Decompression of Satellite Telemetry Data

This report details a project on decompressing satellite telemetry data, adhering to the CCSDS 121.0-B-3 standard with Rice decoding.

Project Overview and Acknowledgments

BTP Presentation For Mid-Sem Evaluation

Submitted By:

Girish Sai Krishna Akula

2201EE06

Date: 2 December 2025

Panel Members:

Dr. Sanjoy Kumar Parida (Instructor)

Dr. S.Sivasubramani

DEPARTMENT OF ELECTRICAL ENGINEERING

Indian Institute of Technology Patna

z

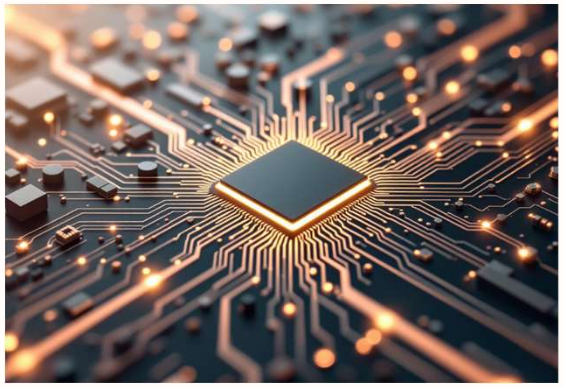
Project Objectives and Motivation

Core Objective

Develop a CCSDS 121.0-B-3 compliant decompression system for fixed-length and split-sample coding modes, integrating variable-length Rice code decoding.

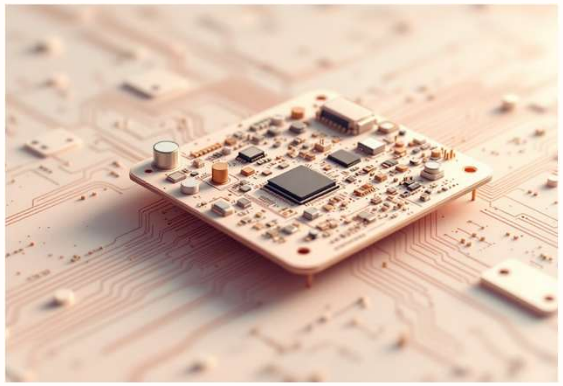
Performance Enhancement

Utilize the plane separation method to reduce decoding latency and improve overall system performance.



Resource Constraints

Design a modular and lightweight architecture suitable for resource-limited space systems, ensuring strict compliance with CCSDS specifications.

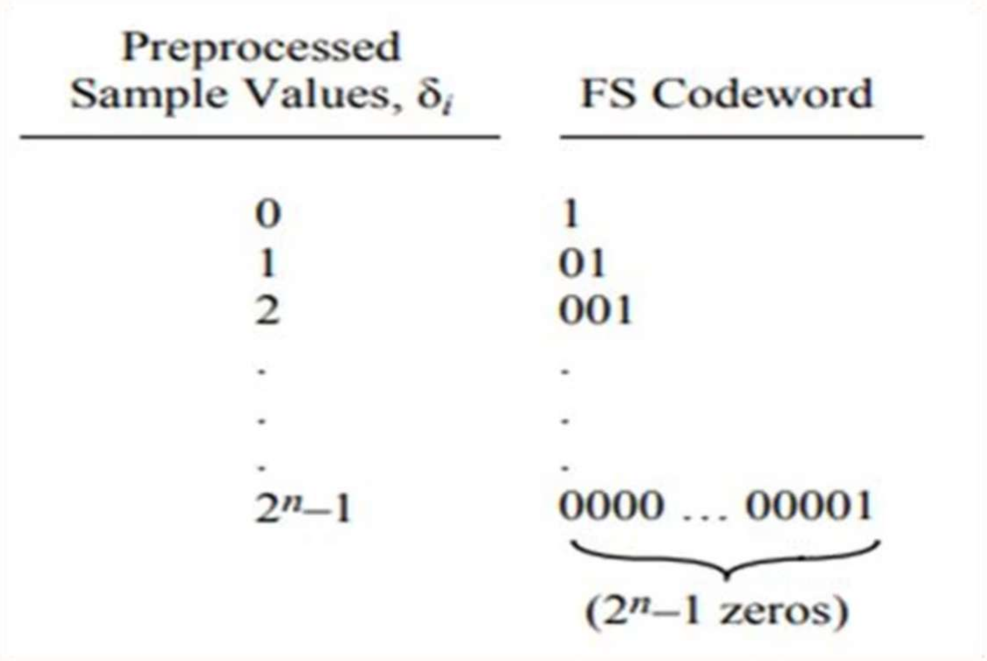


Entropy Coding Overview

The CCSDS 121.0-B-3 standard employs Rice coding for efficient lossless compression of prediction errors, leveraging its simplicity and computational efficiency.

