NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

PROGRAM OF POSTGRADUATE STUDIES

PhD THESIS

Architectures and implementation in FPGA technology of hardware accelerators for forward error correction encoding in on-board processing data-chains for aerospace applications

Dimitrios K. Theodoropoulos

ATHENS

MAY 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αρχιτεκτονικές και υλοποίηση σε τεχνολογία FPGA επιταχυντών υλικού για κωδικοποίηση διόρθωσης σφαλμάτων σε συστήματα επεξεργασίας δεδομένων εν πτήσει για αεροδιαστημικές εφαρμογές

Δημήτριος Κ. Θεοδωρόπουλος

ΑΘΗΝΑ

ΜΑΪΟΣ 2023

PhD THESIS

Architectures and implementation in FPGA technology of hardware accelerators for forward error correction encoding in on-board processing data-chains for aerospace applications

Dimitrios K. Theodoropoulos

SUPERVISOR: Antonis Paschalis, Professor, National and Kapodistrian University of Athens

THREE-MEMBER ADVISORY COMMITTEE:

Antonis Paschalis, Professor, National and Kapodistrian University of Athens

Dimitrios Gizopoulos, Professor, National and Kapodistrian University of Athens

Nektarios Kranitis, Associate Professor National and Kapodistrian University of Athens

SEVEN-MEMBER EXAMINATION COMMITTEE

Antonis Paschalis,

Professor, National and Kapodistrian University of Athens

Dimitrios Gizopoulos,
Professor, National and Kapodistrian University of Athens

Nektarios Kranitis,

Associate Professor National and Kapodistrian University of Athens

Takis Mathiopoulos,
Professor National and Kapodistrian University of Athens

George Alexandropoulos,

Assistant Professor, National and Kapodistrian University of Athens

Mihalis Psarakis,
Assistant Professor, University of Piraeus

Dimitrios Soudris,
Professor, National Technical University of Athens

Examination Date: May 3, 2022

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Αρχιτεκτονικές και υλοποίηση σε τεχνολογία FPGA επιταχυντών υλικού για κωδικοποίηση διόρθωσης σφαλμάτων σε συστήματα επεξεργασίας δεδομένων εν πτήσει για αεροδιαστημικές εφαρμογές

Δημήτριος Κ. Θεοδωρόπουλος

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Αντώνης Πασχάλης, Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

ΤΡΙΜΕΛΗΣ ΕΠΙΤΡΟΠΗ ΠΑΡΑΚΟΛΟΥΘΗΣΗΣ:

Αντώνης Πασχάλης, Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Δημήτριος Γκιζόπουλος, Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Νεκτάριος Κρανίτης, Αναπληρωτής Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

ΕΠΤΑΜΕΛΗΣ ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ

Αντώνης Πασχάλης,

Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Δημήτριος Γκιζόπουλος,
Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Νεκτάριος Κρανίτης,

Αναπληρωτής Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Παναγιώτης Μαθιόπουλος,
Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Γεώργιος Αλεξανδρόπουλος,

Επίκουρος Καθηγητής, Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Μιχαήλ Ψαράκης,
Αναπληρωτής Καθηγητής, Πανεπιστήμιο Πειραιώς

Δημήτριος Σούντρης,
Καθηγητής Εθνικό Μετσόβιο Πολυτεχνείο

Ημερομηνία Εξέτασης: 03 Μαΐου 2022

ABSTRACT

A Forward Error Correction (FEC) encoder is an integral part of an on-board processing data chain. The current Thesis deals with the problem of the design and VLSI implementation of efficient hardware encoder architectures for such systems. More specifically, two focus areas have been identified and targeted: bit-level channel coding, which is applied mostly to telemetry data transfer in near-earth and deep-space communications and packet-level erasure coding, which has emerged as a promising approach for high data rate optical space communications, or for intermittent connectivity scenarios.

Regarding the first focal point of the hereto described work, Quasi-Cyclic Low-Density Parity-Check Codes (QC-LDPC) have been adopted by the Consultative Committee for Space Data Systems (CCSDS) as recommended standard for on-board channel coding in Near-Earth and Deep-Space communications. It is shown, however, that the encoder architectures which have been so far proposed for other error-correction schemes, are either altogether inapplicable to the CCSDS QC-LDPC codes, or their direct application comes with significant performance penalties that render them inefficient for high-throughput hardware implementations. In the work presented in this thesis, a novel architecture for the multiplication of a dense QC matrix with a bit vector, which is a fundamental operation of QC-LDPC encoding, is proposed. Based on this architecture, efficient encoders for CCSDS codes are proposed, according to all the applicable LDPC encoding methods, which are analytically described and compared in terms of resource utilization efficiency for the CCSDS QC-LDPC codes. Moreover, in the special case of the specific code defined in the CCSDS standard for Near-Earth communications, specialized techniques are also introduced, which efficiently handle the challenges arising from the generator's matrix circulant size (511 bits).

The proposed architectures have been implemented in various Field Programmable Gate Array (FPGA) technologies and extensively validated and tested in the commercial counterpart of the Xilinx space-grade Kintex UltraScale XQRKU060 FPGA. The achieved performance defines the state-of-the-art in this area, being able to achieve up to more that 70 times higher encoding throughput than the corresponding implementations by NASA/JPL, on the same device and with a low resource budget. It is also the first work to introduce an extensive and realistic testing framework including modern SpaceFibre data links, in order to be as close as possible to a real mission system. Together with the detailed power measurements provided, the current work breaks untouched ground for the adoption of the CCSDS channel codes in application areas from which they had been so far considered unfavorable, due to their encoding complexity, like the upcoming high-performance Free-Space Optical space communication standards.

Protograph based QC-LDPC codes are widely considered an advantageous option for forward error correction (FEC) on magnetic recording (MR) media as well, because of their excellent performance characteristics and their inherent possibilities efficient implemen-

tation. The vast majority of related research, however, has so far been focused on the analytical optimization of code design and algorithms. Although high-speed encoding and decoding with low hardware footprint are important for MR media, hardware implementations for such encoding schemes have so far been scarce. Leveraging the architecture of LDPC encoders for space applications, efficient encoder designs for the protograph-based LDPC codes proposed for MR media are also introduced. The proposed designs are implemented in hardware as FPGA accelerators. Their efficiency is demonstrated on an FPGA development board, achieving multi-Gbps of encoding throughput, adequate for modern MR application standards. This is the first time such a study has been conducted and could prove revolutionary in the field.

Packet-level erasure coding has been considered by the CCSDS in the 131.5-O-1 experimental specification for application in high data rate near-earth and deep-space communications, since it can protect against long error bursts as they may come along with the effect of scintillation outages or transmission errors. However, implementations of packet-level encoding and decoding so far exist only in software, running on a general-purpose CPU, with limitations on the achievable performance, resource and power. In the second focus area of this work, architectures for hardware acceleration of packet-level encoding function are introduced, that allow integration on a high-speed on-board data processing chain with very low footprint and power consumption. The hardware implementations have been validated and the efficiency of the proposed architectures has been demonstrated on a Xilinx KCU105 development board, reaching an encoding throughput of over 13Gbps. Apart from offloading packet-level encoding from the on-board embedded processor, the proposed accelerators are shown to achieve a significant speedup (over 80 times), when compared with on-board software implementations of the corresponding NASA algorithms of the ION delay tolerant network (DTN) implementation, by porting on some of the most commonly used and state-of-the-art space-qualified embedded processors. This is the first documented hardware implementation of packet-level encoders and the first time that encoding throughput performance and power baselines are recorded. As with the channel codes of the first thematic area of this thesis, these findings unlock new horizons for the re-evaluation of packet-level erasure codes for use in the upcoming high-performance Free-Space Optical space communication standards.

SUBJECT AREA: Digital Design and Computer Architecture, Embedded Systems, Space Communication Systems

KEYWORDS: Hardware Design, LDPC encoders, Hardware accelerators, Field Programmable Gate Arrays, CCSDS, Packet-Level encoders, Space Communications

ΠΕΡΙΛΗΨΗ

Η κωδικοποίηση διόρθωσης σφαλμάτων είναι αναπόσπαστο μέρος μιας ενσωματωμένης αλυσίδας επεξεργασίας δεδομένων ενός συστήματος εν πτήσει. Η παρούσα διατριβή πραγματεύεται το πρόβλημα του σχεδιασμού και της VLSI υλοποίησης αποδοτικών αρχιτεκτονικών κωδικοποιητών υλικού για τέτοιες αλυσίδες επεξεργασίας δεδομένων. Πιο συγκεκριμένα, δύο τομείς ενδιαφέροντος έχουν στοχευτεί: η κωδικοποίηση καναλιού σε επίπεδο bit, κυρίως για τη μεταφορά δεδομένων τηλεμετρίας σε επικοινωνίες κοντά στη γη και στο βαθύ διάστημα και η κωδικοποίηση επιπέδου πακέτων, η οποία έχει αναδειχθεί ως μια πολλά υποσχόμενη προσέγγιση για υψηλό ρυθμό δεδομένων επικοινωνίες οπτικού χώρου ή για σενάρια διακοπτόμενης συνδεσιμότητας.

Όσον αφορά την πρώτη ερευνητική περιοχή της εργασίας, οι κώδικες Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) έχουν προτυποποιηθεί από τη Συμβουλευτική Επιτροπή για Συστήματα Διαστημικών Δεδομένων (CCSDS) για την κωδικοποίηση καναλιού σε επικοινωνίες κοντά στη Γη και στο Βαθύ Διάστημα. Μετά από ενδελεχή μελέτη του συνόλου της υφιστάμενης βιβλιογραφίας, αποδεικνύεται ωστόσο στην παρούσα ότι οι αρχιτεκτονικές κωδικοποιητών που έχουν προταθεί μέχρι τώρα για άλλα σχήματα διόρθωσης σφαλμάτων, είτε είναι παντελώς ανεφάρμοστες στους κώδικες CCSDS QC-LDPC, είτε η άμεση εφαρμογή τους συνοδεύεται από τόσο μειωμένη απόδοση που καθίστανται αναποτελεσματικές για υλοποιήσεις υψηλών απαιτήσεων σε ταχύτητες κωδικοποίησης. Κατά συνέπεια, προτείνεται μια νέα αρχιτεκτονική για τον πολλαπλασιασμό ενός πυκνού QC πίνακα με ένα διάνυσμα από bits, πράξη η οποία αποτελεί θεμελιώδη λειτουργία της κωδικοποίησης QC-LDPC Με βάση αυτή την αρχιτεκτονική, προτείνονται αποδοτικοί κωδικοποιητές για κώδικες CCSDS, σύμφωνα με όλες τις εφαρμοστέες μεθόδους κωδικοποίησης LDPC, οι οποίες περιγράφονται αναλυτικά και συγκρίνονται ως προς την αποδοτικότητα χρήσης πόρων για τους συγκεκριμένους κώδικες. Επιπλέον, στην ειδική περίπτωση του συγκεκριμένου κώδικα που ορίζεται στο πρότυπο CCSDS για επικοινωνίες κοντά στη Γη, εισάγονται εξειδικευμένες τεχνικές, οι οποίες χειρίζονται αποτελεσματικά τις προκλήσεις που προκύπτουν από το μέγεθος των υποπινάκων του QC πίνακα-γεννήτορα του κώδικα (511 bit).

Οι προτεινόμενες αρχιτεκτονικές υλοποιήθηκαν σε διάφορες τεχνολογίες FPGA και επικυρώθηκαν και δοκιμάστηκαν εκτενώς στο εμπορικό αντίστοιχο της διαστημικής κατηγορίας Kintex UltraScale της Xilinx (XQRKU060), το οποίο περιλαμβάνεται στην αναπτυξιακή κάρτα KCU105. Η απόδοση που επιτυγχάνεται αποτελεί ορόσημο στον τομέα, καθώς μπορεί να επιτύχει έως και 70 φορές υψηλότερη απόδοση από τις αντίστοιχες προτάσεις της NASA/JPL, όταν υλοποιηθεί στο ίδιο FPGA, και με χαμηλό προϋπολογισμό πόρων υλικού και κατανάλωσης ισχύος. Είναι επίσης η πρώτη εργασία που εισάγει ένα εκτεταμένο και ρεαλιστικό πλαίσιο δοκιμών που περιλαμβάνει σύγχρονες συνδέσεις δεδομένων SpaceFibre, προκειμένου να είναι όσο το δυνατόν πιο κοντά σε ένα πραγματικό σύστημα. Μαζί με τις λεπτομερείς μετρήσεις ισχύος που παρέχονται, η τρέχουσα εργασία ανοίγει νέες δυνατότητες για την υιοθέτηση των κωδικών του CCSDS σε εφαρμογές από τις οποί-

ες μέχρι στιγμής θεωρούνταν απαγορευτικές, λόγω της πολυπλοκότητας κωδικοποίησης τους, όπως τα επερχόμενα πρότυπα οπτικών επικοινωνιών ελεύθερου χώρου υψηλής απόδοσης του CCSDS.

Οι κώδικες QC-LDPC που βασίζονται σε πρωτογράφους θεωρούνται ευρέως μια πλεονεκτική επιλογή κωδικοποίησης διόρθωσης σφαλμάτων (FEC), επίσης και σε μέσα μαγνητικής εγγραφής (MR), λόγω των εξαιρετικών χαρακτηριστικών απόδοσης και των εγγενών δυνατοτήτων αποτελεσματικής υλοποίησής τους στο υλικό. Ωστόσο, η συντριπτική πλειονότητα της σχετικής έρευνας έχει επικεντρωθεί μέχρι στιγμής στην αναλυτική βελτιστοποίηση του σχεδιασμού των κωδίκων και των συναφών αλγορίθμων. Αν και η κωδικοποίηση και η αποκωδικοποίηση υψηλής ταχύτητας με χαμηλό αποτύπωμα υλικού είναι σημαντικές για τα μέσα MR, δεν υφίστνται επί του παρόντος αναφορές σε μελέτες υλοποιήσεων. Αξιοποιώντας την αρχιτεκτονική των κωδικοποιητών LDPC για διαστημικές εφαρμογές, εισάγονται προσαρμοσμένες αρχιτεκτονικές για τους αντίστοιχους κώδικες MR. Οι προτεινόμενες αρχιτεκτονικές υλοποιούνται σε υλικό ως επιταχυντές FPGA. Η αποτελεσματικότητά τους αποδεικνύεται στην πλακέτα ανάπτυξης FPGA ZC706, επιτυγχάνοντας απόδοση πολλαπλών Gbps, επαρκή για τα σύγχρονα πρότυπα μαγνητικής αποθήκευσης. Η παρούσα εργασία είναι η πρώτη μελέτη που καταγράφεται στον συγκεκριμένο χώρο.

Η κωδικοποίηση διαγραφής σε επίπεδο πακέτων έχει προταθεί από το CCSDS στην πειραματική προδιαγραφή 131.5-O-1 για εφαρμογή σε διαστημικές επικοινωνίες υψηλού ρυθμού διαμεταγωγής στο εγγύς και στο βαθύ διάστημα, καθώς μπορεί να προστατεύσει από μεγάλες εκρήξεις σφαλμάτων, συνέπεια διαλείψεων σπινθηρισμού ή σφαλμάτων μετάδοσης. Ωστόσο, εφαρμογές κωδικοποίησης και αποκωδικοποίησης σε επίπεδο πακέτων υπάρχουν μέχρι στιγμής μόνο σε λογισμικό, που εκτελείται σε CPU γενικής χρήσης, με περιορισμούς στην επιτεύξιμη απόδοση, τους πόρους και την ισχύ. Στη δεύτερη περιοχή ενδιαφέροντος αυτής της εργασίας, εισάγονται αρχιτεκτονικές για την επιτάχυνση υλικού της λειτουργίας κωδικοποίησης σε επίπεδο πακέτων, που επιτρέπουν την ενσωμάτωση σε μια αλυσίδα επεξεργασίας δεδομένων υψηλής ταχύτητας με πολύ χαμηλό αποτύπωμα υλικού και περιορισμένη κατανάλωση ενέργειας. Οι υλοποιήσεις αυτές δοκιμάστηκαν και η αποτελεσματικότητά τους τεκμηριώθηκε στην πλακέτα ανάπτυξης Xilinx KCU105, η οποία περιλαμβάνει το εμπορικό ισοδύναμο του πιστοποιημένου για διαστημικές εφαρμογές Xilinx Kintex UltraScale (XQRKU060), επιτυγχάνοντας απόδοση κωδικοποίησης άνω των 13 Gbps. Εκτός από την ελάφρυνση του επεξεργαστικού φορτίου του κεντρικού επεξεργαστή, μέσω ανάθεσης της κωδικοποίησης επιπέδου πακέτων στους εξειδικευμένους επιταχυντές υλικού, οι προτεινόμενοι κωδικοποιητές επιτυγχάνουν σημαντική επιτάχυνση (πάνω από 80 φορές), σε σχέση τις αντίστοιχες λύσεις που περιλαμβάνονται στη σουίτα λογισμικού για δίκτυα διαλειπτόμενης συνδεσιμότητας της NASA (ION DTN), όταν εκτελεστούν σε μερικές από τις πιο κοινές και σύγχρονες ενσωματωμένες CPU για διαστημικές αποστολές, με ταυτόχρονη εξοικονόμηση ισχύος (πάνω από 3-5 φορές βελτιωμένο πηλίκο ρυθμαπόδοσης ανά μονάδα καταναλισκομένης ισχύος). Αυτή είναι η πρώτη και μοναδική μέχρι στιγμής καταγεγραμμένη υλοποίηση κωδικοποιητών επιπέδου πακέτου και ειδικότερα της πειραματικής προδιαγραφής 131.5-O-1 σε υλικό. Είναι επίσης η πρώτη ουσιαστική μελέτη των παραμέτρων απόδοσης και ισχύος. Όπως και με τους κώδικες καναλιών της πρώτης θεματικής περιοχής αυτής της διατριβής, αυτά τα ευρήματα ξεκλειδώνουν νέους

ορίζοντες για την επαναξιολόγηση των κωδίκων διαγραφής σε επίπεδο πακέτων για χρήση σε διαστημικές εφαρμογές υψηλής απόδοσης.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Ψηφιακή Σχεδίαση και Αρχιτεκτονική Υπολογιστών, Ενσωματωμένα Συστήματα, Συστήματα Διαστημικών Επικοινωνιών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Ψηφιακή Σχεδίαση, Κωδικοποιητές LDPC, Επιταχυντές Υλικού, FPGA, CCSDS, Κωδικοποιητές διόρθωσης σφαλμάτων επιπέδου πακέτου, Διαστημικές επικοινωνίες

ΣΥΝΟΠΤΙΚΗ ΠΑΡΟΥΣΙΑΣΗ ΤΗΣ ΔΙΔΑΚΤΟΡΙΚΗΣ ΔΙΑΤΡΙΒΗΣ

Η παρούσα διατριβή, όπως περιγράφεται και από τον τίτλο της, πραγματεύεται αρχιτεκτονικές υλικού για κωδικοποίηση διόρθωσης σφαλμάτων σε συστήματα επεξεργασίας δεδομένων σε αεροδιαστημικές εφαρμογές. Μελετήθηκαν δύο κύριες κατευθύνσεις: η κωδικοποίηση καναλιού σε επίπεδο bit και η κωδικοποίηση πακέτων, όπως αυτές περιγράφονται στα αντίστοιχα πρότυπα του οργανισμού CCSDS. Για την πρώτη περίπτωση, λόγω της ομοιότητας των κωδίκων, προέκυψε επιπρόσθετα η δυνατότητα αξιοποίησης των αποτελεσμάτων της έρευνας για χρήση στον τομέα της μαγνητικής αποθήκευσης. Η υλοποίηση των αρχιτεκτονικών κωδικοποιητών πραγματοποιήθηκε σε πλατφόρμες FPGA και SoC/MPSoC οι οποίες έχουν προταθεί ή χρησιμοποιούνται σε αεροδιαστημικές εφαρμογές, ενώ δοκιμάστηκαν σε περιβάλλον το οποίο προσομοιώνει αυτό ενός συστήματος επεξεργασίας δεδομένων εν πτήσει.

Στην εισαγωγή δίνεται αρχικά μια γενική περιγραφή της κωδικοποίησης διόρθωσης σφαλμάτων. Στη συνέχεια της Ενότητας 1 περιγράφονται τα προβλήματα για τα οποία η παρουσιαζόμενη έρευνα αναζητά και δίνει απαντήσεις. Ακολούθως τεκμηριώνεται η συνεισφορά της παρούσας διατριβής με αναφορά στις δημοσιεύσεις σε περιοδικά και συνέδρια που πραγματοποιήθηκαν στο πλαίσιο της εκπόνησής της. Τέλος, παρατίθενται οι τεχνολογίες στις οποίες πραγματοποιήθηκε η έρευνα και στις οποίες αποκτήθηκε υψηλό επίπεδο εξειδίκευσης, ως αποτέλεσμα της εκτεταμένης χρήσης τους.

Στην επόμενη ενότητα παρατίθενται σημαντικές πληροφορίες υποβάθρου. Αρχικά περιγράφονται μοντέλα καναλιών διαστημικών επικοινωνιών, με έμφαση σε αυτά που περιγράφονται στα πρότυπα του CCSDS που υλοποιούνται στην παρούσα. Έπειτα παρουσιάζεται η στοίβα πρωτοκόλλων του CCSDS. Ο CCSDS είναι ένας διεθνής οργανισμός που περιλαμβάνει τις σημαντικότερες αεροδιαστημικές υπηρεσίες του πλανήτη, και σκοπός του είναι η ανάπτυξη προτύπων για συστήματα διαστημικών επικοινωνιών, έτσι ώστε να υποστηριχθεί η διαλειτουργικότητά τους και να ελαχιστοποιηθεί το ρίσκο και το κόστος ανάπτυξης. Μεταξύ άλλων, έχει εισάγει πρότυπα στα οποία προδιαγράφεται η χρήση κωδίκων LDPC. Τα πρότυπα αυτά αφορούν κατά βάση το επίπεδο κωδικοποίησης καναλιού, το οποίο στη στοίβα πρωτοκόλλων του CCSDS είναι υπο-επίπεδο του επιπέδου σύνδεσης (data link). Στη συνέχεια αναλύονται οι γενικές απαιτήσεις και τα χαρακτηριστικά των συστημάτων επεξεργασίας δεδομένων σε αεροδιαστημικές εφαρμογές και πως αυτά διαφοροποιούνται σε σχέση με τα εμπορικά αντίστοιχά τους. Συγκεκριμένα, περιγράφονται τα χαρακτηριστικά των FPGAs, SoCs, MPSoCs και επεξεργαστών (CPUs) τα οποία είναι κατάλληλα για αεροδιαστημικές εφαρμογές, εστιάζοντας περισσότερο σε αυτά για τα οποία διατίθενται αναπτυξιακές κάρτες στο εργαστήριο (DSCAL). Τα εξαρτήματα αυτά καλούνται να λειτουργήσουν σε ιδιαίτερα επιβαρυμένο περιβάλλον από πλευράς φυσικών συνθηκών (θερμοκρασίας, κραδασμών κτλ) και ακτινοβολίας, ενώ ταυτόχρονα το περιθώριο αστοχίας είναι πολύ μικρό, λόγω του κόστους μιας διαστημικής αποστολής. Ταυτόχρονα, οι απαιτήσεις απόδοσης, κατανάλωσης ισχύος, όγκου και βάρους είναι ιδιαίτερα υψηλές. Στο τέλος της δεύτερης ενότητας περιγράφεται το πρωτόκολλο SpaceFibre, το οποίο είναι

σειριακό πρωτόκολλο για επικοινωνία συστημάτων εν πτήσει. Η σημασία του είναι ιδιαίτερη στην παρούσα διατριβή, καθώς χρησιμοποιήθηκε εκτενώς στο περιβάλλον δοκιμών όχι μόνο των κωδικοποιητών που αναπτύχθηκαν στο πλαίσιο της παρούσας, αλλά και λοιπών έργων στα οποία συμμετείχα κατά τη διάρκεια της έρευνάς μου και υποστήριξαν σχετικές δημοσιεύσεις του DSCAL. Περιγράφεται συνοπτικά το περιβάλλον το οποίο δημιουργήθηκε για την εκτέλεση δοκιμών, με την αποστολή και λήψη δεδομένων σε και από προς FPGAs, τα οποία περιλάμβαναν τις υπό δοκιμή υλοποιήσεις, καθώς και το σχετικό λογισμικό το οποίο αναπτύχθηκε.

Η επόμενη ενότητα 3 καλύπτει την κωδικοποίηση LDPC σε επίπεδο bit. Αρχικά καλύπτεται το θεωρητικό υπόβαθρο των κωδίκων QC-LDPC. Μαζί με τους τούρμπο κώδικες, οι κώδικες LDPC αποτελούν την πλέον σύγχρονη συνεισφορά της επιστήμης της θεωρίας πληροφορίας στο πρόβλημα της αξιόπιστης μετάδοσης δεδομένων, υπό την επίδραση θορύβου. Οι πρώτοι κώδικες που περιεγράφηκαν από τον Robert G. Gallager, ο οποίος τους εισήγαγε, ήταν εντελώς τυχαίοι. Η τυχαιότητα αυτή, σε συνδυασμό με μεγέθη μπλοκ της τάξης μεγέθους χιλιάδων ή δεκάδων χιλιάδων bits, καθιστά ιδιαίτερα απαιτητική τη σχεδίαση κωδικοποιητών και αποκωδικοποιητών που θα συνδυάζουν υψηλή ρυθμαπόδοση με μικρές απαιτήσεις σε υπολογιστικούς πόρους. Αποδεικνύεται ωστόσο ότι είναι εφικτή η σχεδίαση κωδίκων που προσεγγίζουν το όριο της χωρητικότητας και ταυτόχρονα ο πίνακας ισοτιμίας τους έχει δομή, η οποία διευκολύνει την υλοποίηση. Η συνηθέστερη δομή που εμφανίζεται στους σύγχρονους κώδικες είναι η ”σχεδόν κυκλική” (εφεξής quasi-cyclic ή QC), κατά την οποία ο πίνακας ισοτιμίας του κώδικα αποτελεί μια διάταξη κυκλικών υποπινάκων. Μια επιπλέον συνηθισμένη απλοποίηση των QC κωδίκων προκύπτει από μια μέθοδο κατασκευής κωδίκων κατά την οποία ένας βασικός πρωτο-γράφος Tanner αναπτύσσεται σε πλήρη γράφο. Οι πίνακες ισοτιμίας που προκύπτουν είναι QC, αλλά επιπλέον οι κυκλικοί υποπίνακες είναι ίσων διαστάσεων μηδενικοί πίνακες ή πίνακες μετάθεσης (μετατοπισμένοι μοναδιαίοι πίνακες). Πέραν αυτών των απλοποιήσεων, έχουν προταθεί και χρησιμοποιηθεί ευρέως στην πράξη κώδικες, οι πίνακες ισοτιμίας των οποίων περιλαμβάνουν επιπλέον δομές και απλοποιήσεις οι οποίες στοχεύουν συγκεκριμένα στην ακόμα ευκολότερη δυνατή υλοποίηση, όπως για παράδειγμα οι LDPC κώδικες των προτύπων IEEE 802.16.

Στη συνέχεια περιγράφονται οι QC-LDPC κώδικες που προτείνονται από τον CCSDS, τα πλεονεκτήματά τους σε σχέση με τους παλαιότερους κώδικες (concatenated RS & convolutional) καθώς και οι παράγοντες που επηρρεάζουν την απόδοσή τους. Πρόκειται για ένα σύνολο εννέα κωδίκων τύπου AR4JA και έναν κώδικα ο οποίος αναφέρεται στη σχετική βιβλιογραφία ως C2. Οι AR4JA προτάθηκαν αρχικά για χρήση σε επικοινωνίες στο βαθύ διάστημα (deep space), ωστόσο υιοθετήθηκαν στη συνέχεια και σε άλλου τύπου επικοινωνίες (π.χ. proximity-1) και εντάσσονται στην κατηγορία των QC κωδίκων που βασίζονται σε πρωτο-γράφο. Ο C2 είναι QC LDPC κώδικας, κατασκευασμένος με μια τεχνική που βασίζεται σε ευκλείδεια γεωμετρία σε πεπερασμένο διανυσματικό χώρο.

Η ενότητα συνεχίζει με την σχολαστική περιγραφή του συνόλου των LDPC κωδικοποιητών που έχει παρουσιαστεί στη βιβλιογραφία. Οι αρχιτεκτονικές υλοποίησης που έχουν προταθεί είναι γενικά βελτιστοποιημένες για ένα συγκεκριμένο υποσύνολο κωδίκων που υποστηρίζουν η κάθε μια, στοχεύοντας το αντίστοιχο πρωτόκολλο και εκμεταλλευόμενοι

τις ιδιότητες της δομής των συγκεκριμένων πινάκων ισοτιμίας κάθε φορά. Διαπιστώθηκε ότι οι μέθοδοι κωδικοποίησης που έχουν προταθεί μέχρι σήμερα κατατάσσονται επί της ουσίας σε τέσσερις μεθόδους κωδικοποίησης:

- Υπολογισμός κωδικής λέξης από τον πίνακα-γεννήτορα του κώδικα, που αποτελεί και την πιο απλή μέθοδο. Θα αναφέρεται εφεξής ως η ευθεία μέθοδος. Το μειονέκτημα της μεθόδου αυτής είναι ότι ο πίνακας-γεννήτορας είναι κατά κανόνα πυκνός πίνακας, αυξάνοντας την πολυπλοκότητα.
- Μέθοδος των Richarson-Urbanke (εφεξής R-U), η οποία είναι βέλτιστη για κώδικες των οποίος ο πίνακας ισοτιμίας εμφανίζει κάτω τριγωνική δομή, ή μπορεί να πλησιάσει αυτή τη δομή μέσω γραμμικών μετασχηματισμών. Όσο πλησιέστερα στην κάτω τριγωνική μπορεί να μετασχηματιστεί ο πίνακας ισοτιμίας, τόσο πιο αποδοτική εμφανίζεται η συγκεκριμένη μέθοδος, μειώνοντας τις διαστάσεις των πυκνών πινάκων που εμπλέκονται στις εξισώσεις υπολογισμού.
- Υβριδική μέθοδος, σύμφωνα με την οποία ένα υποσύνολο των ψηφίων ισοτιμίας υπολογίζεται με την ευθεία μέθοδο και ένα άλλο με την R-U.
- Υπολογισμός της κωδικής λέξης απευθείας από τον πίνακα ισοτιμίας του κώδικα, σε δύο βήματα, με κατάτμηση του τελευταίου σε δύο υποπίνακες, ο δεξιός εκ των οποίων υφίσταται αναστροφή και μετατρέπεται σε πυκνό πίνακα. Σε κάποιες παραλλαγές αυτής της μεθόδου, ο ανεστραμμένος πίνακας αποσυντίθεται με τη σειρά του σε ένα άνω και ένα κάτω τριγωνικό πίνακα.

Όλες οι βιβλιογραφικές αναφορές εξετάστηκαν λεπτομερώς, όσον αφορά την καταλληλότητά τους για τους κώδικες του CCSDS, με τεκμηρίωση είτε της πλήρους απουσίας δυνατότητας εφαρμογής τους, λόγω της απαίτησης για την ύπαρξη συγκεκριμένων δομών στους πίνακες ισοτιμίας, είτε των προκλήσεων που θα αντιμετώπιζαν στην περίπτωση υιοθέτησής τους για τους κώδικες αυτούς. Γενική πάντως διαπίστωση είναι σε όλες τις μεθόδους κωδικοποίησης εμφανίζεται μια πράξη πολλαπλασιασμού διανύσματος με πυκνό QC πίνακα, η οποία εισάγει προκλήσεις στην υλοποίηση υψηλού ρυθμού κωδικοποίησης, χωρίς καμία από τις προτεινόμενες αρχιτεκτονικές στο υλικό να φαίνεται ότι διαχειρίζεται ικανοποιητικά την πράξη αυτή. Προτείνεται επομένως στην παρούσα μια αρχιτεκτονική υλικού για την μέγιστη εκμετάλλευση της ενδογενούς παραλληλίας που χαρακτηρίζει ένα QC πίνακα, η οποία και περιγράφεται αναλυτικά, με την κατάλληλη μαθηματική διατύπωση, προκειμένου τα αποτελέσματα από την τεκμηρίωσή της να χρησιμοποιηθούν στις αναλύσεις που ακολουθούν.

Η κωδικοποίηση των AR4JA κωδίκων μπορεί να πραγματοποιηθεί και με τις τέσσερις μεθόδους κωδικοποίησης που απαριθμούνται ανωτέρω, προσφέροντας διαφορετικούς συνδυασμούς κατανάλωσης υπολογιστικών πόρων, ρυθμαπόδοσης και υστέρησης εξόδου. Τα ισοζύγια αυτά μάλιστα διαφοροποιούνται ανάλογα με το ρυθμό του εκάστοτε κώδικα. Στην παρούσα διατριβή, παρέχονται αναλυτικοί τύποι υπολογισμού των παραμέτρων αυτών, για κάθε μια από τις προτεινόμενες αρχιτεκτονικές, για κάθε μέθοδο ξεχωριστά. Η υβριδική μέθοδος αποδεικνύεται αναλυτικά ότι δεν πλεονεκτεί σε κανένα τομέα. Από τις

υπόλοιπες μεθόδους, η μέθοδος R-U αποδεικνύεται ότι πλεονεκτεί σε χαμηλότερους ρυθμούς, σε σχέση με τη μέθοδο του κατατετμημένου πίνακα ισοτιμίας. Η απευθείας μέθοδος δεν πλεονεκτεί όταν ο στόχος είναι η υλοποίηση σε FPGA, συμπέρασμα το οποίο διαπιστώνεται και εργαστηριακά. Στις συγκρίσεις των προτεινόμενων αρχιτεκτονικών για την κάθε μέθοδο παρατίθενται και συγκριτικά στοιχεία με αντίστοιχες υλοποιήσεις στη βιβλιογραφία, οι οποίες θα μπορούσαν δυνητικά να εφαρμοστούν απευθείας στους κώδικες AR4JA και μέσω των οποίων τεκμηριώνονται τα πλεονεκτήματα που εισάγει η προτεινόμενη βασική αρχιτεκτονική πολλαπλασιασμού διανύσματος με πυκνό QC πίνακα.

Πλην της απευθείας μεθόδου, οι αρχιτεκτονικές που παρουσιάζονται βασίζονται στην παραδοχή ότι όλα τα bits προς κωδικοποίηση είναι ταυτόχρονα διαθέσιμα στην είσοδο του κωδικοποιητή. Η αρχιτεκτονική της απευθείας μεθόδου υποστηρίζει ούτως ή άλλως σειριακή είσοδο και έξοδο ενδογενώς. Η παραδοχή αυτή έγινε προκειμένου οι συγκρίσεις να είναι επί της βάσης της πολυπλοκότητας της μεθόδου κωδικοποίησης και όχι των περιορισμών που επιβάλει η διεπαφή εισόδου και εξόδου. Παρόλα αυτά, η σχεδίαση πρακτικών κωδικοποιητών με διεπαφές εισόδου-εξόδου τύπου stream ή FIFO, αποτελεί ουσιαστική πρόκληση σχεδίασης. Παρέχονται επομένως στη συνέχεια τροποποιημένες αρχιτεκτονικές κωδικοποιητών για τις μεθόδους R-U και κατατετμημένου πίνακα ισοτιμίας, προσαρμοσμένες για σειριακή είσοδο-έξοδο. Τα βήματα εκτέλεσης των αντίστοιχων αλγορίθμων αντιστοιχίζονται σε στάδια γραμμών διοχέτευσης σε συστολικές αρχιτεκτονικές κωδικοποιητών που εξασφαλίζουν τη βέλτιστη αξιοποίηση των υπολογιστικών πόρων. Τα αποτελέσματα υλοποίησης (στην κάρτα KCU105) δίνουν εξαιρετικά μεγάλους αριθμούς ρυθμαπόδοσης, ορίζοντας το μέτρο σύγκρισης για οποιαδήποτε μελλοντική υλοποίηση. Η KCU105 φέρει το FPGA XCKU040, που αποτελεί το εμπορικό αντίστοιχο του πιστοποιημένου για διαστημικές εφαρμογές XQRKU060

Όσον αφορά τον κώδικα C2, η μοναδική μέθοδος κωδικοποίησης που μπορεί να εφαρμοστεί είναι η απευθείας. Ωστόσο η μεγαλύτερη πρόκληση όσον αφορά την υλοποίηση συγκεκριμένο κώδικα είναι η διάσταση των υποπινάκων του πίνακα ισοτιμίας (και κατά συνέπεια του πίνακα-γεννήτορα), η οποία είναι 511-bit. Οι λύσεις που έχουν προταθεί μέχρι στιγμής για το πρόβλημα αυτό είναι απλές υλοποιήσεις της απευθείας μεθόδου οι οποίες δεν εκμεταλλεύονται τους υπολογιστικούς πόρους με το βέλτιστο τρόπο, εισάγοντας αδρανείς κύκλους στη γραμμή διοχέτευσης. Η υλοποίηση της απευθείας μεθόδου που προτείνεται αντίθετα στην παρούσα διατριβή για τον C2, αξιοποιεί τη βασική αρχιτεκτονική του πολλαπλασιασμού διανύσματος με πίνακα QC, ελαχιστοποιώντας έτσι τους απαιτούμενους πόρους. Επιπλέον, μέσω μιας μονάδας προ-επεξεργασίας στην είσοδο του κωδικοποιητή, η οποία εισάγει μηδενικά σε κατάλληλα σημεία της ακολουθίας εισόδου, διαχειρίζεται όχι μόνο το πρόβλημα των διαστάσεων των υποπινάκων του πίνακα-γεννήτορα, αλλά και κάποιες επιπλέον πολυπλοκότητες που περιγράφονται στο πρότυπο, οι οποίες αφορούν την εισαγωγή μηδενικών στις ακολουθίες εισόδου και εξόδου, προκειμένου το μήκος της κωδικής λέξης να είναι δύναμη του 2 (8160 bit). Αντίστοιχα με τους AR4JA, τα αποτελέσματα υλοποίησης σε FPGA τεκμηριώνουν την συνεισφορά της προτεινόμενης αρχιτεκτονικής.

Η στατική και δυναμική ισχύς των προτεινόμενων υλοποιήσεων αρχιτεκτονικών μετρήθηκε στην KCU105, με χρήση του κατάλληλου υποσυστήματος του FPGA και της κάρτας,

το οποίο χαρακτηρίζεται από μεγάλη ακρίβεια. Η παρούσα είναι η πρώτη μελέτη που λαμβάνει υπόψιν τις παραμέτρους κατανάλωσης ενέργειας, σε σχέση με τις αντίστοιχες βιβλιογραφικές εφαρμογές.

Όλοι οι κωδικοποιητές δοκιμάστηκαν εκτενώς, τόσο με συμπεριφορική προσομοίωση του κώδικα, κατά την οποία το ζητούμενο ήταν η κάλυψη κώδικα 100%, όσο και στο υλικό. Αρχικά αναπτύχθηκε ένα μοντέλο της κωδικοποίησης στο λογισμικό GNU/Octave, σύμφωνα με το οποίο παράχθηκαν τυχαία διανύσματα δοκιμής και οι προσδοκώμενες βάσει αυτών αποκρίσεις των κυκλωμάτων. Στη συνέχεια, τα δεδομένα δοκιμής τροφοδοτήθηκαν στους κωδικοποιητές στο υλικό, είτε μέσω Η/Υ με διασύνδεση SpaceFibre, είτε απευθείας στο υλικό, με χρήση γεννήτριας ψευδοτυχαίας ακολουθίας. Το σύνολο των αποκρίσεων στην πρώτη περίπτωση ελήφθην μέσω της διασύνδεσης SpaceFibre, ενώ στη δεύτερη, οι αποκρίσεις του κυκλώματος συμπιέστηκαν με κατάλληλο κύκλωμα, του οποίου η τελική τιμή διαβάστηκε από το υλικό. Σε όλες τις περιπτώσεις τα δεδομένα συγκρίθηκαν με τα αναμενόμενα. Η σκοπιμότητα της διάκρισης των δύο δοκιμών έγκειται στη δυνατότητα δοκιμής σε πλήρη ταχύτητα, στην περίπτωση της πλήρους δοκιμής στο υλικό, σε σύγκριση με τη δοκιμή σε ένα περιβάλλον το οποίο θα προσομοιώνει, στο μέτρο του δυνατού, αυτό μιας πραγματικής αποστολής και στο οποίο τα διάφορα εξαρτήματα της αλυσίδας επεξεργασίας διασυνδέονται μέσω ζεύξεων SpaceFibre.

Κατά την μελέτη των AR4JA κωδίκων, διαπιστώθηκε ότι κώδικες LDPC βασισμένοι σε πρωτογράφους έχουν προταθεί για ένα εντελώς διαφορετικό πεδίο εφαρμογής, αυτό της μαγνητικής αποθήκευσης. Η σχετική έρευνα ωστόσο περιορίζεται στη σχεδίαση και τις ιδιότητες των κωδίκων, χωρίς να έχει προυσιαστεί κάποια αρχιτεκτονική υλοποίησης. Κάποιοι από τους κώδικες που βασίζονται σε πρωτο-γράφους που έχουν προταθεί για αυτό το πεδίο μοιράζονται αρκετά κοινά χαρακτηριστικά με τους AR4JA, με το κυριότερο να είναι οι περιορισμοί των υφιστάμενων αρχιτεκτονικών κωδικοποιητών, λόγω της απουσίας δομών που να διευκολύνουν την υλοποίηση (πέραν της QC δομής). Με βάση τις αρχιτεκτονικές υλικού για τους AR4JA κώδικες, προτείνονται κατά συνέπεια προσαρμοσμένες αρχιτεκτονικές υψηλής απόδοσης και για τους εν λόγω κώδικες. Λόγω των γενικά υψηλών ρυθμών που απαντώνται στο συγκεκριμένο πεδίο εφαρμογής (πρακτικά μεγαλύτερο από 4/5), οι υλοποιήσεις που προτείνονται βασίζονται στη μέθοδο του κατατετμημένου πίνακα ισοτιμίας.

Η Ενότητα 4 πραγματεύεται την κωδικοποίηση πακέτων, όπως αυτή περιγράφεται στο αντίστοιχο πειραματικό πρότυπο του CCSDS (CCSDS 131.5). Η διόρθωση σφαλμάτων σε επίπεδο πακέτου καλείται να αντιμετωπίσει ένα διαφορετικό πρόβλημα από το αντίστοιχο πρόβλημα σε επίπεδο bit. Υπάρχει επί του παρόντος έντονο ενδιαφέρον για την ανάπτυξη συστημάτων οπτικών επικοινωνιών ελεύθερου χώρου (χωρίς δηλαδή τη χρήση μέσου), βασισμένων σε laser, για την επίτευξη ιδιαίτερα υψηλών ταχυτήτων μεταφοράς δεδομένων. Η σχετική ομάδα εργασίας του CCSDS είναι επί του παρόντος από τις πιο δραστήριες. Σε μια τέτοια περίπτωση, η επίδραση των ατμοσφαιρικών φαινομένων με τη μορφή διακυμάνσεων του δείκτη διάθλασης και εκτεταμένης νεφοκάλυψης, όπως επίσης και φαινόμενα απώλειας συγχρονισμού του δέκτη λόγω προβλημάτων στη στόχευση των κεραιών, είναι δυνατόν να οδηγήσουν στην απώλεια σημαντικού αριθμού από bits, σε σημείο που να μην είναι εφικτή η ανάκτηση της αρχικής κωδικής λέξης, σε ένα κλασικό σχήμα

κωδικοποίησης καναλιού. Αντίστοιχα, τα επίγεια πρωτόκολλα που βασίζονται σε μηνύματα επιβεβαίωσης (ARQ), δεν είναι αποδοτικά λόγω των μεγάλων καθυστερήσεων διάδοσης. Οι κλασσικοί τρόποι αντιμετώπισης αυτών των προβλημάτων χρησιμοποιούν κώδικες καναλιού (συνήθως Reed-Solomon) με αναδιάταξη των κωδικολέξεων (interleaving), έτσι ώστε μια διάλειψη να διαμοιραστεί σε μεγαλύτερο αριθμό κωδικολέξεων και οι κώδικες καναλιού να μπορέσουν να αποδώσουν.

Μια εναλλακτική προσέγγιση αντιμετωπίζει ολόκληρες κωδικές λέξεις ενός κώδικα καναλιού σαν σύμβολα, το καθένα από τα οποία έχει μέγεθος αρκετών χιλιάδων bits και είτε λαμβάνεται επιτυχώς χωρίς σφάλματα, είτε θεωρείται σαν απωλεσθέν στο σύνολό του. Ένας κώδικας διόρθωσης σφαλμάτων επιπέδου πακέτου σε αυτή την περίπτωση εισάγει πρόσθετα σύμβολα ισοτιμίας στην μεταδιδόμενη ακολουθία, έτσι ώστε να είναι εφικτή η ανάκτηση της αρχικής ακολουθίας, υπό την επίδραση διαγραφών. Στο πρότυπο CCSDS 131.5 περιγράφεται ένα τέτοιο σχήμα, ωστόσο δεν υπάρχουν κωδικοποιητές σε υλικό για αυτό. Η μοναδική υλοποίηση που υπάρχει είναι υλοποίηση λογισμικού στη σουίτα ION της NASA, για χρήση σε δίκτυα διαλλειπτόμενης συνδεσιμότητας (Delay Tolerant Networks-DTN). Σε ένα σύστημα εν πτήση, η υλοποίηση ενός συστήματος κωδικοποίησης πακέτου σε λογισμικό εκτελούμενο σε επεξεργαστή γενικής χρήσης θα ήταν ιδιαίτερα απαιτητική σε υπολογιστικούς πόρους και ενέργεια. Κατά συνέπεια προτείνονται στην ενότητα 4 κωδικοποιητές υλικού για το εν λόγω πειραματικό πρότυπο του CCSDS.

