2024 Girl Hackathon Ideathon Round: Solution Submission

Project Name: Simulated Annealing

Pariticipant Name: Anuvarshini G

Participant Email ID: anuvarshinigeceskct@gmail.com

Participant GOC ID: **280885235186**

ReadMe File Links (Eg Github)
https://github.com/Anu-243/Google-woman-hackathon_Anu/blob/main/soc%20design%20using%20simulated%20annealing.pdf

Brief summary

One potential area where simulated annealing can be implemented on an SoC is in the optimization of resource allocation and scheduling in embedded systems. In embedded systems, resources such as processing units, memory, and communication channels need to be efficiently allocated and scheduled to meet performance requirements and constraints.

Simulated annealing can be utilized in SoCs for optimization tasks in various domains such as signal processing, image processing, network optimization, and system-level design. By integrating the algorithm into the hardware of an SoC, it can provide real-time optimization capabilities and improve system performance.

Problem Statement

- Simulated annealing on SOC using pre-defined requirements.
- The simulated annealing algorithm is used to find the optimal solution that minimizes the objective function while satisfying the constraints.
- By implementing simulated annealing on an SoC in embedded systems, it can help in optimizing resource allocation, task scheduling, and power management. The algorithm can be used to find near-optimal solutions for complex optimization problems in real-time embedded systems. For example, in a system with multiple tasks competing for resources, simulated annealing can be employed to allocate resources dynamically based on changing system requirements and constraints.
- The code is written for individuals or organizations who are interested in optimizing a multi-objective function with constraints. This could include researchers, engineers, or data analysts who are working on complex problems where finding a global optimum is impossible or computationally expensive. The code provides a powerful optimization algorithm that can find approximate global optima in complex problems, and it can be customized to fit specific problem requirements.

The approach used to generate the algorithm/design.

- The simulated annealing algorithm is a global search optimization algorithm inspired by the annealing technique in metallurgy.
- It is used to solve complex optimization problems where finding a global optimum is difficult or computationally expensive.
- The algorithm works by generating a random start point in the search space and evaluating it using the objective function.
- It then generates a new point by perturbing the current point and evaluates it using the objective function.

Proof of Correctness.

- The proof of correctness of the simulated annealing algorithm is based on the fact that if the objective function is a symmetric function, the probability of accepting a new point is still high even if it is worse than the current point, as long as the difference in objective function values is not too large.
- This ensures that the algorithm can escape local optima and explore the solution space effectively.
- The rate of convergence of the algorithm can be improved by choosing the temperature reduction factor and the initial temperature appropriately.
- The code provided is an implementation of the simulated annealing algorithm for optimizing a multi-objective function with constraints, using a weighted sum of latency, bandwidth, power, buffer occupancy, and arbitration rate as the objective function and defining constraints based on the maximum allowed values for latency, bandwidth, buffer occupancy, and arbitration rate.

Complexity Analysis

- The time complexity of simulated annealing can drop to O(bm/2), where b is the branching factor and m is the depth of the search space. This indicates that the time complexity can be significantly reduced, allowing for deeper solvable depths.
- Overall, the time complexity of simulated annealing is typically polynomial, making it an efficient algorithm for optimization problems. The space complexity is generally low compared to other algorithms, as it does not require extensive memory usage.

Alternatives considered

- Alternate used is designing soc with simulated appealing approach
- Python and Pythia programming is used for designing soc

References and appendices

1. https://arxiv.org/pdf/2109.12021.pdf: Pythia: A Customizable Hardware Prefetching Framework Using Online Reinforcement Learning
2.https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf - Reinforcement Learning: An Introduction
3. https://ieeexplore.ieee.org/abstract/document/1511971: The Network-on-Chip Paradigm in Practice and Research

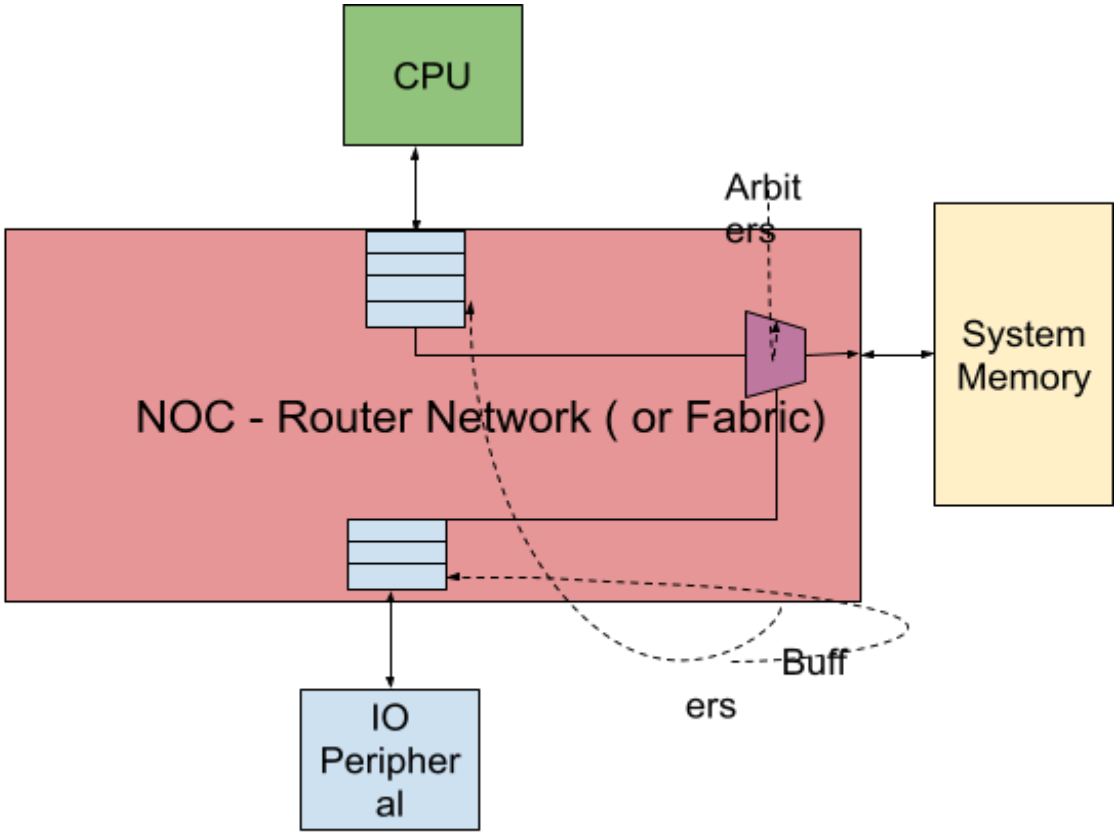

Fig 1.0: System on a Chip