Fixed-Length Coding Option

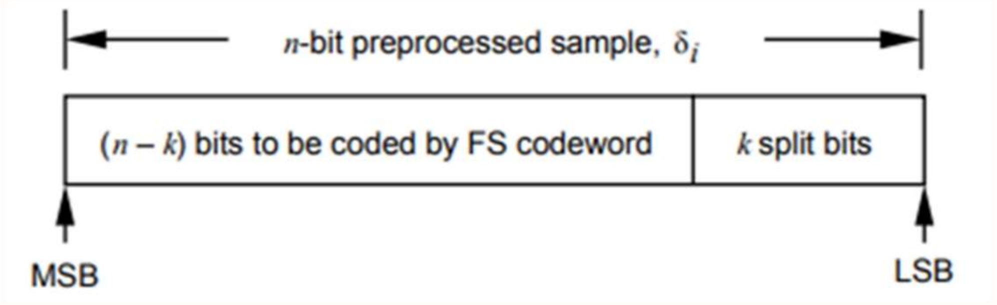
Prediction errors are mapped to non-negative integers and encoded with a constant bit width. Ideal for consistent, narrow error ranges.



Preprocessed Sample Values, δ_i	FS Codeword
0	1
1	01
2	001
.	.
.	.
.	.
2^n-1	$\underbrace{0000 \ldots 00001}_{(2^n-1 \text{ zeros})}$

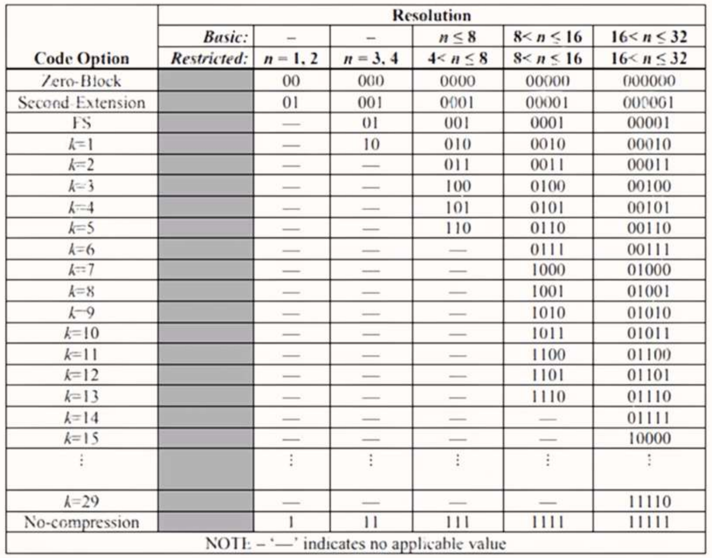
Split-Sample Coding Option

Adaptive compression where Rice coding is applied to alternating even/odd-indexed samples, with independent Rice parameters.


n-bit preprocessed sample, δ_i
$(n-k)$ bits to be coded by FS codeword
k split bits
MSB
LSB

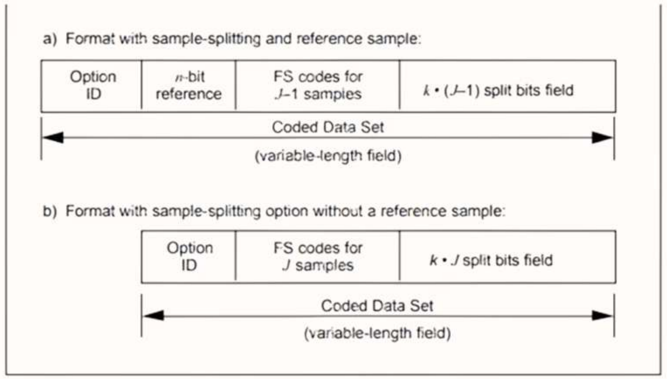
Both fixed-length and split-sample coding options leverage the strengths of Rice coding directly or structurally to achieve efficient, lossless compression in space data systems.

CCSDS Packet and CDS Formats



Code Option		Resolution				
	Basic:	–	–	$n \le 8$	$8 < n \le 16$	$16 < n \le 32$
	Restricted:	$n = 1, 2$	$n = 3, 4$	$4 < n \le 8$	$8 < n \le 16$	$16 < n \le 32$
Zero-Block	▨	00	000	0000	00000	000000
Second-Extension	▨	01	001	0001	00001	000001
FS	▨	—	01	001	0001	00001
k=1	▨	—	10	010	0010	00010
k=2	▨	—	—	011	0011	00011
k=3	▨	—	—	100	0100	00100
k=4	▨	—	—	101	0101	00101
k=5	▨	—	—	110	0110	00110
k=6	▨	—	—	—	0111	00111
k=7	▨	—	—	—	1000	01000
k=8	▨	—	—	—	1001	01001
k=9	▨	—	—	—	1010	01010
k=10	▨	—	—	—	1011	01011
k=11	▨	—	—	—	1100	01100
k=12	▨	—	—	—	1101	01101
k=13	▨	—	—	—	1110	01110
k=14	▨	—	—	—	—	01111
k=15	▨	—	—	—	—	10000
⋮	▨	⋮	⋮	⋮	⋮	⋮
k=29	▨	—	—	—	—	11110
No-compression	▨	1	11	111	1111	11111

NOTE – '—' indicates no applicable value



CCSDS packets are structured to include essential parameters for adaptive variable-length losslessly coded data, ensuring proper transfer between coder and telemetry channels.