Η ενότητα ξεκινάει με μια γενική περιγραφή της κωδικοποίησης πακέτων και συνεχίζει με μια μαθηματική περιγραφή των κωδίκων του προτύπου, η οποία διευκολύνει την περαιτέρω περιγραφή των κωδικοποιητών που πρόκειται να υλοποιηθούν. Ακολούθως περιγράφονται τα χαρακτηριστικά απόδοσης και οι παράγοντες που τα επηρρεάζουν. Στη συνέχεια παρουσιάζονται δύο αλγόριθμοι κωδικοποίησης: ο CNO (Check Node Oriented) και ο VNO (Variable Node Oriented). Ο VNO είναι στην ουσία ο αλγόριθμος που περιγράφεται στο πρότυπο, με τη διαφορά ότι η περιγραφή του στην ενότητα αυτή στοχεύει στην υλοποίησή του σε υλικό. Σύμφωνα με αυτόν, καθώς τα σύμβολα εισέρχονται στον κωδικοποιητή, ανανεώνουν τα σύμβολα ισοτιμίας που συνδέονται με αυτά, σύμφωνα με τον πίνακα ισοτιμίας συμβόλων του κώδικα. Όπως είναι προφανές, ο υπολογισμός των συμβόλων ισοτιμίας σε αυτή την περίπτωση ολοκληρώνεται σχεδόν ταυτόχρονα για όλα τα σύμβολα, αμέσως μόλις ολοκληρωθεί η επεξεργασία του τελευταίου εισερχόμενου συμβόλου. Αντίθετα, ο αλγόριθμος CNO είναι συνεισφορά της παρούσας εργασίας. Σύμφωνα με αυτόν, όλα τα εισερχόμενα σύμβολα αποθηκεύονται αρχικά σε μια μνήμη. Τα σύμβολα ισοτιμίας υπολογίζονται διαδοχικά, αθροίζοντας για το καθένα το υποσύνολο των συμβόλων πληροφορίας που συνδέονται με αυτά, σύμφωνα πάλι με τον πίνακα ισοτιμίας του κώδικα. Αντίστοιχα, στον αλγόριθμο αυτό, κάθε σύμβολο ισοτιμίας υπολογίζεται και εκπέμπεται ξεχωριστά, αφού έχει ολοκληρωθεί η είσοδος όλων των συμβόλων πληροφορίας στον κωδικοποιητή. Είναι προφανές ότι και στους δύο αλγόριθμους, απαιτείται η προσπέλαση μεγάλου όγκου δεδομένων σε ένα υποσύστημα εξωτερικής (ως προς τον κωδικοποιητή) μνήμης, καθώς δεν θα ήταν εφικτή η διαχείριση όλων των συμβόλων από πόρους σε ένα microchip. Σημαντική όμως διαφορά των δύο αλγορίθμων, με σοβαρές επιπτώσεις σε μια πραγματική υλοποίηση σε υλικό, είναι η αλληλουχία προσπελάσεων της μνήμης: στην περίπτωση του VNO, η μνήμη προσπελαύνεται εναλλάξ για ανάγνωση και

εγγραφή πολλές φορές για κάθε σύμβολο, αφού κάθε εισερχόμενο σύμβολο εκτελεί μια αλληλουχία ανάγνωσης-ενημέρωσης-εγγραφής μιας θέσης μνήμης στην οποία με το πέρας της επεξεργασίας θα έχει προκύψει ένα σύμβολο ισοτιμίας. Αντίθετα ο CNO ομαδοποιεί τις εγγραφές και τις αναγνώσεις της μνήμης: το σύνολο των εγγραφών πραγματοποιείται στην αρχή του αλγορίθμου, όταν τα εισερχόμενα σύμβολα εισέρχονται στον αποκωδικοποιητή. Αντίθετα, το σύνολο των αναγνώσεων από την μνήμη πραγματοποιείται κατά τον υπολογισμό και την έξοδο/εκπομπή των συμβόλων ισοτιμίας. Προκύπτει ότι η ομαδοποίηση αυτή των αναγνώσεων και εγγραφών ευνοεί την απόδοση των κωδικοποιητών. Το σύνολο των συμβιβασμών του κάθε αλγορίθμου και το ισοζύγιο απαιτήσεων σε πόρους και ενέργεια υπολογίζεται αναλυτικά.

Στη συνέχεια παρουσιάζονται αρχιτεκτονικές στο υλικό για τους δύο αλγόριθμους, στοχεύοντας την αναπτυξιακή κάρτα KCU105, επιτυγχάνοντας ρυθμούς κωδικοποίησης μεγαλύτερους απο 13 Gbps. Οι κωδικοποιητές χρησιμοποιούν την μνήμη DDR4 της κάρτας για τους ενδιάμεσους υπολογισμούς, στην οποία συνδέονται μέσω κατάλληλου ελεγκτή μνήμης (MIG 7-series controller). Οι δοκιμές επαλήθευσης των κωδικοποιητών στο υλικό έγιναν, όπως και στην περίπτωση των κωδίκων καναλιού της προηγούμενης ενότητας, με συνδυασμό γεννήτριας ψευδοτυχαίας ακολουθίας δεδομένων και συμπίεσης αποκρίσεων, αλλά και αποστολής και λήψης δεδομένων από Η/Υ μέσω διασύνδεσης SpaceFibre. Αντίστοιχα πάλι με τους κώδικες καναλιού, αναπτύχθηκε κωδικοποιητής σε λογισμικό, βασισμένος στη δημοσιοποιημένη υλοποίηση ανοιχτού κώδικα (OpenFec project), η οποία υιοθετείται από τη σουίτα ION και με βάση τον οποίο υπολογίστηκαν οι αναμενόμενες αποκρίσεις των κωδικοποιητών στο υλικό. Το περιβάλλον αυτό αξιοποιήθηκε και για τις μετρήσεις ισχύος που πραγματοποιήθηκαν, με χρήση πάλι του κατάλληλου υποσυστήματος του FPGA και της κάρτας και οι οποίες, μαζί με την αξιολόγησή τους στην παρούσα, αποτελούν μια ακόμα συνεισφορά της παρούσας εργασίας, καθώς είναι η πρώτη φορά που εξετάζεται το ζήτημα της κατανάλωσης ενέργειας για τις υλοποιήσεις κωδικοποίησης επιπέδου πακέτου.

Οι κωδικοποιητές πακέτων που παρουσιάζονται, εκτός από το να απελευθερώνουν υπολογιστικούς πόρους από την κεντρική μονάδα επεξεργασίας ενός συστήματος επεξεργασίας δεδομένων εν πτήσει, είναι σε θέση να υποστηρίξουν υψηλούς ρυθμούς κωδικοποίησης, ικανούς να για τις απαιτήσεις των νέων προτύπων οπτικών επικοινωνιών, με πολύ αποτελεσματικότερο τρόπο από έναν επεξεργαστή γενικής χρήσης. Για την τεκμηρίωση της πρότασης αυτής, οι προτεινόμενοι κωδικοποιητές υλικού συγκρίνονται με υλοποιήσεις λογισμικού σε επεξεργαστές γενικής χρήσης που έχουν προταθεί ή χρησιμοποιηθεί σε αεροδιαστημικές εφαρμογές. Το λογισμικό του κωδικοποιητή που αναφέρθηκε ανωτέρω για τον υπολογισμό των αναμενόμενων αποκρίσεων εκτελέστηκε διαδοχικά στους επεξεργαστές LEON3, LEON5 και NOEL-V (RISC5). Για τη δοκιμή χρησιμοποιήθηκαν μοντέλα των επεξεργαστών αυτών σε FPGA (soft processors), συνδεδεμένα στον ίδιο ελεγκτή μνήμης (MIG7-series) στην πλακέτα KCU105 και η σύγκριση έγινε στη βάση των απαιτούμενων κύκλων ρολογιού για τον υπολογισμό μιας κωδικολέξης, αντί για απόλυτο ρυθμό κωδικοποίησης, προκειμένου οι συγκρίσεις να είναι δίκαιες. Οι μετρήσεις έδειξαν μια βελτίωση της ταχύτητας κατά έναν παράγοντα μεγαλύτερο από 80 φορές, με χρήση των προτεινόμενων κωδικοποιητών. Μια ακόμα σύγκριση πραγματοποιήθηκε με το

ενσωματωμένο σύστημα επεξεργασίας ARM Cortex A9 της σειράς ZynQ-7000 και συγκεκριμένα του MPSoC της κάρτας ZC706, καθώς η τεχνολογία ZynQ-7000 έχει πρόσφατα προταθεί για χρήση σε αεροδιαστημικές εφαρμογές. Σε αυτή τη δοκιμή, χρησιμοποιήθηκε ο ελεγκτής μνήμης του υποσυστήματος ARM για τη διασύνδεση των κωδικοποιητών με τη μνήμη της κάρτας. Η επιτάχυνση που επιδείχθηκε ήταν από 4,7 έως 9 φορές μεγαλύτερη απόδοση των κωδικοποιητών υλικού.

Στην τελευταία ενότητα ολοκληρώνεται η διατριβή και περιγράφονται οι μελλοντικοί ερευνητικοί ορίζοντες που ανοίγει η παρούσα εργασία.

To my beloved wife, Valentina

ACKNOWLEDGEMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor Professor Antonis Paschalis, for all the guidance and support that he provided to me through all these years: from our first academic encounter during my MSc studies, up to the present moment and for the further possibilities that he opened to me for future collaboration and work on the research areas outlined in the last Section of this thesis.

This journey would have not been possible without the support of Prof. Kranitis, who was always holding the map and showing the research direction to pursue. Professor Kranitis has always been by my side, supporting me when sailing in unknown and unmapped waters. I would like also to express my deepest appreciation to Prof. Gizopoulos, for all his colossal academic achievements and the work that he has done for the DSCAL. In addition, I owe special thanks to Elias Machairas and Panagiotis Chatziantoniou, both members of the DSCAL team who supported this work, and whose impressive research achievements have always been a source of inspiration and admiration.

Last but not least, I owe special thanks to Dr. Antonis Tsigkanos, for all the assistance he has provided in my work, through all these years in DSCAL. Important parts of the code basis of the work presented in this thesis have been kindly provided by him, saving me a significant amount of time and effort. Above all, his assistance and support in troubleshooting the numerous bugs and dealing with the hundreds of dilemmas I was confronted with, deserve, as a minimum acknowledgement of his support, my highest respect and my deepest gratitude.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Introduction to Forward Error Correction schemes

Forward Error Correction (FEC) codes are used extensively in almost every communication and data processing system, in order to increase the reliability of transmission and storage of data. This is especially important in space communications scenarios, due to the challenging environmental conditions and the extremely stringent power requirements of deep-space links, or conversely, the high data rates and low latency necessary in near-earth satellite communication scenarios. Channel coding is the process that implements the FEC capability, by transforming the information that is going to be transmitted over a channel, in a suitable form that can lead to this reliability increase. Generally, in channel coding, redundant information is added to the transmitted or stored information, so that its recovery at the receiver is possible, in the presence of errors which are caused by noise, interference, disruptions of the transmission path between the transmitter and the receiver, failure of the storage medium, or any other phenomena that could potentially result in loss of information. Apparently, an efficient channel coding scheme is one that combines the maximum information recovery with the minimum redundancy in the transmitted/stored information. In his seminal work in [54], Shannon defined the capacity of a channel as the upper bound of the rate at which it can convey information, and it determined by the channel's physical characteristics and the mathematical model which describes it. Reliable communication can only be established if the rate at which the information is transmitted over the channel is lower than its capacity. In other words, he proved that channel codes exist, which can result in transmission rates that are arbitrarily close to the channel's capacity. What, however, he did not provide in his work, is the definition of the specific codes themselves. The holly grail of information theory has been, since then, the discovery of channel codes that can perform as close to the capacity limit of the channel as possible. At the same time, an efficient channel coding scheme implementation in a realistic scenario has to balance contradicting requirements and offer a variety of trade-offs in terms of error correcting efficiency, encoding/decoding complexity, throughput, hardware resources utilization and power consumption.

The traditional approach which has been widely adopted since the dawn of digital communications implements channel coding at the bit-level. Well known codes in this area include Reed-Solomon (RS) [108] and Turbo [23] codes. Another highly advantageous class of channel codes are the Low-Density Parity-Check (LDPC) codes, which are linear block codes, characterized by large block lengths and sparse parity-check matrices. Introduced by R.G. Gallager in 1960 [64], LDPC codes had in following years generally succumbed to oblivion, due to the current era's technology limitations, which could not allow their implementation at a reasonable cost. However, advances in VLSI technology, together with the application of efficient code design techniques that facilitate encoder/decoder implementation have annihilated those barriers. Among the entire range of modern error correcting codes (ECC), they are currently the most promising approach towards the capacity limit described by Shannon [54]. This has established them as the optimal choice for FEC in

modern applications.

The error correcting capability of conventional bit-level channel coding, however, is limited in high speed and deep fading scenarios, such as those encountered in modern earth-to-satellite and satellite-to-satellite laser links. Moreover, these environments are characterized by high latency and the fading effect of the communication channel is so deep that bit-level channel codes cannot provide the required reliability, since even a single scintillation effect can span a high volume of the transmitted information sequence. Error correction in this case takes place at a higher level of the communications protocol stack than bit-level channel coding (which is typically a function of the data link layer in OSI protocol stack). Contrary to the bit-level channel codes, where most of the encoding and decoding takes place in specialised hardware, in this case traditionally, software-based approaches had been followed, since the volume of the data to be processed is enormous. This, in turn, introduces additional challenges in terms of resources, latency and power.

The vast volume of data transmitted in a a aerospace data-link is downstream data (from the platform to the terrestrial ground station). In this scenario, an encoder is a hardware or software element of the on-board data processing chain that generates the suitable FEC sequence to be transmitted, and a decoder is the corresponding element on the receiver's side (ground station). At the same time, the on-board resources are generally scarce, contrary to the ground station's, where resource and power abundance can be presumed. Since the focus of this thesis is on aerospace applications, the first challenge is the efficient implementation of encoders for FEC schemes. That is does not mean in any case that the design of efficient decoders is trivial: deep space and near-earth communication is always bidirectional and the upstream data normally refers to mission-critical control and navigation information, that also needs a high level of protection. Moreover, space-to-space communication (for example between satellites or between a rover and a satellite) requires that the spaceborne platform is able to carry out both functions (encoding & decoding).

1.2 Problem description and motivation

LDPC codes are linear block codes, characterized by large block lengths and sparse parity-check matrices. The initial Gallager codes [64] were random and although they exhibited excellent error-correcting capabilities, hardware implementation was challenging. In order to reduce implementation complexity and encoding/decoding speed, additional structure has been designed into the parity check matrices of all practical LDPC codes in modern applications, so that they consist of an array of juxtaposed cyclic sub matrices, named the circulants, which can be efficiently implemented. These structured codes are collectively referred to as Quasi-Cyclic (QC) LDPC codes.

A special class of QC-LDPC codes, the protograph-based QC codes have recently received considerable research interest in many modern standards [57]. The protograph codes proposed in [60] exhibit outstanding performance over the partial response (PR) channel, used to model magnetic recording (MR) media systems. The work in [40] shows

the application of LDPC codes to physical layer network coding (PNC) and describes the research on LDPC codes for PNC as an emerging research trend. A class of protograph based LDPC codes for use on PNC are also proposed in [40]. Finally, a novel family of root-protograph QC-LDPC codes have recently been proposed in [59] for modern point-to-point and multirelay wireless communication applications, modelled according to the block (or slow) fading channel. Most importantly, the Consultative Committee for Space Data Systems (CCSDS) has standardized in [32] a number of protograph-based QC-LDPC code families for space communication protocols, as alternatives to concatenated convolutional and Reed-Solomon codes, over which they offer substantially higher error-correcting performance. In this paper, the focus is on these specific codes.

A multitude of encoder architectures for QC-LDPC codes has been proposed in the literature. Most of these architectures, however, focus on a specific standard or class of standards, leveraging the specific properties of the particular code. The result is that although they exhibit outstanding performance characteristics for the specific code family, they are either altogether non-applicable to CCSDS codes and codes that are similard to CCSDS, or their adoption to them comes with a significant performance penalty. Most of these encoder architectures require a specific structure in the parity check matrix of the code, which is not present in CCSDS codes. Some examples of such cases can be found in [127, 97, 133, 99].

A new approach therefore is required, that can lead to efficient encoding of QC-LDPC codes, without, however, duplication of the existing knowledge in the field: a thorough investigation and analytical comparison of the existing encoding methods, a grouping of the so far proposed encoder architectures into these methods and a comparison of the results of their application to protograph-based QC-LDPC codes are fundamental research requirements and solid justifications of the contributions of this work.

Packet-level (PL) erasure coding is a new approach for mitigation of burst errors in high speed optical communications. CCSDS has introduced an experimental specification for PL coding in [33]. Since their introduction in [33], however, the proposed codes have not matured into a CCSDS recommended (”blue”) standard yet, nor are there any implementation attempts demonstrated in the literature. The current work is the first approach to examine packet-level encoding algorithms and propose, implement and test hardware encoder architectures for these algorithms. With the current research, I support that they can be placed among the options for modern high speed communications.

1.3 Contributions & publications

The contributions provided by the work presented in this thesis can be grouped into two main research areas: bit-level encoding for QC-LDPC codes and packet-level coding for the erasure channel and system unit testing using SpaceFibre equipment as an auxiliary area. The individual components of each contribution area are enumerated and described in the current Section.

1.3.1 Bit-level QC-LDPC encoding

1. All the LDPC encoding methods which have been proposed so far are summarised in Chapter 3 and their applicability to the protograph-based CCSDS QC-LDPC codes of the CCSDS standard [32] and the codes for MR media is analytically examined.

2. The entirety of the published works in the research field of LDPC encoding implementations is thoroughly reviewed, grouped according to the encoding method that each reference implements and their performance for the codes of the previous item is questioned.

3. A novel architecture for the multiplication of a dense QC matrix with a bit vector, which is a fundamental operation of QC-LDPC encoding, is proposed. The architecture leverages the inherent parallelism of the QC structure by concurrently processing multiple bits, according to an optimized scheduling.

4. Based on this architecture, efficient encoders for CCSDS codes are proposed, according to all the applicable LDPC encoding methods, which are analytically described and compared in terms of resource utilization efficiency for the CCSDS and MR QC-LDPC codes.

5. In the special case of the specific code defined in the CCSDS standard for Near-Earth communications, a preprocessing algorithm is also introduced, which efficiently handles the challenges arising from the generator's matrix circulant size (511 bits).

6. State-of-the-art encoding throughput performance is demonstrated on a the commercial counterpart of Xilinx space-grade Kintex UltraScale FPGA technology, achieving a significant speed-up compared with previous approaches, while at the same time keeping resource utilization low.

7. Extensive testing in a realistic SpaceFibre-enabled environment has been executed.

8. It is the first work to introduce accurate and detailed power measurements and comparisons for all the encoder implementations presented.

9. The hereto described analysis and results have also been applied to the special case of the protograph-based codes which have been proposed for MR media. These cores bear significant resemblance to the corresponding codes of the CCSDS standard [32]. This is the first time that practical hardware QC-LDPC encoder implementations are proposed for MR applications.

The above topics have been published in [121], as an early stage preliminary work on encoders for CCSDS and in [123] as a mature work. Further progress, however, has been made since their publication, which is described in the corresponding Chapter 3. This additional work is related to the design, implementation and testing of architectures with AXI4-Stream interfaces and testing with SpaceFibre equipment. Some of these results

have been recently demonstrated in [105]. The work regarding MR applications has been published in [124].

1.3.2 Packet-level coding over erasure channels

1. The work presented in Chapter 4 is the first hardware implementation of PL codes in the literature and in the market. Implementations of packet-level encoding and decoding so far exist only in software, running on a general-purpose CPU.

2. A novel encoding algorithm (Check Node Oriented, or CNO) is introduced in Section 4.4, which is shown to offer diversified trade-offs, when it is compared to the conventional encoding algorithm (Variable-Node Oriented, or VNO) that is proposed in [33] and the related references.

3. The two encoding algorithms are analytically described and compared and the different trade-offs are described and explained.

4. Power measurements of the complete FPGA design are introduced, enabling thus the discussion about the inclusion of packet-level codes in power-constrained on-board applications.

5. Hardware implementations are proposed for the two algorithms and their trade-offs are described, including power consumption.

6. Hardware implementations are compared with software implementations on CPUs which have been used in aerospace applications, showing an indisputable acceleration advantage, in favour of the proposed hardware implementations.

The above topics have been published in [122].

1.3.3 Unit-level testing and SpaceFibre integration

1. Gaisler and STAR-Dundee SpaceFibre IP cores have been ported to the available FPGA cards of DSCAL and reference designs that ca be used for testing have been built.

2. Custom software has been developed around the STAR-System API, for test automation.

3. Software for test automation around the ZynQ-7000 series SoC has been built, on the Xilinx Vitis Unified Software Platform.

These contributions have supported the publications in [126] and [37].

1.4 Equipment & Technologies

The theoretical results and analytical estimations described throughout this thesis are in all cases backed by active development and implementations on FPGA and MPSoC hardware, which also include validation and verification procedures: the proposed architectures are implemented as IP cores on the targeted platforms and their responses are compared against a bit-accurate software model, written in C or GNU/Octave. These developmental and testing processes accounted for a significant part of the total research effort and required the solid understanding and proficient use of the corresponding tools:

- Xilinx Vivado and (various versions) for implementation. Heavy use of AXI infrastructure IP and most of the rest of AXI-related IP cores (for example AXI performance monitor, AXI JTAG master and AXI DMA cores) was also mandated for the testing frameworks adopted.
- Xilinx Vitis Unified Software Platform (various versions) for SoC and MPSoC system programming was used, mostly for testing purposes and for the perfomance comparisons of PL erasure codes' implementations.
- Mentor Graphics Modelsim simulator was the tool of choice for behavioural simulation, with code coverage analysis enabled in an effort to comply with ESA IP core specifications [12].
- Synopsis Synplify Premier was used for synthesis, when the maximum targeted clock rate was being sought after
- Vunit framework [19] was extensively used for most testbences and especially those involving AXI4-enabled cores.
- GNU/Octave language, which is the open-source equivalent of Mathworks Matlab was used for the golden model development.

The DSCAL equipment available to support the research includes all the above listed software and a variety of development boards, including the XUPv5, Zedboard, ZC706 and ZCU102 boards. These boards are described in more detail in Section 2.3.

1.5 Thesis outline

Chapter 2 introduces some necessary pieces of information about the thematic region of the thesis. First, an overview of space communication channels and protocols is provided, with an emphasis on the CCSDS standards and the channel models which have a direct or indirect relationship with the subjects of this thesis: TM bit-level channel coding, mostly over the AWGN channel and packet-level coding over the erasure channel. Then on-board data processing requirements are briefly analysed. The application of FPGAs

and SoCs/MPSoCs in aerospace environments is described and the development and evaluation boards which are available in DSCAL and include some of the FPGAs and/or SoC/MPSoCs which have been proposed or used for space applications. This Section is followed by a brief introduction into the CPUs that are used in space, with a brief analysis of the emerging trends. Finally, SpaceWire and SpaceFibre protocols are described and the auxiliary contributions of the current work in the field of SpaceFibre unit-level testing is detailed.

The subsequent Chapter 3 describes all aspects of QC-LDPC encoding with which this thesis deals. First, the theoretical background of QC-LDPC codes is provided, to the level of detail that is necessary for the subsequent analysis. Then, the specific codes of the CCSDS standards are described in a way that is meaningful from an encoder implementation perspective. This analysis is quite different from the corresponding definitions which are provided in the CCSDS standards and most importantly, sets a concise terminology and notation that is followed across the Chapter. A detailed description follows, of all the methods which have been proposed so far for LDPC encoding and all the encoding architectures in the known and accessible literature, grouped according to the encoding method that they implement. For each encoding method and architecture, an applicability analysis is provided, regarding their effectiveness for the protograph-based codes of the CCSDS standard and their limitations are highlighted. The basic architecture for the multiplication of a bit-vector with a dense QC matrix is afterwards described, which forms the basis for the hardware encoding architectures proposed in the subsequent Section 3.5, for each one of the possible encoding methods. Analytical calculation of the required resources is provided in each case and comparisons are made against the proposed architectures presented in Section 3.3 which are applicable to the QC-LDPC codes of the CCSDS standard. These architectures are slightly modified in the next Section (3.6), so that practical implementations featuring industry-standard AXI4-Stream interfaces can be designed, in a way that balances the execution steps along a high-performance pipeline. The implementation results targeting space-equivalent FPGA fabric are provided and compared against the (limited) implementations of the codes of the CCSD standard in the literature and in the market. The testing framework is then provided in Section 3.7. In 3.8, which is the last Section of Chapter 3, the protograph-based QC-LDPC codes proposed for MR media systems are briefly described. These codes bear significant resemblance to the corresponding CCSDS codes and the adaptation of the architectures introduced in Section 3.5 is possible. Their implementation results and testing methodology are also described.

Chapter 4 deals with packet-level erasure coding. After a short introductory Section (4.1) which mostly focuses on the problems that PL coding is trying to solve, the codes of the CCSDS standard in [33] are presented and a concise notation is established. Then, in Section 4.4, two encoding algorithms are described, one of which (Check Node Oriented-CNO) is a new contribution of this thesis. The different trade-offs of the two encoding algorithms are identified and compared, as well, in that Section. Hardware architectures for the two encoding algorithms are introduced in Section 4.5, for which the different implementation scenarios favoring the one over the other are explained. Section 4.6 describes the implementation results and performance and power characteristics of the implemen-

tation on the KCU105 development board. Comparisons to software implementations of the PL encoder process of the CCSDS standard on CPUs used in aerospace applications are provided in Section 4.7.

Finally, Chapter 5 concludes the thesis and describes the future research directions that have been inspired by the work described herein.

2. BACKGROUND

2.1 Space communication channels & protocols

A space link is a communications link between a spacecraft and its associated ground system or between two spacecrafts. A space communications protocol is a communications protocol designed to be used over a space link, or in a network that contains one or multiple space links [30]. Space data data links are inherently different from their terrestrial counterparts, since the transmission channel is dominated by longer transmission delays, weather phenomena low transmission power, high attenuation and high noise.

A channel model is a mathematical representation of the effect of a communication channel through which wireless signals are propagated, on these signals; it is modeled as the impulse response of the channel in the frequency or time domain [62]. The selection of the correct channel model for each space communication link is important for the correct design and performance characterization of the FEC code that is used. The work in [102] provides a starting point to the study of channel models used for near-earth, deep-space and optical communications. In the simplest case, the Additive White Gaussian Noise (AWGN) channel is used, especially for low-rate deep space communications. However, as data rates increase, transmission volumes (i.e. block lengths) become higher and atmospheric phenomena need to be modeled in near-earth scenarios, its limitations becomes prominent. Fading effects have to be taken into account and more complex fading channel models need to be considered, like the Nakagami, Rayleigh and Rician channels. Moreover, multipath transmission and solar or cosmic ray interference cause inter-symbol interference (ISI) between the transmission symbols, so that the channel can no longer be considered as memoryless. The Binary Erasure Channel (BEC) is another useful channel model that has also been adopted in the development of space communication protocols. Since all these channel models have been amply studied and documented and the relevant information is covered by almost every relevant academic textbook, it is not the purpose of this thesis to elaborate on the details of their mathematical description and properties. Intermittent connectivity is also common in space communications: whether as a result of atmospheric perturbations due to optical (refractive index) turbulence, meteorological phenomena (e.g. cloud coverage), or loss of line of sight, as a result of orbital movement. In these cases, the Sdelay tolerant networking (DTN) architecture [36] is used to describe and solve the problem of communication. References [28] and [42] provide further insight and the most recent advances in the application of the DTN architecture for space applications.

The Consultative Committee for Space Data Systems (CCSDS) is a multi-national forum for the development of communications & data systems standards for spaceflight. An overview of the space communication protocols standardized by the Committee is provided in [30]. The complete analysis or enumeration of the protocols implemented by the committee is neither feasible, nor meaningful. Instead, attention is limited to the protocols involved in the work described in this thesis, which are displayed in Fig. 2.1 with their

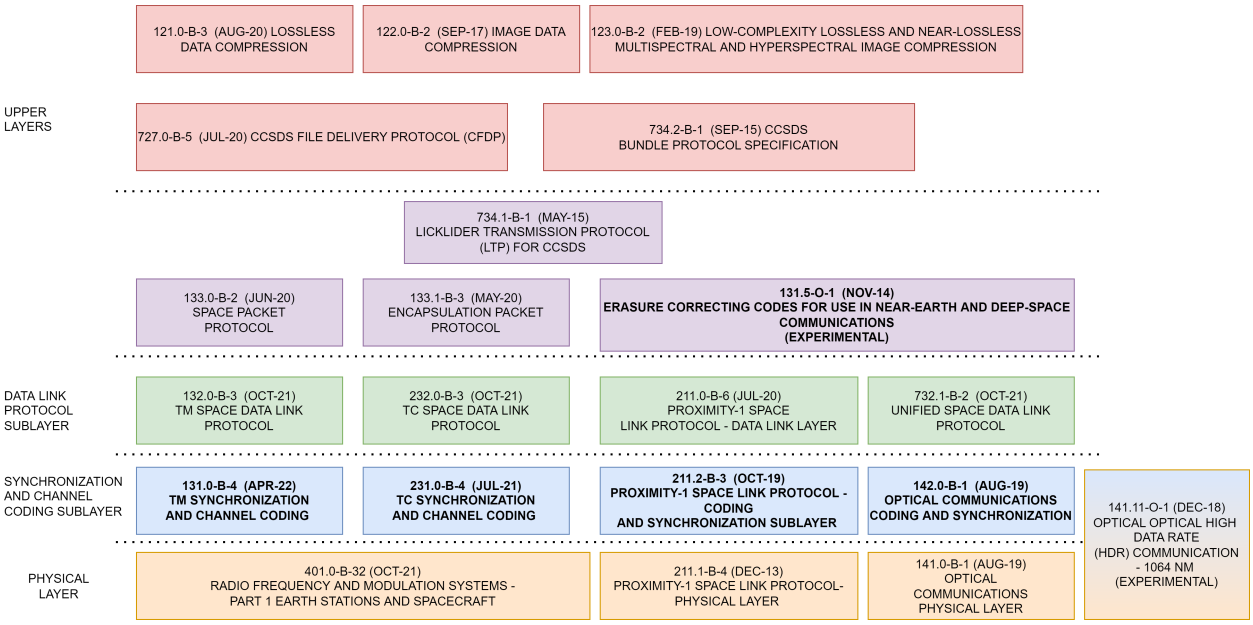

Figure 2.1: Overview of space communication protocols.

correspondence to the OSI protocol reference model (where possible).

Protocols for multispectral and hyperspectral image compression dominate the the top layers, mostly because parts of this work supported the research activities of DSCAL in their respective areas. More specifically, the initial 123.0-B1 standard provided a prediction-based adaptive algorithm that achieves a unique combination of low complexity and high efficiency: a prediction algorithm estimates the image sample value based on the values of nearby samples in a small 3-D neighborhood and an encoder algorithm losslessly encodes the mapped prediction residuals using a sample adaptive encoder or the block-adaptive encoder specified in the CCSDS 121.0-B-3 standard for lossless data compression, as an alternative option. A high throughput parallel implementation of the standard has been introduced in [126], which targets a low cost COTS SoC FPGA device. In February 2019, the standard was updated to version 2 (CCSDS 123.0-B-2) and support was added for near-lossless compression capability, according to which, absolute and/or relative error limits to lossy compression are defined. A high-performance architecture and implementation of the standard has recently been introduced by DSCAL in [37] and it has been tested and validated on space-grade technology. Of the other two protocols listed at the higher layers of Fig. 2.1, CFDP provides file transfer capabilities and the corresponding software user interface, while the bundle protocol implements the DTN network architecture in the context of space communications.

Between the Data Link and the Network Layers of the ISO stack, the Licklider Transmission Protocol (LTP) can provide reliable and unreliable communications over a single data link hop. It is optimised to support CFDP and to interoperate with the bundle protocol. The space and encapsulation packet protocols are part of the CCSDS encapsulation service,

which is attached to the data link layer and serves as a convergence sublayer between this layer and the higher layer protocols, some of which span multiple OSI layers.

The data-link layer is split into two sub-layers: the data-link protocol and the synchronization and channel coding sub-layers. The distinction refers to the different protocol functions, rather than the special needs of space communications. In these layers, four protocol stacks are defined for telemetry, telecommand Proximity-1 and free-space optical links. Proximity-1 links are short range bi-directional links between space equipment, for example between fixed probes or rovers and orbiting relays. Recently, free-space optical laser-based communications have been gaining momentum for space applications, since they offer the potential of an order of magnitude increase in the offered data-rate, compared to their RF counterparts, while requiring less space, power and mass for their electronics. Their advent has called for different protocols. This area of optical space communications is currently evolving rapidly and new functionalities are being constantly added to the corresponding protocols. Currently, three different kinds of space optical communications are considered [56]:

- High Photon Efficiency (HPE) links are photon-starved links, typically expected in deep-space mission scenarios, or in small satellites (cubesats), that cannot support the mass and power of a high-performance laser system.
- High Data Rate (HDR), mainly considered in near-earth scenarios, where throughput performance is the most important criterion.
- On-Off Keying (O3K), where high data rates are required, but the focus is on but implementation simplicity and low cost. This is the currently the most active area of the CCSDS Optical Communications Working Group.

Obviously, many function are common between the four space data-link protocol stacks and the corresponding protocols share common descriptions of them. The focus of this thesis is on the synchronisation and channel coding sub-layer, where FEC codes are implemented. Other than error correction, common functions of this sub-layer are:

- Receiver's and codeword synchronisation, which is implemented typically by appending a synchronisation sequence in the transmitted data stream, so that the codewords are delineated,
- Ensuring enough transition density of the generated output stream, so that the receiver's clock can synchronise. This is typically implemented by pseudo-randomising the generated bit stream. In the case of optical communications, this is not required.

The bundle, encapsulation packet and LTP protocols are important interfaces for the erasure correcting codes of the 2.2 CCSDS 131.5-O-1 experimental specification, with which an important part of the current thesis is involved. In particular, the standard [33] proposes the protocol layer structure of 2.2 for DTN enable space missions. Erasure coding is proposed as a shim layer right before the encapsulation layer (or service).

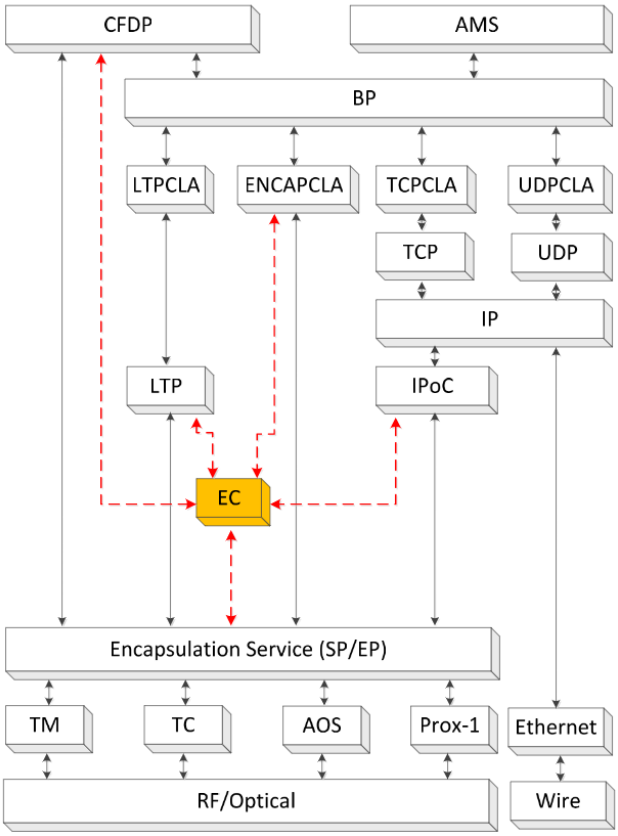

Figure 2.2: CCSDS Protocol Stack with Erasure Coding Functions. Source: [33]

Together with the synchronisation and channel coding sub-layer, this work implements FEC encoding of the TM synchronisation and channel coding standard [32] and the Erasure Coding functionality described in the experimental CCSDS 131.5-O-1 standard [33].

2.2 On-board data processing

The stringent requirements of aerospace applications in terms of reliability and power call for a different approach, when considering on-board data processing equipment. Commercial devices, targeting hugely larger market shares and lower time to market, cannot obviously meet these requirements. Processors in space are required to withstand harsh environmental conditions, mainly due to radiation effects. In addition, the risk margin of the disruption of the mission needs to be significantly lower. To meet these ends, the space industry has established the notion of Technology Readiness Level, and the relevant guidelines for ECSS are provided in [8].

The degradation of the reliability of electronic systems manifests itself in two forms of errors in their operation. The most severe form refers to hard errors. These can happen as a consequence of the gradual or sudden degradation of the system caused by the accu-

mulation or a surge of total ionizing dose (TID) or atomic displacement (Total Non Ionizing Dose-TNID or Displacement Damage-DD) [55]. Another kind of effects are transient phenomena which lead to so-called ”soft errors” in the component’s operation. When the error in the system is caused by the passage of a single particle, the event can be categorised as Single Event Effect (SEE). SEEs can lead to soft errors, for example Single Event Upsets [66], or hard, as is the case with Single Event Upsets (SEU) or Burnouts (SEB).

Depending on the type of the effect, various mitigation techniques are applied at various levels: from the physical layer, which refers to the semiconductor fabrication process up to the system level design. Devices employing these techniques are referred to as radiation tolerant, or radiation hardened devices. Radiation hardening aims to minimise the probability of radiation effect’s occurrence in the first place, mostly by measures on the physical layer and their cost of radiation hardened can be significantly higher than that of their commercial counterparts. Radiation tolerance, on the other hand, assumes that radiation effects are bound to occur and aims at reducing the impact of radiation effects on the system’s operation. ECC in the memories and buses is the fundamental radiation tolerance technique. A summary of mitigation techniques at various levels of design can be found in [79] and the references therein.

The topic of mitigation techniques is widely covered in the literature. Consequently, we limit our brief description to the following techniques, which are more relevant to this work: Triple Modular Redundancy (TMR) and memory scrubbing. In a basic TMR sheme, three redundant circuits perform the same task on the same data. A majority vote process at the system’s output can mask a failure in one of the circuits. Obviously, the cost of this approach is that it requires triple resources. Memory scrubbing, as the name implies, is a method to increase the integrity of data stored in a memory system. It requires that a method of ECC has been applied to the data written in the memory. Its contents are periodically retrieved, any errors are detected and corrected with the ECC and the result is written back to the memory. The frequency of the memory scans needs to be balanced, so that single errors are not accumulated and the ECC fails.

In the near future space computing technology is expected to converge more rapidly with the equivalent terrestrial practices [63], so that, depending on the mission goals, the required balance between performance, resiliency and cost is met: as smaller payloads with a limited lifespan are becoming more popular, the requirements for space-qualified parts can be relaxed. The most extreme example of this scenario is the case with cubesats and nanosattelites in the ”new space” emerging trend [92], the lifespan of which can be as small as a few days [100]. In this aspect, for non-mission critical functions and time-specific payloads, even the use of COTS equipment can be acceptable.

In a typical aerospace on-board processing application scenario, an airborne or space-borne platform is the main producer of information: a sensor or an instrument generates the information to be processed by the terrestrial base station, notwithstanding the emerging trend of executing more complex tasks on-board. We refer to these data as the telemetry. It is of course also true that the direction of the transmission is always bidirectional, with the upstream transmission related mainly to configuration data and control of the space platform and the on-board instruments. This is known as the telecommand

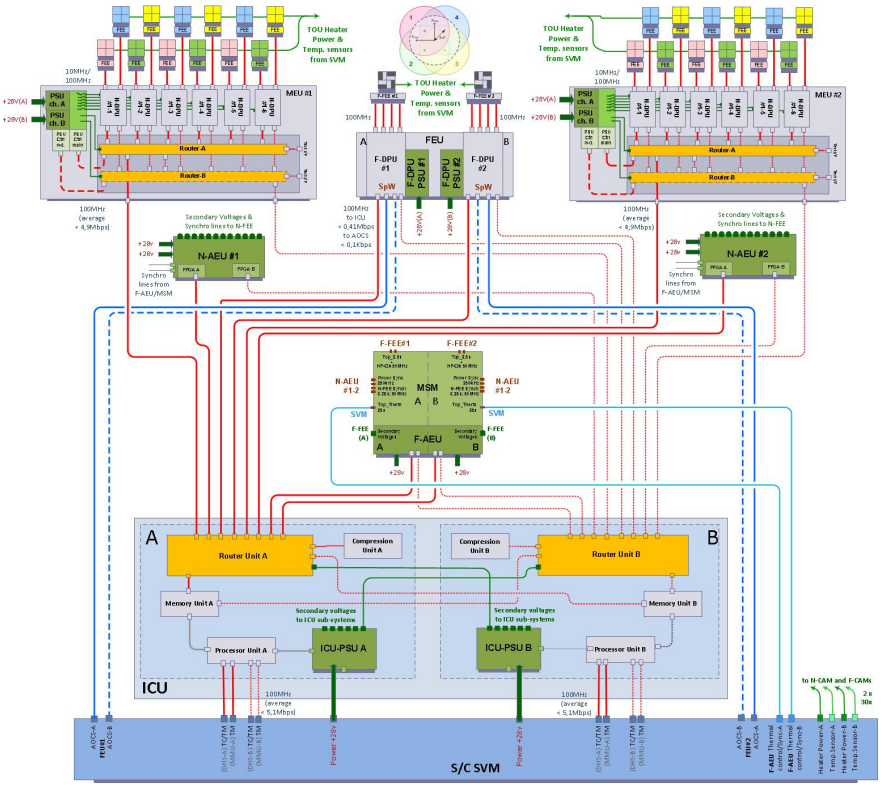

Figure 2.3: PLATO paylolad architecture overview. Source: [107]

transmission.

Typical on-board data handling systems are built around a central processor (OBC-On Board Data Computer), which is mostly responsible for telecommand functions and the coordination of the rest of the platform subsystems: telecommunication, telemetry, mass memory subsystems, sensors, instruments, and payload processors. All these subsystems communicate through highly reliable communication links, typically MIL-STD-1553, or spacewire and spacefibre, which are described separately in Section 2.5. The Space Avionics Open Interface Architecture (SAVOIR) initiative is a move towards the standardization of space avionics and, among other products, it proposes a reference functional architecture reference model.

The PLATO (PLAnetary Transits and Oscillations of stars) payload [107] is a much more complex system which comprises 27 CPU cores, over 30 FPGAs and high speed spacefibre interconnections. A central Instrument Control Unit (ICU), based on a CPU and a radiation hardened FPGA controls the entire instrument and handles data compression. However, multiple peripheral data processing units, built around CPUs and FPGA accelerators are responsible for the vast volume of on-board data processing. An overview of the system is depicted in Fig. 2.3.

On-board data processing, therefore, includes multiple CPUs, which, in turn, rely on FPGA-based accelerators to deliver their time-constrained high-volume mission, and data flows between the various payload components through high-speed serial links, like PCIe and

spacefibre. In the context of this work, we focus on hardware accelerators for the FEC functions of the data processing chain, which are more efficiently implemented as IP cores on the programmable logic part (FPGA). The data flow between the various processing cores within the FPGA is managed by on on-chip protocols, like the Advanced eXtensible (AXI) protocol, which is an open-standard on-chop interconnect specification developed by ARM, as part of its Advanced Microcontroller Bus Architecture (AMBA) specification. Figure 2.4 provides a simplified view of the system architecture adopted throughout this thesis and further detailed later in this Section: data to be encoded are sent to the FPGA through a spacefibre serial interface. A spacefibre bridge converts the incoming stream to an AXI4-Stream bus for on-chip communication and sends them to the encoding core, which is denoted in the image as the accelerator. Processed (encoded) data follow the reverse data flow and sent to the spacefibre network. The operation of the accelerator is normally controlled and monitored through control and status registers, which are accessible through:

- The spacefibre interface, using the Remote Memory Access Protocol (RMAP)[3]. This option allows for the remote configuration and monitoring of the encoder by a processing platform on a different spacecraft processor, like the central CPU or the instrument control unit. More details about the RMAP protocol are provided in Section 2.5.
- The AXI4-Lite memory mapped interface by a properly connected controlling CPU. This possibility is most important in the case when the programmable logic is a part of a System-on-Chip (SoC) design, where the FPGA device fabric includes one or more (soft or hard) CPU cores and the interconnect fabric. As further detailed in Section 2.3, SoC platforms have been extensively used in this thesis for development and testing.
- The AXI4-Lite interface by the JTAG interface provided by the FPGA fabric. This possibility is especially useful during development and debugging, since it allows immediate and easy access to the control and status registers of the accelerator from the developer's PC.

For some processing workloads which involve big volumes of data, it is necessary to provide access to external RAM resources. The accelerator in these cases communicates with the external memory through an AXI (in this thesis AXI3 or AXI4) bus, through a memory controller.

2.3 FPGAs in space

Initially, FPGAs implemented only auxiliary tasks and glue logic in a spacecraft system, while the telemetry and flight control tasks were handled by specialized CPUs, which is the topic of Setion 2.4. However, soon after their introduction FPGAs gained increased popularity for aerospace applications, due to the increased processing power and size,

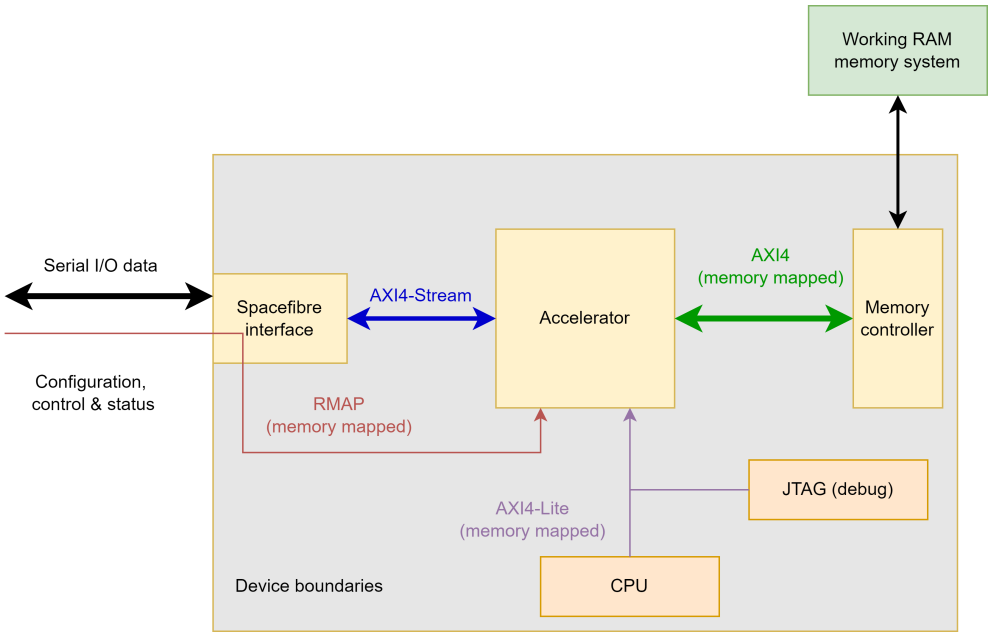

Figure 2.4: Simplified on-board processing reference system.

weight, power, and cost (SWAP-C), when compared to CPUs and GPUs [85], and nowadays, they are widely used for embedded computing in space. They are the only solution to the increasing need of on-board processing resources: contrary to what happened in the past, in modern missions we witness an increasing tendency to decentralise space data processing from the ground stations, by executing more and more tasks on the edge (on-board systems). This tendency embraces even complex algorithms,like artificial intelligence (AI) [62], deep neural networks [67] [129], satellite edge computing [87] and advanced hyperspectral image compression [126].

