# Dynamic Reliability Management for Multi-Core Processor Based on Deep Reinforcement Learning

Zeyu Sun, Han Zhou, and Sheldon X.-D. Tan
Department of Electrical and Computer Engineering, University of California, Riverside

***Abstract*—In this paper, we propose a new dynamic reliability management (DRM) approach with deep reinforcement learning (DRL) for multi-core processors considering device reliability effects (hard error) and transient error of signal (soft error). The proposed method is based on a recently proposed physics-based three-phase electromigration model and an exponential soft error model that considers dynamic voltage and frequency scaling (DVFS) effects. Our work has been inspired by the recent advancements in DRL for various control and game applications. Compared with the traditional Q-learning based method, DRL has better scalability, lower memory and lower computational complexity. A large class of multi-threaded applications are used as the benchmark to validate and compare the proposed dynamic reliability management methods. Experimental results show that the proposed method can significantly reduces memory footprint and computational time compared to the traditional Q-learning based method. Furthermore, we show that the DRL-based DRM method can save $53.50\%$ more energy than the Q-learning based method and $61.29\%$ more than the simple DVFS based method.**

## I. Introduction

With the development of nanometer VLSI design and technology scaling, more cores have been integrated on the chip. However, chip power density is increasing in technology nodes since transistor and voltage scaling are no longer linear. In this case, only a portion of the cores can be powered on the chip to meet power and temperature limits. Power budget and temperature constraints do not allow all cores to work together. In addition, reliability is also one of the dominant issues need to consider for multi-core systems.

For the reliability of VLSI chips, it mainly consists of aging-related permanent hard errors and transient soft errors. Electromigration (EM) is one of the critical aging-related reliability issues (hard errors) [1]. Because of higher voltage and power consumption, interconnects are more vulnerable to hard errors. On the other hand, soft or transient reliability has quite different impact on VLSI chips compared to the aging-related hard errors. This is especially true for chips operating at very low voltage or even near threshold voltage regions. As a result, we need to consider both reliability effects at the same time during the system-level reliability management to find the best trade-offs. Model based approaches are not suitable for this kind of optimization problems because the remaining lifetime and software profiles are changing transiently. So dynamic management methods are required for hard error and softerror optimization since real-time reactions are required.

Reliability management methods for multi-core scaling have been studied in [2], [3]. The work in [2] considers negative-bias temperature instability (NBTI) and time-dependent dielectric breakdown (TDDB), while [3] mainly focuses on mitigating the thermal cycling induced reliability at the operating system level. All of those methods did not have impacts of soft-errors. Recently, dynamic reliability management (DRM) for multi-core system has been proposed [4], [5]. In this work, hard errors and soft errors based on EM have been considered. Due to the conflict of hard error and soft error requirements, the author employs Q-learning as a management method to obtain an optimized working state. However, a large Q-table is required in the traditional Q-learning method, which takes up a large amount of memory space and has a long time to access Q-table, so it is not suitable for run time operations if dimension of state-action space is high.

This work is supported in part by NSF grant under No. CCF-1527324, and in part by NSF grant under No. CCF-1816361 and in part by NSF grant under OISE-1854276.

Deep reinforcement learning (DRL) using neural network instead of large Q-table has recently been proposed. It is suitable to handle problems in high-dimensional state-action spaces like game playing Atari [6] and Alpha Go [7]. It is also employed to solve embedded system problems such as cloud computing resource allocation, residential smart grid task scheduling, and hybrid electric vehicles [8]. As can be seen from these works, deep reinforcement learning achieves faster run time with better memory utilization and is more suitable to solve online optimization problems with high dimension of state-action space.

In this paper, we propose a new deep reinforcement learning dynamic reliability management framework considering both hard errors and soft errors for multi-core processors. We formulate the DRM problem as online minimizing the energy consumption subject to the reliability including hard and soft errors, power, performance and thermal constraints. The hard error model is based on a recently proposed physics-based EM model. The deep reinforcement learning based control method is applied to solve the aforementioned online optimization problem. Experimental results show that compared with the traditional Q-learning DRM method, the proposed method significantly reduces memory footprint and computational time compared to the traditional Q-learning DRM method. Additionally, we show that the DRL-based method can save $53.50\%$ more energy than the Q-learning based method and $61.29\%$ more than the simple dynamic voltage and frequency scaling (DVFS) based method.