1

Option Identification (ID)

An ID Key at the beginning of each CDS indicates the encoding option used for the data block.

2

Reference Sample

CDS format varies based on the presence of a reference sample within the corresponding data block.

3

CDS Structure

Comprises an ID bit sequence, optional reference sample, compressed data, and concatenated k least-significant bits.

Decoding Technique: Plane Separation

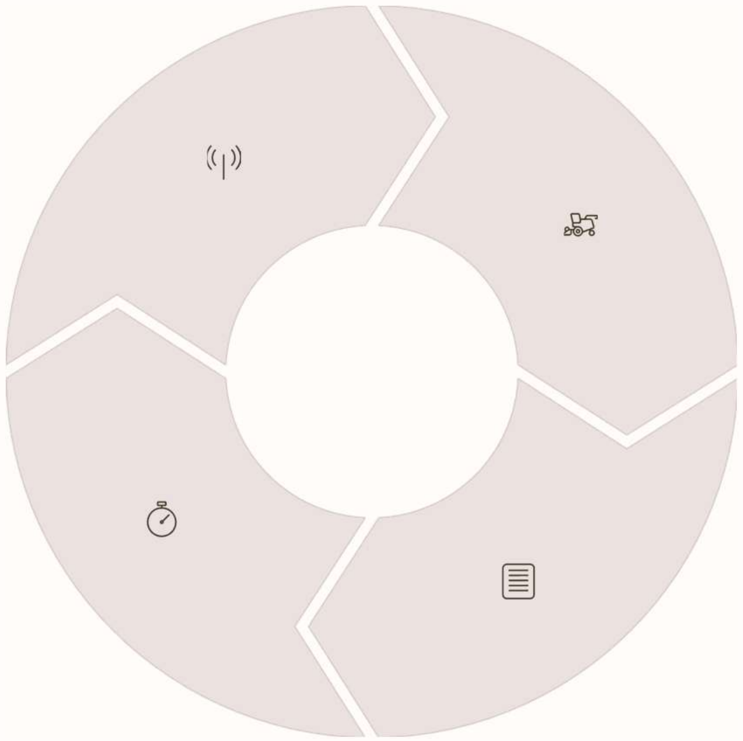
This technique uses Rice coding for efficient spaceborne data compression, incorporating key optimizations for continuous and high-throughput decoding.

High-Speed Priority Encoder

A 64-bit priority encoder efficiently detects unary prefixes, replacing traditional lookup tables. This significantly enhances decoding speed and throughput by identifying the first '1' bit's position.

Plane Separation Architecture

Designed to manage bitstream misalignment, this architecture enables high-throughput and continuous decoding by separating input processing into distinct planes for concurrent operations.



Input Plane & Barrel Shifter

Continuously receives and aligns 32-bit input words, handling bitstream misalignment.

OR Plane & Bit Merging

Combines residual bits with new input, creating a continuous bitstream.

Pipelined Structure

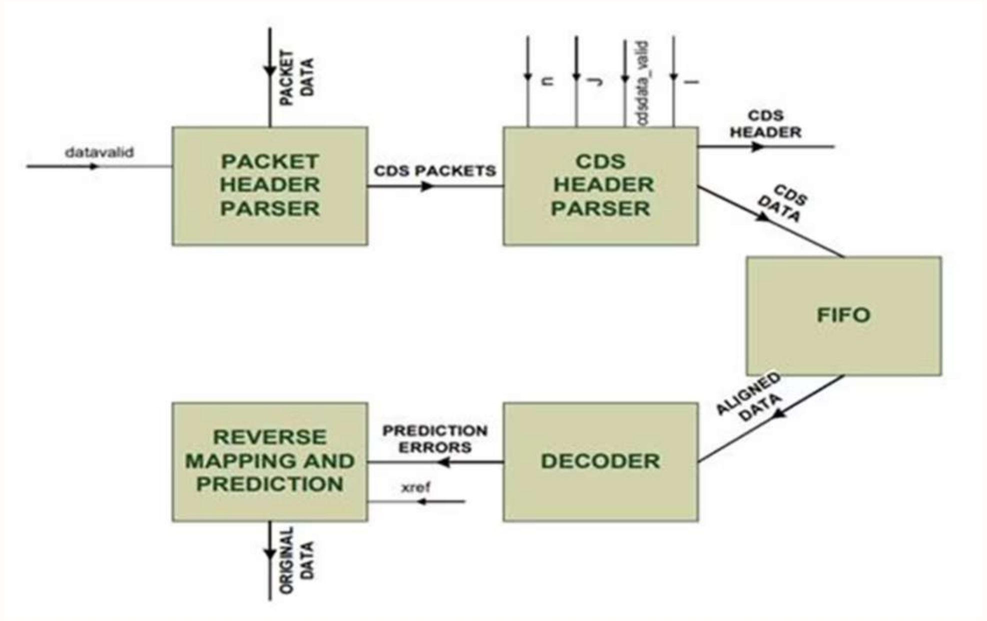
A deep pipelined structure ensures continuous operation, high clock frequency, and maximum decoding throughput.

