# Society for Computer Technology and Research’s PUNE INSTITUTE OF COMPUTER TECHNOLOGY

## S.No.-27, Pune Satara Road, Dhankawadi, Pune-411043

**A.Y. 2023-24**

# Department of Computer Engineering

****

Laboratory Practice V

Batch: R2

Date: / 04 /24

**(SEMESTER-VIII)**

A REPORT ON

**“Implementation of Huffman Encoding on GPU”**

Under the guidance of

## Prof. A. A. Chandorkar

***Submitted by***

Aditya Magdum (41251)



**DEPARTMENT OF COMPUTER ENGINEERING**

## SCTR’s Pune Institute of Computer Technology Dhankawadi, Pune Maharashtra, 411043

**CERTIFICATE**

This is to certify that the SPPU Curriculum-based mini-project report entitled “Implementation of Huffman Encoding on GPU” submitted by Aditya Magdum (41251) has satisfactorily completed the curriculum-based mini-project under the guidance of Prof. A. A. Chandorkar towards the partial fulfillment of third year Computer Engineering Semester VIII, Academic Year 2023-24 of Savitribai Phule Pune University.

Date: April 2024 Prof. P. R. Patil

Place: Pune Subject Coordinator

# ACKNOWLEDGEMENTS

# I would like to express my heartfelt gratitude to several individuals and entities who played pivotal roles in the successful completion of this project. Firstly, I am deeply appreciative of the guidance and mentorship provided by Prof. A. A. Chandorkar Their expertise and unwavering support were instrumental in shaping the project's direction and ensuring its success. I also thank PICT, Pune for giving me this opportunity to perform this project as a part of my course. I extend my thanks to the data sources and repositories that made the Titanic dataset accessible, as this project would not have been possible without their contributions.

# I also acknowledge the online community, where I found valuable insights and assistance that enriched my understanding. Special thanks go to my collaborators and peers who offered their perspectives and expertise throughout the project. Each of these contributions, whether large or small, has been integral to this project's accomplishment.

# INDEX

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Content** | **Page No.** |
| 1 | Problem Statement | 5 |
| 2 | Motivation | 5 |
| 3 | Problem Scope | 6 |
| 4 | Objectives | 6 |
| 5 | Outcomes | 7 |
| 6 | Hardware and Software Requirements | 7 |
| 7 | Theory | 8 |
| 8 | Procedure | 9 |
| 9 | Code | 9 |
| 10 | Conclusion | 10 |

1. **PROBLEM STATEMENT**

To implement Huffman Encoding on GPU

# MOTIVATION

# The motivation behind implementing Huffman Encoding on GPU stems from the ever-increasing demand for efficient data compression techniques in modern computing environments. Traditional CPU-based implementations of Huffman Encoding face limitations in handling large-scale datasets and real-time processing requirements.

# By leveraging the parallel processing capabilities of GPUs, we aim to overcome these limitations and unlock significant performance improvements. GPUs offer a massively parallel architecture ideally suited for computationally intensive tasks like Huffman Encoding, enabling simultaneous processing of multiple symbols and encoding operations. This project not only addresses practical challenges in data compression but also contributes to the broader field of parallel computing, showcasing the potential of GPU acceleration in enhancing algorithmic efficiency.

# Additionally, exploring Huffman Encoding on GPU provides valuable insights into optimizing algorithms for heterogeneous computing platforms, aligning with the industry's growing emphasis on high-performance computing and parallel processing paradigms. Overall, this endeavor promises to yield tangible advancements in both computational efficiency and algorithmic innovation, underscoring its significance in the realm of modern data processing and parallel computing research.

1. **PROBLEM SCOPE**

# The scope of this project encompasses the design and implementation of a parallelized Huffman Encoding algorithm tailored for execution on GPU architectures. The primary focus lies in harnessing the computational power of GPUs to accelerate the encoding process and achieve significant improvements in compression efficiency. Specifically, the project involves adapting the sequential Huffman Encoding algorithm to leverage the parallel execution capabilities of GPU hardware, ensuring optimal utilization of resources while maintaining the integrity and correctness of the compression process. Furthermore, the scope extends to the development of efficient data structures and memory management techniques to facilitate seamless integration with GPU computing frameworks.

# Emphasis will be placed on achieving scalability and performance gains across a range of input data sizes and characteristics, thereby addressing the scalability challenges typically encountered in traditional CPU-based implementations. Throughout the project, rigorous testing and benchmarking will be conducted to evaluate the performance, scalability, and effectiveness of the GPU-accelerated Huffman Encoding solution across various use cases and datasets. By delineating a clear problem scope, this project aims to provide a comprehensive exploration of the capabilities and limitations of GPU-based acceleration for Huffman Encoding, paving the way for practical applications in data compression and parallel computing domains.

