterfeit Integrated Circuits: Detection, Avoidance, and the Challenges Ahead. *Journal of Electronic Testing*, 30:9–23, Feb. 2014.
- [35] Hidenori Gyoten, Masayuki Hiromoto, and Takashi Sato. Area efficient annealing processor for ising model without random number generator. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science(IEICE)*, E101.D:314–323, 2018.
- [36] Hidenori GYOTEN, Masayuki HIROMOTO, and Takashi SATO. Enhancing the solution quality of hardware ising-model solver via parallel tempering. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 1–8, New York, New York, USA, 2018. ACM Press.
- [37] K. He, X. Huang, and S. X.-D. Tan. Em-based on-chip aging sensor for detection and prevention of counterfeit and recycled ics. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 146–151. IEEE, Nov. 2015.
- [38] K. He, X. Huang, and S. X.-D. Tan. Em-based on-chip aging sensor for detection of recycled ics. *IEEE Design & Test*, 33(5):56–64, 2016.

- [39] C.-K. Hu, D. Canaperi, S. T. Chen, L. M. Gignac, B. Herbst, S. Kaldor, M. Krishnan, E. Liniger, D. L. Rath, D. Restaino, R. Rosenberg, J. Rubino, S.-C. Seo, A. Simon, S. Smith, and W.-T. Tseng. Effects of overlayers on electromigration reliability improvement for cu/low k interconnects. In *Reliability Physics Symposium Proceedings, 2004. 42nd Annual. 2004 IEEE International*, pages 222–228. IEEE, 2004.
- [40] X. Huang, V. Sukharev, T. Kim, and S. X.-D. Tan. Electromigration recovery modeling and analysis under time-dependent current and temperature stressing. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 244–249. IEEE, Jan. 2016.
- [41] X. Huang, V. Sukharev, and S. X.-D. Tan. Dynamic electromigration modeling for transient stress evolution and recovery under time-dependent current and temperature stressing. *Integration, the VLSI Journal*, 55:307–315, September 2016.
- [42] X. Huang, T. Yu, V. Sukharev, and S. X.-D. Tan. Physics-based electromigration assessment for power grid networks. In *Proc. Design Automation Conf. (DAC)*, pages 1–6. IEEE, June 2014.
- [43] IBM. Ilog cplex optimizer, 2015. <http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer>.
- [44] International technology roadmap for semiconductors (ITRS), 2015 edition, 2015. <http://public.itrs.net>.
- [45] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze. Split-fabrication obfuscation: Metrics and techniques. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 7–12, May 2014.
- [46] R. Jarvis and M. McIntyre. Split manufacturing method for advanced semiconductor circuits. U.S. Patent 2004 0102019 A1, May 2004.
- [47] JEDEC. Failure Mechanisms and Models for Semiconductor Devices. In JEDEC Publication JEP122-A, Jedec Solid State Technology Association, 2002.
- [48] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton an K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, J. Wang S. Uchaikin, B. Wilson, and G. Rose. Quantum annealing with manufactured spins. *Nature*, 473:194 EP–, 2011.
- [49] G. Karypis and V. Kumar. A fast high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.
- [50] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
- [51] M. A. Korhonen, P. Bo/rgesen, K. N. Tu, and C.-Y. Li. Stress evolution due to electromigration in confined metal lines. *Journal of Applied Physics*, 73(8):3790–3799, 1993.

- [52] David Landau and Kurt Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, New York, NY, USA, 2005.
- [53] R. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.
- [54] M Lin and A Oates. An electromigration failure distribution model for short-length conductors incorporating passive sinks/reservoirs. *IEEE Transactions on Device and Materials Reliability*, 13(1):322–326, March 2013.
- [55] Zhijian Lu, Wei Huang, J Lach, M Stan, and K Skadron. Interconnect lifetime prediction under dynamic stress for reliability-aware design. In *Proc. IEEE/ACM Int. Conf. on Computer Aided Design (ICCAD)*, pages 327–334. IEEE, November 2004.
- [56] Andrew Lucas. Ising formulations of many np problems. *Frontiers in Physics*, 2:5, 2014.
- [57] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.
- [58] V. Mishra and S. S. Sapatnekar. The Impact of Electromigration in Copper Interconnects on Power Grid Integrity. In *Proc. Design Automation Conf. (DAC)*, pages 1–6, 2013.
- [59] J. A. Mydosh. *Spin Glasses: An Experimental Introduction*. Taylor & Francis, 1993.
- [60] S.R. Nassif. Power grid analysis benchmarks. In *2008 Asia and South Pacific Design Automation Conference*, March 2008.
- [61] N. Nishimori. *Statistical Physics of Spin Glasses and Information Processing: An Introduction*. Oxford University Press, 2001.
- [62] NVIDIA. CUDA C programming guide. docs.nvidia.com/cuda/cuda_c_programming_guide/index.html, March 2018.
- [63] NVIDIA Corporation. CUDA (Compute Unified Device Architecture), 2011. http://www.nvidia.com/object/cuda_home.html.
- [64] NVIDIA Corporation. NVIDIA’s next generation CUDA compute architecture: Kepler gk110/210, 2014. White Paper.
- [65] A. Odabasioglu, M. Celik, and L. Pileggi. PRIMA: Passive reduced-order interconnect macro-modeling algorithm. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, pages 645–654, 1998.
- [66] Necati Ozisik. *Finite Difference Methods in Heat Transfer*. CRC Press, April 1994.
- [67] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