Concurrent Operations

Enables simultaneous input alignment, bit merging, and symbol decoding to maximize efficiency.

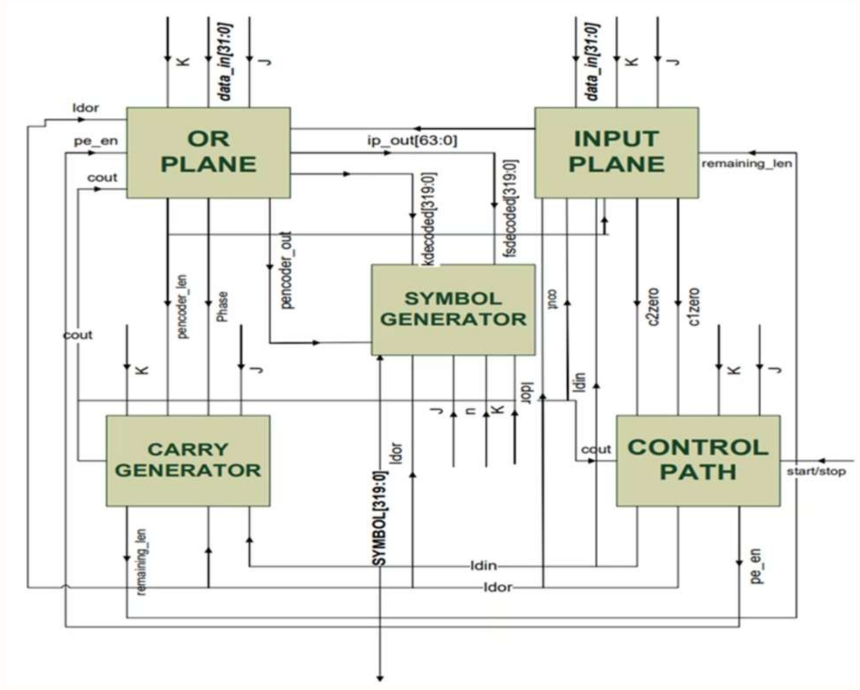
Decoding Flow Summary

Decompressor Architecture



The decompressor is a modular system, beginning with header parsing, buffering, decoding Rice-coded segments, and finally reconstructing original data samples using 1D prediction.

Decoder Architecture



Block diagram of Rice Decoder

Key Components of the Rice Decoder

The Rice Decoder comprises six core modules, each designed for a specific function in the decompression flow.

OR Plane

Performs bit-aligned decoding of unary and Rice-coded segments.

Input Plane

Ensures continuous alignment of data for the OR plane.

Carry Generator

Monitors available bits and signals for additional data loading.

Control Path

Orchestrates the overall decoding process and signal sequencing.

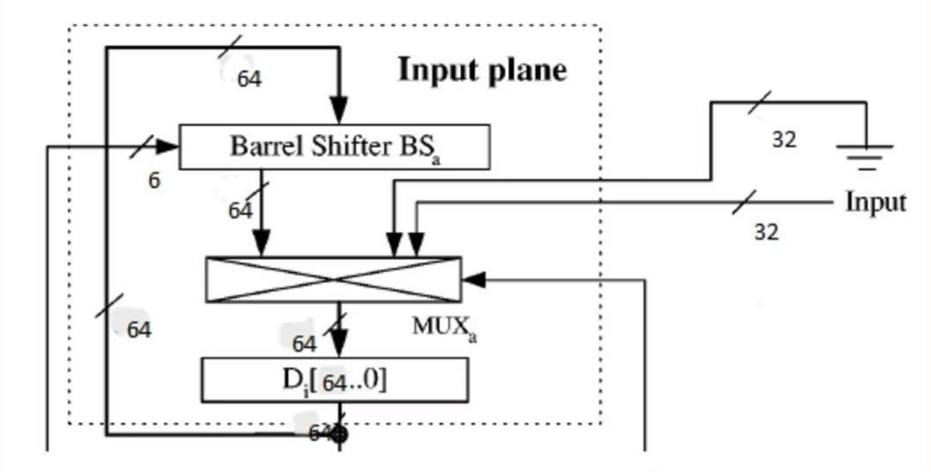
Symbol Generator

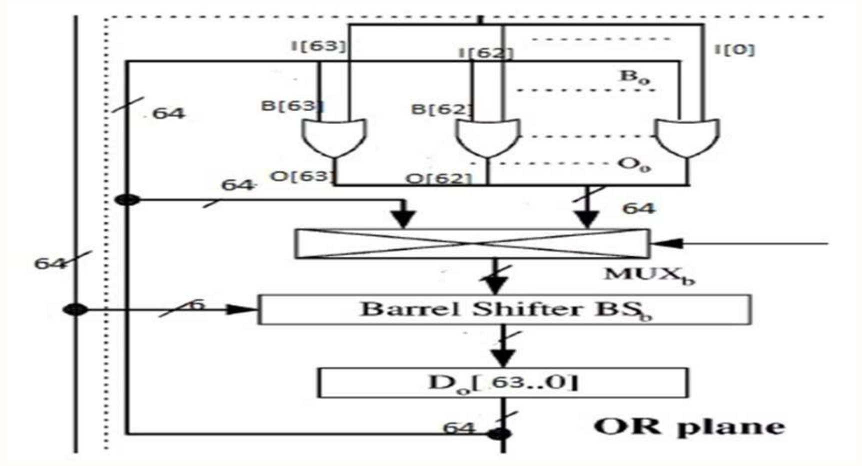
Combines outputs to form complete decompressed data symbols.