However, despite their obvious advantages, FPGA devices that can support spaceflight need to be able to withstand the challenging space environment, as already described in Section 2.2 and there is only a limited number of Radiation Hardened By Design (RHBD) FPGAs in the market. The most important device families for spaceflight are manufactured by Microsemi and Xilinx. A common feature of these families is that the configuration memory is based on SRAM technology instead of flash, since the latter is susceptible to radiation effects [91], with an obvious impact on the cost. The products of both vendors share a rich mission heritage, an extensive overview of which is provided in [86].

In addition to the standard hardening techniques, Microsemi PolarFire radiation tolerant FPGAs chips use Silicon-Oxide-Nitride-Silicon (SONOS) Non-Volatile (NV) technology [114], which provides immunity against SEU effects, in addition to low power. The physical layer manufacturing details of the SONOS technology, as well as its rad-hard attrbutes are widely covered in [114]. TMR in the user logic, when required, is assured through suitable provisions from the bundled software (Libero sinplify). On the other hand, the Xilinx SRAM radiation hardened FPGA range includes the legacy Virtex-4QV FPGA (90nm) device family, the Virtex-5QV FPGA (65nm) XQR5VFX130 device and the RT Kintex UltraScale (20

nm) XQRKU060 device, which is currently the state-of-the-art in terms of performance.

With their vendors being based in the USA, however, all these products from Microsemi and Xilinx are subject to USA export controls, like the International Traffic in Arms Regulations (ITAR), which adds insecurity to the European missions' planning. Recently, the NanoXplore family of RHBD devices has been introduced as a European solution [86], although it has not yet practical presence in any real space mission.

Interestingly, an emerging trend for extending the application area of commercial Xilinx ZynQ and ZynQ Ultrascale+ SoCs into aerospace applications has recently risen. A number of research activities has been focusing on the study of the susceptibility of the ZynQ-7000 series SoCs. The SEU behaviour of the ZynQ-7020 SoC's integrated ARM Processing System is the subject of the work in [76]. In [117], the authors present their results on heavy proton SEU testing of the same device, while [24] evaluates the SEE behaviour of the NINANO board used in EYE-SAT nanosatellite. The work in [130] is the most complete analysis of the SEE behaviour of ZynQ-7000 series programmable logic and configuration memory under heavy ion irradiation. One of its major contributions is that it provides the tools for the design of efficient mitigation techniques, including effective ECC and configuration memory scrubbing. At the same time, a multitude of research activities incorporate these SoCs for aerospace applications. In [126], for example, we have introduced a high performance parallel implementation of an accelerator for the CCSDS 123.0-B-1 hyperspectral compression algorithm. This work leverages the resources of both the processing system and the programmable logic to deliver state-of-the-art throughput performance. The authors in [111] propose a hybrid convolutional neural network accelerator for semantic segmentation of image, which is widely used in space applications. Their work is evaluated on Xilinx ZynQ and ZynQ Ultrascale+ MPSoCs, while performing error injection and radiation-beam testing, in order to characterise the response of the proposed architectural framework in the presence of radiation phenomena. In all these cases, mostly soft techniques are used as mitigation measures. TMR effectiveness under heavy ion radiation is evaluated in [115] for a ZynQ 7000 SoC supporting a CCSDS 121.090-B-2 compression IP core, demonstrating a 40% increased Mean Time To Failure (MTTF). A rather complete study of the effectiveness of soft methods is presented in [82]. The key takeaway is that for non mission critical systems, soft SEE mitigation techniques can provide the resilience required for space applications.

In DSCAL, the following FPGA and MPSoC boards are available and used in the scope of the current thesis:

- The KCU105 evaluation board, built around the Kintex UltraScale XCKU040 device, which is the commercial equivalent of the radiation tolerant Kintex Ultrascale XQRKU060. Regarding the rest of the board's equipment, of notable interest to the purposes of this thesis are the two SFP+ cages, which were used for spacefibre integration and the 2 GB of DDR4 RAM component memory at 2400MT/s (over a 64-bit datapath width).
- The ZC706 board, featuring a Zynq-7000 XC7Z045 SoC at speed grade 2, with two ARM Cortex-A9 MPCore hard processors. The specific SoC is a mid-range device

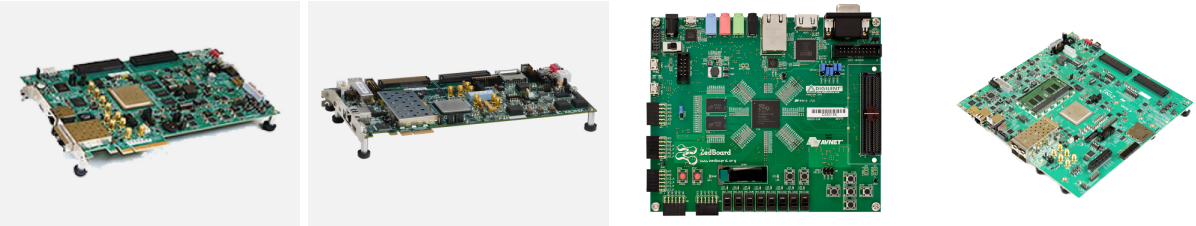

Figure 2.5: From left to right: The KCU105, the ZC706 and the Zedboard development boards. Source: Xilinx website.

of the Zynq-7000 class of heterogeneous SoC devices, which integrates a dual-core ARM (Cortex-A9) processor with Kintex-7 FPGA fabric. One SFP+ cage is included. The board also includes 1 GB of DDR3 RAM connected to the processing system (PS) build around the two ARM processors (component memory), as well as 1 GB of DDR3 RAM for the programmable logic (SODIMM memory). Access to the two memories is independent (both memories can be accessed at the same time).

- The Zedboard, with the Zynq-7000 SoC XC7Z020. The board has no SFP+ connections, but it includes 512MB DDR3 memory connected exclusively to the processing system. Access to the memory space from the programmable logic can be provided from the ZynQ's PS AXI3 high performance (HP) ports.
- The ZCU102 board, which includes an Ultrascale+ XCZU9EG MPSoC, with a quad-core Arm Cortex-A53, 4 GB of DDR4 SODIMM RAM for the processing system, 512 GB of DDR4 component memory for the programmable logic and 4 SFP+ cages, among other things.

2.4 Space-grade CPUs

Similarly to what is described in Section 2.2, the requirements of CPUs are radically different between terrestrial and aerospace applications. Commercial CPUs, target hugely larger market shares, can meet lower time to market requirements and include advanced features and vastly higher performance. Not being able to withstand the harsh environmental conditions typically met in spaceflight, however, they fail to meet the reliability requirements of space missions. Space-qualified CPUs have therefore been developed to mitigate these issues. These CPUs are based on commercial Instruction Set Architectures (ISAs), so that the cost of the ecosystem around them is reduced. The ecosystems includes hardware design processes and tools, as well as software tools for applications development. Historically, MIL-STD-1750 architecture [1] dominated space missions, due to its already widespread adoption by military airborne computers. Quickly, however, following the evolution of commercial architectures, the market was dominated by SPARC and PowerPC. A pictorial overview of the historical evolution of space CPUs is provided in Fig.2.6.

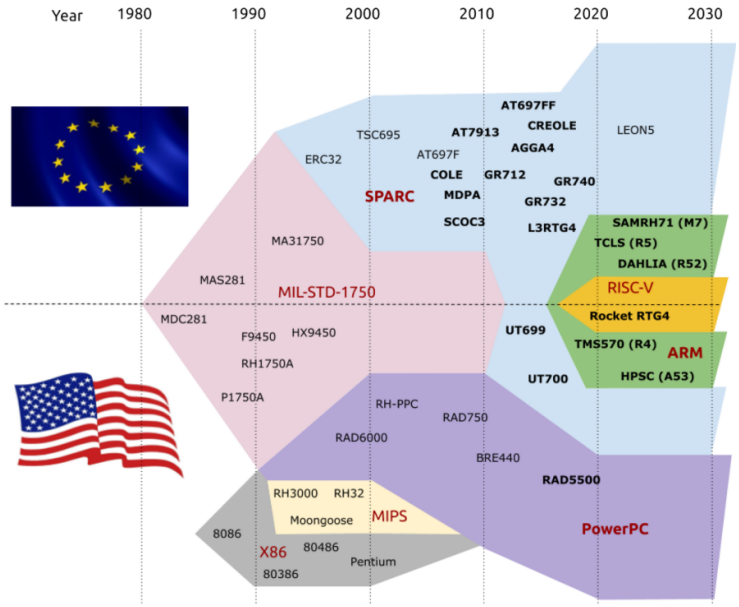

Figure 2.6: History of CPU architectures used in space missions. Source: [53]

The European Space Agency (ESA) has generally opted for the SPARC architecture, mainly because of the widespread availability of software and its open architecture, which allowed the Agency's independence from specific vendors. To this aim, ESA funded the development of the LEON processor in late 1997. One of the principal objectives of the project was the integration of fault tolerant-by-design techniques. The processor should be able to detect and tolerate one error in any register without software intervention, and to suppress effects from Single Event Transient (SET) errors in combinational logic [16]. LEON evolved in the following years and currently, LEON3 [44] is the most widely adopted platform for ESA missions. LEON3 is distributed as synthesizable VHDL model of a 32-bit processor compliant with the IEEE-1754 (SPARC V8) architecture by Aeroflex Gaisler. The distribution is under the GNU GPL license allowing use for any purpose without licensing fee. The most significant upgrades over the previous LEON2 is the support of Symmetric Multi Processing (SMP) and pipelined operation at 5 stages. In space missions, a fault-tolerant version of the processor (LEON3FT) is the one that is widely used. Fault tolerance is assured by the implementation of ECC coding of all on-hip RAM blocks, which is able to detect and correct up to four errors per 32-bit RAM words or per cache memory tag, and all these without performance impact (completely transparent to user applications). The main means for achieving fault-tolerance is by using ECC coding of all on-chip RAM blocks. A famous LEON3-based SoC is the GR712RC from Aeroflex Gaisler.

LEON5 is the latest version of the LEON processor family[45] and it primarily targets high-end FPGA's. Although it has not yet been implemented in space missions, it provides backward compatibility for most of the software implementations that have targeted LEON3 and LEON4 processors, claiming up to 85% higher performance. Nevertheless, the Reduced Instruction Set Computer version V (RISC-V) Instruction Set Architecture

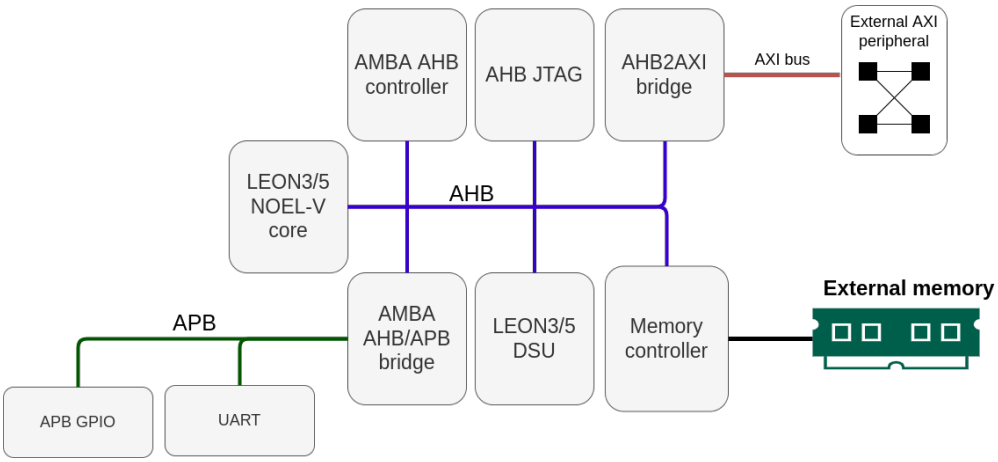

Figure 2.7: A typical GRLIB LEON/NOEL SoC design

(ISA) is expected to dominate upcoming on-board processing applications [53]. In European missions for example, the De-RISC project [48] has recently shown the first milestones for a multi-core RISC-V processor for aerospace designs. The project is based on the NOEL-V 64-bit RISC-V processor core [46] from Aeroflex Gaisler and state-of-the-art hypervisor technology to accomplish high performance workloads, on a complete processing platform for space. The three processor models (LEON3/5, NOEL-V) are distributed as parts of the open-source GRLIB IP library, which is an integrated set of reusable IP cores, designed for system-on-chip (SoC) development and they are available also in fault-tolerant versions for FPGA and ASIC implementations. Typically, they are interconnected through Advanced Microcontroller Bus Architecture (AMBA) Advanced High-performance Bus (AHB) and Advanced Peripheral Bus (APB) interfaces.

A typical SoC reference design used across this thesis and built around a single LEON/NOEL processor core with the peripherals included in the distribution, is depicted in Fig. 2.7. As its name implies, the AHB JTAG component shown in the image provides a JTAG debug link to the SoC, allowing among other things the uploading and debugging of user software to the processor's memory, through the GRMON software tool, which is part of the processor's software ecosystem. Other useful software tools included in the ecosystem are the cross-compiler for the CPU architecture and a simulator (TSIM). The debugging capabilities are completed with the Debug Support Unit (DSU) depicted in the image. This module communicates with the CPU through a dedicated debug interface (in addition to AHB) and has complete control over its pipeline, its registers and the contents of the instruction trace buffer. The processor can be set into debug mode by the DSU, which halts the pipeline. As part of the current work, the LEON/NOEL ecosystem has been set up on the KCU105 and Zedboard development boards, to allow for interaction with custom FPGA peripherals, as well as for software comparison

2.5 Spacewire and spacefibre

Other on-board processing systems examples are instruments, mass-memories, processors, and downlink telemetry physical transmission equipment. The interconnection of these systems is a challenging task: fault-tolerance, error recovery capabilities, low power, simplicity, performance and architectural flexibility place stringent requirements on the design of an on-board network. The interconnections of the equipment in a network with these requirements is feasible only with serial links [20].

Initially, space agencies and manufacturers followed their proprietary approaches to address this issue. The diversity of communication links that arose resulted in high cost, limited development and test time and interoperability issues. Traditionally, NASA missions systems were built around the Serial RapidIO (SRIO), which is a non-proprietary, high-bandwidth, packet-switched system level interconnect, with PCIe also gaining significant attention from the Agency, mainly by virtue of its widespread commercial adoption as a high speed serial link in commercial applications.

In 1992, the demand for interconnection of distributed signal-processing systems led ESA to assign the development of a new standard to the University of Dundee [104]. This process resulted in the first version of ECSS-E-ST-50-12C (SpaceWire) standard [11]. The standard defined a high speed data-handling on-board network and technology. It provided bidirectional, full-duplex data-links at speeds of 2 to 200 Mbit/s, which connect together SpaceWire enabled equipment. Data-handling networks can be built to suit particular applications using point-to-point SpaceWire data-links and routing switches.

The next generation of SpaceWire is the SpaceFibre technology, standardised as ECSS-E-ST-50-11C [10]. Except for higher data rates (6,25 Gbps signalling rate), SpaceFibre comes with other significant enhancements:

- Fibre-optic cabling, with electrical support for backwards compatibility with SpaceWire.
- Multi-laning, which can combine the throughput of multiple physical links (lanes) to support well over 20 Gbit/s.
- Advanced Quality of Service (QoS) mechanisms, like prioritization of Virtual Channels, bandwidth reservation and support of deterministic delivery constraints.

DSCAL is a partner of the Hi-SIDE project (`https://www.hi-side.space/`) consortium and as such, it has been granted a access to SpaceFibre test equipment, featuring a STAR-Ultra PCIe interface and link analyzer card, along with the necessary software tools (GUI and API for the development of custom applications and performance measurements) and an accompanying encrypted IP core netlist for the FPGA's side of the SpaceFibre link. The Hi-Side IP core exposes up to 7 AXI4-Stream 128-bit interfaces to the user logic (one for each virtual channel) and a RMAP port for configuration and control. From the software's side on the host PC, the communication of the software application with the logic implemented on an FPGA is transparent: files are sent to and received from the user logic on the FPGA through straightforward API calls.

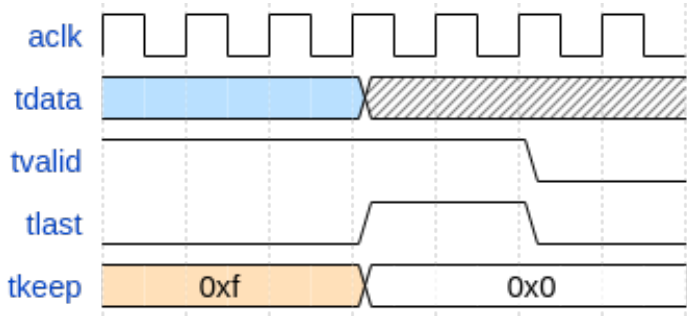

Figure 2.8: Hi-SIDE IP core's behaviour with regard to TLAST signalling.

The provided equipment (host API and the Hi-SIDE core) also supports the multi-laning feature of SpaceFibre, which means that it can aggregate the traffic from multiple SpaceFibre lanes. More specifically, the STAR Ultra PCIe interface card provides two 4-lane SpaceFibre links to a suitable corresponding interface of up to 20 Gbit/s per link. The maximum data rate can be achieved when all four 6,25 Gbit/s lanes are used. Because of the 8b/10b encoding on the SpaceFibre link, only 80% of the lane bandwidth is available to the user logic. Of the equipment available at DSCAL, only the Zynq UltraScale+ MPSoC ZCU102 development board with 4 SFP+ connector cases can support the maximum data rate of 20 Gbit/s. The ZCU105 card, however, which has been extensively used in this work can only provide up to 10 Gbit/s of user bandwidth.

As part of the Hi-Side project's deliverables, a sample design for the KCU105 has also been provided to DSCAL. The reference design includes a basic demo for a loopback test through the card's FMC connectors, without connectivity to the host PC. The reference design had to be modified, so that a suitable SERDES is mapped to the transceivers allocated to the SFP+ connectors of the board. The transceivers also needed to be parameterized and a suitable clock source to be configured on the board and connected to the transceivers' CPLLs. This process was different for the various development boards used in DSCAL: the KCU105 and ZCU102 boards use GTH transceivers, while ZC706 uses GTX transceivers.

Another challenge that had to be addressed with the delivered equipment was related to the behaviour of the AXI stream interface of the provided core: the TLAST signal at the end of a SpaceFibre packet's transmission is asserted on invalid TDATA, which, however, are framed by an asserted TVALID and de-asserted TKEEP signals, as depicted in Fig. 2.8. Existing AXI4-stream bridges proved to be unable to handle this behaviour properly and custom logic was needed for that. The observed behaviour is mainly caused by the structure of the a space packet: data characters are followed by an End Of Packet (EOP) character. At the receiving side, this is translated into TLAST by the SpaceFibre core. If the valid payload data are not aligned with the 128-bit wide AXI4-Stream interface, one or more de-asserted TKEEP signals need to be inserted. The SpaceFibre IP core obviously issues another beat of de-asserted TKEEP signals, in order to ensure consistent behaviour.

Figure 2.9 is a block diagram of the resulting environment used for testing throughout

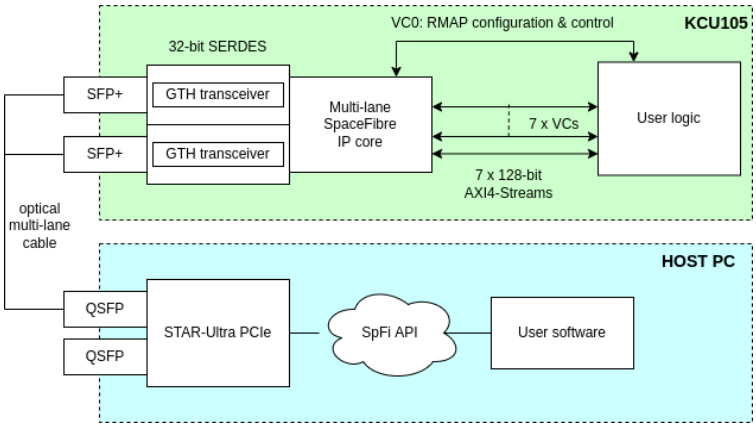

Figure 2.9: Block diagram of the SpaceFibre equipment and environment

this thesis, as well as to the contributing projects. The design depicted refers to the configuration used on the KCU105 board, but except for the number of lanes and type of transceivers, it is the same for all the other boards.

On the PC side, the STAR-System software provided as part of the Hi-SIDE equipment provides the drivers of the SpaceFibre STAR-Ultra PCIe board, as well as the software tools for transmitting and receiving packets to and from the SpaceFibre IP code. The software bundle provides two options: a set of GUI applications and a complete API for the development of custom software tools. In both cases, statistics and performance data can be derived. Figure 2.10 gives an example of the GUI applications. In order to streamline the automatic execution of scripts including SpaceFibre transactions, the following custom applications were developed, based on the STAR-System API (optional parameters are in brackets):

- spfilink $-l < lanes > -r < rate > [-p < port > -x < TX\ scrambling >$: resets all lanes and establishes a new link with the required parameters. Default is port 1 and TX scrambling ON.

- rmapRead $< address >$: reads a 4-byte register at a user-specified address.

- rmapWrite $< address >< value >$: writes a 32-bit value at a register at a user-specified address.

- filesend -c $< channel > -i < file >$: sends a binary file to the specified channel of the master stream interface of the core.

- filereceive -c $< channel > -o < file >$: receives data from the specified channel until an End Of Packet (EOP) character is received and writes the stream into a file.

- filesendreceive -c $< channel > -i < file > -o < file >$: execute the previous actions in the same call.

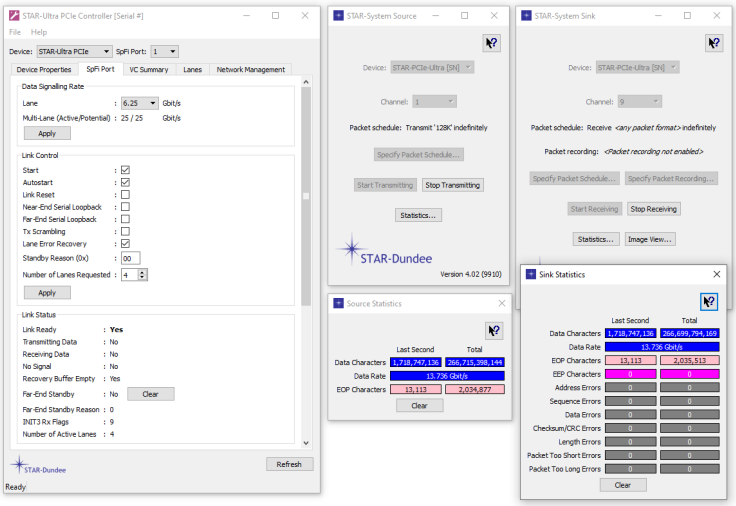

Figure 2.10: GUI applications examples

The GRLIB library from Gaisler [43] also includes a SpaceFibre core, which was also connected to the GTX and GTH transceivers of the DSCAL boards. Because of its lack of multi-lane support, however, that core could not provide connectivity with the Hi-Side host PC, like in Fig. 2.9, which is exclusively multi-lane: even when a single line is connected between a multi-line enabled SpaceFibre end and a single-line one, the multi-lane layer of the PC lane needs to be initialised in order for the link to be successfully established. The Gaisler IP core does not have AXI4-Stream interfaces, but a modified FIFO interface for each of the virtual channels (up to 32) it supports. In this interface, the EOP and FILL control characters are being transferred as data, but they are marked with a special 4-bit side-band signal, each bit of which is asserted when a control byte is present in the corresponding byte of the 4-byte data bus. Wrapper logic needed to be created, so that control characters are removed from the stream and the AXI4-Stream signals (TKEEP and TLAST) behave according to the protocol specification. The lack of a single-lane enabled host PC at DSCAL limits the application of this core only to communication between two cards, which, at the time of writing is work in progress.

Overall, the SpaceFibre interface was extensively used in this thesis, for at-speed testing of the IP cores, from the host PC.

3. QC LDPC ENCODER IMPLEMENTATIONS

3.1 QC LDPC Linear Block Codes

In this section, a brief introduction into the elements of coding theory that are relevant to this work is attempted, which defines the notation for the rest of the Section. The analysis provided herein is by no means complete, since the break-down of the theoretical properties of the CCSDS codes is not a goal of this thesis and it is adequately covered by the relevant textbooks on information theory.

Let's assume a source producing a sequence of information bits, or formally, symbols over $\mathbb{F}_2$. In order to establish reliable communication over a noisy channel, this information sequence needs to be transformed into another one with a higher rate, with redundant information as a countermeasure to noise. There are generally two methods for this, which correspond to the two broad categories of error correcting codes: convolutional and block codes. In the case of convolutional codes, the transformation of the input sequence into the output (encoded) takes place in a stream-oriented fashion: the output of the encoder at any given time is a function of the current and all the previous values of the input. In the case of block codes, on the other hand, a grouping of k information bits is considered as an individual message and encoded separately. Let $s = \left[s_0, s_1, ...s_{k-1}\right]$ be the information sequence to be encoded. It follows that there are 2^k possible messages, each of which is mapped into a n-bit sequence c, which is an element of the k-dimensional subspace of $\{\mathbb{F}_2\}^n$:

$$s \rightarrow c : s \in \{\mathbb{F}_2\}^k, c \in \{\mathbb{F}_2\}^n \tag{3.1}$$

The binary sequence c is a linear function of s and it is a codeword of the code. Obviously, there are 2^k codewords of what is referred to as a (n, k) linear block code. The rate of the code is the ratio k/n of total bits, which means that the encoded sequence is larger than the initial by $n - k$ additional bits. In most practical codes, exactly $n - k$ parity bits form a separate sequence which is appended to the information sequence. In other words, the codeword has the form $c = \left[s, p\right]$, where p is the sequence of the parity bits. These linear block codes are referred to as systematic codes and they exhibit obvious implementation advantages over the non-systematic ones during encoding. All the codes in this thesis are systematic codes.

A linear block code is uniquely defined by its generator matrix, denoted as G in this thesis. The generator matrix defines the linear relationship between s and c. Simply set, if we represent s and c as row vectors, this is described by (3.4):

$$s = \begin{bmatrix} s_0 & s_1 & \dots & s_{k-1} \end{bmatrix} \tag{3.2}$$

$$c = \begin{bmatrix} c_0 & c_1 & \dots & c_{n-1} \end{bmatrix} \tag{3.3}$$

$$c = s^T G \tag{3.4}$$

G is a $k \times (n - k)$ matrix and for linear codes, each codeword is a linear combination of

the rows of G. In the special case of systematic codes, the firs k columns are the $k \times k$ identity matrix. Conversely, a linear block code is defined by the null space of G, which is called the parity-check matrix of the code:

$$GH^T = \mathbb{0}_{n \times k} \tag{3.5}$$

In (3.5), $\mathbb{0}_{n \times k}$ is the $n \times k$ zero matrix. It can also be shown that a binary sequence of n-bits is a codeword of the code defined by H if and only if equation 3.6 holds. This is called the parity-check equation and it is important for decoding, as well as for encoding, since the decoding process attempts to calculate the vector s that satisfies (3.6) and which has the smallest distance from the received codeword $\hat{c}$: $s = \underset{s}{argmin}\{d_{min}(c, \hat{c})\}$.

$$cH^T = \mathbb{0}_{1 \times (n-k)} \tag{3.6}$$

The rows of H can be understood as the linear equations that every codeword has to satisfy (in order for it to be a valid codeword). Although there are always n columns and $n - k$ rows in H, it is not always true that all these parity-check equations are linearly independent: in these cases, there a number of rows is a linear combination of a subset of $(n - k)$ rows and H is not a full-rank matrix. The reasons for this complication are related to the error correcting characteristics of the codes and they reside in the field of information theory. Some of the codes in this thesis fall into this category.

As already mentioned briefly in Section 1.1, LDPC codes are linear block block codes with large code lengths and sparse density matrices. The initial Gallager codes were random, which means that their parity-check matrix had no structure at all. Although this randomness resulted in excellent error-correcting capabilities, their hardware implementation was challenging. It can be shown, however, that capacity approaching codes can be structured codes and various techniques have been proposed for the construction of very good performing structured codes. A famous structure in the design of LDPC codes is the Quasi-Cyclic (QC) structure, according to which the parity check matrix consists of an array of juxtaposed cyclic sub matrices, named the circulants, which can be efficiently implemented. QC-LDPC codes have been adopted by many modern communication standards, such as IEEE 802.11, 802.16 and DVB-S2, and the list of their endorsement is continuously growing. QC LDPC codes have also been adopted by the Consultative Committee for Space Data Systems (CCSDS) as recommended standard for on-board channel coding in Near-Earth and Deep-Space communications.

A further simplification of the QC structure with significant implementation advantages is possible through the protograph code design approach. Protograph codes are based on the Tanner graph representation of the parity-check matrix of LDPC codes. A Tanner (or bipartite) graph [119] contains two types of connected nodes: each row of H is represented as a check node in the graph and each column as a variable node. There is a connection between one check node c_i and a variable node u_j if and only if $h_{i,j} = 1$. For a Tanner graph, $\mathcal{G} = (\mathcal{V}, \mathcal{C}, \mathcal{E})$, $\mathcal{V}, \mathcal{C}$ and $\mathcal{E}$ are the sets of variable nodes, check nodes and edges respectively.

A protograph is defined by a small Tanner graph, comprising a low number of check and variable nodes. Contrary to a full Tanner graph, in which there is always a single con-

nection between a check and a variable node, a protograph can contain parallel edges, which correspond to multiple connections between these nodes. The significance of a multiple node is going to become apparent from the expansion process described later in the current Section. An example of a protograph is Figure 3.2. The cardinality and the members of the sets $\mathcal{V}, \mathcal{C}$ and $\mathcal{E}$ are defined with the help of analytical methods, like extrinsic information (EXIT) chart analysis [26] and asymptotic weight distribution (AWD) [27], so that decoding performance criteria are met. These techniques depend, among other things, on the channel model adopted. Starting from the protograph, the full final graph of the codes we are interested in is obtained in usually two expansion steps as follows: first, the protograph is repeated x times and the edges on the expanded (or derived) graph are permuted among the copies of the protograph, so that any parallel edges in the graph are removed. The edge permutations are calculated according to a computer search algorithm, like Progressive Edge Growth (PEG) [78] or Approximate Cycle Extrinsic Message Degree (ACE) [125], which aims to maximize the girth and the distance of the code. These parameters are strongly related to the performance characteristics of the code and more specifically, the error floor and the decoding threshold. After this first step, a second expansion step follows and the final graph is derived. This time, the nodes are repeated m times but no edge permutations between the expanded nodes are performed: if node u_j is connected to c_i, that means that there is a connection between u_{j+1} and c_{j+1} (addition is modulo-m). It is important, however to note that node degree (i.e. the number of incident nodes) is preserved during each expansion step, and form what is called as the neighborhood of a node: if a variable node $u_j \in \mathcal{V}$ is connected to a $c_i \in \mathcal{C}$ with one edge, any variable node in the expanded graph that has been created as a copy of u_j, is connected with exactly one replica of c_i. It follows that parallel edges in $\mathcal{G}$ result in as many connections between the copies of the connected nodes as the number of these parallel edges.

Following the parity-check matrix equivalent representation for protograph-based codes, let H' be the parity-check matrix after the first expansion step. During the second expansion step, each element $H'(i,j)$ representing a single edge at the initial protograph is assigned the value $I_m^{(0)}$, while elements of the matrix corresponding to multiple parallel edges are assigned the value $I_m^{(x)}$, where $x \in [0, m-1]$, $I_m^{(x)}$ is the $m \times m$ identity matrix I_m, rotated by x positions to the right. Absence of any edge between two nodes u'_j and c'_i in the graph $\mathcal{G}' = (\mathcal{V}', \mathcal{C}', \mathcal{E}')$ which is derived after the first expansion step is indicated with a $m \times m$ zero matrix in $H'(i,j)$.

As a consequence of the expansion process, the resulting code is typically a quasi-cyclic (QC) code, the H matrix of which is an array of juxtaposed permutation or zero matrices. An example of the CCSDS rate 1/2 code is given in Figure 3.1. For CCSDS AR4JA codes, $x = 4$, as explained later in the current Section.

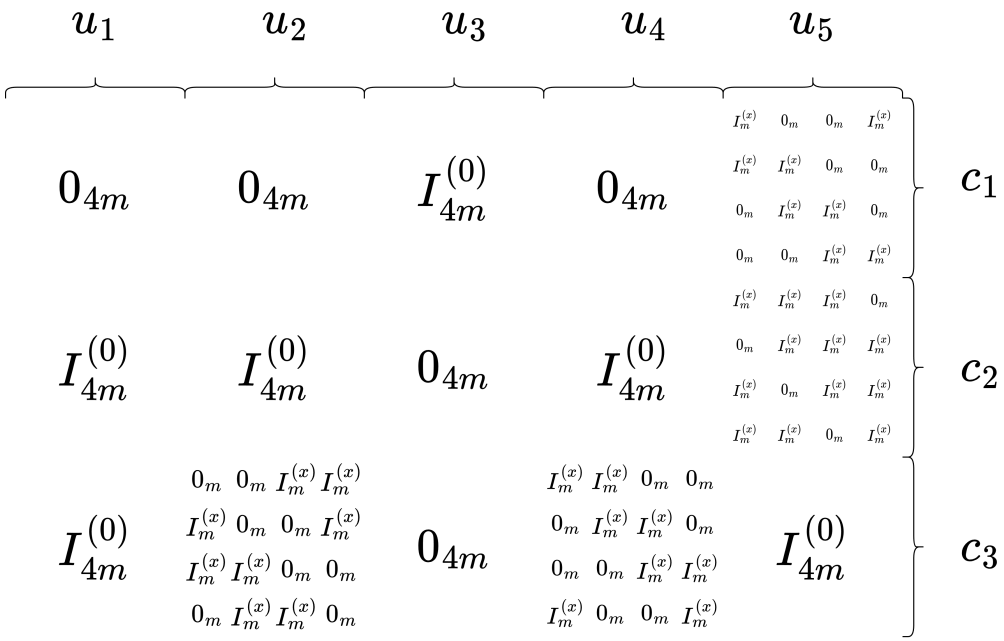

Figure 3.1: Example H matrix of the CCSDS rate 1/2 AR4JA code

3.2 CCSDS codes

Many CCSDS standards recommend QC-LDPC codes at the channel coding and synchronization sub-layer. In particular, CCSDS the current work, the focus is on the codes defined in [32]. Two code class classes are recommended by the standard: one optimized for deep-space Accumulate-Repeat by 4-Jagged Accumulate (AR4JA) and the one for near-earth communications (C2). One member of the AR4JA family (the rate 1/2, length 2048 bits code) has been endorsed by other CCSDS standards as well, including the Proximity-1 space link protocol and the upcoming update of [35] for Optical On-Off Keying. The special properties and challenges of these codes are described in the current sub-section.

The standard provides the option for randomization of the output codeword to ensure sufficient transition density on the transmitted vector. The encoder's output is the Channel Access Data Unit (CADU), consisting of the optionally randomized codeword, prepended by a 64-bit (for AR4JA) or 32-bit (C2) synchronization sequence (Attached Sync Marker-ASM).

3.2.1 AR4JA codes for deep-space communications

For deep-space communications, nine protograph-based AR4JA codes are defined in the recommended standard, which are the result of the combination of three block length

Table 3.1: M parameter of AR4JA codes

Rate	k=1024	k=4096	k=16384
$1/2$	512	2048	8192
$2/3$	256	1024	4096
$4/5$	128	512	2048

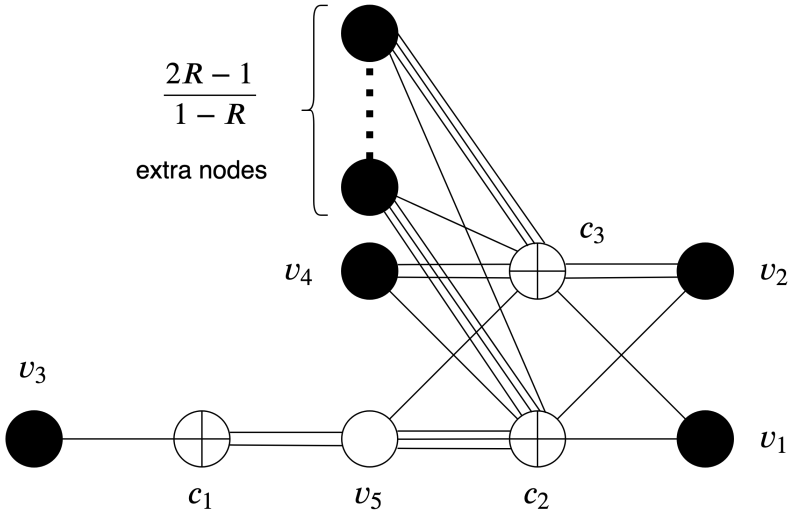

Figure 3.2: AR4JA protograph. R is the rate of the code

sizes (1024, 4096 and 16384 bits) over three code rates: 1/2, 2/3 and 4/5. The parity check matrices of these codes consist of circulant sparse $M \times M$ submatrices, where M is a parameter dependent on block length and code rate and is given in Table 3.1. All these codes are constructed by uplifting the protograph depicted in Fig.3.2, in which node v_5 corresponds to punctured bits in the codeword: the parity bits that are derived during the protograph expansion process from this node are not transmitted, and their sole purpose is to increase the rate of the code.

In the first step of the protograph expansion process, parameter $x = 4$ for all AR4JA codes. For the second expansion step, parameter $m = M/4$. The last $M = 4m$ parity bits, which correspond to the last column of Fig. 3.1 are not transmitted.

From the QC parity-check matrix, the generator matrix can be calculated in systematic QC form, using Gaussian elimination. The generator matrix for each member of the family has the form $G = \begin{bmatrix} I_{MK} & W_{n-k} \end{bmatrix}$, where I_{MK} is the $MK \times MK$ identity matrix and W_{n-k} is a dense matrix of size $MK \times 2M$, and K is a parameter dependent on the code rate. Sub-matrix W itself has the structure of (3.7), where $r = 4K$, $c = 8$ and each W_{ij} is a $M/4 \times M/4$ dense circulant. Note that punctured bits have been omitted from the matrix G.

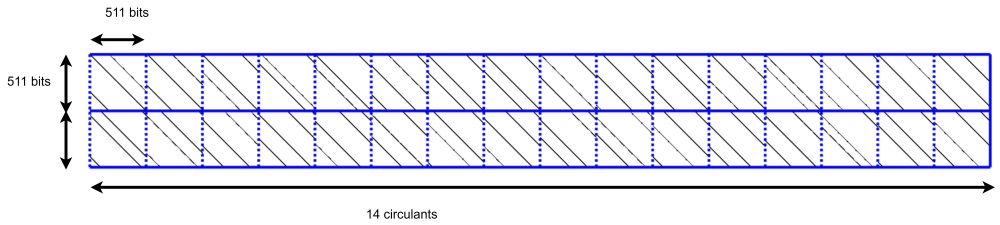

Figure 3.3: The parity-check matrix of the C2 code.

$$W_{n-k} = \begin{bmatrix} W_{1,1} & \dots & W_{1,c} \\ \vdots & \ddots & \vdots \\ W_{r,1} & \dots & W_{r,c} \end{bmatrix} \tag{3.7}$$

3.2.2 C2 code for near-earth communications

For near-earth communications, a basic (8176, 7156) LDPC code is defined, constructed on the 3-dimensional finite geometry EG(3,2^3) over the field GF(2^3). The QC parity check matrix of the code is displayed in Fig. 3.3 and it consists of a 2×14 array of 511×511 sparse circulants. Conversely, the non-systematic part of the generator matrix in systematic circulant form is a 14×2 array of 511×511 dense circulants. The encoding process prepends 18 zero bits to the 7136 bits of incoming frame to be encoded. These bits participate in the encoding process but they are not transmitted as a part of the systematic output of the encoder. Two tailing bits are also added to the final codeword, to ensure that the output codeword length is also divisible by 8 and 16. With the addition of these tailing bits, the dimensions of the recommended code are finally (8160, 7136). The complete encoding process for the C2 code is diagrammatically shown in 3.4. It is obvious that the fact that the circulants' dimensions are not powers of two introduces implementation challenges for a hardware encoded and decoder expansion, as does the shortening and addition of extra bits in the final codeword. Note also that the C2 code is not a protograph code: its cyclic submatrices are not permutation or zero matrices.

3.2.3 Performance characteristics of the various CCSDS codes

The LDPC codes proposed by the CCSDS offer substantially advantageous performance over the convolutional and Reed-Solomon (RS) codes which they replaced. A simplified error-correction performance diagram derived from software simulations by the JPL over the Gaussian channel and BPSK modulation is shown in Fig. 3.5. Limited information is provided concerning the parameters of the simulation [31], however, it is obvious that Turbo and LDPC codes outperform the simple RS and convolutional codes, at least in terms of error-correcting performance. These simple codes, however, have a place in modern spaceflight applications, when implementation complexity is the most important

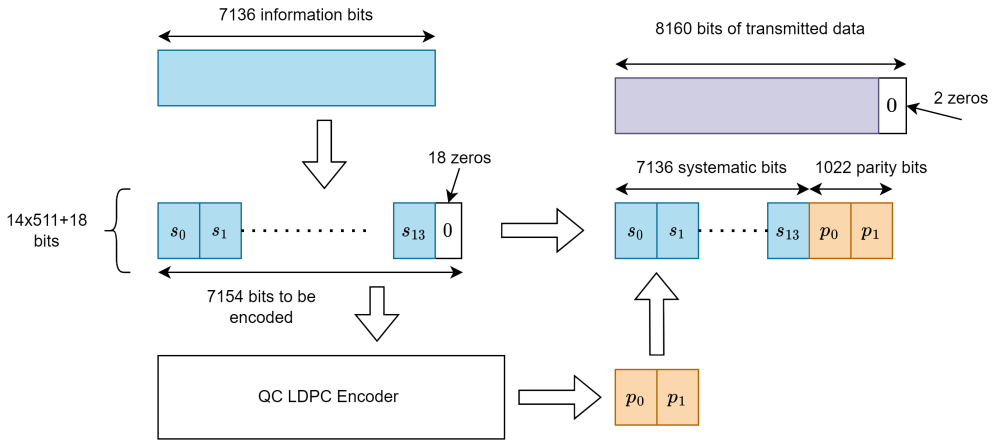

Figure 3.4: The encoding process for the C2 code.

criterion [18]. It has also to be emphasised that the performance of Turbo and LDPC codes is, to a great extent, determined by the decoding processes. This is further explained in the following paragraphs.

There are some fundamental performance limits that pertain to every FEC code. First of all, the capacity of a channel defines the ultimate boundary of the achievable region where channel coding can operate. The minimum achievable BER as a function of E_b/N_0 for various rates is displayed in Fig. 3.6 for the binary input AWGN channel (BI-AWGN). The calculation of these curves starts from Shannon's noiseless source coding theorem: the maximum rate is equal to the mutual information between a channel input and the channel output [25] $R < m$. By inserting $E_b/N_0 = 1/2\sigma^2 R$ and estimating numerically (Monte-Carlo) the BI-AWGN capacity, the performance limits can be derived. The area on the right of each rate defines the achievable region for each code rate.

Another limitation comes from the constraints on the block length of a code. The rate-dependent Shannon performance limits assume infinite block lengths and the minimum achievable BER for each rate is asymptotically approached as code length approaches infinity. Fig.3.7 displays this approximation. The horizontal lines represent the absolute capacity limits (for the BI-AWGN channel) and the "bound" lines the sphere-packing bounds imposed by the limited block length.

Regarding RS codes, the CCSDS recommends a (255,223) RS code for higher error-correcting performance and a (255,239) for lower overhead. RS decoding processes only hard symbols: the input to the RS encoder are symbols of the RS code, which are, in the end, exact bits (either 1, 0) or erasures, according to a slight variation of the basic RS decoder and their performance curves can therefore be analytically calculated, without dependency on the decoder. RS are also maximum distance separable codes (MDS): their minimum distance is equal to the Singleton bound [70], meaning that it is the most optimal. However, RS codes have never been deployed alone in space missions. They are typically combined with interleaving, which further improves their error-correcting performance by spreading error bursts along multiple RS codewords and concatenated with

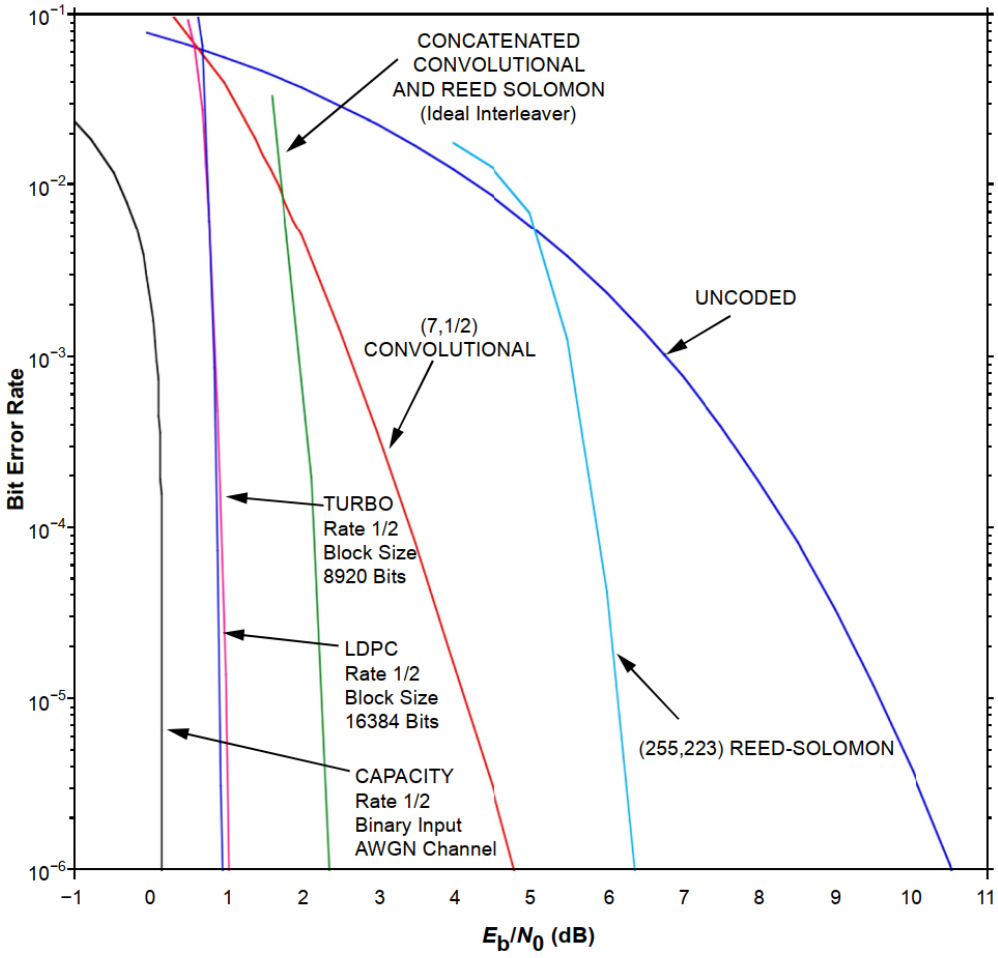

Figure 3.5: Performance comparison of the CCSDS recommended codes. Source: [31].

