Smart Adjustment of Transistor Parameters to Reduce Temperature Rise Due to Self-Heating Effect

Vazgen Melikyan, Petros Petrosyan, Narek Avagyan and Gor Abgaryan

October 10, 2023

Smart Adjustment Of Transistor Parameters To Reduce Temperature Rise Due To Self-Heating Effect

Vazgen Melikyan
Chair of Microelectronic Circuits and Systems National Polytechnic University of Armenia
Synopsys Armenia Educational Department
Yerevan, Armenia
vazgenm@synopsys.com

Petros Petrosyan
Institute of Physics
Yerevan State University
Synopsys Armenia Educational Department
Yerevan, Armenia
petros.petrosyan9722@gmail.com

Narek Avagyan
Institute of Physics
Yerevan State University
Synopsys Armenia Educational Department
Yerevan, Armenia
narekavagyan98@gmail.com

Gor Abgaryan
Institute of Physics
Yerevan State University
Synopsys Armenia Educational Department
Yerevan, Armenia
abgaryan.gor98@gmail.com

Abstract*—The present study introduces a novel approach for the computation of temperature elevation arising from the self-heating phenomenon, coupled with an effective parameter tuning technique aimed at mitigating temperature variations. In contemporary integrated circuits, the self-heating effect emerges as a highly significant and intricate phenomenon, often challenging to quantify accurately. Addressing this challenge demands a comprehensive strategy involving supplementary measurements and meticulous optimizations. This research employs a power measurement methodology to assess the thermal perturbations within individual components of the circuit. Furthermore, the study advances a robust technique for calibrating the parameters of transistors to govern the thermal upswing. The experimental investigations are conducted employing a Multi-Gate MOSFET library within a 14nm technological framework. Comprehensive simulations are executed utilizing the HSPICE simulator, while the scripting aspects are implemented through Python 3.7.

Keywords*—reliability, self-heating, analysis, optimization, parameter tuning

I. INTRODUCTION

The matter of integrated circuit (IC) reliability has become more significant with the progress of technology. Among the crucial factors affecting reliability, self-heating stands out as a key consideration that can modify circuit performance and even impact logic integrity. Higher temperatures within transistors can wield influence over vital characteristics like threshold voltage and electron mobility [1-3].

The growing complexity and miniaturization of modern integrated circuits have led to an increased concern regarding self-heating effects and their impact on circuit reliability [4]. As power consumption continues to rise in integrated circuits, the resulting heat dissipation can introduce temperature fluctuations that adversely affect transistor characteristics [5].

Although temperature variations within an element may be influenced by its environment, the dominant factor remains the self-heating effect. This investigation focuses on the self-heating phenomenon at the logical level, where it hinges upon both power consumption and transistor attributes. Notably, changes in temperature exhibit a linear relationship with power usage. Considering the temporal aspect of temperature alteration, power consumption is appropriately evaluated using the root mean square of power as shown in (1) (power RMS) [6-8].

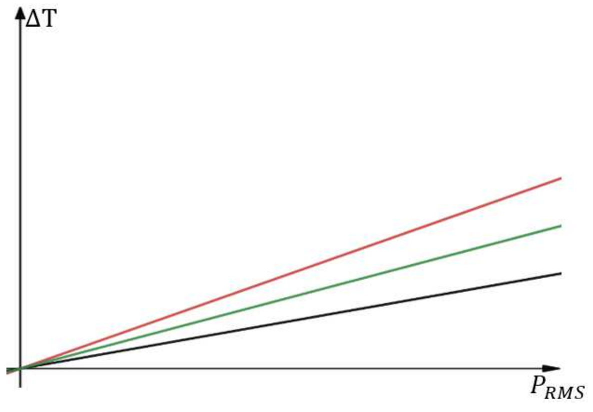


Fig. 1. Power dependency on temperature

$$\Delta T = \mathrm{R_{th}} * P_{rms} \quad (1)$$

$\mathrm{R_{th}}$- is the temperature coefficient or thermal resistance.

In Figure 1, the graph illustrates the dependency of power RMS on temperature. The angle of the line on the x-axis represents the thermal resistance.

Utilizing this interdependence, the calculation of temperature rise for specific elements becomes achievable. Power consumption computation is realized through the use of HSpice simulations. The process of power consumption calculation involves augmenting the HSpice netlist with power consumption measurements for each element [9]. Following this, the modified netlist undergoes HSpice simulations, enabling the acquisition of extensive power consumption information. The final step entails integrating the measured values back into the netlist. Equipped with power consumption data and transistor parameters, the extent of temperature elevation can be precisely ascertained.