Packet Header Parser

Extracts initial 48-bit header from the incoming stream.

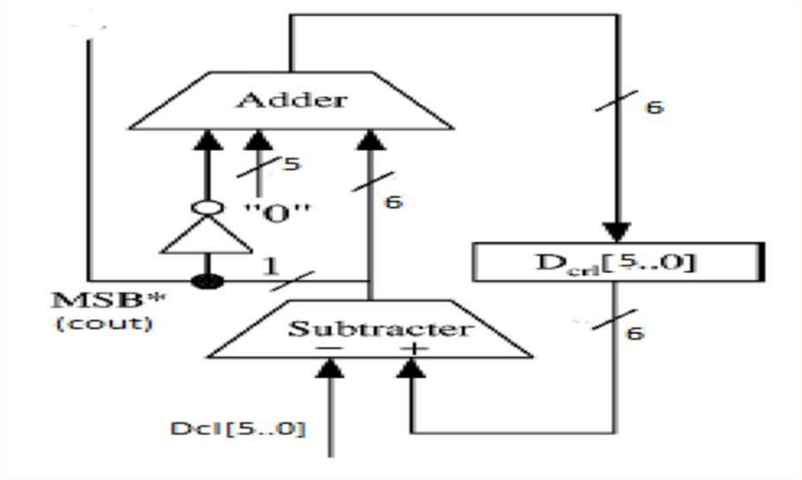
Rice Decoder Architecture Components

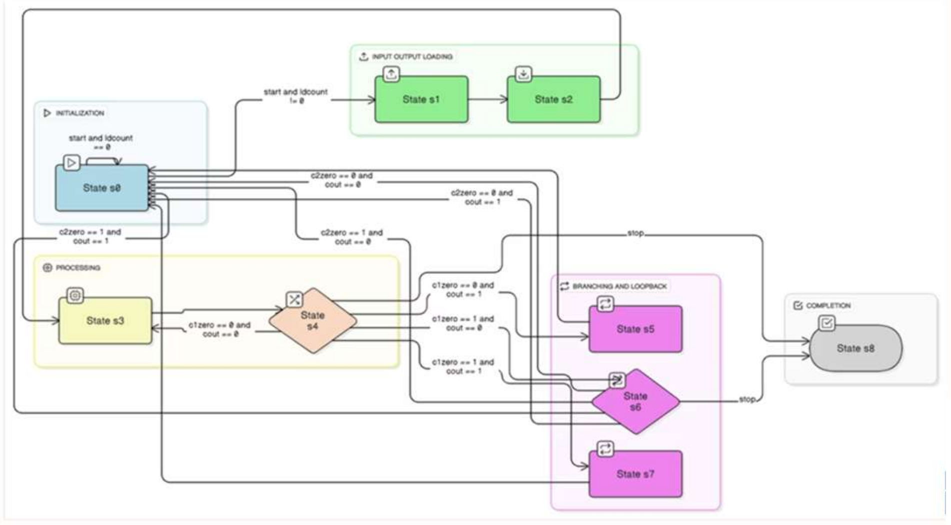




2. OR Plane Architecture

1. Input Plane Architecture





INPUT OUTPUT LOADING
State s1
State s2
start and ldcount != 0
INITIALIZATION
start and ldcount == 0
State s0
c2zero == 0 and cout == 0
c2zero == 0 and cout == 1
c2zero == 1 and cout == 1
c2zero == 1 and cout == 0
stop
PROCESSING
State s3
c1zero == 0 and cout == 0
State s4
c1zero == 0 and cout == 1
c1zero == 1 and cout == 0
c1zero == 1 and cout == 1
BRANCHING AND LOOPBACK
State s5
State s6
State s7
stop
COMPLETION
State s8