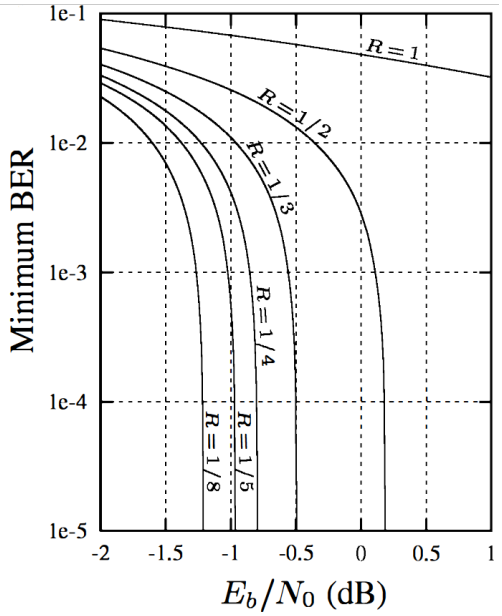

Figure 3.6: The minimum achievable BER for various rates, as a function of E_b/N_0. Source: [25]

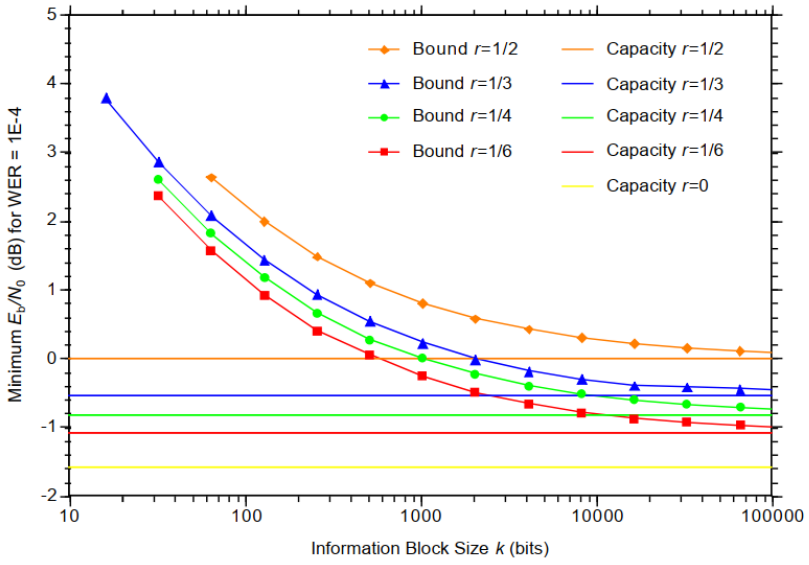

Figure 3.7: Sphere Packings Bound on performance for various rates over the BI-AWGN. Source: [31]

an outer convolutional code: the input sequence is first encoded with the RS code, interleaved and then encoded again with a convolutional code. In this case, as depicted in Fig. 3.5, substantial coding gain is achieved and for many decades, concatenated RS and convolutional codes were the standards channel coding solution.

Convolutional code decoding introduces trade-offs between complexity and performance. There exist various algorithms for the decoding of convolutional codes, the most important of them being the maximum likelihood sequence decoder (MLSD) and the bit-wise maximum a-posteriori decoder (MAP) of the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm [21]. As its name suggests, the former calculates the most probable path in the code's Trellis with the Viterbi algorithm, while the latter calculates the same path with an iterative algorithm. Convolutional decoding is "soft", meaning that the readings from the channel are continuous values, and the selection of the quantisation step is an important parameter affecting the coding performance.

With the advent of Turbo codes, the capacity limit was further approached, as clearly seen in Fig. 3.5. Since Turbo encoding is practically a combination of two convolutional encoders, both of which receive an interleaved version of the same input sequence, the Turbo decoder is accordingly based on two convolutional decoders (typically BCJR), which iteratively exchange messages about the belief each one has on the initial sequence. In each iteration of the BCJR algorithm, each constituent decoder provides an enhanced estimate of the initial sequence, based on the extrinsic knowledge it receives from the other encoder, according to the turbo principle (iterative re-inforcement of belief propagation [112]). There are therefore many trade-offs in the Turbo decoding process, each of which has an effect on the achieved performance. These include the decoding algorithm itself (soft-input soft-output maximum A-Posteriori Probability decoders are the most common), the channel input values and decoders' messages representation (typically log-likelihood ratio, rather than absolute values), the quantisation step of the soft values from the channel and the number of iterations and termination conditions of the algorithm (stopping rules). The total implementation losses, however, can be kept to negligible levels (as low as 0.03 dB in [95]), with adequate resource allocation.

Finally, LDPC decoding is mainly performed with iterative algorithms, according to which the nodes of the Tanner graph of the code exchange soft messages which convey the belief that each node has about its value (zero or one) to their incident nodes on the graph. Fig. 3.8 provides a simplified overview of the basic sum-product decoding algorithm (SPA). It presents a small part of the bipartite graph of the code, which includes two variable nodes (u_i, $u_{i'}$) that correspond to received symbols and two connected check nodes (c_j, $c_{j'}$) that correspond to two parity equations. The variable nodes are initialised with the log-likelihood probability from the received channel, according to (3.8), where c_i is the received code symbol. In each iteration of the algorithm, the variable node v_i calculates a soft value, by summing its own so far knowledge (or belief) with the corresponding values $u_{j\to i}$ that it receives from its adjacent check nodes, as in (3.9) ($u_{j\to i}$ are initialised to zero at the beginning of the algorithm). Then, variable node v_i sends to the connected check node c_j the information is has calculated for its value, minus the check node's own belief of λ_i, as in (3.10). Next, each check node c_j calculates the value $u_{j\to i}^{iter}$, which combines

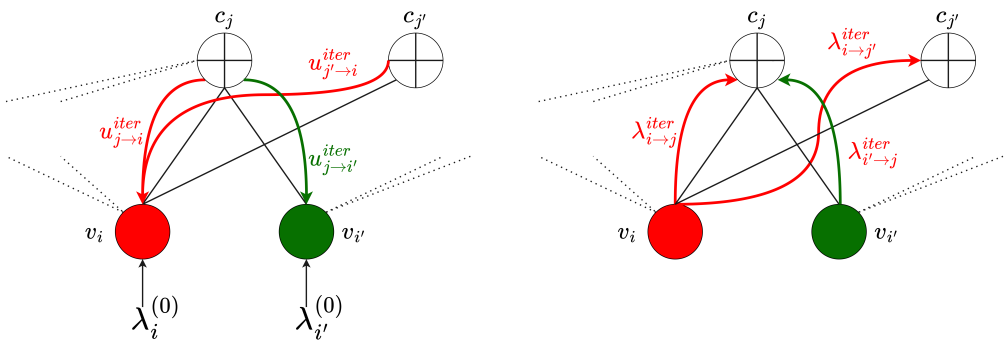

Figure 3.8: Simplified overview of SPA decoding. Variable node update on the left and check node update on the right. The arrows represent message exchanges in each iteration.

the received λ_i values from all connected variable nodes, minus the value of the node i, to which the message $u^{iter}_{j\to i}$ is transmitted. The algorithm ends if a hard decision (based only on the sign of λ_i) leads to a valid codeword (one that satisfies all parity check equations), or an iteration limit has been reached.

$$\lambda_i^{(0)} = log(\frac{Pr(c_i = 0)}{Pr(c_i = 1)}) \tag{3.8}$$

$$\lambda_i^{(iter)} = \lambda_i^{(iter-1)} + \sum_j u_{j\to i} \tag{3.9}$$

$$\lambda_{i\to j}^{(iter)} = \lambda_i^{(iter-1)} + \sum_{j\neq i} u_{j\to i} \tag{3.10}$$

$$u^{iter}_{j\to i} = 2tanh^{-1}\left(\prod_{i\neq j} tanh\frac{\lambda_i^{(iter)}}{2}\right) \tag{3.11}$$

It is obvious from the above description that the selection of the maximum number of iterations and the quantization step are critical design issues that affect the performance of the decoder. An important hardware implementation consideration is also the update message scheduling: QC codes support, for example, layered scheduling [50], which allows for efficient parallel implementations. Most importantly, the hardware implementation of equation (3.11) is challenging and several approximations have been proposed, each with different trade-offs between performance and complexity.

To conclude the current subsection, simple codes like concatenated RS and convolutional codes can offer substantial error-correcting performance and are the obvious selection when implementation complexity is the main issue. On the other hand, LDPC and Turbo codes can perform very close to the capacity limits, if the complexity price is worth being

paid, which mainly refers to the decoder complexity. Since the complexity of Turbo decoders is constant for all rates, while that of LDPC is reduced linearly as the code rate increases, Turbo codes are a preferable option for rates lower than 1/2. For higher rates, LDPC codes can offer better performance with similar complexity. It has to be emphasized, however, that this is a coarse simplification and there is no simple solution for all the design issues of a FEC function within a communications system. Such attempts in [116], [75] have made critical assumptions and they have specified limiting conditions to the comparisons.

3.3 LDPC encoding methods and their limitations

In general, LDPC encoding refers to the process of calculating the mapping $s \rightarrow c$ of a k-bit binary vector $s \in \{\mathbb{F}_2\}^k$ to the proper element c of the k-dimensional subspace $V \subset \{\mathbb{F}_2\}^n$, according to the code definition, which is defined by the parity-check matrix H of the code, so that the parity-check equation $cH^T = 0$ is satisfied.

The encoding methods for LDPC codes that have been proposed so far are the following:

3.3.1 Direct method

The direct method involves the application of Gaussian elimination to calculate the generator matrix G from the null space of the parity-check matrix H of the code, that is to solve the equation $GH^T = 0$. This process takes place offline and depending on the encoder's implementation details, the generator matrix data or structure is stored into the encoder. A codeword c can thus be calculated from the input information block s through the vector-matrix multiplication $c = sG$.

The generator matrix of all practical codes can be transformed in systematic form, through linear operations of its rows and columns. For a (n, k) linear block code in this case $G = \begin{bmatrix} I_k & W_{n-k} \end{bmatrix}$, where I_k is the $k \times k$ identity matrix and for QC codes, W_{n-k} is an array of dense cyclic sub-matrices, with the structure of (3.7). The resulting codeword c is consequently $c = \begin{bmatrix} s & p \end{bmatrix}$, where p is the vector of the $n - k$ parity bits. In this case, the encoders implementing this method need only to store the $r \times c \times m$ bits of the first rows (or columns) of these circulants. However, despite the fact that the initial parity-check matrix H is sparse, the resulting $W_{n,k}$ matrix and consequently its constituent circulants are dense matrices. The generator matrix of the (2048,1024) AR4JA CCDSS code is displayed in Fig. 3.9 as an example: there are 8×8 dense circulants of 128 bits in this code.

Encoders proposed in [121],[140],[94],[142],[17],[135],[136] are based on the direct encoding method. My preliminary work in [121] has introduced an efficient architecture for the parallel execution of the vector-matrix multiplication involved in the direct encoding method, by leveraging the inherent parallelism of the generator matrix of CCSDS codes, achieving.

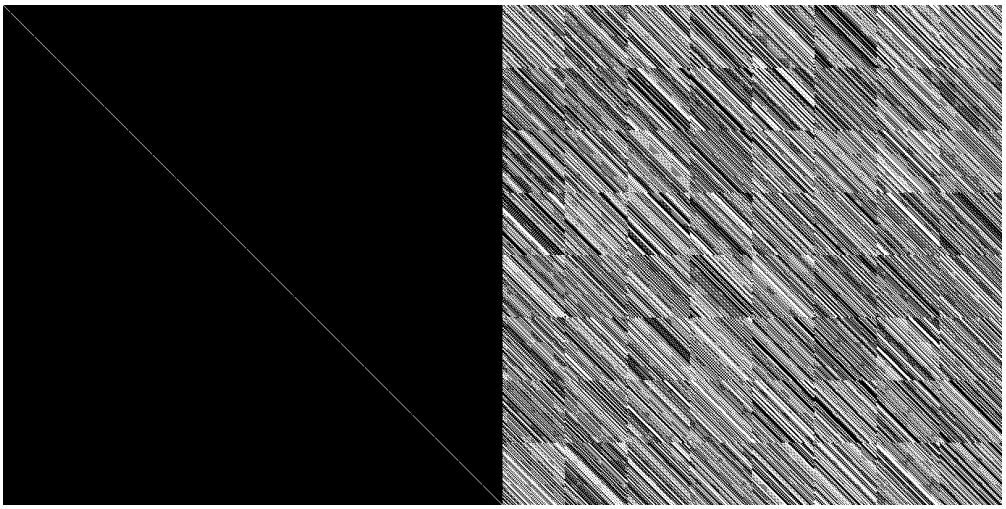

Figure 3.9: The generator matrix of the (2048,1024) AR4JA CCSDS

Although the work in [140] focuses on CCSDS codes, the proposed architecture handles encoding inefficiently, requiring large XOR operations over a significant number of bits ($k/2$ or 2048 in the provided example) and at the same time register resources are wasted. Algorithmically equivalent approaches are found in [135] and the parallel SRAA of [142]. The required logic resources for hardware implementation are inevitably a large portion of a Virtex7-xc7vx485t FPGA, which render their approach impractical.

The authors in [142] propose various types of encoding circuits, based on shift registers, which achieve encoding complexity that is linearly proportional to the number of parity bits of the code ($n - k$ according to the notation in this work), or the n total bits of the code in the case of the parallel approach. The SRAA serial encoding scheme described is practically the naive approach provided in the CCSDS standard [32], based on a shift register for the circulants and a register for the calculation of parity bits. The calculated complexity does not include the memory and the necessary circuitry for the loading of the generators $g_{i,j}$ (first rows of $W_{i,j}$ in (3.7)) of the circulants to the SRAA shift registers, which incur significant resources cost in practical implementations, as it will be shown in a subsequent section. According to the parallel SRAA approach, which achieves encoding in cb cycles (following the writers' notation), all k input bits participate in the calculation of each parity bit in one clock cycle. This architecture could not be implemented with reasonable resources in practical encoders for codes with block lengths in the range of several thousands of bits and should be considered only as a theoretical approach for academic research purposes only. Even in this case however, the AND-XOR binary calculations on a large number of bits would necessitate large combinatorial paths and would severely jeopardize throughput performance. The two-stage encoding scheme which is also described in [142] is practically the *H2-inverse* method described later in the current Section.

The work in [17] proposes an architecture based on Linear Feedback Shift Registers (LFSRs). The input information bits are multiplied with the first rows of the circulants $W_{i,j}$ and

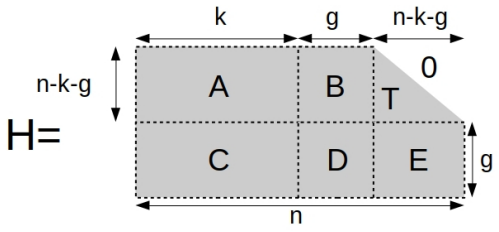

Figure 3.10: Structure or H matrix for the R-U method

instead of rotating the circulant registers, the rotation concerns the output register, which contains the parity bits at the end of the encoding process. The work in [131] describes an implementation of [17].

The approach described in [135] is algorithmically equivalent to the parallel SRAA approach of [142], without taking advantage of the QC structure of the targeted codes (IEEE 802.16e). Its performance however is also dominated by the large XOR binary operation involved. In addition, the memory requirements for the storage of the generator matrix, totalling $(n-k)k$ bits, pose considerable constrains to the associated hardware. The work in [136] is another adoption of the SRAA architecture of [142], employing the direct method and optimized for sparse circulants. Finally, another implementation of the SRAA architecture is described in [38].

3.3.2 R-U method

This method is based on the fact that the codeword can be calculated directly from the H matrix by solving the system of equations defined by the parity-check equation $Hc^T = 0$. The Richardson-Urbanke (R-U) method [110] solves this equation with complexity almost linear to the block length, provided that the parity-check matrix of the corresponding QC-LDPC code has approximate upper-triangular structure, or it can be transformed to such form, which is depicted in Fig.3.10. For a systematic code with a H matrix of size $(n-k)\times k$, the calculated codeword has the form $c = \begin{bmatrix} s & p_1 & p_2 \end{bmatrix}$, where s is the input vector and p_1, p_2 are parity bits vectors of length g and $m-g$ respectively and and the parity bits are calculated according to (3.12), (3.13), (3.14).

$$\varphi = ET^{-1}B + D \tag{3.12}$$

$$p_1^T = \varphi^{-1}(ET^{-1}A + C)s^T \tag{3.13}$$

$$p_2^T = T^{-1}(As^T + Bp_1^T) \tag{3.14}$$

The above equations involve many sparse matrices, but only a single dense, namely φ^{-1}. Sparse matrix operations can be implemented by simplified hardware and the determinant factor affecting the performance of the encoder becomes the $g \times g$ dense matrix φ^{-1}. The

parity-check matrix of many widely adopted LDPC codes has been specifically designed so that the parameter g is small, or in the case of DVB-S2 is zero and the matrix φ^{-1} has a special strucutre which results in efficient hardware implementation. For example, the φ matrix of the LDPC codes adopted for IEEE 802.11ac/n, 802.16e and many other applications is the $g \times g$ identity matrix.

For many other codes, the transformation into approximate lower triangular form, without affecting the QC structure of the matrix is not straightforward. For example, in the case of the CCSDS codes defined in [32], this can be achieved by shifting the last 4 circulants ($4m$ bits) by 8 columns ($8m$ bits) to the left. Since the last $4m$ bits of the code are punctured, this permutation does not affect the encoder's output. Fig.3.11 displays the H matrix before and after the transformation for rate $1/2$ AR4JA code with $k = 1024$. The parameter g is therefore $4m$ and φ^{-1} is a $4m \times 4m$ dense QC matrix of $m \times m$ circulants. Architectures proposed in [84], [127], [73], [139], [133], [71] are examples of application of the R-U method.

The work in [84] does not target QC codes, nor can it efficiently handle large dense φ matrices. The parameter g is 2 in the provided implementation examples and the resulting encoders occupy a large amount of the resources of Xilinx XC2V4000-6 FPGA, including a number of Block RAMS. The encoder architecture in [127] targets IEEE 802.11n codes where φ matrix is the identity matrix, but is not applicable to CCSDS codes. Targeting Wimax LDPC codes, [133] also assumes that φ is the identity matrix. In [73] the authors propose a code construction method, together with encoder-decoder architectures. The proposed encoder implements the R-U method, but the code construction aims at minimizing the parameter g. The dense matrix multiplication (3.13) involving φ in their case is executed on all elements of φ^{-1} in parallel, which obviously does not scale efficiently for large g. The method proposed in [71] and [139] employs SRAA modules introduced in [142] for the dense matrix operations of the RU algorithm. The scalability issues concerning the adoption of SRAA architectures for the direct encoding method, also pertain to the R-U method for CCSDS codes, because of the size of parameter g.

The R-U method is not applicable in the case of CCSDS C2 code: the structure of its H matrix is obviously not lower-triangular and at the same time, the matrix dimensions prohibit the matrix inversions in Eq. (3.12-3.14). In the case of AR4JA codes, however, after the transformation of its H matrix according to Fig. 3.11, the φ^{-1} matrix is a $4m \times 4m$ dense QC matrix, which complicates the calculations of (3.12).

3.3.3 Partitioned-H methods

This class of methods is based on the fact that for all systematic codes, the codeword c consists of the systematic part, which is a copy of the input information block and the parity bits: $c = \begin{bmatrix} s & p \end{bmatrix}$. The parity-check matrix can therefore be partitioned into a $(n-k) \times k$ submatrix H_1 and a $(n-k) \times (n-k)$ submatrix H_2, where $H = \begin{bmatrix} H_1 & H_2 \end{bmatrix}$, so that the parity bits vector can be calculated by (3.15), (3.16), (3.16).

$$Hc^T = H_1 s^T + H_2 p^T = 0 \tag{3.15}$$

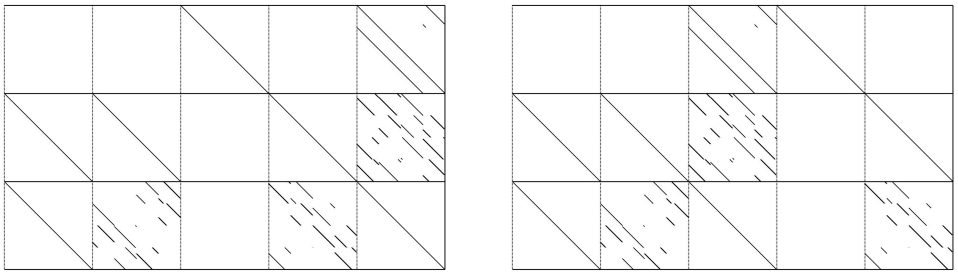

Figure 3.11: H matrix before (left) and after (right) the transformation into lower triangular form.

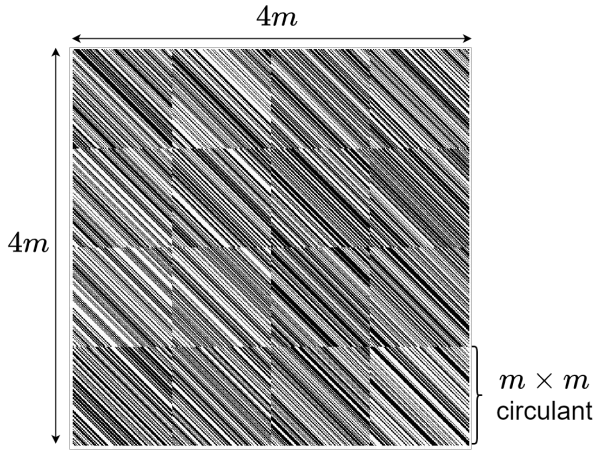

Figure 3.12: φ^{-1} submatrix for AR4JA codes. $m = 128$ bits for this example of the (2048,104) member.

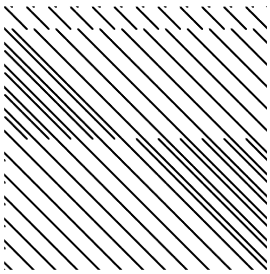

Figure 3.13: H_2^{-1} submatrix for the Wimax (2016,1008) code.

$$H_2 p^T = H_1 s^T \tag{3.16}$$

$$p^T = H_2^{-1} H_1 s^T \tag{3.17}$$

Submatrix H_1 is sparse and the vector $H_1 s^T$ can be easily calculated. For many practical codes, submatrix H_2^{-1} exhibits regular structure, which facilitates the involved calculations. A common structure in the parity-check matrix of many codes is the dual-diagonal: the rightmost part or of H_2, or even the entire submatrix (IEEE 802.11 n/ac, 3GPP2 DVB-S2) is a dual-diagonal matrix. For example, the H_2^{-1} matrix of the IEEE 802.16e (Wimax) LDPC codes exhibits a simple, regular structure displayed in Fig. 3.13. On the contrary, for CCSDS codes, as well as for most of the codes that do not have a dual-diagonal structure in the parity-check matrix, the H_2^{-1} matrix is a dense $4m \times 8m$ QC matrix (after omission of the columns corresponding to the punctured bits).

According to a variation of this method ([132], [80], [81], [77], [137], [134]), H_2 matrix is decomposed into a permutation matrix Π and two triangular matrices L, U, using triangular factorization (or LU decomposition). Equation (3.16) is therefore transformed into (3.19), from which parity bits are calculated using back-forward substitution. Conversely, the LU decomposition can be applied on H_2^{-1} matrix, so that parity bits are calculated according to (3.20),(3.21).

$$H_2 = \Pi^{-1}(LU) \tag{3.18}$$

$$L[U(p^T)] = \Pi(H_1 s^T) \tag{3.19}$$

$$H_2^{-1} = \Pi'^{-1}(L'U') \tag{3.20}$$

$$\Pi' p^T = L'[U'(H_1 s^T)] \tag{3.21}$$

The architectures proposed in [69], [14], [141], [74], [106] and [138] all follow algorithmically equivalent approaches which assume a dual-diagonal H_2 matrix. Parity bits can be calculated directly from the vector $H_1 s^T$ using backward substitution. In (3.17), H_2^{-1} is a lower triangular matrix and $H_2^{-1}(i,j) = 1, i \geq j$, and back substitution is applicable.

For another class of codes (for example in IEEE 802.16e), H_2^{-1} has the approximate dual-diagonal structure of (3.22), where $I_i^{(x_j)}$ are permutation matrices. Targeting these cases, [97], [83] and [41] propose encoders with similar algorithmical descriptions, which perform all the necessary permutations along with backward substitution for the calculation of the corresponding parity bits, directly from $H2$ matrix.

$$H_2 = \left[\begin{array}{c|ccccc} I_1^{(x_1)} & I & I & \dots & 0 & 0 \\ \vdots & & & \dots & & \\ I_b^{(x_b)} & 0 & 0 & \dots & I & I \end{array}\right] \tag{3.22}$$

The variation of the method based on L-U decomposition of H_2 according to (3.18)-(3.21) is used in [80], where the authors also propose an offline preprocessor for the triangulation of H_2. The QC structure of the matrix however is not kept in the decomposed matrices, at least for the demonstrated codes. Reference [134] proposes the same encoder architecture with a different decomposition algorithm for CMMB codes, which are not QC. The same encoder architecture is also proposed for CMMB in [132], without details on the decomposition algorithm. The work in [81] targets random Gallager codes. The encoding process is identical to [80], however algorithms are provided for the calculation of permutation matrices, which minimize the density of L, U components. It is shown that the compression achieved for the storage of sparse L, U matrices favors this method over R-U, at least for Gallager codes. The adoption of this encoding method however for CCSDS inflicts a major performance penalty because of the loss of QC structure in L, U matrices. For example, using the triangulation process outlined in [80], the AR4JA rate $1/2$ code with k=1024 bits calls for the storage, proper indexing and processing of a total of around 64K nonzero elements of L, U matrices, compared to the simple storage of the 2K elements of the first rows of the circulants needed for the $4m \times 4m$ φ^{-1} matrix.

The work in [90] modifies the procedure in [80] and adopts LU decomposition of H_2^{-1}, so that parity bits are calculated according to (3.21). It is shown that for the selected codes (Multi-level QC-LDPC codes), this method can result in more efficient storage of the decomposed matrices L', U' and Π' in the encoder's memory than H_2^{-1} or storing the components of H_2 according to [41]. However, the efficiency of the proposed encoding scheme is limited only to the two-step expanded multi-level codes described, which results in a cyclic structure in the triangulated components. Fig.3.14 depicts an example of the generated L' and U' matrices, compared to their CCSDS equivalents. The VMM architectures proposed for the vector-matrix multiplications are evidently not applicable for the random matrices of CCSDS. Memory requirements for indexing the non-zero values are also a prohibitive factor for AR4JA codes. Another fully parallel method is also proposed, targeting high encoding throughput. According to this method, there is no need to store L', U' in memory and the vector-matrix multiplications of (3.21) are executed in parallel on all bits. This method cannot scale for higher block lengths or other codes. Even for CCSDS AR4JA k=1024, rate $1/2$ code, it was discovered that the implementation of this method woukd require more than 72K LUTs, fitting only high-end Virtex 5 FPGAs, with prohibitive routing delays for any practical application.

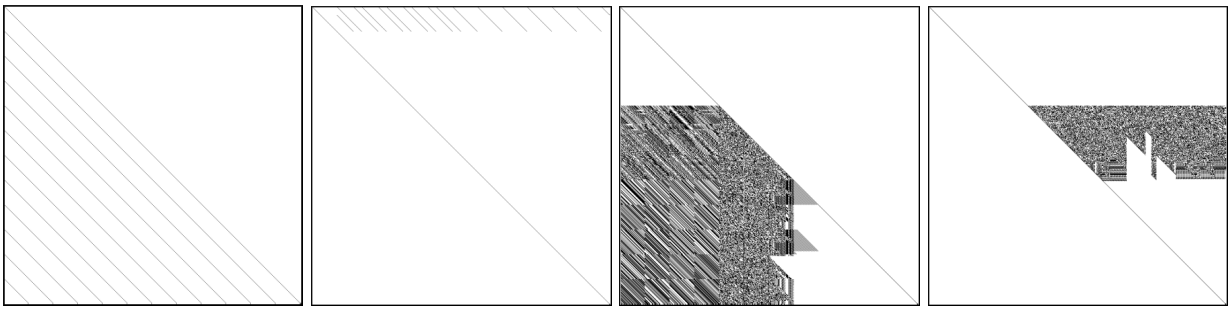

Figure 3.14: L', U' matrices of (2016,1008) example code in [90] (left) and AR4JA k=1024 rate $1/2$ code (right).

3.3.4 Hybrid method

In [47] the authors propose a hybrid approach, according to which the parity-check matrix is transformed in approximate lower-triangular form, as in R-U method. The parity bits are calculated using a mix of the direct and the R-U method: The first subvector p_1 of the g parity bits in R-U equation (3.13) is calculated from the the generator matrix G, according to the direct method. In this case, only the first g columns of the submatrix W_{n-k} in (3.7) need to be stored in encoder memory, avoiding thus the dense vector-matrix operations involving φ^{-1} of the pure R-U method. The work in [47] focuses in IEEE 802.11an codes. For CCSDS AR4JA codes however, $g = 4m$ and parameter r (i.e. the rows of circulants in the generator matrix) is 8, 16, 32 for rates $1/2$, $2/3$, $3/4$ correspondingly, while φ^{-1} is always $4m \times 4m$, for all codes. Pending the detailed performance analysis of the next section, it is evident that no performance gain is achievable from this method for any of the CCSDS codes. On the contrary, memory requirements and critical path are adversely affected from the larger dense matrix involved.

3.4 Description of the proposed basic architecture

In the general case, consider the multiplication $p = sW$ of the bit vector s with the dense QC matrix W, which is an $r \times c$ array of $m \times m$ circulants. Vectors s, p are also partitioned into a number of r and c sub-vectors correspondingly, each consisting of m bits, as shown below in (3.23) and (3.24). Bit j of sub-vector p_i is calculated by equation (3.25), where $W_{l,i}(j)$ is the j-th column of the circulant $W_{l,i}$.

$$s = \begin{bmatrix} s_1 & s_2 & \dots & s_r \end{bmatrix} \tag{3.23}$$

$$p = \begin{bmatrix} p_1 & p_2 & \dots & p_c \end{bmatrix} \tag{3.24}$$

$$p_i(j) = \sum_{l=1}^{r} s_l W_{l,i}(j) \quad (1 \leq i \leq c, 1 \leq j \leq m) \tag{3.25}$$

In a parallel implementation, a number of L input bits of vector s can be concurrently processed at each clock cycle. If parameter L is such that m is an integral multiple of L, equation (3.25) can be completed in mr/L cycles according to the following method.

We define the subvectors of s_l being processed in the current clock cycle as in equation (3.26). We also split the column vector $W_{l,i}(j)$ of equation (3.25) into m/L subvectors of L bits. Subvector $W_{l,i}^{(e)L}(j)$ is defined in (3.27), where $0 \leq e \leq \frac{m}{L} - 1$. In practice, since L bits of input vector p are processed, index e refers to one of the m/L execution cycles in which circulant $W_{l,i}$ is involved.

$$s_l^{(e)L} = \begin{bmatrix} s_l(eL+1) & s_l(eL+2) & \dots & s_l(eL+L) \end{bmatrix} \tag{3.26}$$

$$W_{l,i}^{(e)L}(j) = [\, W_{l,i}(eL+1, j) \quad W_{l,i}(eL+2, j) \dots \quad W_{l,i}(eL+L, j)\,]^T \tag{3.27}$$

Because sub-matrices $W_{i,j}$ are cyclic, equation (3.28) holds. To simplify notation, we define $W_{l,i}^{L}(j) = W_{l,i}^{(0)L}(j)$. Symbol $\overset{m}{\oplus}$ signifies modulo-m addition.

$$W_{l,i}^{(e+v)L}(j) = W_{l,i}^{(e)L}(j \overset{m}{\oplus} vL) \quad 0 \leq v \leq \frac{m}{L} - 1 \tag{3.28}$$

At each clock cycle, for each bit position j of subvector p_i, we calculate a partial result $\varrho_i^{(e)}(j,l)$ in parallel, according to (3.29). Note that $W_{l,i}^{L}(j)$ is independent of the execution cycle. This means that each $\varrho_i^{(e)}(j,l)$ depends only on the input bits and the current circulant index l.

$$\varrho_i^{(e)}(j,l) = s_l^{(e)L} W_{l,i}^{L}(j) \tag{3.29}$$

Each $p_i(j)$ can be calculated by accumulating the corresponding values of $\varrho_i^{(e)}(j,l)$, using shift registers. Equation (3.25) can be rewritten as in (3.30), if we take into account equations (3.28) and (3.29). The partial results which are calculated at each clock cycle e are accumulated into $p_i(j)$, as indicated by the internal summation in (3.30). The external summation updates the values of $W_{l,i}^{L}$ in (3.29) and the accumulation continues for all l.

$$p_i(j) = \sum_{l=1}^{r} \left(\sum_{e=0}^{\frac{m}{L}-1} \varrho_i^{(e)}(j \overset{m}{\oplus} eL, l) \right) \qquad \lfloor \frac{m}{L} \rfloor = 0 \tag{3.30}$$

In this work, we introduce an additional degree of parallelism. In every clock cycle (e), instead of processing L consecutive bits of s, we process L_m bits from each of the r subvectors s_i. We set $L_m = L/r$, so that the total number of bits being concurrently processed

are fixed to L. According to this scheme, at each execution cycle, instead of calculating and accumulating $\varrho_i^{(e)}(j,l)$, we calculate the partial sum $\delta_i^{(e)}(j)$ according to (3.31). The partial results are accumulated into $p_i(j)$ according to (3.32). The total number of bits being concurrently processed is again equal to L, but at each clock cycle, all circulants participate in the calculation of parity bits.

$$\delta_i^{(e)}(j) = \sum_{l=1}^{r} s_l^{(eL_m)} W_{l,i}^{L_m}(j) \quad 0 \leq e \leq \frac{m}{L_m} - 1 \tag{3.31}$$

$$p_i(j) = \sum_{e=0}^{\frac{m}{L_m}-1} \delta_i^{(e)}(j \overset{m}{\oplus} eL_m) \tag{3.32}$$

Note that in this case, $\delta_i^{(e)}(j)$ is independent of the current circulant, in contrast to $\varrho_i^{(e)}(j,l)$ in (3.29), where the dependence on l means that for the calculation of each $p_i(j)$, the corresponding $\varrho_i^{(e)}(j,l)$ is a logical function of $L + log_2(r)$ bits. Conversely, this means that each $W_{l,i}^{L}$ needs to be calculated from a function generator with $log_2(r)$ inputs. According to the introduced method, however, the value of $\delta_i^{(e)}(j)$ depends only on the rL_m bits of the input vector s.

The decoupling of $\delta_i^{(e)}(j)$ from l comes with a significant advantage for hardware implementation. If we define the truth function $\mathbb{1}$ as in (3.33), and set $w(\xi)$ as the ξ-th element of column vector $W_{l,i}^{L_m}(j \overset{m}{\oplus} eL_m)$, then (3.31) can be simplified as (3.34). This simplified statement of partial sums removes the need of a function generator for the elements of W and significantly reduces complexity.

$$\mathbb{1}[x=1] = \begin{cases} 1, x = 1 \\ 0, x \neq 1 \end{cases} \tag{3.33}$$

$$\delta_i^{(e)}(j) = \sum_{l=1}^{r} \left(\sum_{\xi=1}^{L_m} s_l^{(eL_m)}(\xi) \mathbb{1}\left[w(\xi) = 1\right] \right) \tag{3.34}$$

Consequently, for a hardware implementation of this algorithm, at each cycle we need to calculate vector $\delta^{(e)}$ according to (3.34) and accumulate this partial result into a register, according to the summation indicated by (3.32).

The accumulation of $\delta_i^{(e)}(j)$ in (3.32) can be efficiently implemented with a Linear Feedback Shift Register (LFSR), an abstract block diagram of which is presented in Fig. 3.15. The input feed is given by (3.34). After m/L_m processing cycles, registers $u_i(j)$ have accumulated the expected result, according to (3.32).

Table 3.2 lists the calculated resources and performance estimations. The input to each register of the LFSR is a function of the non-zero elements of column vector $W_{l,i}(j)$. For simplicity, we assume that $\sum_{j=1}^{m} W_{l,i}(j) = m/2$, which is a realistic approximation for all the dense QC matrices involved in CCSDS encoding operations, according to all encoding methods outlined in the next Section. Consequently, each $\delta_i^{(e)}(j)$ is a function of $rL_m/2$

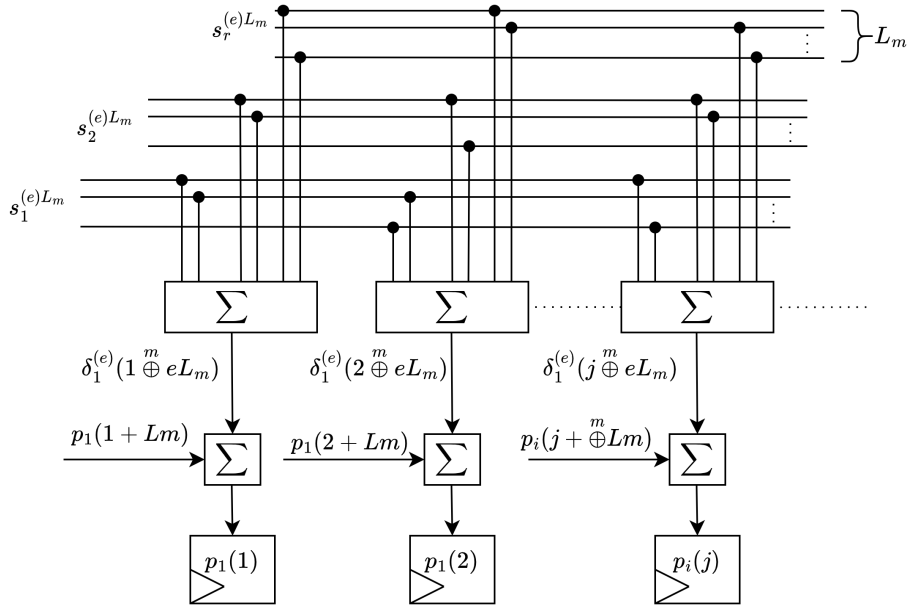

Figure 3.15: LFSR architecture for QC vector-matrix multiplication.

Table 3.2: Resource and Performance Estimations

2-input binary functions (2-input xor)	$cmrL_m/2$
Registers (FFs)	cm
Logic levels (2-input xor)	$\lceil log_2\left(rL_m/2\right)\rceil$
Encoding cycles	m/L_m

input bits. The accumulation between succesive execution cycles adds another variable to the sum of input parameters of each register, to total $rL_m/2+1$. The logical resources required for this operation, in terms of 2-input xor functions, are $rL_m/2$, for each $u_i(j)$. Given that the total number of register bits is cm, the total logic resources needed are those displayed in Table 3.2. According to the previous analysis, each LFSR input depends approximately on $rL_m/2+1$ bits. Consequently, a reasonable approximation for the 2-input logic levels required is $\lceil log_2\left(rL_m/2\right)\rceil$. The introduced architecture accomplishes the calculation of the cm bits of vector $p_i(j)$ in m/L_m cycles. Note that this value is independent of c and r.

3.5 Encoding architectures

The introduced architecture in Section 3.4 can be employed for every linear operation involving the multiplication of a bit vector with a dense QC matrix, even beyond the field of LDPC encoding. Such operations constitute key steps in each of the encoding methods presented in Section 3.3. In the current Section, the applicable encoding methods which

have already been described in Section 3.3 are evaluated for CCSDS codes, using the introduced architecture and compared to the applicable propositions in the literature. Based on the conclusions drawn, practical hardware implementations are presented in the next sub-section. Hardware budget is calculated on the basis of 2-input binary functions and memory bits.

3.5.1 Direct method encoder

The direct method implementation according to the proposed architecture is displayed in Fig. 3.16(a). The architecture introduced in the previous Section is used for the multiplication of $p = sW$. Additional logic is required in this case for rearranging the input data of vector s into $s_l^{(e)L_m}$, as required for the LFSR operation. This can be easily implemented by a series of r non symmetric Parallel In-Parallel Out (PIPO) shift registers. Input data are provided to the encoder L_{io} bits at each clock cycle and stored in one of these shift registers, depending on the current subvector s_i. During calculation (as e increases from 0 to $m/L_m - 1$), L_m bits of each subvector shift register are shifted out. Processing of each input frame is concluded in m/L_m cycles, consequently if $L_{io} = rL_m$, the input to the LFSR module never stagnates and maximum encoding throughput is achieved. Conversely, the entire input information block can be loaded into the array of the PIPO shift registers, at the same time.

Compared to our previous work in [121], the introduced architecture in the current work is algorithmically equivalent to the subcases of L_a=8, 16, 32 for rates 1/2, 2/3 and 4/5 correspondingly. The architecture implemented in [17], which is cited in and recommended by earlier versions of the CCSDS standard [32], processes input bits serially. However, a parallel implementation which processes L_{io} input bits at each clock cycle can be built as in Fig. 3.16(b) and compared to the proposed implementation of the direct method. Examples of resources and performance estimations are listed in Table 3.3 and compared to our proposed direct method encoder. For the same encoding cycles, the proposed architecture achieves shorter critical path and requires fewer combinatorial resources for rates 1/2 and 2/3, at the cost of additional Flip-Flops, which favours FPGA implementations. For rate 4/5, however, both architectures achieve the same encoding throughput performance, while the fixed overhead of the PIPO registers required for generation of vectors $s_l^{(e)L_m}$ dominates required resources. In an FPGA implementation, these registers can be substituted with Block RAM based FIFOs.

3.5.2 R-U method encoder

The equations (3.13), (3.14) involve many sparse matrices, but only a single $g \times g$ dense, namely φ^{-1}, which is the critical factor affecting the performance of the encoder. An example of the φ^{-1} matrix of AR4JA codes is provided in Fig. 3.12.

The R-U method implementation according to the proposed architecture is displayed in Fig. 3.17(a). The 4 L_m-bit vectors $v_i^{(e)L_m}$ are calculated by rotating input vectors s_i by

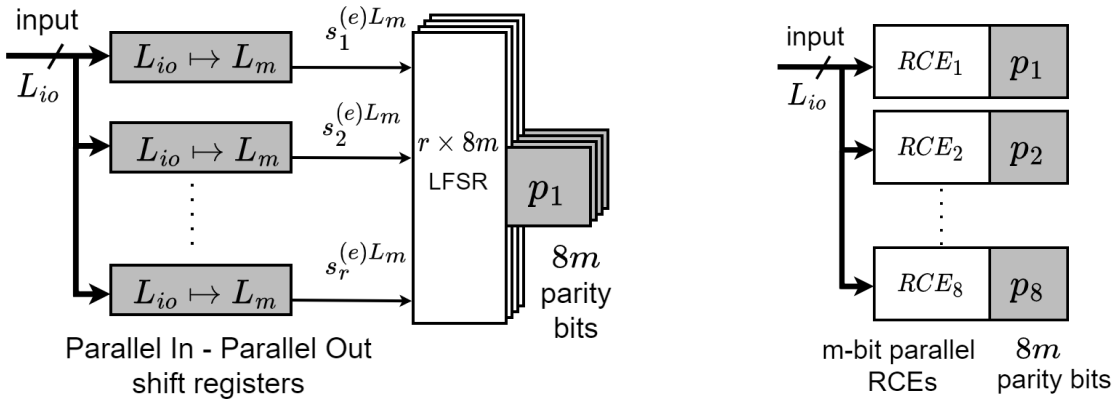

Figure 3.16: Proposed implementation according to the direct method (a) compared to a parallel implementation of [17] (b). $L_{io} = rL_m$

Table 3.3: Direct method estimations

Work	Rate	Logic functions	FFs	Logic levels	Enc. cycles
Proposed	1/2	8192χ+256	2048χ	3	64χ
Fig. 3.16(a)	2/3	8192χ+1024	1536χ	4	32χ
L_{io}=rL_m, L_m=2	4/5	8192χ+4096	1280χ	5	16χ
Based on [17]	1/2	19456χ	1024χ	5	64χ
Fig. 3.16(b)	2/3	10240χ	512χ		32χ
L_{io}=$2r$	4/5	5376χ	256χ		16χ

χ=1, 4, 16 for block lengths 1K, 4K, 16K correspondingly

Table 3.4: R-U method budget

Operation	Logic	FFs	Levels	Rate
Shift registers	16mK	8mK	1	ALL
v^T	$24L_m$	$4L_m$	3	$1/2$
	$72^a L_m$		5	$2/3$
	$168L_m$		6	$4/5$
f^T	$8mL_m$	4m	1+L^b	ALL
a^T	$4L_m$	$4m$	1	$1/2$
	$20L_m$		3	$2/3$
	$56L_m$		4	$4/5$
p^T	$16m$	$8m$	3	ALL
TOTAL	(32+8L_m)m+28L_m	24m+4L_m	Max.	$1/2$
	(48+8L_m)m+92L_m	32m+4L_m		$2/3$
	(80+8L_m)m+224L_m	48m+4L_m		$4/5$

[a]For $k = 1024$, subtract 2 from this value

[b]L=$\lceil log_2(L_m) \rceil$

L_m bits at each clock cycle. The hardware implementation of this particular operation is feasible, since matrix $EA + C$ is sparse, and each $v_i^{(e)L_m}(j)$ depends only on at most one bit of each s_i, since $(EA + C)$ is QC. Vectors $v_i^{(e)L_m}(j)$ are stored in intermediate registers, in order to reduce the critical path. After the m/L_m cycles required for the dense matrix multiplication, the $4m$ vector f^T is stored into the register of the LFSR. Vector b^T is calculated based on addition of the permutations of f^T. Since node v_1 is not connected to node c_1 in Fig. 3.2, it is evident that for all AR4JA codes, the first $4m$ bits of matrix A are zero. Vector a^T, is calculated in parallel with f^T and stored into the intermediate shift registers in the figure. At the final step of the process, the calculation of $b^T = Bf^T$ and addition to a^T are performed at the same clock cycle, to form the parity bits. The analytical detailed estimations of the resources consumption for this architecture are provided in Table. 3.4.

