CE-Based Optimization for Real-time System Availability under Learned Soft Error Rate

Liying Li[†], Tongquan Wei[†], Junlong Zhou[*], Mingsong Chen[§], Xiaobo Sharon Hu[‡]
[†]Department of Computer Science and Technology, East China Normal University
[*]School of Computer Science and Engineering, Nanjing University of Science and Technology
[§]Shanghai Key Lab of Trustworthy Computing, East China Normal University
[‡]Department of Computer Science and Engineering, University of Notre Dame

***Abstract*—As the density of integrated circuits continues to increase, the possibility that real-time systems suffer from transient and permanent failures rises significantly, resulting in a degraded availability of system functionality. In this paper, we investigate the dynamic modeling of transient failure rate based on Back Propagation (BP) neural network, and propose an optimization strategy for system availability based on Cross Entropy (CE). Specifically, the neural network is trained using cross-layer simulation data obtained from SPICE simulation while the CE-based optimization for system functionality availability is achieved by judiciously selecting an optimal supply voltage for processors under timing constraints. Simulation results show that the proposed method can achieve system availability improvement of up to 32% compared to benchmarking methods.**

I. Introduction

As the feature size of transistors shrinks and the integration of systems increases, the probability of system transient failures drastically rises [1]. Transient failure, also known as soft error, refers to a logical fault that forms a random, temporary state change in the semiconductor due to the collision of high-energy particles and electromagnetic interference [1]. The duration of the soft errors is short and does not permanently damage the hardware. Once soft errors are generated, they can propagate across the circuit layer, the component layer, and the system layer [2]. Several soft errors can be masked during this propagation, while some of the soft errors can possibly reach the system layer, which may lead to incorrect execution results, abnormal termination or even an application crash. Therefore, modeling soft errors across multiple layers and estimate their impact on the availability of system functionality is of particular importance to real-time applications such as avionics and autonomous driving.

In addition to soft errors, increased power consumption and elevated chip temperature have accelerated the aging of microprocessors, leading to rising permanent failures [3]. Permanent failure, also known as hard error, refers to the hardware failure because of the aging of the circuit, which does not disappear once they are formed. Only by repairing or replacing hardware can hard errors be eliminated. Similar to soft errors, hard errors can also cause a system malfunction. The system fails to operate correctly before it is repaired. In fact, the duration of the system functionality availability is jointly determined by both soft and hard errors. In this paper, we consider the impact of soft errors and hard errors on system functionality availability, and explore the optimization strategies for functionality availability of real-time systems.

Modeling and estimation of cross-layer soft error rate (SER) is the crucial step to optimize system availability due to soft errors. At the circuit layer, Hazucha et al. [4] investigated scaling of the atmospheric neutron soft error rate which affects the reliability of CMOS circuits at different altitudes, and proposed a method to predict a linear decrease of SER per bit with decreasing feature size. At the component layer, Heijmen et al. [3] investigated the factors that affect the critical charge for a soft error in a memory cell and studied the situation of soft errors in logic gates. At the system layer, Kiamehr et al. [5] presented a chip-level electrical masking analysis which accurately considers the impact of voltage fluctuation across the chip. In recent years, many research efforts no longer simply consider the soft error rate of an independent layer, but consider multiple layers at the same time. Riera et al. [2] considered the soft error propagation process from the circuit layer to the component layer, and obtained the soft error rate of the component under different CMOS manufacturing processes through the HSPICE simulation circuit.

System availability is essentially subject to both hard errors and soft errors. Zhou et al. [6] formulated and solved the problem of maximizing system availability considering both types of errors. They first modeled the system functionality availability due to hard and soft errors, then presented a task replication and frequency selection strategy to optimize system functionality availability.

Although the estimation of soft error rate and optimization for functionality availability has been extensively studied in the literature, most of these works derive soft error rate using the SPICE simulator. However, the time-consuming simulation-based approach is not suitable for safety-critical real time applications where environmental parameters constantly change and timeliness is of particular importance.

In this paper, we propose a BP neural network-based dynamic soft error rate estimation scheme, followed by a system availability optimization algorithm based on cross entropy (CE). The BP neural network is trained by using data obtained from the SPICE simulations, and the system availability is optimized by selecting an optimal supply voltage for the system under timing constraints. Simulation results show that the proposed method can achieve system availability improvement of up to 32% compared to benchmarking methods.

This work was supported in part by the Shanghai Municipal NSF under Grant 16ZR1409000, in part by the China HGJ Project under Grant 2017ZX01038102-002, in part by the National NFSC under Grant 61802185 and Grant 61872147, in part by Jiangsu NSF under Grant BK20180470, and in part by the Fundamental Research Funds for the Central Universities.