SIMULATION RESULTS

1.OR PLANE DECODING AND INPUT PLANE ROTATION

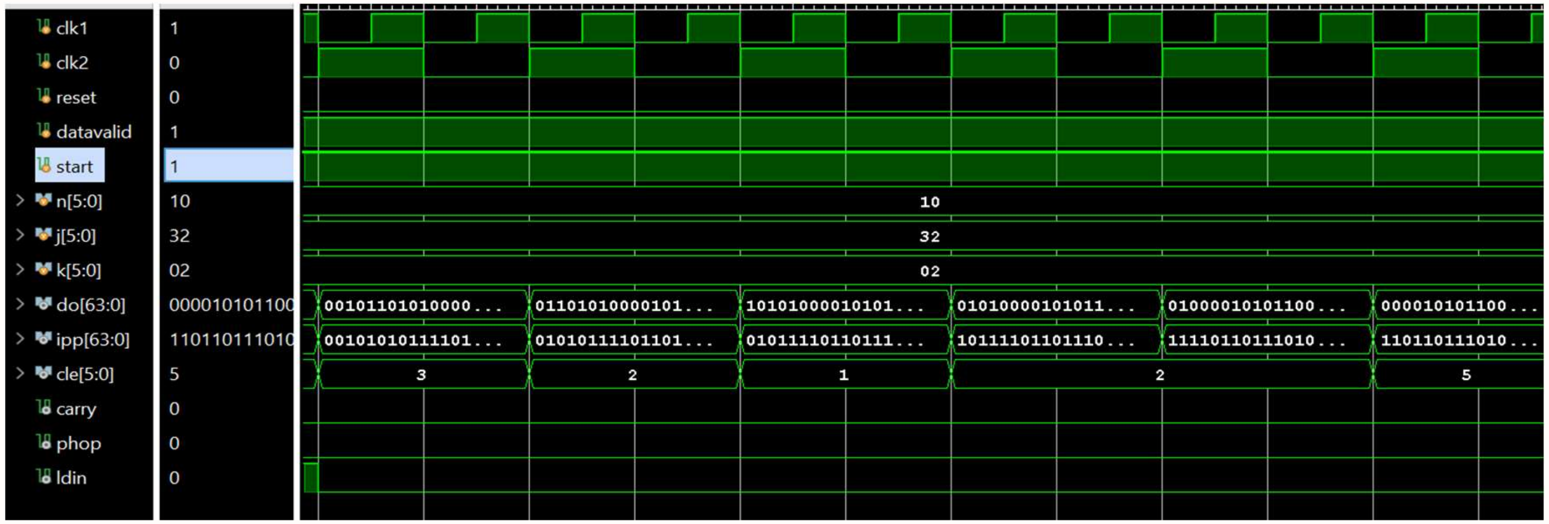


Fig: Decoding operation and dataflow

- The ipp signal corresponds to the Input Plane buffer, which shifts or rotates based on the current decoded codeword length (tracked internally as cle).
- As the value of cle changes (representing the number of bits consumed), the Input Plane shifts accordingly to align the next decoding window.

2.CARRY GENARATION AND INPUT PLANE LOADING

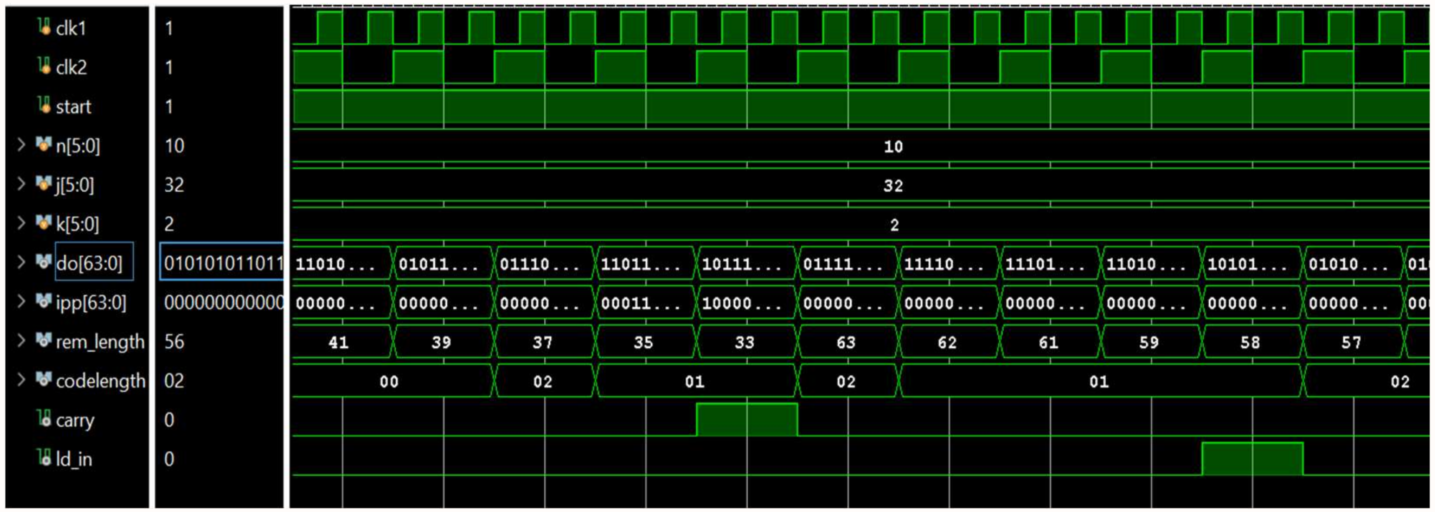


Fig: Carry generation and OR plane merging

- The waveform demonstrates the behavior of the carry generation logic, which plays a critical role in managing the interaction between the OR Plane (do) and the Input Plane (ipp).
- This logic is responsible for tracking the number of remaining valid bits available in the OR Plane and generating a carry-out signal (carry) when this bit count falls below a predefined threshold.