Examining the previous works implementing the R-U method and applicable to CCSDS codes, the architecture proposed in [73] for Block-LDPC codes is not expected to result in efficient implementations of CCSDS codes encoders. The resources and performance metrics are dominated by the dense matrix operation involving matrix φ^{-1}. A more suitable approach would be based on the architecture described in [71], which leverages the SRAA modules introduced in [142] for φ^{-1} multiplication. A diagram of a CCSDS encoder based on that architecture is displayed in Fig. 3.17(b), in which the layered approach is followed for sparse matrix operations. Table 3.5 summarizes the estimated resources and performance metrics of the two encoder architectures displayed in Fig. 3.17. Al-

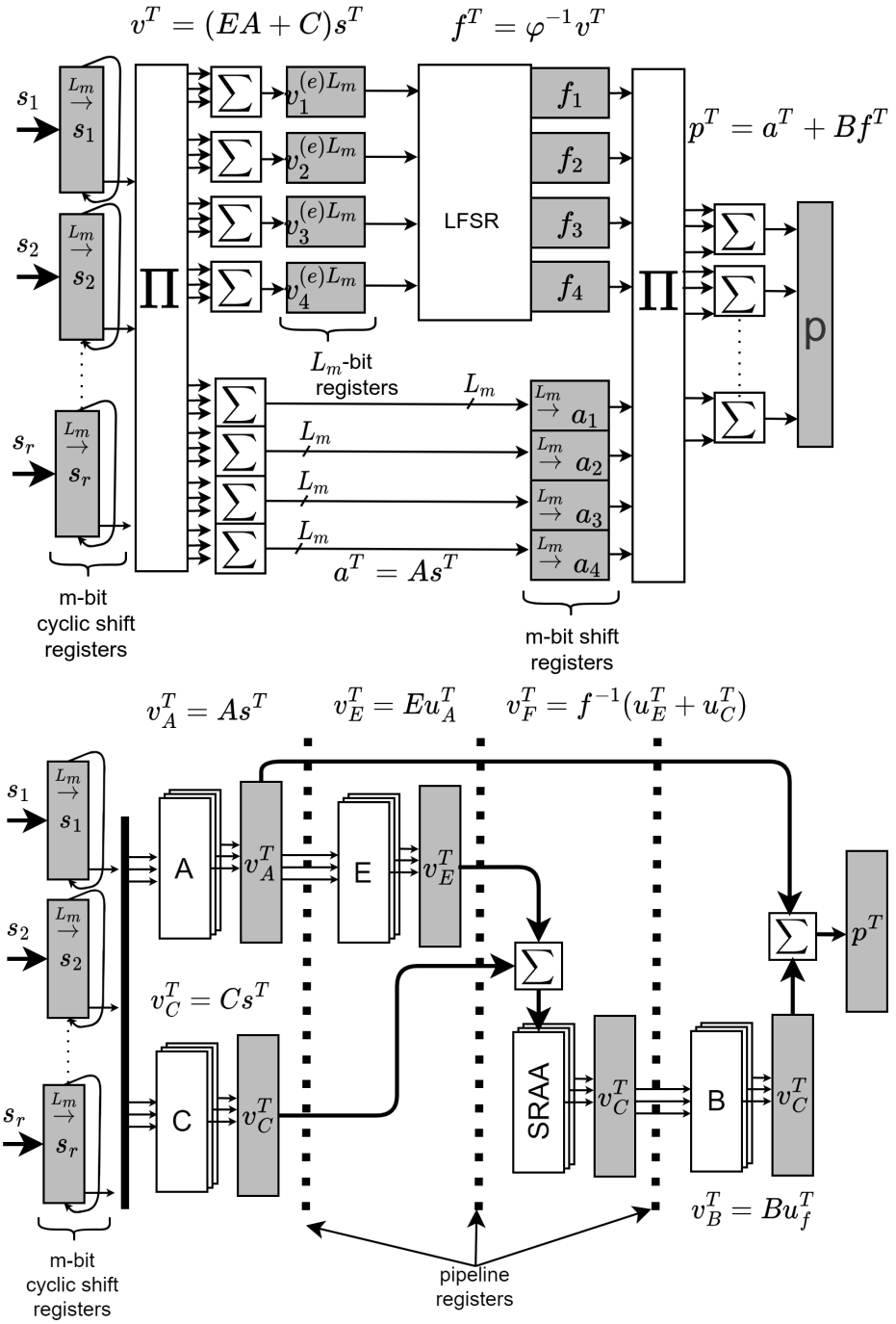

Figure 3.17: Proposed implementation according to the R-U method (top), compared to the classical approach taken in [71](bottom). All bits of vector s are considered to be available at the encoder's input.

Table 3.5: R-U method estimations

Work	Rate	Logic functions	FFs	Logic levels	Enc. cycles
Proposed	1/2	6144χ+56	3072χ+8	3	64χ
Fig. 3.17(a)	2/3	4096χ+184	2048χ+8	5	32χ
$L_m = 2$	4/5	3072χ+448	1536χ+8	6	16χ
Based	1/2	5120χ+26	8192χ	2	512χ
on [71]	2/3	3584χ+58	4608χ	3	256χ
Fig. 3.17(b)	4/5	2816χ+116	2816χ	4	128χ

χ=1, 4, 16 for block lengths 1K, 4K, 16K correspondingly

Table 3.6: Hybrid method estimations

Work	Rate	Logic functions	FFs	Logic levels	Enc. cycles
Proposed	1/2	8192χ+8	3072χ	3	64χ
Fig. 3.18	2/3	7168χ+40	2048χ	4	32χ
L_m=2, L_{io}=2r	4/5	6656χ+112	1536χ	5	16χ

χ=1, 4, 16 for block lengths 1K, 4K, 16K correspondingly

though the architecture of 3.17(b) requires slightly less combinatorial resources and has a smaller critical path than the proposed, it concludes encoding of one input frame in 8 times more cycles, since the operations on each layer and the SRAA modules are executed serially. Consequently, the introduced architecture achieves considerably increased encoding throughput performance, for the same resources.

3.5.3 Hybrid method encoder

A block diagram of an encoder implementing the hybrid method described in [47] is depicted in Fig.3.18. The $4m$ wide vector $f(T)$ is in this case calculated from the generator matrix, in an identical way as in the direct method. Intermediate vectors a^T, b^T and the final calculation of parity bits are calculated as in R-U method. Note that in this case there is the need for shift registers at the encoder's input for LFSR operation. Table 3.6 lists the resources for this method.

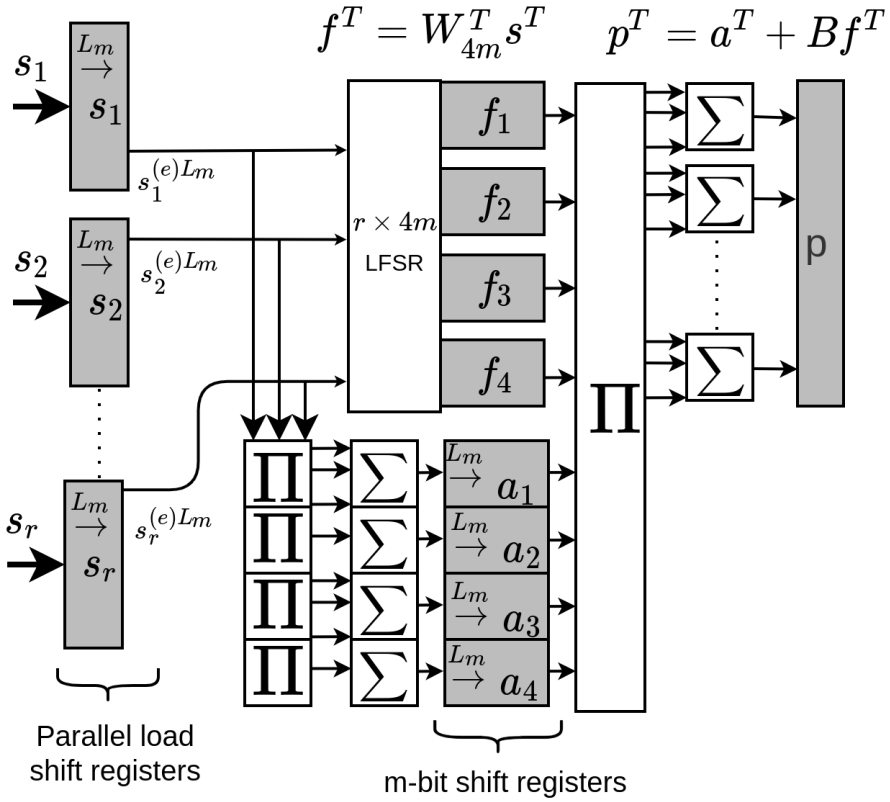

Figure 3.18: Hybrid method implementation ([47])

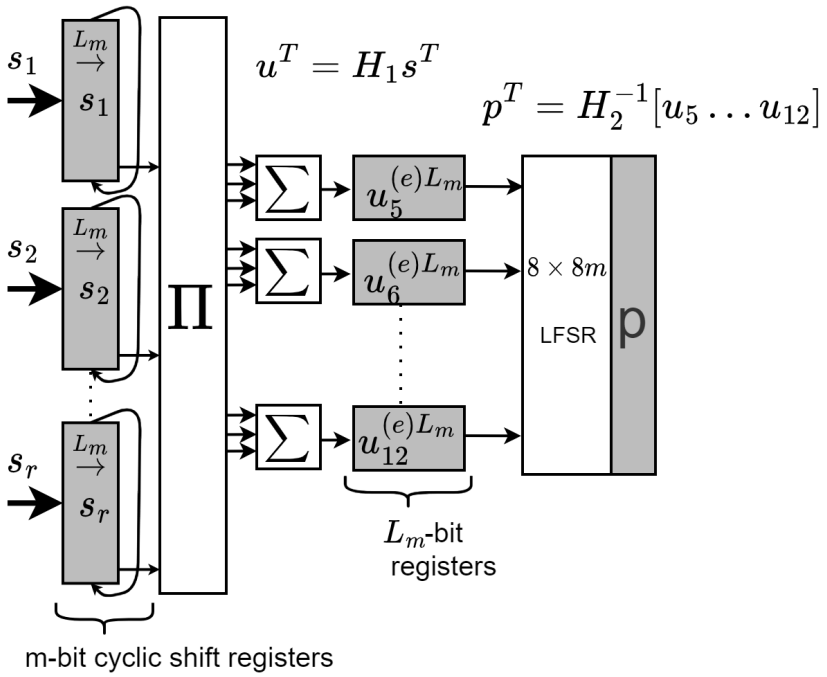

Figure 3.19: Proposed implementation according to the partitioned-H method. The architecture of Fig. 3.15 is used for dense matrix operation involving H_2^{-1}. All bits of vector s are considered to be available at the encoder's input.

3.5.4 Partitioned-H method encoder

The block-diagram of the proposed architecture implementing this method is displayed int Fig. 3.19. At each clock cycle, vectors $u_i^{(e)L_m}$ are calculated directly from input bits, in a way similar to $v_i^{(e)L_m}$ of the R-U method, and stored into L_m-bits registers. These intermediate results are provided to a LFSR, which implements the binary multiplication with H_2^{-1}. Calculation of p^T is concluded after m/L_m cycles.

Table 3.7 summarizes the involved estimations for the example case of $L_m = 2$. The existing implementations of the partitioned-H method, either directly or through triangular decomposition, require a specific structure of the parity check matrix, incompatible with AR4JA codes. Consequently, this is the first time that an encoder architecture for this method is proposed and thus no fair comparisons can be made.

3.5.5 Special case: C2 code

For the C2 code, the only applicable method is the direct, at least without any modification of the parity-check matrix. Special manipulation is required, however, because of the problematic circulant size of the code, which is not a power of 2. In our work in [121] we had introduced an efficient scheduling of the stream of input bits, by adding one zero bit

Table 3.7: Partitioned-H method estimations

Work	Rate	Logic functions	FFs	Logic levels	Enc. cycles
Proposed	1/2	10240χ+24	2048χ+16		64χ
Fig. 3.19	2/3	6144χ+88	1536χ+16	3	32χ
$L_m = 2$	4/5	4096χ+216	1280χ+16		16χ

χ=1, 4, 16 for block lengths 1K, 4K, 16K correspondingly

every 511 input bits. The effect of that addition was counterbalanced by performing one less rotation of the parity bits shift register. In the current work, we applied the architecture of Fig. 3.16 by setting $m = 511$, $r = 14$. A block diagram of the proposed encoder is displayed in Fig. 3.20(a). Input bits are supplied to the encoder in pieces of L_{io} bits and are padded with zeros as necessary, by setting $s_i^{(m/L_m)-1} = 0_{L_m}$, where 0_{L_m} is a sequence of L_m zeros. The effect of these extra $14L_m$ zeros added to each circulant can be balanced by a permutation of the calculated parity bits by L_m positions to the left.

In order to address the problematic circulant size of the code, the authors in [94] propose a packing-unpacking scheme, as displayed in Fig. 3.20(b). Input data are packed into groups of 21 bits, before they participate in the corresponding parity calculation. An architecture which is algorithmically equivalent to the Recursive Convolutional Encoders (RCEs) of [17] implements the cumulative parity calculations. At each clock cycle, the RCEs perform a shift or a shift-accumulate operation of 7,16 or 21 bits. In Fig. 3.20(b), we model the input of each RCE register (Flip-Flop) as a binary function generator of 30 parameters. Note also that because of the difference between the input bus size (16-bit) and the shift-accumulate step (21 or 7-bit), latent cycles are introduced in the parity generators' operation.

Table 3.8 compares the analytical estimations of the proposed architecture and the one based on [94]. The two architectures offer different trade-offs: Fig. 3.20(a) is optimized for combinatorial resources utilization and logic levels, while Fig. 3.20(b) for Flip-Flop use and it concludes encoding in approximately 12% less cycles. The efficiency of each architecture therefore depends on the targeted technology and the parameter which is needed to be optimized. Targeting high encoding throughput on FPGA technology, in which combinatorial logic is mapped to LUTs and Flip-Flops are an abundant resource, the hardware implementation of the proposed architecture achieves higher clock rate. This is because of the fewer logic levels and the lower routing delays imposed by the total footprint of the proposed encoder. The higher clock rate compensates for the increased number of cycles required for encoding, so that the actual encoding throughput of the proposed architecture is higher. In our measurements, targeting Virtex-5/XC5VLX110T-1 FPGA, we were able to achieve 30% higher clock rate with the proposed architecture, compared to that of Fig. 3.20.

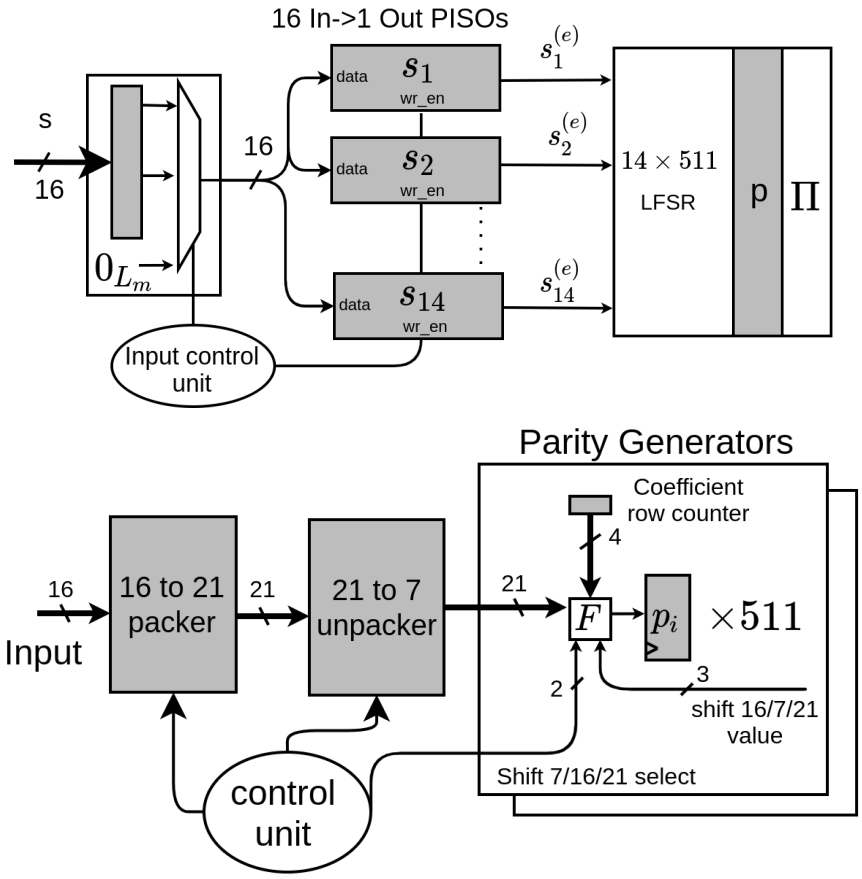

Figure 3.20: Proposed stream input implementation for C2 code (a), compared to the packing-unpacking scheme proposed in [94] (b).

Table 3.8: C2 code estimations

Work	Logic functions	FFs	Logic levels	Enc. cycles
Proposed (Fig. 3.20(a))	7602	8176	3	510
[94] (Fig. 3.20(b))	29638	1107	5	448

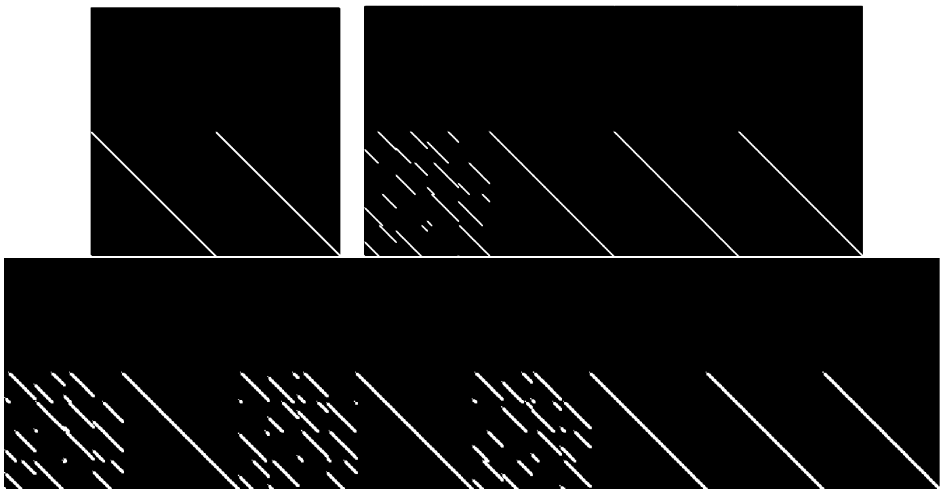

Figure 3.21: Top to bottom, left to right: A matrices of $k = 1024$ codes: R12, R23 and R45.

3.6 Implementation and results

In this Section, practical CCSDS encoder implementations, based on the architectures proposed in Section 3.5 are demonstrated. The different encoding methods offer various trade-offs between combinatorial logic and Flip-Flop usage and critical path logic levels. However, the following remarks pertain to all cases and can limit the selection of the optimal architecture for each rate:

- As already mentioned in Section 3.3, the Hybrid method does not offer any advantages over the other methods. Consequently, no implementations have been built, which are based on the Hybrid method.

- The R-U method is more favourable for lower rates, over partitioned-H. The dimensions of the φ^{-1} are always $4m \times 4m$, whereas the size of H_2^{-1} is $4m \times 8m$. However, the size of $EA + C$ is k bits, and the parameter m decreases by half for every rate increase (e.g. for k=1024, m is 128 for rate 1/2, 64 for 2/3, 32 for 4/5 and so on). This means that as the rate increases, the calculation of $v^T = (EA + C)s^T$ (refer to Fig. 3.17) is becoming increasingly more challenging. Conversely, the AR4JA protograph in Fig. 3.2 adds more weight-4 nodes as code increases. In practical implementations on the FPGA, the increase of the rate of k/m reaches a point where the synthesis and place & route results favour the partitioned-H method over the R-U, for rates higher than 1/2.

- The A matrix in R-U calculations for all the AR4JA codes has zeros in its first M lines. However, the A matrix of the rate 1/2 codes has the very interesting property of consisting of two $M \times M$ identity matrices, which greatly facilitates implementation. This property is not valid for the rest rates. Fig. 3.21 shows the structure of all the A matrices for $k = 1024$ codes.

Note also that the implementations of the R-U, partitioned-H and hybrid methods described in the previous Section require that all input bits are available at the encoder's input, while only the direct method can naturally encode a stream of input data. This assumption was made in order to provide direct comparisons of the encoding algorithms, without favouring any one of them, by virtue of the stream interface. Also, many references in the literature (albeit referring to other codes) make this assumption. Consequently, the architectures proposed in Section 3.5 form the theoretical basis for comparisons with any other architecture that adopt this assumption.

For practical implementations, however, and especially for the higher block lengths (4K and 16K), it is not realistic to assume that all the k bits of an information block are available at the same time. Consequently, the architectures of Section 3.5 were modified accordingly. Most importantly, the calculations in the different stages of the encoders needed to be balanced, so that optimal pipelined operation between several successive transfer frames is ensured. This involves the prudent selection of L_{io} (input/output bus width) and L_m of the various modules of the architectures.

For rate $1/2$ AR4JA codes, we developed a hardware implementation based on the R-U method, as shown in Fig. 3.22. Because of the structure of A matrix of the rate 1/2 codes, the calculation of $a^T = As^T$ is very easy and can be implemented using only two FIFOs, which sum the first $4m$ with the last $4m$ bits of the input information block. It is important to emphasize that this calculation can be performed in groups of L_{io} bits, and a large register of $4m$ bits for the entire vector a^T is not required. The calculation of u^T, which involves the sparse matrix $EA + C$ can be performed by a simple LFSR, according to the direct method: its combinatorial resources cost is low enough to not hinder the overall system performance. The calculation o the vector u^T needs k/L_{io} cycles to be complete, as is the time required for an entire input transfer frame to be stored into the systematic part's FIFO. Afterwards, u^T is loaded into a series of 4 parallel load shift registers, so that the calculation of the next transfer frame can continue. The dense matrix calculation involving φ^{-1} is executed as in Section 3.5. Vector $b = Bf^T$ is a summation of a low number of subvectors of f^T and can be easily calculated directly from f^T, without requiring an intermediate buffer. Finally, each successive L_{io} bits of b^T are added with the corresponding L_{io} bits of a^T, to give the corresponding parity bits. The fact that the parity bits are being calculated only in groups of L_{io} bits for many of the involved sparse matrix calculations, has an important result in the limitation of the required resources. Finally, a multiplexor at the encoder's output alternates between the 64-bit ASM sequence, the systematic bits and the parity bits. By selecting $L_m = 4$ and $L_{io} = 64$, the encoder operates at its maximum throughput, when an uninterrupted stream of successive input transfer frames is presented to its slave interface: all its modules are busy for the most part of the time in a pipelined fashion and no idle cycles are required at its master interface for parity calculations to complete. The pipelined operation of the encoder is shown on the timing diagram of 3.23, where successive frames are represented with different colors.

For rates 2/3 and 4/5 the partitioned-H method is the most preferable in therms of overall resources utilisation and encoding throughput performance. The modified architecture that we implemented for stream I/O is shown in Fig. 3.24. In this case, $u^T = H_1 s^T$ is

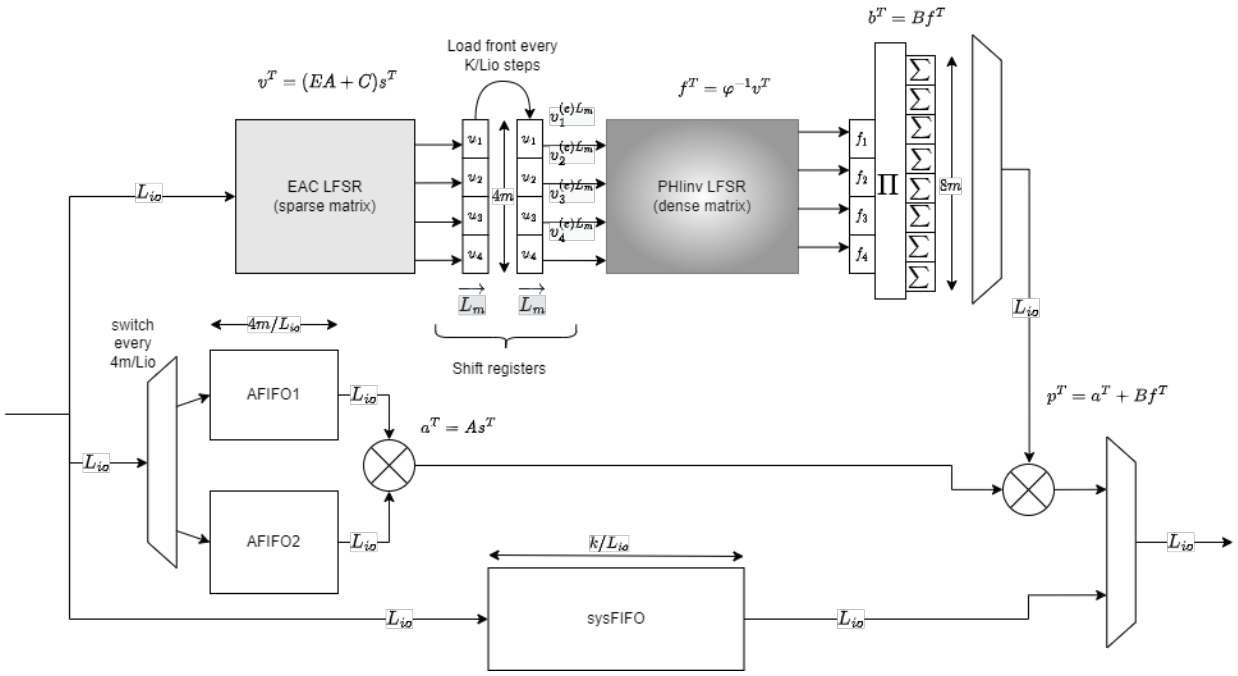

Figure 3.22: Proposed implementation according to the RU method with serial I/O.

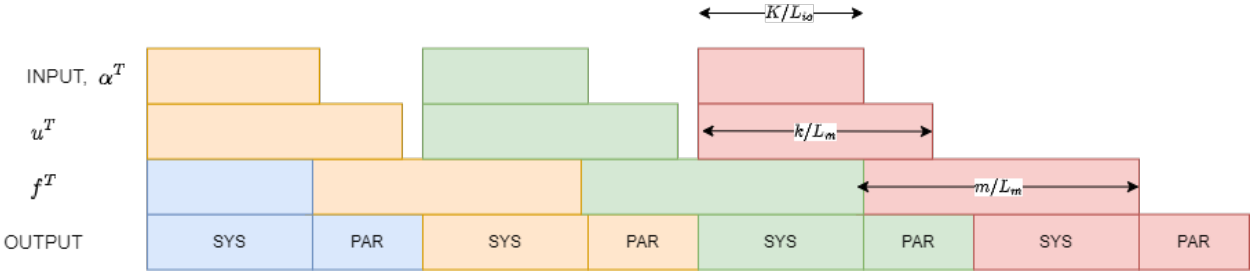

Figure 3.23: Pipelined operation over successive transfer frames for the R-U implementation. Each frame is represented with different color.

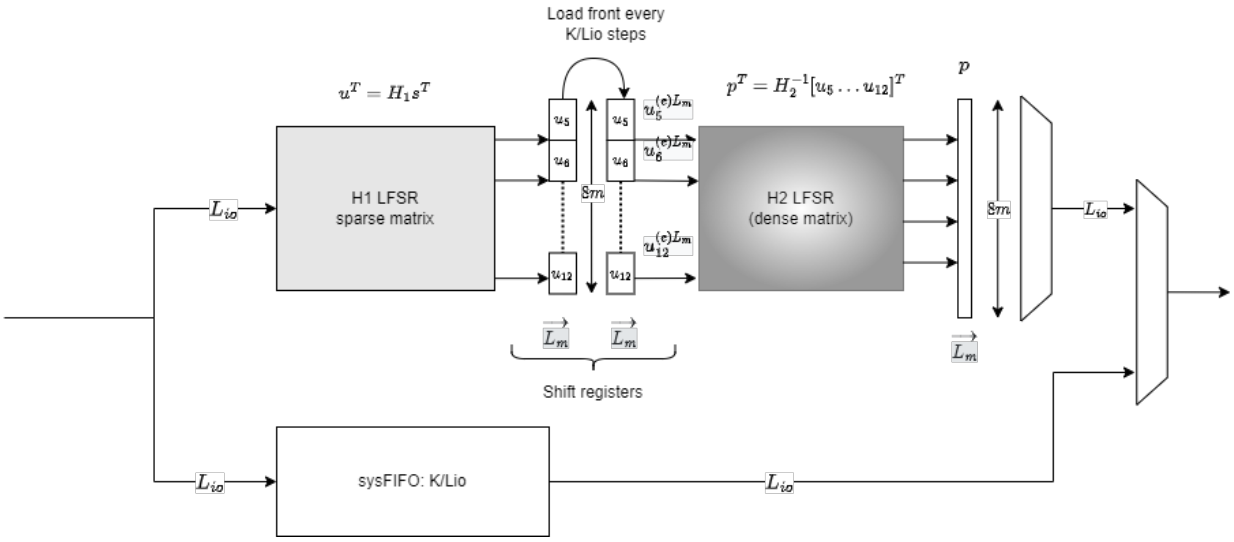

Figure 3.24: Proposed implementation according to the partitioned-H method with serial I.O.

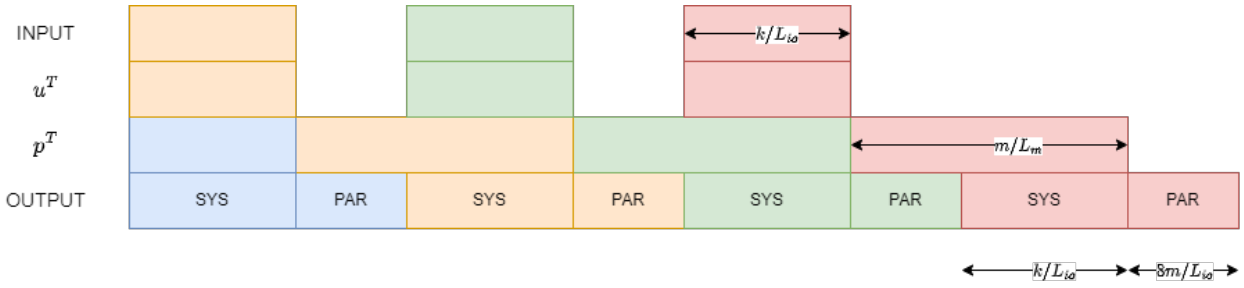

Figure 3.25: Pipelined operation over successive transfer frames for the partitioned-H implementation. Each frame is represented with different color.

calculated with a simple LFSR, according to the direct method, in the same way as u^T in Fig. 3.22. The calculation of u^T, however, takes only k/L_{io} cycles to complete, instead of k/L_m in the case of the R-U method. Afterwards, a parallel-load shift register loads the intermediate result, which is loaded to the front register that implements $u_j^{(e)L_m}$, until the previous calculation of p^T is completed. Like the R-U implementation, a multiplexor alternates between the 64-bit ASM sequence, the systematic bits and the parity bits. The pipelined operation of the encoder is shown on the timing diagram of 3.25.

The implementation results of the above stream architectures are shown in Table 3.9. For the C2 code, the results of the architecture of Fig. 3.20 are being displayed. These results are place and route results on the target device, and they were derived from Synopsys Synplify Premier (Synthesis) and Vivado 2022.1 (P&R). It is obvious that for the codes with $k = 1024, 4096$, the performance derived is mainly defined by the device's switching limits, whereas only the 16K and the C2 is affected by combinatorial paths.

Table 3.9: CCSDS-131.0 encoders implementation results on the KCU105 board

Rate	Codeword length (n)	Encoding Method	Resources			Max. clock (MHz)	Throughput (Gbps)
			LUTs	Flip-Flops	BRAMs		
1/2	2048	R-U	3053	2782	3	480	30,6
2/3	1536	Partitioned-H	3286	1992	1	490	31,4
4/5	1280	Partitioned-H	2494	1205	1	460	29,4
1/2	8192	R-U	8605	9013	3	480	30,7
2/3	6144	Partitioned-H	8685	6828	1	470	30
4/5	5120	Partitioned-H	6770	3670	1	470	30
1/2	32768	R-U	30823	33712	3	220	14
2/3	24576	Partitioned-H	25291	25287	1	240	15,3
4/5	20480	Partitioned-H	15835	12780	1	350	22,4
7/8 (C2)	8160	Direct	3287	1883	-	360	5.76

The full encoding throughput testing setup described in Section 3.7 was used to measure the power consumption of the implemented cores. A Xilinx SYSMON core was instantiated at the top-level entity of the testing design, and its recorded measurements were read from the board's system controller, through the integrated I2C bus of the board. Several power supplies support the operation of the FPGA. The supply rail that powers user logic (CLBs, BRAMS) is the VCCINT power supply, at 0,95V. The SYSMON core can monitor all the voltage and current rails of the device. For each implementation, two measurements were performed: one with the design loaded, but with the encoder in an inactive state (not encoding) and another one with the design running and encoding pseudo random frames at the maximum clock frequency. The difference in the recorded power between the two tests is the dynamic power and it corresponds to the switching activity. The measurements were averaged over a period of 2 minutes, with ovε measurement taking place every second.

Comparisons with the recent implementations targeting specifically AR4JA CCSDS codes are made in Table 3.11. The architectures of the current work were synthesized for the same FPGA device as the corresponding references. Compared to our previous work in [121], the efficiency of the introduced architecture results in almost five times higher encoding throughput, with similar resources. The large XOR operations over 2048 bits, which are introduced by the architecture in [140] result in a large amount of required resources and poor timing, compared to the implementation proposed in this work. Finally, to the best of our knowledge, there is no other documented implementation of 16K AR4JA CCSDS codes, other than the implementation by the NASA/JPL which has been presented in the CCSDS Optical Communications Working Group monthly meeting in 23 June 2020. In that meeting, the discussion revolved around the possible use of the AR4JA codes for the oncoming O3K standard for free-space optical communications. Implementation complexity was a major concern in this case, since the O3K is supposed to become a simple,

Table 3.10: Power measurements on the KCU105 board

Rate	Codeword length (n)	Clock (MHz)	Static Power (W)		Operating Power (W)		Dynamic Power (W)	Power Efficiency Mbps/mW*
			VCCINT	Total	VCCINT	Total		
1/2	2048	480	0,259	0,887	0,507	1,18	0,294	25,9
2/3	1536	490	0,199	0,797	0,53	1,147	0,35	27,3
4/5	1280	460	0,168	0761	0,408	1,008	0,247	29,2
1/2	8192	480	0,323	0,927	1,361	1,974	1,047	15,6
2/3	6144	470	0,263	0,842	1,247	1,832	0,99	16,4
4/5	5120	470	0,218	0,798	0,785	1,371	0,574	21,9
1/2	32768	220	0,398	0,979	2,204	2,807	1,828	5
2/3	24576	240	0,312	0,92	2,511	3,097	2,178	4,9
4/5	20480	350	0,298	0,914	1,363	1,962	1,048	11,4
7/8 (C2)	8160	360	0,177	0,748	0,384	0,951	0,203	6,1

*Total operating power is considered for the calculations

Table 3.11: AR4JA implementation comparisons (synthesized design) with previous work in the literature

Rate	k	Work	Encoding Method	FPGA Device	Resources			Clock (MHz)	Throughput (Gbps)
					LUTs	Flip-Flops	BRAMs		
1/2	1024	[121]	Direct	XC5VLX110T-1	3258	2176	-	230	3.59
		This	R-U		3024	2741	3	280	17,9
4/5	4096	[140]	Direct	XC7VX485T-1	101173	141411	-	200	8
		This	Part-H		6862	3724	3	320	20,5

low-cost option. The comparison of the architectures implemented by the JPL with the ones presented here are displayed in Table 3.11. Although the footprint of the JPL encoders is 2-3 times lower than the proposed, their throughput is lower by 4-7 times an order of magnitude. More importantly, it is mentioned that a multi-core implementation is required, in order to achieve a 10 Gbps performance, which is required for the specific optical application. Such a solution would occupy up to 80% of a XQRKU060 FPGA, without taking into account the additional design complexity which would be introduced by the management of multiple parallel cores operating on a single stream of data to be encoded, as well as the routing delays that would emerge in such a congested design. It is obvious that the proposed architectures can easily achieve this performance requirement, with a small fraction of the FPGA resources.

Table 3.13 compares the encoder based on Fig. 3.20(a) with prior work targeting CCSDS C2 code. Contrary to the rest entries in the Table, the proposed encoder implements all the functions of the TM-SDLP protocol (synchronisation and randomisation). The PISO registers in Fig. 3.20(a) are mapped to LUT RAM, hence the difference in Flip Flop count between estimations in Table 3.8 and actual count in Table 3.13. Our previous work in [121] implemented the direct encoding method, based on a different scheduling of the in-

Table 3.12: AR4JA 16K implementation comparisons (implemented design) with NASA/JPL on the Virtex UltraScale+ XCVU9PFLGA2104-2L FPGA

Rate	Work	Resources						Clock	Throughput
		LUTs	%	Flip-Flops	%	BRAMs	%	(MHz)	(Gbps)
1/2	JPL	16360	4,15%	16665	2,11%	0	0%	500	0,5
	This	35707	9,06%	34050	4,32%	3	0,42%	320	20,5
2/3	JPL	10988	2,79%	8245	1,05%	0	0%	500	0,5
	This	22789	5,78%	25334	3,21%	1	0,14%	480	30,7
4/5	JPL	6764	1,72%	4155	0,53%	0	0%	467	0,5
	This	14857	3,77%	12956	1,64%	1	0,14%	590	37,8

put bits. Instead of the PISO registers of Fig. 3.20(a), it was based on an ping-pong buffer at the encoder input, which handled the boundaries between the 511-bit circulants. Compared to that work, the architectural optimizations of the current work result in an increase in encoding throughput, while at the same time minimizing the required resources. The work in [109] also implements the direct encoding method. It buffers an entire input frame and partitions it into 14 sub-vectors of 511 bits each. All parity bits are being calculated in parallel. It requires, however, significantly more resources than the encoder of the current work, while at the same time achieving lower throughput performance, even on a more advanced Kintex-7 FPGA. Prior works listed in [4, 7, 6, 9] refers to commercial products, for which limited information is available. The encoder in [4] has 8-bit input-output interfaces and implements the direct encoding method. It stores two circulant tables: one for processing input bits which correspond to the same 511-bit circulant of the generator matrix and another for the cases when the 8 input bits span two circulants. This implementation requires a large amount of resources. A low complexity and low throughput encoder is provided in [6]. It implements the direct encoding method and the input-output buses are bit-serial. Another encoder for C2 code is provided in [9]. It also implements the direct encoding method and input-output bus is 8 bits. Block RAM is used for input-output buffering. Finally, for [7], no information about the underlying architecture is provided other than what is displayed in Table 3.13. Implementation results on the space-grade Virtex5-based XUPV5 development board are provided, for easier comparisons. Like in the case of the AR4JA codes, of particular interest is the comparison with the NASA/JPL implementation, which target an ultrascale+ FPGA. Our implementation on the same device yields more than 15 times higher performance, for 2-3 times the required resources, which are in any case a very small percentage (less than 0,8% and 0,2% for LUTs and FlipFlops correspondingly).

Table 3.13: C2 implementation comparisons (synthesized design)

Work	Targeted FPGA technology/ Demonstrated device	Resources			Clock (MHz)	Throughput (Gbps)
		LUTs	Flip-Flops	BRAMs		
[121]	Virtex-5/XC5VLX110T-1	9128	1156	0	200	3,12
[109]	Kintex-7/XC7325T	54747	92233	38	297	2.97
[4]	Virtex-5/XC5VLX30-1	9.2K	N/A	N/A	164	1.14
[6]	Virtex-6/XC6VLX240T-1	N/A*	N/A*	0	418	0,418
[9]	Artix-7/100T-1	6873	3219	1	239	1,55
[7]	Various FPGA/ASIC	N/A	N/A	0	200	1,6
Proposed	Virtex-5/XC5VLX110T-1	3338	1340	0	260	4,16
NASA/JPL	Virtex UltraScale+/	1297	1042	0	600	0,66
Proposed	XQVU3P-FFRC1517-2-i	3336	1906	0	700	11,2

*The design takes up 290 slices. One Virtex-6 slice contains 4 LUTs and 8 Flip-Flops.

3.7 Testing

The introduced architectures have been extensively tested to ensure compliance with the standard. A bit-accurate golden model has been built with GNU Octave, which produced a number of test vectors and the corresponding expected results for all the codes of the standard, including the randomization option and the synchronisation sequence (ASM). These data were produced and stored off-line as separate files on disk. These test source data were then handled by a testbench which supplied them to the encoder and received the generated responses in three phases, in order to test marginal conditions in the code and ensure 100% code coverage in all cases:

1. Full throttle operation: During this phase, the testbench provided an uninterrupted flow of data to the encoder and the receiver was assumed to be always able to accept encoder's generated output. No idle cycles on the encoder's master interface were experienced here: an assert statement would immediately raise an error in this case. The purpose of this phase was to validate the operation of the encoder at full speed.

2. A series of successive reset signals were afterwards sent to the encoder, in order to verify the correct behaviour of the code during all its FSM transitions. The timing of the reset assertions was determined with the help of modelsim's signal spy package.

3. Specific amounts of input data were sent to the encoder, immediately followed by invalid (TVALID de-asserted) cycles, while the encoder's output was halted (TREADY on the encoder's master interface is de-asserted). This phase excited certain FSM transitions, which are generally related to the FIFOS in Fig. 3.22, 3.24 and 3.16 becoming full when the next parity calculation is complete.

4. Channel interface validation. The flow of data to and from the encoder was not constant in this phase, like in phase 1. The sender to the encoder and the receiver paused operation randomly through the corresponding TVALID and TREADY signals on the input and output interfaces respectively. The rules that govern their behavior, however, were consistent with the AXI4-Stream protocol (i.e. once TVALID is asserted, it must remain high until the handshake occurs). The purpose of this phase was to verify protocol operation on input and output interfaces.

On the completion of the testbench, the recorded responses from the encoder were compared to the expected values from the Octave golden model and any discrepancies were reported.

Apart from the software simulations of the encoder's RTL description, the implemented architectures were also tested at-speed on the target hardware. Two tests setups have been implemented: one that tests the encoder at full throughput operation and another one that is based on a SpaceFibre channel with a host PC, according to Fig. 2.9. Both environments are described below.

In the full throttle scenario, the encoders were connected to a 64-bit LFSR, which provided them with pseudo-random data to be encoded. The generated responses were compressed with a 64-bit Multiple-Input Shift Register. Both these components were operating in the same clock domain and were working at 100% duty cycle: once started through an AXI4-Lite configuration interface, the LFSR continued to provide an uninterrupted stream of input data. At the same time, the MISR had always is TREADY signal asserted. After a specified amount of cycles, which was defined through the LFSR's configuration interface, the LFSR's operation was halted and the MISR value was read and compared against the expected value of the Octave golden model.The performance of the streaming interfaces of the core, as well as their compliance to the AXI4-Stream specification were monitored through an AXI performance monitor core from Xilinx. All read and write operations were executed over the JTAG interface of the FPGA, through a JTAG to AXI master core and an AXI smart connect core, also from Xilinx.

The test environment with SpaceFibre core integration is shown in Fig. 3.27, which describes in detail the user logic part of Fig. 2.9. The latter gives the broader pictorial description of the overarching testing environment. Again, test data were provided through the SpaceFibre interface to the encoder and the generated responses were captured and compared aginst the golden model data. The tests were performed using the two lanes of the SpaceFibre link between the KCU105 board and the Dundee PC, providing 10Gbps of user bandwidth on each direction. The link statistics over the SpFi link are shown in Fig. 3.28, for an example case of a rate 1/2 code. It is apparent that the downlink connection was saturated by the encoder core.

Finally, in an implementation of the full throttle test environment on the XUPV5-LX110T prototyping board, during the validation of the direct method's implementation in [121], all the rates of the $k = 1024$ codes where also tested on-board sequentially, using the partial reconfiguration feature of the Xilinx Virtex-5 XC5VLX110T FPGA.

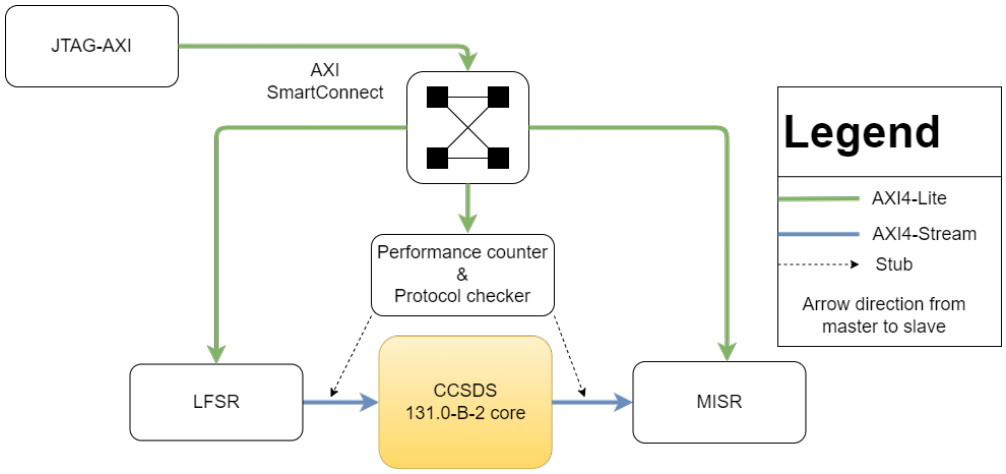

Figure 3.26: Full throughput testing environment.

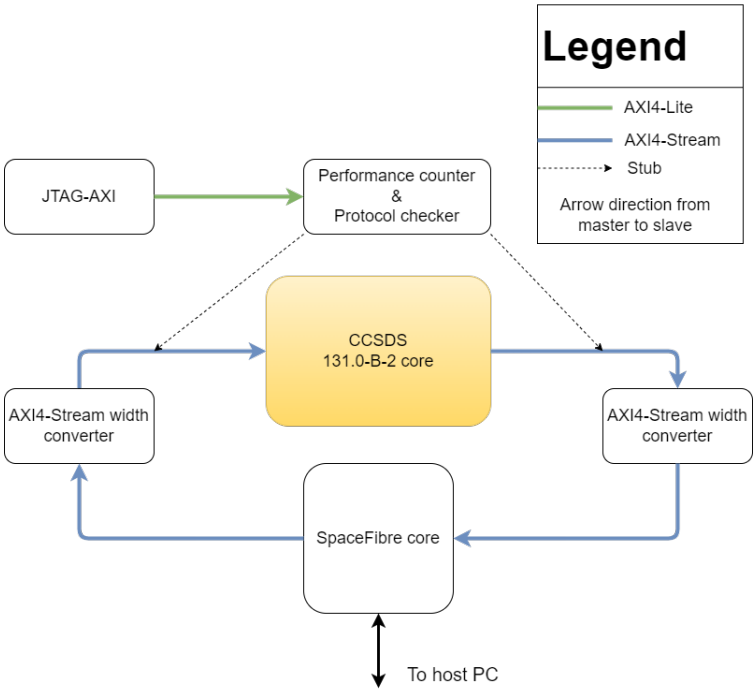

Figure 3.27: User logic overview of the SpaceFibre test environment.