The rest of the paper is organized as follows. Section II reviews hard and soft error models employed in this work. Section III reviews the deep reinforcement learning method and analyzes the advantage of that model over traditional Q-learning method and presents our proposed optimization framework for multi-core system. Section IV provides numerical experimental results on the simulation framework. Section V concludes the paper.

## II. Review of the EM and soft error reliability models

Reliability is one of the most important design considerations in modern microprocessors with aggressive design strategies and decreasing feature size. In more advanced nanoscale technologies, it is a major challenge and limiting factor for VLSI design as temperature, current, and voltage increase. Nowadays, with larger and more complex multi-core systems, both hard and soft errors may occur and damage the system. Hard errors such as EM effect are long term issues and soft errors are short term issues. Even though they have different reliability effects, both of them are very critical issues in current VLSI designs. However, they have opposite failure conditions. High voltages and power cause hard errors while low voltages even near threshold regions cause soft errors. As can be seen from Fig. 1, when the power is high, which means a high operation voltage, the soft error rate (SER) is small but EM lifetime is also short. In the other hand, when power is low, the SER is high but EM lifetime is long.

In order to get reliability information of the multi-core system, system-level EM and soft errors will be discussed in the following subsections.

### A. New physics-based three-phase EM modeling and analysis

EM is a physical phenomenon of the migration of metal atoms along a direction of applied electrical field. Atoms (either lattice atoms or defects/impurities) migrate toward the anode end of metal wire along the trajectory of conducting electrons, leading to the formation of voids and consequently causing resistance to change. However, if the void is smaller than the critical void size of the

978-1-7281-1201-5/19/$31.00 

wire, no change in resistance will be observed. In order to model this phenomenon, a recently proposed three-phase EM model [1] has been employed in this work.

In this model, the EM wear-out process consists of three phases: (a) nucleation phase; (b) incubation phase; (c) growth phase. During the three phases, the stress in the confined copper wire is modeled by the stress diffusion equations with blocked material flux boundaries at the via terminals. In the nucleation phase, stress starts to build-up over time. When it exceeds the critical stress, a void is formed and the wire enters the incubation phase. When the void is larger than the critical size, the wire resistance starts to change and the growth phase begins. The growth phase persists until the void reaches its saturation volume. The new three-phase EM model gives a more accurate time-to-failure estimation and can be applied to more general multi-segment wires since it is based on the stress diffusion physics in the confined copper wires.

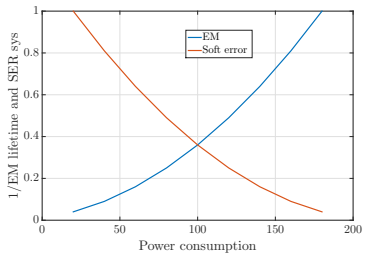


Fig. 1. Comparison between EM-induced lifetime and system-level soft error rate

In the system-level EM analysis, we assume that each core can have its own voltage (voltage island) and frequency settings, and DVFS can be done locally for each core. For the microarchitecture of EM failures we focus on power grid since it is the most vulnerable part to EM failure. A simple mesh-structured power grid is generated for each core, and each of them has its power island with its power regulator. We assume that power grids for each core are the same and are less coupled as far as EM effects are concerned. When the voltage drop caused by resistance increase of the power grid reaches a certain level, the core is considered to be fail.

In order to consider lifetime impacts of different tasks, a system-level model is proposed in [9]:

$$MTTF_{EM} = \frac{1}{(\sum_{m=1}^{n}(\Delta t_m \frac{1}{MTTF_m}))/T} \quad (1)$$

where $MTTF_m$ is the actual MTTF (mean time to failure) under the $m$-th task for $\Delta t_m$ period, assuming the chip works through $n$ different power and frequency settings and $T = \sum_{m=1}^{n} \Delta t_m$.

### B. Soft error reliability model considering DVFS impacts

Soft errors, or single event upset, are defined as transient faults inside the logic or memory on a chip, and result in an incorrect system output. Soft errors may be caused by cosmic radiation, alpha particle decay, and thermal neutrons. SER is the rate at which a chip or system encounters a soft error and typically can be expressed as the number of failures at the given time. While there is still a lack of consensus on the exact SER for a particular chip and system, it is obvious that the SER per chip is practically increasing due to the increasing number of components or cores on the chip. Recently a new exponential soft error model has been introduced to explain those effects [10].