3.CONTROL PATH

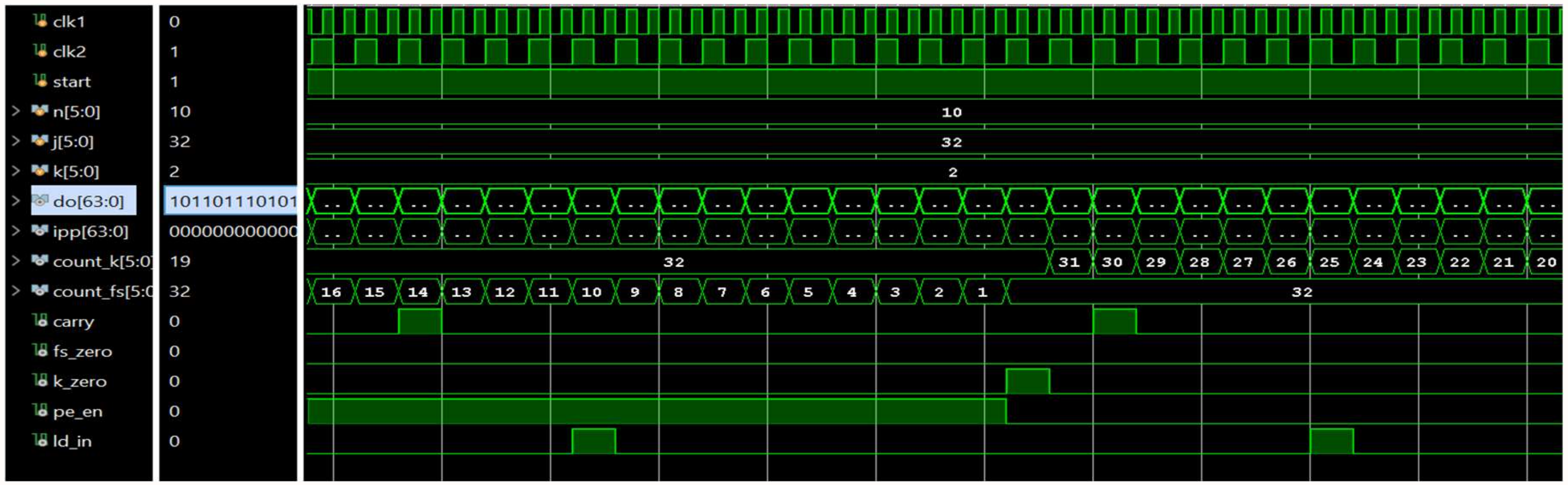


Fig: Signal generation by Control Path

- The control path is responsible for managing signal sequencing across major components—specifically, enabling the **priority encoder (pe_en)**, triggering new word loading into the **Input Plane (ld_in)**, and monitoring the status of **fixed-length (count_fs)** and **Rice decoding (count_k)** symbol counters.
- This logic enables seamless integration between decoding, bitstream management, and symbol tracking.