T. Wei is the corresponding author, Email: tqwei@cs.ecnu.edu.cn.

The rest of the paper is organized as follows. Section II introduces models used in this paper and formalizes the problem studied. Section III describes the BP-based SER estimation method and CE-based system availability optimization method proposed in this paper. Section IV presents the experimental results, and Section V concludes the paper.

II. System Models and Problem Definition

The focus of this paper is on availability enhancement of a real-time system in the presence of soft and hard errors. Below, we introduce the processor and task models, cross-layer soft error model and system availability model. Then we formalize the system availability optimization problem in this paper.

A. Processor and Task Model

In this paper, we adopt a dynamic voltage/frequency scaling (DVFS) enabled uniprocessor system that supports various frequencies ranging from the minimum frequency f_{min} to the maximum frequency f_{max}. We denote the L frequencies as $\{f_1, f_2, ..., f_L\}$, and assume that $f_{min} = f_1 \leq f_2 \leq ... \leq f_L = f_{max}$ holds. The L frequencies $\{f_1, f_2, ..., f_L\}$ correspond to L voltages $\{v_1, v_2, ..., v_L\}$. For the sake of easy presentation, the frequencies are normalized with respect to f_{max}, that is, f_{max} = 1.0.

We consider a task set Γ, which consists of N independent tasks and is denoted by $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_N\}$. Real-time task τ_i $(1 \leq i \leq N)$ can be represented using a tuple $\tau_i : \{P_i, D_i, c_i\}$, where P_i is the period and D_i is the deadline of task τ_i. c_i denotes the execution time of task τ_i when the processor executes τ_i at the maximum frequency f_{max} (f_L). Given c_i, we can derive the execution time of task τ_i at a given frequency f_j $(1 \leq j \leq L)$, denoted by E_i, as $E_i = c_i/f_j$.

B. Cross-Layer Soft Error Model

At the Earth's surface approximately 95% of the particles capable of causing soft errors are energetic neutrons [7]. Neutrons hit transistors of an electronic device and produce a certain amount of electrical charge. When the charge exceeds the threshold, the transistor is activated and unexpected results such as losing the stored data or a glitch in the output of a logic gate are produced. Hazucha et al. [4] proposed a prediction method for the soft error rate at the circuit layer ($\mathrm{SER_{cir}}$), as given by

$$\mathrm{SER_{cir}} = a' \times F \times S' \times e^{(-\frac{Q'_c}{Q'_s})}, \tag{1}$$

where a' is a constant parameter depending on the circuit technology and design, F denotes the flux of neutrons dependent on geographic locations, and S' represents the area of the circuit that is sensitive to soft errors. The critical charge of circuits, denoted by Q'_c, is the threshold that can cause a circuit malfunction, it can be estimated in circuit models by injecting different current pulses into sensitive nodes [3]. Q'_s denotes the charge collection efficiency (i.e., the ratio of collected charge to generated charge), which can be derived through accelerated neutron tests or device physics models [2].

As soon as soft errors are generated at the circuit layer, they propagate across the circuit layer, the component layer, and the system layer. Since the sensitive circuit area at different layers shrinks, not every soft error at the circuit layer can propagate and eventually reach the component and the system layer. Riera et al. [2] provided a soft error characterization of all the basic components. The soft error rate at the component layer, denoted by $\mathrm{SER_{com}}$, can be approximately calculated by only considering SRAM cells, latches and logic gates, and is given by

$$\mathrm{SER_{com}} = a \times F \times S \times e^{(-\frac{Q_c}{Q_s})}, \tag{2}$$

where a is a constant dependent on the component type, F is the geographic location based flux of neutrons, and S is the sensitive circuit area of the component. Q_c and Q_s are the threshold value to cause a circuit malfunction and the charge collection efficiency at the component layer, respectively. Q_s in fact equals the average of Q'_c at the circuit layer.

The propagation of sort errors arrived at the component layer may be affected by a variety of derating factors such as timing factors and logical factors [8]. Therefore, some of the soft errors finally propagate to the system layer and may cause software errors or system failures. Mukherjee et al. [8] formalized the notion of derating factors as the architectural vulnerability factor (AVF) and proposed a method to calculate the value of various components' AVF. The soft error rate at the system layer, referred to as $\mathrm{SER_{sys}}$, is related to the soft errors at different components. We establish the soft error rate model at the system layer as

$$\mathrm{SER_{sys}} = \sum\nolimits_{i=1}^{M} r_i \times \mathrm{AVF}_i \times \mathrm{SER_{com_i}}, \tag{3}$$

where M is the number of component types. We consider 3 types of components in processors including SRAMs, latches, and logic gates, thus M = 3. r_i represents the ratio of the number of components of type i to the total number of components. $\mathrm{AVF_i}$ denotes the architectural vulnerability factor of components of type i. $\mathrm{SER_{com_i}}$ is the soft error rate of components of type i, which can be obtained by Eq. (2).