For our problem, we employ an existing exponential model considering DVFS effects on the SER, which assumes that the radiation induced failure follows a Poisson distribution. Average SER can be expressed in terms of operating frequency $f$, voltage $Vdd$ and $SER_0$ which is the average failure rate at the maximum frequency $f_{max}$ and voltage $V_{max}$, (so, $f_{min} < f < f_{max}$ , $V_{min} < V < V_{max}$) in (2).

$$SER(f, V_{dd}) = SER_0 e^{\frac{d(f_{max}-f)}{(f_{max}-f_{min})}} \quad (2)$$

where $d$ is an architecture dependent constant, which is the sensitivity of failure rate with DVFS. We also employ model of relationship between operating frequency and supply voltage to further simplify (2). DVFS-aware SER equation can then be derived as the function of only supply voltage $V_{dd}$ [10]:

$$SER(V_{dd}) = SER_0 e^{\frac{d(f_{max}-\beta V_{dd}-2V_{th}+\frac{V_{th}^2}{V_{dd}})}{f_{max}-f_{min}}} \quad (3)$$

## III. DRL framework for Dark silicon optimization

In this section, a multi-core processor framework based on the DRL method will be introduced. The background of the DRL method will be reviewed and the DRL-based framework applied to this work will be discussed in detail.

### A. Background of deep reinforcement learning

Reinforcement learning is a learning method whose goal is to map situations to actions so as to maximize the rewards or minimize the penalties [11]. It can handle problems with stochastic transition without any adaptation and is a method able to converge close to the optimal solution of a state-action function $(s, a)$ for an arbitrary policy. After each action, a Q-value ($Q(s, a)$) will be calculated with reward/penalty function. The Q-value depends only on the current state and action takes, and stores in a large Q-table to determine the next optimal state. The dimension of the Q-table is $(|A||S|)$ where $A$ is the number of actions and $S$ is the number of states since each $(s, a)$ pair has a corresponding Q-value. The iterative steps to get the final result will be in $O(|A||S|^2)$. When the $(s, a)$ space is large, the Q-learning algorithm consumes a lot of memory and requires many steps to converge to the optimized state.

In order to resolve these problems, the DRL is usually used in order to handle large state action space [8]. A Q-function in Eq. (4) is used to estimate the Q-value of the $(s, a)$ pair.

$$Q(s, a) \approx f(s, a) \quad (4)$$

As can be seen in Eq. (4), only when the result of the Q-function is accurate, Q-learning process can find the optimal choice. Neural network is employed so as to achieve the accurate Q-function and it can lead to an accurate Q-value. The input parameters for training are $(s_t, a_t, r_t, s_{t+1})$ and the output corresponds $Q(s, a)$. Since the training goal is to make the estimated Q-value closer to the real Q-value, so the loss function can be written as:

$$L = E[(Q(s, a) - Q^*(s, a))]. \quad (5)$$

Here $Q(s, a)$ is the real Q-value and $Q^*(s, a)$ is the estimated Q-value.

### B. Deep reinforcement learning based optimization framework

In this subsection, we develop new dynamic reliability management for multi-core processors based on deep reinforcement learning. The purpose is minimizing energy considering a hard and soft error induced lifetime by controlling the p-state of cores. p-state is the performance state subject to power budget, performance deadline, and temperature constraints. The dynamic reliability management method including the DRL is implemented in software which is used to control the hardware performance. Learning platform formulation and and implementation details will be discussed next.

*1) DRL method framework based formulation and solution:* Based on the DRL algorithm proposed in [6], a framework is developed in Fig. 2. In the first step, a learning agent is put in a random initial state. In the proposed framework, the state is the set of cores at different p-states. Since each core has its own unique power, EM and SER, p-state and its associated penalty function. The agent will be transferred from state to state and each action will provide a penalty for the agent. The learning process will stop when the minimized total penalty is achieved. After initializing the learning agent, transition profiles $(s_t, a_t, r_t, s_{t+1})$ and its corresponding Q-value $Q(s_t, a_t)$, these data are restored in an experience memory $D$ and used for neural network training ($NN$). This training is done offline which means the dataset used to in this step is static.