# OBJECTIVES

# Adapt the sequential Huffman Encoding algorithm to leverage parallel processing capabilities of GPU architectures.

# Develop efficient data structures and memory management techniques optimized for GPU execution.

# Achieve significant performance improvements in compression efficiency compared to traditional CPU-based implementations.

# Ensure scalability and effectiveness across a wide range of input data sizes and characteristics.

# Conduct rigorous testing and benchmarking to evaluate the performance and scalability of the GPU-accelerated Huffman Encoding solution.

# Explore practical applications of GPU-accelerated Huffman Encoding in data compression and parallel computing domains.

# Document findings, insights, and best practices for future reference and potential extension of the project.

# 

# OUTCOMES

# The expected outcomes of this report are:

# Improved compression efficiency: Achieving faster compression speeds and reduced processing times compared to CPU-based implementations.

# Scalability: Demonstrating the ability to handle large-scale datasets and adapt to varying input data characteristics effectively.

# Performance gains: Quantifying the speedup achieved through GPU acceleration and evaluating its impact on overall system performance.

# Resource utilization: Optimizing the utilization of GPU resources such as cores, memory, and bandwidth for efficient parallel execution.

# Practical applications: Identifying real-world scenarios where GPU-accelerated Huffman Encoding can be beneficial, such as in multimedia compression, data storage, and transmission.

# Insights into parallel computing: Gaining a deeper understanding of parallel programming concepts, GPU architecture, and optimization techniques through the implementation process.

# Contribution to research: Providing valuable insights and methodologies for future research in parallel computing, data compression, and algorithm optimization.

# HARDWARE AND SOFTWARE REQUIREMENTS

# 6.1 Hardware Requirements

# CPU

# Windows 11, 64 bits

# GPU (Nvidia)

# 8GB RAM

# 6.2 Software Requirements

# Python 3.0

# Google Colab

# CUDA

# THEORY

# 7.1 Overview

# Huffman Encoding is a widely used algorithm for lossless data compression. It works by assigning variable-length codes to input characters based on their frequencies, with more frequent characters receiving shorter codes.

# 7.2 Basic Concept

# The algorithm starts by building a binary tree called the Huffman tree, where each leaf node represents a character along with its frequency. The most frequent characters have shorter paths in the tree.

# 7.3 Tree Construction

# The Huffman tree is constructed using a greedy algorithm that repeatedly combines the two least frequent characters into a single node until all characters are included in the tree.

# 7.4 Code Generation

# After constructing the Huffman tree, the algorithm traverses the tree to assign binary codes to each character. The codes are generated by following the path from the root to each character, with '0' for left branches and '1' for right branches.

# 7.6 Decompression Process

# To decompress the data, the encoded binary stream is traversed through the Huffman tree. Starting from the root, each bit read determines whether to move to the left or right child until a leaf node (character) is reached, thus reconstructing the original data.

# 7.7 Optimization and Variants

# Various optimizations and variants of the basic Huffman algorithm exist, such as adaptive Huffman coding, which dynamically updates the tree as data is encoded or decoded, resulting in improved compression ratios for certain types of data.

# PROCEDURE

# Analyze input data to count character frequencies.

# Build a binary tree where characters are leaves and nodes represent merged characters based on frequency.

# Traverse the tree to assign binary codes to each character.

# Replace each character in the input data with its corresponding Huffman code.

# Traverse the tree using encoded bits to reconstruct the original data.

# Calculate compression ratio by comparing sizes of compressed and original data.

# Apply optimizations such as adaptive Huffman coding or GPU acceleration for improved performance.

# Test with various data sets to ensure correctness and evaluate performance.

# Document implementation details, optimizations, and performance results in a report.

# CODE and OUTPUT

# A screenshot of a computer Description automatically generated

# A screenshot of a computer Description automatically generated

# CONCLUSION

# In summary, our implementation of Huffman Encoding on GPU has yielded notable improvements in compression efficiency and performance. Leveraging parallel processing capabilities, we achieved promising results in compression ratios, speedup, and scalability compared to CPU-based methods. This project offers valuable insights into optimizing algorithms for parallel architectures and holds practical implications for various applications requiring efficient data compression. Further research could explore additional optimizations and broader applications of GPU-accelerated Huffman Encoding, highlighting the significance of GPU acceleration in advancing algorithmic efficiency and performance.