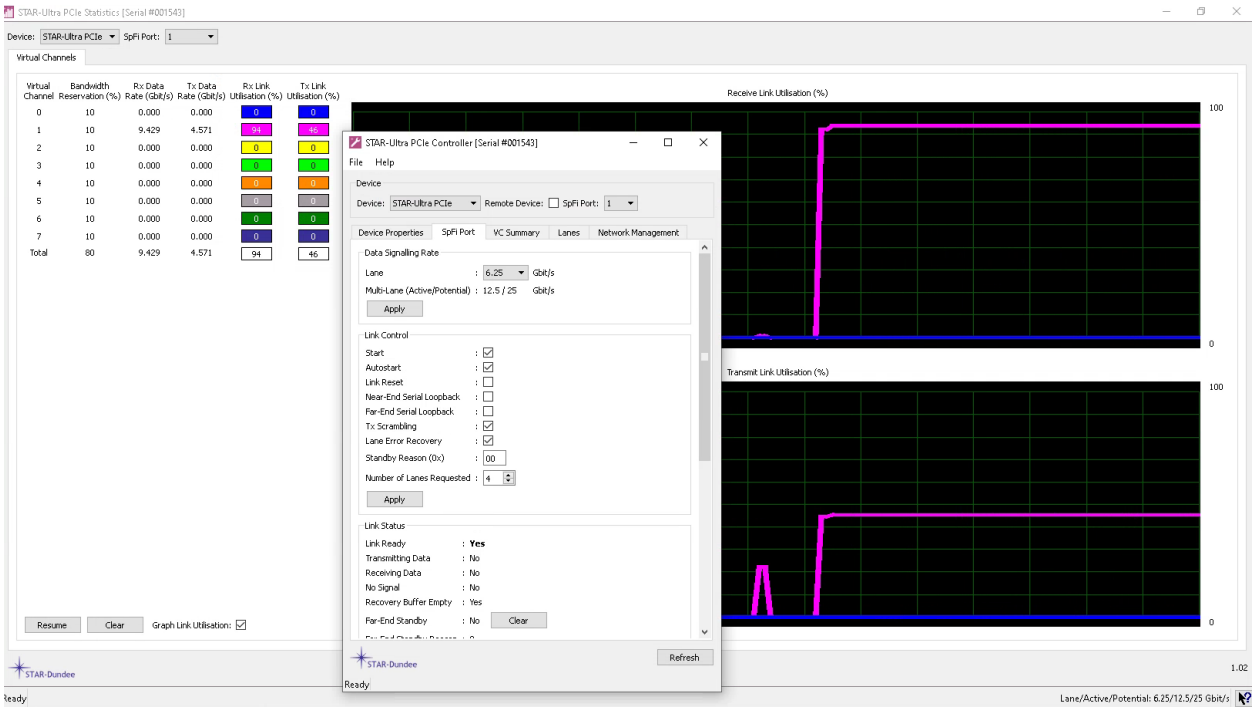

Figure 3.28:

3.8 Special topic: QC encoding for magnetic media recording

Interestingly, a special application of the encoders for AR4JA had been found to be in the domain of magnetic media recording. QC-LDPC codes are widely considered an advantageous FEC option for magnetic recording (MR) media. The vast majority of related research, however, has so far been focused on the analytical optimization of code design and algorithms. Although high-speed encoding and decoding with low hardware footprint are important for MR media, hardware implementations for such encoding schemes have so far been scarce. Among the proposed LDPC code variants, protograph-based codes are a promising option, because of their excellent performance characteristics and efficient implementation. In the work in [124], we leveraged the architecture of the work on LDPC encoders for space applications and we proposed efficient encoder designs for the protograph-based LDPC codes proposed so far for MR media. The proposed designs were also implemented in hardware as FPGA accelerators. The efficiency of the introduced architectures was demonstrated on the ZC706 FPGA development board, achieving multi-Gbps encoding throughput, adequate for modern MR application standards.

Protograph LDPC codes for MR media have been proposed in [61] and [60], [39], [128]. In the following subsections, these codes are briefly described and encoder designs for each one of them are proposed. The notation of Section 3.1 is maintained, where applicable.

3.8.1 IARA

In [60], the authors propose a series of three punctured protograph LDPC codes for use in the partial response channel. Puncturing is a technique to maintain the desired code rate by omitting (i.e. not transmitting) some of the parity bits. The authors call these codes *Improved Accumulate-Repeat-Accumulate* or IARA codes. It is shown that they exhibit lower decoding thresholds, improved convergence speed and overall better BER performance than conventional protograph LDPC codes for the AWGN channel. In particular, the comparison is made against AR3A, AR4JA and a regular LDPC code with column weight 3 in the dicode and EPR4 channels. Furthermore, the performance of one variant (rate 1/2 IARA2) of these codes over the ergodic Nakagami fading channels is examined in [58].

The protographs for IARA codes and the corresponding parity-check matrices for rate $4/5$ are displayed in Fig. 3.29. The parity check matrices of rate 4/5 of IARA codes are also displayed in Fig. 3.29 as an example. I_M and 0_M are the $M \times M$ identity and zero matrix correspondingly, where $M = xm$ (x is the first protograph expansion step, according to the notation established in Section 3.1). Each variable node, after protograph expansion, corresponds to one column of the parity check matrix. In the figure, node u_1 corresponds to the rightmost column and the rest columns are numbered successively from right to left. Check nodes correspond to the rows of the parity check matrix and they are numbered from the top to the bottom of the matrix (c_1 corresponds to the first row). Parity bits are derived from the expansion of nodes v_1, v_2, v_3. Node v_1, however, is a punctured node: the parity bits generated from this node are not transmitted. A submatrix P_M^i is a $x \times x$ array of either $m \times m$ permutation matrices, or $m \times m$ zero matrices. The number of permutation matrices on each row or column of P_M^i is i. The exact position of the permutation matrices in the array, as well as the permutation values, are defined by the selected protograph expansion algorithm. Note also that the protograph of IARA3 code is almost identical to that of AR4JA code, with the selection of the punctured node being the only difference between them.

$$H_{2(IARA)}^{-1} = \begin{bmatrix} 0_M & W_m(1,1) & \dots & W_m(1,2x) \\ 0_M & \vdots & \ddots & \vdots \\ I_M & W_m(3x,1) & \dots & W_m(3x,2x) \end{bmatrix} \tag{3.35}$$

Submatrix H_1 corresponds to the expanded protograph nodes u_4 and higher. The particular numbering of protograph nodes in Fig. 3.29 results in an efficient structure of matrix H_2^{-1}, which is given in (3.35). It is composed of the $M \times M$ zero and identity matrices, 0_M and I_M respectively, and an $3x \times 2x$ array of juxtaposed $m \times m$ dense circulants, $W_m(i,j)$. Specifically for the IARA3 code, $W_m(i,j) = 0_m, i > 2w$. Note also that the last M bits of the encoded codeword are punctured, consequently the last M rows of H_2^{-1} can in any case be substituted with zeros.

Leveraging the structure of H_2^{-1} matrix, we can introduce an efficient encoding scheme,

based on the partitioned-H encoding method. The first x bits of intermediate vector u^T can be omitted because the first M columns of H_2^{-1} are zeros (bits corresponding to I_M in (3.35) are punctured. This leads to a significant simplification of the encoder architecture. If submatrix W is the $3x \times 2x$ array of W_m circulants, parity bits can be calculated by multiplication of the last $2xm$ bits of intermediate vector u^T with W.

The proposed encoder design for IARA codes is displayed in Fig. 3.30 and is actually similar to the corresponding AR4JA architecture in Fig. 3.19. Input bits are initially stored into a series of $r \times m$-bit cyclic shift registers, where $r = k/m$. At each clock cycle, these registers are rotated cyclically by L_m bit positions. The permutation and summation modules calculate L_m of totally m bits of each subvector $u_i, 1 \leq i \leq 3x$ at the same clock cycle. Subvector $u_i^{(e)L_m}$ includes the L_m bits of u_i during the e-th execution cycle. Obviously, $1 \leq m/L_m \leq 3$. These $3xL_m$ bits are fed to the LFSR structure we introduced in Section 3.4 for the efficient multiplication of a vector with a QC matrix. After $m/L_m + 1$ cycles, the calculation of the $2xm$ parity bits will have been concluded.

3.8.2 2-D-P1 and 2-D-P2

The authors in [39] examine the performance of protograph LDPC codes over the 2-D Inter-Symbol Interference (ISI) channel. Based on Extrinsic Information Chart (EXIT) analysis [120], two types of codes are constructed: 2-D-P1 and 2-D-P2 codes. The proposed codes have better decoding thresholds, lower floors and better error performance in 2-D ISI channel than conventional protograph AR4JA and AR3A codes, which are optimized for the AWGN channel.

The protograph for these codes is given in Fig. 3.31. We propose a reordering of protograph nodes, as described in the figure, which results in an efficient structure of H_2^{-1}. This structure is the same for 2-D-P1 and 2-D-P2 and it is described in (3.36). The definition of matrices I_M, 0_M and $W_m(i,j)$ is the same as the previous paragraph. Because of the structure of H_2^{-1}, the last xm parity bits are equal to the first m bits of the intermediate vector u^T. If W is the $2x \times 3x$ array of $W_m(i,j)$ circulants, $p = \begin{bmatrix} p_1 & p_2 \end{bmatrix}$ is the parity bits vector, parity calculation is summarized in (3.37).

$$H_{2(2DP)}^{-1} = \begin{bmatrix} W_m(1,1) & \dots & W_m(1,3x) \\ \vdots & \ddots & \vdots \\ W_m(2x,1) & \dots & W_m(2x,3x) \\ I_M & 0_M & 0_M \end{bmatrix} \tag{3.36}$$

$$\begin{aligned} p_1^T &= Wu^T \\ p_2^T &= [u_0 \dots u_{xm-1}]^T \end{aligned} \tag{3.37}$$

The proposed encoding design for these codes is displayed in Fig. 3.32 and it will be shown later in the current Section that it can been also used for other codes, with adjustment of the size of the LFSR (parameter f in the figure). The calculation of p_1^T is performed

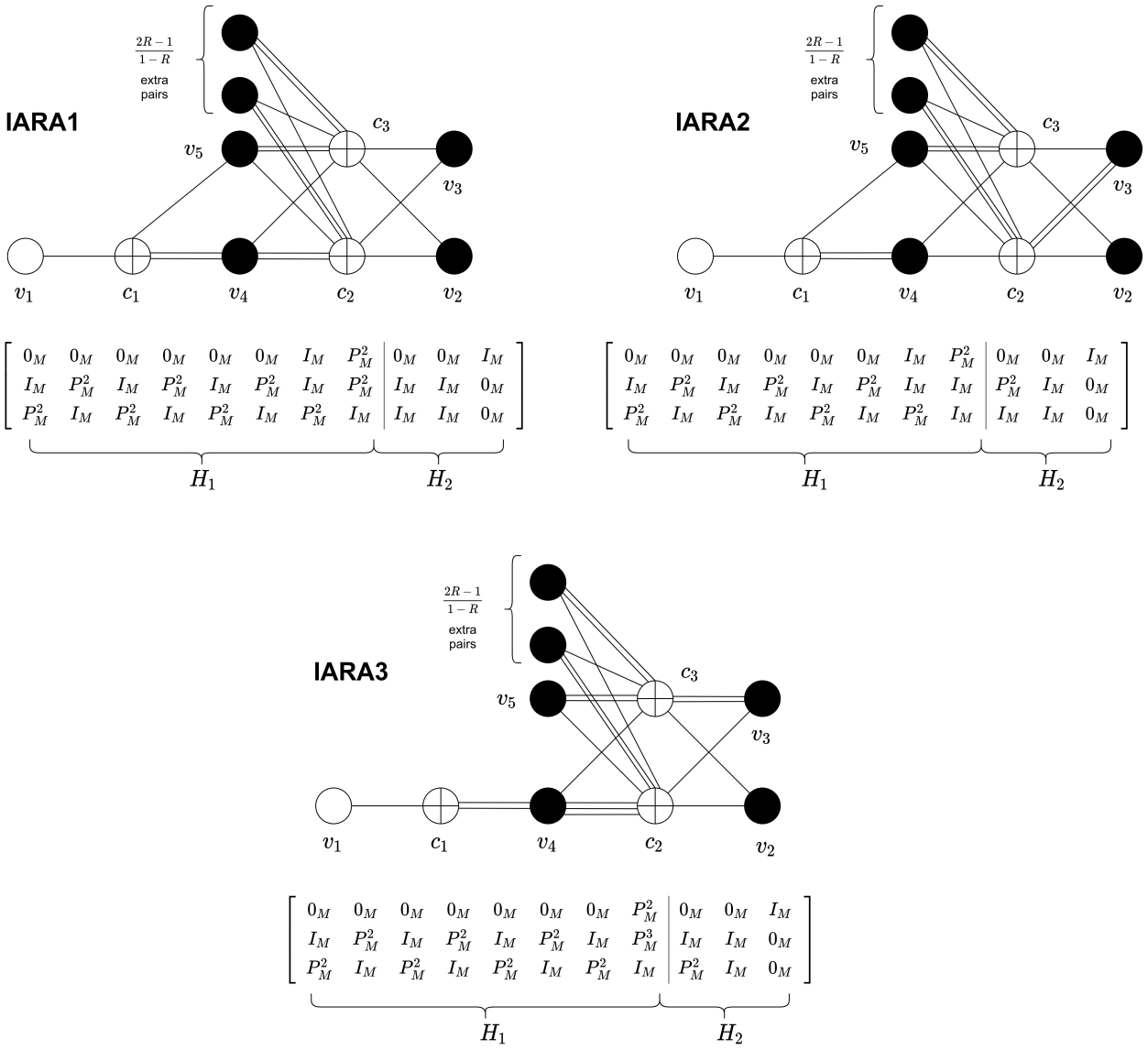

Figure 3.29: IARA protographs and parity-check matrices for rate 4/5 codes.

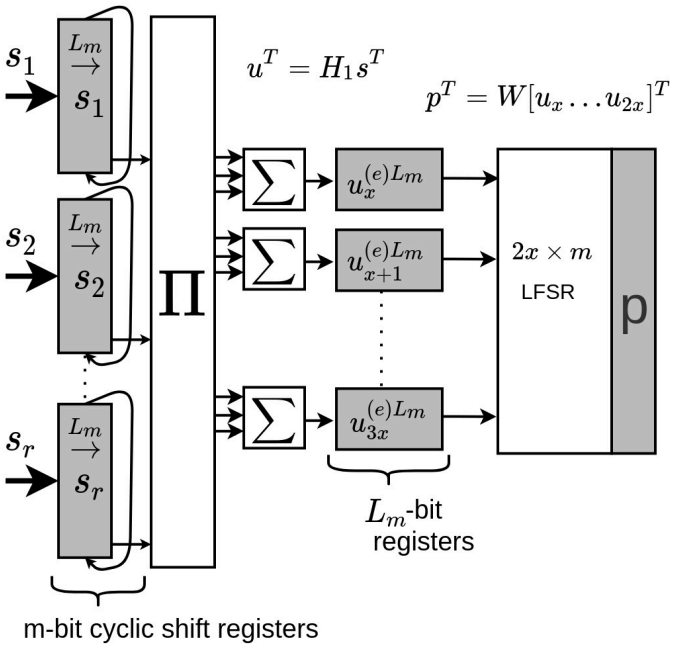

Figure 3.30: Encoder design for IARA codes. Last m parity bits are punctured

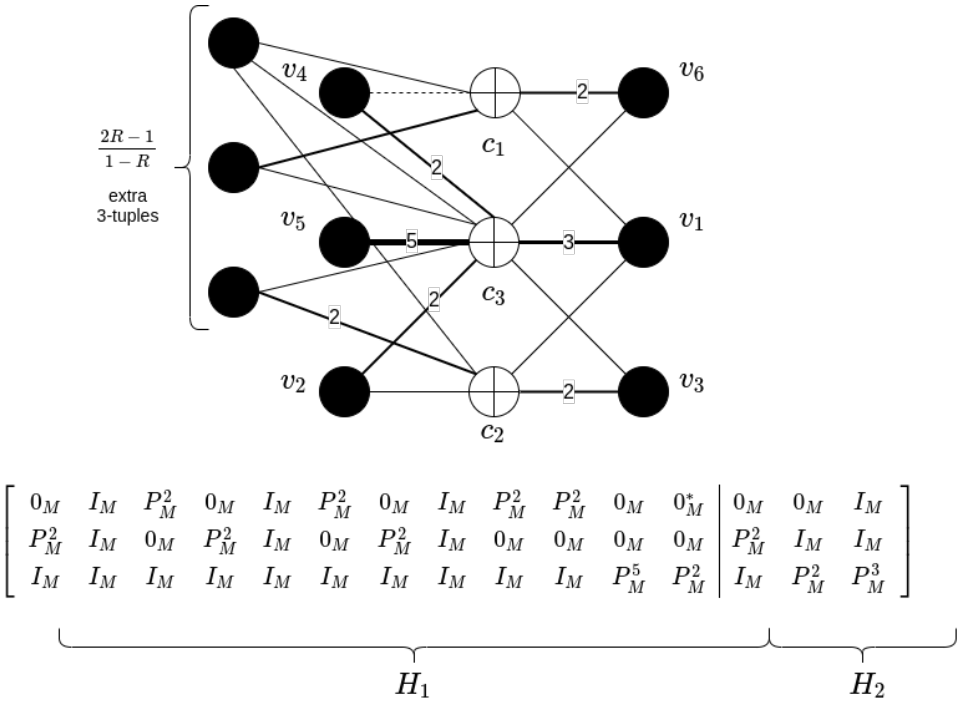

Figure 3.31: 2-D-P protographs and parity-check matrices for rate 4/5 codes. The dashed line between u_4 and c_2 is present only for 2-D-P2 code. Conversely, the marked submatrix 0_M^* becomes I_M for 2-D-P2. Edge degree > 1 is given as a number along each edge.

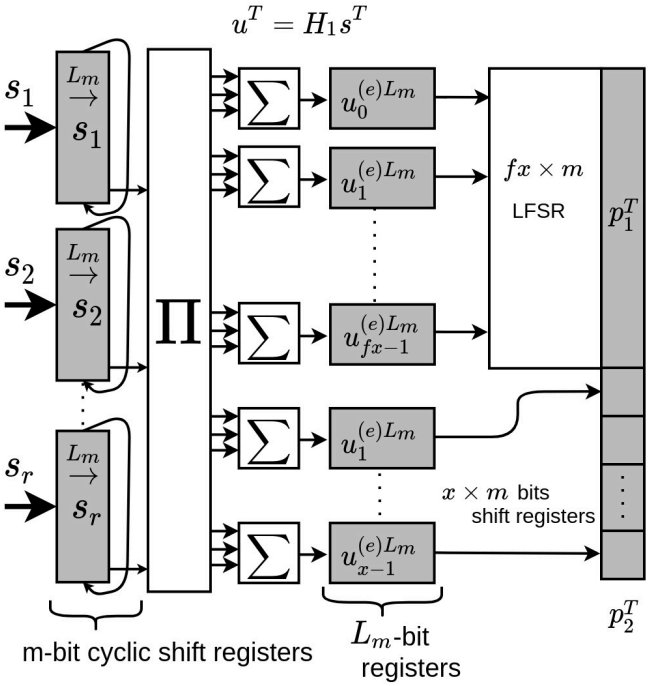

Figure 3.32: Encoder design for 2-D-P [39], nested high-rate ISI [128] and RCOP [61] codes. Parameter f is 3 for 2-D-P and RCOP codes and 2 for ISI.

as with IARA codes. For the calculation of p_2^T, successive $u_i^{(L_m)}$ subvectors are accumulated at each encoding cycle into simple shift registers. All calculations are completed after m/L_m cycles.

3.8.3 Nested high-rate ISI codes

In [128], the authors show the limitations of punctured protograph codes, when they are used with the Bahl-Cocke-Jelinek-Raviv (BCJR) equalizer and propose a method for designing rate-compatible, capacity approaching protograph codes for partial response channels. In [98], they extend their previous work and they propose two families of protograph codes, optimized for the EPR4 and dicode channels: nested high-rate and extended rate-compatible codes. All the proposed codes perform within 1.1 dB of the independent and uniformly distributed (i.u.d.) capacity limit. In this work, we focus on the first of the proposed code families, henceforth referred to as nested high-rate ISI codes, which, starting from a basic rate 1/2 protograph, build higher-rate variants by adding more variable nodes. In contrast, extended rate-compatible codes build higher rates through the addition of check nodes also, leading inevitably to larger graphs.

The proposed protograph structure for nested high-rate ISI codes is displayed in Fig. 3.33, along with an example of rate 4/5 parity check matrix. The degree distribution of nodes u_7 and higher is determined by protograph EXIT chart analysis, aiming at minimizing decoding threshold and it is not constant as in other protograph codes. In the diagram, check

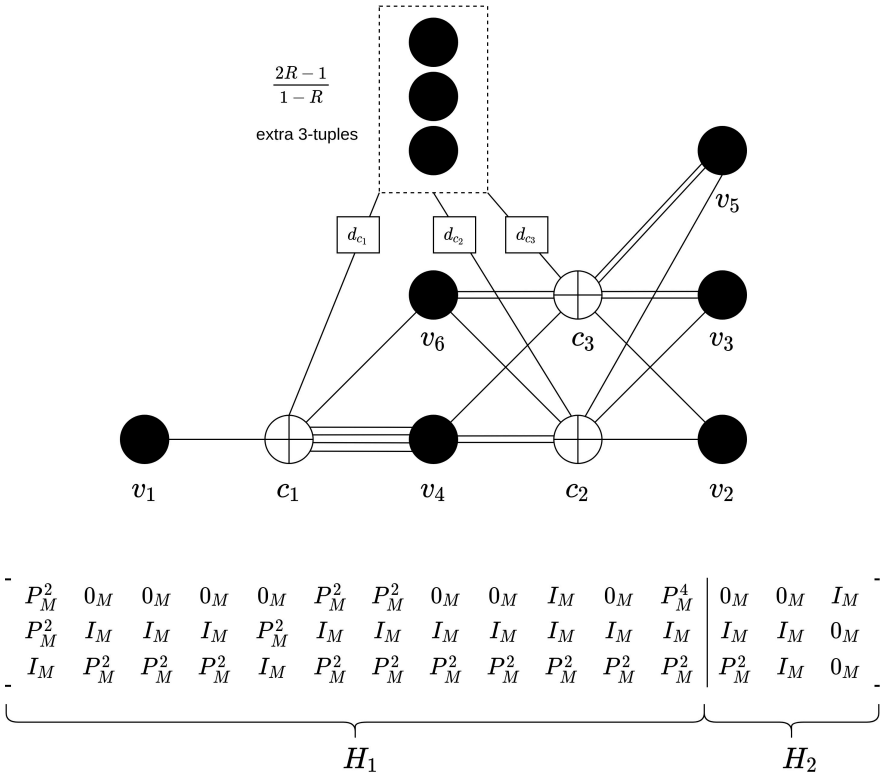

Figure 3.33: Nested high-rate ISI code protograph and parity-check matrix for rate 4/5.

node degree distribution for this subset of variable nodes is represented as $d_{c_i}, i = 1, 2, 3$.

The structure of matrix H_2^{-1} is provided in (3.38) and it is identical to that of IARA3 code, without puncturing. As with previous codes, however, the zero elements allow for simplification of the encoding process, by omiting the first xm bits of u^T from the dense matrix multiplication Wu^T . The last xm parity bits are calculated directly from intermediate vector u^T, as in the case of 2-D-P codes. Consequently, encoding can take place in an identical way as in (3.37). It follows that the encoder architecture is the same. Note, however, that submatrix W is considerably smaller than 2-D-P codes and consequently, the LFSR for dense matrix calculations is simpler.

$$H_{2(ISI)}^{-1} = \begin{bmatrix} 0_M & W_m(1,1) & \dots & W_m(1,2x) \\ & \vdots & \ddots & \vdots \\ & \vdots & \ddots & \vdots \\ 0_M & W_m(2x,1) & \dots & W_m(2x,3x) \\ I_M & 0_M & & 0_M \end{bmatrix} \tag{3.38}$$

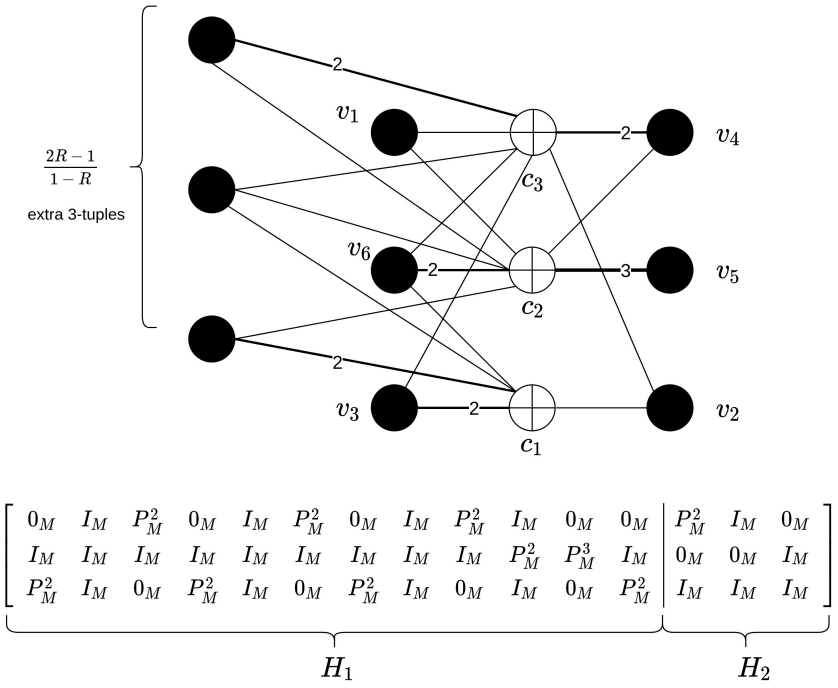

Figure 3.34: RCOP code protograph and parity-check matrix for rate 4/5.

3.8.4 RCOP

Targeting also the two-dimensional ISI channel, [61] describes a rate-compatible, non-punctured protograph LDPC code family. Starting from a basic rate 1/2 protograph and following a similar methodology to 2-D-P codes [39], the authors leverage the Finite Length (FL) EXIT chart analysis to construct higher rate compatible optimized codes (RCOP). The protograph is displayed in Fig. 3.34. The structure of matrix H_2^{-1} for this code is identical to (3.36). Consequently, the parity bits are calculated as in (3.37) and the proposed encoder architecture is that of Fig. 3.32.

3.8.5 Implementation results and testing

In this section, we provide hardware implementation results and describe the testing methodology for the designs proposed so far in the current Section, targeting FPGA technology. The implemented encoders were designed as IP cores for Zynq-7000 SoC technology and tested on the Xilinx ZC706 development board. The interfaces of the IP core interfaces follow AMBA AXI4-Stream specification [2], featuring a slave for receiving information bits to be encoded and a master interface for encoded parity bits.

Implementation results are listed in Table 3.14. In all cases, we selected rate 4/5, as this is the lower rate for practical applications on MR media. For IARA codes, since the maximum edge degree is 3, we select the value of the first expansion step $x = 4$, so that parallel edges are removed and the total codeword length is comparable to the other codes listed in

Table 3.14: Implementation results (synthesized design) on Xilinx Zynq XC7Z045-2 SoC

Code	x	m	Lm	Sector data (Bytes)	Codeword length (Bytes)	Resources LUTs	Resources F/Fs	Clock (MHz)	Throughput (Gbps)
IARA [60]	4	1024	8	4096	5120	51138	41423	395	25.28
2DP [39]/RCOP [61]	5	552	4	4140	5175	39007	41658	367	22.02
Nested ISI [39]	5	552	4	4140	5175	38082	41595	375	22.50

the table. For the other codes, we set parameter $x = 5$, which is the maximum edge degree of 2-D-P codes. The encoding throughput is calculated as the average number of parity bits calculated per second. Selection of the second expansion step parameter m was such that the resulting block length is close to 4KB, as required by the International Disk Drive Equipment and Materials Association (IDEMA) Advanced Format (AF) specification. The encoding throughput performance achieved satisfies commercial magnetic recording media requirements, while resource utilization is low.

For testing, we performed behavioural simulation of the encoder designs hardware descriptions against a GNU/Octave bit-accurate model. Random data were generated and provided to the design under test (DUT). Encoder responses were compared to GNU/Octave golden template and correct operation of the encoder was verified. For on-chip verification and validation of the implemented designs, we developed a testing environment on the Zynq 7045 SoC, featuring AXI4-Stream memory maps connected to the encoder IP core's interfaces. The contents of the register on the slave interface were initialised with random data through the SoC JTAG port. The generated parity bits were collected to another memory map connected to the core's master interface and read by the testing software, again through JTAG port and compared to the expected vectors of the GNU/Octave model. A diagram of the testing environment is provided in Fig. 3.35.

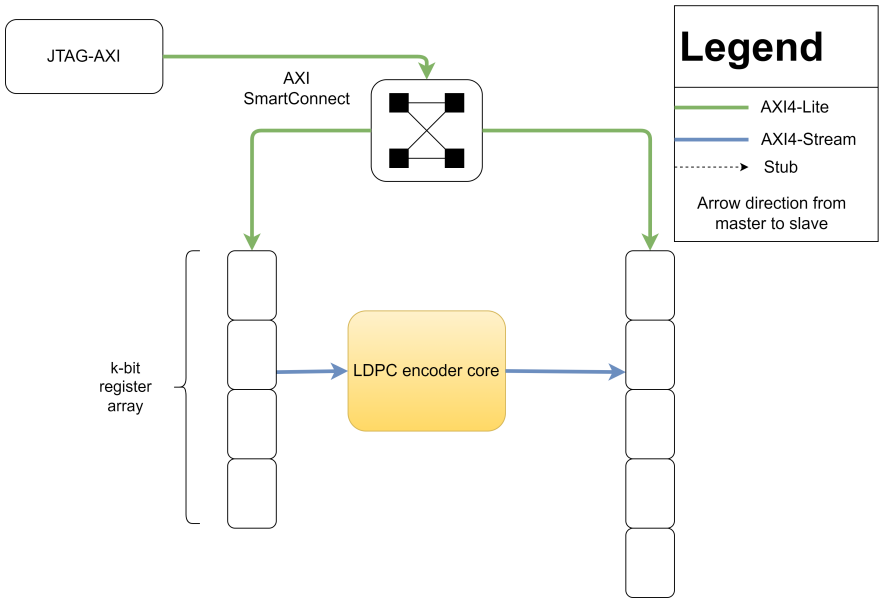

Figure 3.35: Testing environment for the magnetic recording media encoders.

4. PACKET-LEVEL ENCODER IMPLEMENTATIONS

4.1 Packet level erasure codes introduction

Packet-level erasure coding is a technique to increase reliability in many modern applications, such as edge computing [88], underwater acoustic sensor networks [65], magnetic recording media [72], hybrid broadcasting broadband television (HbbTV) [93] and delay tolerant networks (DTN) over deep space communication systems [15], in which traditional automatic repeat request (ARQ) schemes are not applicable or practical, or the fading effect of the communication channel is so deep that bit-level channel codes cannot provide the required reliability. Essentially, packet-level coding treats entire groups of information bits as symbols and performs error control coding over these symbols, instead of individual bits.

Typically, erasure coding takes place at the highest levels of a communication systems protocol stack, in which the erroneously received packets from the lower layers' channel codes, as well as the completely missed packets from deep fading phenomena, are treated as erasures. Packet-level erasure coding schemes can coexist with traditional channel FEC, offering different trade-offs. Joint use of erasure coding and bit-level FEC schemes in different scenarios has been studied in [22, 49, 101]. In [13], the authors combine RaptorQ codes at the application layer with real-time channel prediction and adaptively varying turbo codes at the physical layer.

The most common approach for coding over block erasure channels in space missions is the combination of Reed-Solomon (RS) codes [108] with interleaving. The (255,223) RS code with 8-bit symbols is the most popular example, having been a part of many missions' communication systems and recommended by CCSDS [32], along with its rate (255,239) variant. RS codes are maximum distance separable (MDS) codes: if (n, k) are the dimensions of a code, they can recover from the erasure of any $n-k$ or fewer symbols. Consequently, they can provide optimal error recovery capability. An interleaver is typically connected to the output of the RS encoder in order to protect against deep fading. Such coding schemes have been proposed in [34] and [35] for optical space communications. The limitation of RS codes, however, is high encoding complexity, which imposes the use of short block lengths. The polynomial arithmetic operations involved in encoding and decoding operations result in non-linear encoding/decoding complexity, even in the base case proposed in [118].

Another promising approach is the use of packet-level Low-Density Parity-check (LDPC) erasure codes, according to which encoded symbols are entire blocks of information bits. Although these codes are not MDS codes, they can achieve significant performance gains [29]. Capacity approaching ensembles can perform very close to the Singleton bound [70] and both encoder and decoder complexity can scale linearly with block length. Packet-level LDPC erasure coding has been proposed by the Consultative Committee for Space Data Systems (CCSDS) in experimental specification 131.5-O-1[33] for near-earth and deep space communications. The integration of packet-level LDPC codes in the CCSDS

protocol suite, along with their performance evaluation is described in [15].

The CCSDS experimental specification [33] describes two families of LDPC code design schemes: the *"online"* and the *"ad-hoc"*. In the former, the code is dynamically constructed for any desired block length and rate, based on a predefined node degree distribution and a pseudo-random permutation of the parity-check matrix columns. This method is also referred to as Flexible Irregular Repeat Accumulate (F-IRA) [89]. For applications, however, in which this flexibility is not required, a series of nine LDPC codes with fixed block lengths of 512, 2048 and 16384 symbols and rates $2/3$, $4/5$ and $8/9$ is also provided. These codes achieve better performance than the F-IRA and they are generated according to an algorithm which is similar to DVB-S2 LDPC codes [5].

To the best of my knowledge, the only implementation of the proposed codes is their integration into the the Interplanetary Overlay Network (ION) software suite [96], which is a software implementation of the bundle protocol for Delay Tolerant Networks (DTN), through the endorsement of OpenFec library [51]. In [15], a multithreaded implementation of the ION libraries is proposed for better performance. However, the purely software approach proposed is expected to exert considerable strain on the on-board general purpose processor and mass memory subsystem of a space Software Defined Radio (SDR), which is typically responsible for these functions. Offloading these tasks to a small footprint hardware accelerator integrated into a FPGA is especially important in the case of microsats and cubesats and high data rate optical communications, to achieve reduced size, weight, power, and cost (SWaP-C). Typically, spacecraft subsystems usually include FPGAs responsible for command and data handling (C&DH) tasks and the recent trend is to fully utilize these devices for multiple combined functions [52]. Moreover, FPGA hardware acceleration of packet-level coding enables a very high speed data processing chain providing data rates in the scale of several Gbps.

In the contribution described in the current Chapter, hardware architectures of packet-level erasure coding schemes for high data rate satellite communications are introduced, implemented and validated in a FPGA board, for the first time. First, an alternative description of the encoding algorithm presented in the CCSDS specification is provided, which is suitable for a hardware implementation, without modifying the code itself. In addition, an entirely new encoding algorithm is introduced, which can offer improved encoding throughput performance in certain implementation scenarios defined in this Thesis. Then, efficient encoder architectures for these encoding algorithms are provided and highly flexible and low-cost implementations of IP cores are developed, which achieve high encoding throughput, while using only a minimal percentage of FPGA resources. Although the encoder's primary function is offloading of LDPC PL-FEC functions from the on-board CPU, a considerable acceleration compared to the software implementations of encoding operations on state-of-the-art and common processor IP cores for spaceflight is additionally achieved.

+2

Table 4.1: Codes parameters

Rate	k=512	k=2048	k=16384
$2/3$	C_1	C_2	C_3
$4/5$	C_4	C_5	C_6
$8/9$	C_7	C_8	C_9

4.2 Background

In this Section, the structure of the packet-level erasure codes is briefly described. An alternative description to that provided in the standard [33] is adopted, and the corresponding notation, which facilitates the coherent presentation of the architectures in Section 4.5.

Nine such codes are defined in the standard [33], which are the combinations of three information block lengths over three code rates, as in Table 4.1. The information block to be encoded is partitioned into k packets of L bits each. Typically, each packet u_i encapsulates a single Protocol Data Unit (PDU) of the adjacent layer in the protocol stack, for example a Licklider Transmission Protocol (LTP) segment. These packets are hereto referred as information symbols of the code. From these information symbols, m redundant (parity) symbols p_j are generated. The union of the k information symbols and the m parity symbols constitutes one longword (LW) of the PL erasure code. LWs are transmitted through an erasure channel, at the receiving end of which, a packet is either received correctly in its entirety, or it is completely lost and it is considered as an erasure. The purpose of the redundant (parity) packets is to allow the reconstruction of the initial information block, even in the presence of erasures in the channel. It should become evident at this point that a packet erasure can occur when the underlying bit-level FEC algorithm has failed for this packet, or a CRC which is typically associated with the packet (e.g. in [35]) has failed as well. The packet-level codes of the CCSDS standard can be described equivalently as extended LDPC codes, by considering symbols of L bits, instead of individual bits. The parity check matrices of these codes are structured according to the LDPC codes defined in the DVB-S2 standard [5]. For each information symbol s_i, we define as F_{vn}^i a small pseudo-random subset of b_i parity symbols p_x which are affected by s_i: $F_{vn}^i = \{p_{x_1}, p_{x_2} \dots p_{x_{b_i}}\}$. The parity check matrix H of the packet-level codes is then a juxtaposition of a $m \times k$ sparse matrix H_u and a dual-diagonal $m \times m$ matrix H_p, as in (4.1).

$$
\begin{aligned}
H &= \begin{bmatrix} H_u & H_p \end{bmatrix} \\
H_u(x,y) &= \begin{cases} 1, p_x \in F_{vn}^y \\ 0, else \end{cases} \\
H_p(x,y) &= \begin{cases} 1, (x=y) \parallel (y=x-1), x \geq 1 \\ 0, else \end{cases}
\end{aligned}
\tag{4.1}
$$

In order to facilitate encoding and decoding, the sets F_{vn}^{i} are not defined randomly for all values of i, or, equivalently, submatrix H_u is structured. For each of the nine codes in Table 4.1, an array $\mathcal{M}_c$ of k/q sets is defined in the standard, with q being a parameter of the code and c the code number. Each row $\mathcal{M}_c(j) = \{x_1, x_2 \dots x_{b_j}\}$ contains a small subset ($b_i = 3 \leq 16$) of pseudo-random integers in the range 1 to m, the cardinality of which is not necessarily the same across the entire array: in other words, the length of each row in $\mathcal{M}_c$ is not constant. If $\alpha = m/q$ and $\mathcal{M}_c(x)$ is the x-th row of $\mathcal{M}_c$, the sets F_{vn}^{i} are calculated by adding a constant to each element of $\mathcal{M}_c(x)$, according to (4.2).

$$\begin{aligned} F_{vn}^{i} &= \left\{ p_\psi : \psi \in \{ x_j \overset{m}{+} c \mid x_j \in \mathcal{M}_c(i \div q) \} \right\} \\ c &= (i \bmod q)\alpha \end{aligned} \tag{4.2}$$

The symbol $\div$ in (4.2) denotes an integer division, whereas symbol $\overset{m}{+}$, a modulo-m addition. From (4.2), it obvious that b_i is constant for every q information symbols. The exact values of the $\mathcal{M}_c$ array are optimized for maximum performance of the resulting code, based on analytical code design tools and they are provided in the specification as constants.

4.3 Performance characteristics of packet-level erasure codes

The Singleton bound is the upper bound on the size of a block code. If n is the block length, d is the minimum distance of the code and q is the alphabet size, the Singleton bound states that the maximum number of codewords in the code is $|C| \leq q^{n-d_1}$. The performance of an error-correction code in the binary erasure channel is defined by how close the code is to the Singleton bound, or in other words, how much lower is the minimum distance of the code from the maximum value it can take for the specific code parameters. For example, in the case of a (n, k) linear code, the Singleton bound implies that the minimum distance of the code is $d_{min} \leq n - k + 1$. If the erasures in the channel are independent, the Singleton bound imposes a lower limit in the decoding failure probability:

$$P_e \geq \sum_{i=n-k+1}^{n} e^i (1-e)^{n-i} \tag{4.3}$$

In (4.3), e is the symbol erasure probability and P_e the codeword failure probability (i.e. the probability of an ideal decoder failure). MDS codes, like the RS codes, approach this limit, however, LDPC codes do not, which introduces a substantial performance penalty. For the codes in the standard, this performance gap is shown in Fig. 4.1 to be remarkably small.

As with the channel codes of Section 3.1, the performance of the packet-level codes depends also on the decoder. Decoding of PL codes can be performed either iteratively, with the MAP algorithm described in Section 3.2.3 or with the maximum likelihood algorithm. The former comes naturally with an inherent performance penalty, mostly as a result of

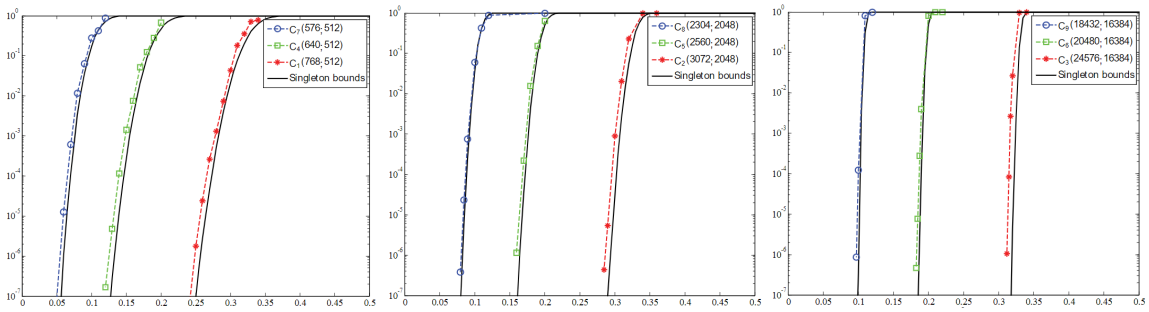

Figure 4.1: The codeword error rate (P_e vertical axis) vs the symbol erasure probability for the packet-level codes of this work. Source: [33]

cycles in the parity-check matrix of the code. Experimental results of the severity of this kind of loss under iterative encoding can be found in [103]. Maximum likelihood decoding, on the other hand, is the most optimal decoding method and it consists in solving the set of linear equations described in (4.4). The transmitted vector of N symbols is $\boldsymbol{x} = (x_1, X_2 \dots x_N)$ and the received vector that contains erasures is $\boldsymbol{y} = (y_1, y_2 \dots y_N)$. We denote as $\mathcal{K}$ the set of the indices of correctly received symbols in $\boldsymbol{y}$ and as $\bar{\mathcal{K}}$ its complement, i.e. the set of erasures indices. If $\mathscr{E}$ is an erased symbol, $\bar{\mathcal{K}} = \{i : u_i = \mathscr{E}\}$.

$$H_{\bar{\mathcal{K}}} \boldsymbol{x}_{\bar{\mathcal{K}}}^T = H_{\mathcal{K}} \boldsymbol{x}_{\mathcal{K}}^T \tag{4.4}$$

The complexity of the maximum likelihood algorithm is $O(N^3)$, however practical approaches have been introduced in [27] (improved probabilistic approach) and [103] (pivoting algorithms), that offer different trade-offs between complexity and performance. The data in Fig. 4.1 have been drawn with the maximum likelihood with pivoting method introduced in [103].

To conclude, the performance of the proposed codes is very close to the Singleton bound, or conversely, the proposed are almost MDS codes. Maximum likelihood decoding, however, remains a challenging task, although efficient decoding algorithms exist.

4.4 Encoding Algorithms

The processing challenge in the case of packet-level encoding is related to the large length of information symbols, rather than the processing complexity of the linear encoding operations. Each symbol spans several thousands bits, consequently, the direct application of traditional LDPC encoding algorithms used in bit-level coding, like the Richardson-Urbanke algorithm [110] and triangular factorization [81], is not feasible. Symbols in this case reside in a memory system and a packet-level encoder needs to access information symbols resident in that memory efficiently.

The two possible information symbols access patterns define two algorithms for parity symbols calculations, which are described in the current Section: the variable node oriented (VNO) and the check node oriented (CNO) algorithms. VNO is the standard encod-

ing algorithm presented in the CCSDS standard [33] (albeit described differently), while CNO is a novel algorithm introduced in this contribution. The complexity of both algorithms is linear to the symbol block length L.

4.4.1 Variable node oriented algorithm

According to the description in Section 4.2, the VNO algorithm progresses by updating all the parity symbols which depend on each successive information symbol. Following the LDPC code equivalence, the columns of H_u are processed one by one. The execution steps are summarized in Algorithm 1.

Algorithm 1 Variable node oriented calculation

1. All parity symbols bits are initialized to zero:
   $p_j(l) = 0, 1 \leq l \leq L, 1 \leq j \leq m$
2. For each information symbol u_i:
   (i) The list of parity symbols affected by u_i is calculated from (4.2).
   (ii) Symbol u_i is bitwise added to all the affected parity symbols in F_{vn}^i:
   $p_\psi(l) = u_i(l) \oplus p_\psi(l), 1 \leq l \leq L, \forall p_\psi \in F_{vn}^i$
3. Parity symbols are accumulated into successive symbol positions:
   $p_j(l) = p_j(l) \oplus p_{j-1}(l), 1 \leq l \leq L, 2 \leq j \leq m$

4.4.2 Check node oriented algorithm

On the contrary, the check node oriented (CNO) algorithm calculates each parity bit p_j by bitwise adding a subset F_{cn}^j of information symbols and the previous parity symbol, p_{j-1}, for $j > 1$. Contrary to the VNO algorithm in which array $\mathcal{M}_c$ is provided in the standard, the generation algorithm of F_{cn}^j needs to be defined. This is possible through a transformation of (4.2) as follows: for each code, starting from H_u, we define a base matrix $\mathcal{M}_c'$ with α rows. Each row r contains the indices of the non-zero elements of the r-th row of H_u, as in (4.5). Again, as in the case of $\mathcal{M}_c$, the number of elements in each row is not necessarily constant.

$$\mathcal{M}_c'(r) = \{\rho \mid H_u(r, \rho) = 1\} \tag{4.5}$$

We then observe a repetition pattern in the indices of the non-zero elements of the remaining rows of H_u: M_c' is repeated for totally m/α times, adding one at each successive copy. We define matrix $\mathcal{A}_c$ as the matrix of these non-zero elements. The repetition pattern resident in its structure is shown in Fig 4.2 and the resulting analytical expression for F_{cn}^j is given in (4.6).