With the offline trained $NN$ model, we can perform the online learning process, which means training is performed as the data comes in. The agent selects the action based on a $\varepsilon$ greedy policy, which is used to control the level of greed [11]. The probability of

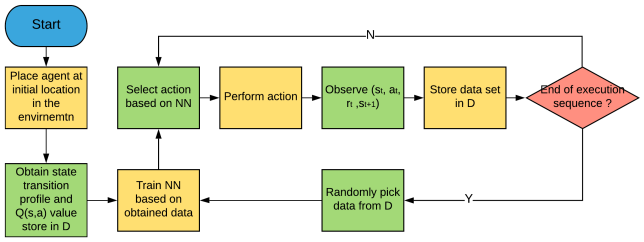



Fig. 2. The proposed deep learning framework for reliability management flow

selecting an action is $1-\varepsilon$, and the action is to maximize the estimated Q-value $a_t = argmax_a Q(s_t, a)$. Otherwise, a random action is selected. This random selection is used to avoid sub-optimal decisions such that the learning agent does not get stuck in the local optimized point and will perform global optimized results. After performing the selected action, the agent will take the action and go to the next state. New transition profiles and their Q-values are observed and stored in memory $D$. If the agent does not reach the end of the execution sequence, a new learning step is performed and a new action will be selected. Then the loop will continue until it reaches the end of an execution sequence and then the $NN$ model will be updated. Experience replay is performed and data are randomly picked from memory $D$ to update $NN$ if the agent reach the end of an execution. By using the experience replay, the behavior distribution is averaged over previous states so that the learning result can be smoother and avoid the oscillations in the output.

*2) Implementation of multi-core system evaluation platform:* In order to evaluate the proposed DRL-based dynamic reliability, an evaluation platform is implemented as shown in Fig. 3. In this work,

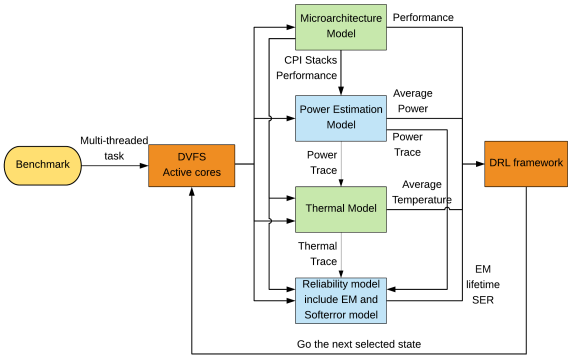



Fig. 3. The evaluation platform for multi-core system based on DRL method

p-state of cores are defined by DVFS. In that state, a microarchitecture simulator is used to obtain the performance and cycles per instruction (CPI), which are passed to power simulator. The power simulator provides dynamic and static power estimation with performance and CPI. A thermal simulator uses all the information from the simulator described above to estimate thermal traces. Power and thermal trace are passed to reliability models discussed in Section II to calculate the EM lifetime and SER.

After obtaining all the results, they are passed to the DRL-based framework, as shown in Fig. 2. Energy is the primary goal of optimization, while other factors such as temperature, performance, EM and SER are served as constraints. With the constrains and optimization goal, we can define the penalty function. The portion of the factor greater than its constraint is multiplied by a large penalty coefficient and all these penalties and energy are used to calculate the penalty function of each action as shown in Eq. (6).

$$PT = PT_{energy} + C \times \sum PT_{constraint} \quad (6)$$

Here $PT$ is the total penalty, $PT_{energy}$ is the penalty from energy. $C$ is a large penalty coefficient and $PT_{constraint}$ is the penalty from constraints. The system will run at this state for an execution period. The model then calculates the updated parameters and selects a new optimal working state.

## IV. Numerical Results and Discussions

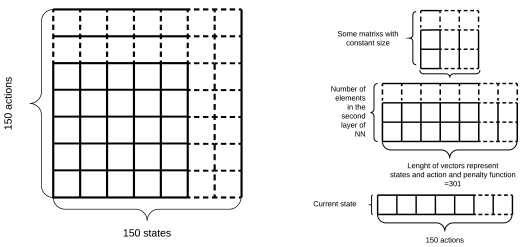
### A. Experiment Setup

The proposed platform is implemented in Python with the numerical library Numpy. Neural network is trained with tensorflow. The microarchitectural simulator is Sniper, the power estimator is McPAT, and the thermal simulator is HotSpot. In this work, SPLASH-2 benchmark is used for evaluation.