Addressing temperature fluctuations necessitates the optimization of power consumption or the modification of thermal resistance. Achieving power consumption optimization without altering logic presents challenges and may not be feasible in some cases. The proposed methodology centers on reducing thermal resistance through parameter tuning, prioritizing it over explicit power consumption optimization. It's crucial to acknowledge that this approach can incidentally influence power consumption, potentially in various directions. As a result, each alteration necessitates invoking temperature calculations to ensure precise temperature assessments. This iterative process upholds the accuracy of temperature evaluations amid evolving parameter modifications.

In the model employed in this study, thermal resistance is determined by (2).[8]

$$R_{th} = \frac{RTH_0}{NF*(WTH_0+FPITCH*NFIN)} \qquad (2)$$

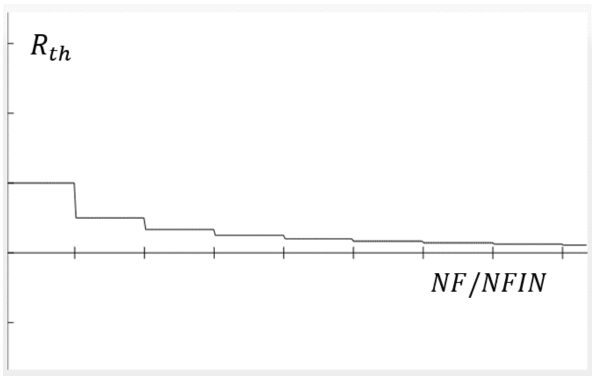


Fig. 2. Thermal resistance dependency on number of fingers

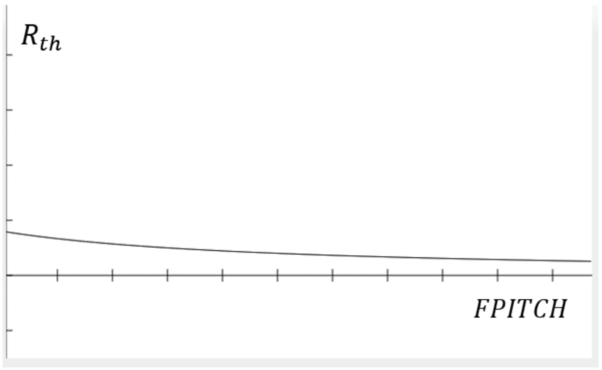


Fig. 3. Thermal resistance dependency on fin pitch (fpitch)

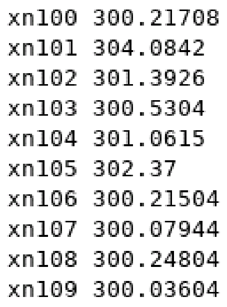
Thermal resistance exhibits an inversely proportional correlation with several key factors, including the quantity of fins, the number of fingers, and the fin pitch. Model-based constants, denoted as RTH_0 and WTH_0, contribute to this interplay. Manipulating these three parameters offers a means to curtail temperature elevation within this particular model. For alternative transistor models, minor adjustments are warranted to facilitate parameter changes and achieve analogous temperature reduction effects.

II. THE PROPOSED PARAMETER TUNING MODEL

In this section, outlined the innovative model for parameter tuning, which underpins the strategy to manage temperature fluctuations induced by the self-heating effect. The approach is anchored in the scrupulous calibration of transistor parameters, yielding a deliberate reduction in temperature variations. By orchestrating parameter adjustments within predefined limits, we pave the way for effective temperature optimization while adhering to the established temperature fluctuation threshold. This model serves as a robust framework for enhancing the reliability and thermal performance of integrated circuits, ensuring that temperature variations remain well within the acceptable range.

Calculating temperature variations mandates the preliminary execution of the power measurement script [10]. This script operates with the sole input of the HSpice netlist. By harnessing the acquired power consumption measurements, the subsequent determination of temperature change becomes attainable. The second script, functioning sequentially, accepts the HSpice netlist and the measured root mean square (RMS) powers. Within this framework, the script proceeds to extract the transistor parameters, specifically nf, nfin, and fpitch. Leveraging these parameters, it computes the thermal resistance. The ensuing calculation of temperature change hinges upon the interplay between thermal resistance and power consumption, effectively encapsulating the thermal behavior of each element. This dual-step process effectively equips us to forecast potential temperature fluctuations, relying solely on the HSpice netlist and the self-heating equation inherent in the transistor model.