C. System Availability Model

Considering the impact of soft errors only, the reliability of task τ_i, denoted by R_i, is given by

$$R_i = e^{-\mathrm{SER_{sys}} E_i}, \tag{4}$$

where E_i denotes the execution time of task τ_i. Then we derive the probability of soft error occurrences during the execution of tasks in Γ, which is denoted by P_{fail} and given as

$$P_{fail}(\Gamma) = 1 - \prod\nolimits_{i=1}^{N} R_i, \tag{5}$$

where N is the number of tasks in Γ and R_i denotes the reliability.

Zhou et al. [6] derived an effective analytical expression to calculate the mean time to failure for soft errors, which is denoted by $\mathrm{MTTF_T}$, as given by

$$\mathrm{MTTF_T} = \frac{T_{exe}(\Gamma) + T_{exp}(\Gamma)}{P_{fail}(\Gamma)} - T_{exe}(\Gamma), \tag{6}$$

where $T_{exp}(\Gamma)$ denotes the expected time to failure, which can be obtained by using $T_{exp}(\Gamma) = \sum_{i=1}^{N} E_i \cdot P_{fail}(\tau_i)$. E_i represents the execution time of task τ_i, and $T_{exe}(\Gamma) = \sum_{i=1}^{N} E_i$ denotes the total execution time of the N tasks in Γ.

Continuous increasing in power density leads to elevated operating temperature and frequent temperature variations, which accelerates chip wear-out due to electro-migration, time dependent dielectric breakdown (TDDB), stress migration, and thermal cycling [9]. System failures caused by the above 4 factors are defined as hard errors. We mainly consider hard errors caused by the TDDB. TDDB refers to the deterioration of the gate oxide layer. Gate current due to hot electrons causes defects in the oxide, which eventually form a low-impedance path and cause the transistor to permanently fail. This effect is influenced by temperature and is increasing with the reduction of gate oxide dielectric thickness and non-ideal supply voltage. The mean time to failure due to hard errors ($\mathrm{MTTF_P}$) is defined by [9]

$$\mathrm{MTTF_P} = A_{TDDB}\left(\frac{1}{v}\right)^{(\vartheta_1-\vartheta_2 T)} e^{\frac{A+\frac{B}{T}+CT}{\rho T}}, \tag{7}$$

where A_{TDDB} is a fitting constant, v is the supply voltage, and T denotes temperature. ϑ_1, ϑ_2, ρ, A, B, and C are empirical fitting parameters.

We aim at maximizing system availability, which can be defined as $\frac{\mathrm{MTTF}}{\mathrm{MTTF+MTTR}}$ [6]. MTTR is the mean time to repair. For a system that may suffer from both soft and hard errors, system availability depends on the smaller of $\frac{\mathrm{MTTF_T}}{\mathrm{MTTF_T+MTTR_T}}$ and $\frac{\mathrm{MTTF_P}}{\mathrm{MTTF_P+MTTR_P}}$, where $\mathrm{MTTR_T}$ and $\mathrm{MTTR_P}$ denote the mean time to repair due to soft errors and hard errors, respectively [6]. Therefore, the system availability $\mathrm{MTTF_{sys}}$ is defined as

$$\mathrm{MTTF_{sys}} = \min\{\tfrac{\mathrm{MTTF_T}}{\mathrm{MTTF_T+MTTR_T}}, \tfrac{\mathrm{MTTF_P}}{\mathrm{MTTF_P+MTTR_P}}\}, \tag{8}$$

which can be rewritten as

$$\mathrm{MTTF_{sys}} = \min\{\frac{\mathrm{MTTF_T}}{\mathrm{MTTR_T}}, \frac{\mathrm{MTTF_P}}{\mathrm{MTTR_P}}\}. \tag{9}$$

Define φ as $\varphi = \frac{\mathrm{MTTR_P}}{\mathrm{MTTR_T}}$. Since $\mathrm{MTTR_P}$ and $\mathrm{MTTR_T}$ are given, φ is constant. Thus, the system availability $\mathrm{MTTF_{sys}}$ can be converted into

$$\mathrm{MTTF_{sys}} = \min\{\varphi\mathrm{MTTF_T}, \mathrm{MTTF_P}\}. \tag{10}$$