Two performance states (p-state) with the clustered DVFS [12] are chosen which have been employed to reduce the simulation time with small solution quality degradation. . Global DVFS method is used as the baseline to compare the energy optimization. In this work, the full power mode is (2.0GHz, 1.2V) and the low power mode is (1.0GHz, 0.9V). In the experiment, 150 states are used and there are 150 actions. The states and actions are decided by realistic processor running conditions.

### B. Memory Usage

In this subsection, the memory efficiency of proposed DRL algorithm will be discussed. In the experiment, $N$ =150 states are used. As shown in Fig. 4(a), Q-learning requires an $N \times N$, which is $150 \times 150$ size Q-table to store Q-value because the number of states is equal to the number of states in the framework. The space complexity is $Q(N^2)$. Memory used by the DRL method includes the memory for $NN$ model and Q-value. Assume there are $A_i$ elements in each layer. Weight between hidden layers can be calculated as $A_i \times A_{i+1}$, where the element in hidden layer $i$ is $A_i$ and the element in the next hidden layer is $A_{i+1}$. Space complexity of this part is $O(1)$. The space taken by the first hidden layer connected with input layer is $(2 \times N + 1) \times A_1$ and in this case is $2 \times 150 + 1$. Space complexity of this part is $O(N)$. In the Q-value part, Q-value estimated by the neural network of $(s_t, a)$ pair needs to be stored so as to find the minimum of them. $N \times 1$ space is used to store these Q-values. The size here is 150 and the space complexity is $O(N)$. Therefore, as shown in Fig. 4(b), the total space complexity of DRL method is $O(N)$.




(a) (b)

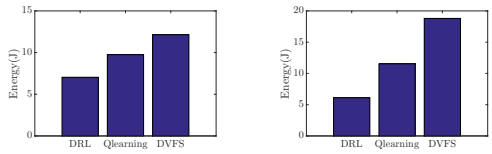
Fig. 4. (a) Memory used of q-learning (b) Memory used of DRL

It can be seen that the DRL algorithm reduces the space complexity from $Q(N^2)$ to $O(N)$.

### C. Energy Consumption

In this subsection, energy consumption of the working state selected by DRL method and Q-learning method will be compared. In transient runtime applications, the time for state switching is limited. So the learning process needs to be completed before the deadline. In the experiment, the time to finish the learning is set to be $0.008s$, which is the limit of the runtime operation. Energy consumption at the selected state with loose and tight constraint of DRL method, Q-learning and DVFS will be compared in Fig. 5. Constraints are selected based on realistic processor running conditions.

Fig. 5(a) shows the results using a loose bound for constraints. Power budget is $800W$, performance constraint is $40.3ms$, temperature constraint is $345K$, EM lifetime constraint is more than 3 years




(a) (b)

Fig. 5. (a) Energy consumption with loose bound (b) Energy consumption with tight bound

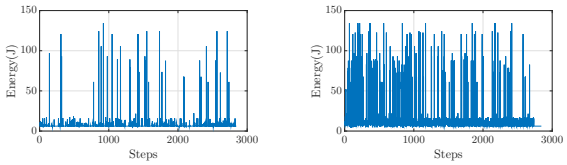
and SER constraint is smaller than 0.6. As we can see, the DRL-based method is $27.79\%$ more energy efficient than the Q-learning based method and $42.01\%$ more energy efficient than the DVFS method. Note that the energy reduction ratio $Energy_{r-ratio}$ between the two methods is calculated using the following equation:

$$Energy_{r-ratio} = \frac{E_{Q-learning} - E_{DRL}}{E_{Qlearning}} \quad (7)$$

Fig. 5(b) uses a tight bound for constraints. Power budget is $400W$, performance constraint is $24.1ms$, temperature constraint is $340K$, EM lifetime constraint is more than 5 years and SER constraint is smaller than 0.4. In this case, the DRL-based method can reduce energy by $53.50\%$ compared to the Q-learning method and by $61.29\%$ compared to the DVFS method. From the results we can see that the DRL method significantly outperforms the Q-learning method and DVFS method in terms of energy reduction. In addition, the energy saving of the DRL method is even more significant compared to the two methods with the tight constraints as tight constraints can help the DRL method converge faster.