The calculated temperature differentials offer a foundation for implementing strategic optimizations aimed at curbing temperature escalation. For instance, envision an initial temperature set at 300K. The resulting output file, illustrated in Figure 4, serves as a tangible depiction of these calculated temperature variations and underscores their significance within the context of the optimization process.



```
xn100 300.21708
xn101 304.0842
xn102 301.3926
xn103 300.5304
xn104 301.0615
xn105 302.37
xn106 300.21504
xn107 300.07944
xn108 300.24804
xn109 300.03604
```

Fig. 4. Example of measured temperatures for different transistors

This article establishes a crucial criterion, deeming temperature fluctuations of up to 5 degrees as acceptable. This threshold serves as a benchmark, necessitating remedial action when temperature deviations exceed this limit. To achieve this objective, modifications to transistor parameters emerge as a viable solution. Adhering to the prescribed guidelines, each parameter holds the potential for adjustment within a range of up to 100% from its original value. It's worth noting that these predefined limits can be flexibly adapted as circumstances warrant. This strategic approach empowers the optimization process, facilitating the attenuation of temperature variations beyond the stipulated 5-degree tolerance.

The most straightforward approach to optimization involves exhaustively assessing all potential parameter values. For each transistor, we calculate temperature fluctuations across a spectrum of acceptable parameter variations. For instance, if a 20% deviation from a parameter's baseline is permissible, then temperature computations are conducted across this entire range. Yet, this exhaustive approach can consume a considerable amount of time due to the involved HSpice simulations.

To mitigate this computational burden, a judicious alternative is smart parameter tuning. This strategic refinement entails a discerning selection of parameter adjustments based on their positive impact. Specifically, we focus on changes that yield favorable outcomes, while disregarding those that do not contribute positively in subsequent iterations. This targeted approach optimizes the utilization of HSpice simulations, streamlining the overall process.

Furthermore, additional avenues for optimization encompass methods like random search, among others. The chosen methodology aligns as an optimal choice due to the directed nature of parameter dependencies. Through these strategies, an equilibrium is sought between comprehensive exploration and computational efficiency, thereby enhancing the efficacy of the parameter tuning endeavors.

Considering the relatively minor influence of parameter adjustments on the temperatures of other elements, the process can be optimized by reorienting the approach to temperature calculations. Rather than invoking temperature calculations for each individual element, we streamline the process by performing HSpice simulations on a per-iteration basis. In each iteration, we methodically evaluate one parameter for all elements. If the temperature rise diminishes for a specific element, we promptly update the parameter value accordingly.

This revised strategy capitalizes on the interdependence of parameter effects, enabling us to substantially reduce the number of HSpice simulation calls while achieving effective temperature optimization. By orchestrating parameter changes more strategically and collaboratively, we strike a balance between computational efficiency and accurate temperature management across the circuit elements.

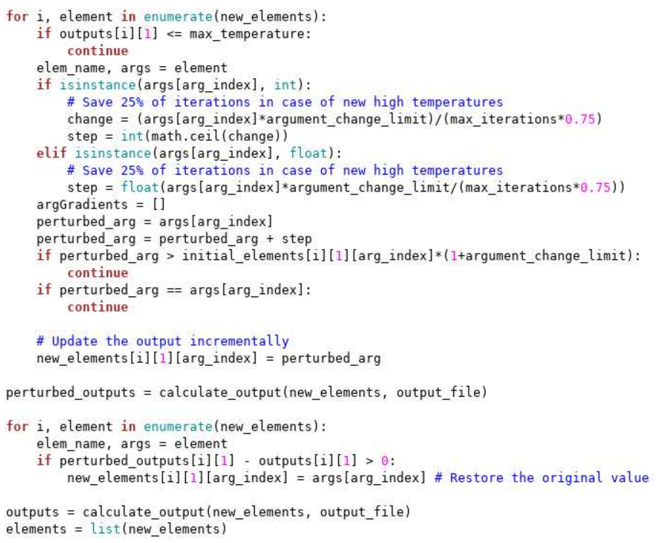
In essence, the approach optimizes the utilization of HSpice simulations by adopting a parameter-centric perspective. Rather than invoking simulations for every individual circuit element, we strategically execute simulations for each parameter adjustment. This refined process enables us to minimize computational overhead while efficiently managing temperature variations.