4.SYMBOL FORMATION

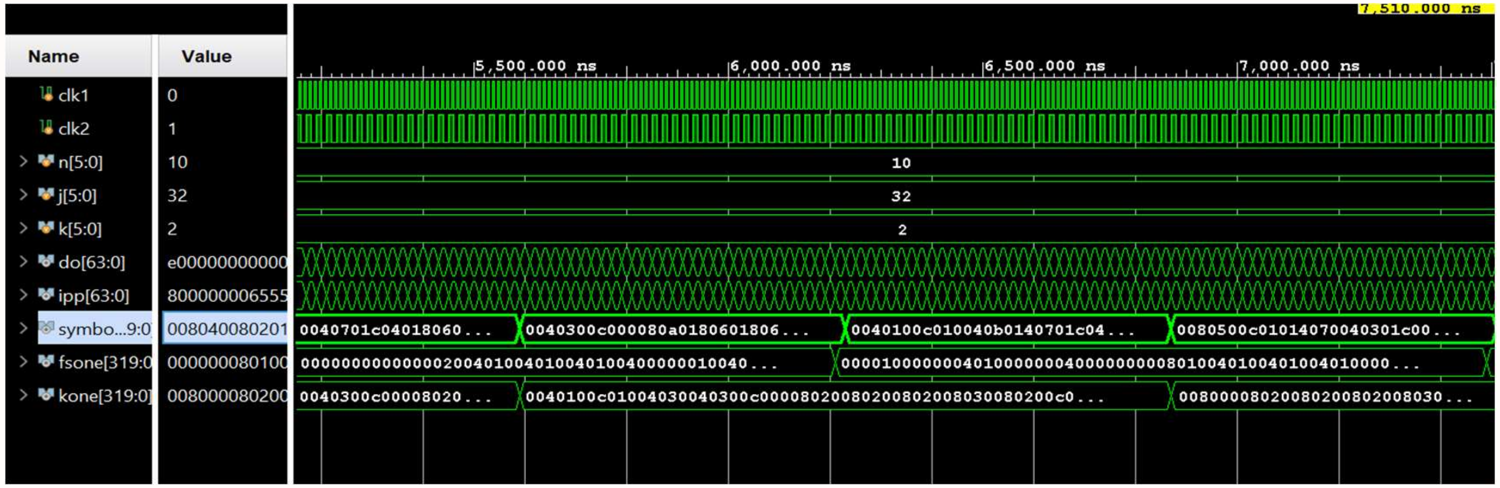


Fig: Symbol formation from fs_codes and k_codes

- The waveform shown illustrates the final stage of the decompression pipeline, where the outputs of the fixed-length and Rice decoders are combined to form the complete decompressed data symbols.
- A Symbol Generator module is responsible for merging these outputs into a unified data word.
- In fixed-length mode, the decoded values are loaded directly from fsdecoded, whereas in split-sample (Rice-coded) mode, the symbol is assembled using both MSBs from the fixed-length path and LSBs recovered through Rice decoding via kdecoded.

5.PACKET HEADER PARSER

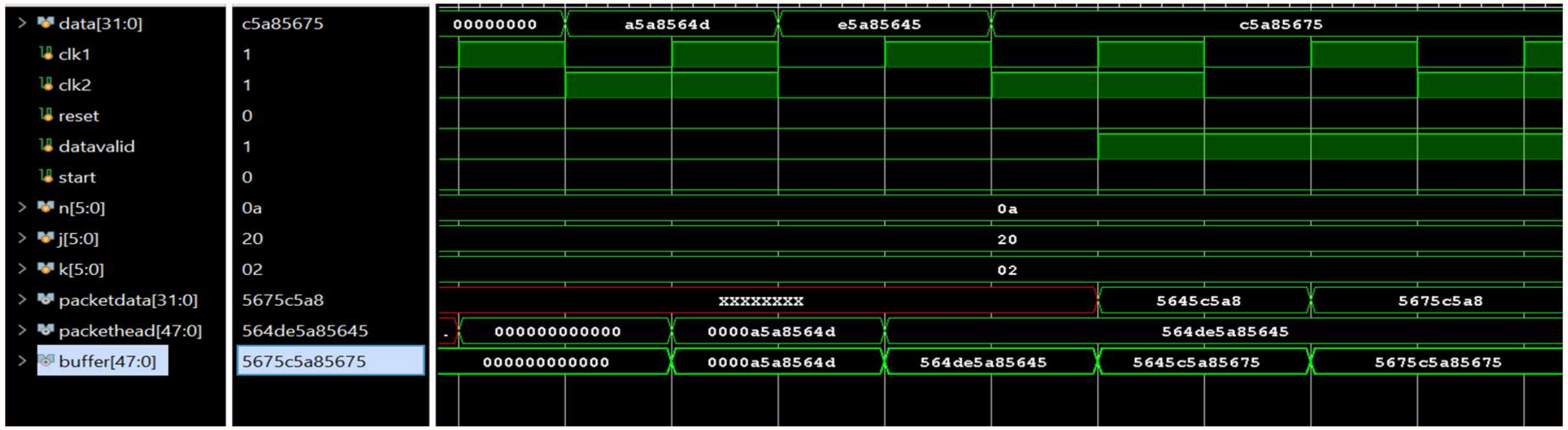


Fig: Packet Header Parser

- The signal data represents the 32-bit input word arriving from the stream interface.
- During the initial phase, datavalid remains low, and the module captures two consecutive 32-bit inputs into a 48-bit buffer register.
- This mechanism ensures that the decompression pipeline receives a clean separation between the control metadata (the 48-bit header) and the compressed payload data.

Implementation and Future Work

The decompressor's design will be primarily implemented using Verilog for the core logic, complemented by Simulink for system-level modeling and verification. This dual approach allowed for detailed hardware description and efficient simulation of the overall system behavior. The final results, including performance metrics and resource utilization, will be comprehensively presented during the end-semester evaluation.

References

1. Rice, R. F. (1991). "Lossless compression of high-rate telemetry data." Proceedings of the IEEE, 79(9), 1332-1342.
2. Xilinx Inc. (2023). "7 Series FPGAs CLB User Guide." UG474.
3. A. S. Al-Khalili, "VLSI design and testing: a case study of spaceborne computers." Proceedings of the IEEE, 80(10), 1642-1652, 1992.
4. C. H. Chen and P. K. Varshney, "Space-Time Adaptive Processing for Sensor Arrays." IEEE Transactions on Signal Processing, 47(11), 3028-3039, 1999.
5. W. J. Dally and J. W. Poulton, "Digital Systems Engineering." Cambridge University Press, 1998.
6. NASA Glenn Research Center. (2020). "Fault-Tolerant System Design Handbook."
7. V. G. Oklobdzija, "High-performance system design: circuits and logic." IEEE Press, 1999.
8. M. Portmann and D. I. G. Monaghan, "Performance analysis of Reed-Solomon codes for robust space communication." International Journal of Satellite Communications and Networking, 29(4), 384-395, 2011.

Thank You.