D. Problem Formulation

The $(\vartheta_1 - \vartheta_2 T)$ in Eq. (7) is positive when temperature is between $0°C$ and $100°C$ [9]. Therefore, $\mathrm{MTTF_P}$ is negatively related to the supply voltage when the temperature and neutron flux are fixed. However, the system soft error rate increases when the supply voltage decreases [10]. Thus, the mean time to failure due to soft errors (i.e., $\mathrm{MTTF_T}$) is positively related to the supply voltage. As can be see from Eq. (10), an elegant tradeoff between $\varphi\mathrm{MTTF_T}$ and $\mathrm{MTTF_P}$ is needed to maximize the system availability, which can be achieved by adjusting the supply voltage when φ is given. We define $\mathrm{MTTF_{Delt}}$ as

$$\mathrm{MTTF_{Delt}} = |\varphi\mathrm{MTTF_T} - \mathrm{MTTF_P}|. \tag{11}$$

The system availability is maximized when $\mathrm{MTTF_{Delt}}$ is minimized, that is, $\mathrm{MTTF_{Delt}} \in (0, \varepsilon)$ holds for an arbitrarily small number $\varepsilon > 0$.

Assume Earliest Deadline First (EDF) algorithm is adopted for task scheduling. EDF is a dynamic scheduling algorithm that dictates the arrangement of tasks in a priority queue [11]. In the EDF algorithm, the necessary and sufficient condition for tasks to be completed before their deadline is that the CPU utilization of the system, denoted by U, is less than 1. The optimization of the system availability is thus formulated as

$$\text{Minimize: } \mathrm{MTTF_{Delt}} \tag{12}$$

$$\text{subject to: } U < 1 \tag{13}$$

where U is given by $U = \sum_{i=1}^{N} E_i/D_i \leq 1$, in which E_i and D_i denote the execution time and the deadline of task τ_i, respectively. Note that we use EDF for task scheduling in this paper. However, the proposed system availability optimization approach is applicable to any scheduling algorithms.

III. BP-based System SER Modeling and CE-based Availability Optimization

We aim at maximizing the system availability, which is equivalent to minimizing the $\mathrm{MTTF_{Delt}}$ given in Eq. (11). To this end, we first need to calculate $\varphi\mathrm{MTTF_T}$ and $\mathrm{MTTF_P}$. $\mathrm{MTTF_P}$ can be derived by using Eq. (7) while $\mathrm{MTTF_T}$ given in Eq. (6) depends on the system soft error rate. A common method for calculating system soft error rate is to use SPICE, a simulation tool widely used in the integrated circuit and board-level design to check the integrity of circuit designs and to predict circuit behavior [2]. However, using SPICE to estimate the system soft error rate is time-consuming and usually takes several hours to get results. Thus, SPICE-based simulation method is not suitable for real-time applications such as avionics where environmental parameters constantly change. In this section, we train a BP neural network by using data from the SPICE simulation and adopt the BP-based neural network approach to quickly and dynamically estimate the system soft error rate, then optimize the system availability based on the derived soft error rate.

A. BP-based Dynamic SER Modeling

BP neural network refers to the artificial neural network that uses the BP algorithm to calculate and update network weights and biases. It generally has one input layer, one or more hidden layers, and one output layer. There are one or more nodes in each layer and multiple connections between layers. The input-layer nodes receive the input signals that are usually dynamically changing and pass them to the hidden layer. The hidden-layer nodes then calculate the results from the input-layer nodes and propagate them to the output layer. The output-layer nodes process and output the final results.

In the estimation of $\mathrm{SER_{sys}}$, 4 variables including supply voltage, temperature, neutron flux and the system critical charge fluctuate dynamically. Neutron flux changes with latitude, longitude, and altitude of the system's location. The system critical charge, denoted by $Q_{c_{sys}}$, is a threshold value for charges beyond which a system malfunction occurs. It changes with the supply voltage and temperature and can be derived by using

$$Q_{c_{sys}} = \sum\nolimits_{i=1}^{M} r_i \times Q_{c_i}, \tag{14}$$

where Q_{c_i} is the critical charge of components of type i, r_i represents the ratio of the number of components of type i to the total number of components, and M is the total number of component types.

The BP-based neural network takes the 4 variables as input and outputs the system soft error rate. The number of the hidden-layer nodes of the network, denoted by n_h, given as

$$n_h = \sqrt{n_{in} + n_{out}} + a, \tag{15}$$

where a is a constant number between 1 and 10, and n_{in} and n_{out} are the number of nodes in the input layer and the output layer, respectively. The output of the hidden-layer node j, denoted by H_j, is calculated by using

$$H_j = f(\sum\nolimits_{i=1}^{n_{in}} \omega_{ij} x_i + m_{ij}), \tag{16}$$

where x_i is the input signal, ω_{ij} and m_{ij} denote the weights and biases between the input-layer node i and hidden-layer node j, respectively.