Furthermore, the optimization strategy extends to parameter adjustments themselves. We prioritize changes that exhibit positive effects, focusing solely on meaningful adjustments. By doing so, we ensure that parameter alterations are conducted purposefully and only when deemed necessary. This holistic approach harmonizes the intricacies of parameter tuning, simulation efficiency, and temperature optimization, culminating in a well-rounded methodology for enhancing integrated circuit reliability and thermal performance.

To attain the optimal parameter value, it is essential to repeat these processes iteratively. In each iteration, the parameter is adjusted by a step size determined by the number of previous iterations. This iterative refinement ensures a thorough exploration of parameter space, allowing us to converge towards the most favorable parameter configuration. Through this iterative fine-tuning, we enhance the precision and effectiveness of the optimization strategy, ultimately achieving a robust and well-optimized solution for managing temperature fluctuations in integrated circuits.

Certainly, the adjustment of parameters has a twofold impact, influencing both thermal resistance and power consumption. As we embark on parameter tuning, it's imperative to account for transistors whose power consumption surpasses their initial values. To address this, an additional phase of iterations is introduced. These supplementary iterations target the affected transistors, enabling us to finely calibrate their parameters to align with both thermal and power requirements. This strategic extension enhances the adaptability of the approach, ensuring a comprehensive optimization process that takes into consideration the dual effects of parameter adjustments.

Script for each argument are presented on Figure 5.



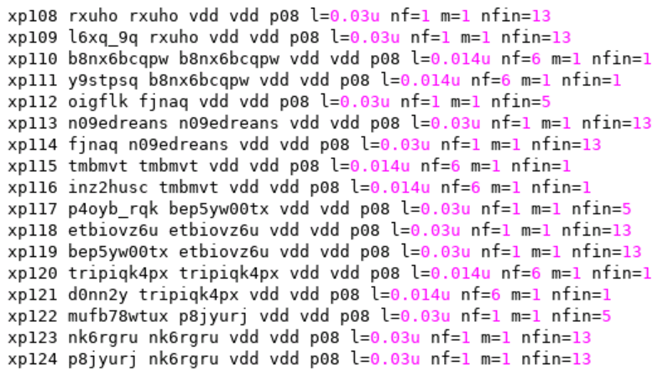
```
for i, element in enumerate(new_elements):
    if outputs[i][1] <= max_temperature:
        continue
    elem_name, args = element
    if isinstance(args[arg_index], int):
        # Save 25% of iterations in case of new high temperatures
        change = (args[arg_index]*argument_change_limit)/(max_iterations*0.75)
        step = int(math.ceil(change))
    elif isinstance(args[arg_index], float):
        # Save 25% of iterations in case of new high temperatures
        step = float(args[arg_index]*argument_change_limit/(max_iterations*0.75))
    argGradients = []
    perturbed_arg = args[arg_index]
    perturbed_arg = perturbed_arg + step
    if perturbed_arg > initial_elements[i][1][arg_index]*(1+argument_change_limit):
        continue
    if perturbed_arg == args[arg_index]:
        continue

    # Update the output incrementally
    new_elements[i][1][arg_index] = perturbed_arg

perturbed_outputs = calculate_output(new_elements, output_file)

for i, element in enumerate(new_elements):
    elem_name, args = element
    if perturbed_outputs[i][1] - outputs[i][1] > 0:
        new_elements[i][1][arg_index] = args[arg_index] # Restore the original value

outputs = calculate_output(new_elements, output_file)
elements = list(new_elements)
```

Fig. 5. Script for tuning each argument of transistors

III. Demonstrative Case Study: Temperature Management in Action

This segment takes readers through a real-world case study that highlights the practical application of the proposed parameter tuning methodology to mitigate temperature concerns. By observing the method in action, one can gain a deeper understanding of its efficacy in addressing temperature fluctuations within integrated circuits. The demonstrated approach showcases its potential to enhance the operational dependability and thermal efficiency of contemporary electronic systems. Through this illustrative use case, a tangible depiction emerges, accentuating the concrete advantages that the method introduces to real-life scenarios.