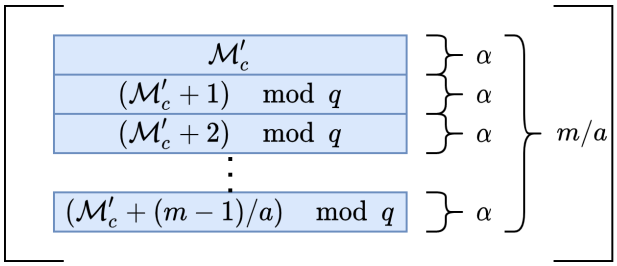

Figure 4.2: Structure of matrix $\mathcal{A}_c$. Only the first $2q$ rows are shown. Additions shown are modulo q

$$F_{cn}^{j} = \Big\{ u_\xi : \xi \in \big[(x_\delta \div q)q + (x_\delta \bmod q + c) \bmod q\big] \mid x_\delta \in \mathcal{M}'_c(j \bmod \alpha) \Big\} \qquad c = (j \div q)(q/\alpha) + (j \bmod q) \div \alpha \tag{4.6}$$

Taking into account the dual diagonal structure of H_p, the algorithm progresses by processing each row of the parity check matrix of the code and generates one parity symbol after the other. Algorithm 2 summarizes the above steps.

Algorithm 2 Check node oriented calculation

1. All information symbols u_i are initialized to their values.
2. For each parity symbol p_j:
   (i) The set F_{cn}^{j} of contributing information symbols is calculated by (4.6)
   (ii) Symbols $u_\xi \in F_{cn}^{j}$ are bitwise added to p_j:
   $p_j(l) = u_\xi(l) \oplus p_j(l), 1 \le l \le L, u_\xi \in F_{cn}^{j}$
   (iii) The previous parity symbol is added to the above sum, if $j > 1$:
   $p_j(l) = p_j(l) \oplus p_{j-1}(l), 1 \le l \le L, 2 \le j \le m$

4.4.3 VNO and CNO tradeoffs

VNO and CNO algorithms are equivalent and their complexity is exactly the same. However, their performance can be significantly differentiated by the scenario in which they will be called to operate.

VNO has the advantage of being able to begin encoding processing before all the information symbols become available in their entirety. Also, it does not require the storage of

Table 4.2: algorithms trade-offs

	VNO	CNO
Buffer space	mL bits	kL bits
Memory access pattern	Random read/write $\rightarrow$ sequential read	Sequential write $\rightarrow$ random read
Starting condition	None	All information symbols in buffer
Parity symbols generation	All symbols in the end concurrently	One symbol after the other during execution

the information symbols in a suitable data structure: symbols are processed one by one and once operations involving a symbol u_i have finished, this symbol is no longer necessary for the further progression of the algorithm. In the end, all the parity symbols become available on the memory buffer practically at the same time.

On the other hand, CNO requires that all the information symbols are available on a buffer to start. Consequently, the buffer size ($kL/8$ bytes) is considerably higher than VNO ($mL/8$ bytes). Also, the dependency on the availability of all information symbols for the parity generation process to start, may introduce latency in the data processing chain. CNO algorithm is suitable for scenarios in which all the information symbols have been generated from the previous processing step in the data processing chain and stored in a shared memory buffer, when the encoding operation is requested. CNO also features a completely different memory access pattern than VNO, by separating read and write operations. After the first step of algorithm 2, no further memory write operations are required, which can lead to improved memory performance. Note also that all CNO write operations are to successive memory addresses of a possibly contiguous area, while VNO memory access operations are to apparently random positions. The absence of spatial locality in the consecutive read and write operations can significantly impact the performance of an implemented hardware system, by limiting cache systems efficiency. These trade-offs are summarized in Table 4.2.

4.5 Hardware Architectures

In the current Section, the hardware architectures for the two encoding algorithms of Section 4.4 are introduced, for an on-board data processing scenario requiring streaming input and output interfaces. Block diagrams of both the CNO and VNO algorithms are illustrated in Fig. 4.3 and Fig. 4.5 respectively. From a high-level perspective, they incorporate the following major features and components:

- Streaming input and output interfaces, so that on-the-fly operation in a data process-

ing chain is supported. This corresponds to a single read and a single write channel. The data width of all the encoder interfaces is d bits.

- A configuration interface and a memory for loading of matrices $\mathcal{M}$ and $\mathcal{M}'$ and the selection of the code (C_1 to C_9) of the current encoding operation.
- Read and write channel modules, which support access to a RAM subsystem.
- A RAM subsystem, which is used as scratchpad memory. The introduction of this subsystem is necessary for the calculation of parity symbols, since the large symbol length does not allow the implementation of encoding algorithms using exclusively on-chip memory resources.
- Internal AXI4-Stream interfaces with elastic buffers, which simplify the design and minimize the critical paths.
- Glue and control logic, as described in the current section

4.5.1 CNO architecture

We start the description of the proposed architectures with the one for the CNO algorithm, which is apparently the simplest of the two and it is shown in Fig. 4.3. Initially, all the information symbols are written in the external memory, taking up k successive symbol positions. Afterwards, for each parity symbol p_j, module F_{cn} calculates the indices ξ of the information symbols u_ξ which contribute to p_j. Indices ξ, in turn, define the source address for the successive read operations required. Register r is initialized to zero and accumulates all consecutive u_ξ, but valid parity symbols are present at the output interface only when the last i_ξ from the set F_{cn}^j is being read from memory. Note that all read symbols are accumulated into r, which is a simplification of step 2(iii) of Algorithm 2. The CNO encoder's operation is detailed in Algorithm 3.

The calculation of indices ξ is executed by the module R_{cn}, a block diagram of which is displayed in Fig. 4.4. It contains a configuration memory, which stores the parameters $\mathcal{M}'_c$, a small lookup table $\mathcal{F}$ for the code parameters q and α and simple elements like logical shifters and adders, which implement equation (4.6). The number of the code for the current LW (among the nine codes defined in the CCSDS specification) and index of the current parity symbol (j) are the inputs of the module and ξ is the output. An important detail concerning the storage scheme of parameters $\mathcal{M}'_c$ is that we add an extra bit to the stored parameters, which is set to '1' for the last parameter of each one of the α sets of F_{cn}^j. This extra bit is necessary for the implementation of the condition in line 7 of Algorithm 3. Since the maximum value of j is 8192, the size of register j is 13 bits. Also, elastic buffers have been inserted into the module as required, in order to assure timing closure.

Operations on successive packets can be pipelined using double buffering, provided that there is the necessary buffer space available: during the calculation and the output of the current longword ($\mathcal{L}_i$) from the output serial interface, the write channel can receive the

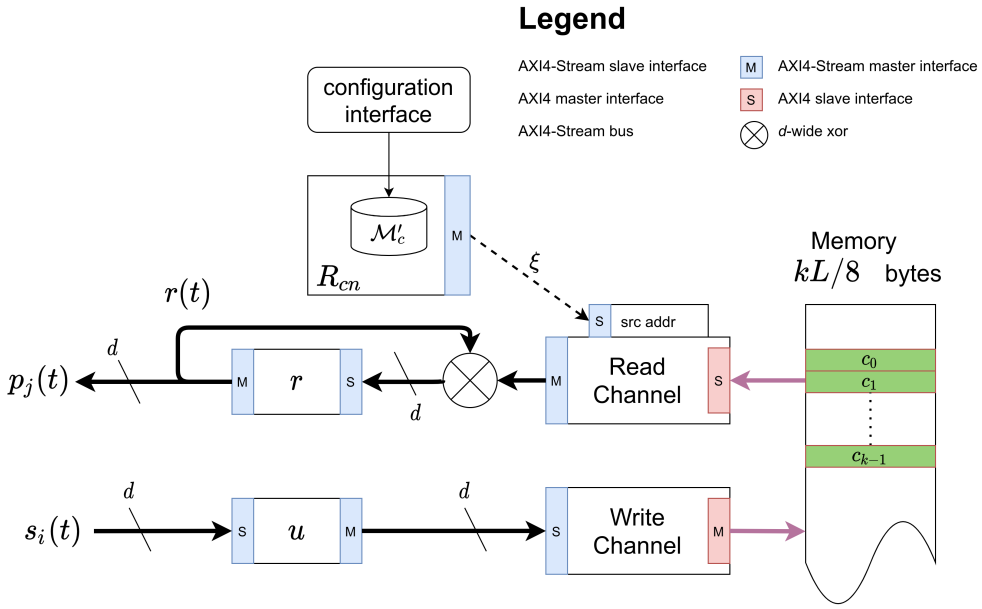

Figure 4.3: CNO architecture

Algorithm 3 CNO encoder's operation

for $i = 1 \ldots k$ **do**
    $u \leftarrow s_i$, $c_i \leftarrow u$ ▷ Write s_i in memory
end for
$c_k \leftarrow u$, $u \leftarrow 0$ ▷ Receive last s in memory and reset
for $j = 1 \ldots m - 1$ **do**
    for all $c_\xi \in F_{cn}^j$ **do**
        if ξ is the last element of F_{cn}^j **then**
            $p_j \leftarrow r$, $r \leftarrow r \oplus c_\xi$
        else
            $r \leftarrow r \oplus c_\xi$
        end if
    end for
end for

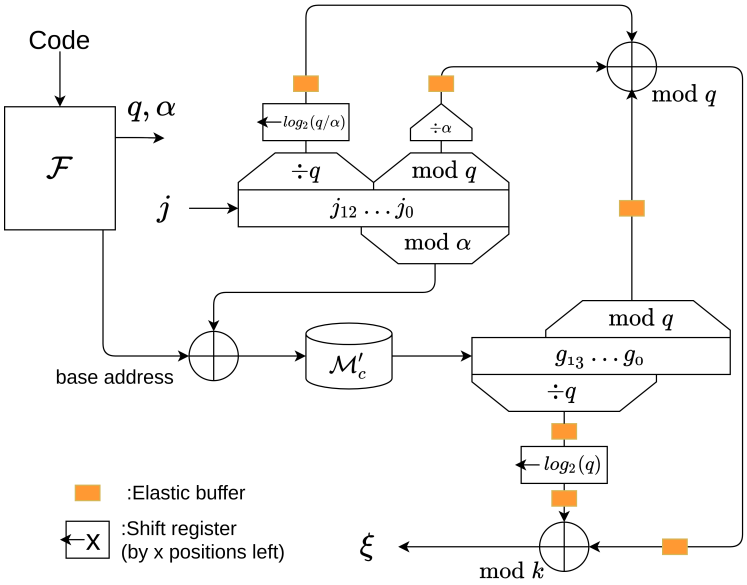

Figure 4.4: Block diagram of R_{cn} module.

information symbols of the next one ($\mathcal{L}_{i+1}$), writing the incoming information symbols into a different address base. After all parity operations of $\mathcal{L}_i$ have been concluded, they start again from the address base of $\mathcal{L}_{i+1}$.

4.5.2 VNO architecture

The block diagram of the proposed VNO architecture is provided in Fig. 4.5. The information symbols s_i are presented to the stream input interface in subvectors $s_i(t)$ of d bits, with $1 \leq t \leq L/d$ and accumulated into buffer u after L/d steps. In the meanwhile, the list of the indices of parity symbols $\{\psi\}$ which affected by the current symbol is retrieved from the corresponding module (R_{vn}), based on configuration parameters and (4.2). Parameters ψ also define the source and destination addresses for the read and write operations of the corresponding memory access modules.

For each index ψ, module R_{vn} calculates also the logical variable $\Xi(\psi)$, which is the index of the first information symbol which updates parity symbol p_ψ. Based on this information, and assuming a byte addressable RAM, a read operation is executed at symbol address $L_\psi/8$ only if $\Xi(\psi) > i$, which indicates that symbol p_ψ needs to be updated in RAM. If $\Xi(\psi) \leq i$, the data of symbol c_ψ do not need to be updated and only a write operation is executed at the current destination address. With this optimization, our architecture eliminates the first initialization step of Algorithm 1. Also, step 2)(ii) is simplified, since we set p_ψ equal to u_i.

The encoder's operation is summarized into Algorithm 4. Initially, during the first loops in lines 1-11, $\Xi(\psi)$ always returns zero, which means that c_ψ is accessed from the first time and a read operation is not necessary. Afterwards, as an increasing number of par-

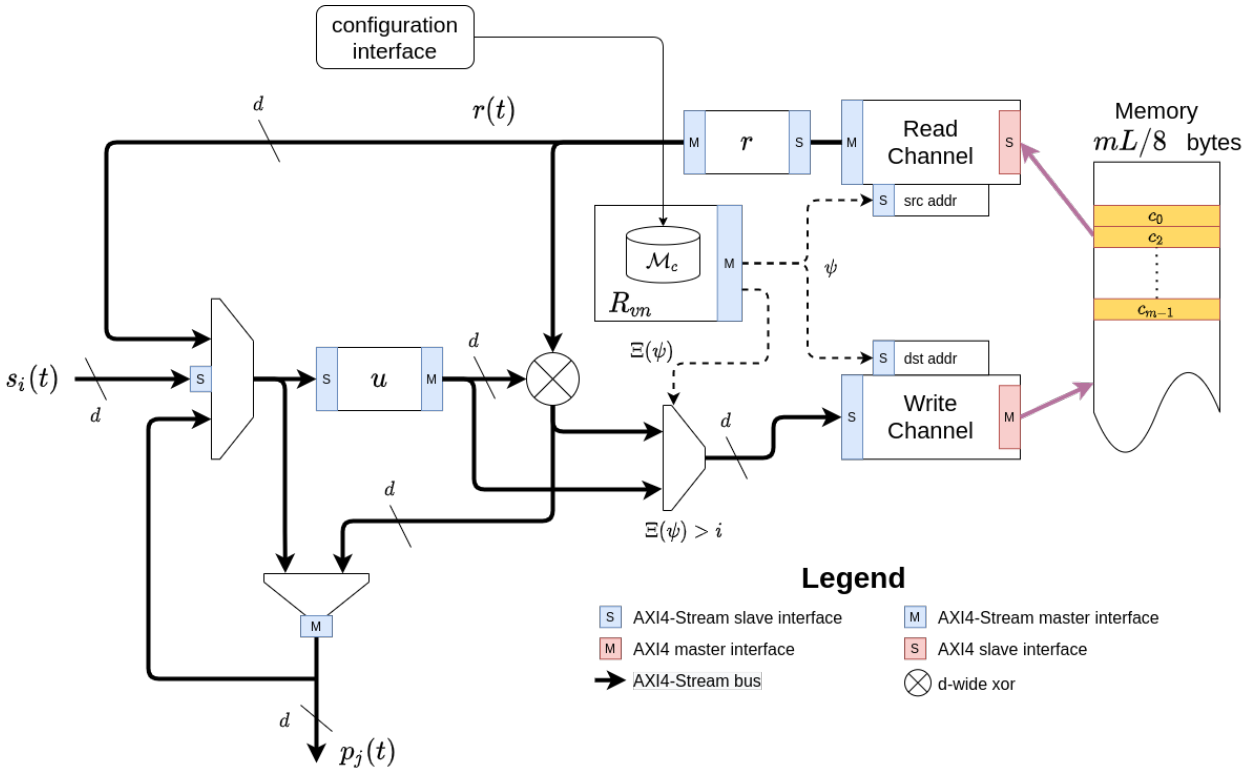

Figure 4.5: VNO architecture

ity symbols have been written into, write-only cycles are executed together with update cycles, which are cycles which are indicated in lines 7 and 8 of Algorithm 4. Finally, after a specific value of i, which is different for each code, $\Xi(\psi) > i, \forall \psi < m, i < k$, which indicates that only update cycles are executed and the check in line 4 of Algorithm 4 becomes redundant.

Another optimization introduced by the proposed architecture is the elimination of the accumulation step in Algorithm 1: as the parity bits of the j-th symbol are transmitted from the stream output interface, buffer u is reused and the same parity bits are shifted into it. At the same time, d bits of parity symbol $j-1$ are shifted out of u, in order to participate in the accumulation operation of the Algorithm in line 16 of Algorithm 4.

Algorithm 4 VNO encoder's operation

```
1:  for i = 1...k do
2:      u ← s_i                                   ▷ Receive s_i into buffer
3:      for all c_ψ ∈ F_vn^i do
4:          if Ξ(ψ) = 0 then                      ▷ First access to c_ψ
5:              c_ψ ← u                           ▷ Write symbol to memory
6:          else                                  ▷ c_ψ has already been affected
7:              r ← c_ψ                           ▷ Read c_ψ from memory
8:              c_ψ ← u ⊕ r                       ▷ Update c_ψ
9:          end if
10:     end for
11: end for
12: r ← c_1
13: p_1 ← r, u ← r
14: for j = 2...m do
15:     r ← c_j
16:     u ← r ⊕ u, p_j ← r ⊕ u                    ▷ Accumulate parity symbols
17: end for
```

A detailed block diagram of the R_{vn} module is displayed in Fig. 4.6. The calculation of parameters ψ is based on (4.2) and can be implemented by using simple logic elements: shift registers, multiplexers, adders and a small lookup table $\mathcal{F}$, as shown in Fig. 4.6. For each information symbol i, a total of b_i values of ψ is calculated. Since the maximum value of i is 16384, the size of register i is 14 bits. The $log_2(q)$ least significant bits of register i are multiplied by α, which is always a power of 2. Consequently, the calculation of $(i \bmod q)\alpha$ is done with a shift register. The rest $14-log_2(q)$ most significant bits of register i are added to a base address retrieved from $\mathcal{F}$, which is constant for the entire encoding operation and this sum specifies the absolute address of the indices in $\mathcal{M}_c$. Finally, the two intermediate results $\mathcal{M}_c(i \div q)$ and $(i \bmod q)\alpha$ are added modulo-m to calculate the final parameter ψ.

For the calculation of $\Xi(\psi)$, the R_{vn} module incorporates a R_{cn}^0 module, which receives the indices ψ at its input and for each of them, generates only the first index from the set F_{cn}^{ψ}. The returned value is the index of the first information symbol which affects parity symbol p_ψ. Elastic buffers are also inserted in the module as required in order to assure timing

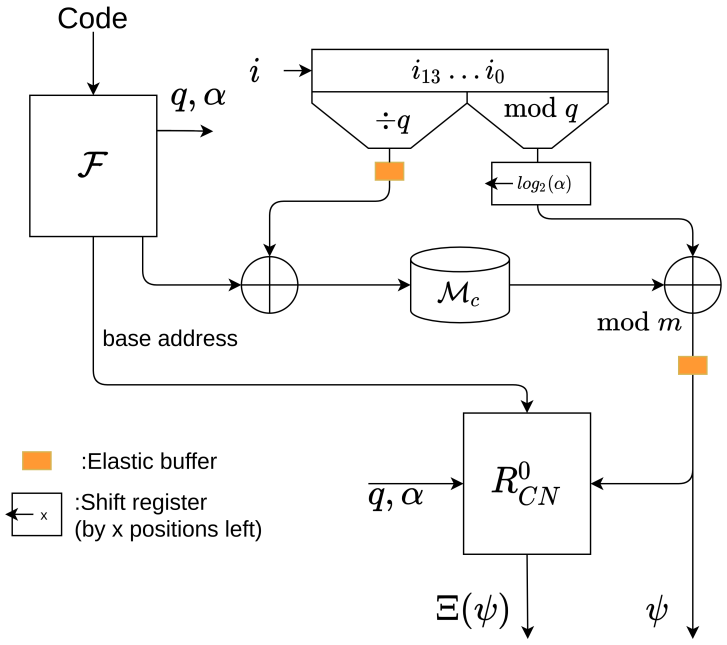

Figure 4.6: Block diagram of the R_{vn} module.

closure.

The accumulation of information symbols in line 2 of Algorithm 4 is pipelined between successive symbols, in order to accelerate the encoding process. The pipelined operation of the architecture between consecutive longwords, however, requires double buffering: as soon as processing of all information symbols of the current longword ($\mathcal{L}_t$) have completed, the parity symbols are ready to be transmitted from the front memory buffer, through the read interface. From this point, the information symbols of $\mathcal{L}_{t+1}$ start updating the corresponding parity symbol positions of the back buffer. Less obvious, however, is the solution of the contention problem for the read memory interface during the pipelined operation: the read channel is used both for update operations on $\mathcal{L}_{t+1}$ and for the output of parity symbols of $\mathcal{L}_t$. The solution is to add an arbitration mechanism for the read requests coming from these two sources, which assigns priority to the reads related to the update operations. The parity output is available only when the read channel is not used, which happens when $\Xi(\psi) > i$, as in Fig. 4.7. With this mechanism, read and write channels are always busy and encoding throughput is maximized.

4.5.3 Design Considerations

Since the processing complexity of the two algorithms is the same, the combinatorial logic hardware requirements of both architectures are not expected to differ significantly. Instead, it is the memory access pattern, that raises performance challenges to the RAM subsystem, that differentiates the two architectures. The resource and performance requirements for both architectures are listed in Table 4.3. The VNO architecture performs

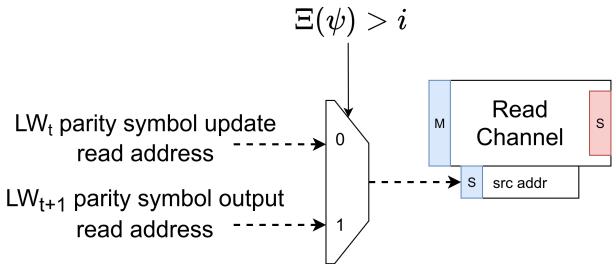

Figure 4.7: Read channel arbitration of the VNO architecture.

b_i write memory operations for each one of the k symbols ($b_i = |F^i_{vn}|$,$b'_j = |F^j_{cn}|$). The optimization introduced by avoiding the first read operation for a empty symbol position (using variable $\Xi(\psi)$), however, removes m read operations. We define parameter $\mathcal{W}$ as the sum of the non-zero elements in H_u, which can be expressed equivalently as in (4.7). At the end of the encoding process, m symbols are read from the memory into the output interface. Due to the pipelined operation of the architecture during encoding of consecutive longwords, parity symbol reads are executed simultaneously with encoding calculations of the subsequent longword. Consequently, the total cycles required for encoding (T_c) can be considered to be equal to those required for lines 1-11 of Algorithm 4. Supposing that read and write channels can work simultaneously and independently, total encoding time is attributed to the time required for RAM writes only, as shown in Table 4.3.

$$\mathcal{W} = \sum_{i=1}^{k} b_i = \sum_{j=1}^{m} b'_j = \sum H_u(i,j) \tag{4.7}$$

Conversely, the CNO architecture performs k write operations per each longword, corresponding to the k information symbols, followed by b'_j, writes for each one of the m parity symbols. Thanks to pipelining, write operations of longword $\mathcal{L}_{t+1}$ in lines 1-3 of Algorithm 2 can be executed concurrently with the reads in lines 6-13 of the current longword $\mathcal{L}_t$, and hence T_c can be considered to be equal only to the time required for reading. In the total calculation times listed in the table, a constant number of $\mathcal{C}$ idle cycles for each memory access is added, which accounts for the memory latency. This accounts for the time required for the setup of the AXI transaction, until actual data start moving on the interface.

The minimum latency (Λ_{min}) of each architecture is defined as the minimum number of cycles between the of reception $s_0(0)$ and the output of the first parity symbol $p_0(0)$, as in Fig. 4.8, assuming no delay cycles on the input interface. We observe that both architectures complete calculations in the same number of cycles, but the minimum latency of CNO architecture is considerably lower. However, as described in Section 4.4, all information symbols need to be available at the first step of the algorithm and the amount of RAM required is significantly higher than VNO, especially when double buffering is used in order to support the pipelined operation. VNO architecture is preferable in a data chain comprising a sending entity which transmits sporadically packets of data in the form of symbols, with pauses between them and at the same time, the entity which receives data after encoding is capable of handling a continuous stream of parity symbols. The situa-

Table 4.3: Resource and performance estimation

Architecture	VNO	CNO
RAM total read (bytes)	$LW/8$	$LW/8$
RAM total write (bytes)	$LW/8$	$Lk/8$
Total cycles (T_c)	$LCW/8d$	$LCW/8d$
Minimum latency (Λ_{min})	$LW/8d$	$Lk/8d$
RAM buffer bytes	$mL/4$	$kL/4$

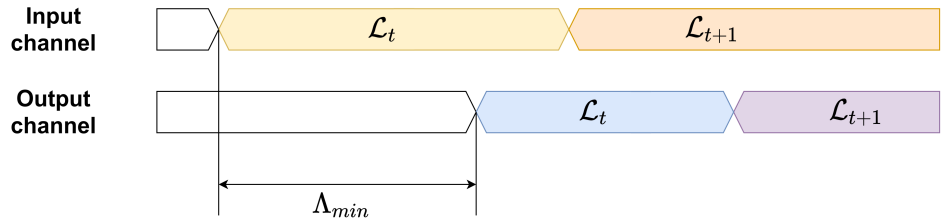

Figure 4.8: Minimum latency definition.

tion is reversed in the case of CNO, which favors a scenario with a continuous sending process and intermittently receiving endpoint.

4.6 Hardware Implementation and Validation

In the current Section, the hardware implementation and validation of the two architectures VNO and CNO which were introduced in Section 4.5 is decribed, in the form of IP cores for various FPGA Systen-on-Chip (SoC) platforms.

The following implementation requirements for the IP cores are considered:

- Flexibility in the selection of each individual code on a per codeword basis through a configuration register update, without affecting throughput performance. The rate and block length can change during the mission, in response to varying channel fading characteristics.
- Flexibility in erasure symbol block length. Typically, this parameter is expected to remain constant in a mission and must be compatible with underlying protocol stack. For CCSDS protocols [31], typical values range from 128 to up 4K bytes.
- Access to a Random Access Memory (external to the FPGA), which is used as scratchpad memory for the parity symbols calculations.
- Streaming input-output of data. Typically, on-board subsystems communicate over high-speed serial data links, e.g. ECSS-E-ST-50-11C (SpaceFibre)[10].

- Hardware footprint should be minimum, while encoding throughput should be adequate for current and envisioned space communications standards, like the upcoming CCSDS Optical On-Off Keying specification [56].

The KCU105 board is the first implementation target used in the current work and the first platform in which the correct operation of the implemented encoders was validated and for which performance data were derived. In the KCU105 board, the XCKU040 FPGA is connected to 2GB of DDR4 SODIMM RAM memory, accessible from the programmable logic subsystem through a Memory Interface Generator (MIG) IP core, which exposes an AXI4 interface to the user logic.

The block diagram of the implemented design is provided in Fig. 4.9, which also exhibits the environment which has been used to validate the functional operation and evaluate the performance of the proposed IP cores. The encoders' parameters are configured from an AXI4-Lite interface and feature AXI4-Stream input-output interfaces. The memory-mapped interface for access to the RAM implements AXI4 protocol. The encoders are connected to the AXI4 user interface of the Xilinx MIG IP core, which provides access to the external DDR4 RAM. Test data are generated by a Fibonacci-type Linear Feedback Shift Register (LFSR) and the produced parity symbols are accumulated into a Multiple Input Signature Register (MISR). Both registers (LFSR and MISR) are based on primitive polynomials and they are therefore maximal length. The reason we have selected these modules for test vector generation and signature compression is their implementation simplicity, which allows them to operate at full clock speed, without introducing idle cycles in the AXI4-Stream interfaces of the encoders. Protocol compliance with AXI4 Specification is validated by a protocol checker core, which also records performance data: namely the total number of bytes transferred through the interfaces and the number of clock cycles required for each encoding operation. The initialization of the encoding core and the test modules is done through AXI4-Lite configuration registers. The same protocol is used for the readback of the encoders' status register and the test modules output data (MISR value recorded and total number of cycles). Finally, all these parameters are accessible from a user application through a JTAG to AXI4-master core from Xilinx.

The expected MISR signature has been calculated off-line by a bit-accurate software model of LFSR and MISR modules and an encoder based on the OpenFEC [51] project. This software model was executed for several longwords and the results were compared with those of the behavioral simulation of the Register Transfer Level (RTL) description of the reference design in Fig. 4.9, as well as with the FPGA-in-the-loop results on the KCU105 board. For the RTL simulation, the Vunit open source simulation framework [19] was used, in order to model the memory-mapped AXI4 interfaces of the design, assuming a perfect memory model: a RAM which can be accessed concurrently for read and write with zero latency.

Implementation results and performance data for the two architectures of Section 4.4 are summarised in Table 4.4. The derived performance data refer to encoding with the CCSDS code C_3, which is the code with the largest parameters n, k and which is therefore the most demanding case, in terms of hardware resources. The simulated number of cycles (T_c)

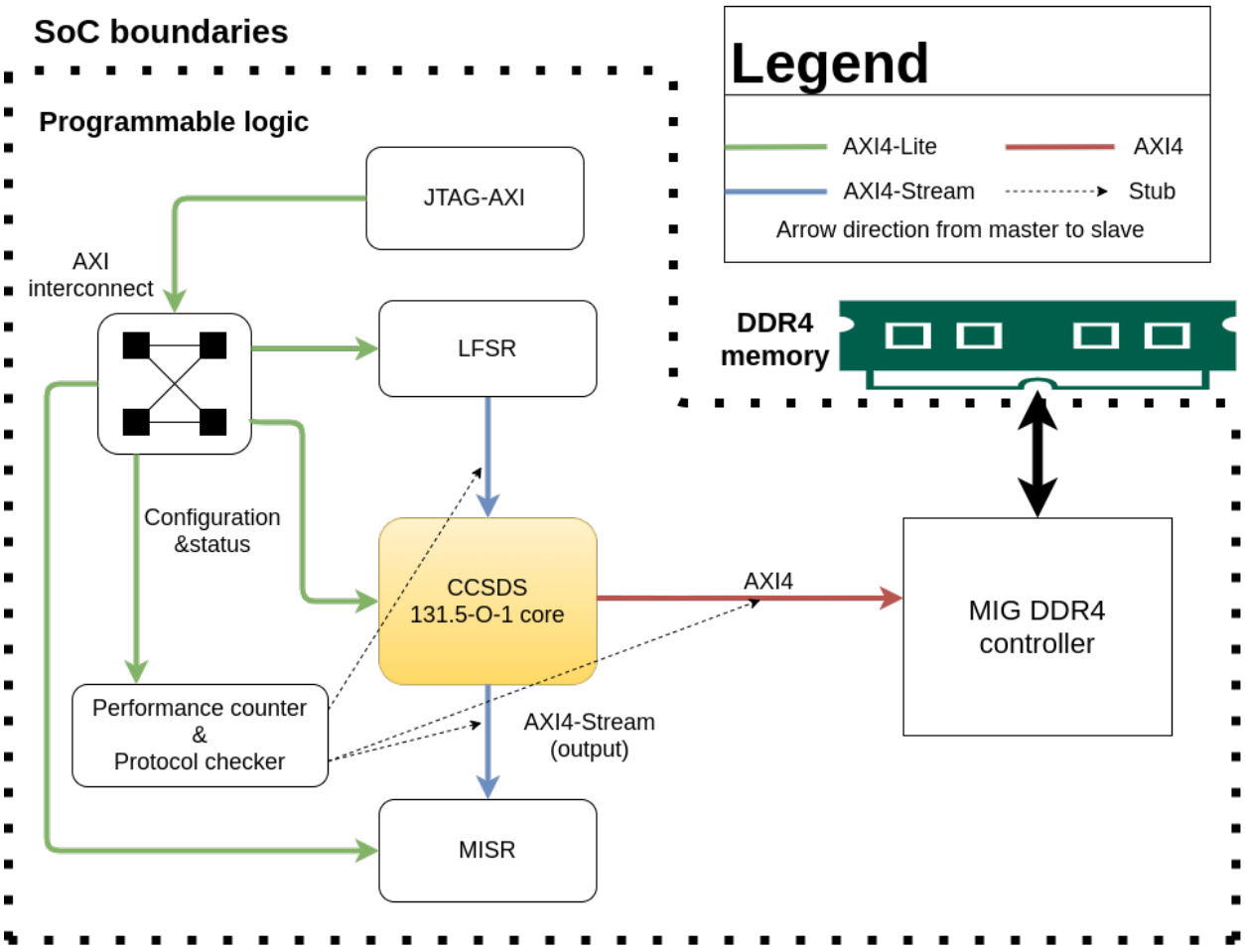

Figure 4.9: Encoders' reference design on the KCU105

data were derived from the RTL simulation (assuming an ideal memory). The only idle cycles introduced are those required for AXI4 protocol transaction setup and handshaking, which set parameter $\mathcal{C}$ to a value slightly greater than 1. The value of parameter $\mathcal{C}$, however, is considerably higher in the implemented designs. Especially in the case of the implemented VNO architecture, the continuous alternation between read and write operations results in an increase of the value of parameter $\mathcal{C}$ in the implemented design, which is expected: the DDR memory can be accessed only for read or write at a given time. Consequently, lines 7 and 8 of Algorithm 4 cannot be executed in parallel, contrary to what the AXI4 channel abstraction may suggest. This limitation, however, does not pertain to the CNO implemented design, in which read and write operations do not occur concurrently and memory can be accessed more efficiently.

Analysing the performance profile of the implemented architectures, it becomes evident that packet-level encoding is heavily dependent on data transfer operations and the critical factor defining performance is the memory bandwidth. In the KCU105 development board, the DDR4 component memory can provide up to 2400 MT/s on a 64-bit interface, accessible through a 512-bit AXI4 bus at 300MHz. As shown in Table 4.3, the CNO architecture requires fewer total memory transactions and is therefore favoured in a memory bandwidth constrained environment, such as the one we present in Fig. 4.9: not only is the final encoding throughput achieved with the CNO core higher, but this also is feasible with marginally more than half of the available RAM bandwidth. This allows for another benchmarking test on the KCU105 board with two cores, which, in the case of the CNO cores, leverages almost the full memory bandwidth, as shown in Table 4.4. A similar test with two VNO cores instead, yielded no significant performance benefit against a single core, because memory bandwidth has already been almost exhausted with a single core.

Table 4.4 demonstrates also the different implementation trade-offs for the two hardware architectures. Both architectures require only a small percentage of the chip and board logic resources. The resource utilization of the VNO architecture is higher than that of CNO. A large part of this difference is attributed to the double buffering on register u in Fig. 4.5, or correspondingly, line 2 of Algorithm 4, and the large burst size (4 KiB) of AXI4 memory transactions. The hardware requirements for both architectures can be reduced significantly, if a lower total burst size (burst length $\times$ beat width) or parameter d is set, as example in Section 4.7, where parameter d set to 128 bits. It is evident, however that the performance is reduced proportionally. Finally, the power was measured on the KCU105 board, using the SYSMON core from Xilinx. SYSMON was connected to the board's system controller through its I2C bus and power measurements were recorded during continuous operation of the cores. Totally 120 samples were averaged, in order to obtain the recorded values. The frequency settings that are presented in the table are the maximum frequencies that ensure the saturation of the AXI4 interface of the MIG DDR4 core on this board: setting the encoders' clock to a higher frequency has no positive effect on the encoding throughput. The lower clock frequency of the 2-core implementations has a key effect on the total power consumption. However, the total power of the design is dominated by the MIG controller and the IO associated with it (VCC1V2 power rail). Finally, it is evident that the CNO architecture, by virtue of its lower footprint and memory

Table 4.4: Resource and performance measurements on the KCU105

	VNO		CNO	
	1-core	2-core	1-core	2-core
LUTs (% of FPGA utilization)	5377 (2,22%)	10754 (4,44%)	2031 (0,84%)	4062 (1,68%)
Flip-Flops (% of FPGA utilization)	5714 (1,18%)	11428 (2,36%)	3560 (0,73%)	7120 (1,47%)
36Kbit BRAMs (% of FPGA utilization)	24,5 (4,08%)	49 (8,17%)	17 (2,83%)	34 (5,67%)
External RAM (% of KCU105 utilization)	64 MiB (3,13%)	128 MiB (6,25%)	128 MiB (6,25%)	256 MiB (13%)
T_c (simul.)	5043455		4981250	
C (simul.)	1,092		1,078	
T_c (recorded)	12357857	23982085	10699412	14017186
C (recorded)	2,68	2,6	2,32	1,52
Throughput	13,3 Gbps	13,4 Gbps	15,05 Gbps	22,98 Gbps
RAM bandwidth utilization	75%	77%	53%	81%
Clock frequency	300 MHz	150 MHz	300 MHz	150 MHz
Static power	2,55 W	2,53 W	2,48 W	2,12 W
Dynamic power	2,76 W	2,8 W	1,74 W	2 W
Power efficiency (Gbps/W)	2,5	2.51	3,57	5,58

d=512-bits, L=4KiB, C_3 code, Burst size=4KiB, read priority MIG scheduling.

access pattern, has more favourable power characteristics.

4.7 Comparison to CPU Implementations

Although the encoders' primary role is offloading of LDPC PL-FEC functions from the on-board CPU, the achieved performance can lead to a considerable acceleration compared to the software implementations of encoding operations on a general purpose CPU. In the current Section, we compare the performance of our hardware accelerator implementations with that of the purely software encoding procedure on some commonly used and state-of-the-art space-grade CPUs. The comparisons are against the software encoder implementations on the LEON3, LEON5 and NOEL-V soft processors.

For the performance benchmarking measurements, a resource efficient SoC design was built, with the necessary GRLIB IP modules and the proposed encoder cores, as shown in Fig. 4.10. Apart from the cores required for a basic LEON/NOEL-V subsystem, the SoC of Fig. 4.10 comprised the proposed hardware encoding cores as the design in Fig. 4.9, although in this case the parameters of the encoder had been initialized at synthesis time. The bus width of all AMBA buses was set to 128-bits and the entire design was clocked by a single clock source: the clock generated by the MIG controller, which was set at a very low frequency (10 MHz). Since the aggregated read and write bandwidth of the the AXI4 channel of the MIG controller (32 MiB/s) is a small fraction of the total DDR4 memory bandwidth of the board (19200 MiB/s), the memory model provided by the MIG DDR4 controller is as close as possible to a zero-latency memory and ensures a common

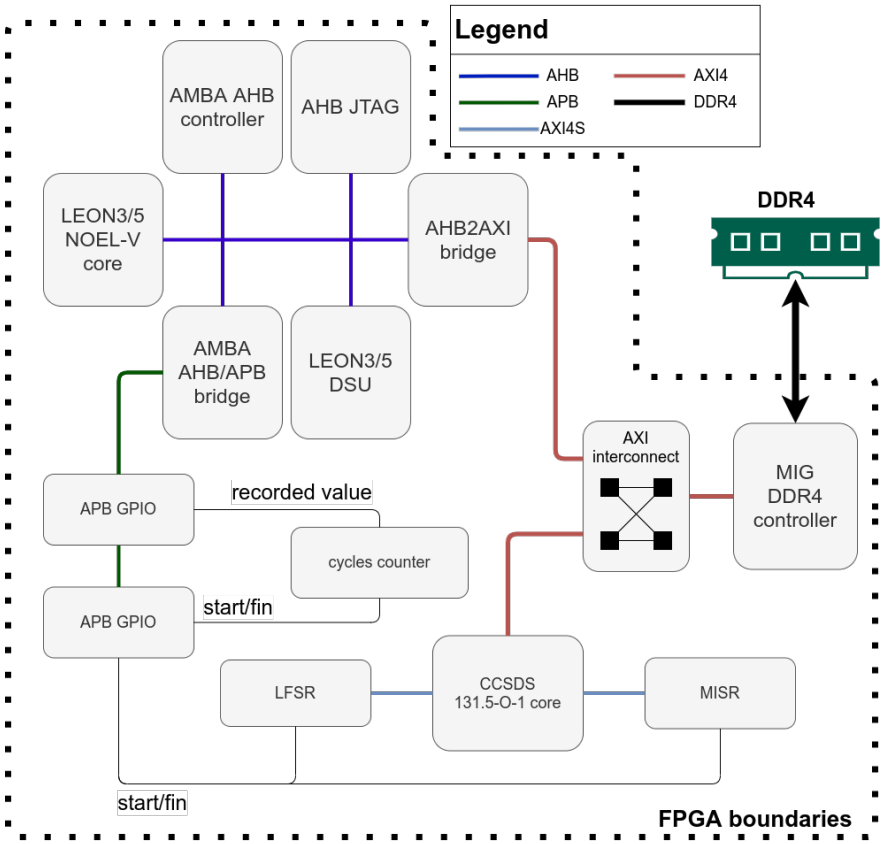

Figure 4.10: System design for performance comparisons.

basis for the comparison, since the total cycles required for encoding are not affected by the behavior of the memory. The performance metric which we have adopted for the comparisons was the average number of the 10 MHz clock cycles which were required for encoding one LW, instead of the actual encoding throughput. A direct comparison on the basis of raw performance achieved would not be meaningful in the case of soft processors because the actual data-rate performance depends on the achieved frequency of each mission specific place and routed FPGA SoC design.

Two tests have been run on the SoC: a hardware test as in Section 4.6 and a purely software test on the LEON/NOEL-V processor, which is a specially designed simplified and optimized version of the corresponding routine provided in [96], in which erasure coding is implemented as an additional software layer to protect LTPs segments and called Erasure Coding Link Service Adapter, or ECLSA. We compiled the software with the vendor's cross compilers, setting the highest optimization level for the corresponding ISAs. Test vectors and expected results were identical in both test cases, through a software implementation of the LFSR and MISR modules. The compiled software was loaded on the processor core through JTAG interface, using the vendor's *grmon* utility. Both tests were initiated by the processor system and the clock cycles required in both cases were recorded by a specially designed performance counter core, which was started and stopped through General Purpose Input-Output (GPIO) signals: the first (start) is asserted at the beginning

Table 4.5: Acceleration characteristics against LEON/NOEL-V soft processor on the KCU105

		RAM buffer	Average T_c	Speedup factor
Software	LEON3	192 MiB	1796901689	1
	LEON5		566576395	3,2
	NOEL-V		564176669	3,2
CNO	LUT:898 FF:1267 BRAM:6	128 MiB	20522702	87,6
VNO	LUT:1827 FF:1866 BRAM:8	64 MiB	20512826	87,6

L=4KiB, C_3 code. $d = 128$.

of a LW encoding procedure and the second (fin) at its completion. Performance data were averaged over 100 LWs.

In [15], a software implementation is described, which uses three threads: one for filling the encoding matrix with the information symbols, a second for actual encoding and a third for transmission of the encoded parity symbols. On the contrary, the hardware architectures introduced in the current work use double buffering techniques to pipeline the encoding of consecutive longwords. Both techniques succeed in absorbing the time required for the reception of information symbols and the transmission of encoded parity data into the time required for actual encoding. Consequently, the performance comparison made in this Section takes into account only the encoding time, that is the time required for calculation of the parity symbols, provided that information symbols are available as soon as they are requested. We set $L = 4$ and $d = 128$ bits in the comparisons, but the encoding time is proportional to the information symbol block length and memory bus width in both cases.

In Table 4.5, the total number of clock cycles which were recorded by the performance counter module are listed. Although in the memory bandwidth constraint test environment described Section 4.6 the CNO core outperformed the VNO by a significant margin, in this scenario, both cores recorded almost the same performance, as it was expected by the theoretical analysis of the corresponding algorithms in Section 4.4 and the data of Table 4.3. Also, the performance of the specific memory-bound workload is almost the same for LEON5 and NOEL-V. Finally, compared to the hardware implementations, the RAM buffer space of the software implementation is larger. From the data listed in Table 4.5, it is evident that the proposed architectures offer a significant acceleration of packet-level encoding functions, compared with the on-board software implementation on embedded processors used in space applications.

Another performance comparison was made against the dual-core ARM Cortex-A9 pro-

cessor of the Zynq XC7Z045 System-on-Chip (SoC) featured by the ZC706 board. As already descibed in Chapter 2 and more specifically in Section 2.3, the ZynQ processing system is currently gaining increasing interest for aerospace applications. The software implementation of the PL coding engine which was used for comparisons with the LEON/NOEL processors was ported to the ARM platform, using the Vitis Unified software platform from Xilinx.

The system's design block diagram used for the comparisons is displayed in Fig. 4.11, which is similar to Fig. 4.9, except that the encoder cores are connected to two HP ports, instead of the MIG controller and initialization and control of the components in the PL region are performed through the General Purpose (GP) AXI4-Lite interface of the PS, under software control. In the ZC706 board, the Processing System (PS) is connected to 1 GB of component DDR3 SDRAM, which is not directly accessible from the Programmable Logic (PL). Instead, the PL subsystem exposes four 64-bit high performance (HP) AXI3 Slave interfaces to the PL and shares the same DRAM controller with the embedded ARM processors. Of course, as already mentioned in Section 2.3, the PL logic can use the 1GB of SODIMM memory, which is dedicated to it (the PL). However, the component memory option was selected, so that both the memory controller as well as the memory itself are the same in both tests.

The controller of the ZynQ PS on the ZC706 is clocked at 533 MHz, which corresponds to a total peak data rate of 4,264 MB/s. In practice, however, the effective data rate at the HP ports is lower than this value [113], [68]. The hardware encoders were clocked at 250 Mhz. For the software encoder implementation on the ARM system, an encoding kernel was cross-compiled in the form of a C language bare-metal application. Full optimization level was selected as a compilation option and the ARM NEON optimizations were activated, in order to take advantage of the advanced Single Instruction Multiple Data (SIMD) technology in the ARMv8 architecture. The ARM software made use of the DMA's interrupt outputs to signal the beginning and the end of encoding process, in order to calculate the performance data. The PS for the test was clocked at the maximum value of 800MHz. The performance achieved is listed in Table 4.6 and is limited only by the available bandwidth of the memory controller of the ZynQ SoC and the board RAM. The RAM memory system's bandwidth utilisation in the experiments was consistent with that in Table 4.4. In all cases, significant speedup has been achieved. However, it is not possible to measure the PS power on the ZC706 board, since the power regulators do not provide the necessary current measurement outputs on the PMBUS of the card.