Afterwards, the signals propagate from the hidden layer to the output layer, and the result of the output-layer node k, denoted by O_k, is given by

$$O_k = f(\sum\nolimits_{j=1}^{n_h} \omega_{jk} H_j - m_{jk}), \tag{17}$$

where ω_{jk} and m_{jk} denote the weights and biases between the hidden-layer node j and output-layer node k, respectively. The results from the output layer are processed by an activation function, which is given by

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{18}$$

The propagation error of the BP neural network (Err) can be formulated as

$$Err = \frac{1}{2}\sum\nolimits_{k=1}^{n_{out}} (B_k - O_k)^2, \tag{19}$$

where B_k and O_k represent the expected value and the result of output-layer node k. If the propagation error is higher than expected, back propagation is needed, that is, the propagation error is transmitted from the output layer to the hidden layer and then to the input layer, and the weights in each layer are modified. The above process is repeated until the network propagation error is smaller than a predefined threshold value.

B. CE-based System Availability Optimization

We focus on optimizing the system functionality availability by judiciously selecting an optimal supply voltage. It is well known that our voltage selection is an NP-hard problem, which can be effectively solved by cross entropy (CE) method. Owing to space restrictions, we only introduce the application of cross entropy in the proposed method, and skip the theoretical foundation of the CE method. Readers are referred to [12] for further reading. CE method first generates several optional voltage solution samples according to a specified mechanism such as Gaussian distribution, and then interactively changes the mechanism to obtain better solution samples.

We design a CE-based voltage selection method to optimize the system functionality availability. The pseudo code of our heuristic is shown in Algorithm 1, it takes the task set (Γ), neutron flux (F), supply voltages (v) and temperature (T) as input, and outputs the optimal supply voltage (v_{opt}). The algorithm first obtains the training data through SPICE simulation and trains the BP neural network by using the training data (lines 1-2). It estimates the system soft error rate in line 3, and initializes parameters (mean and variance) of Gaussian distribution and the number of iterations in lines 4-5. The algorithm iteratively derives the optimal voltage from line 6 to 11. Specifically, the algorithm generates $\mathcal{J}$ voltage samples according to Gaussian distribution $N(\theta_\eta, \xi_\eta)$ and selects $\mathcal{Q}$ samples that satisfy the CPU utilization constraint given in Eq. (13) by using acceptance-rejection method (lines 7-8). Afterwards, the selected samples are evaluated in terms of $\mathrm{MTTF_{Delt}}$ and the top $\mathcal{K}$ elite samples are chosen to update the parameters of the Gaussian distribution (lines 9-12). According to the basic principle of cross entropy, the mean (θ_η) is updated by using $\theta_\eta = \sum_{i=1}^{\mathcal{K}} v_i/\mathcal{K}$, and the variation (ξ_η) is calculated by using $\xi_\eta = \sqrt{\sum_{i=1}^{\mathcal{K}} (v_i - \theta_\eta)/\mathcal{K}}$. The iteration stops if the predefined convergence criteria are met or the number of iterations (η) reaches the predefined value. The optimal voltage (v_{opt}) is generated by using the iteratively obtained distribution $N(\theta_\eta, \xi_\eta)$.

Algorithm 1 System availability optimization algorithm

```
Input: The task set (Γ), the neutron flux (F), L supply voltages
    {$v_1, v_2, ..., v_L$} and temperature (T).
Output: The optimal supply voltage ($v_{opt}$).
 1: Obtain training data by using SPICE simulation;
 2: Train BP neural network by using the training data;
 3: Estimate system soft error rate by using trained BP neural network;
 4: Initialize the mean ($\theta_1$) and variation ($\xi_1$) of Gaussian distribution;
 5: $\eta$ = 1; /*Initialize the number of iterations ($\eta$) */
 6: repeat
 7:     Generate $\mathcal{J}$ solution samples $[v_1, v_2, \ldots, v_{\mathcal{J}}]$ using Latin hypercube
        sampling method according to distribution $N(\theta_\eta, \xi_\eta)$;
 8:     Select $\mathcal{Q}$ ($\mathcal{Q} < \mathcal{J}$) feasible solution samples satisfying constraint in
        Eq. (13) using acceptance-rejection method;
 9:     Evaluate the $\mathcal{Q}$ selected solution samples based on $\mathrm{MTTF_{Delt}}$
        calculated by Eq. (11);
10:     Choose top $\mathcal{K}$ elite samples with respect to $\mathrm{MTTF_{Delt}}$
11:     $\eta$++;
12:     Update $\theta_\eta$ and $\xi_\eta$;
13: until predefined convergence criteria are met or the number of iterations
    ($\eta$) reaches the predefined value;
14: Sort tasks in Γ with EDF algorithm;
15: return the optimal supply voltage ($v_{opt}$);
```

IV. Evaluation

In order to validate the effectiveness of our method, we conduct a series of simulation-based experiments. We first verify the accuracy of our dynamic soft error rate estimation model based on the BP neural network. Then we verify the effectiveness of the system availability optimization algorithm.