To assess the method's functionality and efficiency, a straightforward flash ADC is employed as an illustrative example. Notably, thermal resistance exhibits an inverse relationship with three key parameters (nf, nfin, fpitch), making transistors with lower parameters exhibit the highest thermal resistance. The excerpt provided below in Figure 6 showcases a portion of the netlist.

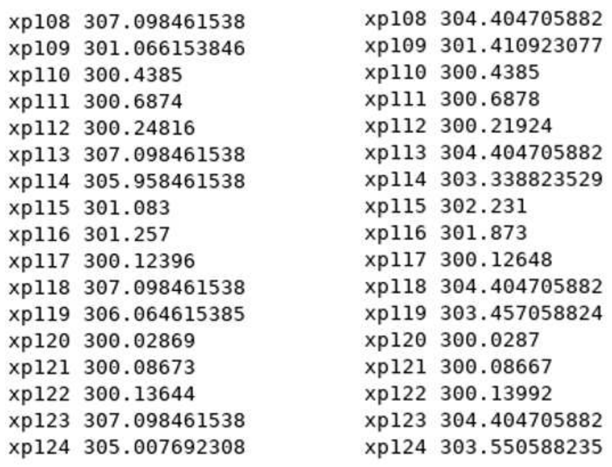


```
xp108 rxuho rxuho vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp109 l6xq_9q rxuho vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp110 b8nx6bcqpw b8nx6bcqpw vdd vdd p08 l=0.014u nf=6 m=1 nfin=1
xp111 y9stpsq b8nx6bcqpw vdd vdd p08 l=0.014u nf=6 m=1 nfin=1
xp112 oigflk fjnaq vdd vdd p08 l=0.03u nf=1 m=1 nfin=5
xp113 n09edreans n09edreans vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp114 fjnaq n09edreans vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp115 tmbmvt tmbmvt vdd vdd p08 l=0.014u nf=6 m=1 nfin=1
xp116 inz2husc tmbmvt vdd vdd p08 l=0.014u nf=6 m=1 nfin=1
xp117 p4oyb_rqk bep5yw00tx vdd vdd p08 l=0.03u nf=1 m=1 nfin=5
xp118 etbiovz6u etbiovz6u vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp119 bep5yw00tx etbiovz6u vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp120 tripiqk4px tripiqk4px vdd vdd p08 l=0.014u nf=6 m=1 nfin=1
xp121 d0nn2y tripiqk4px vdd vdd p08 l=0.014u nf=6 m=1 nfin=1
xp122 mufb78wtux p8jyurj vdd vdd p08 l=0.03u nf=1 m=1 nfin=5
xp123 nk6rgru nk6rgru vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
xp124 p8jyurj nk6rgru vdd vdd p08 l=0.03u nf=1 m=1 nfin=13
```

Fig. 6. Extract from the netlist showing transistor parameters

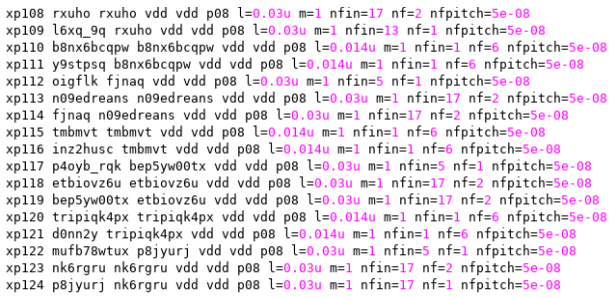
Observably, certain temperatures experience a rise exceeding 5 degrees. However, leveraging the method outlined earlier facilitates a reduction in all temperatures. The objective of this temperature reduction approach is to curtail temperature fluctuations and align them within an acceptable range. Figure 7 portrays the temperatures post-application of the method. These alterations transpire due to modifications made in the HSpice netlist file. A segment of the modified netlist is illustrated in the Figure 8.