- [68] S. Sadiqbatcha, Z. Sun, and S. X. D. Tan. Accelerating electromigration aging: Fast failure detection for nanometer ics. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2019.
- [69] Yuriy Shiyonovskii, Francis G. Wolff, Christos A. Papachristou, Daniel J. Weyer, and W. Clay. Exploiting semiconductor properties for hardware trojans. *CoRR*, abs/0906.3834, 2009.
- [70] Taber Smith, Vikas Mehrotra, and David White. Dummy fill for integrated circuits. U.S. Patent 7 380 220 B2, May 2008.
- [71] Aswin Sreedhar, Sandip Kundu, and Israel Koren. On reliability trojan injection and detection. *Journal on Low Power Electronics*, 8(5):674–683, Dec. 2012.
- [72] V. Sukharev, X. Huang, and S. X.-D. Tan. Electromigration Induced Stress Evolution Under Alternate Current and Pulse Current Loads. *Journal of Applied Physics*, 118:034504–1–034504–10, 2015.
- [73] V. Sukharev, A. Kteyan, and X. Huang. Postvoiding stress evolution in confined metal lines. *IEEE Transactions on Device and Materials Reliability*, 16(1):50–60, 2016.
- [74] Z. Sun, E. Demircan, M. D. Shroff, T. Kim, X. Huang, and S. X.-D. Tan. Voltage-Based Electromigration Immortality Check for General Multi-Branch Interconnects. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 1–7, Nov. 2016.
- [75] Z. Sun, S. Sadiqbatcha, H. Zhao, and S. X.-D. Tan. Accelerating Electromigration Aging for Fast Failure Detection for Nanometer ICs. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 623–630, Jan. 2018.
- [76] Z. Suo. *Reliability of Interconnect Structures*, volume 8 of *Comprehensive Structural Integrity*. Elsevier, Amsterdam, 2003.
- [77] S. X.-D. Tan, H. Amrouch, T. Kim, Z. Sun, C. Cook, and J. Henkel. Recent advances in EM and BTI induced reliability modeling, analysis and optimization. *Integration, the VLSI Journal*, 60:132–152, Jan. 2018.
- [78] Sheldon X.-D. Tan and L. He. *Advanced Model Order Reduction Techniques in VLSI Design*. Cambridge University Press, 2007.
- [79] M. Tehranipoor and F. Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design Test of Computers*, 27(1):10–25, 2010.
- [80] Mohammad Tehranipoor, Hassan Salmani, and Xuehui Zhang. *Integrated Circuit Authentication*. Springer, 2014.
- [81] Trust-HUB. <http://trust-hub.org/home>.
- [82] Hayato Ushijima-Mwesigwa, Christian F. A. Negre, and Susan M. Mniszewski. Graph partitioning using quantum annealing on the d-wave system. In *Proceedings of the Second International Workshop on Post Moores Era Supercomputing, PMES’17*, pages 22–29, New York, NY, USA, 2017. ACM.

- [83] K. Vaidyanathan, B.P. Das, E. Sumbul, R. Liu, and L. Pileggi. Building trusted ICs using split fabrication. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 1–6, May 2014.
- [84] J. Villasenor and M. Tehranipoor. Are you sure its new? the hidden dangers of recycled electronics components. In *IEEE Spectrum*, 2012.
- [85] J. Villasenor and M. Tehranipoor. Chop shop electronics. *IEEE Spectrum*, 50(10):41–45, October 2013.
- [86] J. M. Wang and T. V. Nguyen. Extended Krylov subspace method for reduced order analysis of linear circuit with multiple sources. In *Proc. Design Automation Conf. (DAC)*, pages 247–252, 2000.
- [87] Martin Weigel. Performance potential for simulating spin models on gpu. *Journal of Computational Physics*, 231(8):3064 – 3082, 2012.
- [88] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhundia, and M. Tehranipoor. Hardware trojans: Lessons learned after one decade of research. *ACM Trans. on Design Automation of Electronics Systems*, (1):6:1–6:23, May 2016.
- [89] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno. A 20k-spin ising chip to solve combinatorial optimization problems with cmos annealing. *IEEE Journal of Solid-State Circuits*, 51(1):303–309, Jan 2016.
- [90] Chihiro Yoshimura, Masato Hayashi, Takuya Okuyama, and Masanao Yamaoka. Implementation and evaluation of fpga-based annealing processor for ising model by use of resource sharing. *Internation Journal of Networking and Computing*, 7, 2017.
- [91] L. Zhang. *Effects of Scaling and Grain Structure on Electromigration Reliability of Cu Interconnects*. PhD thesis, University of Texas at Austin, 2010.
- [92] X. Zhang, N. Tuzzio, and M. Tehranipoor. Identification of recovered ICs using fingerprints from a light-weight on-chip sensor. In *Proc. Design Automation Conf. (DAC)*, 2012.
- [93] Xuehui Zhang and Mohammad Tehranipoor. Path delay Fingerprinting for Identification of Recovered ICs. In *IEEE Int. Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems(DFT)*, 2012.