A. Simulation Settings

The simulations are conducted on a machine equipped with 2.5GHz Intel i7 quad-core processor and 8GB DDR4 memory, and running a Windows version of Matlab x64 and Eclipse. We generate 1000 real-time tasks. The period P_i, deadline D_i, and execution time c_i of task τ_i are randomly generated in the interval of (250, 1500)ms, (50, 100)ms, and (50, 100)ms, respectively.

We use the SPICE simulator to generate 1000 sets of sample data. Each sample contains 5 parameters, which are the threshold value that causes a circuit malfunction in the component layer (Q_c), the neutron flux (F), the supply voltage (v), the chip temperature (T), and the system soft error rate ($\mathrm{SER_{sys}}$). Q_c for components SRAM, latch, and logic gate are randomly generated in the interval of [1.94, 120.10]fC, [1.90, 39.05]fC and [3.33, 214.4]fC, respectively. The values of F evenly distribute in $(0, 550]\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$. The processor voltage v varies in the interval $[0.7, 1.2]$V with a step size of 0.1V, the corresponding frequency changes in the interval $[1.0, 2.25]$GHz, and the temperature is randomly generated in the interval of $[0, 100]$°C. The AVF for components SRAM, latch, and logic gate is 0.1, 0.15 and 0.08 according to [13]. The ratio of the number of components of the above 3 types to the total number of components are set to 0.775, 0.025, and 0.2, respectively [14]. As for the calculation of $\mathrm{MTTF_P}$, we set the empirical fitting parameters A, B, C, ρ, ϑ_1, and ϑ_2 to 1.63, 0.07, 0.01, 0.3, 1.5, and 0.003, respectively [15]. The fitting constant A_{TDDB} is set to 2.15 [15].

According to Eq. (15), the number of hidden-layer nodes in the BP neural network is between 3 and 12. The relationship between the number of the hidden-layer nodes and the number of iterations and mean square error is shown in Fig. 1. It can be seen that when the number of hidden-layer nodes is 12, the number of iterations and mean square error of the BP neural network are minimal. Therefore, we set the number of hidden-layer nodes to 12.

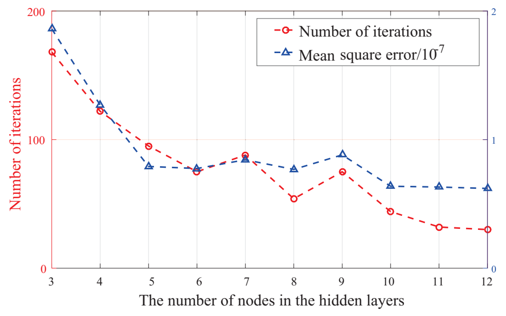


Figure 1: The optimal number of hidden-layer nodes.

B. Results on SER Estimation

We obtain 1000 sets of example data through SPICE simulation, use 800 of them to train a BP neural network, and use the other 200 sets to test the accuracy of the trained BP neural network. In the 800 sets of training data, the maximum difference between the estimated value of the trained BP neural network and the ground truth data is 3% and the minimum difference is 0%. In the 200 sets of testing data, the maximum difference is 3.4% and the minimum difference is 0%. Therefore, the trained BP neural network can accurately estimate the system soft error rate. Fig. 2 shows 20 examples of the comparison between the estimated $\mathrm{SER_{sys}}$ from the trained BP neural network and the ground truth data.

C. Results on Availability Optimization

In order to verify the effectiveness of the proposed availability optimization method, we compare the proposed optimization method with 3 other benchmarking methods. The first method is referred to as gradual voltage scaling (GVS). In this method, the system initially executes tasks in a task set at the lowest supply voltage, then gradually increases the supply voltage with a step size of 0.01V each time the tasks in the task set are completed until the system meets its timing constraints. The second method is referred to as random voltage scaling (RVS) where the system executes tasks at a