In light of parameter adjustments impacting the power consumption of all transistors, it follows that modifying parameters for one transistor can inadvertently yield adverse effects on others. This inherent interdependency underscores the rationale behind adopting a collective approach rather than addressing each element individually. Moreover, to preempt temperature surges during the final iteration, a certain proportion of iterations need to be reserved. In this instance, the initial 75% of iterations are allocated for adjusting parameters up to their maximum values, while the remaining 25% are dedicated to rectifying unforeseen temperature spikes.



```
xp108 307.098461538        xp108 304.404705882
xp109 301.066153846        xp109 301.410923077
xp110 300.4385             xp110 300.4385
xp111 300.6874             xp111 300.6878
xp112 300.24816            xp112 300.21924
xp113 307.098461538        xp113 304.404705882
xp114 305.958461538        xp114 303.338823529
xp115 301.083              xp115 302.231
xp116 301.257              xp116 301.873
xp117 300.12396            xp117 300.12648
xp118 307.098461538        xp118 304.404705882
xp119 306.064615385        xp119 303.457058824
xp120 300.02869            xp120 300.0287
xp121 300.08673            xp121 300.08667
xp122 300.13644            xp122 300.13992
xp123 307.098461538        xp123 304.404705882
xp124 305.007692308        xp124 303.550588235
```

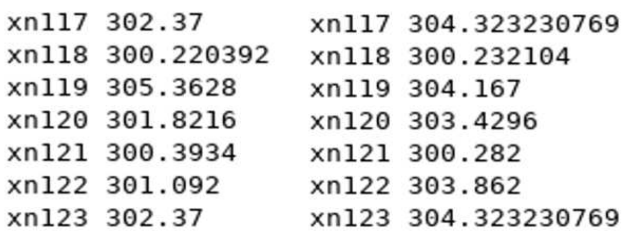
Fig. 7. Temperatures before and after applying the temperature reduction method



```
xp108 rxuho rxuho vdd vdd p08 l=0.03u m=1 nfin=17 nf=2 nfpitch=5e-08
xp109 l6xq_9q rxuho vdd vdd p08 l=0.03u m=1 nfin=13 nf=1 nfpitch=5e-08
xp110 b8nx6bcqpw b8nx6bcqpw vdd vdd p08 l=0.014u m=1 nfin=1 nf=6 nfpitch=5e-08
xp111 y9stpsq b8nx6bcqpw vdd vdd p08 l=0.014u m=1 nfin=1 nf=6 nfpitch=5e-08
xp112 oigflk fjnaq vdd vdd p08 l=0.03u m=1 nfin=5 nf=1 nfpitch=5e-08
xp113 n09edreans n09edreans vdd vdd p08 l=0.03u m=1 nfin=17 nf=2 nfpitch=5e-08
xp114 fjnaq n09edreans vdd vdd p08 l=0.03u m=1 nfin=17 nf=2 nfpitch=5e-08
xp115 tmbmvt tmbmvt vdd vdd p08 l=0.014u m=1 nfin=1 nf=6 nfpitch=5e-08
xp116 inz2husc tmbmvt vdd vdd p08 l=0.014u m=1 nfin=1 nf=6 nfpitch=5e-08
xp117 p4oyb_rqk bep5yw00tx vdd vdd p08 l=0.03u m=1 nfin=5 nf=1 nfpitch=5e-08
xp118 etbiovz6u etbiovz6u vdd vdd p08 l=0.03u m=1 nfin=17 nf=2 nfpitch=5e-08
xp119 bep5yw00tx etbiovz6u vdd vdd p08 l=0.03u m=1 nfin=17 nf=2 nfpitch=5e-08
xp120 tripiqk4px tripiqk4px vdd vdd p08 l=0.014u m=1 nfin=1 nf=6 nfpitch=5e-08
xp121 d0nn2y tripiqk4px vdd vdd p08 l=0.014u m=1 nfin=1 nf=6 nfpitch=5e-08
xp122 mufb78wtux p8jyurj vdd vdd p08 l=0.03u m=1 nfin=5 nf=1 nfpitch=5e-08
xp123 nk6rgru nk6rgru vdd vdd p08 l=0.03u m=1 nfin=17 nf=2 nfpitch=5e-08
xp124 p8jyurj nk6rgru vdd vdd p08 l=0.03u m=1 nfin=17 nf=1 nfpitch=5e-08
```

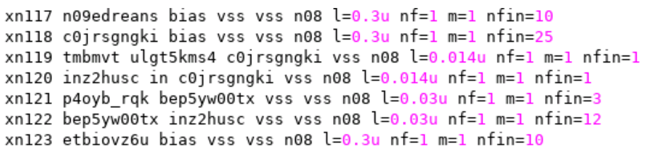
Fig. 8. Modified netlist segment reflecting temperature-reducing changes

An illustrative scenario highlighting the potential adverse impact of modifications on other elements is depicted in Figure 9. The temperature of the xn117 transistor experiences an increase subsequent to the alterations. Furthermore, Figure 10 and Figure 11 provide a visual representation of the HSpice netlist files both before and after the temperature-reducing modifications. As evident from the netlist file, specific parameters of the transistor have been adjusted to ensure temperature remains within an acceptable range.