Another series of experiments has therefore been conducted for the comparisons against the quad-core ARM Cortex A53 Application Procession Unit of the ZynQ Ultrascale+ PS system of the ZCU102 board. The reference design of Fig. 4.12 was built to support the measurements. On the ZCU102, the high performance ports are AXI4 compliant, which makes integration easier. For the PL system tests with the hardware encoders, the programming of the configuration and control registers of the encoders, as well as the control of MISR compression and were performed by the PS. On the other hand, for the PS system tests, AXI4 slave interfaces were created for the LFSR and MISR modules, through which the PS was able to read the random data and send the output of the software encod-

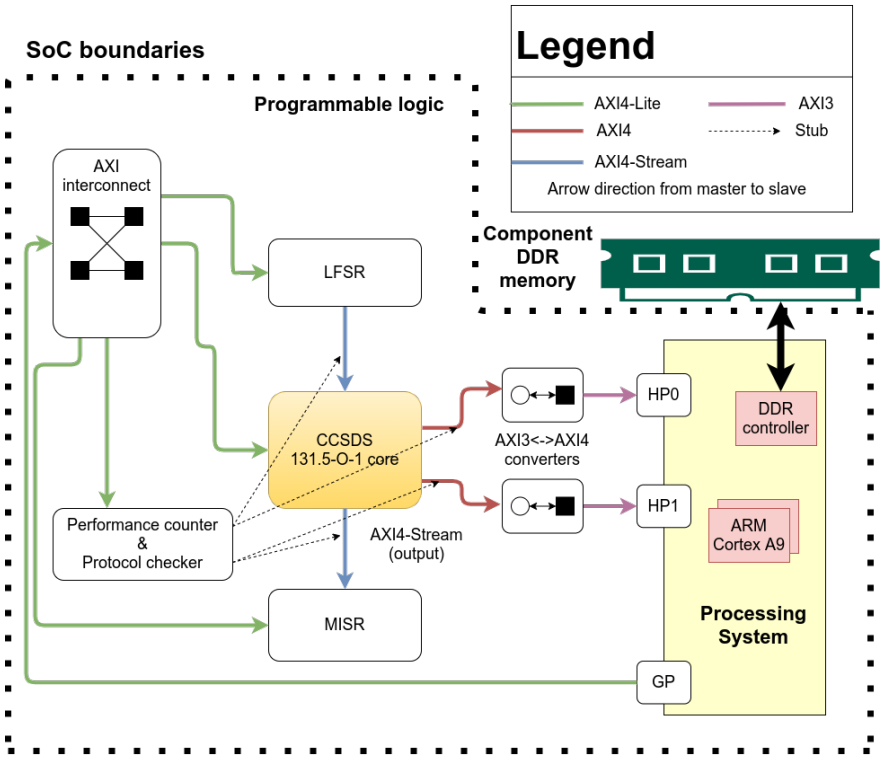

Figure 4.11: System design for performance comparisons against the ARM embedded processor on the ZC706.

Table 4.6: Acceleration characteristics against the embedded ARM Cortex A9 processor on the ZC706

	LW time (msec)	Enc. throughput (Gbps)	Speedup factor	Mem. saturation
Software (on ARM)	890	0,66	1	UNKOWN
VNO (1 core)	190	2,83	4,7	73%
VNO (2 cores)	177	3,03	5	78%
CNO (1 core)	170	3,15	5,2	50%
CNO (2 cores)	96	5,57	9,3	88%

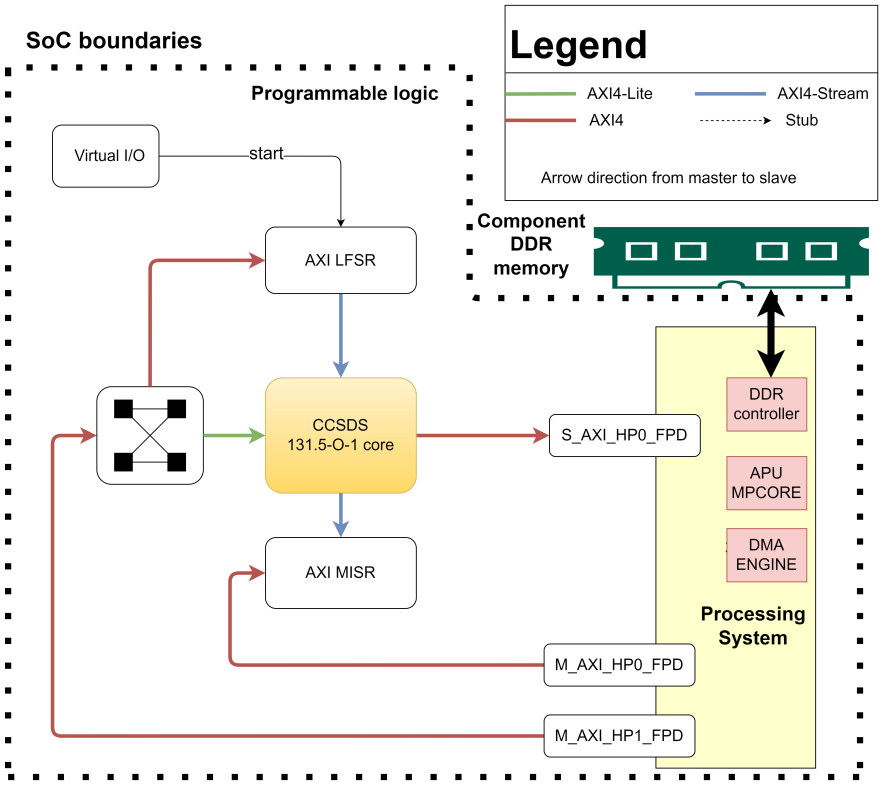

Figure 4.12: System design for performance comparisons against the ARM embedded processor on the ZCU102.

ing routine to the MISR module. These two functions (writing of the data to be encoded to a memory region and sending the encoded output to the MISR AXI4 address), were performed by the DMA controller of the PS, through special software routines, which were synchronised through the interrupt controller. It is also important to note that the encoding software was single-threaded, since bare-metal applications do not support threads. This means that the software encoding process cannot pipeline input of data, encoding and output, as is the case with the hardware encoders. However, the DMA controller on the ZynQ Ultrascale+ is very efficient, and input and output of data account for a small (5%) percentage of the total encoding time. The clock of all the PL modules was set at 333 MHz, which the maximum frequency supported by the HP ports of the PS. Both cores (CNO and VNO) saturated the bandwidth of the port and consequently, there was no point in extending the tests to multiple encoder cores per HP port. Finally, the data bus of all the AXI interfaces was set at 128-bits.

The performance and power measurements results are displayed in Table 4.7. The frame encoding time is the average time for a full frame (longword) reception from the LFSR, encoding and parity output to the MISR. This setup implements a realistic scenario of a potential on-board system and takes into account the totallity of the processing steps required to receive data from a sensor and to transmit the processed data to the next recipient on the data-processing chain. The power efficiency is calculated by taking into

Table 4.7: Resource and performance measurements on the ZCU102

	VNO	CNO	ARM A53
Frame encoding time (ms)	67,79	69,81	319
Encoding throughput (Gbps)	3,96	3,84	0,84
Dynamic power (W)	1,29	0,66	0,94
Power efficiency (Gbps/W)	3,07	5,82	0,9

d=128-bits, L=4KiB, C_3 code.

account the dynamic power dissipated during the encoding process. It can be seen that hardware encoding is substantially more efficient than the software counterpart, on the ZynQ Ultrascale+ platform.

5. CONCLUSIONS AND FUTURE WORK

The novelties introduced in the current thesis, regarding QC-LDPC encoder architectures, are shown to achieve state-of-the-art performance, with the lowest documented hardware footprint, as a result of the optimized architectures which are introduced and an efficient bit vector multiplication with dense matrices. LDPC codes without the dual-diagonal structure or any other focused provisions in the structure of their parity-check matrix, or their protograph, in order to facilitate encoding, have been so far considered unfavorable for high-speed, resource starved implementations, even though they exhibit some favorable FEC performance features. The work in this Thesis manages to bridge this gap between encoding efficient LDPC codes (like those of the IEEE802.11ac/n standard) and the general case of QC-LDPC codes, with potentially better FEC behavior, broadening their application field.

In addition, it is shown, for first time in the literature, that PL-FEC encoding can be implemented efficiently in hardware and offers another option for increasing the reliability of modern applications, other than traditional systems combining error correction codes and deep interleavers. The efficiency of the proposed hardware architectures is such that they can be implemented even in low cost FPGA platforms and they can easily satisfy the performance requirements of current and upcoming high data rate communication standards. The proposed encoders can readily form part of a high-speed data chain for small satellites and CubeSats in a single chip data processing unit implemented in FPGA technology, combined for example with the parallel implementation for hyperspectral compression on a Zynq-7000 platform which have been proposed in [126]. This contribution enables the application of PL-FEC for next generation high data rate reliable satellite communications in space environments where burst errors are dominant, under low signal-to-noise ratio regimes.

One of the most active and promising development areas among the CCSDS working groups is related to the standardization in the domain of free-space optical communications. A multitude of options is being considered for various application scenarios: from high photon efficiency aiming mostly deep-space communications to low-complexity options of the O3K standard. A broad overview of the activities taking place within CCSDS and the Interagency Operation Advisory Group (IOAG) is given in [56]. Of particular interest is the possibility of endorsement of the rate $1/2$ AR4JA LDPC codes of [32] into the new free-space optical communication standards [34]. Protograph-based Raptor-like (PBRL) and Accumulate-Repeat-Accumulate (ARA) LDPC codes are also candidates for the emerging O3K standard and an already future research direction is to provide hardware accelerators for them, following the accumulated expertise on LDPC encoding. Erasure coding is also mentioned [34] as an option for optical high data rate communications, consequently, the work presented in Chapter 4 is going to be adapted accordingly. Finally, the focus of this thesis has been the encoding process of either bit-level or packet-level codes. LDPC and most importantly, packet-level decoding are also challenging tasks. Especially in the case of packet-level coding, there are no known hardware implementations. The design of a complete codec, therefore, would be an invaluable contribution.

ABBREVIATIONS - ACRONYMS

3GPP2	Third Generation Partnership Project 2
ACE	Approximate Cycle Extrinsic Message Degree
AHB	Advanced High-performance Bus
AI	Artificial Intelligence
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
API	Application Programming Interface
AR4JA	Accumulate, Repeat by 4, Jagged-Accumulate
ARQ	Automatic Repeat reQuest
ASIC	Application-Specific Integrated Circuit
ASM	Attached Synchronization Marker
AWD	Asymptotic Weight Distribution
AXI	Advanced eXtensible Interface
AWGN	Additive White Gaussian Noise
BCJR	Bahl–Cocke–Jelinek–Raviv
BEC	Binary Erasure Channel
BER	Bit-Error Rate
BRAM	Block RAM
C&DH	Command and Data Handling
CADU	Channel Access Data Unit
CCSDS	Consultative Committee for Space Data Systems
CFDP	CCSDS File Delivery Protocol
CLB	Configurable Logic Block
CMMB	China Mobile Multimedia Broadcasting
CNO	Check Node Oriented
COTS	Commercial Off-The-Shelf

CPLL	Channel Phase Locked Loop
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
DD	Displacement Damage
DMA	Direct Memory Access
DDR	Double Data Rate
DSCAL	Digital Systems Computer Architecture Laboratory
DSU	Debug Support Unit
DTN	Delay Tolerant Network
DUT	Design Under Test
DVB	Digital Video Broadcasting
ECC	Error-Correcting Code
ECLSA	Erasure Coding Link Service Adapter
ECSS	European Cooperation for Space Standardization
EG	Euclidean Geometry
EOP	End of Packet
ESA	European Space Agency
EXIT	EXtrinsic Information Chart
FEC	Forward Error Correction
FF	Flip-Flop
F-IRA	Flexible Irregular Repeat Accumulate
FIFO	First In, First Out
FMC	FPGA Mezzanine Card
FPGA	Field-Programmable Gate Array
GF	Gallois Field
GPIO	General Purpose Input-Output
GUI	Graphical User Interface
HDR	High Data Rate
HP	High Performance

HPE	High Photon Efficiency
IARA	Improved Accumulate Repeat Accumulate
IDEMA	International Disk Drive Equipment and Materials Association
IEEE	Institute of Electrical and Electronics Engineers
IOAG	Interagency Operation Advisory Group
ION	Interplanetary Overlay Network
IP-core	Intellectual Property (core)
IRA	Irregular Repeat Accumulate
ISA	Instruction Set Architecture
ISI	Inter-Symbol Interference
ITAR	International Traffic in Arms Regulations
JTAG	Joint Test Action Group
L-U	Lower-Upper
LDPC	Low-Density Parity-Check (code)
LFSR	Linear-Feedback Shift Register
LLR	Log-Likelihood Ratio
LTP	Licklider Transmission Protocol
LUT	Look-Up Table
LW	Long-Word
MAP	Maximum A-Posteriori
MDS	Maximum Distance Separable
MIG	Memory Interface Generator
MISR	Multiple Input Shift Register
MLSD	Maximum Likelihood Sequence Decoder
MPSoC	Multiprocessor System on a Chip
MR	Magnetic Recording
MT/s	MegaTransfers per second
MTTF	Mean Time To Failure
NASA	National Aeronautics and Space Administration

NV	Non-Volatile
O3K	Optical On-Off Keying
OBC	On Board Data Computer
OSI	Open Systems Interconnection
PC	Personal Computer
PCIe	Peripheral Component Interconnect Express
PDU	Protocol Data Unit
PEG	Progressive Edge Growth
PISO	Parallel Input Serial Output
PL	Packet-Level, Programmable Logic
PLATO	PLAnetary Transits and Oscillations of stars
PNC	Physical-layer Network Coding
PR	Partial Response
PBRL	Protograph-based Raptor-like
QC	Quasi-Cyclic
RS	Reed-Solomon
RT	Radiation Tolerant
RTL	Register Transfer Level
R-U	Richardson-Urbanke
RAM	Random Access Memory
RCOP	Rate Compatible Optimized Codes
RHBD	Radiation-Hardened by Design
RISC	Reduced Instruction Set Computer
RMAP	Remote Memory Access Protocol
RTL	Register Transfer Level
RS	Reed-Solomon
SAVOIR	Space Avionics Open Interface Architecture
SDR	Software Defined Radio
SEB	Single Event Burnout

SEE	Single Event Effects
SERDES	Serializer/Deserializer
SET	Single Event Transient
SEU	Single Event Upset
SoC	System On a Chip
SODIMM	Small Outline Dual Inline Memory Module
SONOS	Silicon-Oxide-Nitride-Silicon
SRIO	Serial Rapid Input Output
SPA	Sum-Product Algorithm
SpFi	Space Fibre
SRAA	Shift Register Adder Accumulator
SRAM	Static RAM
SRIO	Serial RapidIO
SWaP-C	Size, Weight, Power and Cost
TID	Total Ionizing Dose
TM	Telemetry
TMR	Triple Module Redundancy
TNID	Total Non Ionizing Dose
VLSI	Very Large Scale Integration
VNO	Variable Node Oriented

BIBLIOGRAPHY

[1] Sixteen-Bit Computer Instruction Set Architecture. DoD Military Standard MIL-STD-1750, Defense Information Systems Agency (DISA), August 1980.

[2] AMBA AXI4-Stream protocol v1.0A. Specification, ARM, March 2010.

[3] SpaceWire – Remote Memory Access Protocol. ECSS Standard ECSS-E-ST-50-52C, European Cooperation for Space Standardization-ECSS, February 2010.

[4] LCE01C CCSDS (8160,7136) LDPC Encoder. Product specification, Small World Communications, March 2013.

[5] Digital Video Broadcasting (DVB) Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications Part 1: DVB-S2. ETSI European standard REN/JTC-DVB-341-1, ETSI, 2014.

[6] LDPC NASA Encoder/Decoder IP Core v2.0. Specification sheet, IPrium LLC, 2014.

[7] CCSDS (8160, 7136) LDPC Encoder and Decoder. Product brief, CREONIC GmbH, 2016.

[8] Space engineering Technology Readiness Level (TRL) guidelines. ECSS Handbok ECSS-E-HB-11A, European Cooperation for Space Standardization-ECSS, March 2017.

[9] CCSDS LDPC C2 code encoder/decoder. VHDL source code overview / IP core Overview COM-1811SOFT, COMBLOCK, 2019.

[10] SpaceFibre – Very high-speed serial link. ECSS Standard ECSS-E-ST-50-11C, European Cooperation for Space Standardization-ECSS, May 2019.

[11] SpaceWire – Links, nodes, routers and networks. ECSS Standard ECSS-E-ST-50-12C Rev.1, European Cooperation for Space Standardization-ECSS, May 2019.

[12] ESA IP Core Technical Requirements. ESA Requirements specification TEC-EDM/2010.61/KM, European Space Agency (ESA), May 2022.

[13] Rojina Adhikary, John N. Daigle, and Lei Cao. Dynamic Code Selection Method for Content Transfer in Deep-Space Network. *IEEE Transactions on Aerospace and Electronic Systems*, 56(1):456–474, February 2020.

[14] Alaa Aldin Al Hariri, Fabrice Monteiro, Loic Sieler, and Abbas Dandache. A high throughput configurable parallel encoder architecture for Quasi-Cyclic Low-Density Parity-Check Codes. In *2013 IEEE 19th International On-Line Testing Symposium (IOLTS)*, pages 163–166. IEEE, July 2013.

[15] Nicola Alessi, Carlo Caini, Tomaso De Cola, and Marco Raminella. Packet Layer Erasure Coding in Interplanetary Links: The LTP Erasure Coding Link Service Adapter. *IEEE Transactions on Aerospace and Electronic Systems*, 56(1):403–414, February 2020.

[16] Jan Andersson, Jiri Gaisler, and Roland Weigand. Next generation multipurpose microprocessor. In *DAta Systems In Aerospace 2010 (DASIA2010)*, 2010.

[17] K. Andrews, S. Dolinar, and J. Thorpe. Encoders for block-circulant LDPC codes. In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pages 2300–2304, Adelaide, SA, Australia, September 2005.

[18] Kenneth S. Andrews, Dariush Divsalar, Sam Dolinar, Jon Hamkins, Christopher R. Jones, and Fabrizio Pollara. The Development of Turbo and LDPC Codes for Deep-Space Applications. *Proceedings of the IEEE*, 95:2142–2156, November 2007.

[19] Lars Asplund. VUnit: a test framework for HDL v.4.4.0. https://vunit.github.io/, last accessed on 02/05/2023, 2020.

[20] Abhijit Athavale and Carl Christensen. *High-Speed Serial I/0 Made Simple*. Xilinx, 1.0 edition, 2005.

[21] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv. Optimal decoding of linear codes for minimizing symbol error rate (corresp.). *IEEE Transactions on Information Theory*, 20(2):284–287, March 1974.

[22] Christian R. Berger, Shengli Zhou, Yonggang Wen, Peter Willett, and Krishna Pattipati. Optimizing joint erasure- and error-correction coding for wireless packet transmissions. *IEEE Transactions on Wireless Communications*, 7(11):4586–4595, November 2008.

[23] C. Berrou, A. Glavieux, and P. Thitimajshima. Near shannon limit error-correcting coding and decoding: Turbo-codes. 1. In *Proceedings of ICC '93 - IEEE International Conference on Communications*, volume 2, pages 1064–1070, Geneva, Switzerland, 1993.

[24] F. Bezerra, D. Dangla, F. Manni, J. Mekki, D. Standarovski, R. G. Alia, M. Brugger, and S. Danzeca. Evaluation of an Alternative Low Cost Approach for SEE Assessment of a SoC. In *2017 17th European Conference on Radiation and Its Effects on Components and Systems (RADECS)*, pages 1–5, Boston, MA, USA, July 2017.

[25] Frey J. Brendan. *Graphical Models for Machine Learning and Digital Communication*. The MIT Press, July 1998.

[26] Stephan Ten Brink. Convergence behavior of iteratively decoded parallel concatenated codes. *IEEE Transactions on Communications*, 49(10):1727–1737, October 2001.

[27] David Burshtein and Gadi Miller. Asymptotic enumeration methods for analyzing LDPC codes. *IEEE Transactions on Information Theory*, 50(6):1115–1131, June 2004.

[28] C. Caini. Delay-tolerant networks (dtns) for satellite communications. *Advances in Delay-Tolerant Networks (DTNs): Architecture and Enhanced Performance*, pages 23–46, January 2021.

[29] Gian Paolo Calzolari, Marco Chiani, Franco Chiaraluce, Roberto Garello, and Enrico Paolini. Channel coding for future space missions: New requirements and trends. *Proceedings of the IEEE*, 95(11):2157–2170, 2007.

[30] Overview of Space Communication Protocols. Informational report CCSDS 130.0-G.3, CCSDS, July 2014.

[31] TM synchronization and channel coding - summary of concept and rationale. Informational report CCSDS 130.1-G-3, CCSDS, June 2020.

[32] TM Synchronization and Channel Coding. Recommended standard CCSDS 131.0-B-3, CCSDS, September 2017.

[33] Erasure Correcting Codes for Use in Near-Earth and Deep Space Communications. Experimental specification CCSDS 131.5-O-1, CCSDS, November 2014.

[34] 1064 nm Optical High Data Rate (HDR) Communication. Experimental specification CCSDS 141.11-O-1, CCSDS, December 2018.

[35] Optical communications coding and synchronization. Recommended standard CCSDS 142.0-B-1, CCSDS, August 2019.

[36] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-tolerant networking architecture. RFC 4838, RFC Editor, 2007. `http://www.rfc-editor.org/rfc/rfc4838.txt`.

[37] Panagiotis Chatziantoniou, Antonis Tsigkanos, Dimitris Theodoropoulos, Nektarios Kranitis, and Antonis Paschalis. An Efficient Architecture and High-Throughput Implementation of CCSDS-123.0-B-2 Hybrid Entropy Coder Targeting Space-Grade SRAM FPGA Technology. *IEEE Transactions on Aerospace and Electronic Systems*, 58(6):5470–5482, December 2022.

[38] Dongying Chen, Pingping Chen, and Yi Fang. Low-Complexity High-Performance Low-Density Parity-Check Encoder Design for China Digital Radio Standard. *IEEE Access*, 5:20880–20886, July 2017.

[39] Pingping Chen, Lingjun Kong, Yi Fang, and Lin Wang. The Design of Protograph LDPC Codes for 2-D Magnetic Recording Channels. *IEEE Transactions on Magnetics*, 51(11), November 2015.

[40] Pingping Chen, Zhaopeng Xie, Yi Fang, Zhifeng Chen, Shahid Mumtaz, and Joel J.P.C. Rodrigues. Physical-Layer Network Coding: An Efficient Technique for Wireless Communications. *IEEE Network*, pages 1–7, 2019.

[41] Chia-Yu Lin, Chih-Chun Wei, and Mong-Kai Ku. Efficient encoding for dual-diagonal structured LDPC codes based on parity bit prediction and correction. In *APCCAS 2008 - 2008 IEEE Asia Pacific Conference on Circuits and Systems*, pages 1648–1651, Macao, China, November 2008.

[42] N. L. Clarke, B. V. Ghita, and S. M. Furnell. Delay-tolerant networks (DTNs) for deep-space communications. *Advances in Delay-Tolerant Networks (DTNs): Architecture and Enhanced Performance*, pages 47–58, January 2021.

[43] Cobham Gaisler AB. GRLIB IP library processor. https://www.gaisler.com/index.php/products/ipcores/soclibrary, last accessed on 02//05/2023.

[44] Cobham Gaisler AB. LEON3 processor. https://www.gaisler.com/index.php/products/processors/leon3, last accessed on 02//05/2023.

[45] Cobham Gaisler AB. LEON5 processor. https://www.gaisler.com/index.php/products/processors/leon5, last accessed on 02//05/2023.

[46] Cobham Gaisler AB. NOEL-V Processor. https://www.gaisler.com/index.php/products/processors/noel-v, last accessed on 02//05/2023.

[47] A.E. Cohen and K.K. Parhi. A Low-Complexity Hybrid LDPC Code Encoder for IEEE 802.3an (10GBase-T) Ethernet. *IEEE Transactions on Signal Processing*, 57(10):4085–4094, October 2009.

[48] CORDIS. De-RISC: Dependable Real-time Infrastructure for Safety-critical Computer. https://cordis.europa.eu/project/id/869945, last accesed on 26/06/2022.

[49] Thomas A. Courtade and Richard D. Wesel. Optimal allocation of redundancy between packet-level erasure coding and physical-layer channel coding in fading channels. *IEEE Transactions on Communications*, 59(8):2101–2109, August 2011.

[50] Z. Cui, Z. Wang, and X. Zhang. Reduced-complexity column-layered decoding and implementation for ldpc codes. *IET Communications*, 5:2177–2186(9), October 2011.

[51] Mathieu Cunche, Jonathan Detchart, Jérome Lacan, Vincent Roca, Kévin Chaumont, Julien Laboure, Christoph Neumann, Alexandre Soro, and Valentin Savin. OpenFEC.org project. http://openfec.org/, last accessed on 21/04/23.

[52] Faramaz Davarian, Alessandra Babuscia, John Baker, Richard Hodges, Damon Landau, Chi-wung Lau, Norman Lay, Matt Angert, and Vanessa Kuroda. Improving Small Satellite Communications in Deep Space—A Review of the Existing Systems and Technologies With Recommendations for Improvement. Part I: Direct to Earth Links and SmallSat Telecommunications Equipment. *IEEE Aerospace and Electronic Systems Magazine*, 35(7):8–25, July 2020.

[53] Stefano Di Mascio, Alessandra Menicucci, Eberhard Gill, Gianluca Furano, and Claudio Monteleone. Leveraging the openness and modularity of RISC-V in space. *Journal of Aerospace Information Systems*, 16(11):454–472, January 2020.

[54] Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 1948.

[55] R. Ecoffet. Overview of in-orbit radiation induced spacecraft anomalies. *IEEE Transactions on Nuclear Science*, 60(3):1791–1815, June 2013.

[56] Bernard L. Edwards, Robert Daddato, Klaus-Juergen Schulz, Randall Alliss, Jon Hamkins, Dirk Giggenbach, Bryan Robinson, and Lena Braatz. An Update on the CCSDS Optical Communications Working Group Interoperability Standards. In *2019 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, pages 1–9, Portland, OR, USA, October 2019.

[57] Yi Fang, Guoan Bi, Yong Liang Guan, and Francis C.M. Lau. A Survey on Protograph LDPC Codes and Their Applications. *IEEE Communications Surveys and Tutorials*, 17(4):1989–2016, October 2015.

[58] Yi Fang, Guofa Cai, Zhaojie Yang, Pingping Chen, and Guojun Han. Performance of protograph LDPC codes over ergodic Nakagami fading channels. In *2017 17th International Symposium on Communications and Information Technologies, ISCIT 2017*, pages 1–5, Cairns, QLD, Australia, July 2017.

[59] Yi Fang, Pingping Chen, Guofa Cai, Francis C.M. Lau, Soung Chang Liew, and Guojun Han. Outage-Limit-Approaching Channel Coding for Future Wireless Communications: Root-Protograph Low-Density Parity-Check Codes. *IEEE Vehicular Technology Magazine*, 14(2):85–93, June 2019.

[60] Yi Fang, Pingping Chen, Lin Wang, and Francis C.M. Lau. Design of protograph LDPC codes for partial response channels. *IEEE Transactions on Communications*, 60(10):2809–2819, 2012.

[61] Yi Fang, Guojun Han, Guofa Cai, Francis C.M. Lau, Pingping Chen, and Yong Liang Guan. Design Guidelines of Low-Density Parity-Check Codes for Magnetic Recording Systems. *IEEE Communications Surveys and Tutorials*, 20(2):1574–1606, April 2018.

[62] Fares Fourati and Mohamed-Slim Alouini. Artificial intelligence for satellite communication: A review. *Intelligent and Converged Networks*, 2(3):213–243, 2021.

[63] Gianluca Furano and Alessandra Menicucci. *Roadmap for On-Board Processing and Data Handling Systems in Space*, pages 253–281. Springer International Publishing, Cham, 2018.

[64] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, January 1962.

[65] K. S. Geethu and A. V. Babu. Performance analysis of erasure coding based data transfer in Underwater Acoustic Sensor Networks. In *2015 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2015*, pages 2145–2151, Kochi, India, sep 2015.

[66] Jeffrey S. George. An overview of radiation effects in electronics. *AIP Conference Proceedings*, 2160(1):060002, oct 2019.

[67] Gianluca Giuffrida, Luca Fanucci, Gabriele Meoni, Matej Batič, Léonie Buckley, Aubrey Dunne, Chris van Dijk, Marco Esposito, John Hefele, Nathan Vercruyssen, Gianluca Furano, Massimiliano Pastena, and Josef Aschbacher. The Φ-Sat-1 Mission: The First On-Board Deep Neural Network Demonstrator for Satellite Earth Observation. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–14, 2022.

[68] Matthias Göbel, Ahmed Elhossini, Chi Ching Chi, Mauricio Alvarez-Mesa, and Ben Juurlink. A quantitative analysis of the memory architecture of FPGA-SoCs. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 10216 LNCS, pages 241–252. Springer Verlag, 2017.

[69] Marco Gomes, Gabriel Falcao, Alexandre Sengo, Vitor Ferreira, Vitor Silva, and Miguel Falcao. High throughput encoder architecture for DVB-S2 LDPC-IRA codes. In *2007 Internatonal Conference on Microelectronics*, pages 271–274, Cairo, Egypt, December 2007. IEEE.

[70] Albert Guillén i Fàbregas. Coding in the block-erasure channel. *IEEE Transactions on Information Theory*, 52(11):5116–5121, November 2006.

[71] Haibin Zhang, Jia Zhu, Huifeng Shi, and Dawei Wang. Layered Approx-Regular LDPC: Code Construction and Encoder/Decoder Design. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 55(2):572–585, March 2008.

[72] Yang Han and William E. Ryan. Packet-LDPC codes for tape drives. *IEEE Transactions on Magnetics*, 41(4):1340–1347, April 2005.

[73] Hao Zhong and Tong Zhang. Block-LDPC: a practical LDPC coding system design approach. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 52(4):766–775, April 2005.

[74] Alaa Aldin Al Hariri, Fabrice Monteiro, Loic Sieler, and Abbas Dandache. Configurable and high-throughput architectures for Quasi-cyclic low-density parity-check codes. In *2014 21st IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 790–793, Marseille, France, December 2014. doi: 10.1109/ICECS.2014.7050104.

[75] Alaa Hassan, M.I. Dessouky, Atef Abouelazm, and Mona Shokair. Evaluation of Complexity Versus Performance for Turbo Code and LDPC Under Different Code Rates. In *SPACOMM 2012 : The Fourth International Conference on Advances in Satellite and Space Communications*, January 2012.

[76] David M. Hiemstra and Valeri Kirischian. Single event upset characterization of the Zynq-7000 ARM® Cortex™-A9 processor unit using proton irradiation. *IEEE Radiation Effects Data Workshop*, November 2015.

[77] Junyang Hu and Kangming Jiang. The improved LU-based decomposition algorithm for sparse matrix of LDPC code. In Zhu E. Sambath S., editor, *Frontiers in Computer Education. Advances in Intelligent and Soft Computing*, volume 133, pages 867–874. Springer, Berlin, Heidelberg, 2012.

[78] Xiao Yu Hu, Evangelos Eleftheriou, and Dieter M. Arnold. Regular and irregular progressive edge-growth tanner graphs. *IEEE Transactions on Information Theory*, 51(1):386–398, January 2005.

[79] Qiang Huang and Jin Jiang. An overview of radiation effects on electronic devices under severe accident conditions in NPPs, rad-hardened design techniques and simulation tools. *Progress in Nuclear Energy*, 114:105–120, 2019.

[80] Jia-ning Su, Hou Iang, Ke iu, Iao yang eng, Ao Min, and Ao Min. An Efficient Low Complexity LDPC Encoder Based On Factorization With Pivoting. In *2005 6th International Conference on ASIC*, volume 1, pages 168–171, 2005.

[81] Yuichi Kaji. Encoding LDPC Codes Using the Triangular Factorization. *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, E89A(10), October 2006.

[82] Ogun O. Kibar, Prashanth Mohan, Paolo Rech, and Ken Mai. Evaluating the Impact of Repetition, Redundancy, Scrubbing, and Partitioning on 28-nm FPGA Reliability Through Neutron Testing. *IEEE Transactions on Nuclear Science*, 66(1):248–254, 2019.

[83] Sunitha Kopparthi and Don M. Gruenbacher. Implementation of a Flexible Encoder for Structured Low-Density Parity-Check Codes. In *2007 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing*, pages 438–441, August 2007.

[84] Dong-U Lee, Wayne Luk, Connie Wang, Christopher Jones, Michael Smith, and John Villasenor. A Flexible Hardware Encoder for Low-Density Parity-Check Codes. In *12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 101–111, 2004.

[85] George Lentaris, Konstantinos Maragos, Ioannis Stratakos, Lazaros Papadopoulos, Odysseas Papanikolaou, Dimitrios Soudris, Manolis Lourakis, Xenophon Zabulis, David Gonzalez-Arjona, and Gianluca Furano. High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation. *Journal of Aerospace Information Systems*, 15(4):178–192, 2018.

[86] Vasileios Leon, Ioannis Stamoulias, George Lentaris, Dimitrios Soudris, David Gonzalez-Arjona, Ruben Domingo, David Merodio Codinachs, and Isabelle Conway. Development and Testing on the European Space-Grade BRAVE FPGAs: Evaluation of NG-Large Using High-Performance DSP Benchmarks. *IEEE Access*, 9:131877–131892, 2021.

[87] Qing Li, Shangguang Wang, Xiao Ma, Qibo Sun, Houpeng Wang, Suzhi Cao, and Fangchun Yang. Service Coverage for Satellite Edge Computing. *IEEE Internet of Things Journal*, 9(1):695–705, 2022.

[88] Lixin Liang, Huan He, Jian Zhao, Chengjian Liu, Qiuming Luo, and Xiaowen Chu. An erasure-coded storage system for edge computing. *IEEE Access*, 8:96271–96283, 2020.

[89] Gianluigi Liva, Paola Pulini, and Marco Chiani. On-line construction of irregular repeat accumulate codes for packet erasure channels. *IEEE Transactions on Wireless Communications*, 12(2):680–689, 2013.

[90] Ahmed Mahdi and Vassilis Paliouras. A Low Complexity-High Throughput QC-LDPC Encoder. *IEEE Transactions on Signal Processing*, 62(10):2696–2708, May 2014.

[91] Matthew J. Marinella. Radiation Effects in Advanced and Emerging Nonvolatile Memories. *IEEE Transactions on Nuclear Science*, 68(5):546–572, 2021.

[92] Pierre-Philippe Mathieu, Maurice Borgeaud, Yves-Louis Desnos, Michael Rast, Carsten Brockmann, Linda See, Ravi Kapur, Miguel Mahecha, Ursula Benz, and Steffen Fritz. The ESA's Earth Observation Open Science Program [Space Agencies]. *IEEE Geoscience and Remote Sensing Magazine*, 5(2):86–96, 2017.

[93] Ferdaouss Mattoussi, Matthieu Crussiere, Jean Francois Helard, and Gheorghe Zaharia. Analysis of Coding Strategies Within File Delivery Protocol Framework for HbbTV Based Push-VoD Services over DVB Networks. *IEEE Access*, 7:15489–15508, 2019.

[94] L.H. Miles, J.W. Gambles, G.K. Maki, W.E. Ryan, and S.R. Whitaker. An 860-Mb/s (8158,7136) Low-Density Parity-Check Encoder. *IEEE Journal of Solid-State Circuits*, 41(8):1686–1691, August 2006.

[95] T. Miyauchi, K. Yamamoto, T. Yokokawa, M. Kan, Y. Mizutani, and M. Hattori. High-performance programmable siso decoder vlsi implementation for decoding turbo codes. In *GLOBECOM'01. IEEE Global Telecommunications Conference (Cat. No.01CH37270)*, volume 1, pages 305–309 vol.1, San Antonio, TX, USA, 2001.

[96] NASA. Interplanetary Overlay Network (ION) software distribution (4.0.2). https://sourceforge.net/projects/ion-dtn/.

[97] Nelson Alves Ferreira Neto, Joaquim Ranyere S. de Oliveira, Wagner Luiz A. de Oliveira, and Joao Carlos N. Bittencourt. VLSI architecture design and implementation of a LDPC encoder for the IEEE 802.22 WRAN standard. In *2015 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 71–76, Salvador, Brazil, September 2015. IEEE.

[98] Thuy Van Nguyen, Aria Nosratinia, and Dariush Divsalar. Rate-compatible protograph-based LDPC codes for inter-symbol interference channels. *IEEE Communications Letters*, 17(8):1632–1635, 2013.

[99] Tram Thi Bao Nguyen, Tuy Nguyen Tan, and Hanho Lee. Efficient QC-LDPC Encoder for 5G New Radio. *Electronics*, 8(6):668, June 2019.

[100] Daniel L Oltrogge and Kyle Leveque. An Evaluation of CubeSat Orbital Decay. In *25th Annual AIAA/USU Conference on Small Satellites*, 2011.

[101] Pouya Ostovari and Jie Wu. Reliable broadcast with joint forward error correction and erasure codes in wireless communication networks. In *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*, pages 324–332, Dallas, TX, USA, 2015.

[102] Xiaohan Pan, Yafeng Zhan, Peng Wan, and Jianhua Lu. Review of channel models for deep space communications. *Science China Information Sciences*, 61:1–12, 3 2018.

[103] Enrico Paolini, Gianluigi Liva, Balazs Matuz, and Marco Chiani. Maximum Likelihood Erasure Decoding of LDPC Codes: Pivoting Algorithms and Code Design. *IEEE Transactions on Communications*, 60(11):3209–3220, November 2012.

[104] Steve Parkes. *SpaceWire User's Guide*. Star-Dundee, 2012.

[105] Antonis Paschalis, Panagiotis Chatziantoniou, Dimitris Theodoropoulos, Antonis Tsigkanos, and Nektarios Kranitis. High-Performance Hardware Accelerators for Next Generation On-Board Data Processing. In *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)*, pages 1–4, Patras, Greece, 2022.

[106] Jesús M. Pérez and Víctor Fernández. 3GPP2/802.20 RC/QC-LDPC encoding. In *2010 European Wireless Conference (EW)*, Lucca, Italy, 2010.

[107] *PLATO DPS: State of the art on-board data processing for Europe's next planet-hunter*. Zenodo, June 2021.

[108] I. S. Reed and G. Solomon. Polynomial Codes Over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8:300–304, June 1960.

[109] Weiji Ren and Hao Liu. The Design and Implementation of High-Speed Codec Based on FPGA. In *2018 10th International Conference on Communication Software and Networks, ICCSN*, pages 427–432, October 2018.

[110] T.J. Richardson and R.L. Urbanke. Efficient encoding of low-density parity-check codes. *IEEE Transactions on Information Theory*, 47(2):638–656, 2001.

[111] Sebastian Sabogal, Alan George, and Gary Crum. ReCoN: A Reconfigurable CNN Acceleration Framework for Hybrid Semantic Segmentation on Hybrid SoCs for Space Applications. In *2019 IEEE Space Computing Conference (SCC)*, pages 41–52, 2019.

[112] P.H. Siegel, D. Divsalar, E. Eleftheriou, J. Hagenauer, D. Rowitch, and W.H. Tranter. Guest editorial the turbo principle: from theory to practice. *IEEE Journal on Selected Areas in Communications*, 19(5):793–799, 2001.

[113] Valery Sklyarov, Iouliia Skliarova, Joao Silva, and Alexander Sudnitson. Analysis and comparison of attainable hardware acceleration in all programmable systems-on-chip. In *Proceedings - 18th Euromicro Conference on Digital System Design, DSD*, pages 345–352, October 2015.

[114] Ted Speers. PolarFire SONOS Technology. https://www.microsemi.com/blog/2018/04/10/polarfire-sonos-technology/, April 2018.

[115] Antonio Sánchez, Yubal Barrios, Lucana Santos, and Roberto Sarmiento. Evaluation of TMR effectiveness for soft error mitigation in SHyLoC compression IP core implemented on Zynq SoC under heavy ion radiation. In *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–4, Noordwijk, Netherlands, 2019.

[116] Bashar Tahir, Stefan Schwarz, and Markus Rupp. BER comparison between Convolutional, Turbo, LDPC, and Polar codes. In *2017 24th International Conference on Telecommunications (ICT)*, pages 1–7, Limassol, Cyprus, 2017.

[117] Lucas Antunes Tambara, Fernanda Lima Kastensmidt, Nilberto H. Medina, Nemitala Added, Vitor A. P. Aguiar, Fernando Aguirre, Eduardo L. A. Macchione, and Marcilei A. G. Silveira. Heavy Ions Induced Single Event Upsets Testing of the 28 nm Xilinx Zynq-7000 All Programmable SoC. In *2015 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–6, Boston, MA, USA, 2015.

[118] Nianqi Tang and Yun Lin. Fast Encoding and Decoding Algorithms for Arbitrary F2m. *IEEE Communications Letters*, 24(4):716–719, April 2020.

[119] R. Michael Tanner. A Recursive Approach to Low Complexity Codes. *IEEE Transactions on Information Theory*, 27(5):533–547, 1981.

[120] S. ten Brink. Convergence of iterative decoding. *Electronics Letters*, 35(13):1117–1118, 1999.

[121] Dimitris Theodoropoulos, Nektarios Kranitis, and Antonios Paschalis. An efficient LDPC encoder architecture for space applications. In *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 149–154, Sant Feliu de Guixols, Spain, July 2016.

[122] Dimitris Theodoropoulos, Nektarios Kranitis, Antonis Tsigkanos, Elias Machairas, and Antonios Paschalis. Efficient Hardware Architectures and Implementations of Packet-Level Erasure Coding Schemes for High Data Rate Reliable Satellite Communications. *IEEE Transactions on Aerospace and Electronic Systems*, 58(3):2269–2280, 2022.

[123] Dimitris Theodoropoulos, Nektarios Kranitis, Antonis Tsigkanos, and Antonios Paschalis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(5):1118–1127, March 2020.

[124] Dimitris Theodoropoulos, Nektarios Kranitis, Antonis Tsigkanos, and Antonios Paschalis. Efficient LDPC Encoder Designs for Magnetic Recording Media. In *2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pages 1–6, Frascati, Italy, 2020.

[125] Tao Tian, Christopher R. Jones, John D. Villasenor, and Richard D. Wesel. Selective avoidance of cycles in irregular LDPC code construction. *IEEE Transactions on Communications*, 52(8):1242–1247, August 2004.

[126] Antonis Tsigkanos, Nektarios Kranitis, DImitris Theodoropoulos, and Antonios Paschalis. High-Performance COTS FPGA SoC for Parallel Hyperspectral Image Compression with CCSDS-123.0-B-1. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 28(11):2397–2409, November 2020.

[127] Georgios Tzimpragos, Christoforos Kachris, Dimitrios Soudris, and Ioannis Tomkos. A low-complexity implementation of QC-LDPC encoder in reconfigurable logic. In *2013 23rd International Conference on Field programmable Logic and Applications*, pages 1–4, Porto, Portugal, September 2013.

[128] Thuy Van Nguyen, Aria Nosratinia, and Dariush Divsalar. Protograph-based LDPC codes for partial response channels. In *2012 IEEE International Conference on Communications*, pages 2166–2170, Ottawa, ON, Canada, 2012.

[129] Jason Vidmar, Pierre Maillard, Troy Jones, Minal Sawant, Giulio Gambardella, and Nicholas Fraser. Space DPU: Constructing a Radiation-Tolerant, FPGA-based Platform for Deep Learning Acceleration on Space Payloads. In *2nd European Workshop on On-Board Data Processing (OBDP 2021)*, June 2021.

[130] Vasileios Vlagkoulis, Aitzan Sari, John Vrachnis, Georgios Antonopoulos, Nikolaos Segkos, Mihalis Psarakis, Antonios Tavoularis, Gianluca Furano, Cesar Boatella Polo, Christian Poivey, Veronique Ferlet-Cavrois, Maria Kastriotou, Pablo Fernandez Martinez, Ruben Garcia Alia, Kay-Obbe Voss, and Christoph Schuy. Single Event Effects Characterization of the Programmable Logic of Xilinx Zynq-7000 FPGA Using Very/Ultra High-Energy Heavy Ions. *IEEE Transactions on Nuclear Science*, 68(1):36–45, 2021.

[131] Fei Wang, Peng Zhang, Xin Wan, and Jin Liu. Design of a multi-rate quasi-cyclic low-density parity-check encoder based on pipelined rotate-left-accumulator circuits. In *Proceedings - 2014 7th International Congress on Image and Signal Processing, CISP 2014*, pages 1105–1109, Dalian, China, January 2014.

[132] Peng Wang and Yong-en Chen. Low-Complexity Real-Time LDPC Encoder Design for CMMB. In *2008 International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pages 1209–1212, Harbin, China, August 2008.

[133] Xiumin Wang, Tingting Ge, Jun Li, Chen Su, and Fangfei Hong. Efficient Multi-rate Encoder of QC-LDPC Codes Based on FPGA for WIMAX Standard. *Chinese Journal of Electronics*, 26(2):250–255, March 2017.

[134] Xiangran Sun and Dongxin Shi. Design and optimization of LDPC encoder based on LU decomposition with simulated annealing. In *2011 International Conference on Computer Science and Service System (CSSS)*, pages 2181–2184, Nanjing, China, June 2011.

[135] Hemesh Yasotharan and Anthony Chan Carusone. A flexible hardware encoder for systematic low-density parity-check codes. In *2009 52nd IEEE International Midwest Symposium on Circuits and Systems*, pages 54–57, Cancun, Mexico, August 2009. IEEE.

[136] Shao-Wei Yen, Shiang-Yu Hung, Chih-Lung Chen, Hsie-Chia Chang, Shyh-Jye Jou, and Chen-Yi Lee. A 5.79-Gb/s Energy-Efficient Multirate LDPC Codec Chip for IEEE 802.15.3c Applications. *IEEE Journal of Solid-State Circuits*, 47(9):2246–2257, September 2012.

[137] Hang Yin, Weitao Du, and Nanhao Zhu. Design of improved LDPC encoder for CMMB based on SIMD architecture. In *2013 IEEE 3rd International Conference on Information Science and Technology (ICIST)*, pages 1292–1295, Yangzhou, China, 2013.

[138] Yongmin Jung, Chulho Chung, Jaeseok Kim, and Yunho Jung. 7.7Gbps encoder design for IEEE 802.11n/ac QC-LDPC codes. In *2012 International SoC Design Conference (ISOCC)*, pages 215–218, Jeju Island, November 2012.

[139] Shuo Yu, Changyin Liu, Peng Zhang, and Lanxiang Jiang. Efficient encoding of QC-LDPC codes with multiple-diagonal parity-check structure. *Electronics Letters*, 50(4):320–321, February 2014.

[140] Qiuyu Wu Zhaohui Wang, Xin Hao, Changxing Lin. An Efficient Hardware LDPC Encoder Based on Partial Parallel Structure for CCSDS. In *2018 IEEE 18th International Conference on Communication Technology (ICCT)*, pages 136–139, Chongqing, 2018.

[141] Zhiyong He, S. Roy, and P. Fortier. Encoder architecture with throughput over 10 Gbit/sec for quasi-cyclic LDPC codes. In *2006 IEEE International Symposium on Circuits and Systems*, page 4, Kos, Greece, 2006.

[142] Zongwang Li, Lei Chen, Lingqi Zeng, S. Lin, and W.H. Fong. Efficient encoding of quasi-cyclic low-density parity-check codes. *IEEE Transactions on Communications*, 54(1):71–81, January 2006.