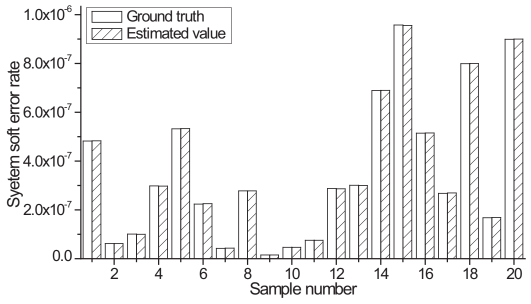


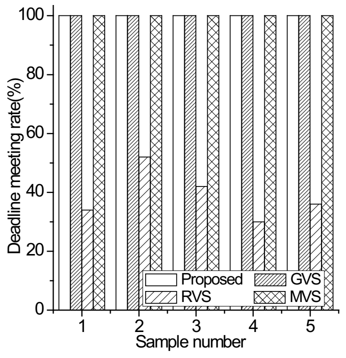
Figure 2: Comparison of estimated value and ground truth.

randomly selected supply voltage. The third method is called maximum voltage scaling (MVS) where the system executes tasks at the maximum supply voltage.

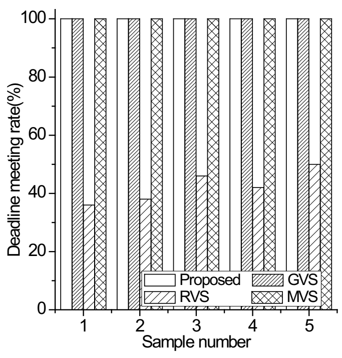
We compare the proposed scheme with the above 3 methods in terms of the deadline meeting rate and the system availability when the temperature is set to 25°C, 50°C, and 75°C, respectively. The deadline meeting rate is defined as the ratio of tasks completed on time to all tasks, and the system availability is measured by $\mathrm{MTTF_{sys}}$, the system mean time to failure. Each experiment is repeated 500 times and the ultimate results are averaged over the repeated experiment results.

Fig. 3 shows that the proposed method, GVS, and MVS can ensure all tasks' timing constraints. The MVS allows the system to execute all tasks at the maximum supply voltage, thus, the execution speed of tasks is relatively fast and the performance of MVS in the deadline meeting rate is the best among the 4 methods. However, MVS accelerates the aging of circuits and reduces the availability of the system when high performance is provided. The GVS allows the system to initially run the tasks at a minimum supply voltage, and gradually increase the voltage by a given step until all tasks are completed in time. The proposed method judiciously selects appropriate supply voltages for tasks such that the tasks are completed in time. The performance of GVS in deadline meeting rate is close to the performance of the proposed method when the step size of GVS is small. However, there is a tradeoff between step size and convergence time of GVS. If the step size is too small, GVS spends a long time to converge. Conversely, if the step size is too large, the performance of GVS is close to MVS, thus also impairing system availability. The RVS allows the system to run the tasks at a randomly selected supply voltage. Thus, some tasks cannot complete in time when the selected supply voltage is too low. The deadline meeting rate of RVS varies between 30% and 52%.

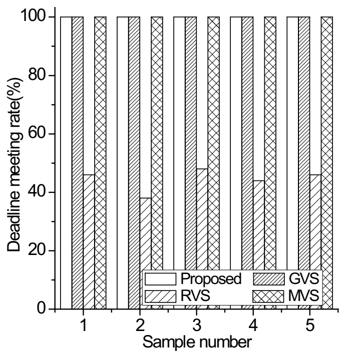
Fig. 4 shows that when T=25°C, the $\mathrm{MTTF_{sys}}$ by using the proposed method, GVS, RVS, and MVS are 2.43 years, 1.95 years, 1.98 years, and 2.12 years, respectively. Compared with the GVS, RVS, and MVS, the proposed method can enhance the system availability by up to 25%, 23%, and 15%, respectively. And when T=50°C, using the proposed method can enhance the system availability by up to 25%, 14%, and 14% compared with using the GVS, RVS, and MVS, respectively. Compared with the GVS, RVS, and MVS, the proposed method has 32%, 17%, and 18% improvement in $\mathrm{MTTF_{sys}}$ when T=75°C.