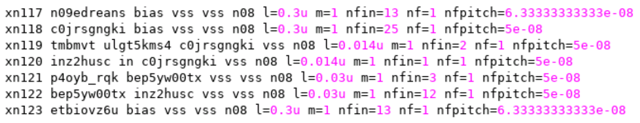
```
xn117 302.37          xn117 304.323230769
xn118 300.220392      xn118 300.232104
xn119 305.3628        xn119 304.167
xn120 301.8216        xn120 303.4296
xn121 300.3934        xn121 300.282
xn122 301.092         xn122 303.862
xn123 302.37          xn123 304.323230769
```

Fig. 9. Illustration of adverse impact of modifications on other elements



```
xn117 n09edreans bias vss vss n08 l=0.3u nf=1 m=1 nfin=10
xn118 c0jrsgngki bias vss vss n08 l=0.3u nf=1 m=1 nfin=25
xn119 tmbmvt ulgt5kms4 c0jrsgngki vss n08 l=0.014u nf=1 m=1 nfin=1
xn120 inz2husc in c0jrsgngki vss n08 l=0.014u nf=1 m=1 nfin=1
xn121 p4oyb_rqk bep5yw00tx vss vss n08 l=0.03u nf=1 m=1 nfin=3
xn122 bep5yw00tx inz2husc vss vss n08 l=0.03u nf=1 m=1 nfin=12
xn123 etbiovz6u bias vss vss n08 l=0.3u nf=1 m=1 nfin=10
```

Fig. 10. HSpice netlist before temperature-reducing modifications



```
xn117 n09edreans bias vss vss n08 l=0.3u m=1 nfin=13 nf=1 nfpitch=6.33333333333e-08
xn118 c0jrsgngki bias vss vss n08 l=0.3u m=1 nfin=25 nf=1 nfpitch=5e-08
xn119 tmbmvt ulgt5kms4 c0jrsgngki vss n08 l=0.014u m=1 nfin=2 nf=1 nfpitch=5e-08
xn120 inz2husc in c0jrsgngki vss n08 l=0.014u m=1 nfin=1 nf=1 nfpitch=5e-08
xn121 p4oyb_rqk bep5yw00tx vss vss n08 l=0.03u m=1 nfin=3 nf=1 nfpitch=5e-08
xn122 bep5yw00tx inz2husc vss vss n08 l=0.03u m=1 nfin=12 nf=1 nfpitch=5e-08
xn123 etbiovz6u bias vss vss n08 l=0.3u m=1 nfin=13 nf=1 nfpitch=6.33333333333e-08
```

Fig. 11. HSpice netlist after temperature-reducing modifications

IV. Measurement of Runtime for the Proposed Method

Within this section, a comprehensive analysis of the runtime for the proposed method is undertaken. By scrutinizing the time needed for the execution of the approach, valuable insights are garnered into its computational efficiency and pragmatic viability. Through a meticulous examination of runtime measurements, a holistic comprehension of the method's real-world performance is presented.

The temperature reduction method necessitates a considerable amount of time due to its reliance on multiple HSpice simulations. Specifically, the total count of simulations performed is equivalent to the product of the number of arguments and the number of iterations.

In addition to the HSpice simulations, the method involves calls to temperature calculation scripts, further influencing overall performance. To provide a detailed perspective, we conducted runtime measurements for various netlists, shedding light on the method's performance characteristics.

Table 1 illustrates the runtime of the temperature reduction program and the HSpice simulation across different designs. A total of ten iterations were executed during the experimentation process. The results highlight that the duration of HSpice simulation aligns with the presence of elements within the design, while the temperature reduction algorithm's runtime corresponds to that of the HSpice simulation.

The additional scripts responsible for calculating temperature changes and adjusting transistor parameters exhibit rapid execution, rendering them inconsequential in runtime assessments.

Given the potential prolonged duration of HSpice simulations, achieving temperature reduction can pose challenges. In scenarios where time constraints are at play, the algorithm facilitates control over the iteration count. This control stems from the fact that HSpice simulation, which verifies temperature alterations, is invoked following each iteration. For instance, in instances of limited time where only a single HSpice simulation is feasible, the approach involves configuring the maximum parameter change and setting the iteration count to 1. This strategy compels the algorithm to take substantial steps and assess temperature fluctuations. Subsequent to the evaluation, the algorithm makes a determination whether to retain the newly computed value or not based on the outcomes.