### D. Steps to converge

In this subsection, we compare the convergence steps of the DRL method and the Q-learning method. In the comparison, no deadline is set so both methods can converge. The steps for the convergence of the two methods are shown in Fig. 6. We can see that the DRL




(a) (b)

Fig. 6. (a) Time to converge of DRL method (b) Time to converge of Q-learning method

method in Fig. 6(a) reaches a steady state after several hundred iterations. This means that it does not take very long time to reach the selected state. The jump after steady state is caused by random action selection, whose probability is $1 - \varepsilon$, mentioned in Section III-B1. That can help the learning agent find the global optimization action instead of local optimization one. So the spikes on top do not mean the optimization is not finished. Final state with highest possibility to appear is selected. However, the Q-learning based method keeps oscillating until approximately 2700 iterations as shown in Fig. 6(b), which explains why the DRL method has better energy savings. With limited learning time, Q-learning method cannot converge to optimized state while DRL method can reach the optimized state quickly.

## V. Conclusion

In this paper, we propose a new dynamic optimization approach with deep reinforcement learning for multi-core processors considering both hard and soft errors. The reliability model is based on a recently proposed physics-based three-phase electromigration model and an exponential soft error model. Proposed DRL model which has better scalability and has lower memory complexity and computational complexity comparing with traditional reinforcement learning is applied so as to obtain cost-effective solutions for online optimization. This dynamic reliability management method is validated and compared using a large class of multi-threaded applications. Experimental results on a 64-core processor show that the proposed DRL method based framework significantly reduce memory consumed and computational time in comparison to traditional Q-learning based method. DRL-based method can lead to energy reduction of $53.50\%$ better than the Q-learning based method and $61.29\%$ better than the simple DVFS based method. Furthermore, the DRL-based method also shows much smaller memory print and fewer steps to converge than the Q-learning based method.

## References


[1] 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*, vol. 60, pp. 132–152, Jan. 2018.

[2] S. Feng, S. Gupta, A. Ansari, and S. Mahlke, "Maestro: Orchestrating lifetime reliability in chip multiprocessors," in *Proceedings of the 5th International Conference on High Performance Embedded Architectures and Compilers*, ser. HiPEAC'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 186–200.

[3] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli, "Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems," in *Proceedings of the 51st Annual Design Automation Conference*, ser. DAC '14. New York, NY, USA: ACM, 2014, pp. 170:1–170:6. [Online]. Available: http://doi.acm.org/10.1145/2593069.2593199

[4] T. Kim, Z. Sun, H.-B. Chen, H. Wang, and S. X.-D. Tan, "Energy and lifetime optimizations for dark silicon manycore microprocessor considering both hard and soft errors," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 9, pp. 2561–2574, 2017.

[5] S. X.-D. Tan, M. Tahoori, T. Kim, S. Wang, Z. Sun, and S. Kiamehr, *VLSI Systems Long-Term Reliability – Modeling, Simulation and Optimization*. Springer Publishing, 2019.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[7] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[8] P. Zhao, Y. Wang, N. Chang, Q. Zhu, and X. Lin, "A deep reinforcement learning framework for optimizing fuel economy of hybrid electric vehicles," in *Proceedings of the 23rd Asia and South Pacific Design Automation Conference*, ser. ASPDAC '18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 196–202. [Online]. Available: http://dl.acm.org/citation.cfm?id=3201607.3201648

[9] Z. Lu, W. 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)*. IEEE, November 2004, pp. 327–334.

[10] L. Tan, S. Song, P. Wu, Z. Chen, R. Ge, and D. Kerbyson, "Investigating the interplay between energy efficiency and resilience in high performance computing," in *Parallel and Distributed Processing Symposium (IPDPS), 2015 IEEE International*, May 2015, pp. 786–796.

[11] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[12] T. Kolpe, A. Zhai, and S. Sapatnekar, "Enabling improved power management in multicore processors through clustered dvfs," in *Proc. Design, Automation and Test In Europe. (DATE)*, March 2011, pp. 1–6.