(a) T = 25°C

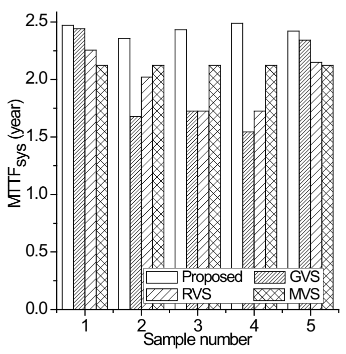


(b) T = 50°C

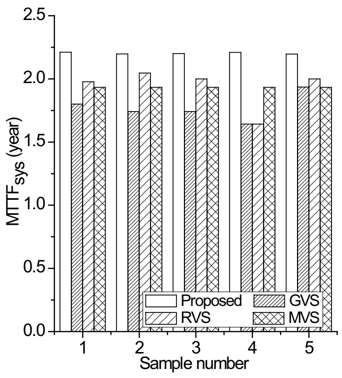


(c) T = 75°C

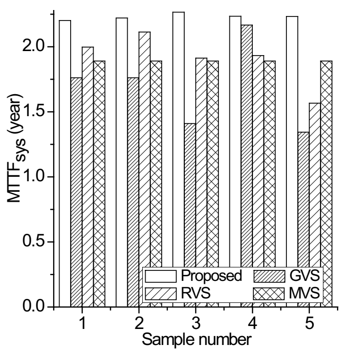
Figure 3: Compare the proposed method and 3 benchmarking method in terms of deadline meeting rate.



(a) T = 25°C



(b) T = 50°C



(c) T = 75°C

Figure 4: Compare the proposed method and 3 benchmarking method in terms of $\mathrm{MTTF}_{\mathrm{sys}}$.

V. Conclusion

In this paper, we aim at enhancing the availability of real-time systems that suffer from soft and hard errors. We propose a dynamic SER estimation method based on BP neural network. The BP neural network is trained using data obtained from the SPICE simulation. The simulation results show that the proposed estimation method can quickly and accurately estimate the system soft error rate even when environmental parameters such as temperate and geographic location constantly change. We further model the system availability considering both soft and hard error, and propose a CE-based system availability optimization algorithm. The simulation results show that under timing constraints, the proposed CE-based system availability optimization algorithm can achieve a significant improvement of system availability by up to 32% compared to benchmarking methods.

References

[1] J. Martinez, J. Maestro, and P. Reviriego, Evaluating the Impact of the Instruction Set on Microprocessor Reliability to Soft Errors, *IEEE Transactions on Device and Materials Reliability*, pp. 1-1, 2018.

[2] R. Marc, C. Ramon, A. Jaume, and G. Antonio, A detailed methodology to compute Soft Error Rates in advanced technologies, *Design, Automation & Test in Europe (DATE)*, pp. 217-222, 2016.

[3] T. Heijmen, D. Giot, and P. Roche, Factors that impact the critical charge of memory elements, *IEEE International On-Line Testing Symposium*, pp. 57-62, 2006.

[4] P. Hazucha, and C. Svensson, Impact of CMOS technology scaling on the atmospheric neutron soft error rate, *IEEE Transactions on Nuclear Science*, pp. 2586-2594, 2000.

[5] S. Kiamehr, M. Ebrahimi, and F. Firouzi, Chip-level modeling and analysis of electrical masking of soft errors, *IEEE 31st VLSI Test Symposium (VTS)*, pp. 1-6, 2013.

[6] J. Zhou, X. Sharon Hu, Y. Ma, and T. Wei, Balancing lifetime and soft-error reliability to improve system availability, *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 685-690, 2016.

[7] J. Ziegler, "Terrestrial cosmic rays", *IBM Journal of Research and Development. IBM.* vol. 40, no. 1, pp. 19-40, 1996.

[8] S. Mukherjee, C. Weaver, J. Emer, S. Reinhardt, and T. Austin, A systematic methodology to compute the architectural vulnerability factors for a high-performance microprocessor, *IEEE/ACM International Symposium on Microarchitecture*, pp. 29-40, 2003.

[9] JEDEC Solid State Technology Association and others, Failure mechanisms and models for semiconductor devices, *JEDEC Publication JEP122-B*, 2003.

[10] N. Mahatme, N, Gaspard, and S. Jagannathan, Impact of Supply Voltage and Frequency on the Soft Error Rate of Logic Circuits, *IEEE Transactions on Nuclear Science*, vol. 60, no. 6, pp. 4200-4206, 2010.

[11] "Earliest deadline first scheduling" [Online]. Available: https://en.wikipedia.org/wiki/Earliestdeadlinefirstscheduling.

[12] X. Zhao, Y. Guo, Z. Feng, and S. Hu, Parallel hierarchical cross entropy optimization for on-chip decap budgeting, *Design Automation Conference*, pp. 843-848, 2010.

[13] E. Mojtaba, L. Chen, H. Asadi, and M. Tahoori, CLASS: Combined logic and architectural soft error sensitivity analysis, *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 601-607, 2013.

[14] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, Exploiting structural duplication for lifetime reliability enhancement, *International Symposium on Computer Architecture*, pp. 520-531, 2005.

[15] X. Yun, C. Thidapat, D. Robert, X. Sharon Hu, and, S. Li, System-level reliability modeling for MPSoCs, *IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pp. 297-306, 2010.