Table 1. Runtime Comparison of Temperature Reduction Program and HSpice Simulation for Various Designs

	Number of transistors	HSpice simulation time, s	optimization time, s
1.	6	0.267	3.26
2.	154	1.154	20.4
3.	771	4.45	74.4
4.	1720	9.5	162

Figure 12 depicts the runtime dependency on the number of transistors.

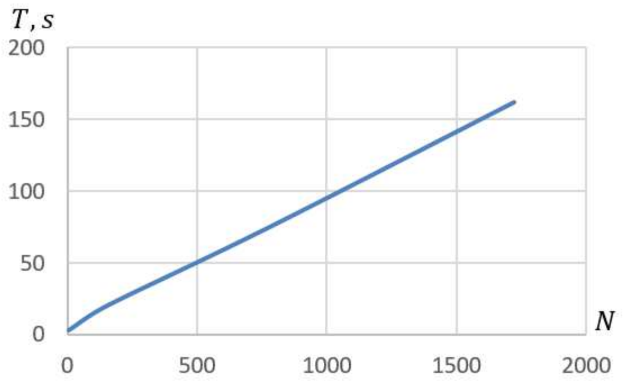


Fig. 12. Variation of runtime with number of transistors

V. Acknowledgment

The work was supported by the Science Committee of RA, in the frames of the research project № 21T-2B002.

VI. Conclusion

In conclusion, this study introduces a method for mitigating self-heating effects in modern integrated circuits. By optimizing transistor parameters and leveraging HSpice simulations, we achieve a targeted reduction in temperature fluctuations. Our approach streamlines the parameter-tuning process, resulting in efficient temperature control. The method's flexibility across transistor models and its compatibility with various technological contexts showcase its versatility. The validation through runtime measurements underscores its practical feasibility, making it a valuable tool for enhancing circuit reliability and performance.

References

[1] Goel, N.: 'Temperature effects on Threshold Voltage and Mobility for Partially Depleted SOI MOSFET', Int. Journal of Computer Applications in Technology, 2012, 42, (21), pp. 56-58

[2] Dhar, S., Kosina, H., Palankovski, V., et al.:'Electron mobility model for strained-Si devices', IEEE Transactions on Electron Devices, 2005, 52, (4), pp. 527-533

[3] Oxley, C., Uren, M., Coates, A., et al.:'On the temperature and carrier density dependence of electron saturation velocity in an AlGaN/GaN HEMT', IEEE Transactions on Electron Devices, 2006, 53, (3), pp. 565-567

[4] Smith, R., & Franzon, P. D. (2000). Power distribution in CMOS integrated circuits: implications for IR drop and self-heating. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 8(6), 632-642.

[5] Braga, M. D. M., & Cardoso, F. A. (2015). Self-heating effects on analog and RF circuits. IEEE Transactions on Circuits and Systems I: Regular Papers, 63(5), 647-657.

[6] Zimin, S., Litian, L., Zhijian, Li.: 'Self-heating effect in SOI MOSFETs', International Conference on Solid-State and Integrated Circuit Technology, Beijing, China, October, 1998, pp. 572-574

[7] Jang, D., Bury, E., Ritzenthaler R., et al.:'Self-heating on bulk Fined from 14nm down to 7nm node', International Electron Devices Meeting, Washington, USA, December, 2015, pp. 11.6.1-11.6.4

[8] Sriramkumar, V., Paydavosi, D., Duarte, J., et al.:'BSIM-CMG 107.0.0: Multi-gate MOSFET compact model: technical manual', (University of California, Berkeley, 2013)

[9] Synopsys, Inc.:'HSPICE user guide: simulation and analysis.(Version B-2008.09).' (Mountain View, CA: Synopsys, Inc, 2008)

[10] V. Melikyan, P. Petrosyan, N. Avagyan and G. Abgaryan, "Self-heating analysis method of integrated circuits," International Conference on Microwave & THz Technologies, Wireless Communications and OptoElectronics (IRPhE 2022), Hybrid Conference, Yerevan, Armenia, 2022, pp. 63-65, doi: 10.1049/icp.2022.2799.